UNIVERSITY OF CALIFORNIA
Santa Barbara

Novel Image Data-Hiding Methodologies for Robust and Secure Steganography with Extensions to Image Forensics

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

by

Anindya Sarkar

Committee in Charge:

Professor B. S. Manjunath, Chair

Professor Upamanyu Madhow

Professor Shivkumar Chandrasekaran

Professor Jerry Gibson

Professor Keith Clarke

Dr Kaushal Solanki

March 2010

The Dissertation of
Anindya Sarkar is approved:

Professor Upamanyu Madhow

Professor Shivkumar Chandrasekaran

Professor Jerry Gibson

Professor Keith Clarke

Dr Kaushal Solanki

Professor B. S. Manjunath, Committee Chairperson

December 2009

Novel Image Data-Hiding Methodologies for Robust and Secure Steganography with Extensions to Image Forensics

by

Anindya Sarkar

Dedicated

to

my mother, father and sister

Acknowledgments

For me, personally, a very interesting part of reading someone else's dissertation is the Acknowledgments part. A PhD program is pretty long in general - 4 to 6 years. A person spends a good chunk of this time focussing on a directed theme of research. However, to repeat a cliche, one should enjoy the journey besides aspiring towards the end goal. The journey would indeed be very difficult without the right guidance and company.

A PhD program is a highly loaded job in the true sense of the word. One has to find newer and interesting problems and keep on producing better than state-of-the-art results using novel methodologies. On and off, one has to do a bit of proposal writing to help the Professor and other students in the lab who may potentially be funded through it. My advisor, Prof Manjunath, has been a calming influence on me and has helped me retain my mental composure and a tension-free attitude in general. He has always been encouraging and supporting without being very pushy. He feels that given enough freedom to pursue the problems one feels strongly about, one is bound to come good sooner rather than later. I remember my first few months at UCSB when I used to immerse myself in course-work and did not have enough research output. I thank Prof Manjunath for not pressurizing me but believing in me in the initial difficult stages. Over due course of time, I feel I have learnt some basic truisms of life from him -

work is never going to get less and it is upto an individual to balance demanding schedules instead of just complaining about them. I admire how he manages so many students and so many projects. That motivated me to even handle interns (in fact, two in one year) in the busiest year of my student career so far. The other big lesson from him is to focus on the bigger picture - I used to (and maybe, still sometimes) lose sight of the main problem by focussing too much on intricate details.

I have also had many fruitful discussions with Prof Madhow. I really admire his work ethic and his ability to track down inconsistencies in logic and in writing. I take this opportunity to apologize to him for the countless times I must have infuriated him with my "not-so-compact" style of writing. He has been very forthcoming in his criticisms and I feel (or rather, I hope) that my writing style has really improved under his influence. I also thank the other members of my committee, Prof Shiv Chandrashekhar, Prof Jerry Gibson, and Prof Keith Clarke for their support and encouragement. Finally, I thank Kaushal Solanki, the person who introduced me to data-hiding, and also helped me out during the early hiccup stages of my data-hiding career.

Work-wise, I feel that group activity is often better as you have more people to bounce your ideas off and two heads are better than one. I started out my research by doing some steganography work with Kaushal Solanki and Ken Sullivan, both

PhDs from my lab. Midway, I got interested in other problems like video duplicate detection where I really enjoyed working with Vishwakarma Singh and Pratim Ghosh. Towards the latter stage of my PhD, I ventured into forensics where I worked with Lakshmanan Nataraj, and feel I learnt a lot from that interaction. In 2009, I had an undergraduate intern, Erick Spaan, and I felt my teaching skills were enhanced from that experience. Finally, I thank Miriam Redi, a Masters student from Italy whom I guided during her Masters project, and it was a really nice experience. I felt my forensics knowledge was improved from my interactions with her and I learned to be patient while dealing with students.

Apart from my technical collaborators, it has been a nice four years spent at the Vision Research Lab (VRL), UCSB. I really enjoyed the guidance I received from Marco Zuliani in my first year in the lab. Over the next few years, it was great to have friends like Mike, Nhat, Thomas, Dmitry, Jim, Zefeng and Diana in VRL. In the Bio-Image Lab, it was fun to discuss research and also hang out with Pratim, Emre, Vignesh, Jiejun, Aruna, Swapna and Luca. The good thing about being a part of a big lab is that there are a large number of people you can go to with your questions - often it takes less time to ask someone and figure something out than doing a Google search and reading through the online solution. I also thank Sang, Vincent and Jim - my screening exam partners.

As I said before, 4 years is a long time in life and Santa Barbara is really a nice place to live in. Coming from India, the weather was near-perfect and sandwiched between the mountains and the ocean, I feel really lucky to have been in the lap of nature. I shall definitely miss it a lot once I leave Santa Barbara.

I consider myself fortunate to have been associated with a very close group of friends during my stay at UCSB. Most of them were my seniors or Masters' students who joined with me and many of them have since moved on to greener pastures. While naming all of them may be difficult, I take this opportunity to thank Rajesh, Susmit, Karma, Harsha, Vinod, Rajdeep, Ankurji, Sharadh, Sharath, Kamal, Mohit, Vignesh, Sayan, Sudipto, Pratim, Footpath, Swaroop, KP, Jassi, Sumit, and Arnab - apologies to the many people I missed. There have been many memorable events, trips, games (soccer/tennis/ping pong/cricket), and those innumerable get-togethers which often helped me to pick myself up and look forward to every new day and new challenges.

Finally, to save the best for the last, I would like to thank my family for bearing with me, for being patient and acknowledging the need to let go at some point of life, for being very understanding of my mood swings, and for being my support system throughout the last four years. My parents have given me more than enough and whatever little I have achieved in the academic field is due to their constant support and blessings. My sister inspired me to study harder when

I was growing up and I still look upto her as a role model of work-ethics and dedication. Lastly, I am sure my four-year old niece offered me many pearls of wisdom and invaluable research inputs whenever we spoke on the phone - its only my fault I often fail to decode her speech.

Curriculum Vitæ

Anindya Sarkar

Education

December 2009	Doctor of Philosophy
	Department of Electrical and Computer Engineering
	University of California, Santa Barbara
July 2005	Master of Engineering (Signal Processing)
	Department of Electrical Engineering
	Indian Institute of Science, Bangalore
June 2003	Bachelor of Engineering
	Department of Electronics and Telecomm. Engineering
	Jadavpur University, Kolkata, India

Fields of Study

Data hiding, watermarking, steganography, steganalysis, image forensics, tamper detection

Publications

(1) A. Sarkar, U. Madhow, S. Chandrasekaran and B. S. Manjunath, "Adaptive MPEG-2 Video Data Hiding Scheme", *Proc. of SPIE - Security, Steganography, and Watermarking of Multimedia Contents (IX)*, January 2007

(2) A. Sarkar, K. Solanki, U. Madhow, S. Chandrasekaran and B. S. Manjunath, "Secure Steganography: Statistical Restoration of the Second Order Dependencies for Improved Security", *IEEE International Conference of Acoustics, Speech and Signal Processing* (ICASSP-07), April 2007 (**shortlisted for the Best Student Paper Award**) (Chap. 2)

(3) K. Solanki, A. Sarkar and B. S. Manjunath, "YASS: Yet Another Steganographic Scheme that Resists Blind Steganalysis", *9th International Workshop on Information Hiding*, Saint Malo, Brittany, France, June 2007 (Chap. 3)

(4) A. Sarkar and B. S. Manjunath, "Estimating Steganographic Capacity for Odd-Even Based Embedding and its Use in Individual Compensation", *IEEE International Conference on Image Processing (ICIP07)*, San Antonio, TX, September 2007 (Chap. 2)

(5) A. Sarkar, K. Solanki and B. S. Manjunath, "Further study on YASS: Steganography based on Randomized Embedding to Resist Blind Steganalysis", *Proc.*

of SPIE - Security, Steganography, and Watermarking of Multimedia Contents (X), January 2008 (Chap. 3)

(6) A. Sarkar, K. Sullivan and B. S. Manjunath, "Steganographic Capacity Estimation for the Statistical Restoration Framework", *Proc. of SPIE - Security, Steganography, and Watermarking of Multimedia Contents (X)*, January 2008 (Chap. 2)

(7) A. Sarkar, P. Ghosh, E. Moxley and B. S. Manjunath, "Video Fingerprinting: Features for Duplicate and Similar Video Detection and Query-based Video Retrieval", *Proc. of SPIE - Multimedia Content Access: Algorithms and Systems II*, January 2008

(8) J. Kleban, A. Sarkar, E. Moxley, S. Mangiat, S. Joshi, T. Kuo and B.S. Manjunath, "Feature Fusion and Redundancy Pruning for Rush Video Summarization", *Proceedings of the International Workshop on TRECVID video summarization*, Augsburg, Bavaria, Germany, September 2007, ACM Press.

(9) A. Sarkar, L. Nataraj, B. S. Manjunath and U. Madhow, "Estimation of Optimum Coding Redundancy and Frequency Domain Analysis of Attacks for YASS - a Randomized Block Based Hiding Scheme", *IEEE International Conference on Image Processing (ICIP08)*, San Diego, CA, October 2008 (Chap. 3)

(10) A. Sarkar, U. Madhow and B. S. Manjunath, "Matrix Embedding with Pseudorandom Coefficient Selection and Error Correction for Robust and Secure Steganography", *submitted to IEEE Transactions on Information Forensics and Security*, July 2009 (Chap. 4)

(11) A. Sarkar, V. Singh, P. Ghosh, B. S. Manjunath and A. Singh, "Efficient and Robust Detection of Duplicate Videos in a Large Database", *submitted to IEEE Transactions on Circuits and Systems for Video Technology*, March 2009

(12) A. Sarkar, L. Nataraj and B. S. Manjunath, "Detection of Seam Carving and Localization of Seam Insertions in Digital Images", *11th ACM Workshop on Multimedia and Security (ACM MMSec 09)*, Sep 2009 (Chap. 6)

(13) A. Sarkar and B. S. Manjunath, "Double Embedding in the Quantization Index Modulation Framework", *IEEE International Conference on Image Processing (ICIP09)*, Nov 2009. (Chap. 3)

(14) L. Nataraj, A. Sarkar and B. S. Manjunath, "Adding Gaussian Noise to "Denoise" JPEG for detecting Image Resizing", *IEEE International Conference on Image Processing (ICIP09)*, Nov 2009. (Chap. 6)

(15) A. Sarkar and B. S. Manjunath, "Image Dependent Log-likelihood Ratio Allocation for Repeat Accumulate Code based Decoding in Data Hiding Chan-

nels", accepted for publication in *Proc. of SPIE - Security, Steganography, and Watermarking of Multimedia Contents (XII)*, January 2010

Abstract

Novel Image Data-Hiding Methodologies for Robust and Secure Steganography with Extensions to Image Forensics

by

Anindya Sarkar

In the modern age, the proliferation of digital multimedia content has led to it being used as a medium of secure communication. The art of secret communication using a covert medium like images is called steganography while the competing technique of detecting the presence of embedded data in media through statistically learnt features is called steganalysis. How can the communication be kept secure while transmitting images with hidden content? For that, the hiding should not introduce perceptual distortions and also be robust against steganalysis. The emphasis of current research has been focused more towards detection than developing novel hiding methods. Hence, the steganalysis performance of state-of-the-art detectors is near-perfect against current steganographic schemes. Our emphasis in this dissertation has been on developing novel, robust and secure hiding schemes that can resist steganalytic detection. The impact has been two-fold. Hiding schemes are characterized by three complementary requirements - security against steganalysis, robustness against distortions in the transmission

channel, and capacity in terms of the embedded payload. Firstly, we show that our proposed schemes achieve significant improvements with respect to the above trade-offs. Secondly, since improvements in one field always fuels research in its complementary field, there has been a host of detection methods designed specifically in response to our proposed hiding methods.

We have contributed towards improving the steganalysis performance and also estimating the hiding capacities of the previously proposed statistical restoration (SR) methods. Since SR was mainly secure against histogram-based features, we have proposed a randomized block-based hiding scheme, tailored for JPEG-based steganalysis. Most detection schemes for JPEG images exploit the fact that hiding works in 8×8 blocks and significant statistical changes can be observed for block-based steganalysis. Our solution of hiding in randomized block locations desynchronizes the steganalyst and results in very low detection rates. We further improve the steganographic security by using matrix embedding and showing how it can be used along with suitable error correction coding schemes - matrix embedding was previously used only for passive (noise-free channels) steganography.

For the secure steganographic methods, we have considered global image attacks and hence, the synchronization between the hiding coefficients is unaffected. However, practical attacks can also include cropping and geometric transformation based attacks. We propose a key-point based hiding method for data recovery

after such attacks. The crux of the proposed method is a geometric transformation estimation algorithm which ensures that the received image can be properly aligned to the original, even after severe compression.

We have also worked on extending the domain of applicability of the steganalysis features. It has been observed that apart from detecting images with hidden content, steganalysis features can also be used for distinguishing real images from tampered ones. This comes under the purview of image forensics where like steganalysis, we also solve a two-class (real and tampered images) classification problem. We propose robust re-sampling detection methods (re-sampling is commonly present in tampered images) and also show how seam carving can be used for image tampering and object removal.

Contents

Acknowledgments **v**

Curriculum Vitæ **x**

Abstract **xv**

List of Figures **xxii**

List of Tables **xxxi**

1 Introduction **1**
1.1 Introduction to Data Hiding 3
1.1.1 Example of a Data Embedding System 7
1.1.2 Decoding Process 11
1.2 Steganography and Secure Communication 13
1.3 Thesis Outline 17
1.4 Overview of Contributions and Impact 24

2 Statistical Restoration Based Steganography **28**
2.1 Statistical Restoration (SR) 31
2.1.1 Literature Survey 33
2.2 Second Order Statistical Restoration 34
2.2.1 Earth Mover's Distance for Statistical Restoration 35
2.2.2 Odd-Even Based Hiding Framework 38
2.2.3 EMD Formulation for Image Steganography 39
2.2.4 Joint Intra- and Inter-Block Compensation 42
2.2.5 Experiments and Results 47
2.3 Finding Optimal Hiding Fraction 52

2.3.1 Using the Odd-Even Hiding Framework 55
2.3.2 Hiding Fraction and Rate Computation 56
2.4 Extension to Higher Order Statistics 59
2.4.1 Variation of Hiding Rate With Order of Co-occurrence Statistics 61
2.5 Total Compensation vs Individual Compensation 64
2.5.1 Use of Individual Frequency Coefficients 65
2.6 Steganographic Capacity Estimation 70
2.6.1 Transform Domains used for Hiding 72
2.6.2 Steganographic Capacity Constraints 73
2.6.3 Satisfying Statistical and Perceptual Constraints 77
2.6.4 Accounting for Channel Distortions 78
2.6.5 Results and Discussions 82

3 YASS - a Randomized Block based Hiding Framework 88
3.1 Steganography and Steganalysis - a Review 91
3.2 Requirements of a Blind Steganalysis Scheme 94
3.3 YASS for JPEG Steganography 97
3.4 Coding Framework 100
3.5 Optimum Coding Redundancy Factor 102
3.6 YASS: Experiments and Results 107
3.6.1 Embedding Rates 108
3.6.2 Detection Results 110
3.6.3 Comparison with Other Hiding Methods 114
3.6.4 Comparison with Standard Hiding at Same Rate 117
3.7 Extensions to YASS 119
3.7.1 Approaches Used to Increase the Embedding Rate 120
3.7.2 Further Randomization: A Mixture based Approach 121
3.7.3 Attack-aware Iterative Embedding 126
3.8 YASS Extensions: Experiments and Results 127

4 Matrix Embedding for Robust and Secure Steganography 134
4.1 Matrix Embedding based Active Steganography 137
4.2 Overall Hiding Setup 145
4.3 ME-RA Embedding 148
4.4 ME-RA Decoding 152
4.4.1 LLR Computation- Method M1 and Method M2 155
4.4.2 LLR Computation For ME-RA - Method 3 (M3) 161
4.5 Accounting for Channel Effects 164
4.6 Punctured RA Codes 169

4.7 Comparison of LLR Computation Methods . . . 178
4.8 Experiments and Results . . . 180
4.8.1 Setup for Steganalysis Experiments . . . 181
4.8.2 Discussion of Experimental Results . . . 184
4.9 Estimating Redundancy Factor for ME-RA . . . 193
4.10 Summary . . . 200

5 Robust Key-point based Data Hiding 203
5.1 The Challenges in Robust Hiding . . . 206
5.1.1 Specific Contributions . . . 212
5.2 Related Work . . . 213
5.3 Robust Image Data Hiding Framework . . . 221
5.4 Peak based Alignment . . . 228
5.4.1 Estimating Transformation Matrix . . . 228
5.4.2 Peak Insertion Methods . . . 232
5.4.3 Ratio based Peakiness Function . . . 235
5.4.4 Experimental Comparison of Various Methods . . . 237
5.5 Accurate Parameter Estimation . . . 240
5.5.1 Using Prior Assumption about the Geometric Transformation 241
5.5.2 Using JPEG Peak Location Property . . . 246
5.6 Experimental Results . . . 255
5.6.1 Effects of Varying DFT Size . . . 256
5.6.2 Effects of Cropping . . . 257
5.6.3 Robustness to Small Local Transformations . . . 259
5.7 Method for Robust Key-point Selection . . . 260
5.7.1 Design Criteria - selecting the cutoff for key-point detection (δ_{th}) and the embedding region size (B) . . . 264
5.8 Overall Results for the Hiding Framework . . . 268
5.9 Summary . . . 276

6 Extensions to Image Forensics 280
6.1 Common Image Tampering Methods . . . 282
6.2 Re-sampling Detection: a Forensics Tool . . . 285
6.3 Detecting Resizing after JPEG Attacks . . . 287
6.3.1 Effects of JPEG Compression . . . 290
6.3.2 Masking JPEG Peaks using AWGN . . . 292
6.3.3 Effect on the EM-based Method . . . 296
6.3.4 Summary . . . 299
6.4 Detection of Seam Carving and Localization of Seam Insertions . 300
6.4.1 Seam Carving for Image Tampering . . . 301

6.4.2 Seam Carving and Seam Insertion 308
6.4.3 Markov Feature to Detect Seam Carving and Insertion . . 313
6.4.4 Detection Using Markov Feature based Models 316
6.4.5 Localizing Seam Insertions 318
6.4.6 Probabilistic Approach for Detecting Seam Insertions . . . 335
6.4.7 Rotation/Scale Detection 338
6.5 Localization of Seam Carving based Object Removal 341
6.5.1 Object Removal 344
6.5.2 Multi-classifier based Approach to Tamper Localization . . 347
6.5.3 Results for Object Removal Localization 351
6.5.4 Summary 353

7 Conclusions and Future Work **355**

Bibliography **361**

List of Figures

1.1 Block diagram of a data hiding system - the composite signal y after channel attacks is a function ($f(\cdot,\cdot)$) of the stego signal s and the channel noise n. $D(\cdot,\cdot)$ refers to a perceptual distance measure. 4
1.2 An end-to-end data hiding system which uses suitable error correction coding 6
1.3 A step-by-step of the high volume image data hiding framework developed by researchers at the Vision Research Lab (VRL), UCSB (2001-05) - embedder side 8
1.4 The RA code based framework is used to determine the code bits for a given sequence of data bits. The code bits get embedded at the selected image locations using QIM. 12
1.5 The steganography problem 15

2.1 Explaining the 1-D statistical restoration framework (pmf = Probability Mass Function = Normalized Histogram) 31
2.2 Histogram computation row-wise, considering overlapping and non-overlapping pairs of coefficients, per 8×8 blocks 40
2.3 (a) Alternate scanning along the rows of an image to obtain spatially correlated blocks [42] (b) The intra and inter-block based matrix A has H and C terms arranged alternately. 43
2.4 Explanation of the intra-block and inter-block histogram computation for the $N_r \times N_c$ matrix A; in this example, N_r=N_c=4 44
2.5 Comparison of detection curves for a variety of features using (a) joint compensation and (b) EMD-based compensation methods 51
2.6 Average detection error (P_{total}), averaged over all the test images, computed for different QDCT streams after statistical compensation: "Individual" and "Total" refer to the individual and total compensation schemes, respectively. 67

2.7 Comparison of PMF differences, after statistical restoration, for individual and total compensation based methods, for different QDCT streams: here, "Orig", "Indiv" and "Total" refer to the original PMF, PMF after hiding and individual compensation, and PMF after hiding and total compensation, respectively. 69
2.8 (a) DWT computation using 3 levels - the modified scanning procedure for the 3-level decomposition is shown, (b) zigzag scan for block-based DCT coefficients as in the JPEG coding standard 74
2.9 The framework to hide the maximum number of databits, maintaining the statistical (use $\lambda^\star$ as the hiding fraction) and attack constraints (use minimum redundancy that ensures zero BER) is shown here. The perceptual constraint is implicit in the QIM based hiding scheme. At first, $\lambda^\star$ is computed and then, after separating X into H and C based on $\lambda^\star$, the optimum redundancy factor q_{opt} is determined. Here, T denotes the threshold where quantized DCT coefficients in the range $[-T, T]$ are used for hiding. 79
2.10 Computation of the data hiding capacity depends on the 2×3 transition probability matrix mapping R to Z' - shown here for DCT domain hiding and for JPEG compression attack (LLR = Log Likelihood Ratio) 80
2.11 Variation of the optimum hiding fraction, $\lambda^\star$ (expressed as %), with the design quality factor QF_h, for DCT and DWT domains. 84
2.12 Variation of mean($\mathcal{R}_{max}$) and mean($\mathcal{R}_{prac}$) with the attack quality factor (QF_a) for JPEG attack, for different design quality factor QF_h, for DCT domain hiding. 85
2.13 Variation of mean($\mathcal{R}_{max}$) and mean($\mathcal{R}_{prac}$) with the attack quality factor (QF_a) for JPEG attack, for different design quality factor QF_h, for DWT domain hiding. For fixed QF_h, higher QF_a indicates finer quantization and hence a less noisy channel, and so we achieve higher data-rates. 86
2.14 Variation of mean($\mathcal{R}_{max}$) and mean($\mathcal{R}_{prac}$) with the inverse compression ratio (ICR) for JPEG-2000 based attack, for different design quality factor QF_h, for DWT domain hiding. A higher value of ICR indicates less severe compression and a higher data-rate. 87

3.1 Description of Self-calibration mechanism - $F(\cdot)$ is the functional which computes the feature vector for a given image, IDCT = Inverse DCT 95

3.2 (a) YASS hiding methodology: Data is hidden in randomly chosen 8×8 blocks within a big block of size $B \times B$, with $B > 8$. This example uses $B = 10$. (b) The block structure as seen by a steganalyst, who gets *out-of-sync* with the blocks used during embedding, and cannot synchronize even if the embedding method and its parameters are known. 99
3.3 The data hiding system using RA-code based error correction, where q is efficiently estimated at the decoder 103

4.1 The entire hiding framework using RA coding, matrix embedding and YASS-based coefficient selection 142
4.2 There is a 7-element sequence of DCT coefficients $\mathbf{y}$ where a 3-tuple bit sequence $\mathbf{b}$ is embedded. The encoder embeds $\mathbf{b}$ in $\mathbf{y}$ by changing the minimum number of coefficients in $\mathbf{y}$, thus modifying it to $\mathbf{y}^e$ which is transmitted through the channel and $\mathbf{y}^r$ is the corresponding received sequence. The decoder uses $\mathbf{y}^r$ to estimate the LLR values for the 3 bit locations. The sequence $\mathbf{a}^r$, which is derived from $\mathbf{y}^r$ ($a_i^r = \text{mod}(\text{round}(y_i^r), 2),\ \forall i$), is not explicitly shown at the decoder side. 145
4.3 Embedding logic used at the encoder for a (7,3) matrix embedding example : cases (A)-(D) correspond to cases where embedding is possible, while case (E) is where embedding fails. 150
4.4 The mapping between the binary RA code bits to the ternary sequence obtained from the continuous valued LLRs at decoder output is shown here for the ME framework - "channel" refers to the JPEG compression channel that introduces errors and erasures in the mapping from $\mathbf{z}$ to $\hat{\mathbf{c}}$. The additional erasure locations for the **ME-RA-puncture** scheme are selected in the final RA encoded sequence ($\mathbf{c}$) and not in the intermediate sequences ($\mathbf{r}$ or $\mathbf{x}$). **For RA decoding, continuous valued LLRs are used - the ternary sequence ($\hat{\mathbf{c}}$) is used only to represent the channel as a 2×3 transition probability matrix**. . 173
4.5 **PF-274** is used for steganalysis in these plots - (a) trade-off of hiding rate vs detection accuracy is shown considering different parameter settings for ME and QIM based hiding, as used in Tables 4.13-4.14, (b-d) comparison of ROCs is done for (b) ME vs QIM at comparable bpnc, (c) variants of ME, and (d) variants of QIM (varying λ). Here, B=9 unless otherwise mentioned. The diagonal line corresponding to a fully random detector is kept as reference - the closer a ROC curve is to this line, more secure is the hiding method. 193

4.6 The data hiding setup for ME-RA method where the decoder has to correctly estimate the q that was independently decided upon by the encoder. 195

5.1 (a) original image, (b) image after global attack (ocean-ripple filter in Photoshop), (c) image after geometrical transformation (rotation and scale), cropping (local attack) and global attack on resultant image (dust filter in Photoshop), (d) same as (c) but dust filter is replaced by offset filter which involves translations along x and y-axes. 207
5.2 (a) original image with key-points (KP) marked, with the square boxes showing the embedding regions, (b) a geometrically transformed image is aligned back with the original grid - the KP that are common with (a) are shown, (c) image after geometrical transformation and cropping and it is not aligned with the original image grid - the KP that are common with the original are shown, but the hiding regions are not aligned with those used at the encoder, (d) image is created using "pinch" filter in Photoshop, and all KP are shown. The KP in (a)-(d) are obtained using Nobel-Forstner [32, 77] detector followed by pruning, as discussed in Section 5.7. 209
5.3 The end-to-end data hiding framework (a) **encoder**: the same encoded sequence is embedded in every local region around a key-point, (b) **channel**: the stego image can be subjected to geometrical transformations, followed by image processing attacks, (c) **decoder**: it aligns the received image with the original image grid and decodes data from the local regions around detected key-points. The boxes outlined in bold represent the main challenges/tasks at the encoder and the decoder. . . 211
5.4 Block diagram at the encoder side - numbers (1)-(4) correspond to the encoder side modules. 227
5.5 Block diagram at the decoder side - numbers (1)-(5) correspond to the decoder side modules. 227
5.6 It is seen that the striations become more visible as the PSNR decreases: (a) original image; the other images are obtained after template addition in the DFT domain at different PSNRs (dB) (b) 31.34 (c) 33.46 (d) 36.17 (e) 37.91 (f) 39.68 (g) 41.59 (h) 45.85. The periodic patterns are very clearly evident in the lower PSNR images and the visibility of the periodic patterns progressively decreases with increased PSNR. . . 233

5.7 The figure shows the 512×512 $\mathbf{T}_\Delta$ plot for a transformation of $\{\theta=10^\circ, s_x=s_y=1.1\}$, along with 20% cropping, and QF_a=75. P_i, the original peak location, is shifted to R_i after the geometric transformation. The 20 topmost peaks are shown per quadrant. Due to the window based peak insertion, many peak locations are clustered together; hence we see fewer peaks per quadrant. We observe that JPEG-induced peaks are generally separated at multiples of 64 units apart, horizontally and vertically. 247
5.8 Similar plot, as in Fig. 5.7, with the transformation corresponding to $\{\theta=30^\circ, s_x=1, s_y=1.2\}$, along with 20% cropping, and QF_a=75. . . 248
5.9 (from left to right) (a)-(d) correspond to $\theta = 10^\circ, s_x = s_y = 1.1$, $\theta=30^\circ, s_x=1, s_y=1.2$, $\theta=20^\circ, s_x=s_y=1.1$, and $\theta=15^\circ, s_x=1.1, s_y=1.3$, along with 20% cropping and $QF_a = 75$. The circled locations denote $\{R_i\}_{i=1}^4$, while the horizontal and vertical lines show how the JPEG-induced peaks (white dots) are at multiples of 64 units apart from $\{R_i\}_{i=1}^4$. 249
5.10 Various cropping approaches for a $N_1 \times N_2$ image - the greyish part denotes the cropped out (discarded) image part. 259
5.11 The KP pruning steps are shown. For two prospective KP ((M_3, M_4) or (M_5, M_6)) which are less than B apart from each other, we retain the one with the higher strength. When the distance between two points (e.g. (M_1, M_2)) is greater than or equal to B, both can be retained. . . 261
5.12 Fig.(a) and (b): The figures show the fraction of blocks that are correctly detected using SIFT key-points at the decoder side before and after pruning, respectively. 263
5.13 Fig. (a) shows the fraction of detected and successfully decoded blocks. Fig. (b) shows the number of key-points detected at the decoder side (K_{dec}), the number of local blocks that are correctly detected ($K_{dec}.fr_{match}$) and decoded ($K_{dec}.fr_{match}.fr_{BER,0}$). Fig.(c): The number of successfully embedded data bits is shown. The number of data bits embedded in a $B \times B$ region $= \lfloor \frac{B}{8} \rfloor^2 . \frac{\lambda}{q_{opt}}$ where λ elements are used for hiding per 8×8 block and the minimum RA-code redundancy factor needed for proper decoding is q_{opt}. We use $QF_h = 60, \lambda = 5, QF_a = 75$. 267
5.14 Based on the above experiments, Nobel-Forstner (NF) key-points perform better than Harris and SIFT key-points; here p_{succ} is the fraction of images for which we successfully retrieve the embedded data. The experiments are performed on 250 images and the average fr_{match} is reported. In (c)-(d), a crop fraction of 60% means 60% of the image is retained along both the axes. 269

5.15 The same experiments as in Fig. 5.14 are repeated under a different set of transformations. In (g), R=30 refers to a rotation angle of 30°, S=0.75 refers to a scaling factor of 0.75 for both the axes. 270
5.16 (a)-(l) images after various Photoshop attacks 277
5.17 (a)-(l) corresponding DFT magnitude plots after various Photoshop attacks 278

6.1 Visual illustration of methods for image tampering: (a) Object 1 is taken from Image 1. Re-sampling involves scaling the object in order to ensure that it fits properly in Image 2. Image 3 is the result of splicing Object 2 in Image 2, (b) in copy move forgery a portion of Image 1 (Object 1) is copied and pasted in the same image, (c) inpainting fills the void left from object removal with texture and structure expansion. 284
6.2 The DFT for the 1-D signal obtained by taking the mean of the second difference computed per row is shown here. X-axis shows the DFT indices. (a) DFT for the original TIFF image, without re-sampling, does not show any peaks as the second difference signal does not show any periodicity. (b) DFT for a re-sampled image, re-sampling factor being 1.5, shows peaks at locations S_1 and S_2. The DFT size is changed depending on the image size. E.g., the original image was 256×256 and so the DFT size used in Fig. (a) was 256. After re-sampling by 1.5, the new image is 378×378. The DFT size used in Fig. (b) is 378. It should be noted that if we maintained a suitably larger value for DFT size (to prevent aliasing), we would still see the re-sampling peaks. Here, for ease of illustration, we maintain the DFT length as equal to the image dimension. 289
6.3 The DFT for the mean of the second difference signal is shown for the following images: (a) DFT for the original JPEG image, without re-sampling, shows peaks $J_1, \ldots, J_6$ at an interval of $\frac{1}{8}\times$(DFT length) (b) DFT for a re-sampled image, re-sampling factor being 1.5, shows peaks at locations S_1 and S_2. 290
6.4 JPEG compression is performed (after bilinear interpolation by a factor of 3) at different QF, from 30-100. The size of DFT used is 1500 while the sampling peaks S_1 and S_2 lie at 500 and 1000. These sampling peaks can occur due to sampling factors of 1.5 and 3. $\frac{1}{\text{sampling factor of 3}} = \frac{500(\text{Location of } S_1)}{1500(\text{DFT size})}$. Also, $\frac{1}{\text{sampling factor of 1.5}} = \frac{1000(\text{Location of } S_1)}{1500(\text{DFT size})}$ 291

6.5 By adding suitable amount of Gaussian noise, the peaks due to JPEG compression are suppressed while still retaining the sampling peaks - as seen at 20 dB SNR. Below 20 dB SNR, both the JPEG and sampling peaks are suppressed. 293
6.6 Filtering the JPEG compressed image, at QF of 80 294
6.7 Filtering the JPEG compressed image, at QF of 70 295
6.8 Filtering the JPEG compressed image, at QF of 60 295
6.9 DFT of the p-map after (a) resizing the image by a factor of 3: note 3 peaks (b) JPEG on the resized image at a QF of 85: no distinct peaks can be observed (c) adding AWGN on JPEG image at 35 dB SNR, (d) 40 dB SNR: 3 peaks can again be seen in (c) and (d). 297
6.10 DFT of the p-map for the (a) original image (b) image rotated by 5° (c) rotated image is JPEG compressed at QF = 75 (d) AWGN is then added to the rotated JPEG image at 40 dB SNR 298
6.11 (a) original image, (b), (c) and (d) are modifications of (a) with the same number of rows and 1%, 5% and 10% more columns, respectively. 302
6.12 It is similar to Fig. 6.11 except that the seams are also clearly shown for (b), (c) and (d) to explain how the image has been modified to increase its width. 303
6.13 Steps (a)-(h) show how the object removal is done using seam carving and seam insertion: (a) original image, (b) the mask for object removal is shown in black, (h) the image after object removal and having the same size as the original is shown. 306
6.14 Example of seam carving for a 4x5 matrix **a** 312
6.15 Example of seam insertion: (a) $\{a_{i,j}\}_{i=1,j=1}^{i=4,j=5}$ and (b) $\{b_{i,j}\}_{i=1,j=1}^{i=4,j=6}$ are the image matrices before and after seam insertion, respectively. For points along the seam, the values are modified as shown for the first row: $b_{1,1}=a_{1,1},\ b_{1,2}=a_{1,2},\ b_{1,3}=\text{round}(\frac{a_{1,2}+a_{1,3}}{2}),\ b_{1,4}=\text{round}(\frac{a_{1,3}+a_{1,4}}{2}),\ b_{1,5}=a_{1,4},\ b_{1,6}=a_{1,5}$. 319
6.16 Understanding linear relationship between seam inserted pixels . . 319
6.17 The five images shown here are gray-scale versions of color images obtained from the UCID database - we have used gray-scale images for all our experiments. These are the original images based on which the visual examples in Section 6.4.5 are obtained. 323

6.18 (a) the image, after 3% seam insertion, with seams marked in red, (b) P matrix for the seam inserted image, (c)/(d) pruned P matrix with points having valid parent/child nodes marked in white (e) P_B matrix, with retained points (marked in white) having valid parent and child nodes. Comparing (a) and (e), we see that the detected seams are very similar to the actual inserted seams. 325
6.19 (Localizing seam insertions: parts (a,d,g,j), (b,e,h,k) and (c,f,i,l) correspond to 1%, 5% and 10% seam insertions. The seams are marked with thicker red lines in 2^{nd} row for ease of visualization. The 3^{rd} and 4^{th} rows display the actual and the detected seams, respectively. 326
6.20 The seam insertion localization is performed on another image, similar to the steps in Fig. 6.19. 327
6.21 (a) 3% seam-inserted image (b) P matrix for seam inserted image (no smoothing) (c) P matrix after parent based pruning (d)/(e) P_B for seam-inserted/original image (f) seam-inserted image after smoothing using s_{frac} of 0.98 (g) P matrix after smoothing (h) corresponding P matrix after parent based pruning (i)/(j) P_B for seam-inserted/original image (using s_{frac} of 0.98). **The green image means an all-zero image**. 330
6.22 (a) Original image's Canny edge-map (b) 8% seam-inserted image's Canny edge-map (c) original image after smoothing using s_{frac} of 0.65, (d)/(e) P_B for seam inserted/original image (s_{frac} of 0.65) (f) 8% seam-inserted image (g)/(h) smoothing on seam-inserted/original image (s_{frac} of 0.45) (i)/(j) P_B for seam-inserted/original image (s_{frac} of 0.45). The columns discarded from the seam-inserted image are similar to the original for s_{frac} of 0.65 (hence, only 1 figure (c) is shown) - however, the columns removed from the seam-inserted and original images differ significantly for s_{frac} of 0.45, as shown in (g) and (h). **The green image means an all-zero image**. 331
6.23 (a)/(b)/(c) False alarm (P_{FA}) and missed detection (P_{MD}) rates are computed for TIFF images/ JPEG images/ JPEG images with "relaxed conditions for obtaining positive elements in P". Similarly, (d)/(e)/(f) show the detection error rates for these three cases. The number in parentheses denotes the seam-insertion percent, for example, in (a), $P_{MD}(40\%)$ denotes the probability of missed detection for 40% seam insertion. 334

6.24 (a) image with 25% seams inserted, (b) p-map matrix for seam inserted image, (c) binary matrix obtained after using δ_{th} of 0.98 for p-map, (d) only the pixels in (c) with valid parent nodes are retained, (e) the pixels in (c) with valid parent and child nodes are shown - these indicate the seams detected, (f) original image, (g) p-map for original image, (h) corresponding binary matrix using δ_{th} of 0.98, (i) the binary matrix with parent node based pruning, (j) after (parent+child) node based pruning, the resultant binary matrix has all zeros. **The green image means an all-zero image**. 337
6.25 Object removal with the seam carving algorithm: (a) original image, (b) the white spot shows the region of interest, (c) seams to be removed: they are forced to pass through the region of interest (marked as ROI): we use μ_i and η_i to denote the first and last column affected by seam carving in the i^{th} row, respectively, and c_r denotes the number of seams removed, (d) is the image after seam carving, and (e) shows the zones affected by seam carving in the tampered image, (f) the seam insertions are shown which restore the original image size. 342
6.26 Zones to be removed by seam carving in its traditional use for content-aware image resizing are shown in (a). The seams which are to be removed for object removal are shown in (b-d). In (b) the arrows highlight how in the object area all the seams are concentrated in one chunk; see a zoomed image in (c). In (e) the arrows show the scattered seams to be removed outside the object area, and a zoomed-in version is shown in (d). 348
6.27 Block based approach: the division of the image into blocks of different types is shown in (a), the identification of the three classes is shown in (b), the zones affected by the seam carving and consequent labeling of the different zones for the two-phase approach is shown in (c). Figure (d) shows the two-phase approach: 1) each vertical slice is classified as tampered(containing object removal region)/untampered. 2) the vertical slice is divided into horizontal chunks for more precise localization of the object removal region. 350
6.28 Classification accuracy for (a) identifying the tampered vertical slice from out of 3 slices (b) identifying the tampered zone from out of 3×3=9 zones. 352

List of Tables

1.1 **Commonly used Acronyms in the Dissertation** 2

2.1 Comparison of the performance of the three compensation methods, when steganalysis experiments are performed using five different features. P_{FA} and P_{miss} represent the probability of false alarm and of missed detection, respectively. **The rows contain the five features while the columns denote the three methods**, which are not to be confused though they have some names in common. The effective probability of error $P_{error} = \frac{1}{2}(P_{FA} + P_{miss})$. 49
2.2 Variation of the optimum hiding threshold, fraction and capacities with the order of co-occurrence, being averaged over 4500 images - since the threshold can assume only integer values, T_{opt}, after averaging, is changed to the nearest integer higher than it. 63

3.1 Number of information bits that can be hidden in some standard images of size 512×512. The big-block size $B = 9$. 109
3.2 Hiding rates for the 512×512 Lena image for different big-block sizes B. *bpnc* denotes bits per non-zero coefficients and *data-bits* denotes the number of hidden information bits. 110
3.3 Hiding rates for the 512×512 Lena image for larger big-block sizes $B = 9$, 25, 49 and 81, which can incorporate several 8×8 blocks. 110
3.4 JPEG dataset: Steganalysis results for randomized block based hiding, when big-block size B is varied. It can be seen that the detection is random for most of the configurations. 113
3.5 TIFF dataset: Steganalysis results for randomized block based hiding, when the big-block size B is varied. It can be seen that the detection is random for most of the configurations. 114

3.6 Using larger big-block size B: Steganalysis results for randomized block based hiding on the TIFF image dataset, for block-size $B = 9$, 25 and 49. For 9×9, 25×25 and 49×49 blocks, we use 1, 9 and 36 8×8 sub-blocks respectively for hiding. 115
3.7 Using larger big-block size B: Steganalysis results for randomized block based hiding on the TIFF image dataset, for block-size $B = 9$, 25 and 49. For 9×9, 25×25 and 49×49 blocks, we use 1, 9 and 36 8×8 sub-blocks respectively for hiding - using cropping of 1 pixel per dimension. 115
3.8 Steganalysis results for comparing the randomized block based scheme (YASS) with OutGuess (OG) and Steghide (SH) schemes, used at rates of $\frac{1}{10}$ and $\frac{1}{40}$, for the TIFF image dataset. For OutGuess and Steghide, the images are JPEG compressed using a quality factor of QF_a before being presented to the steganographic scheme. Note that the QF_h parameter is applicable only for the YASS scheme. 117
3.9 Comparison of steganalysis results for randomized block (RB) hiding (i.e. YASS) with $B = 9$, and randomized frequency (RF) based hiding. 119
3.10 Variation in bpnc with variation in QF_h : QF_a is set at 75. The bpnc is maximum for QF_h=50. For lower QF_h, bpnc decreases as more coefficients get erased; for higher QF_h, the attack due to QF_a is more severe and introduces more bit errors. Big block size B is set to 9. . . . 129
3.11 Variation of detection accuracy with change in the design quality factor for hiding, QF_h for DCT-based YASS scheme, with *big-block size* B=9. After hiding, the image is JPEG compressed to an attack quality factor of QF_a=75 and this same quality factor is used for steganalysis. For ensuring that the detection accuracy $P_d \leq 0.60$, we need QF_h to be ≥ 69. 130
3.12 Variation in bpnc with variation in QF_h - e.g. in the first row, the average $QF_h \approx 60$ for all the 3 methods. The 1^{st} method uses a constant QF_h for all the blocks. The 2^{nd} method uses a mixture of {50,60,70} for hiding, with the QF per block being decided randomly. The 3^{rd} method uses the same mixture, of {50,60,70}, for hiding, with the QF per block being decided based on the local block variance. It is seen that the embedding rate (bpnc) is higher with the mixture scheme, only if the QF allocation is done based on the block variance. B is set to 9. 131

3.13 The effect of varying the partitioning of the variance values is studied. Here, Avg. QF_h refers to the average QF obtained, after considering the QF allocation over all the blocks. Note that in this table, the interpretation of, $[0, 1, 4, \infty]$ where the mixture has $QF_h = 50$, 60 and 70, is that a QF_h of 70/60/50 is used for the zone [0,1)/[1,4)/[4, ∞) of block-based variance values, respectively. 132
3.14 Comparison of YASS-M1 iterative embedding scheme with other methods - the interpretation of the schemes in the left-most column is as follows: (i) YASS: $QF_h = 60$ refers to the original YASS hiding using a constant $QF_h = 60$, (ii) *50-60-70-rand* refers to hiding using a mixture of QF values, 50, 60 and 70, when the allocation is done randomly, (iii) *50-60-70-rand-M1* refers to the M1 iterative embedding scheme (1 iteration) being used after hiding using *50-60-70-rand* method, (iv) *50-60-70-var* refers to hiding using a mixture of QF values, 50, 60 and 70, when the allocation is done based on local variance, and (v) *50-60-70-var-M1* refers to the M1 iterative embedding scheme (1 iteration) being used after hiding using *50-60-70-var* method. 132
3.15 Variation of the average detection accuracy with the hiding rate (in bpnc) for F5 133
3.16 Variation of the average detection accuracy with the hiding rate (in bpnc) for F5 133

4.1 **Glossary of Notations** 146
4.2 Explanation of how $\{b_1, b_2, b_3\}$ is embedded by minimally changing $\{a_1, a_2, \ldots, a_7\}$ at the encoder side: here, we make minimum changes to $\mathbf{a} = \{a_1, \ldots, a_7\}$ to obtain $\mathbf{a}^e = \{a_1^e, \ldots, a_7^e\}$ such that $\mathbf{b}^T = \mathbf{H}(\mathbf{a}^e)^T$, where $\mathbf{b} = (b_1, b_2, b_3)$. 148
4.3 For (7,3) ME, there are 8 possible relations between **b**, **the 3-tuple of bits to be embedded**, and $\mathbf{b}'$, **the syndrome obtained from y**. There is always a single coefficient y_i, whose modification to y_i^e (ensuring $a_i^e = \overline{a_i}$ where $a_i = \text{mod}(\text{round}(y_i), 2)$ and $a_i^e = \text{mod}(\text{round}(y_i^e), 2)$) ensures proper embedding. If y_i is in the erasure zone, the candidate 2-tuples/3-tuples for embedding are mentioned. 149
4.4 For (7,3) ME, there are 8 possible relations between **b**, **the 3-tuple of bits to be embedded**, and $\mathbf{b}'$, **the syndrome obtained from y**. There is always a single coefficient y_i, whose modification to y_i^e (ensuring $a_i^e = \overline{a_i}$ where $a_i = \text{mod}(\text{round}(y_i), 2)$ and $a_i^e = \text{mod}(\text{round}(y_i^e), 2)$) ensures proper embedding. If y_i is in the erasure zone, the candidate 2-tuples/3-tuples for embedding are mentioned. 149

4.5 Explanation of LLR allocation for Example-1: conditions $\{0, 3, 4, 7\}$ are the NEB conditions. Here, the syndrome based on $\mathbf{y}^r$, $\{b_1^r, b_2^r, b_3^r\} = \{0, 0, 1\}$. **If condition j is NEB, the actual bit sequence that was embedded is assumed to be $\{b_1^r, b_2^r, b_3^r\}$, while for an EB condition, the bit sequence that should have been embedded is assumed to be** $\{f_j(b_1^r), f_j(b_2^r), f_j(b_3^r)\}$ for M2 (functions f_j are obtained from Tables 4.3 and 4.4). For a certain condition, "Bits" refers to the original sequence that is embedded (for a NEB condition) or the sequence that was supposed to be embedded (for an EB condition). We assume $w_{NEB} = 1/2$. 161
4.6 LLR computation for Method 3 based on (4.9) - explained using Example 2, and using $w_{EB}=0.5$, $w_{NEB}=1$ 164
4.7 LLR computation for Method 3 based on (4.9) - explained using Example 3, and using $w_{EB}=0.5$, $w_{NEB}=1$ 164
4.8 Variation of the hiding rate (bpnc) with different LLR computation methods for (7,3) ME-RA, with B=9, QF_a=75, and using the first 19 AC DCT terms for hiding ($\lambda=19$), and M3 to compute individual LLRs. 169
4.9 For each image, the bpnc is computed using **ME-RA-puncture**, using QF_h=60 and B=9. The bpnc increases for a suitable range of η (rate of additional erasures). Here, **LRE** is used for erasure distribution. 176
4.10 The bpnc values are computed using **ME-RA-puncture** for different erasure distribution methods, for varying QF_h and η, and B=9. The channel effects are considered for LLR computation for more noisy channels (QF_h of 60 and 70) using (4.14), while an ideal channel is assumed for QF_h of 50. 177
4.11 The bpnc values are computed using **ME-RA-puncture** for different η and QF_h and **LRE** for erasure distribution; we set $B = 9$ and use M3 for individual LLR computation. 177
4.12 Table comparing the hiding rate (bpnc) obtained after using M1, M2 and M3, for LLR computation, using $B = 9$ and $QF_a = 75$. For M1 and M2, the optimum α (that results in highest bpnc for a set of hiding parameters) value for LLR scaling is empirically determined and the reported bpnc corresponds to the optimum α. For ME-RA and ME-RA-non-shrinkage, we use $\lambda = 19$, while for ME-RA-puncture, we use $\eta=19$. The effective bpnc obtained using M3 is higher, in general, than that using M1 and M2. 179

4.13 Comparing detection performance (P_{detect}) and embedding rate (bpnc) using QIM-RA and "ME-RA-puncture" schemes - for B=9 and 10, QF_h=50, QF_a=75. The bpnc for "ME-RA-puncture (7,3)" (or ME-RA-puncture (3,2)) is higher than that of "QIM-RA: 4 terms" (or QIM-RA: 6 terms) while the latter is more detectable, for the self-calibration based features. 185
4.14 Comparing P_{detect} and bpnc for **QIM-RA** and **ME-RA-puncture** - for B=25 and 49, QF_h=50, QF_a=75 185
4.15 Comparing P_{detect} and bpnc for QIM-RA and "ME-RA-puncture", for B=9 and 10, $QF_h = 50$, and using randomly chosen DCT coefficients for QIM-RA. "ME-RA-puncture(7,3)" performs better than "QIM-RA: rand-8/10" while "ME-RA-puncture(3,2)" performs better than "QIM-RA: rand-12". 187
4.16 Comparing P_{detect} and bpnc for QIM-RA and "ME-RA-puncture", using B=9 and QF_h=60 and 70, and using randomly chosen DCT coefficients for QIM-RA. "ME-RA-puncture(7,3)" performs better than "QIM-RA: rand-8" while "ME-RA-puncture(3,2)" performs better than "QIM-RA: rand-10/12". 187
4.17 Comparing bpnc under various attacks - **"QIM-n"** refers to the **"QIM-RA: n terms"** method, while **(p,q)** refers to the **"ME-RA-puncture (p,q)"** method. For hiding, we use $QF_h = 50$, $B = 9$, and after the attack, the images are JPEG compressed using $QF_a = 75$. Here, the bpnc for "ME-RA-puncture(7,3)" and "ME-RA-puncture(3,2)" are compared with that of QIM-4 and QIM-6, respectively. 188
4.18 We repeat the experiments in Table 4.17 and replace the gamma variation attack by AWGN addition based attack. 189
4.19 Comparing P_{detect} for a variety of recently proposed features to detect YASS, using $QF_h = 50$. We use B=9 for the QIM schemes. The acronyms used for the various methods are the same as used in Table 4.17. 190
4.20 Comparing P_{detect} for a variety of recently proposed features to detect YASS, using $QF_h = 70$. We use B=9 for the QIM schemes. . . . 190
4.21 Comparing P_{detect} for two different sized datasets, using a variety of steganalysis methods, and different variants (embedding methods) of the YASS scheme - B=9 is used along with QF_h=50 and QF_a=75. . . 191
4.22 The bpnc values are compared for **ME-RA** and **QIM-RA** methods, before and after puncturing, at QF_h=50. 192

4.23 The results of estimating q over 50 images, where the q used for RA coding is varied from 10-43, are presented here, for (7,3) ME-RA and QIM-RA methods. Thus, the total number of cases over which q is estimated = $50 \times 34 = 1700$. The big-block size B is set to 9, while QF_a=75. An "error" occurs when the top peak in the correlation based method does not correspond to the actual q or its sub-multiples. Based on just the correlation, ME-RA performs better than QIM-RA in q-estimation. 200

5.1 **Glossary of Notations** 222
5.2 We compare methods M1 and M2 for estimating A correctly. **M2 performs better than M1**. 239
5.3 We compute p_A and average PSNR for **various window sizes** Δ and **varying window peak strengths** $[v_1 \cdots v_{\Delta+1}]$ - these denote the starting peak strengths. QF_a of 75 is used. 239
5.4 We repeat the same experiments as shown in Table 5.3 - the only difference being that we use the line-based method for peak insertion instead of the window-based method. 240
5.5 The geometric transformation parameter estimation results are presented for 3 different methods, for varying QF_a - here, $\mu = 10$, $\delta_\theta = 0.5°$ and $\delta_s = 0.05$. PSNR values are reported in dB. 245
5.6 The geometric transformation parameter estimation results are presented with/without using JPEG peak location property, for varying QF_a - here, $\delta_\theta = 0.5°$ and $\delta_s = 0.05$. 255
5.7 The rotation angle is fixed at 20° while the scale factor ($s_x = s_y$) is varied from 1.1 to 0.70, and $p_{A,m2}$ is computed using angle and scale resolutions of 0.5° and 0.05, respectively. 255
5.8 $p_{A,m}$ is computed for $\theta = 20°$, $s_x = 1.1$, $s_y = 1.1$ along with 20% cropping, for $QF_a = 75$, and various resolutions for rotation (δ_θ) and scale (δ_s) parameters, e.g. (0.5, 0.05) indicates $\delta_\theta = 0.5°$ and $\delta_s = 0.05$. PSNR values are in dB. 256
5.9 p_A is computed for different DFT sizes: 257
5.10 For the cropping experiments, $QF_a = 75$, the starting windowed peak strengths $[v_1\ v_2\ v_3\ v_4] = [14\ 10\ 4\ 2]$, and the results are shown after using various cropping methods - methods (a)-(c) are explained in Fig. 5.10. 258
5.11 Results with varying amounts of local transformations, using $QF_a = 75$ 260
5.12 As the threshold δ_{th} is increased, K_{enc} slightly decreases, and (E_{KP}+ E_{dec}) also increases slightly. We use $QF_h = 60$, $\lambda = 5$ and $QF_a = 75$. . 266

5.13 Effect of seam carving attacks for λ=5, $QF_h = 60$, $QF_a = 99$, B=80, the parameter t is such that nearby key-points are removed when they are $\frac{B}{t}$ apart. 273
5.14 Results with 3x3 gaussian blur, with varying σ, for $\lambda = 5$, $QF_h = 60$, $B = 80$ 274
5.15 Results with AWGN addition at varying SNR, for $\lambda = 5$, $QF_h = 60$, $B = 80$ 274
5.16 Results with gamma correction attack at varying γ, for $\lambda = 5$, $QF_h = 60$, $B = 80$ 275
5.17 Image editing software based filtering attacks: p_A **is low for local nonlinear filter based attacks**. Also, the shear filter used here performs local nonlinear distortions, and cannot be modeled as a global 2×2 matrix. 276

6.1 Strength of AWGN to add for a given JPEG QF 294
6.2 Table of Notations 310
6.3 Seam-carving detection accuracy for different training-testing combinations: "test"/"train" refers to the seam-carving percent for the testing/training images, "mixed" refers to that case where the dataset used for training/testing consists of images with varying seam-carving percentages (images with seam-carving percentages of 10%, 20%, 30% and 50% are uniformly distributed in the "mixed" set). 317
6.4 Seam insertion detection accuracy for different training-testing combinations: the meaning of "test", "train" and "mixed" are same as in Table 6.3. 318
6.5 Detection of scaling using Shi-324 feature and SVM models for different train-test combinations: "test" refers to the scale-factor for the positive examples in the test set, while "train" refers to the scale-factor for the positive examples in the training set. 339
6.6 Rotation detection using Shi-324 and trained SVM models: "test" refers to the (crop fraction + rotation angle combination) for the positive examples in the test set, while "train" refers to the combination for the positive examples in the training set. For example, (60%, 10°) refers to positive examples which are first rotated by 10° and then 60% of the image is retained per dimension. 340
6.7 Table of Notations used in connection with seam carving based object removal 344

Chapter 1

Introduction

There has been considerable interest in data hiding and watermarking in the last decade. The advent of the digital age, the proliferation of computers and the widespread usage of the Internet have created digital multimedia as a host for reliable and secure data transfer. Data embedding in digital multimedia finds use in applications ranging from digital watermarking, secret communications, copyright protection, to content authentication. In today's world, both secure/secret communication (through data hiding) and the detection of such communication (steganalysis) are of paramount importance. We first outline the concepts of data-hiding, watermarking and steganography before discussing our contributions in these fields. A list of useful acronyms that have been frequently used in the dissertation is presented in Table 1.1.

Table 1.1: Commonly used Acronyms in the Dissertation

Acronym	Full form
QIM	Quantization Index Modulation (a commonly used embedding method)
ME	Matrix Embedding (another embedding technique)
RA code	Repeat Accumulate code (a coding scheme for providing error resilience)
QF	quality factor (determines extent of JPEG compression)
ECC	Error Correction Coding (adds redundancy to data-bits to survive channel attacks)
PMF	Probability Mass Function (normalized histogram)
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
YASS	Yet Another Steganography Scheme (our proposed JPEG steganographic scheme)
WM	Watermark
SR	Statistical Restoration (a steganographic scheme robust to histogram-based detection)
BER	Bit Error Rate (our aim is to have zero BER for our hiding systems)
EMD	Earth Mover's Distance (a common measure to find distance between two distributions)
KP	key-point

1.1 Introduction to Data Hiding

Let us first consider a basic data hiding setup in Fig. 1.1. The sender plans to send a message m to the receiver across a noisy channel. In data-hiding, the bits that are being transferred are "piggy-bagged" on a signal so that what gets transferred across the channel is the composite signal with embedded data. Since we have considered images as our primary source medium, we will refer to the signal as an image. In practice, data-hiding techniques have been devised for videos, speech and audio signals. In the dissertation, we will denote the sender, receiver and detector (of whether data embedding has occurred for a given image) as "he".

Here, the host signal is x and after the message m is embedded, the resultant signal is called s. A question to be asked here is how is m actually embedded in x. We need to ensure that the distortion introduced after data hiding is not large enough so that the signal s still appears perceptually transparent even after the embedding. In Fig. 1.1, if $D(\cdot, \cdot)$ refers to some perceptual distortion measure, then the hiding function should be such that the distortion between the host signal x and the signal with embedded content s, $D(s, x) \leq D_1$, where D_1 is the maximum perceptual distortion that can be tolerated (beyond that, the embedding can cause perceptually detectable distortions).

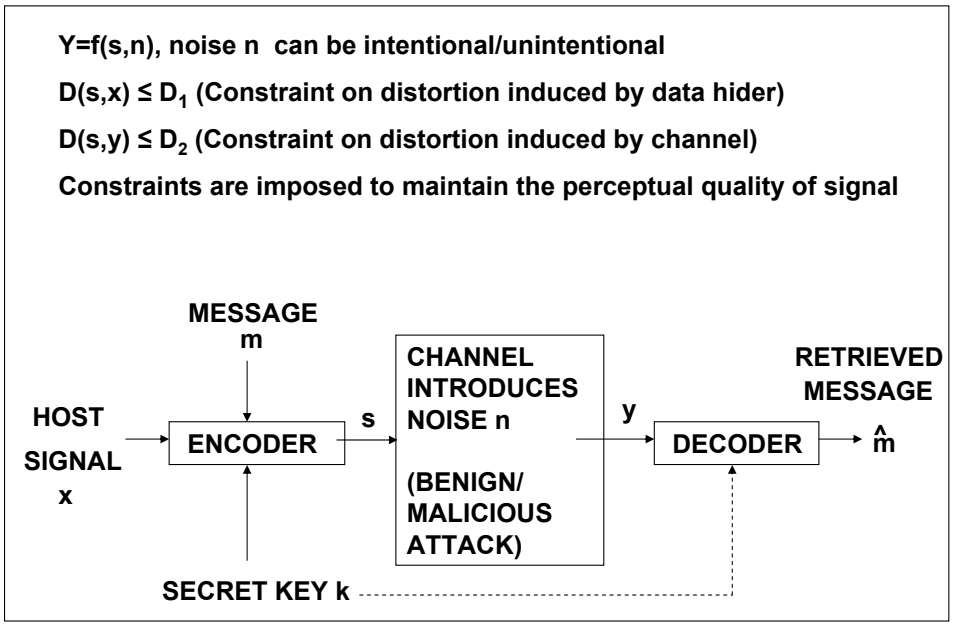


Figure 1.1: Block diagram of a data hiding system - the composite signal y after channel attacks is a function ($f(\cdot,\cdot)$) of the stego signal s and the channel noise n. $D(\cdot,\cdot)$ refers to a perceptual distance measure.

In data-hiding parlance, the original signal which is used to embed the data bits is referred to as the "cover signal" (x in this example). The composite signal, containing the embedded data, is called the "stego signal". After hiding, the stego signal s is transmitted over the channel. The channel can be a source of unintentional/benign or deliberate attacks. For example, a compression attack is a benign attack where a channel has to compress the input signal because of bandwidth constraints. A deliberate attack can be a tampering attack where we crop or locally blur some selected content in the stego image.

What is the trade-off involved here? If we make large enough changes to an image while embedding, it will be easier to recover the embedded data at the decoder side. Thus, the system will be more robust at the cost of increased perceptual distortions. Since the source of errors is the noise introduced by the channel, we can design the hiding system so that it is robust to the expected channel distortions. In the given example, we assume a constraint on the maximum noise introduced by the channel - $D(s, y) \leq D_2$, i.e. the system should be robust enough to allow data recovery after channel distortions where $D(s, y) \leq D_2$.

When is the data hiding system considered to be a success? Obviously, the end goal for the receiver is to have access to the same data bits which the sender has transmitted, i.e. the retrieved message (as in Fig. 1.1) $\hat{m} = m$, for this example.

Importance of Error Correction: For a practical hiding setup, we incorporate error resilience in our hiding scheme by transmitting a code-word having a higher number of bits than the original message. The redundancy factor involved in the codeword selection depends on the trade-off between the number of data bits we wish to transmit and the desired level of robustness. While Fig. 1.1 describes the schematic of a hiding scheme, Fig. 1.2 shows an end-to-end scheme. What are the design issues for such a scheme? Firstly, we need to come up with a good error correction scheme to address the trade-off between embedding rate and robustness. Secondly, we require a hiding scheme which achieves a proper

trade-off between embedding distortion and the recoverability of the embedded data.

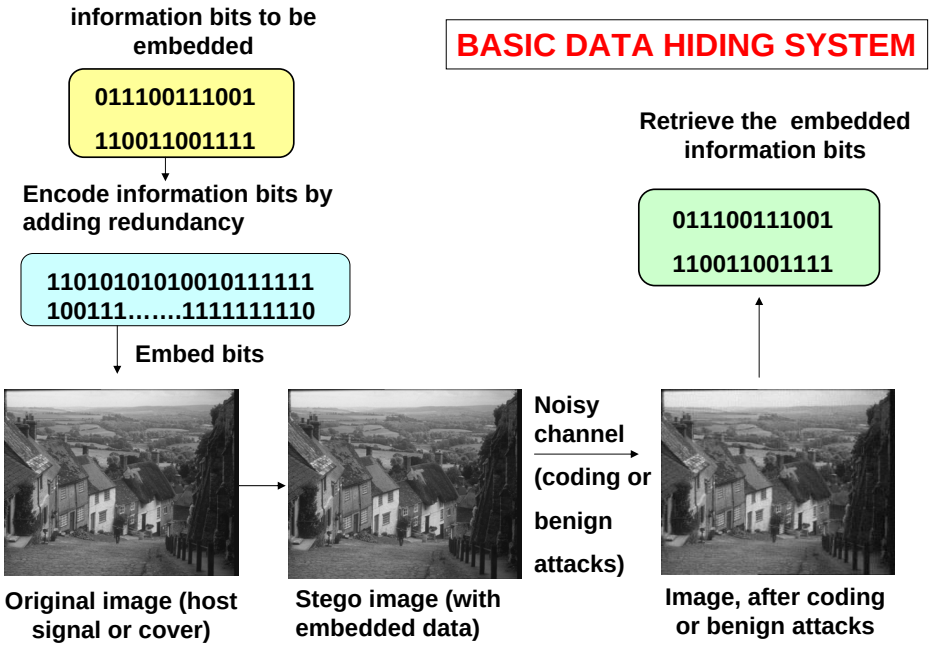


Figure 1.2: An end-to-end data hiding system which uses suitable error correction coding

Watermarking: Data-hiding and watermarking are related but different applications. Watermarking is mainly a detection problem where the receiver has to decide whether or not a certain watermark has been embedded in the given image. The data-hiding problem is mainly a decoding/communication problem where the receiver assumes that the sender is transmitting some data and his goal is to decode the embedded bits. The data-hiding problem is more difficult as there is no

reference sequence for comparison. For watermarking, there is a always a known reference sequence (the watermark) whose presence/absence is being verified.

1.1.1 Example of a Data Embedding System

The discussion so far has introduced the data hiding problem. We now describe "how" the embedding is done in practice. This example is based upon the "high volume data hiding in images" work performed by the data-hiding group at the Vision Research Lab (VRL) in UCSB [51, 105–107]. Since we will be consolidating upon the past data hiding work at VRL in the dissertation, we provide a brief introduction into the high volume hiding setup, as illustrated in Fig. 1.3. It is assumed that the most common attack for our hiding setup is the JPEG compression attack. The hiding method closely follows the steps used during JPEG compression before the embedding occurs. During compression, the raw image is modified in such a way that the perceptual distortion w.r.t. the original cannot be easily distinguished. Thus, there is a basic similarity between a data hiding scheme and a good compression method. For both, the image needs to be modified while maintaining perceptual transparency. Using the JPEG compression as a motivating factor, we divide the image into 8×8 blocks and perform hiding in the quantized discrete cosine transform (DCT) domain of individual blocks. In the JPEG framework, the coefficients obtained after block-wise DCT operation

are further divided element-by-element by a certain quantization matrix, which corresponds to the quality factor (QF) at which the output JPEG image is to be advertised/saved.

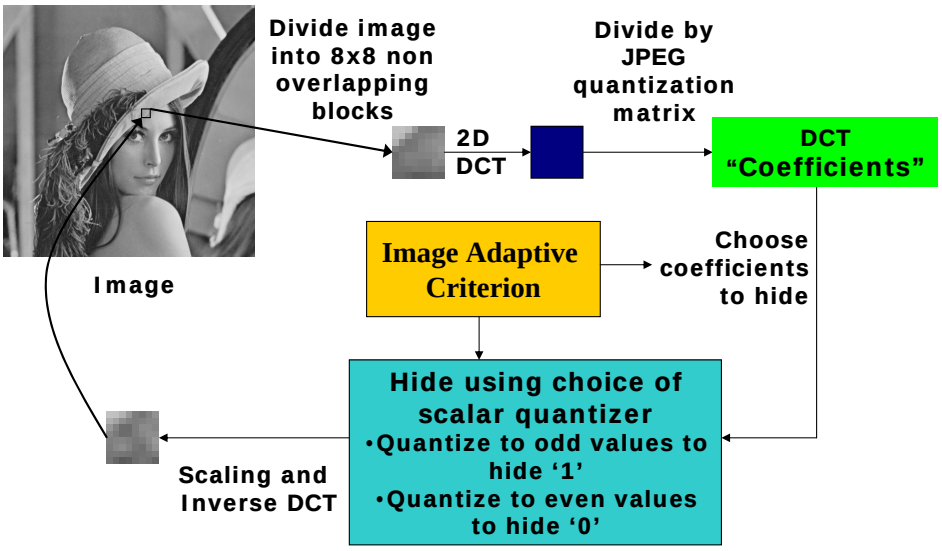


Figure 1.3: A step-by-step of the high volume image data hiding framework developed by researchers at the Vision Research Lab (VRL), UCSB (2001-05) - embedder side

QIM-based Hiding Scheme: For the data hiding system, we choose a design quality factor (QF_h) which decides the quantization matrix used to divide the transform domain coefficients obtained after block-wise DCT operation. We use quantization index modulation (QIM) [16] for embedding in the quantized DCT domain. In the QIM framework, there are two quantizers and a quantized DCT coefficient is mapped to the nearest codeword, depending on whether 0/1 is to be embedded. For example, if the codewords are located unit distance apart, we

can assume that $\{2m, m \in \mathbb{Z}\}$ correspond to one quantizer (to embed a 0) and $\{(2m+1), m \in \mathbb{Z}\}$ correspond to the other quantizer (to embed a 1).

Erasures: It has also been observed that for maintaining the perceptual quality of the image, image coefficients very close to zero should be left unmodified. Hence, quantized DCT coefficients in [-0.5, 0.5] are not used for hiding and are instead left unmodified. These coefficients get converted to zero after quantization using QF_h-based JPEG matrix and subsequent rounding. We refer to such cases where we end up not embedding the desired bits as erasures. An erasure is subsequently denoted as e.

Example of QIM-based hiding: Say, a DCT coefficient after division by the quantization matrix equals 3.7. If we wish to embed 0 (or 1) using this coefficient, it is modified to the nearest even (or odd) integer, i.e. 4 (or 3). On the other hand, if the coefficient is valued 0.37, it cannot be used for hiding. It is left unmodified at the embedding stage and it is converted to zero after rounding off the quantized DCT coefficients. Just as in JPEG, the inverse DCT step produces the 8×8 pixel blocks after the data embedding.

We have experimentally observed that for a proper choice of the frequency band, perturbing the quantized DCT coefficients still maintains the perceptual transparency, the change being in the range [-1,1]. Changing the DC term produces significant perceptual distortion but modifying the AC DCT coefficients in

a low and mid frequency band by small amounts ($|\text{change}| \leq 1$) does not.

Trade-offs involved in the embedding scheme

1. If we choose a higher separation margin (Δ) between two nearest codewords (instead of 1 as in the odd-even based QIM-scheme mentioned above), we will obtain more robust embedding at the cost of increased perceptual distortion.

2. When the stego image is subjected to JPEG compression while being transmitted, let QF_a denote the corresponding JPEG quality factor at which it is compressed. If we choose a lower QF_h for hiding while keeping QF_a fixed, the system will be more robust than if a higher QF_h is used. For a higher QF_h, the quantization matrix consists of elements closer to one and there is finer quantization as the elements are divided by smaller valued matrix coefficients. If the elements are first subjected to a quantization based on QF_h, it is less likely that there will be decoding errors (for the same QF_a) for a coarser first quantization. A thing to note here is that for a low enough QF_h, the erasure rate will be high enough leading to fewer coefficients actually available for hiding. We have experimented with $QF_a = 75$, and after varying QF_h from 30-70 in steps of 10, the maximum hiding rate is found to correspond to $QF_h = 50$. The erasure rate progressively increases as we

decrease QF_h from 70 to 30. The robustness to hiding also increases as QF_h decreases from 70 to 30. Due to the competing tendencies (erasure rate increases $\Rightarrow$ less coefficients available for hiding, but robustness also increases, as QF_h decreases from 70 to 30), the maximum hiding rate occurs at QF_h = 50, a value in between 30 and 70.

3. When the hiding band consists of the first few non-zero AC DCT coefficients encountered during zigzag scan, they are more likely to be of higher magnitude (and hence outside the erasure zone) as compared to the DCT elements which occur as we progress further along the zigzag scan process. Let λ denote the size of the embedding band in a 8×8 block. If we take a larger sized embedding band (higher λ), the effective embedding rate often does not increase by the same ratio as the newer coefficients that are encountered often correspond to erasures.

1.1.2 Decoding Process

We assume that the hiding band and the quantization matrix used for hiding (corresponding to QF_h) are known also to the decoder. For successful decoding, we have used an iterative decoding scheme where the soft decision values at all the coefficient locations have to be properly initialized for successful decoding. The steps involved at the decoder side are as follows. The image is considered

as a collection of 8×8 blocks and after taking block-wise DCT, the coefficients are divided by the JPEG quantization matrix corresponding to QF_h. Since the encoder and decoder share knowledge about λ and QF_h, and we assume only global attacks in the channel, the decoder has access to the same coefficients which were used by the embedder for hiding. For every location, it assumes an erasure if the image coefficient rounds off to zero; else it guesses whether 0/1 was embedded based on whether the corresponding transform domain image coefficient rounds off to an even/odd integer (assuming Δ=1). These guessed estimates serve as the initial soft decisions, called log-likelihood ratios or LLRs (explained in detail later in Section 4.4), for successful iterative decoding.

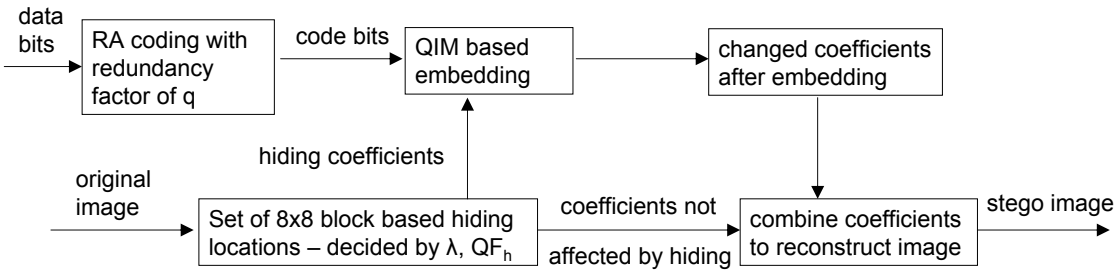


Figure 1.4: The RA code based framework is used to determine the code bits for a given sequence of data bits. The code bits get embedded at the selected image locations using QIM.

A portion of the image coefficients is used for hiding as shown in Fig. 1.4. The coding and embedding procedures work as follows: let the image dimensions be $M \times N$. The number of 8×8 blocks we obtain from this image equals $N_B = \lfloor \frac{M}{8} \rfloor \lfloor \frac{N}{8} \rfloor$. Using λ coefficients for hiding per 8×8 block, we end up with $L = \lambda N_B$

embeddable coefficients. The codeword $\mathbf{c}$ can have, at the most, L bits. The message sequence $\mathbf{m}$ can be accommodated while using a redundancy factor of q for the error correction coding if the number of message bits does not exceed $\lfloor \frac{L}{q} \rfloor$. An error correction coding (ECC) framework receives $\mathbf{m}$ and q as inputs, and outputs the code word $\mathbf{c}$ of length $|\mathbf{m}|q$, where $|\mathbf{m}|$ denotes the number of bits in the message $\mathbf{m}$.

Once the decoder computes the initial soft decisions based on whether the coefficient corresponds to $0/1/e$, the iterative decoding proceeds and if the decoding converges successfully, it should output the original $\mathbf{m}$. Repeat accumulate (RA) code is the ECC [25,59] used by our hiding framework since the data hiding channel involves a higher fraction of erasures. Besides being simple to implement, RA code has also been shown to be near-optimal for high erasure-rate channels [107]. Our hiding framework, based on hiding in DCT coefficients and avoiding embedding in [-0.5, 0.5], also has a high rate of erasures.

1.2 Steganography and Secure Communication

An extension of image-based data hiding is to use it for secure communication through "**steganography**". Steganography is the science of communicating in a manner such that the existence of the communication is not detectable by an ex-

ternal entity. The method of detecting the existence of the secret communication is "**steganalysis**". The steganalyst uses a group of features to detect the presence of hidden data. One can view the classification problem as a two class one - where the steganalyst has to distinguish between "cover" (original signal without embedded data) and "stego" (signal with embedded data). The data hider (steganographer) knows that the steganalyst can use a suitable set of features for detection and he has to modify his hiding scheme such that the hiding is not only imperceptible but also statistically undetectable. *Thus, the steganographer and steganalyst are like two eternal competitors - one always tries to defeat the other.*

Fig. 1.5 introduces the steganography problem. Maintaining perceptual transparency is the first requirement for a proper hiding scheme. Once that is achieved, the next question is whether an external agent, who can snoop into the network, can detect whether any secret communication is taking place. For example, in Fig. 1.1, when the stego signal s is transferred over the channel, it can be easily detected by someone having access to the channel. It is however highly non-trivial to detect whether there is some embedded information in the transmitted signal, i.e. to decide whether the signal s corresponds to a cover or a stego. The steganographer's task is to make this task difficult for the person having access to the channel, the steganalyst. From the steganalyst's perspective, he has to decide, based on a single received image, whether the image has embedded content.

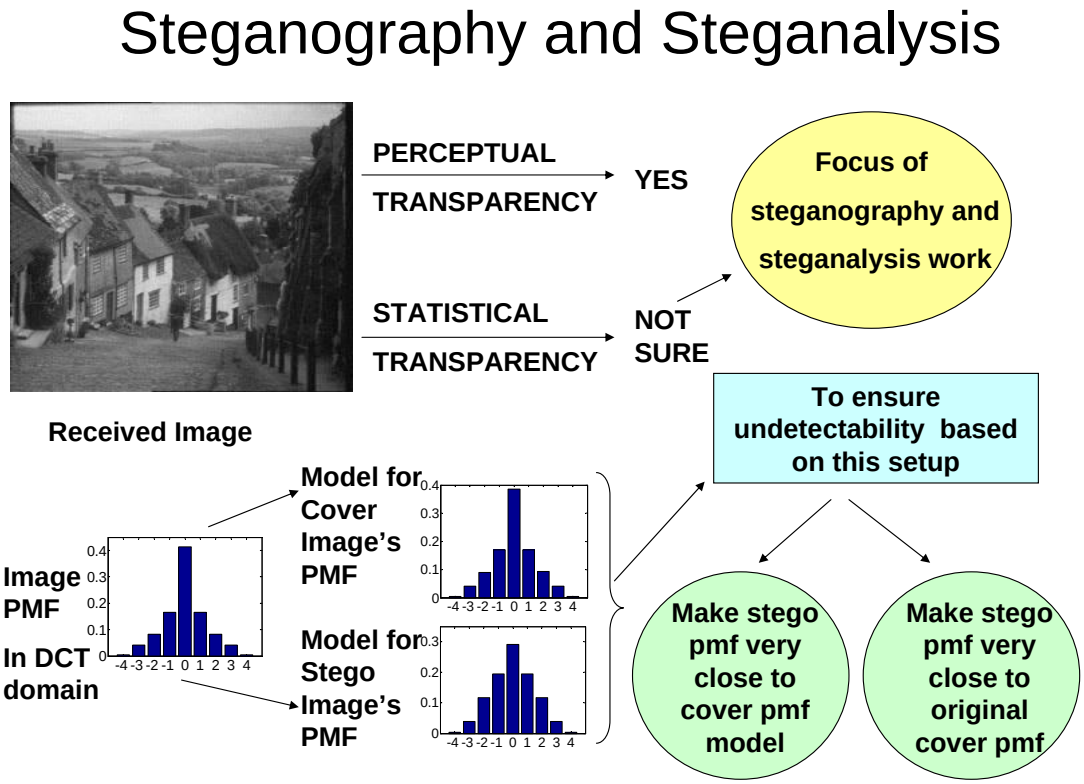


Figure 1.5: The steganography problem

To allow a fair competition between the steganographer and the steganalyst, the steganographer makes his hiding method public. He reveals the details of the embedding method but does not reveal the secret key(s), if one (or more) keys are used. The steganalyst uses a machine learning approach to identify image features which can help in distinguishing cover from stego images, assuming that the stego images are generated by a certain embedding method.

The steganalyst uses the embedding method that he expects the steganographer to have used to embed in half of the images and the other half of images is

left untouched. The challenge now is to discover image features which consistently vary between the two image classes (cover and stego). The intuition here is that if the same method is used to modify image coefficients to embed data, it will cause similar statistical changes in images. Classification techniques like support vector machines (SVM) can be used to verify the utility of a given feature set for solving this two-class problem. For example, in Fig. 1.5, the distribution of the image DCT coefficients is used to distinguish between cover and stego images.

If it is known apriori which feature shall be used for steganalysis, then appropriate changes can be made in the image to ensure that the concerned steganalysis feature remains unchanged. In Fig. 1.5, the DCT domain distribution is used for steganalysis - hence, if the steganographer performs hiding such that the first order distribution remains unchanged, steganalysis based on this first order distribution will fail. Of course, *the big assumption here is that the steganographer knows what features will be used for steganalysis.*

Thus, we see that there are two main approaches for steganography:

- Model-based approach: the steganographer knows that the steganalyst has developed models by using a training based method for cover and stego images (assuming the hiding method is known to the steganalyst). A secure hiding method should resist detection even if the steganalyst has access to

the hiding method - i.e. the hiding method should be such that the steganalyst cannot obtain reliable cover and stego models.

- Statistical Restoration: if the steganographer knows which feature is to be used for steganalysis, he can use a fraction of the available image coefficients for hiding and the rest are used to compensate for the change in the statistical feature due to hiding.

Having introduced the concepts of data hiding, steganography and steganalysis, we present the thesis outline, discuss the individual chapters and briefly introduce the corresponding contributions.

1.3 Thesis Outline

Statistical Restoration based Steganography: In Chapter 2, we consider the problem of statistical restoration (SR). The previous SR-based approach [109, 110, 112] was able to restore only first order statistics. We have generalized the statistical restoration framework to resist detection against higher order distributions and also present detailed analysis for capacity computation for known channels with known attack levels.

1. The SR framework has been extended to restore higher order statistics. We have found a correspondence between the Earth Mover's Distance (EMD)

[92] and the statistical restoration method, which we utilize for higher order histogram matching. We also propose a joint compensation method to account for higher order statistics in the scenario where overlapping pairs of coefficients are considered and where the EMD-based formulation cannot be applied. This work is described in Section 2.2 and also in [98].

2. We also find the maximum fraction of image coefficients that can be used for hiding, for a certain order of statistics to be restored, so that the rest of the available coefficients can compensate for the changes introduced by hiding. This work is discussed in Section 2.3 (for first order statistics), Section 2.4 (for higher order statistics), and in [96].

3. We perform a complete analysis of the capacity estimation of the SR framework. For example, if the first order distribution has to be modified, then the total number of data-bits that can be embedded, while maintaining perceptual, statistical and attack-related constraints, is computed. The capacity estimation work is elaborated upon in Section 2.6 and [100].

Randomized Block-based Hiding for Improved Security: In Chapter 3, we consider the scenario where the steganographic framework is secure against a variety of steganalysis features, and not against a pre-decided feature as is the case for SR. Since most steganographic methods consider JPEG images as input

and output, we focus on JPEG steganography. For JPEG stego images, it has been shown by Fridrich et al. [82, 83] that the statistics of the cover image can be estimated by cropping - it is called the self calibration method. Most successful detection methods are based on the self-calibration method as it can provide very accurate models for the original image statistics, while having access to only the stego image. To resist the steganalyst from obtaining very accurate image models from the stego images, we propose hiding in randomized block locations, instead of hiding using the regular grid of 8×8 blocks. We call our hiding framework "Yet Another Steganographic Scheme" (YASS). This method was originally proposed in [108] and is described in Section 3.3. Further variants of YASS to increase the hiding rate without significantly affecting the detectability have been described in Section 3.7 and in [99].

To further enhance the trade-off of detectability vs embedding rate in YASS, we optimally vary the redundancy factor for a given image so that the used redundancy for an ECC framework is just sufficient for data recovery for a given image, and we can obtain the maximum data-rate for a given image, and for a given attack channel with known characteristics. The method is applicable for any hiding scheme which performs QIM-based hiding and RA-code based error correction. This is elaborated upon in Section 3.5 and in [97].

Use of Matrix Embedding for more secure hiding: Chapter 4 discusses matrix embedding [20] and how it can be used in conjunction with RA-code based error correction codes for secure steganography. We have explored the use of matrix embedding (ME) to reduce the detectability of our steganographic scheme, YASS. ME is an embedding method with a *higher embedding efficiency*, compared to commonly used techniques such as quantization index modulation. The embedding efficiency is defined as the average number of embedding changes (number of image coefficients which are modified - AC DCT coefficients in our example) involved in hiding one bit. For example, in the QIM scheme, the average embedding change is 1/2 to embed one bit. If 0 is to be embedded and the coefficient rounds off to an even integer, it need not be modified at all. However, if 1 is to be embedded, the coefficient needs to be modified to the nearest odd integer. Thus, the effective embedding efficiency is 1/2 for QIM.

An example of (7,3) matrix embedding is modifying, at the most, one of 7 coefficients to embed 3 bits. Thus, the effective embedding efficiency is 1/3 for (7,3) ME. In Section 4.1, we explain (7,3) ME using actual examples.

Matrix embedding has been used previously for data hiding applications. However, it has generally been used for *passive* steganography (we assume that there are no significant channel attacks, i.e. channel noise $n = 0$ in Fig. 1.1) and not for *active* steganography (when there are channel attacks, i.e. channel noise $|n| > 0$).

ME has a higher embedding efficiency than QIM but it also has reduced robustness to attacks. We have used RA-code based error correction coding to generate the code bits, and used ME to embed these code bits. Since ME involves a many-to-one mapping (multiple coefficients determine the bit embedded at a single bit location), computing the initial soft decision values for RA decoding remains a challenge. Why is proper initialization of the RA decoder important? With suitable initial confidence values, the iterative decoder converges at a lower redundancy factor, thus ensuring a higher data-rate for the same noise channel. In Section 4.4, we discuss this issue in detail. Though we present our analysis for the YASS scheme and for RA-based coding, the methods proposed are generic enough to be applicable to steganographic schemes and other iterative decoding based ECC schemes. Chapter 4 is devoted entirely to the use of matrix embedding for robust and secure steganography. The relevant paper which describes our matrix embedding work is [95].

Robust key-point based hiding: The methods presented so far have considered only global attacks. However, what happens if the image is subjected to rotation, scale, translations, cropping and other local attacks? A different strategy needs to be used to ensure that the data can still be recovered even after these attacks. Chapter 5 highlights our work in robust key-point based data hiding where we demonstrate robustness against local and geometric attacks. To resist

cropping attacks, the data bits need to be repeatedly embedded in local regions, spread throughout the image. To resist geometric attacks, we insert peaks in the frequency domain such that the geometric transformation which the stego image has undergone can be estimated, and then compensated for. To enable the decoder to consider the same regions as used by the encoder for data recovery, we use robust key-points to determine the local regions which will be used for hiding. Robustness to JPEG compression attacks is also discussed in detail as it is mainly responsible for improper geometric alignment. JPEG compression introduces spurious frequency-domain peaks due to the induced periodicity which lead to wrong peaks being detected as the synchronization peaks.

Image Forensics and Tamper Detection: Chapter 6 describes our work in image tampering detection and localization. While data hiding and steganography are important for providing a secure communication channel, the other aspect about information reliability is to verify the authenticity of a given image. Just as verifying whether an image intercepted in the channel contains some hidden information (i.e. if it is a stego image) is of interest for detecting secure communication, another valid question is to find whether the image is an original or it has been subjected to some sort of tampering. Thus, the question we are answering is whether we can "trust" in the contents of the image. Image forensics is the science of verifying the authenticity of the image. It is of great research interest

because image tampering is very easy to execute, owing to the proliferation of image editing software. The link between our steganography work and the forensics work is that there are steganalysis features which can be used not only to detect cover from stego, but also to detect tampered from un-tampered images. In brief, steganalysis features are used to solve a more general problem, i.e. both cover/stego and tampered/untampered classification.

The specific contributions that we have made to the field of image forensics are as follows:

1. Most forensics methods for re-sampling detection are not robust to compression attacks, as they depend on the geometric relationship that exists between corresponding pixels in the original and tampered images. We propose a method to decrease the effect of the JPEG compression artifacts while the traces of re-sampling are still retained. This is described in Section 6.3 and in [74].

2. Seam carving is a recently proposed technique for content-aware image resizing. We use seam-carving for image tampering, through image resizing and object removal. The aim is not only to detect whether tampering has occurred (through seam carving and seam insertions) but also to localize

where the tampering has occurred. This is covered in Sections 6.4 and 6.5 and is also described in [101].

Conclusions: In Chapter 7, we summarize our contributions and outline various avenues for future research as logical extensions of the different research problems that we have considered.

1.4 Overview of Contributions and Impact

In the earlier sections, we have introduced the data hiding problem and have outlined our contributions to secure communication (steganography) and robust hiding. Also, contributions in the field of image forensics have been proposed. Here, we summarize the contributions and also discuss possible areas of impact.

For secure communication, specially in military zones where all messages are more likely to be intercepted, steganography becomes important to guarantee the security of the transmitted messages. To guarantee such security, the message either needs to be encrypted or it needs to be imperceptibly embedded in a host medium (steganography). The encryption step does not ensure that the communication remains undetectable - it merely ensures that an adversary cannot decode or interpret our messages. Steganography solves the "detectability" problem as it strives to be both perceptually and statistically undetectable. Also, the

embedding and decoding procedures are much less computationally involved for a steganographic framework than for a cryptographic system. The emphasis of present day steganography is security over data rate. At lower data rates, the security provided by our YASS framework is higher than that provided by other state-of-the-art methods. Using matrix embedding (ME), we further decrease the detectability of YASS. The chief contribution in incorporating ME is using it with an error correction framework to make it robust against real-world channels such as the JPEG compression channel. Active steganography is important for practical applications. In a real-world channel, it may be that the message itself is undetectable but the noise introduced by the actual channel (for lower bit rate as is the case where the attack is a compression attack) is such that error resilience is required. Our RA-coding framework provides the necessary error robustness. The security is provided by pseudo-random selection of hiding blocks and by the matrix embedding framework.

There are applications when robustness becomes more important than security. Consider a scenario where an adversary cannot block the communication but can modify the transmitted signal. In such cases, the attacker may not know where the message is embedded but he may modify the image such that the image still retains its visual content but data recovery becomes possible. For example, slight cropping or geometric transformations will render entire image-based hiding

methods ineffective but the visual content is only slightly affected. With the advent of image processing software, it is easy to modify an image such that the modified image looks natural (*traces of image re-touching are not visually obvious*) but these changes often render data recovery impossible from the local regions which have been modified. Hence, there is a need to focus on localized hiding than global embedding. The proposed method is robust to affine global geometrical transformations and cropping. The key to successful data recovery is that there should be proper geometric alignment so that both the encoder and decoder have access to the same image grid. Since our alignment method is based on frequency domain peak based synchronization, a common problem is the effect of JPEG compression on the frequency domain matching. We have exploited the characteristics traces left by JPEG and used it to distinguish between the actual synchronization peaks and the JPEG induced peaks.

The final contribution is in the field of image forensics. In today's information age, the reliability on images as an information medium is high, thus prompting the question as to what if the source itself is not authentic. With easy access to image editing software, manipulating an image to give it a new meaning is not a laborious task. Forensic analysis of an image tests for the presence of traces introduced by a single (or more) tampering methods on the image. Forensics, as a field of active research, is making rapid strides and our contributions in robust

re-sampling detection can help other forensic techniques which work for uncompressed images but not for JPEG compressed images. Seam-carving has been primarily used for image re-sizing but we have shown that it can be easily incorporated in an image tampering (and object removal) framework. Though we have shown the utility of steganalysis features to detect seam carving and seam insertion based tampering, it would be worthwhile to study the generalizability of the steganalysis features for detecting other image tampering methods. Localization of tampering is also important as it identifies which parts of the image one can trust and which parts one cannot.

Chapter 2

Statistical Restoration Based Steganography

The early approaches for detecting data embedding used first order statistics, namely the histogram, based on the transform domain coefficients which were used for hiding. To counter such approaches, the statistical restoration framework was proposed which ensures that the histogram, the feature used for steganalysis, remains unchanged after hiding. The main idea behind statistical restoration is as follows - if the feature to be used for steganalysis were explicitly known beforehand, a portion of the coefficients available for hiding can be used for data embedding while the remaining coefficients can be suitably modified to ensure that the statistical feature used for steganalysis remains unchanged. Previous

work based on statistical restoration yielded promising results against first-order histogram based steganalysis. In this chapter, we extend the statistical restoration framework to resist detection against higher order features, such as second order co-occurrence statistics. Also, we determine the maximum fraction of hiding coefficients that can be used for embedding which ensures that properly changing the remaining coefficients can fully restore the detection statistic. Finally, for an active steganographic setting with channel distortions, we take the error correction capability of our coding system into account to obtain an estimate of the effective data-rate for statistical restoration.

These enhancements to the statistical restoration method have been elaborated upon in this chapter. In Section 2.1, we briefly describe the statistical restoration framework and provide references to previous work. In Section 2.2, an Earth Mover's distance (EMD) based method is proposed to determine the transportation flows between histogram bins for statistical restoration. The transportation flow computation method holds for any arbitrary order histogram or co-occurrence matrix based statistic. For the EMD-based method, results are presented for restoration of the second order statistics, under the constraint that non-overlapping pairs of coefficients are used to create the histogram. For further generalization, we propose a joint compensation scheme which accounts for both

intra and inter-block (block = "8×8 block" in pixel domain) based correlations, and also accounts for overlapping pairs.

In Section 2.3, the optimum hiding fraction is determined for the statistical restoration framework for the first order statistic. Section 2.4 extends the analysis for higher order statistic. In Section 2.5, total compensation (the histogram statistic consisting of all the frequency coefficients in a certain hiding band is restored) is compared with individual compensation (the individual histograms corresponding to the frequency band are separately restored) and it is observed that some frequency coefficients perform better for steganalysis than others.

A technique for capacity estimation for the statistical restoration framework is presented in Section 2.6, where the perceptual, statistical and attack based constraints are accounted for. We use our estimation of the optimum hiding fraction to determine the partitioning between hiding and compensating coefficients. A repeat accumulate code based error correction framework is used. For the error correction scheme, the minimum possible redundancy is used that ensures perfect data recovery and thus, the maximum possible data-rate is obtained for a given image and an attack channel.

2.1 Statistical Restoration (SR)

In this section, we explain how the SR framework operates. Let the input feature set available for hiding be X. We divide it into two disjoint sets - H for hiding and C for compensation, as in (2.1).

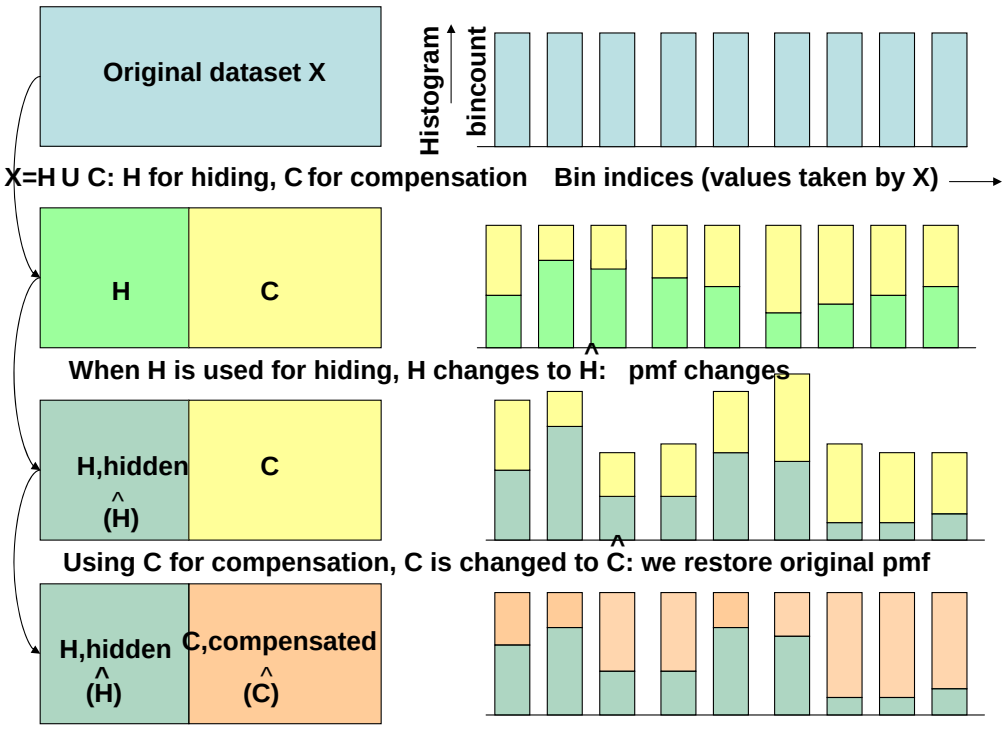


Figure 2.1: Explaining the 1-D statistical restoration framework (pmf = Probability Mass Function = Normalized Histogram)

Fig. 2.1 shows how the 1-D statistical restoration proceeds. Given a dataset X, we hide using H, a subset of X. After hiding, H is changed to $\hat{H}$. To compensate for the change in the overall distribution of $(\hat{H} \cup C)$ due to changing H to $\hat{H}$, we

have to change C accordingly to $\hat{C}$ to ensure that the distribution of $(H \cup C)$ and $(\hat{H} \cup \hat{C})$ are the same. For the statistics to be perfectly restored, we should know beforehand what is the exact distribution we are trying to maintain. In Fig. 2.1, that distribution is the first-order distribution. What happens if the steganalyst uses higher-order statistics instead of 1-D statistics? The steganographer has to keep on restoring whatever statistics the steganalyst uses. Though Fig. 2.1 shows the restoration of the 1-D histogram, the basic idea holds for the restoration of 2-D and higher order statistics, i.e. we hide in a fraction of the dataset and use the other fraction to compensate for the statistical change caused by the hiding. For example, if the histogram has 10 bins in 1-D case, the 2-D and 3-D co-occurrence matrices (comprising of non-overlapping tuples of 2 and 3 elements) will have 10^2 and 10^3 bins, respectively. For 1-D SR, the 10-bin histogram for C needs to be modified to the 10-bin histogram for $\hat{C}$. Similarly, for 2-D and 3-D SR, the compensation needs to be done over a larger number of bins (100 and 1000, respectively). If a solution is developed for histogram matching for a given number of bins, that method can be applied for 10, 100 and 1000 bins. Keeping this assumption in mind (the steganographer knows what order distribution is used for steganalysis), any order statistic can be restored, in general, using the statistical restoration framework.

2.1.1 Literature Survey

There have been several approaches in the past that attempt to restore the first-order statistics so as to resist histogram-based steganalysis. These include Provos' OutGuess algorithm [89], Eggers et al.ś histogram-preserving data-mapping [27], Franz's suggestion [33] of hiding in independent pairs of values, Guillon's idea [45] of companding to a uniform distribution prior to quantization-based hiding, and Wang and Moulin's stochastic quantization index modulation (QIM) [120]. Some of these approaches are limited by their inability to handle continuous host data, while others cannot achieve exact host probability mass function (PMF) when communicating at high rates. Many of these schemes are also fragile against any noise or attacks. In [109,110,112], the proposed statistical restoration scheme addresses some of these deficiencies. It was shown that one could achieve zero Kullback-Leibler (K-L) divergence between the host and the stego PMFs while hiding at high rates. The scheme was also robust against distortion-constrained attacks.

In previous SR-based work [109], statistical restoration was used to achieve exact matching of first order DCT histograms. The problem of modifying the histogram of the compensation coefficients was posed as a Minimum Mean Square Error (MMSE) problem. This MMSE problem signifies that the histograms of C and $\hat{C}$ can be matched while making minimum changes to C (in a MSE sense). The

MMSE problem was solved by Mese et al. [72] using integer linear programming. A simpler solution to this problem was provided by Tzschoppe et al. [115]. They showed that for MMSE mapping, all the bins in the target histogram should be filled in an increasing order by mapping the input data with values in increasing order.

2.2 Second Order Statistical Restoration

In the statistical restoration framework, the set of host symbols X is divided into two disjoint sets: H for hiding and C for compensation. Data is embedded using the hiding function f_1 into the hiding set H to get $\hat{H}$, as shown in (2.1). We divide the host symbol set X into 2-D bins and find their respective bin-counts (number of terms per bin). We use $B_X(i,j)$ to denote the bin-count of the $(i,j)^{th}$ bin of X. Since the normalized bin-count gives the PMF, compensating for the bin-counts is equivalent to restoring the PMF. The aim is to find the function f_2 that modifies C to $\hat{C}$ such that the 2-D PMF $P_Y(Y)$ (Y having been defined in (2.1)), obtained after hiding and compensation, is same as that of X. To evade second order steganalysis, a lower hiding rate (computation of the optimum hiding

fraction is discussed later in Section 2.3) is employed compared to the 1-D case.

$$X = H \cup C,\ \hat{H} = f_1(H),\ \hat{C} = f_2(C),\ Y = \hat{H} \cup \hat{C} \tag{2.1}$$

$$H \cap C = \phi \Rightarrow B_X(i,j) = B_H(i,j) + B_C(i,j),\ \forall (i,j) \tag{2.2}$$

$$\hat{H} \cap \hat{C} = \phi \Rightarrow B_Y(i,j) = B_{\hat{H}}(i,j) + B_{\hat{C}}(i,j),\ \forall (i,j) \tag{2.3}$$

$$\text{To obtain } P_Y = P_X,\ \text{we need } B_Y(i,j) = B_X(i,j),\ \forall (i,j) \tag{2.4}$$

$$\Rightarrow B_{\hat{C}}(i,j) = B_C(i,j) + B_H(i,j) - B_{\hat{H}}(i,j),\ \forall (i,j) \tag{2.5}$$

The functions f_1 and f_2 are obtained based on the perceptual and statistical constraints. The perceptual distortion between the original and stego images in the transform domain should be as low as possible to maintain perceptual transparency. The statistical constraint is that the statistical feature used for steganalysis (2-D PMF in this example) should be as well-matched as possible. Quantitatively, we wish to ensure $P_Y = P_X$ while incurring the lowest MSE between C and $\hat{C}$ (the modification of the elements in H to obtain $\hat{H}$ depends on the embedding method used and the goal of the SR framework is to make the minimum changes to elements in C to achieve perfect statistical match).

2.2.1 Earth Mover's Distance for Statistical Restoration

The EMD [92] between two PMF's is defined as the minimum "work" done in converting one PMF to the other. Here, *work* refers to the redistribution

of weights among the various bins in the discrete distribution. EMD returns the optimal transportation flows among the bins. For our statistical restoration problem, we have to convert a 2-D histogram B_C to $B_{\hat{C}}$, according to (2.5), where the normalized histogram is the PMF. Thus, by definition, EMD provides the optimum way of redistributing weights in B_C to obtain $B_{\hat{C}}$.

The problem of PMF compensation, as in (2.4), while simultaneously minimizing the perturbations required to convert C to $\hat{C}$, as in (2.5) can be connected with finding the EMD [92] and hence, the transportation flow between the two PMFs. Solving for the EMD between two PMFs shows how with the "minimum work" done, one can redistribute the weights in one PMF to make it resemble the other.

Let S and T denote two 2-D signatures, each having M clusters. The weight of each cluster is the fraction of points it contains. Let the center for the k^{th} $(k = (i, j))$ cluster of S be $\{s_i, s_j\}$ while the ℓ^{th} $(\ell = (m, n))$ cluster center of T is denoted by $\{t_m, t_n\}$. The square Euclidean distance between the k^{th} cluster center of S and the ℓ^{th} cluster center of T is called $d_{k\ell}$.

$$d_{k\ell} = (s_i - t_m)^2 + (s_j - t_n)^2,\ k = (i, j), \ell = (m, n) \tag{2.6}$$

The EMD problem is "optimally" changing S (considered as the source distribution) to make it as similar as possible to T (the target distribution). For our problem, the source S is the PMF P_C of the compensation coefficients while the

target T is $P_{\hat{C}}$, the PMF of $\hat{C}$. The weight of each bin is the PMF value for that bin. Our aim is to find a flow matrix $F = [f_{k\ell}]$, where $f_{k\ell}$ is the flow from the k^{th} bin of S to the ℓ^{th} bin of T that minimizes the total *work* done. The work done in modifying a source distribution S to a target distribution T using the transportation flow F is:

$$WORK(S, T, F) = \sum_{k=1}^{M} \sum_{\ell=1}^{M} d_{k\ell} f_{k\ell} \tag{2.7}$$

Thus, the *work* done for PMF-matching can be directly related with the total perturbations needed to convert C to $\hat{C}$.

EMD gives precisely the optimum flows from the bins of C to $\hat{C}$ that match P_X to P_Y, where X and Y are defined in (2.1), under the minimum mean-squared error (MMSE) criterion. The above formulation can be similarly generalized to the n-D case. If n-tuples constitute a histogram bin and each element can be quantized to one of m levels, then there are m^n possible bins for the n-D case. The EMD-based flows can be computed for these m^n bins and hence, the higher order statistics can be matched. However, there are issues arising from computational complexity and overlapping elements for n-tuple bins (where $n > 1$) as mentioned later in Sec. 2.2.3.

The actual embedding (modifying elements in H to the corresponding elements in $\hat{H}$ to embed the code bits) takes place using an odd-even based hiding framework (Section 2.2.2), a variant of QIM.

2.2.2 Odd-Even Based Hiding Framework

The luminance part of the image is used for hiding. We divide the luminance image into 8×8 blocks, perform block-wise DCT, divide element-wise by a certain quality factor matrix and then select a certain frequency band for hiding. The DCT coefficients thus selected are rounded off to produce the quantized DCT (QDCT) based dataset X. For hiding, we use odd/even embedding (a simple version of QIM) to convert the terms to their nearest odd or even integer, depending on whether the input bit is 1 or 0, respectively. Suppose, a QDCT term is 4 and we wish to embed 0 - then the QDCT term gets mapped to the nearest even number, which is 4. For embedding 1, we use a dither sequence, with numbers in the range [-0.5,0.5] which are produced by a pseudo-random generator, to decide whether to map 4 to 3 or 5.

$$\text{To embed } 1 \rightarrow q = \text{round}(p + 1 - \text{mod}(p - \delta, 2)), \tag{2.8}$$

$$\text{to embed } 0 \rightarrow q = \text{round}(p + 1 - \text{mod}(p + 1 - \delta, 2)) \tag{2.9}$$

where p, the original QDCT term, is mapped to q, δ denotes the corresponding number obtained from the dither sequence, "mod(p,2)" is the remainder obtained after dividing p by 2 and "round" denotes the rounding off operation. If p is an even(odd) number and 1(0) is to be embedded, it is mapped to $(p-1)$ or $(p+1)$ depending on whether δ belongs to the range (0,0.5] or [-0.5,0], respectively.

2.2.3 EMD Formulation for Image Steganography

EMD-based statistical compensation can be used for restoring higher order statistics for image steganography. The DCT is computed for 8×8 blocks in the image, and the block-wise DCT coefficients are then divided by a JPEG quality factor matrix (corresponding to QF_h, the QF used for hiding) and data is embedded in the quantized DCT coefficients using odd-even embedding [109]. The aim is to compensate for the intra-block based 2-D statistics in the quantized DCT domain. Some important issues regarding the implementation of 2-D EMD based statistical restoration are now discussed.

• *Overlapping Pairs:* For accurate 2-D PMF estimation, all possible consecutive pairs of block-based quantized DCT coefficients in the hiding band need to be considered. However, if overlapping pairs are used, then, when considering a flow from one bin to another, the bin-counts in adjacent bins are inadvertently modified. The remedy is to consider **non-overlapping pairs** of DCT coefficients per block, as shown in Fig. 2.2. If both the row-wise and column-wise pairs are considered, the redistribution of weights for PMF matching along both the vertical and horizontal directions creates ambiguous flows, as perturbation to a horizontal pair also changes bin-counts along the vertical direction and vice-versa.

• *Computational Complexity:* Let the number of bins be N in the 1-D case. By considering all possible pairs in the 2-D case, the total number of resultant

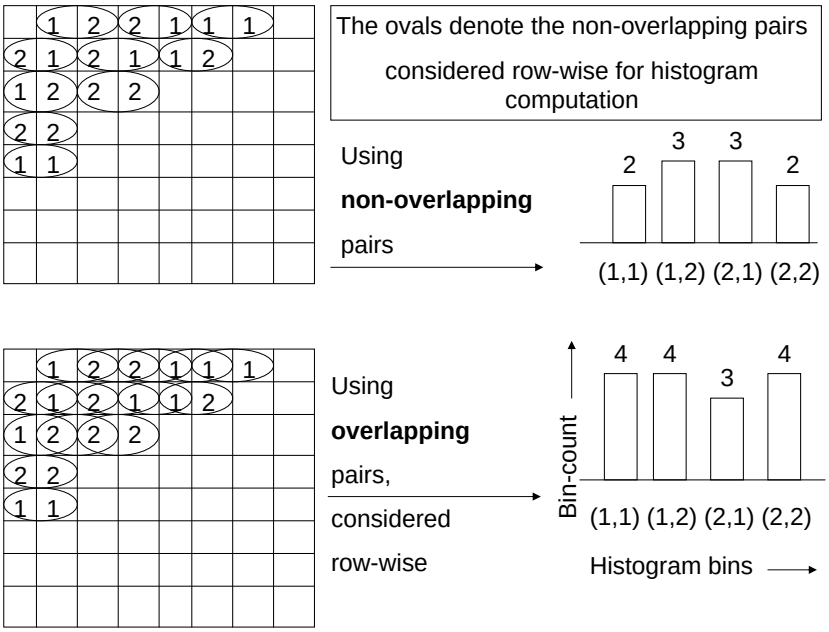


Figure 2.2: Histogram computation row-wise, considering overlapping and non-overlapping pairs of coefficients, per 8×8 blocks

bins is N^2. Due to the increased number of bins, storing the cost transportation matrix ($N^2 \times N^2$) becomes difficult and solving the 2-D EMD problem becomes computationally intensive. We have used 350 bins in our 2-D implementation [1] due to the memory constraints. The top 350 bins for which the bin-count difference between the source and target PMFs is most significant are considered while solving the histogram-matching problem.

[1]The EMD implementation used is available online at http://ai.stanford.edu/∼rubner/emd/default.htm

Thus, at best, only an approximate 2-D PMF-matching can be obtained. Also, if the first order statistics are not fully restored and the mismatch is high enough, detection can be achieved using only 1-D PMFs. Thus, we need to solve the simplified (based on non-overlapping pairs of DCT coefficients) second-order PMF matching problem, under the constraint that first order PMFs are fully matched (or closely enough to resist steganalysis).

• *Perceptual limitations:* In [115], it was shown that for the 1-D PMF case, the optimum flow (for the MMSE solution) involves perturbations of $\pm$ 1 only. For the 2-D PMF case, during EMD-based compensation, perturbations of magnitude greater than 2 may occur for the quantized DCT coefficients. For perceptual transparency, we limit the perturbations to $\pm$ 1.

To deal with these limitations, we propose a new approach that modifies the compensation coefficients based on a local criterion. This approach is sub-optimal (in terms of flow computation), but is computationally tractable, and can closely match the distribution of overlapping pairs. This method is referred to as "joint compensation" (discussed in Section 2.2.4) as it restores intra and inter-block based correlation histograms, both of which were shown to be useful for steganalysis in [42].

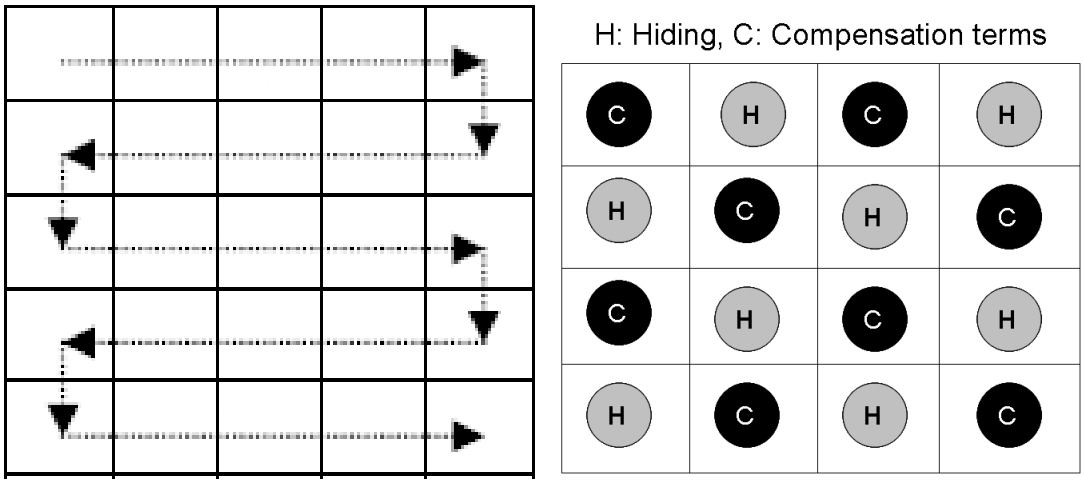
2.2.4 Joint Intra- and Inter-Block Compensation

In the EMD-based method, we considered the row-wise pairing among successive DCT coefficients in the same block such that the intra-block dependency was accounted for. In [42], scanning techniques have been discussed to construct a matrix of DCT coefficients which captures both intra and inter-block correlations.

- The AC DCT coefficients obtained from a zigzag scan along the 8×8 block are generally in descending order of magnitude. Hence, the (intra-block) correlation among consecutive terms will be more significant if the coefficients are arranged in the zigzag scan order.

- In an image, two neighboring 8×8 blocks are very likely to be similar due to the high low-frequency content. Therefore, high (inter-block) correlation can be expected between the same AC DCT term among neighboring blocks. An alternate block scanning method is shown in Fig. 2.3(a) which ensures that sequentially scanned blocks are spatially correlated.

As proposed in [42], the DCT coefficients are arranged in a $N_r \times N_c$ matrix A (shown in Fig. 2.4). The first N_c AC DCT terms occurring in the zigzag scan order in the same block of an image are placed in the same row. The N_r 8×8 blocks in the image scanned in the alternate sequence constitute the rows. Thus, two consecutive terms along the same row (column) provide the intra(inter)-block



(a) (b)

Figure 2.3: (a) Alternate scanning along the rows of an image to obtain spatially correlated blocks [42] (b) The intra and inter-block based matrix A has H and C terms arranged alternately.

correlation. The row-wise and column-wise PMFs computed using overlapping pairs of elements of the matrix A provide intra-block and inter-block based joint statistics which have been shown to be useful for steganalysis [42]. While considering non-overlapping pairs either along the rows or columns of A, intra-block or inter-block compensation can be done respectively using the EMD formulation, but not both simultaneously.

For allowing overlapping pairs in the PMF computation, the matrix A is divided into two classes 'C' and 'H', as in Fig. 2.3(b). If a certain point belongs to 'C', its 4 nearest (D_4) neighbors all belong to 'H' while the 4 diagonal neighbors belong to 'C' and vice versa. Hiding is performed in certain terms in 'H' while

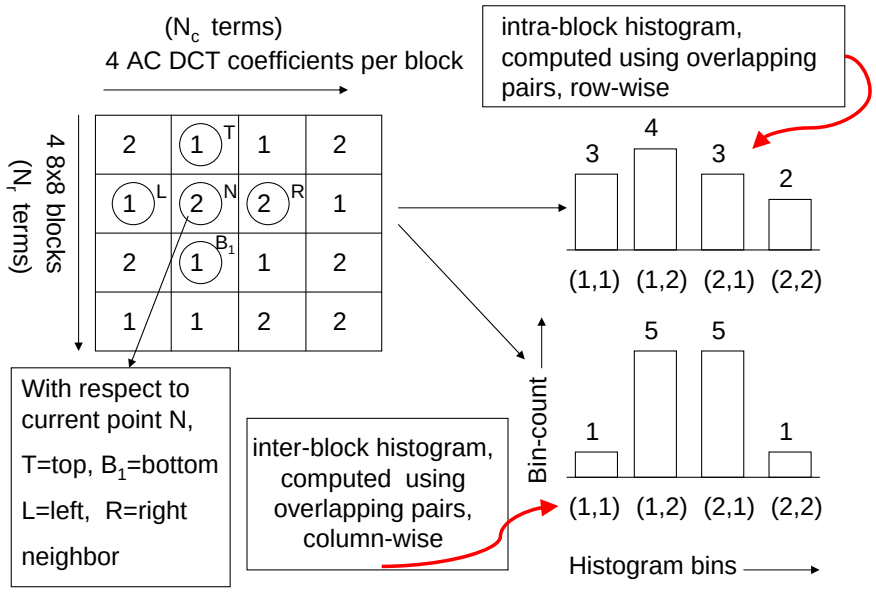


Figure 2.4: Explanation of the intra-block and inter-block histogram computation for the $N_r \times N_c$ matrix A; in this example, N_r=N_c=4

only the elements in 'C' are used for compensation. When a certain term in 'C' is modified, it affects the bin-count in pairs comprising itself and its D_4 neighbors. At each point, considering the bin-count difference, computed between original and target 2-D PMFs, for these 4 pairs, we decide on whether to perturb a certain value by $\pm$ 1 (perturbation is limited to $\pm$ 1 for perceptual transparency) or retain it. Thus, the decision taken at each compensation point is optimal provided that none of its D_4 neighbors change, which is ensured by constraining its D_4 neighbors to belong to the non-compensation stream.

Let B_{intra} (2.10) and B_{inter} (2.11) denote the intra-block and inter-block bin-counts obtained using the matrix A.

$$B_{intra}(a,b) = \sum_{i,j} \mathcal{I}_{i,j},\ \mathcal{I}_{i,j} = 1 \text{ if } \{A_{i,j} = a, A_{i,j+1} = b\},\ \text{else } 0 \tag{2.10}$$

$$B_{inter}(a,b) = \sum_{i,j} \mathcal{J}_{i,j},\ \mathcal{J}_{i,j} = 1 \text{ if } \{A_{i,j} = a, A_{i+1,j} = b\},\ \text{else } 0 \tag{2.11}$$

where $A_{i,j}$ is the element in the i^{th} row and j^{th} column of A. After data hiding without compensation, let the modified versions of the intra and inter-block bin-counts of the matrix A be B'_{intra} and B'_{inter}, respectively. Let us consider the element $A_{i,j}$, which equals N, while (as in Fig. 2.4) its D_4 neighbors are $A_{i,j-1} = L$ (left), $A_{i,j+1} = R$ (right), $A_{i-1,j} = T$ (top) and $A_{i+1,j} = B_1$ (bottom).

Since a perturbation of only ± 1 is allowed, N can be mapped to one of $\{N-1, N, N+1\}$. We compute the 4 bin-count difference values for $N' \in \{N-1, N, N+1\}$:

$$\mathcal{D}(N',1) = B_{intra}(L,N') - B'_{intra}(L,N')$$

$$\mathcal{D}(N',2) = B_{intra}(N',R) - B'_{intra}(N',R)$$

$$\mathcal{D}(N',3) = B_{inter}(T,N') - B'_{inter}(T,N')$$

$$\mathcal{D}(N',4) = B_{inter}(N',B_1) - B'_{inter}(N',B_1)$$

If the point N lies on the boundary, and lacks one or more of the D_4 neighbors, we just replace the corresponding $\mathcal{D}$ term with 0.

The squared difference between the original and modified histograms for the intra and inter-block cases are considered. For every modification of N to one of $\{N-1, N, N+1\}$, there are 4 (2 intra, for L and R, and 2 inter, for T and B_1) histogram entries that are changed. So, a maximum of $4\times3 = 12$ $\mathcal{D}$ terms may vary depending on how N is changed, as in the expression for the squared error cost function J in (2.12). When N is changed to $(N \pm 1)$, the B'_{intra} and B'_{inter} terms, associated with $(N \pm 1)$ and its D_4 neighbors, are increased by 1 and the corresponding bin-counts, associated with N and its D_4 neighbors, are decreased by 1 - this explains the use of the $I_{\delta,k}$ indicator function in (2.12). N is converted to that N_{opt} (2.13) for which the squared difference term J is minimized.

$$J(N+\delta) = \sum_{i=1}^{4} \{\mathcal{D}(N,i) + 1 - I_{\delta,0}\}^2 + \sum_{i=1}^{4} \{\mathcal{D}(N-1,i) - I_{\delta,-1}\}^2 + \sum_{i=1}^{4} \{\mathcal{D}(N+1,i) - I_{\delta,1}\}^2,\ \delta = \{-1,0,1\} \tag{2.12}$$

where the indicator function $I_{\delta,k} = 1$ if $\delta = k$ and $= 0$ otherwise

$$N_{opt} = \underset{N',\ N' \in \{N-1,N,N+1\}}{\arg\min} J(N') \tag{2.13}$$

We repeat this process to obtain a locally optimal solution for each compensation location of A. Thus, it is to be noted that a greedy approach is used. The collection of all these locally optimal solutions provides an answer to the following problems, which **could not be addressed** by the EMD-based solution:

- 2-D histogram compensation considering overlapping pairs,
- simultaneous intra-block and inter-block compensation and
- deciding which DCT coefficients should be perturbed. EMD just provides the flow from one bin to the other but does not suggest the particular elements to modify. For every element N, where $N \in C$, we decide whether to leave N untouched or to change N to $(N \pm 1)$.

2.2.5 Experiments and Results

For evaluation of our steganographic schemes, support vector machine (SVM) based steganalysis is used to detect the stego images. It is to be noted that the hiding fraction is empirically set at 10% for the stego images. Methods to compute the maximum hiding fraction such that statistical security is still ensured are discussed later in Section 2.3. The hiding methods have been statistically compensated using three different schemes (Table 2.1). 4500 images are used for our experiments - half for training and the other half for testing. Both the training and testing sets have half the images as cover and the other half as stego. During the training phase, separate SVM classifiers are obtained while training on each feature used for steganalysis in Table 2.1. The SVM classifiers are then used to distinguish between cover and stego images in the testing phase.

While computing the DCT-domain 2-D histograms, only those coefficients with magnitude less than T (threshold T=30 is used) are considered. Since the distribution of the DCT coefficients falls off sharply for higher values, higher valued terms may be ignored in PMF estimation. In EMD-based 2-D compensation, 20 AC DCT terms are considered per block (Fig. 2.2). The top 350 bins, for which the bin-count difference between the source and target PMF's is maximum (350 bins are chosen due to computational complexity issues as discussed in Section 2.2.3), may vary from image to image. Due to the high low-frequency content in a natural image, the $(0,0)^{th}$ bin and most of the bins near it often have a large bin-count. Hence, a square window of bins is considered, (with $(0,0)$ as the center and each side of length 21) consisting of bin-counts of the bins it contains as the feature vector.

For the **joint intra-inter based compensation problem**, the first 15 AC DCT coefficients that occur during zigzag scan per 8×8 block constitute the hiding band. We then consider square windows of size 13×13 and centered at the $(0,0)^{th}$ bin for both the intra and inter-block histogram matrices. This results in a feature vector of length $2\times13^2 = 338$ ($13^2 = 169$ terms for intra and $13^2 = 169$ terms for inter-block features) used for joint compensation.

In Table 2.1, the steganographic methods that are used are:

- "Joint" - intra and inter block PMFs are jointly restored

- "EMD" - 2-D non-overlapping pair based intra-block PMF restoration
- "1D" - 1-D PMF restoration

We test the methods against SVM classifiers trained on the following features:

- Intra - overlapping pair based intra-block histogram
- Inter - overlapping pair based inter-block histogram
- Joint - accounts for both intra and inter-block histograms
- 1D - first order PMF (256 bins), using first 15 AC DCT coefficients per block (zigzag scan order) with values in [-128,127]
- 2D - non-overlapping pair based intra-block histogram

Table 2.1: Comparison of the performance of the three compensation methods, when steganalysis experiments are performed using five different features. P_{FA} and P_{miss} represent the probability of false alarm and of missed detection, respectively. **The rows contain the five features while the columns denote the three methods**, which are not to be confused though they have some names in common. The effective probability of error $P_{error} = \frac{1}{2}(P_{FA} + P_{miss})$.

Feature Used	P_{FA}			P_{miss}			$(P_{FA} + P_{miss})$	
	Joint	EMD	1D	Joint	EMD	1D	Joint	EMD
Intra	0.21	0.04	0.04	0.59	0.28	0.31	**0.80**	0.32
Inter	0.27	0.14	0.16	0.58	0.32	0.32	**0.85**	0.46
Joint	0.28	0.04	0.04	0.52	0.27	0.30	**0.80**	0.31
1D	0.37	0.33	0.28	0.58	0.65	0.72	**0.95**	**0.98**
2D	0.24	0.35	0.09	0.19	0.47	0.28	0.43	**0.82**

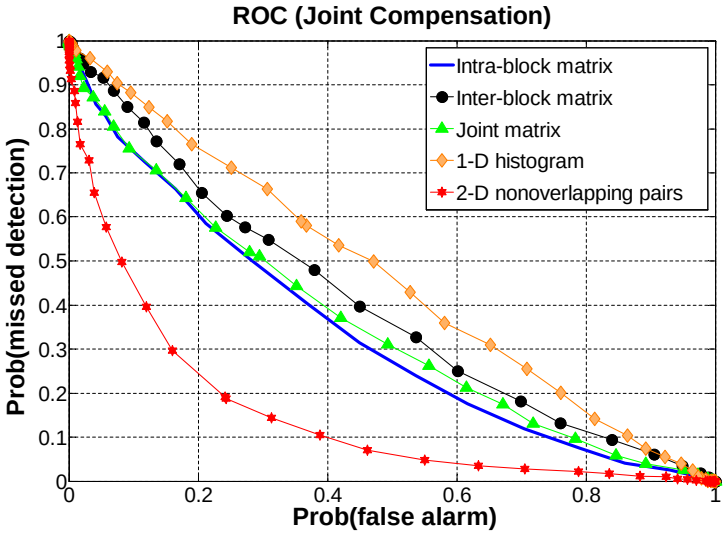
In Table 2.1, all the combinations of a steganalysis feature and a steganographic scheme that result in a high value of $(P_{FA} + P_{miss})$ are shown in bold.

P_{FA} and P_{miss} represent the probability of false alarm and of missed detection, respectively. The more undetectable a hiding method is using a certain feature, the corresponding $(P_{FA} + P_{miss})$ score will be closer to 1.

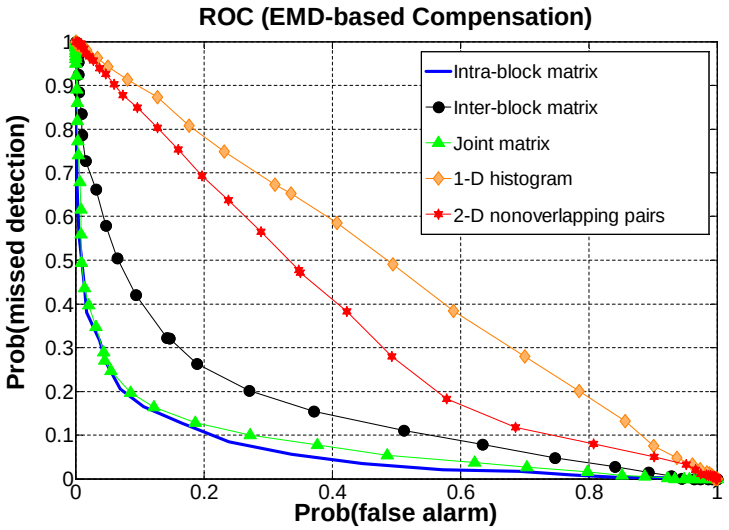
Discussion of Results: From Fig. 2.5, we observe that for steganalysis based on intra-block, inter-block and joint matrices (overlapping pair based) features, the joint compensation based steganography scheme performs better than the EMD-based scheme as we explicitly compensate for these features. For 1-D PMF as input, both the schemes do well, while for non-overlapping pair based 2-D intra-block PMF (feature described in Fig. 2.2), the EMD scheme does better.

Computational Complexity: We end this section with a brief note on the relative computational costs of the two methods. For a n bin EMD problem, the complexity $\mathcal{C}_{EMD}$ is at best $O(n^3 \log n)$ [92]; for joint compensation, the complexity $\mathcal{C}_{joint}$=$O(|A|)$, $|A|$ being the cardinality of the joint correlation based matrix A (explained in Section 2.2.4). For example, let the image size be 512×512, with 15 AC DCT terms being considered per block, while there are 350 bins in the EMD formulation. Then, $\frac{\mathcal{C}_{EMD}}{\mathcal{C}_{joint}} \approx 10^4$. Also, the complexity for the joint compensation scheme increases linearly with the image size due to the dependence on $|A|$.

In summary, we have demonstrated practical steganographic methods that provide improved security by closely matching the second order statistics. We report results for several different classifiers trained on different features, such as



(a)



(b)

Figure 2.5: Comparison of detection curves for a variety of features using (a) joint compensation and (b) EMD-based compensation methods

1-D PMFs, overlapping and non-overlapping pair based 2-D PMFs, and PMFs derived from joint (intra and inter-block based) correlations. It is seen that both the EMD-flow based and the joint compensation based schemes are quite effective in evading steganalysis based on these features.

2.3 Finding Optimal Hiding Fraction

The previous section had focussed on a method to restore second-order (or higher-order, assuming the use of non-overlapping tuples of coefficients) statistics. The change in the statistics can be compensated for if the fraction of coefficients used for hiding is small enough. The problem that is considered here is finding the maximum hiding fraction such that after hiding, the relevant statistics can be perfectly restored using the compensation coefficients.

We first define "steganographic capacity" and then present a brief survey of past methods that look at achieving hiding capacity for their respective methods.

Relevant Past Work: The concept of ϵ-secure steganography was introduced by Cachin [13]. He proposed an information-theoretic model for steganography where security is assured if the relative entropy (Kullback-Leibler divergence) between the cover and stego PMFs is less than a predefined constant ϵ. Cachin's work thus provides a theoretical framework to define the steganographic security.

While the ϵ-secure analysis of steganographic security assumes the best possible detector, irrespective of computational complexity, more practical ways of defining the steganographic security have been discussed in [53].

Fridrich et al. [36] have defined "steganographic capacity" as follows - for a host signal, it is the maximal message length that can be embedded without producing perceptually or statistically detectable distortions. It generally depends on the hiding method. Chandramouli et al. [15] have analyzed capacity estimation for Least Significant Bit based image steganography, where the cover is assumed to follow a zero mean Gaussian distribution.

In [112], a secure hiding rate was obtained for the quantization index modulation (QIM) scheme [16], with the cover signals being generated from Gaussian distributions. The statistical restoration method [109, 110] was used for steganography. The framework is general enough to be used for other hiding methods. As explained in Section 2.2.2, we consider odd-even based embedding in the block-based quantized discrete cosine transform (DCT) domain. We present the analysis for the optimum hiding fraction for the first order histogram matching case and then show its generalization for higher orders of co-occurrence statistics.

Problem Description: The symbols X, H, C, $\hat{H}$ and $\hat{C}$ have already been explained in (2.1) and (2.5). We call the hiding fraction λ, which equals $\frac{|H|}{|X|}$, where $|X|$ denotes the cardinality of the set X. We divide the feature set into bins and

find their respective bin-counts (number of terms per bin). The normalized bin-count is regarded as the PMF.

The aim is to find the maximum hiding fraction λ_{opt}, which maximizes $|H|$, subject to the constraint that enough terms are left for compensation so that the PMF of the feature set, before and after hiding, denoted by P_X and P_Y, respectively, remains the same. Let $B_X(i)$ denote the number of elements which gets mapped to the i^{th} bin of X.

$$X = H \cup C,\ Y = \hat{H} \cup \hat{C},\ H \cap C = \phi,\ \hat{H} \cap \hat{C} = \phi \tag{2.14}$$

$$\hat{H} \cap \hat{C} = \phi \Rightarrow B_Y(i) = B_{\hat{H}}(i) + B_{\hat{C}}(i),\ \forall\ i \tag{2.15}$$

$$\text{To obtain } P_Y = P_X, \text{ we need } B_Y(i) = B_X(i),\ \forall\ i \tag{2.16}$$

$$\Rightarrow B_{\hat{C}}(i) = \{B_X(i) - B_{\hat{H}}(i)\} \geq 0,\ \forall\ i \tag{2.17}$$

$$\lambda_{opt} = \underset{\lambda=\frac{|H|}{|X|}}{\arg\max}\{|H| = |\hat{H}| :\ B_X(i) - B_{\hat{H}}(i) \geq 0,\ \forall\ i\} \tag{2.18}$$

For a dataset X, $B_X(i)$ is known; after data hiding and changing H to $\hat{H}$, $B_{\hat{H}}(i)$ can be found - thus, $B_{\hat{C}}(i)$ can be computed using (2.17). As shown in (2.17), perfect restoration is possible only if the required number of terms in every bin of $\hat{C}$ is non-negative. As λ increases, the distance between the two PMFs P_X and P_Y increases and there are less terms available for compensation.

2.3.1 Using the Odd-Even Hiding Framework

We introduced the odd-even embedding based hiding framework in Section 2.2.2. Here, we exploit the fact that under this hiding framework, rounded image coefficients can be perturbed at most by ± 1 after hiding.

Let λ be the common hiding fraction for all bins. Let $X(i)$ and $\hat{H}(i)$ denote the elements mapped to the i^{th} bins of X and $\hat{H}$, respectively. Now, assuming an equal number of 0's and 1's in the input message that affects the elements in $X(i)$, $\frac{\lambda}{4}$ fraction of coefficients from $X(i)$ gets transferred to both $\hat{H}(i+1)$ and $\hat{H}(i-1)$. Also, $\frac{\lambda}{2}$ fraction of coefficients is moved to $\hat{H}(i)$. **Explanation** - let the value of the input QDCT coefficient be i, an even number, and if the input bit is 0, the output term, obtained using (2.9), is i itself. Since about half the bits in the input sequence are 0, about $\frac{\lambda}{2}$ terms in $X(i)$ are moved to $\hat{H}(i)$. If the input bit is 1, the output term gets mapped to the nearest odd number, which can be $(i-1)$ or $(i+1)$, depending on whether the dither value (δ in (2.8)) is positive or negative. By a similar logic, $\frac{\lambda}{4}$ fraction of terms from bins $X(i-1)$ and $X(i+1)$ will be shifted to $\hat{H}(i)$. Thus, based on this analysis, the number of terms in $\hat{H}(i)$ is as follows:

$$B_{\hat{H}}(i) \approx \frac{\lambda B_X(i)}{2} + \frac{\lambda B_X(i-1)}{4} + \frac{\lambda B_X(i+1)}{4} \tag{2.19}$$

To reiterate, the main assumptions behind this analysis are : (i) the input message has equal number of 0's and 1's and (ii) the dither values are equally likely to be

positive or negative. The assumptions are valid only if both the message and the dither sequence are long enough (minimum image size considered is 256×256). The goodness of this assumption is experimentally verified in Section 2.5.1.

2.3.2 Hiding Fraction and Rate Computation

While computing the 1-D histograms for QDCT coefficients, we only consider those with magnitude less than a certain threshold T. Since the distribution of the QDCT coefficients is very peaky near 0 and falls off sharply for higher values, higher valued terms may be ignored in PMF estimation. For a given T, there are $(2T+1)$ bins from $[-T, T]$, and we optimally hide in all the bins, except the two extreme ones. For the $(-T)^{th}$ and T^{th} bins, perfect compensation may not be possible as we consider neighboring bins at one side only. From (2.5) and (2.19), considering the i^{th} bin, the hiding fraction λ needs to satisfy:

$$B_{\hat{H}}(i) \leq B_X(i) \Rightarrow \lambda \leq \left\{ \frac{B_X(i)}{\frac{B_X(i-1)}{4} + \frac{B_X(i)}{2} + \frac{B_X(i+1)}{4}} \right\} \tag{2.20}$$

For ease of notation, we define

$$\lambda_i = \left\{ \frac{B_X(i)}{\frac{B_X(i-1)}{4} + \frac{B_X(i)}{2} + \frac{B_X(i+1)}{4}} \right\} \tag{2.21}$$

It is to be noted that the whole analysis, especially, the expression for $B_{\hat{H}}(i)$ (2.19) as was derived in Section 2.3.1, assumed an equal hiding fraction for all the bins. For the i^{th} bin, λ_i can be viewed as $\frac{B_X(i)}{B_{\hat{H}}(i)}$ where $B_{\hat{H}}(i)$ is computed using a

hiding fraction of unity. In Section 2.4, we shall be using this notation for $B_{\hat{H}}$ for the higher order cases. The effective hiding fraction $\lambda^{\star}(T)$, for a given T, is the minimum of all these λ_i terms (since the hiding fraction $\lambda \leq \lambda_i, \forall i$, using (2.20) and (2.21)).

$$\lambda^{\star}(T) \quad = \quad \min_{-T<i<T} \{\lambda_i : \ \lambda_i > 0\}. \tag{2.22}$$

The condition ($\lambda_i > 0$) in (2.22) ensures that the hiding fraction will not be reduced to zero for bins with no elements. This may lead to PMF mismatches in bins with no elements before hiding but the mismatch is unlikely to be statistically and steganalytically significant, and hence not too useful for detection. Also, just as for the equal number of 0's and 1's assumption in Section 2.3.1, the experimental results in Section 2.5.1 indicate that it is a valid assumption for the first order histogram matching case.

Once we select a certain frequency band for hiding after performing block-wise DCT, the maximum fraction of the terms which can actually be used for hiding at a given threshold under the statistical restoration constraint is called the "rate" for that threshold. Let $G(T)$ denote the fraction of terms available for hiding at threshold T, while the hiding rate corresponding to a threshold T is $R(T)$.

$$G(T) \quad = \quad \sum_{-T<i<T} P_X(i) \tag{2.23}$$

$$R(T) \quad = \quad \lambda^{\star}(T).G(T) \tag{2.24}$$

where P_X is the PMF of X. As T increases, $G(T)$ increases while $\lambda^\star(T)$ decreases, since we are finding the minimum value over a larger set of T's (2.22). We vary the thresholds and select the threshold T_{opt} for which the rate is maximized.

$$T_{opt} = \arg\max_{T} R(T) \tag{2.25}$$

Thus, the maximum attainable rate for the QDCT based feature set using odd-even embedding and first-order compensation is $R(T_{opt})$, computed using (2.21)-(2.25).

The contributions of our proposed approach are as follows:

1. given a certain image and the quantized DCT coefficient being the feature used for odd/even based hiding, we can provide an estimate of the maximum hiding rate λ when hiding is done in a select band of mid-frequency coefficients having values in the range $[-T, T]$.

2. We are also able to estimate the optimal threshold value T_{opt} which provides the maximum hiding rate $R(T_{opt})$.

Note: A question that arises here is how the decoder knows which are the hiding coefficients and which are the compensation coefficients. If the hiding fraction is pre-decided, then the embeddable coefficients can be selected using a key which generates a pseudo-random sequence which identifies the hiding locations

(e.g., if there are 1000 embeddable coefficients and the encoder and decoder decide that the hiding fraction is 10%, then a sequence of length 100 is generated based on a shared key so that both the encoder and decoder know which 100 coefficients are actually used for hiding). When the encoder determines the optimum hiding fraction based on the image DCT-domain PMF, then the decoder has to independently compute the hiding fraction. If the hiding fraction is wrongly computed, the decoding will be erroneous. Since the SR framework restores the first (or second) order statistics, our experiments have confirmed that the decoder can independently and correctly retrieve the optimal hiding fraction, as was used by the encoder. If the image is subjected to more severe compression (e.g., when $QF_a < QF_h$) which distorts the stego image (and also its DCT-domain histogram) significantly, the decoder may be unable to retrieve the accurate value of the hiding fraction. If a "significantly" noisy channel is expected, a more conservative estimate of the hiding fraction can be used instead of aiming for the maximum hiding rate.

2.4 Extension to Higher Order Statistics

In the odd-even based hiding scheme, let us consider two coefficients at a time (2-D co-occurrence scenario). Let the two terms have values i and j respectively.

If we call this pair as (i, j), then owing to an incoming bit, the new coefficient pair can be (i', j') where $i' \in \{i-1, i, i+1\}$ and $j' \in \{j-1, j, j+1\}$. We now obtain the optimum hiding fraction, given a certain threshold T, in a manner identical to the 1-D steganography case. Let $B_X(i, j)$ denote the bin-count in the $(i, j)^{th}$ bin of X.

$$B_{\hat{H}}(i,j) = \sum_{(i',j') \in D_8 \setminus D_4} \frac{B_X(i',j')}{16} + \sum_{(i',j') \in D_4} \frac{B_X(i',j')}{8} + \frac{B_X(i,j)}{4} \tag{2.26}$$

$$\lambda_{i,j}(T) = \frac{B_X(i,j)}{B_{\hat{H}}(i,j)} \tag{2.27}$$

$$\lambda^{\star}(T) = \min_{-T<i,j<T} \{\lambda_{i,j}(T) : \lambda_{i,j}(T) > 0\} \tag{2.28}$$

The $B_{\hat{H}}$ term in (2.26) is the bin-count for the $(i, j)^{th}$ bin of $\hat{H}$ computed using a hiding fraction of 1. Then, (2.27) and (2.28) are just the 2-D versions of (2.21) and (2.22), respectively. In (2.26), the set of the four nearest neighbors of the current 2-D point (i, j) is called D_4 while the set of D_4 and the four diagonal neighbors is called D_8.

A generalization for the n^{th} order co-occurrence statistic is now presented. A single bin will consist of n elements, say $(i_1, i_2, ..., i_n)$. Since we perform odd-even based hiding, the i_1 component can be mapped to i_1, $(i_1 - 1)$ or $(i_1 + 1)$ with probability $\frac{1}{2}, \frac{1}{4}$ and $\frac{1}{4}$, (valid under the same two assumptions as in Section 2.3.1) respectively. Thus, $(i_1, i_2, ..., i_n)$ can be mapped to $(i_1 + \delta_1, i_2 + \delta_2, ..., i_n + \delta_n)$, where $\delta_j \in \{-1, 0, 1\}, 1 \leq j \leq n$.

$$f(0) = \frac{1}{2}, f(1) = \frac{1}{4}, f(-1) = \frac{1}{4} \tag{2.29}$$

$$B_{\hat{H}}(i_1, i_2, ..., i_n) = \sum_{\delta_1} \sum_{\delta_2} ... \sum_{\delta_n} [f(\delta_1) f(\delta_2) ... f(\delta_n)] \times B_X(i_1 + \delta_1, i_2 + \delta_2, ..., i_n + \delta_n) \tag{2.30}$$

For the co-occurrence order $n = 1$ and 2 in (2.30), we compute $B_{\hat{H}}$ using (2.19) and (2.26), respectively. The optimal hiding fraction, $\lambda^\star(T)$ can be computed as in (2.27) and (2.28) for the 2-D case, by taking the ratio of the B_X and the $B_{\hat{H}}$ terms, and finding the minimum over a range specified by T. The hiding rate and optimal threshold estimates, $R(T)$ and T_{opt}, can then be obtained, using (2.24) and (2.25), respectively.

2.4.1 Variation of Hiding Rate With Order of Co-occurrence Statistics

As we proceed from 1-D to 2-D co-occurrence statistic for the QDCT coefficients, the number of coefficients per bin decreases - the total number of coefficients remains the same but the number of bins in the 2-D case is the square of the number of bins in the 1-D case. Thus, there are many empty 2-D bins. This poses a danger to the 2-D statistical restoration scheme. Say, in the original image, there are no elements in bin (i, j), i.e. $B_X(i, j) = 0$. Now, after hiding, say

some elements from nearby bins ($X(i-1,j)$, $X(i+1,j)$ and so on) have shifted to $\hat{H}(i,j)$, making $B_{\hat{H}}(i,j)$ positive. Thus, using (2.5), for the target compensation stream $\hat{C}$, the required bin-count in bin (i,j), $B_{\hat{C}}(i,j)$ becomes negative, making compensation for bin (i,j) impossible. To reiterate, lesser the number of elements per bin, more will be the number of bins where $B_{\hat{C}}(i,j)$ is negative and vice versa. *Thus, the hiding fraction that can be allowed so that statistical restoration is possible will be decreased with increasing order.* This also makes sense intuitively.

We put a tolerance limit ($p\%$) on the number of bins in X with zero elements. The threshold T is gradually increased from 1 till we found there were more than $p\%$ bins which had zero elements. Let the threshold corresponding to $p\%$ bins having zero elements be T_p. The threshold is varied from 1 to T_p and the threshold T_{opt} (2.25) is found at which the hiding rate $R(T)$ (2.24) is maximum. We use $p = 5$ in the experiments (Table 2.2).

For generating the QDCT terms for the luminance part of an image, we use a quality factor of 75. The first 19 AC DCT coefficients, that occur during zigzag scan, are considered for a 8×8 block, for hiding and compensation. The range of allowed threshold values is limited to [-30,30]. For every image, after computing the QDCT terms X, $B_X(i)$ is computed for all the bins for a certain threshold T ($i \in [-T,T]$). The hiding fraction $\lambda^\star(T)$ is obtained using (2.22). The maximum

attainable rate $R(T_{opt})$ is then computed using (2.25). This process is repeated for higher orders of co-occurrence statistics. In Table 2.2, the steganographic capacity is computed in 3 ways: (i) the maximum attainable rate $R(T_{opt})$, (ii) the bits hidden per pixel in the image and (iii) the total number of bits embedded per image. The experiment is performed on 4500 images and the optimal hiding parameters, averaged over the entire set, are reported.

Table 2.2: Variation of the optimum hiding threshold, fraction and capacities with the order of co-occurrence, being averaged over 4500 images - since the threshold can assume only integer values, T_{opt}, after averaging, is changed to the nearest integer higher than it.

Order	T_{opt}	$\lambda^{\star}(T_{opt})\%$	$R(T_{opt})$	Bits/pixel	bits/image ($\times 10^3$)
1	27	48.434	0.502	0.141	25.120
2	6	29.895	0.264	0.074	13.242
3	3	8.253	0.057	0.016	2.895

The maximum allowed hiding fraction expectedly decreases as we compensate for higher orders of co-occurrence. The amount of data that we need to hide decides the maximum order of co-occurrence upto which we need to compensate. Once co-occurrence statistics are restored upto a certain order, detection is always possible using a higher order statistic - we experimented upto the third order statistic.

In summary, we have demonstrated a method to compute the maximum hiding fraction and hiding rate for odd-even based hiding for quantized DCT coefficients

such that the hiding remains undetectable after first order statistical restoration. The method is generalized for higher orders of co-occurrence statistics.

2.5 Total Compensation vs Individual Compensation

In the statistical restoration method discussed so far, the coefficient set used for hiding consisted of 19 coefficients chosen per 8×8 block. We call this method "**total compensation**", while contrasting it with "**individual compensation**" where the steganographer explicitly restores the histograms of individual AC DCT components. There are some issues to consider here. Even if the steganographer manages to restore the first order statistics considering all the 19 coefficients, the histograms themselves may not be matched, considering the individual frequency terms. If there is a consistent pattern in the variation of the individual coefficient histogram between the original and the stego image, it can be exploited for successful steganalysis. In absence of a consistent pattern, even if the individual histograms are not matched, it may not help in detection.

In the following discussion (Section 2.5.1), we show that after performing total compensation based steganography and using individual coefficients for steganalysis, detection becomes easier only for certain frequency coefficients. The

steganographer should explicitly restore the individual coefficient histograms for these frequencies to avoid detection.

2.5.1 Use of Individual Frequency Coefficients

The steganographer may consider a certain band of QDCT terms for hiding in the "total compensation" procedure. The steganalyst is however free to choose a feature of his choice; for example, he may consider first order histograms corresponding to each individual frequency coefficient stream. For each individual stream, there may be PMF mismatches between a cover and the corresponding stego image. However, as mentioned before, machine-learning based steganalysis will be able to detect the hiding only if the mismatches are consistent enough across images. Here, for the total and individual compensation cases, the maximum possible data is embedded while ensuring that first order restoration is still possible for the entire frequency band based histogram and for each individual frequency band based histogram, respectively, using (2.22).

We use 4500 images for the experiments - half for training and the other half for testing. Both the training and testing sets have half the images as cover and the other half as stego. During the training phase, separate support vector machine (SVM) classifiers are obtained based on each individual QDCT stream. The SVM classifiers are then used to distinguish between cover and stego images in the test-

ing phase. For steganography, we experiment with both the total compensation and individual compensation methods. Data is embedded in those QDCT coefficients whose magnitude is less than 30. The QDCT features are generated using a hiding quality factor QF_h of 75 and a hiding band of 19 AC DCT coefficients. The probabilities of missed detection (P_{miss}) and false alarm (P_{FA}) are computed after performing histogram-based steganalysis using the individual QDCT streams. For undetectable hiding, the total detection error $P_{total} = (P_{FA} + P_{miss})$ should be close to 1.

The P_{total} term is found to be close to 1 for all the 19 AC DCT streams for individual compensation - indicating undetectable hiding. The fact that the optimal hiding fraction based scheme turns out to be undetectable shows that the assumptions (mentioned in Section 2.3.1) based on which the hiding parameters were derived are practically justified. After performing total compensation and using individual QDCT streams for detection, we found that the P_{total} term varies for different QDCT streams. We compute the difference between the P_{total} values, averaged over all the test images, for the individual and total compensation cases, for each of the 19 streams. Let QDCT(i, j) denote the QDCT stream corresponding to the i^{th} row and j^{th} column of the 8×8 QDCT matrix ($1 \leq i, j \leq 8$). The difference in P_{total} is significantly greater than 0 for certain streams (0.21-QDCT$(1, 2)$, 0.10-QDCT$(1, 3)$, 0.15-QDCT$(1, 5)$, 0.44-QDCT$(1, 6)$) while it is very close to 0

for the remaining 15 terms. We present the difference in P_{total} for some frequency coefficients in Fig. 2.6. A high difference, as in Fig. 2.6(a) and (b), indicates that individual stream based steganalysis is able to detect total compensation based hiding for these streams while a small difference, as in (c) and (d), suggests that explicit compensation is not needed for these streams to avoid detection. *Thus, we observe that total compensation based hiding is most detectable when* $QDCT(1,6)$ *is used for individual frequency band based steganalysis.*

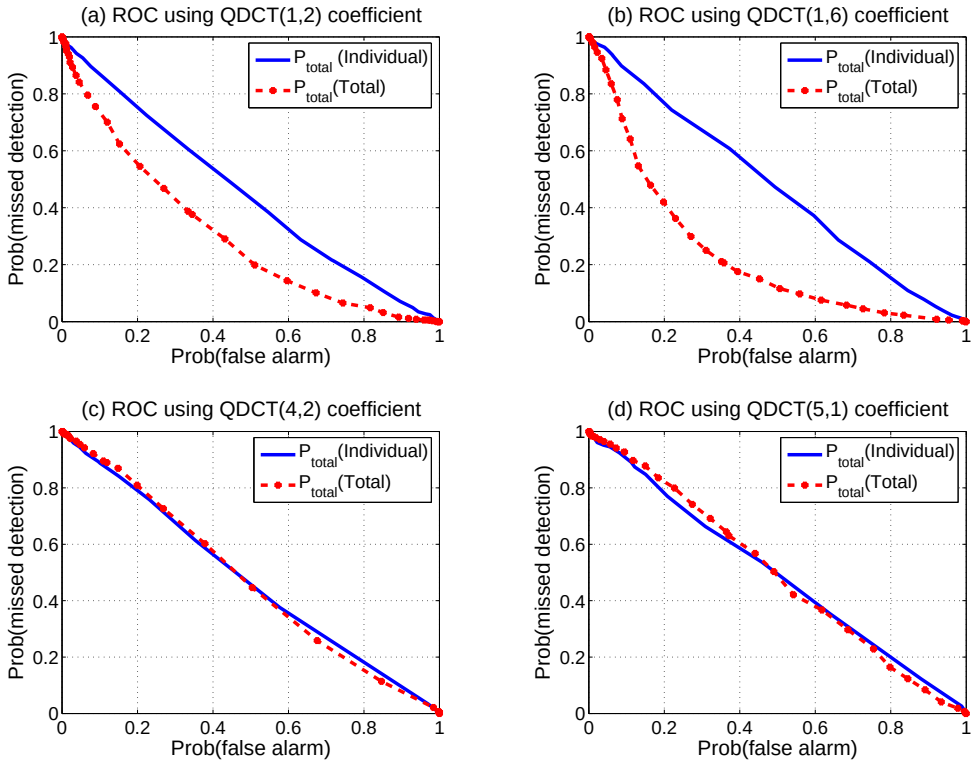


Figure 2.6: Average detection error (P_{total}), averaged over all the test images, computed for different QDCT streams after statistical compensation: "Individual" and "Total" refer to the individual and total compensation schemes, respectively.

To explain the superior performance of QDCT$(1,6)$ over other streams, the PMF difference is computed between the original image's QDCT stream and the data-embedded (and compensated) QDCT stream, for each individual stream and for both the compensation methods. Examples of the average PMF difference (averaged over the test stego images) are presented in Fig. 2.7. It is observed that whenever the PMF difference is consistently high for the low magnitude bins, i.e. $\{-1,0,1\}$, it reflects in increased detection accuracy (Fig. 2.7(b)) - this occurs due to the peaky nature of the PMF near 0. While the peak PMF difference is as low as $3{\times}10^{-3}$ for QDCT$(5,1)$ (Fig. 2.7(a)), it is as high as 0.10 for QDCT$(1,6)$ (Fig. 2.7(b)). The PMF of QDCT$(1,6)$ is found to be much more peaky compared to that of other frequency streams. As the PMF of a certain frequency coefficient X becomes more peaky near 0, $B_X(0)$ becomes much greater than $B_X(1)$ and $B_X(-1)$. Using (2.21), the hiding fraction λ_i for $i=\{-1,1\}$ becomes very small due to the dominance of $B_X(0)$. The effective hiding fraction $\lambda^\star(T)$ (2.22) for QDCT$(1,6)$ will therefore be much smaller compared to other frequency streams, whose PMF is less peaky. When "total compensation" is performed, we consistently hide much more data in QDCT$(1,6)$ than is permitted by $\lambda^\star(T)$ (2.22); *hence, statistical restoration cannot compensate for the mismatched PMF and this leads to enhanced detection.*

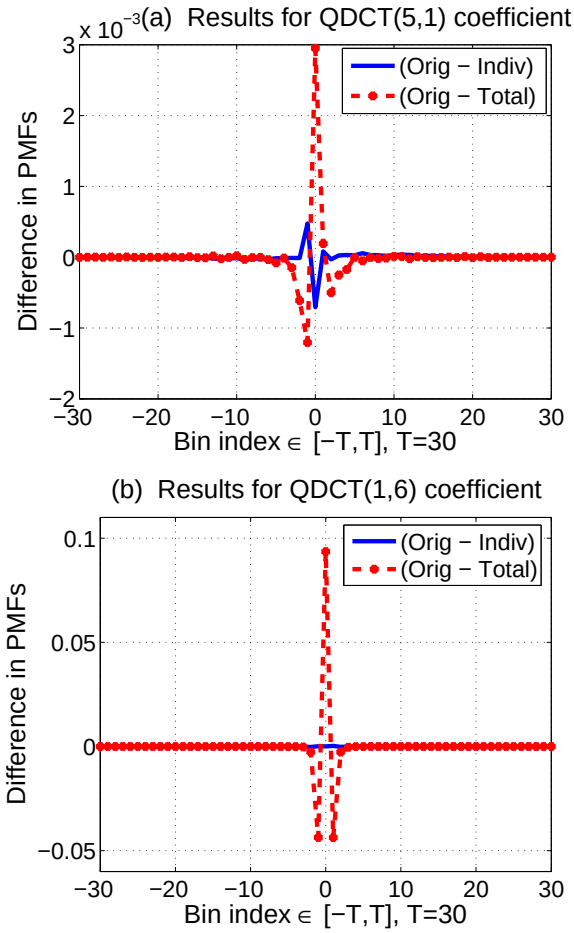


Figure 2.7: Comparison of PMF differences, after statistical restoration, for individual and total compensation based methods, for different QDCT streams: here, "Orig", "Indiv" and "Total" refer to the original PMF, PMF after hiding and individual compensation, and PMF after hiding and total compensation, respectively.

In summary, from a steganalyst's perspective, we have looked at first order histograms of individual frequency streams. It is observed that a certain quantized DCT stream (pertaining to the 1^{st} row and 6^{th} column per 8×8 block) is particularly effective for first order steganalysis. There is a direct relationship

between the peaky nature of the PMF of an individual quantized DCT stream and its usefulness in first order steganalysis.

2.6 Steganographic Capacity Estimation

For a practical implementation of an active (channel attacks are permissible) steganographic system, the hiding has to satisfy perceptual, statistical and attack constraints. Of these various constraints, the statistical constraint is the most difficult to satisfy for a practical system - there are excellent blind steganalysis methods [4, 82, 83, 103] that are able to detect most of the modern-day steganographic schemes. Here, by statistical security, we refer only to the first order histogram based security. The focus here is not on designing a system which is statistically secure even to the most sophisticated detectors. Instead, our aim is to link up the quantization index modulation based hiding method with the statistical restoration based method, along with the addition of sufficient redundancy in the error-correction code framework so as to survive channel errors (satisfy the attack constraint). In practice, a steganographic system should be able to withstand a variety of attacks - here, the system is designed only for a pre-defined level and type of attack. Steganographic capacity is a quantification of the maximum hiding rate under these constraints. The estimated capacity (*to emphasize,*

it is our estimate of the hiding capacity and not a theoretical limit) depends on the transform domain used for hiding, the embedding method used, the statistical feature used for steganalysis, and the error correction code used. Strictly, capacity is independent of the ECC code - however, since we do not know the ideal channel code and RA codes are used under the assumption that they are a close enough approximation of the ideal channel code, the capacity estimate that is obtained depends on the RA code. The mismatch between the ideal capacity and the estimated capacity is caused by the difference between the minimum redundancy needed by an ideal channel code and that needed by the RA code for a given image and a given attack channel.

In this section, we provide *an end-to-end framework where given an image, and the simple assumptions about the first order statistical security and allowable levels for a given attack, its steganographic capacity can be computed.* It is based upon work in Section 2.3, except that the attack constraint had not been considered there. Why do we need to fix so many parameters (hiding parameters, detection feature, ECC) to find the capacity? This is because the composite hiding channel also consists of the image - hence the dependence on the image transform domain used for hiding. The effective channel model depends on the changes done to the image coefficients - hence, the embedding method needs to be fixed. To survive the channel attacks, the ECC should have proper redundancy - the redundancy used

varies depending on the ECC method used. For statistical security, depending on the features used for steganalysis, the hiding rate varies as shown in Section 2.3.

The capacity of data hiding channels has been a well-studied theoretical problem [73]; the rate of perfectly secure data hiding in noiseless channels [36, 112] has also been investigated. However, putting all the constraints together as required by a practical steganographic scheme, has only been done recently by Wang et al. [121].

2.6.1 Transform Domains used for Hiding

As mentioned before, the channel model and hence, the effective hiding capacity depends on the choice of hiding coefficients, i.e. on the transform domain used and the selection of the embedding band.

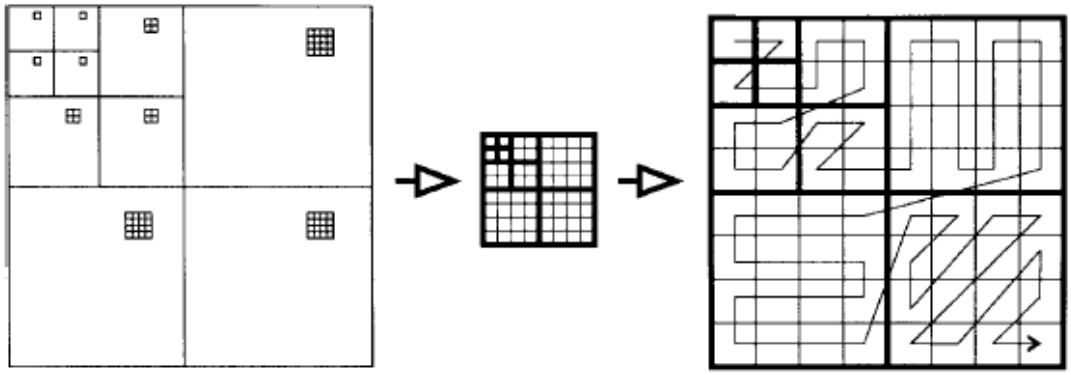
DCT domain: The hiding band used in DCT domain is the same as that used in Section 2.1 and 2.3. After selecting a design quality factor QF_h for hiding, the DCT coefficients, computed for every 8×8 block, are divided element-wise by the 8×8 quantization matrix, corresponding to QF_h. The first 19 AC DCT terms, that occur during a zigzag scan (Fig. 2.8(b)), are used for hiding.

DWT domain: For the wavelet domain based scheme, we use the Haar wavelet as the basis function and use the periodic Discrete Wavelet Transform (DWT) mode, with 3 level decomposition, to obtain a 8×8 matrix of DWT coefficients,

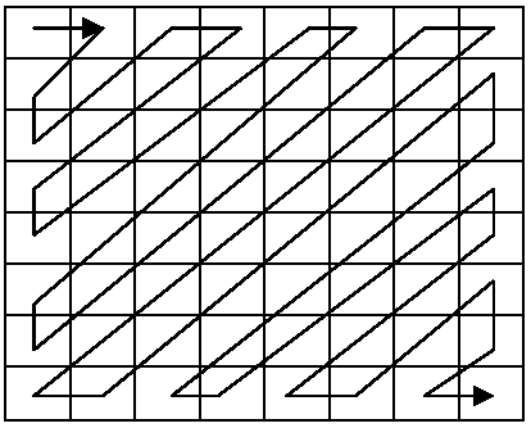
given a 8×8 pixel block. In [23] and [1], a method is described to generate a 8×8 wavelet domain quantization matrix which corresponds to a certain design quality factor. It ensures that the embedding bit-rate for DWT based hiding using the wavelet domain quantization matrix is comparable to the embedding bit-rate obtained using the corresponding quality factor for DCT based hiding. For choosing the frequency band for hiding, a modified scanning procedure is used, as described in [23]. The first 19 DWT terms (leaving the top leftmost LLL coefficient) that occur during the modified scanning procedure are used for embedding (Fig. 2.8(a)).

2.6.2 Steganographic Capacity Constraints

As mentioned before, we consider the problem of *active* steganography, in which the adversary can also modify the stego signal (JPEG and JPEG-2000 based compression attacks for our case) - thus introducing an attack constraint. Let X denote the cover signal, S the stego signal, and Y the received signal (all in the same transform domain) at the decoder after attack. The problem of finding the capacity of such active warden stegosystems for an i.i.d. (independently identically distributed) cover signal reduces to maximizing the embedding rate with the following constraints. We present the three constraints - perceptual, statistical and attack constraints.



(a)



(b)

Figure 2.8: (a) DWT computation using 3 levels - the modified scanning procedure for the 3-level decomposition is shown, (b) zigzag scan for block-based DCT coefficients as in the JPEG coding standard

1. **Perceptual constraint:** The perceptual distortion between the original and stego images in the transform domain should not exceed a certain maximum amount, D_1, for some perceptual distance measure $d(\cdot, \cdot)$. Thus, we must have $d(X, S) \leq D_1$. Distortion constraints for limiting the perceptual

distortion have long been used in the information-theoretic analysis of the data hiding problem ([16, 19, 73]).

The **perceptual distortion constraint** can be interpreted as the mean square error between the transform domain feature matrices, before and after hiding. For DCT and discrete wavelet transform (DWT) domains, it is assumed that a distortion of ± 1 for the coefficients in the chosen frequency band ensures perceptual transparency of the stego signal - with a proper choice of frequency domain terms for hiding, perceptual transparency for the DCT domain is shown in our prior work [106, 107]. Instead of specifying a fixed value for the distortion constraint D_1, our aim is to minimize the perceptual distortion during hiding while ensuring that the other constraints are maintained.

2. **Statistical constraint:** The embedding process should not modify the statistics of the host signal more than a very small number, ϵ, for some statistical distance measure. Cachin [13] proposed the use of Kullback-Leibler (K-L) divergence for defining the statistical security of steganography. Thus, denoting the cover and stego distributions by P_X and P_S, respectively, the statistical constraint can be given as, $\mathcal{D}(P_X||P_S) \leq \epsilon$, where $\mathcal{D}(\cdot,\cdot)$ is K-L distance measure. For perfect security, we aim at obtaining $P_X = P_S$. With

the addition of more constraints (more complicated features than the first order histogram), the system will become statistically more secure and the capacity will decrease. However, in the context of this discussion, only the first order histogram for the entire hiding band is considered for statistical security.

3. **Attack constraint:** The embedded data must be recoverable after the stego signal has undergone an attack distortion of at most D_2, i.e. if $d(S, Y) \leq D_2$.

   For varying the severity of the compression attack, the experiment is repeated for various combinations of quality factors (for data embedding and for generating the compressed image) and compression rates, for JPEG and JPEG-2000 based attacks, respectively. For the **attack constraint**, we do not fix D_2 but our aim is to maximize the hiding rate, by using the minimum redundancy during error correction coding, which ensures zero bit error rate (BER) at the decoder, even after compression attacks. The assumption here is that *the encoder knows the exact attack; hence, at the encoder side, the sender can simulate the exact attack and obtain the minimum redundancy at which perfect decoding is possible.*

2.6.3 Satisfying Statistical and Perceptual Constraints

We use the same symbols to define the hiding and compensation terms for the SR framework as in Section 2.1 and 2.3. Our compensation framework also ensures that the perturbation to an individual coefficient is limited to [-1,1].

For maximizing the embedded bit-rate, under the perceptual and statistical constraints, the two objectives are:

- find the maximum hiding fraction $\lambda = \frac{|H|}{|X|}$, where $|X|$ denotes the cardinality of the set X
- find the optimal way to modify C to $\hat{C}$, once the optimal hiding fraction is determined.

The steps involved in obtaining C and $\hat{C}$ for a given set X are as follows:

1. find the hiding fraction - $\lambda^*(T_{opt})$, using (2.22) and (2.25)
2. using $\lambda^*(T_{opt})$, divide X to H and C
3. modify H to $\hat{H}$ to hide $|H| = \lambda^*(T_{opt}).|X|$ bits - this is the total number of code bits
4. since P_C and $P_{\hat{C}}$ are known, we make the relevant changes using the transportation flows derived from the EMD formulation

In Section 2.3, we have already shown how the maximum hiding fraction, that ensures that first-order statistics are maintained, can be computed. Thus, given a set of coefficients X, we first find the maximum hiding fraction $\lambda^{\star}(T_{opt})$ (2.22). Once the maximum hiding fraction has been obtained, we perturb C to $\hat{C}$ where the transportation flows are given by the EMD formulation. Here, we have accounted for the perceptual and statistical constraints - only the attack constraint remains.

2.6.4 Accounting for Channel Distortions

The only issue remaining to be considered is finding q_{opt}, the minimum redundancy factor that helps in data recovery for a given image and a given noise channel. Once q_{opt} is known, the maximum possible databits that can be embedded, while maintaining the statistical and attack constraints, is determined - the sequence of steps involved is shown in Fig. 2.9.

The overall system flow for DCT based hiding is briefly outlined in Figure 2.10. For a generalized framework, the DCT coefficients can be replaced by any other transform domain suitable for hiding and the JPEG attack block, denoted by the $Z \rightarrow Z'$ mapping, can be substituted by any other distortion channel. The maximum allowable size of the message M to be embedded depends on $\lambda^{\star}$ and q_{opt}, the computation of which is shown in Fig. 2.9. It is to be noted that the

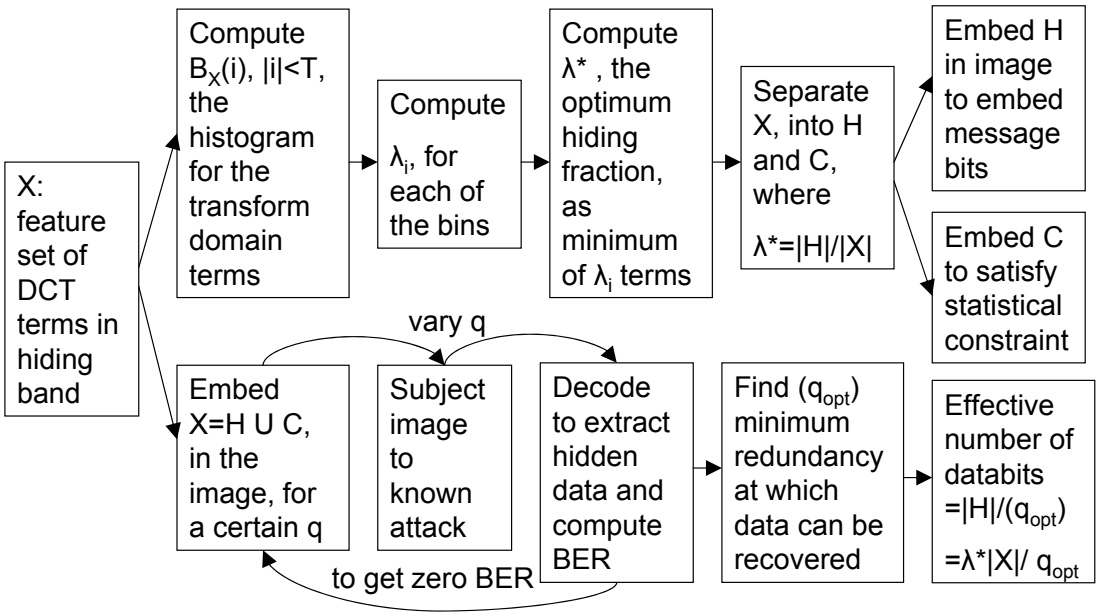


Figure 2.9: The framework to hide the maximum number of databits, maintaining the statistical (use $\lambda^\star$ as the hiding fraction) and attack constraints (use minimum redundancy that ensures zero BER) is shown here. The perceptual constraint is implicit in the QIM based hiding scheme. At first, $\lambda^\star$ is computed and then, after separating X into H and C based on $\lambda^\star$, the optimum redundancy factor q_{opt} is determined. Here, T denotes the threshold where quantized DCT coefficients in the range $[-T, T]$ are used for hiding.

encoder does not initially know q_{opt}. So, the process of hiding and then recovering the data simulating the channel attacks at the encoder is repeated till the encoder finds the minimum redundancy which ensures zero bit error rate (BER). A more systematic approach to compute q_{opt} is presented in Section 3.5 where the structure of RA-encoded sequences is exploited.

The message, M, is first coded to R using the turbo-like repeat-accumulate (RA) code [25] with redundancy q (RA-q), the data being hidden in a band of

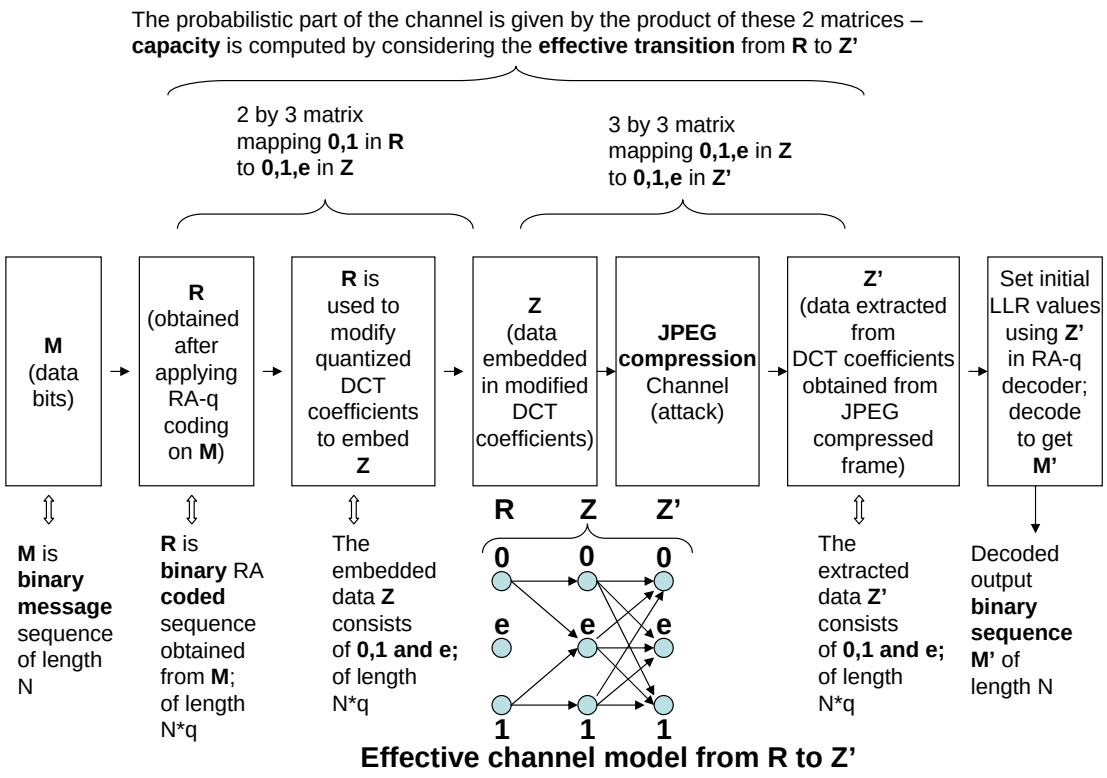


Figure 2.10: Computation of the data hiding capacity depends on the 2×3 transition probability matrix mapping R to Z' - shown here for DCT domain hiding and for JPEG compression attack (LLR = Log Likelihood Ratio)

low-frequency DCT coefficients, having n elements per 8×8 block. Therefore, the effective number of embedded bits per block=$\frac{n}{q}$ (we use n=19 in our experiments). At the time of embedding, the code symbols corresponding to the DCT coefficients beyond a predetermined threshold (T=30 in our case), are *erased* at the encoder - erasures are denoted by e in Fig. 2.10 and Eq. (2.31). Further errors are introduced due to the JPEG compression attack. The embedded data can still be recovered because of the added redundancy using RA codes.

We experimentally obtain the transition probability matrices from R to Z (*depends on the distribution of the image transform domain coefficients and the hiding method*) and from Z to Z' (*depends on the attack channel characteristics*). The mutual information $I(R, Z')$ between the input (R) and output (Z') terms in the channel is maximized to compute the capacity $\mathcal{C}_{channel}$ (2.31) for a given image and a given attack channel, which is then used to compute the minimum redundancy factor needed for data recovery *if an ideal channel code were used* - it equals $\lceil 1/\mathcal{C}_{channel} \rceil$. This minimum redundancy factor is called q_{min} (2.32).

$$\mathcal{C}_{channel} = \max_{p(r)} I(R, Z') = \max_{p(r)} \sum_{r \in \{0,1\},\ z' \in \{0,1,e\}} p(r, z') \log \left\{ \frac{p(r|z')}{p(r)} \right\} \tag{2.31}$$

$$q_{min} = \lceil 1/\mathcal{C}_{channel} \rceil \tag{2.32}$$

The capacity estimation framework is general enough to be applied for hiding in any other transform domain (affects mapping $R \rightarrow Z$) and for any other attacks (affects mapping $Z \rightarrow Z'$).

There are two parameters to be properly estimated to maximize the embedding rate for zero BER - the hiding fraction λ and the code redundancy factor q. We separately optimize for λ and q - the estimation of the optimal λ is explained in Section 2.3. Let the maximum embedding rate obtained using an ideal channel code be $\mathcal{R}_{max}$ (2.33) while that practically obtained using the RA code, with optimal redundancy, is $\mathcal{R}_{prac}$ (2.35), both computed assuming knowledge of the

optimal hiding fraction $\lambda^\star$ - the threshold T in (2.22) is fixed at 30. The minimum redundancy factor, using RA codes, that produces zero BER for a given image and a given attack channel, called the optimum redundancy factor q_{opt}, is experimentally obtained and $\frac{1}{q_{opt}}$ is regarded as $\mathcal{R}_{undetectable}$ (2.34) - the rate achievable using a practical code without the statistical constraint. Using a fraction $\lambda^\star$ of the available terms for hiding, $\mathcal{R}_{prac}$ can be obtained from $\mathcal{R}_{undetectable}$ (2.35).

$$\mathcal{R}_{max} = \lambda^\star \cdot \mathcal{C}_{channel} \tag{2.33}$$

$$\mathcal{R}_{undetectable} = \frac{1}{q_{opt}} \tag{2.34}$$

$$\mathcal{R}_{prac} = \lambda^\star \cdot \mathcal{R}_{undetectable} \tag{2.35}$$

2.6.5 Results and Discussions

We run the experiments for a variety of design quality factors and attack quality factors. The results are averaged over 500 images for each case. The hiding parameters used for the DCT and DWT based domains are discussed in Section 2.6.1.

The average value of the maximum embedding rate with an ideal channel code, $\mathcal{R}_{max}$ (2.33), is compared with the rate obtained using the RA code, $\mathcal{R}_{prac}$ (2.35), for the following transform domains and attack scenarios:

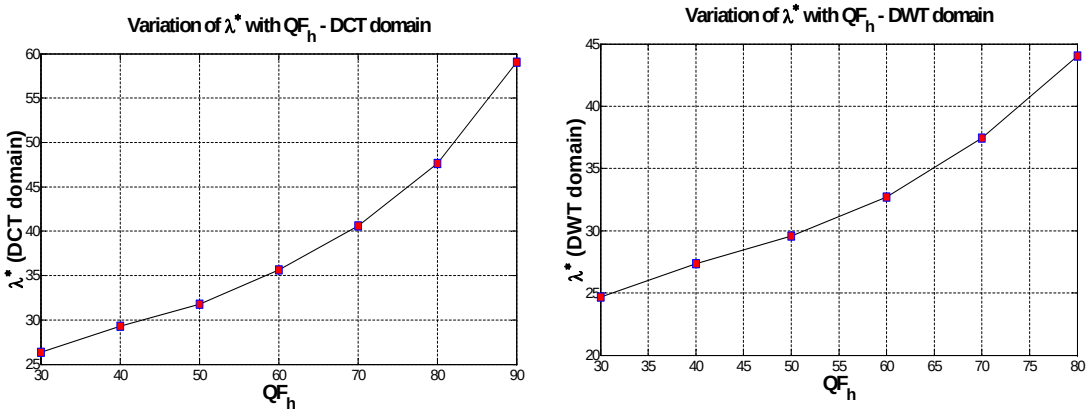
- DCT domain hiding, with JPEG attack

- DWT domain hiding, with JPEG attack
- DWT domain hiding, with JPEG-2000 attack

The hiding rates for both the DCT and DWT domains depend on the quality factor QF_h for hiding. As mentioned in Section 2.6.1, the quantization matrix of the DWT terms can be generated depending on QF_h. It has been shown [107] that when hiding occurs at a certain quality factor, the embedded data can be recovered after JPEG-based compression only if the attack quality factor, QF_a is the same or higher (less severe quantization) than the design quality factor QF_h. When the quantization at the attack stage is more severe than that used while data embedding, the redundancy factor needed for successful data recovery is so high that it makes the effective hiding rate very small. For JPEG-2000 based compression, the inverse compression ratio (ICR) is the ratio between the number of bits needed to represent the compressed image and the number of bits that represent the original image - higher ICR denotes less severe compression. The compression based attack is repeated for different values of ICR.

Our numerical findings are summarized in Figs. 2.11-2.14. In Fig. 2.11, it is seen that as the design quality factor for hiding, QF_h, increases, the optimum hiding fraction, $\lambda^\star$ (2.22) increases. As the design quality factor QF_h decreases, the JPEG quantization matrix consists of larger valued terms (coarser quantization) -

hence, the number of zero-valued DCT coefficients increases with a lower quality factor and coarser quantization. With a lower quality factor, the DCT PMF becomes more peaky near 0 - for example, $B_X(0)$ becomes much greater than $B_X(1)$ and $B_X(-1)$. Using (2.21), the hiding fraction λ_i for $i = \{-1, 1\}$ becomes smaller with lower QF_h due to the dominance of $B_X(0)$ over $B_X(1)$ and $B_X(-1)$. Hence, $\lambda^\star$ (2.22), the minimum of the λ_i terms, decreases. $\lambda^\star$ depends on the PMF of the quantized DCT elements, which is determined by the JPEG quantization matrix corresponding to QF_h and not the severity of the attack (QF_a).



(a) (b)

Figure 2.11: Variation of the optimum hiding fraction, $\lambda^\star$ (expressed as %), with the design quality factor QF_h, for DCT and DWT domains.

The rate $\mathcal{R}_{max}$ (2.33) is the product of $\lambda^\star$ and $\mathcal{C}_{channel}$. For a fixed QF_h, the variation of $\mathcal{R}_{max}$ with different attacks depends on $\mathcal{C}_{channel}$. In Fig. 2.12, it is seen that for DCT domain hiding, when the JPEG attack quality factor QF_a is

varied, the rate is initially high at $QF_a = QF_h$, and then drops with increased QF_a before rising again. With increased QF_a, the JPEG quantization becomes finer and hence, due to less severe attacks, the channel capacity is expected to increase. This trend holds in general except when QF_a equals QF_h - i.e. the attack is matched to the design quality factor for hiding. Hence, there is a valley in the $\mathcal{R}_{max}$ vs QF_a plots. The plot of $\mathcal{R}_{prac}$ follows the same trend as $\mathcal{R}_{max}$.

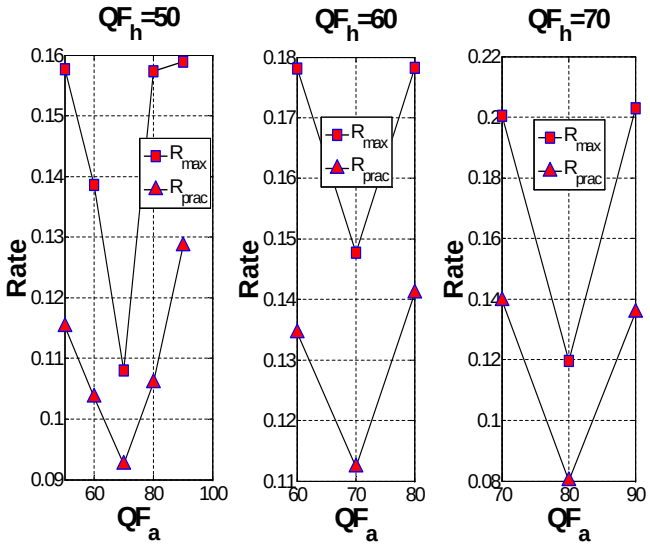


Figure 2.12: Variation of mean($\mathcal{R}_{max}$) and mean($\mathcal{R}_{prac}$) with the attack quality factor (QF_a) for JPEG attack, for different design quality factor QF_h, for DCT domain hiding.

However, when the hiding is in the DWT domain, the rate does not peak, even when QF_a is matched with QF_h. As QF_a is gradually increased from QF_h, the rate also increases as shown in Fig. 2.13.

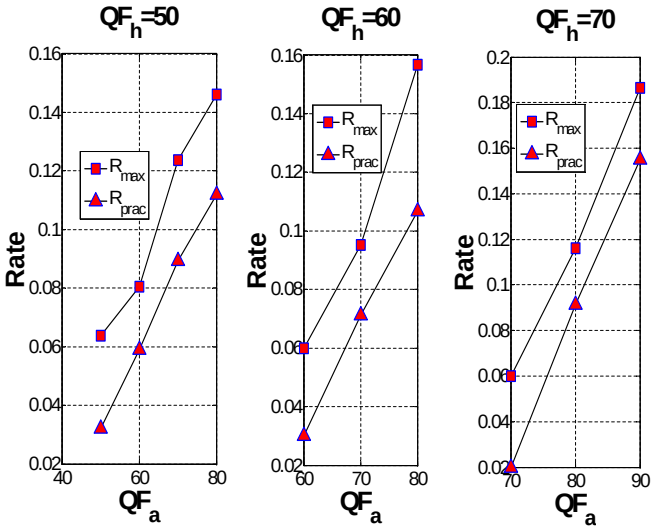


Figure 2.13: Variation of mean($\mathcal{R}_{max}$) and mean($\mathcal{R}_{prac}$) with the attack quality factor (QF_a) for JPEG attack, for different design quality factor QF_h, for DWT domain hiding. For fixed QF_h, higher QF_a indicates finer quantization and hence a less noisy channel, and so we achieve higher data-rates.

When the hiding is in the DWT domain and the attack is JPEG-2000 based compression, the rate increases with less severe compression, as shown in Fig. 2.14. With increased CR, the JPEG-2000 based attack is less severe and hence, the channel capacity and the rate increases.

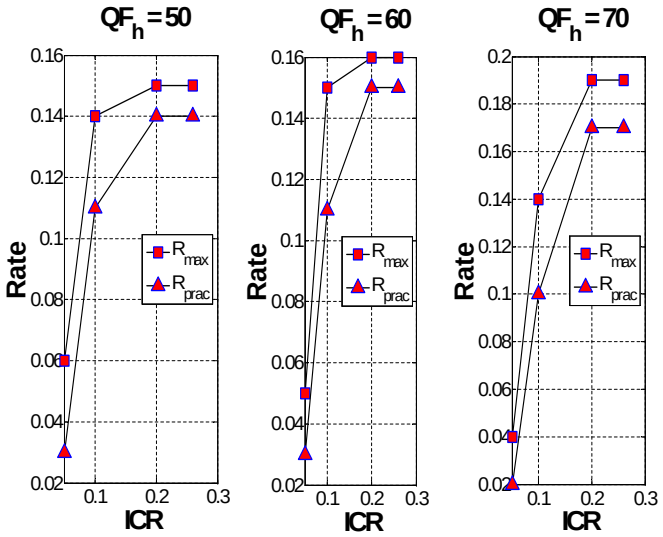


Figure 2.14: Variation of mean($\mathcal{R}_{max}$) and mean($\mathcal{R}_{prac}$) with the inverse compression ratio (ICR) for JPEG-2000 based attack, for different design quality factor QF_h, for DWT domain hiding. A higher value of ICR indicates less severe compression and a higher data-rate.

Chapter 3

YASS - a Randomized Block based Hiding Framework

The statistical restoration (SR) method, discussed in the last chapter, resists histogram-based steganalysis and allows theoretical analysis for computing the hiding capacity considering perceptual, statistical and attack constraints. However, though SR is amenable for theoretical analysis and capacity computation, it is highly detectable using current state-of-the-art steganalysis features.

The solution is to devise a stego scheme which does not explicitly disable detection by a single feature but instead works for a wider class of features. To motivate our proposed hiding method, we need to know what class of steganalysis features we hope to disable. A very powerful detection method is the self-

calibration process where the cover image statistics can be recovered by cropping a certain number of pixels from a JPEG compressed stego image. In this chapter, we present *Y*et *A*nother *S*teganographic *S*cheme (YASS), a method based on embedding data in randomized locations so as to disable the self-calibration process popularly used by blind steganalysis schemes. The errors induced in the bitstream decoding due to the fact that the stego signal must be *advertised* in a specific format such as JPEG, are dealt with by the use of erasure and error correcting codes. For the presented JPEG steganograhic scheme, it is shown that the detection rates of recent blind steganalysis schemes are close to random guessing, thus confirming the practical applicability of the proposed technique.

The randomized block-based hiding scheme de-synchronizes the detector's estimate of cover image statistics, which are computed over a regular 8×8 grid (similar to the JPEG block-based compression). This de-synchronization is the key to the security of YASS against steganalysis. YASS has generated significant interest among the steganalysis community and in the recent past, a number of detection schemes have been proposed to detect YASS. Thus, in the two-party game of steganography and steganalysis, the success of YASS has helped inspire steganalysis research, thus enriching both of the competing fields.

The baseline YASS method can be improved by further randomization and attack-aware embedding. These extensions significantly improve the hiding rate

without compromising on the steganalytic security. We also propose a method where the encoder can set the redundancy factor for repeat-accumulate (RA) code based error correction optimally (the optimal redundancy is the minimum possible value that ensures data recovery for a given noise channel), and the decoder then estimates the redundancy factor, without any side information, exploiting the structure of the RA-encoded sequences.

The outline of this chapter is as follows. Section 3.1 describes state-of-the-art steganalysis methods, going beyond simple histogram based features. The requirements of a blind steganalysis scheme are outlined in Section 3.2 and a detailed description of YASS is provided in Section 3.3. The desirable properties of an error correction coding scheme that allow perfect data recovery for YASS-based hiding are described in Section 3.4. Estimating the RA-code redundancy factor "optimally" at the encoder and then computing it independently at the decoder are explained in Section 3.5. Section 3.6 contains extensive experimental results using the YASS method and shows the trade-off of hiding rate vs detectability against state-of-the-art detectors. Section 3.7 describes two possible strategies to improve the YASS method, based on further randomization and attack-aware embedding. The improved performance using these variants of YASS is demonstrated in Section 3.8.

3.1 Steganography and Steganalysis - a Review

YASS is a steganographic method for JPEG images. We present a review of state-of-the-art approaches for hiding and detection for JPEG images. JPEG is arguably the most popular format for storing, presenting, and exchanging images. It is not surprising that steganography in the JPEG format, and its converse problem of steganalysis of JPEG images to find ones with hidden data, have received considerable attention from researchers over the past decade. There are many approaches and software available for JPEG steganography, which include OutGuess [89], StegHide [49], model-based steganography [93, 94], perturbed quantization [37], F5 [123], and statistical restoration [109, 110].

Approaches for JPEG steganography have focused on hiding data in the least significant bit (LSB) of the quantized discrete cosine transform (DCT) coefficients. In order to avoid inducing significant perceptual distortion in the image, most methods avoid hiding in DCT coefficients whose value is 0. To detect the presence of data embedded in this manner, steganalysis algorithms exploit the fact that the DCT coefficient histogram gets modified when hiding random information bits. There are various steganographic methods that resist histogram based detection - Provos's OutGuess [89], Eggers et al.ś histogram preserving data mapping (HPDM) [27], and statistical restoration (Chapter 2). Westfield's F5 al-

gorithm [123] increases, decreases, or keeps unchanged, the coefficient value based on the data bit to be hidden, so as to better match the host statistics. Sallee proposed a model based approach for steganography [93, 94] wherein the DCT coefficients were modified to hide data such that they follow an underlying model. Fridrich et al.ś perturbed quantization [37] attempts to resemble the statistics of a double-compressed image.

Many steganalysis schemes (see [82,83,119]) have been able to successfully detect the above steganographic techniques that match marginal statistics or models. They exploit the fact that higher order statistics get modified by data hiding using these stego methods. *It is known that, the higher order statistics, in general, are difficult to match, model, or restore.* Recently, blind steganalysis algorithms [8, 46, 70, 82, 83, 103, 121, 125] have been proposed that employ supervised learning to distinguish between the plain cover and stego images, and also identify the particular hiding algorithm used for steganography. These techniques bank on the fact that there are some image *features* that are consistently modified during the embedding process which can be used for classification. For the success of this approach, it is crucial that these features are very sensitive to the embedding changes, but insensitive to the image content. This requires a good model for natural images against which the suspected stego images can be evaluated.

In spite of the absence of good universal models, recent steganalysis algorithms have been very successful in using a *self-calibration* method to approximate the statistics of the original cover (see, for example, Pevny and Fridrich [82, 83], and Dabeer et al. [22]). The calibration method typically used for JPEG steganography is quite simple. A few rows and/or columns are cropped from the image so as to desynchronize it from the original JPEG grid and the resulting image is compressed again, which forms a good approximation of the cover image. The results reported in [83], a multi-class JPEG steganalysis method that employs such self-calibration, are close to perfect: the steganalyst can determine one out of six different stego algorithms employed for hiding with a detection accuracy of more than 95% in most cases, even at low embedding rates.

In this chapter, we present *Yet Another Steganographic Scheme* (YASS), a method for secure, active, steganography that can successfully resist the aforementioned blind steganalysis schemes. *The technique is based on a simple idea of embedding data in random locations within an image, which makes it difficult for the steganalyst to get a good estimate of the cover image features via the self-calibration process.*

YASS is nearly undetectable against state-of-the-art detection methods. Also, because an error correction coding framework is used, YASS provides robustness against distortion constrained attacks thus enabling active steganography.

3.2 Requirements of a Blind Steganalysis Scheme

Blind statistical steganalysis schemes use a supervised learning technique on *features* derived from plain cover as well as stego signals. This class of methods has been very successful in detecting steganographic methods available today. For example, detection results presented in [83] and also our own experiments indicate that popular JPEG steganographic schemes such as OutGuess [89], StegHide [49], model-based steganography [93, 94], and 1D statistical restoration schemes [109, 110] can be successfully detected. Following are the key ingredients that contribute to the success of these blind steganalysis schemes.

1. **Self-calibration mechanism**: Calibration process is used by the blind steganalysis schemes to estimate the statistics of the cover image from the stego image. For JPEG steganography, this is typically achieved by decompressing the stego image to the spatial domain followed by cropping the image by a few pixels on each side and compressing the image again using the same compression parameters. This gives a new JPEG image whose 8×8 grid is different from that in the original image, which provides an approximation for the DCT statistics of the cover image. Calibrated features are computed as the difference between the features computed from the stego image and from this approximated cover image.

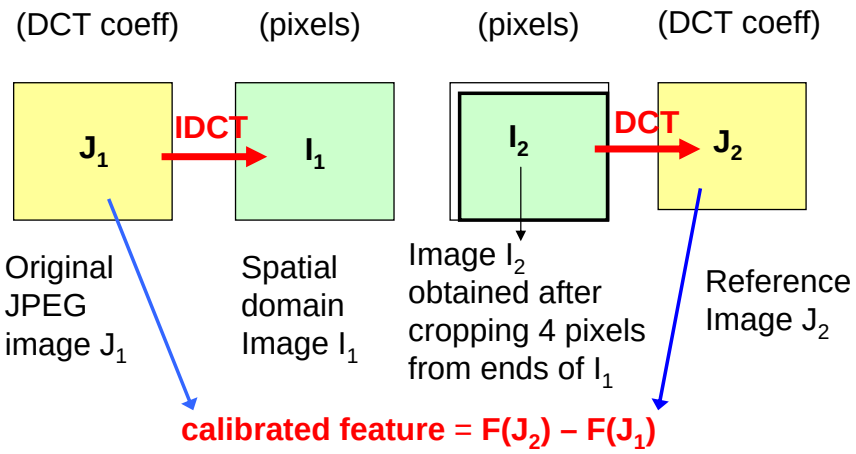


Figure 3.1: Description of Self-calibration mechanism - $F(\cdot)$ is the functional which computes the feature vector for a given image, IDCT = Inverse DCT

How does the self-calibration mechanism work (also explained in [58])?

(i) The spatial shift by 4 pixels ensures that the 8×8 recompression grid is different from the original JPEG compression grid.

(ii) Hence, the DCT coefficients obtained from J_2 (as in Fig. 3.1) are not affected by previous quantization and embedding in the DCT domain.

(iii) Therefore, the statistics of DCT coefficients from J_2 are close enough to the statistics of DCT coefficients from the original cover image.

2. **Features capturing cover memory**: Features that capture higher dimensional dependencies in the cover symbols are crucial in detecting the embedding changes because most steganographic schemes hide data on a per-symbol basis, and typically do not explicitly compensate or preserve

higher order statistics. Cover memory is incorporated into the feature vector in several ways, such as using joint statistics [42], Markov process [103], and so on.

3. **Powerful machine learning**: Use of powerful machine learning techniques and training with several thousand images ensures that even the slightest statistical variation in the features is *learned* by the machine.

The combination of the above factors allows the blind steganalysis schemes to detect the presence of hidden messages even when the embedding rates are quite low (see results in [83] and those presented in Section 3.6). The calibration process is perhaps the most important of the above that allows the steganalyst to get an accurate underlying model of the cover image despite not having access to it. Thus, the computed features succinctly capture the changes due to the hiding process rather than the image itself. With this approach, statistical steganalysis is successful even though good *universal* statistical models for images are not available. In the following section, we discuss a simple approach that can potentially defeat these steganalysis schemes.

3.3 YASS for JPEG Steganography

We now present a JPEG steganography scheme, YASS, that embeds data in 8×8 blocks whose locations are chosen randomly so that they do not coincide with the 8×8 grid used during JPEG compression. For enabling JPEG steganography, the images would still be advertised in JPEG format at a particular quality factor QF_a. This compression invariably leads to errors in the recovered bits, which are dealt with by using erasure and error correction coding framework.

Let the host image be denoted by a $M \times N$ matrix of pixel values. For simplicity, we assume that the image is grayscale (single channel); if it is not, we extract its luminance. Below we describe the main steps involved in this randomized block-based hiding method.

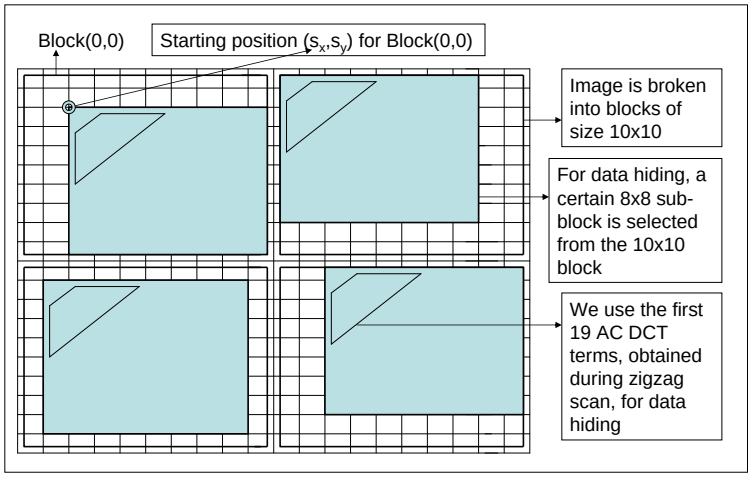
1. Divide the image into blocks of size $B \times B$, where B, which we call *big block size*, is always greater than 8, the size of a JPEG block. Thus we have $M_B \times N_B$ big blocks in the image where $M_B = \lfloor \frac{M}{B} \rfloor$ and $N_B = \lfloor \frac{N}{B} \rfloor$.

2. For each block (i, j) $(0 \leq i < M_B,\ 0 \leq j < N_B)$, we pseudorandomly select a 8×8 sub-block in which to hide data. The key for the random number generator is shared between the encoder and the decoder. The pseudorandom number generator determines the location of the smaller 8×8 block within the big block. This process is illustrated in Figure 3.2(a)

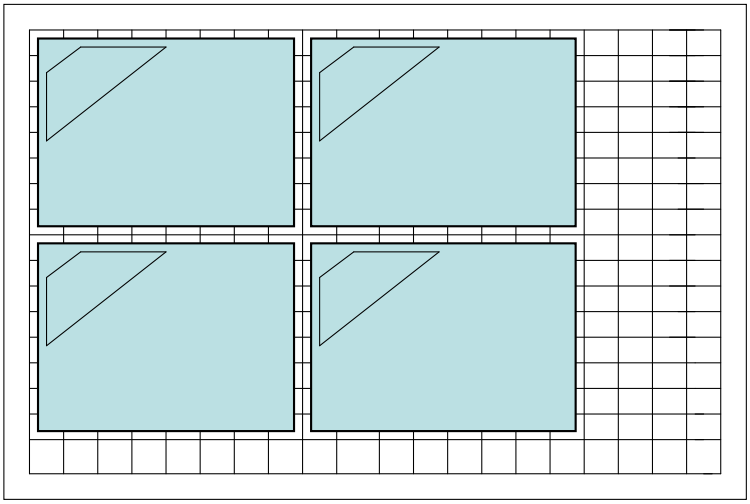
where four example blocks are shown, whose top leftmost corner (s_x, s_y) is randomly chosen from the set $\{0, 1, ..., B-8\}$. Figure 3.2(b) shows the blocks as seen by a steganalyst who gets *out-of-sync* from the embedding blocks, and cannot resynchronize even if the embedding mechanism is known.

3. For every 8×8 block thus chosen, we compute its 2D DCT and divide it by a JPEG quantization matrix at a *design* quality factor QF_h. Data is hidden in a predetermined band of low frequency AC coefficients using quantization index modulation [16]. For maintaining perceptual transparency, we do not hide in coefficients that quantize to zero by the JPEG quantizer (following the selective embedding in coefficients scheme proposed in [107]).

Note that using this approach, we can effectively de-synchronize the steganalyst so that the features computed by him would not directly capture the modifications done to the image for data hiding (see Figure 3.2). It should be noted that with this embedding procedure, we are reducing the embedding rate in two ways. First, some *real estate* of the image is wasted by choosing bigger blocks from which an 8×8 block is chosen to hide data. Note that the above framework can be further generalized to enable lesser *wastage*, by using larger big blocks and putting more 8×8 blocks into them. For example, we can use big blocks of size 33×33 and embed in sixteen ($\lfloor \frac{33}{8} \rfloor^2 = 16$) 8×8 blocks within. We report results



(a)



(b)

Figure 3.2: (a) YASS hiding methodology: Data is hidden in randomly chosen 8×8 blocks within a big block of size $B \times B$, with $B > 8$. This example uses $B = 10$. (b) The block structure as seen by a steganalyst, who gets *out-of-sync* with the blocks used during embedding, and cannot synchronize even if the embedding method and its parameters are known.

for such implementations as well (Section 3.6). The second cause of decrease in rate is that since the embedding grid does not coincide with the JPEG grid, there are errors in the received data which must be corrected by adding redundancy. This process is briefly described in the following section.

3.4 Coding Framework

In order to deal with the errors caused in the image due to JPEG compression, we use a coding framework using repeat-accumulate (RA) codes [25], similar to that proposed in [107]. This framework also allows us to hide in an adaptive fashion, avoiding coefficients that quantize to zero so as to control the perceptual distortion to the image.

For every block, we consider an embedding band comprising of the first n low frequency AC coefficients encountered during zigzag scan which forms the *candidate embedding band*. Data bits are hidden in a coefficient lying in the band if it does not quantize to zero using the JPEG quantizer at QF_h. Before the hiding process, the bit stream to be hidden is coded, using a low rate code, assuming that all host coefficients that lie in the candidate embedding band will actually be employed for hiding. A code symbol is *erased at the encoder* if the local adaptive criterion (of being quantized to zero) for the coefficient is not met. A

rate 1/q RA encoder is employed, which involves q-fold repetition, pseudorandom interleaving and accumulation of the resultant bit-stream. Decoding is performed iteratively using the sum-product algorithm [59]. Later, in Section 3.5, we show how q is properly chosen at the encoder and how it is again properly computed at the decoder. We postpone the detailed explanation of the RA framework till Section 3.5 where Fig. 3.3 explains the steps involved in RA-encoding and decoding.

The use of this coding framework for YASS provides the following advantages.

1. **Protection against initial JPEG compression**: Use of the coding framework provides error-free recovery of the hidden data after the initial JPEG compression so that the image can be advertised in the JPEG format.

2. **Flexibility in choosing hiding locations**: The coding framework allows dynamic selection of the embedding locations in order to limit the perceptual distortion caused to the host image during hiding. It is well known that embedding in DCT coefficients that quantize to zero can lead to visible artifacts in the stego image.

3. **Enabling active steganography**: The use of error correcting codes also provides protection against several distortion constrained attacks that an active warden might perform. The attacks that can be survived include

a second JPEG compression, additive noise, limited amount of filtering, and so on. *It should be stressed that we can survive only global attacks. If cropping or local/geometric attacks occur where input and output images are of different images or are de-synchronized with each other, decoding becomes impossible.*

3.5 Optimum Coding Redundancy Factor

For a RA-q system (RA coding using a redundancy factor of q), the optimum value of q, q_{opt}, ensures that we can hide the maximum data for a given image and noise channel, while we are still able to successfully recover the embedded data. Here, we show how the q value is properly set at the encoder side and how the decoder can independently compute the q for a given RA-codeword.

The serial concatenated turbo (RA) code based error correction is used in our data hiding setup - Fig. 3.3 shows the whole framework except for the iterative decoding part at the RA decoder. Let the total number of possible hiding locations in the image be ℓ. Using a redundancy factor of q, the maximum number of embeddable data-bits, denoted by N, equals $\lfloor \ell/q \rfloor$. The encoder repeats the N-bit data sequence u, as a whole, q-times (3.1), instead of repeating each bit q times. As is shown later, this makes it easier to compute q at the decoder using

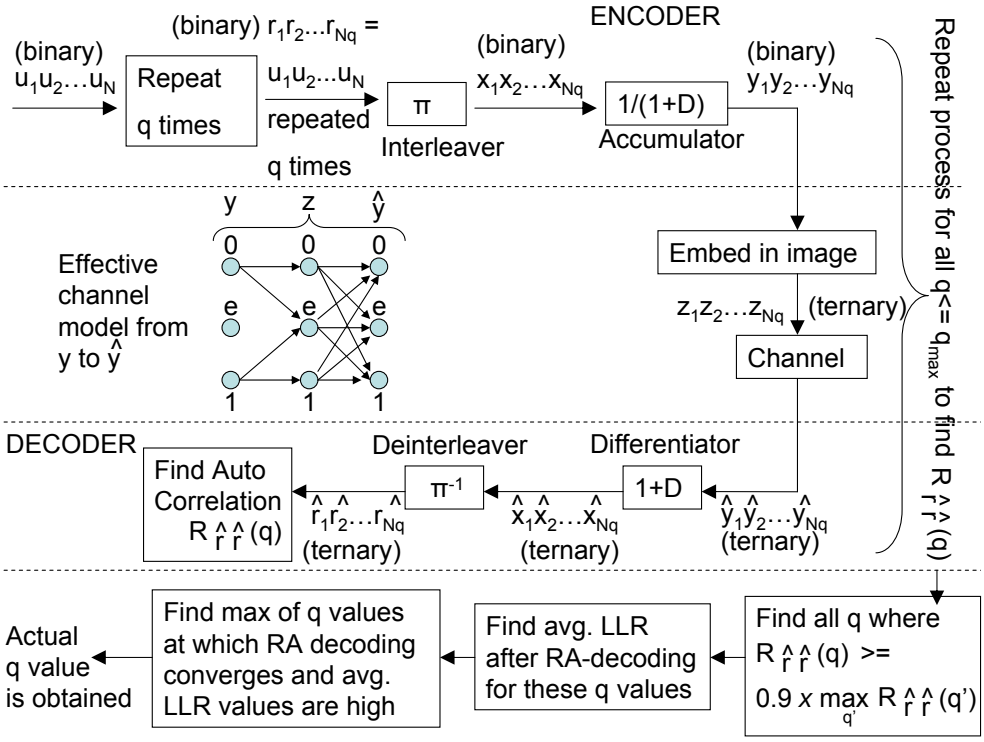


Figure 3.3: The data hiding system using RA-code based error correction, where q is efficiently estimated at the decoder

the auto-correlation (3.7) of the RA-encoded sequence.

Steps involved in mapping from u to y at the encoder

$$[r_{(i-1)N+1}, r_{(i-1)N+2}, \ldots, r_{iN}] = [u_1, u_2, \ldots, u_N],\ 1 \leq i \leq q \tag{3.1}$$

$$x = \pi(r),\ \text{where } \pi \text{ is the interleaver function} \tag{3.2}$$

$$y_1 = x_1,\ y_n = y_{n-1} \oplus x_n,\ 2 \leq n \leq Nq \tag{3.3}$$

After data embedding, we get a ternary sequence z of $\{0, 1, e\}$ based on what is actually embedded, where e denotes an erasure (Fig. 3.3). When a quantized discrete cosine transform (DCT) term in the image lies in the range [-0.5,0.5], an

erasure occurs - this maintains perceptual transparency [107]. For DCT terms of higher magnitude, every DCT term is quantized to the nearest odd/even integer to embed 1/0, respectively. The ternary sequence obtained from the hiding locations in the noisy received image, decoded using the same principles used while embedding by the encoder, is called $\hat{y}$.

We first discuss how the encoder decides on the right value of q. At the encoder side, the sender transmits a sequence u, embeds the RA-encoded sequence y in the image, subjects it to known attacks and finally obtains $\hat{y}$ from the image. Thus, by simulating the exact attack channel, the 2×3 transition probability matrix, $p(\hat{y}|y)$ can be computed. The capacity $\mathcal{C}$, for the channel that maps y to $\hat{y}$, is obtained by maximizing the mutual information $I(Y, \hat{Y})$ between the sequences y and $\hat{y}$, as has been shown before using (2.31). A discrete memoryless channel is assumed here.

From a capacity perspective, the minimum redundancy factor needed for perfect data recovery, *assuming an ideal channel code*, is $q_{min} = \lceil \frac{1}{\mathcal{C}} \rceil$. Thus, the minimum possible value of q_{opt} (q needed for perfect data recovery even after channel distortions) for the RA code is q_{min}. The sender simulates the decoder and attempts to recover the embedded data-bits by varying q. An upper limit (q_{max}) is set on the maximum redundancy factor to be used. Thus, the search for q_{opt}, needs to be done in the range $[q_{min}, q_{max}]$ - it will need at most $\log_2(q_{max} - q_{min})$

searches. *It is assumed here that the encoder knows the exact attack*, allowing it to compute q_{opt} precisely. In practice, the range of attacks may be known - the encoder can then design q_{opt} based on the worst-case attack.

In (3.4) and also later in (3.6), it is assumed that the output of $\oplus$ is an erasure if any of the input bits is erased.

Steps involved in mapping from $\hat{y}$ to $\hat{r}$ at the decoder

$$\hat{x}_1 = \hat{y}_1,\ \hat{x}_n = \hat{y}_n \oplus \hat{y}_{n-1},\ 2 \leq n \leq Nq \tag{3.4}$$

$$\hat{r} = \pi^{-1}(\hat{x}),\ \text{where } \pi^{-1} \text{ is the deinterleaver function} \tag{3.5}$$

Since the decoder knows the hiding method and assuming that the image size is not altered by the attacks, it can compute ℓ - the total number of possible hiding locations. Let the actual q value used by the encoder be q_{act}. If the decoder assumes q=q', the number of data-bits equals $\lfloor \ell/q' \rfloor$. In an ideal case, the sequence $\hat{r}$ will be exactly equal to r, where r consists of the input message sequence u, repeated as a whole. Thus, if $\hat{r}$ is circularly shifted by the assumed input message length $\lfloor \ell/q' \rfloor$, the normalized correlation between the original and the shifted sequences $R_{\hat{r},\hat{r}}(q')$ (3.7) will be very high if q'=q_{act}. In (3.6), $b(q')$ is the sequence obtained after performing element-wise $\oplus$ between the original and shifted sequences, where shift $k = \lfloor \ell/q' \rfloor$. $R_{\hat{r},\hat{r}}(q')$ (3.7) is the fraction of 0's in $b(q')$ (matches in two corresponding bits after $\oplus$ result in 0's), without considering the erasures.

$$\text{sequence } b(q') = (\{\hat{r}_1 \ldots \hat{r}_{kq'}\} \oplus \{\hat{r}_{kq'-k+1} \ldots \hat{r}_{kq'} \hat{r}_1 \ldots \hat{r}_{kq'-k}\}) \tag{3.6}$$

$$\text{and shift } k = \lfloor \ell/q' \rfloor \text{ is the assumed number of data-bits}$$

$$\text{correlation } R_{\hat{r},\hat{r}}(q') = \frac{\text{Number of 0's in } b(q')}{\text{Number of 0's and 1's in } b(q')} \tag{3.7}$$

$$\mathcal{Q}_{top} = \left\{ q' : R_{\hat{r},\hat{r}}(q') >= 0.9 \times (\max_{q' \leq q_{max}} R_{\hat{r},\hat{r}}(q')) \right\} \tag{3.8}$$

The correlation is also high when the shift equals a multiple of the actual message length, i.e. q'=$q_{act}/m,\ m \in \mathbb{Z}^+$. Apart from the correlation peaks at q_{act} and its sub-multiples, other peaks may occur due to errors and erasures. In the experiments, the set of q values, $\mathcal{Q}_{top}$ (3.8), at which the correlation exceeds 90% of the maximum $R_{\hat{r},\hat{r}}$ value, are selected - the 90% cutoff was empirically determined. The turbo decoder is then run for these q values and the log-likelihood ratios (LLR) [1] are computed for the extracted data-bits in each case. It is seen that due to a noisy channel, decoding may converge (two consecutive iterations produce the same output sequence) at values other than $q_{act}/m,\ m \in \mathbb{Z}^+$. *However, the LLR value, averaged over the data-bits, is high only when perfect decoding occurs.* It is seen that the maximum average LLR values occur only at q_{act} and its sub-multiples. Thus, the solution is to consider the maximum of these q values as q_{act}, as shown in Fig. 3.3. This method of estimating q for RA encoding is found to

[1] LLRs are formally defined in Section 4.4 and LLR computation is discussed in detail there in context of matrix embedding.

work even at high erasure rates ($\geq 95\%$).

Observations about the q-estimation method

(i) The use of auto-correlation based peaks reduces the search space for q while the average LLR-based measure, followed by taking the maximum, helps to identify the actual q.

(ii) For our experiments, the search range for q was $[2, 50]$.

(iii) *Though the correlation in $\hat{r}$ is used for q-estimation, this correlation is not detectable by an adversary;* $\hat{r}$ is obtained from $\hat{y}$ only after applying the deinterleaver (π^{-1}) - the key to generate π^{-1} is not known to an adversary.

(iv) The q-estimation method is generic enough to be used for any hiding scheme which uses RA-q based error correction.

3.6 YASS: Experiments and Results

We conduct a comprehensive set of experiments and present the results demonstrating the applicability of the presented approach. First, the results for the embedding capacity are presented for some standard images (Section 3.6.1). Next, in Section 3.6.2, the detection results of our scheme are presented using recent blind steganalysis methods for a couple of datasets (comprising of several thou-

sand natural images). The detection results of our method are compared with those of OutGuess and StegHide in Section 3.6.3, and show that our methods clearly outperform these approaches. It is also instructive to study which randomization is more useful for the hiding rate vs detectability trade-off, as studied in Section 3.6.4. While baseline YASS achieves security by randomized block-based locations, a naive scheme is used for comparison where standard 8×8 grid is used for hiding but the hiding occurs in randomly chosen low-frequency DCT coefficients for every 8×8 block. The embedding rate is kept the same for the baseline YASS and the random frequency coefficient selection based method.

3.6.1 Embedding Rates

In Table 3.1, we list the number of bits that can be hidden in several standard images using YASS with different embedding parameters. QF_h denotes the design quality factor used during hiding, and QF_a denotes the output or advertised image quality factor. Note that for the presented scheme, these two can be different and their values do affect the steganalysis performance. Also note that the number of bits reported in this section are before coding and can be recovered without any errors, even in the presence of distortion constrained attacks. For all the results presented here, we use 19 low-frequency DCT coefficients as the candidate embedding band.

Table 3.1: Number of information bits that can be hidden in some standard images of size 512×512. The big-block size $B = 9$.

$QF_h \rightarrow QF_a$	baboon	peppers	airplane	lena	couple	crowd
$50 \rightarrow 50$	1453	1295	2128	1702	1655	2979
$50 \rightarrow 75$	14896	6620	7448	7448	9930	11916
$75 \rightarrow 75$	1453	1805	2590	2128	2128	3972
$60 \rightarrow 75$	11916	6620	7448	6620	8512	9930

From this table, we see that the number of bits that can be embedded increases when $QF_a > QF_h$, however this also leads to slightly more successful steganalysis performance (as shown later in Tables 3.4-3.6), which gives us a trade-off between the embedding rate and the detection performance. When QF_a equals QF_h, there are more physical errors in the channel leading to a reduced embedding rate.

In Table 3.2, we study the effect of using different big-block sizes B on the embedding capacity. Here we also report the bits per non-zero coefficients (*bpnc*). It is seen from this table that using lower B provides higher embedding capacity, which is because we end-up using more *real estate* of the host image for hiding. We can ensure even higher percentage of image utilization by using larger big-blocks and putting greater number of 8×8 blocks within. For example, we can put four 8×8 blocks in a 17×17 block, thus making an effective big-block size of $B_{eff} = \frac{17}{2} = 8.5$. We experiment with block-size $B = (n\times 8 + 1)$ and use n^2 8×8 blocks for hiding, and report the results in Table 3.3. The embedding rate improves as we use more image area for hiding, but the detection results get worse (from the steganographer's viewpoint, see Table 3.6). Also, using a lower B_{eff}

does not always guarantee a higher embedding capacity because of the fact that more errors may be induced due to JPEG compression, which forces us to use a larger redundancy factor, q, of the RA code.

Table 3.2: Hiding rates for the 512×512 Lena image for different big-block sizes B. *bpnc* denotes bits per non-zero coefficients and *data-bits* denotes the number of hidden information bits.

$QF_h \rightarrow QF_a$	B	9	B	10	B	12	B	14
50 → 50	data-bits	1702	data-bits	1497	data-bits	882	data-bits	464
	bpnc	0.072	bpnc	0.064	bpnc	0.038	bpnc	0.020
50 → 75	data-bits	7448	data-bits	5491	data-bits	4189	data-bits	2736
	bpnc	0.213	bpnc	0.159	bpnc	0.118	bpnc	0.077
75 → 75	data-bits	2128	data-bits	2059	data-bits	1457	data-bits	794
	bpnc	0.059	bpnc	0.057	bpnc	0.040	bpnc	0.022
60 → 75	data-bits	6620	data-bits	5491	data-bits	3724	data-bits	2462
	bpnc	0.187	bpnc	0.158	bpnc	0.105	bpnc	0.069

Table 3.3: Hiding rates for the 512×512 Lena image for larger big-block sizes B = 9, 25, 49 and 81, which can incorporate several 8×8 blocks.

$QF_h \rightarrow QF_a$	B	9	B	25	B	49	B	81
50 → 75	data-bits	7448	data-bits	7834	data-bits	8064	data-bits	8064
	bpnc	0.213	bpnc	0.224	bpnc	0.235	bpnc	0.229
75 → 75	data-bits	2128	data-bits	2350	data-bits	2341	data-bits	1577
	bpnc	0.059	bpnc	0.065	bpnc	0.065	bpnc	0.044
60 → 75	data-bits	6620	data-bits	7050	data-bits	8064	data-bits	7258
	bpnc	0.187	bpnc	0.200	bpnc	0.231	bpnc	0.205

3.6.2 Detection Results

The steganographic security of our scheme is evaluated against the following four blind steganalysis schemes. The names in **bold** are the ones used to denote the steganalysis schemes in the tables presented here.

1. **Farid**: 72-dimensional feature vector based on moments in the wavelet domain [4].

2. **PF-23**: Pevny and Fridrich's 23-dimensional calibrated DCT-domain feature vector [82].

3. **PF-274**: Pevny and Fridrich's 274-dimensional calibrated feature vector that merges Markov and DCT features [83].

4. **DCT hist**: Histogram of DCT coefficients from a low-frequency band [110].

5. **Shi-324**: 324-dimensional feature vector, proposed by Shi et al. [103].

We conduct our experiments on two datasets: the first having 4500 images in compressed (JPEG) format and the other having 2000 images in uncompressed TIFF format. For each dataset, we use half the data for training and the other half for testing. The training and testing sets have an equal number of cover and stego images. The idea behind using datasets in different formats is to study the effect of using already compressed images verses uncompressed images for our method since it hides in a desynchronized 8×8 grid. As we see later, there is not much difference in the observed detection rates for the two datasets. As in all published blind steganalysis approaches, we train a support vector machine (SVM) on a set of known stego and cover images and use the threshold thus obtained, to distinguish between the cover and stego images at the time of testing.

The SVM classifier has to distinguish between two class of images: cover (class '0') and stego (class '1'). Let X_0 and X_1 denote the events that the actual image being observed belongs to classes '0' and '1', respectively. On the detection side, let Y_0 and Y_1 denote the events that the observed image is classified as belonging to classes '0' and '1', respectively. We use the probability of detection, P_{detect} as our evaluation criteria, which is defined as follows.

$$\begin{aligned} P_{detect} &= 1 - P_{error} \\ P_{error} &= P(X_0)P(Y_1|X_0) + P(X_1)P(Y_0|X_1) \\ &= \frac{1}{2}P_{FA} + \frac{1}{2}P_{miss}, \text{ for } P(X_0) = P(X_1) = \frac{1}{2} \end{aligned}$$

where $P_{FA} = P(Y_1|X_0)$ and $P_{miss} = P(Y_0|X_1)$ denote the probability of false alarm and missed detection respectively. Note that the above equation assumes an equal number of cover and stego images in the dataset. For the steganalysis results, we report P_{detect} upto 2 significant digits after the decimal point. An uninformed detector can classify all the test images as stego (or cover) and get an accuracy of 0.5. Thus, P_{detect} being close to 0.5 implies nearly undetectable hiding, and as the detectability improves, P_{detect} should increase towards 1.

In Tables 3.4 and 3.5, we present the detection accuracy obtained on the JPEG and TIFF image dataset respectively. Note that even for the TIFF dataset, the output is always a JPEG image at an advertised quality factor. For this dataset,

Table 3.4: JPEG dataset: Steganalysis results for randomized block based hiding, when big-block size B is varied. It can be seen that the detection is random for most of the configurations.

QF_h (used for hiding)	QF_a (advertised QF)	Steganalysis Method	Detection accuracy: P_{detect}			
			B=9	B=10	B=12	B=14
50	50	Farid	0.52	0.51	0.52	0.51
50	75	Farid	0.55	0.55	0.54	0.51
75	75	Farid	0.52	0.51	0.52	0.51
50	50	PF-23	0.56	0.55	0.54	0.54
50	75	PF-23	0.59	0.59	0.56	0.60
75	75	PF-23	0.53	0.57	0.53	0.52
50	50	PF-274	0.58	0.56	0.53	0.55
50	75	PF-274	0.77	0.79	0.74	0.65
75	75	PF-274	0.59	0.60	0.62	0.54
50	50	DCT-hist	0.53	0.53	0.51	0.53
50	75	DCT-hist	0.64	0.64	0.60	0.54
75	75	DCT-hist	0.55	0.54	0.55	0.53
50	50	Shi-324	0.57	0.51	0.55	0.54
50	75	Shi-324	0.75	0.65	0.60	0.55
75	75	Shi-324	0.54	0.55	0.53	0.53

the plain cover images are JPEG compressed at QF_a during the training as well as testing. It can be seen from the tables that our scheme is undetectable using any of the steganalysis features. The only time the detection is not completely random is when the design quality factor is significantly lower than the advertised one ($QF_h = 50$ and $QF_a = 75$). We also present, in Table 3.6, the steganalysis results when larger big-block sizes are used, so as to incorporate more 8×8 blocks within. P_{detect} remains close to 0.5 in most cases, but for PF-274, it increases as B increases (in Table 3.6), since more area of the image is used for hiding when employing larger big-blocks. To elucidate, for big-block $B \in [9, 15]$, the effective

Table 3.5: TIFF dataset: Steganalysis results for randomized block based hiding, when the big-block size B is varied. It can be seen that the detection is random for most of the configurations.

QF_h (used for hiding)	QF_a (advertised QF)	Steganalysis Method	Detection accuracy: P_{detect}			
			B=9	B=10	B=12	B=14
50	50	Farid	0.51	0.51	0.51	0.51
50	75	Farid	0.53	0.51	0.52	0.51
75	75	Farid	0.50	0.51	0.51	0.51
50	50	PF-23	0.53	0.55	0.53	0.53
50	75	PF-23	0.59	0.66	0.53	0.53
75	75	PF-23	0.54	0.51	0.53	0.53
50	50	PF-274	0.52	0.56	0.55	0.51
50	75	PF-274	0.72	0.81	0.65	0.60
75	75	PF-274	0.56	0.56	0.52	0.53
50	50	DCT-hist	0.51	0.53	0.52	0.48
50	75	DCT-hist	0.60	0.59	0.56	0.56
75	75	DCT-hist	0.52	0.52	0.51	0.52
50	50	Shi-324	0.55	0.55	0.53	0.53
50	75	Shi-324	0.65	0.64	0.56	0.54
75	75	Shi-324	0.55	0.51	0.51	0.50

big-block size B_{eff} (that includes a 8×8 block) equals B. For $B = (8.n + 1)$, where n is an integer greater than 1, then n^2 8×8 blocks can be fitted inside a $B \times B$ big-block and so, B_{eff} equals $8\frac{1}{n}$. Thus, as n increases, B_{eff} decreases and more image area is used for data embedding.

3.6.3 Comparison with Other Hiding Methods

In Table 3.8, we present a comparison of our steganographic scheme, YASS, to OutGuess [89] and StegHide [49]. To enable fair comparison, we must use the same hiding rate for all the schemes. This is complicated by the fact that

Table 3.6: Using larger big-block size B: Steganalysis results for randomized block based hiding on the TIFF image dataset, for block-size B = 9, 25 and 49. For 9×9, 25×25 and 49×49 blocks, we use 1, 9 and 36 8×8 sub-blocks respectively for hiding.

QF_h (used for hiding)	QF_a (advertised QF)	Steganalysis Method	Detection accuracy: P_{detect}		
			B=9	B=25	B=49
50	50	Farid	0.51	0.50	0.50
50	75	Farid	0.53	0.51	0.52
75	75	Farid	0.50	0.49	0.51
50	50	PF-23	0.53	0.53	0.57
50	75	PF-23	0.59	0.58	0.67
75	75	PF-23	0.54	0.53	0.53
50	50	PF-274	0.52	0.57	0.59
50	75	PF-274	0.72	0.73	0.78
75	75	PF-274	0.56	0.58	0.68
50	50	DCT-hist	0.51	0.50	0.51
50	75	DCT-hist	0.60	0.56	0.50
75	75	DCT-hist	0.52	0.51	0.50
50	50	Shi-324	0.55	0.53	0.52
50	75	Shi-324	0.65	0.58	0.57
75	75	Shi-324	0.55	0.52	0.51

Table 3.7: Using larger big-block size B: Steganalysis results for randomized block based hiding on the TIFF image dataset, for block-size B = 9, 25 and 49. For 9×9, 25×25 and 49×49 blocks, we use 1, 9 and 36 8×8 sub-blocks respectively for hiding - using cropping of 1 pixel per dimension.

QF_h (used for hiding)	QF_a (advertised QF)	Steganalysis Method	Detection accuracy: P_{detect}		
			B=9	B=25	B=49
50	50	PF-23	0.51	0.55	0.63
50	75	PF-23	0.56	0.67	0.76
75	75	PF-23	0.51	0.57	0.61
50	50	PF-274	0.53	0.58	0.70
50	75	PF-274	0.67	0.76	0.86
75	75	PF-274	0.57	0.57	0.70

YASS uses an error correcting code whose redundancy factor, q, determines the actual number of information bits hidden in the image. Experiments indicate that, for images in our dataset, the redundancy factor required to ensure zero BER was in the range 10-40, which depends on the particular image and the level of JPEG compression involved. Hence we present results where the amount of data embedded using OutGuess and StegHide corresponds to the hiding rate obtained using $q = 10$ and $q = 40$. It can be seen from the table that both OutGuess and StegHide are almost completely detectable (especially when using PF-23, PF-274, and DCT-hist features), but YASS is not detectable at equivalent hiding rates.

We performed experiments using the F5 steganographic scheme and we found that it was undetectable at the low embedding rates using the 23 [34, 82] and 274-dimensional feature vectors [83], proposed by Fridrich and Pevny. Though the YASS scheme and F5 perform similarly, w.r.t. detectability at similar embedding rates, the YASS scheme is more robust due to the RA-coding based error correction framework. In matrix embedding (matrix embedding is discussed in detail in Chapter 4) based F5 method, errors in 1 term may lead to multiple errors - for the error resilient framework used in YASS, errors in some coefficients can be taken care of using the added redundancy. Thus, YASS scheme is more robust.

Table 3.8: Steganalysis results for comparing the randomized block based scheme (YASS) with OutGuess (OG) and Steghide (SH) schemes, used at rates of $\frac{1}{10}$ and $\frac{1}{40}$, for the TIFF image dataset. For OutGuess and Steghide, the images are JPEG compressed using a quality factor of QF_a before being presented to the steganographic scheme. Note that the QF_h parameter is applicable only for the YASS scheme.

QF_h	QF_a	Steganalysis Method	Detection accuracy: P_{detect}				
			YASS	OG-$\frac{1}{10}$	SH-$\frac{1}{10}$	OG-$\frac{1}{40}$	SH-$\frac{1}{40}$
50	50	Farid	0.51	0.74	0.50	0.77	0.52
50	75	Farid	0.53	0.59	0.50	0.50	0.55
75	75	Farid	0.50	0.59	0.50	0.50	0.55
50	50	PF-23	0.53	0.98	0.78	0.97	0.80
50	75	PF-23	0.59	1.00	0.99	0.99	0.99
75	75	PF-23	0.54	1.00	0.99	0.99	0.99
50	50	PF-274	0.52	1.00	0.98	1.00	0.96
50	75	PF-274	0.72	1.00	1.00	1.00	1.00
75	75	PF-274	0.56	1.00	1.00	1.00	1.00
50	50	DCT-hist	0.51	0.95	0.59	0.94	0.60
50	75	DCT-hist	0.60	1.00	0.91	1.00	0.93
75	75	DCT-hist	0.52	1.00	0.91	1.00	0.93

3.6.4 Comparison with Standard Hiding at Same Rate

In the previous sections, it is seen that our proposed steganographic approach performs quite well against blind steganalysis schemes that we have tested against. We must, however, note that the improved steganographic security comes at the cost of reduced embedding rate. To further investigate whether the good performance of YASS is simply because of reduced rate or not, we present another *simpler* and *naive* extension of the idea of embedding in random locations: we now

embed data in randomly chosen low-frequency AC DCT coefficients computed using the original JPEG grid.

- YASS - the 8×8 pixel-domain blocks are pseudo-randomly selected, while the location of the frequency domain coefficients used for hiding in a 8×8 block remains fixed.

- Randomized Frequency (RF) based naive scheme - the 8×8 pixel-domain blocks are regularly selected while the frequency domain coefficients used for hiding inside the block are pseudo-randomly selected.

This approach would incur minimal distortion to the original cover during the hiding process, and hence would be an ideal *control* experiment for testing our scheme. The results are reported in Table 3.9, where we compare the YASS embedding scheme with this randomized frequency (RF) scheme for three hiding rates: 2 out of 19, 1 out of 19, and 1 out of 38 coefficients. It can be seen that the naive RF scheme performs quite well; however, the performance of YASS is consistently better.

Table 3.9: Comparison of steganalysis results for randomized block (RB) hiding (i.e. YASS) with $B = 9$, and randomized frequency (RF) based hiding.

QF_h (used for hiding)	QF_a (advertised QF)	Steganalysis Method	Detection accuracy: P_{detect}			
			RB (YASS)	RF 1/38	RF 1/19	RF 2/19
50	50	Farid	0.51	0.66	0.65	0.67
75	75	Farid	0.50	0.52	0.58	0.67
50	50	PF-23	0.53	0.69	0.83	0.92
75	75	PF-23	0.54	0.58	0.72	0.83
50	50	PF-274	0.52	0.71	0.79	0.90
75	75	PF-274	0.56	0.62	0.75	0.82
50	50	DCT-hist	0.51	0.55	0.68	0.86
75	75	DCT-hist	0.52	0.54	0.59	0.80

3.7 Extensions to YASS

Continuing to focus on JPEG image steganography, we present a further study on YASS with the goal of improving the rate of embedding. Following are the two main improvements that have been proposed.

1. **Further randomization**: We present results for a *more randomized* scheme in which the quantization matrix used on the transform domain coefficients has been randomized. The design quality factor is varied among the different image blocks and its value, for a given block, is determined based on the local image variance.

2. **Attack-aware embedding**: These schemes utilize the fact that the embedded image undergoes JPEG compression before it is "advertised". As explained in [108], this compression acts as an attack to the embedding sys-

tem which causes errors and reduces the embedding rate. Here, we exploit the fact that this *attack* is known at the encoder and its effect can be reduced via an iterative embedding process.

The results obtained are quite encouraging. The rate of embedding, measured in bits per non-zero coefficients (bpnc), has been improved while maintaining the undetectability against recent blind steganalysis schemes. *An important result is that the attack-aware embedding schemes outperform the F5 algorithm [123], which uses matrix embedding, in terms of the observed detection rates when data is hidden at equivalent embedding rates.* The merged DCT and Markov features proposed by Pevny and Fridrich [83] have been used in these tests, which have also been found to be among the most successful steganalysis schemes for detecting JPEG image steganography.

3.7.1 Approaches Used to Increase the Embedding Rate

We now present two different approaches that we have studied with the goal of improving the embedding rate while maintaining the undetectability of the schemes against recent blind steganalysis techniques.

The first approach is a natural extension of the YASS framework, in which, in addition to choosing randomized locations, we also vary the design quality factor to be used for hiding per block. *It is observed that varying the quality factor based*

on the local variance increases the hiding rate, compared to a random variation. In this manner, we expect to cause different statistical changes to different parts of an image, and a steganalysis scheme that computes statistics over the whole image is likely to get more confused (as compared to the prior scheme in which only the hiding location is randomized). It should be noted that the range of variation of the hiding parameters is adjusted such that the perceptual transparency of the stego image is maintained.

The second approach, which provides better improvement in rate, utilizes the fact that the JPEG compression "attack" that causes errors in the embedded bitstream is known at the encoder. Note that both the approaches can in principle be combined into a third approach which can potentially provide further rate improvement. Below we provide an overview of the two approaches.

3.7.2 Further Randomization: A Mixture based Approach

As described in Section 3.3, the main idea in YASS is to randomly select a 8×8 block in a $B \times B$ *big block* and hide in a certain frequency band in the 8×8 block. To further increase the randomness in the hiding scheme, the block-based hiding parameters can also be varied. Here, we vary the design quality factor for hiding in the different blocks. Thus, a *mixture* of various parameters is used for hiding.

The use of this increased randomization means that this has to be conveyed as side information to the decoder - by the sharing of more secret keys between the sender and the receiver. Earlier, in the randomized block based scheme, a pseudo random sequence was known to both the encoder and decoder - through a common key. Thus, the exact locations of the randomized blocks were available to the decoder. Here, apart from the randomized locations, the exact allocation of the hiding quality factor among the different blocks (hiding quality factor can take any one of the mixture values) can be made available to the decoder, by a similar pseudo random sequence based approach.

Intuitively, it appears that a random allocation of hiding parameters will make the hiding method less detectable by increasing the randomness; experiments showed that though the method became less detectable, the bpnc also decreased. *To increase the bpnc, we then explored the use of image-specific information in the various blocks.* We have explored the following means of varying the quantization matrices.

- *Random variation*: Here we choose the quality factor randomly from a predefined set. The secret key used for the choice of the quality factor as well as the predefined set of quality factors used during hiding are shared with the decoder. Pseudo random selection is the simplest and a natural way to vary the hiding parameter.

- *Image adaptive variation*: We also explore the use of a couple of image-adaptive methods for choosing the embedding parameters. Note that if a content-specific choice of the the hiding parameter is made, the information cannot be conveyed to the decoder. In this scenario, the decoder must determine the hiding parameter value based on the image content itself, which may vary due to the embedding process as well as any attacks. Thus the criteria used in determining the embedding parameter must be robust to compression attacks. Note that these schemes are similar in spirit to the entropy thresholding scheme [107].

  - *Based on coefficient count:* the quality factor used for hiding is varied based on the count of the non-zero coefficients at a selected quality factor. Our experiments indicate that though this leads to greater number of embedded bits, the advantage is lost due to an increased number of instances when the decoder makes a mistake in guessing the right embedding parameter.

  - *Based on block variance:* the quality factor is varied based on the variance of the block. Unlike the coefficient count, the block variance was found to be a more robust criterion for the choice of the hiding parameter. The value of the variance did not vary much, between the

cover and stego images, in presence of attacks, and hence, the decoder made fewer mistakes in guessing the hiding parameters correctly. The variance values are divided into as many partitions as the number of different quality factors in the mixture based scheme. Note that the choice of the variance based partitions, where each partition corresponds to a different design quality factor, is highly non-uniform; most of the blocks have very low variance for natural images and only a small fraction has variance significantly greater than zero. The choice of these partitions is made experimentally so as to ensure that the overall embedding rate improves.

Varying the Design Quality factor Depending on the Variance

For natural images, most blocks have essentially a higher low-frequency content. The distribution of the variance values among the different image blocks is essentially a tapered distribution - higher for the low variance values and a steady decrease with increasing variance values. For simplicity, let us consider a 3-mixture case with quality factors QF_1, QF_2 and QF_3, where $QF_1 < QF_2 < QF_3$. Let the variance be partitioned using $[0, x_1, x_2, \infty)$, where

- QF_3 is allocated to zone $[0, x_1)$
- QF_2 is allocated to zone $[x_1, x_2)$

- QF_1 is allocated to zone $[x_2, \infty)$

Due to the high concentration of values near 0 and the steady decrease in the distribution with increase in variance, the number of blocks having variance in the range $[0, x_1)$ is maximum and the number of blocks with variance in the $[x_2, \infty)$ is minimum. When hiding is done at higher QF, it is generally more difficult to detect it, as shown later in Tables 3.10-3.11. Hence, the highest QF (QF_3) is allocated to the zone with maximum number of blocks, $[0, x_1)$. Also, for increasing the hiding capacity, the embedding rate is increased for a certain QF if the number of erasures is less for that QF (i.e. there are more coefficients available for hiding), and if the attack becomes less severe. Now, if the mixture used for hiding has QF values of 50, 60 and 70, we should use a QF of 50 for those blocks where there are a substantial number of non-zero terms available for hiding, i.e. zone $[x_2, \infty)$. For QF=70, the embedding rate at this design QF is quite low and so, we do not lose much, from an embedding rate perspective, by using QF=70 for blocks with variance values in $[0, x_1)$.

In Section 3.8, the mixture based scheme where there is a random allocation of the QF per block is called a Mixture-random method; the scheme with the local variance based QF allocation is called a Mixture-variance method.

3.7.3 Attack-aware Iterative Embedding

A significant factor contributing to the reduction in the embedding rate of YASS is the JPEG compression that the image must undergo before it is advertised. There are errors caused in the embedded bitstream because the 8×8 blocks employed in data hiding does not coincide with the JPEG blocks. Hence the JPEG compression is thought of as an *attack* to the data hiding system, and an erasures and errors correction coding framework [107] is employed to deal with these errors.

The good news in this scenario is that the attack is carried out at the encoder itself before releasing the image, and hence is *known* to the encoder once the hidden image is known. However, the bad news is that the JPEG compression attack is highly correlated with the host signal and hence there is no simple framework to predict the attack and account for it beforehand. Hence we explore an iterative embedding procedure, in which we repeat the embedding and attack in an iterative manner with the hope that the system will converge towards lower error rate. The embedding algorithm is run in the same way as YASS, followed by the initial JPEG compression, which, as stated earlier, acts as an attack. Then the same hiding procedure is repeated on the resulting attacked images so as to *correct* the errors and erasures caused due to the attack. The raw error rate gets reduced after one such pass, which then translates to higher information bit embedding

rate. The method is referred to as M1 in Section 3.8 - in the experiments, one iteration of the attack-aware embedding has been used.

This method reduces the raw bit error rate, which reduces the required redundancy for the error correcting code thus improving the embedding rate. Why is the hiding rate improved on repeating the JPEG compression? If the compression is repeated, then the erasure rate increases while the error rate decreases (due to increased erasures). It is shown in [3] that when the erasure rate is increased to a certain extent, the hiding rate is increased. When JPEG compression is done once, then due to quantization and rounding, the low frequency content is increased. Therefore, there are more coefficients in the erasure zone when JPEG compression is performed a second time.

3.8 YASS Extensions: Experiments and Results

The aim of these experiments is to compare the embedding rate of the newly proposed hiding methods (mentioned in Section 3.7.1) with the DCT-based YASS system (Section 3.3) as well as with other competing steganographic algorithms such as F5 [123]. We vary the hiding parameters of the schemes such that they yield approximately the same (low) detection accuracy while testing on the same dataset with the same steganalysis techniques.

The steganographic security of our scheme is evaluated against the following blind steganalysis schemes. Note that of the five schemes in Section 3.6, only two are chosen here, because the DCT-based YASS scheme is undetectable even at higher embedding rates, for the other three schemes.

1. **PF-274**: Pevny and Fridrich's 274-dimensional feature vector that merges Markov and DCT features [83].

2. **Shi-324**: 324-dimensional feature vector, proposed by Shi et al. [103].

We conduct the steganalysis experiments on a JPEG image dataset having 4500 images, from MM270K database [2]. Half of the images are used for training and the other half for testing. We train a support vector machine (SVM) on a set of known stego and cover images and use the classifier thus obtained, to distinguish between cover and stego images in the test dataset.

For the DCT-based YASS approach, when the design quality factor used for hiding, QF_h, and the attack quality factor, QF_a, were approximately the same, the hiding was undetectable - however the embedding rate was very low. When we increase the hiding rate by making QF_a much larger than QF_h, the detection accuracy increased significantly for some steganalysis schemes. When the detection accuracy using a certain steganalysis method is less than 0.60, we consider

[2] http://www-2.cs.cmu.edu/yke/retrieval

Table 3.10: Variation in bpnc with variation in QF_h : QF_a is set at 75. The bpnc is maximum for QF_h=50. For lower QF_h, bpnc decreases as more coefficients get erased; for higher QF_h, the attack due to QF_a is more severe and introduces more bit errors. Big block size B is set to 9.

QF_h	40	50	55	60	70
bpnc	0.1941	**0.2031**	0.1956	0.1839	0.1073
PF-274	0.85	0.77	0.69	0.66	0.58
Shi-324	0.86	0.75	0.63	0.61	0.55

the hiding scheme to be statistically undetectable. For computing the embedding rate, we report the number of bits embedded per non-zero coefficients (bpnc) averaged over 500 images.

In Table 3.10, we show how the bpnc and the detection accuracy (P_d), using the aforementioned steganalysis methods, vary with change in the design QF_h used for hiding, for the YASS scheme (with the same QF_h being used for hiding for all the blocks). Why does the bpnc peak at QF_h of 50? As QF_h increases from 40 to 70, the effective channel noise increases, if QF_a is fixed at 75. On the other hand, the erasure rate decreases as QF_a increases from 40 to 70 and hence, more coefficients are available for hiding. *Due to these two opposing effects, it is seen that the bpnc peaks at a QF_h value between 40 and 70, which is 50.*

From this experiment, we see that using the original YASS scheme, we obtain a reasonably low detection accuracy (< 0.60) for QF_h=70, but there the bpnc is as low as 0.1073. To find the nature of the variation of the detection accuracy at

Table 3.11: Variation of detection accuracy with change in the design quality factor for hiding, QF_h for DCT-based YASS scheme, with *big-block size* B=9. After hiding, the image is JPEG compressed to an attack quality factor of QF_a=75 and this same quality factor is used for steganalysis. For ensuring that the detection accuracy $P_d \leq 0.60$, we need QF_h to be ≥ 69.

Steganalysis Method	Detection accuracy: P_d					
	QF_h=55	QF_h=60	$QF_h = 65$	QF_h=67	$QF_h = 69$	$QF_h = 70$
PF-274	0.69	0.66	0.63	0.63	0.60	0.58
Shi-324	0.63	0.61	0.58	0.58	0.56	0.55

QF_h values close to 70, we have the following table (Table 3.11). The big block size B is set to 9.

Now, we employ a mixture-based scheme for hiding, where we experiment with both the randomized and variance-based allocation of QF per block, as shown in Table 3.12. This table shows that there is significant increase in bpnc on using Mixture-variance over Mixture-random for hiding. From a detection perspective, P_d remains almost the same for both the Mixture schemes (for both the random allocation and variance based allocations).

For the Mixture-variance scheme, the question arises as to what partitioning of the variance range is better for the bpnc-detection trade-off. In Table 3.13, it is seen that as the size of the last partition is increased, the effective or average QF_h decreases slightly and P_d increases slightly. Thus, the changes are not significant enough - we use a partitioning of $[0, 1, 4, \infty]$ for all the experiments with Mixture-variance in this section. The interpretation of $[0, 1, 4, \infty]$ where the mixture has

Table 3.12: Variation in bpnc with variation in QF_h - e.g. in the first row, the average $QF_h \approx 60$ for all the 3 methods. The 1^{st} method uses a constant QF_h for all the blocks. The 2^{nd} method uses a mixture of {50,60,70} for hiding, with the QF per block being decided randomly. The 3^{rd} method uses the same mixture, of {50,60,70}, for hiding, with the QF per block being decided based on the local block variance. It is seen that the embedding rate (bpnc) is higher with the mixture scheme, only if the QF allocation is done based on the block variance. B is set to 9.

	YASS ($QF_h = 60$)	Mixture-random (50-60-70-rand)	Mixture-variance (50-60-70-var)
bpnc	0.1839	0.1704	**0.1930**
PF-274 (P_d)	0.66	0.59	0.59
Shi-324 (P_d)	0.61	0.58	0.58
	YASS ($QF_h = 70$)	Mixture-random (60-70-80-rand)	Mixture-variance (60-70-80-var)
bpnc	0.1073	0.0763	**0.1365**
PF-274 (P_d)	0.58	0.56	0.56
Shi-324 (P_d)	0.55	0.54	0.54

$QF_h = 50$, 60 and 70, is QF of 70/60/50 is used for the zone [0,1)/[1,4)/[4, ∞) of block-based variance values, respectively.

In Table 3.14, we compare the performance of the schemes (constant QF-based YASS, Mixture-random, Mixture-variance) for different big block sizes, B=9 and 25. Also, the iterative embedding scheme (M1) is used alongwith the Mixture-random and Mixture-variance schemes to further increase the embedding rate. As explained in [108], the embedding rate increases with B=25, as compared to B=9, at the cost of slight increase in P_d.

Then, we study the performance of the F5 scheme [123], at different embedding rates (in terms of bpnc), in Table 3.15.

Table 3.13: The effect of varying the partitioning of the variance values is studied. Here, Avg. QF_h refers to the average QF obtained, after considering the QF allocation over all the blocks. Note that in this table, the interpretation of, $[0, 1, 4, \infty]$ where the mixture has $QF_h = 50$, 60 and 70, is that a QF_h of 70/60/50 is used for the zone [0,1)/[1,4)/[4, ∞) of block-based variance values, respectively.

partitions	50-60-70	Avg. QF_h	PF-274 (P_d)	Shi-324 (P_d)
$[0, 1, 2, \infty)$	0.1952	58.85	0.60	0.59
$[0, 1, 4, \infty)$	0.1930	59.61	0.59	0.58
$[0, 1, 6, \infty)$	0.1918	60.10	0.59	0.58
partitions	60-70-80	Avg. QF_h	PF-274 (P_d)	Shi-324 (P_d)
$[0, 1, 2, \infty)$	0.1429	68.02	0.55	0.55
$[0, 1, 4, \infty)$	0.1353	68.82	0.54	0.54
$[0, 1, 6, \infty)$	0.1323	69.35	0.54	0.54

Table 3.14: Comparison of YASS-M1 iterative embedding scheme with other methods - the interpretation of the schemes in the left-most column is as follows: (i) YASS: $QF_h = 60$ refers to the original YASS hiding using a constant $QF_h = 60$, (ii) *50-60-70-rand* refers to hiding using a mixture of QF values, 50, 60 and 70, when the allocation is done randomly, (iii) *50-60-70-rand-M1* refers to the M1 iterative embedding scheme (1 iteration) being used after hiding using *50-60-70-rand* method, (iv) *50-60-70-var* refers to hiding using a mixture of QF values, 50, 60 and 70, when the allocation is done based on local variance, and (v) *50-60-70-var-M1* refers to the M1 iterative embedding scheme (1 iteration) being used after hiding using *50-60-70-var* method.

Hiding Method	B=9 (bpnc)	B=9 PF-274 (P_d)	B=25 (bpnc)	B=25 PF-274 (P_d)
YASS: $QF_h = 60$	0.1839	0.66	0.2073	0.67
50-60-70-rand	0.1704	0.59	0.1907	0.60
50-60-70-rand-M1	0.2335	0.63	0.2552	0.64
50-60-70-var	0.1930	0.59	0.2191	0.60
50-60-70-var-M1	0.2375	0.64	0.2651	0.65

Merging the results of Tables 3.14 and 3.15, we demonstrate the superiority of the various randomized hiding based methods in Table 3.16 - a higher bpnc can be obtained than that of F5, while the detection accuracy is also reduced.

Table 3.15: Variation of the average detection accuracy with the hiding rate (in bpnc) for F5

bpnc	0.20	0.18	0.15	0.10	0.08	0.06	0.04
PF-274	0.89	0.86	0.83	0.75	0.67	0.63	0.56
Shi-324	0.64	0.60	0.56	0.53	0.52	0.51	0.50

Table 3.16: Variation of the average detection accuracy with the hiding rate (in bpnc) for F5

Hiding Method	bpnc	PF-274 (P_d)	Shi-324 (P_d)
F5 (bpnc=0.08)	0.0800	0.67	0.52
F5 (bpnc=0.15)	0.1500	0.83	0.56
B=9, YASS: QF_h=60	0.1844	0.66	0.61
B=9, 50-60-70-rand	0.1704	0.59	0.58
B=9, 50-60-70-rand-M1	0.2335	0.63	0.61
B=9, 50-60-70-var	0.1930	0.59	0.58
B=9, 50-60-70-var-M1	0.2375	0.64	0.61

Chapter 4

Matrix Embedding for Robust and Secure Steganography

Matrix embedding has been used for passive steganography, i.e. for noise-free channels. Here, we devise a hiding framework which combines the security provided by matrix embedding with the robustness provided by suitable error correction codes to have a robust and secure active steganographic method. Such a combination involves several challenges which are discussed in this chapter.

The embedding distortion and vulnerability to steganalysis are significantly less for matrix embedding than that of conventional quantization index modulation (QIM) based hiding. However, ME is less robust to attacks, with a single host bit error leading to multiple decoding errors for the hidden bits. Here, we

present the ME-RA scheme, a combination of ME-based hiding with powerful repeat accumulate (RA) codes for error correction, to address this problem. Before explaining the intricacies of using ME along with RA coding for active steganography, we motivate where we need ME-RA as opposed to QIM-RA (using QIM to embed the RA-encoded bits) for steganography. In the previous chapter, QIM-RA was used for YASS and we obtained substantially low detection rates. However, there are hiding conditions where the detection rate is high for QIM-based YASS and for some applications, the emphasis can be on very low detection rates even at the cost of low hiding rates. In this chapter, we show that when a low hiding rate is good enough and the channel attacks are of moderate strength, ME potentially outperforms QIM by allowing a higher data-rate at the same level of detectability (or being more undetectable at similar data-rates).

Why is using ME along-with iterative decoding schemes such as RA codes a non-trivial problem? For proper decoding, the embedding locations have to be initialized with suitable confidence values, termed as log likelihood ratios. For the QIM scheme, each embedding coefficient corresponds to a single code-bit while for ME, a series of embeddable coefficients corresponds to a code-bit. This many-to-one mapping between the host coefficients and an encoded bit makes the computation of the log likelihood ratios (LLRs) for RA decoding a challenging proposition for ME-RA. We propose various methods to compute the LLRs for the ME-RA

scheme. To demonstrate the effectiveness of our proposed method, we present results using YASS, our randomized block-based hiding framework, replacing the QIM-based embedding in YASS by the proposed ME-RA scheme. Similar to the QIM-RA scheme, we also show how the redundancy factor used by the RA-code based error correction scheme can be optimally set at the encoder and then independently and accurately estimated at the decoder for ME-RA. We also show that the embedding performance can be improved by employing punctured RA codes. Through experiments based on a couple of thousand images, we show that for the same embedded data-rate and a moderate attack level, the proposed ME-based method results in a lower detection rate than that obtained for QIM-based YASS.

The outline of this chapter is as follows. An introduction to matrix embedding and its role in the steganography problem is provided in Section 4.1. The overall ME-based hiding system is described in Section 4.2. The embedding method for the **ME-RA** scheme (using ME to embed the RA-encoded bits) is explained in Section 4.3. The various approaches used for the initial LLR computation at the decoder are explained in Section 4.4. The LLR computation assuming errors, apart from erasures, for a given channel is discussed in Section 4.5. The **ME-RA-puncture** scheme, which is a refinement of the ME-RA method and produces a higher effective embedded data-rate than ME-RA through "puncturing" (*deletion of a suitable number of encoded bits at appropriate bit locations*),

is explained in Section 4.6. Section 4.7 compares the data-rate obtained using various methods for the initial LLR computation. It also compares the performance of **ME-RA-puncture** to the **non-shrinkage** method. Comparison of the detectability against similar steganalysis methods and effective hiding rate under different noise channels, for **QIM-RA** (using QIM to embed the RA-encoded bits) and **ME-RA-puncture** methods, are presented in Section 4.8. Methods to determine the redundancy factor used for RA-coding for the ME-RA scheme at the decoder, without any side information, are discussed in Section 4.9. We conclude in Section 4.10 where we describe the contributions of matrix embedding to secure and robust steganography, and also outline directions for future work.

4.1 Matrix Embedding based Active Steganography

Active Steganography: The requirements from a "good" steganographic scheme are a high embedding capacity, while remaining statistically secure. When there is an active adversary in the transmission channel, the hiding scheme is considered to be an example of "active steganography". Various examples of the active warden scenario (active steganography) are in [10, 21, 31, 54, 81]. In our problem formulation, we assume that the "active warden" may modify the trans-

mitted signal which may (or may not) contain the embedded message - the signal being the stego (or cover) image in our case. Unlike passive warden steganography, where the stego signal is not further modified, active steganography has the additional requirement that the hidden data must be recoverable even after benign or malicious processing of the stego signal.

Matrix embedding (ME) [20], an embedding technique with a high embedding efficiency (*defined as the number of data bits embedded per unit embedding distortion*), is generally used for passive steganography. The lower embedding distortions (as compared to embedding methods like quantization index modulation (QIM) [16], while both methods have the same embedded data-rate) make the hiding less detectable. Here, we use ME for active steganography by combining it with powerful error correction codes. We first explain a few terms in relation with coding theory to refresh the reader's mind.

Some Coding Theory Fundamentals: Let us consider a (n, k) code where k data bits are transmitted using a codeword of length n bits. For a n-tuple codeword $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the parity check matrix $\mathbf{H}$ is a $k \times n$ matrix such that $\mathbf{H}\mathbf{x}^T = \mathbf{0}$ (a k-tuple all-zero vector). Given a parity-check matrix $\mathbf{H}$, the corresponding $n \times k$ generator matrix $\mathbf{G}$ satisfies $\mathbf{G}\mathbf{H}^T = \mathbf{0}$ (a n-tuple all-zero vector) and $\mathbf{H}\mathbf{G}^T = \mathbf{0}$ (a k-tuple all-zero vector). Given a k-tuple dataword $\mathbf{u} = (u_1, u_2, \ldots, u_k)$, the codeword $\mathbf{x}$ is given by $\mathbf{x} = \mathbf{u}\mathbf{G}$. Given a received

codeword $\mathbf{x}$ and a parity-check matrix $\mathbf{H}$, the syndrome S is a k-tuple which is expressed as: $S = \mathbf{H}\mathbf{y}^T$. Given a set $\mathbf{X}$ of all linear codes $\mathbf{x}$, for any vector $\mathbf{a}$, the set $(\mathbf{a} + \mathbf{X}) = \{\mathbf{a} + \mathbf{x} \mid \mathbf{x} \in \mathbf{X}\}$ is called a coset of $\mathbf{x}$. The minimum weight vector in this coset is called the coset leader. Thus, given a n-tuple codeword $\mathbf{x}$, and a specific linear code (given by $\mathbf{G}$), the coset and coset leader can be obtained. With this brief introduction of coding theory, we focus on ME and explain it using an example.

Matrix Embedding Example: Consider (7,3) matrix embedding, in which 3 data bits are embedded in 7 host bits. The idea is to perturb the host bits minimally so that they fall in the coset of a linear code, whose syndrome equals the data bits to be hidden.

In particular, we consider the (7,4) Hamming code with parity check matrix $\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$. For a host sequence $\mathbf{a} = (1, 0, 0, 1, 0, 0, 1)$, the syndrome $\mathbf{b}'$ is obtained as: $(\mathbf{b}')^T = \mathbf{H}(\mathbf{a})^T = (0, 1, 0)^T$, where the operations are performed over the binary field. If the data bits to be embedded are $(0, 1, 0)$, we are done; we can send the host bits without perturbation. However, suppose that we wish to embed $(0, 0, 0)$. The aim is to find $\mathbf{\Delta a}$, the perturbation vector for $\mathbf{a}$, with the lowest Hamming weight. Then, $\mathbf{H}(\mathbf{a})^T + \mathbf{H}(\mathbf{\Delta a})^T = (0, 0, 0)^T$.

Therefore, $\mathbf{H}(\mathbf{\Delta a})^T = (0,1,0)^T$. If only the i^{th} element in $\mathbf{\Delta a}$ is 1, then $\mathbf{H}(\mathbf{\Delta a})^T$ equals the i^{th} column in $\mathbf{H}$. The 2^{nd} column in $\mathbf{H} = (0,1,0)^T$. Therefore, $\mathbf{\Delta a} = (0,1,0,0,0,0,0)$. Similarly, to embed the data bits $(1,1,1)$, the perturbation $\mathbf{\Delta a}$ is such that $\mathbf{H}(\mathbf{a})^T + \mathbf{H}(\mathbf{\Delta a})^T = (1,1,1)^T \Rightarrow \mathbf{H}(\mathbf{\Delta a})^T = (1,0,1)^T$. Since the 5^{th} column of $\mathbf{H} = (1,0,1)^T$, $\mathbf{\Delta a} = (0,0,0,0,1,0,0)$. Similarly, for any three-tuple we might wish to embed, we need to change at most one host bit (Hamming weight of $\mathbf{\Delta a} \leq 1$), which illustrates why matrix embedding is so powerful for passive warden steganography.

Comparison with Quantization Index Modulation: Given the popularity of QIM embedding [16,107], it is the natural benchmark against which to compare new steganographic methods. Scalar QIM in the context of JPEG steganographic schemes corresponds to rounding a discrete cosine transform (DCT) coefficient to the nearest even or odd quantization level, depending on the embedded bit. Thus, the DCT coefficient is changed with probability half, relative to rounding to the nearest quantizer level as done in any case during JPEG compression. Therefore, on average, the embedding efficiency is 2 bits per changed coefficient. On the other hand, for (7,3) ME, we embed 3 bits by modifying one of 7 coefficients resulting in a higher embedding efficiency of 3. Given a set of 7 embedding coefficients, QIM and ME schemes can embed 7 and 3 bits, respectively, so that QIM achieves a higher hiding rate for the same number of hiding locations. Also,

a single error affects the decoding of only a single bit for QIM. For (7,3) ME, changing a single coefficient (for example, flipping the 7^{th} bit of $\mathbf{a}$ in the above example) can lead to three bit errors, so that ME is more fragile than QIM for active steganography. Thus the regime in which ME can potentially outperform QIM is when the required hiding rate is low enough so that the lower hiding rate of ME is not an issue. Also, the channel attacks should be of moderate strength so that the corresponding errors can be rectified through powerful error correction codes. This is the regime explored in our work.

Contributions: Our main contributions are outlined below.

(a) We impart noise robustness to the ME-based active steganographic framework by performing error correction coding (ECC) on the data bits to generate the code bits embedded using ME. For maintaining perceptual transparency after hiding, hiding in coefficients of small magnitudes is avoided. This is interpreted as erasures at the encoder, as in our work on QIM based hiding. For ECC, we have used repeat accumulate (RA) codes [25], which are well suited for such high erasure channels [107]. We also show that *punctured* [5, 6] RA codes can yield superior performance over the original RA codes.

(b) For iterative decoding, the decoder needs to be initialized with proper confidence values (log likelihood ratios or LLRs) at the embedding locations. Even if one out of 7 host coefficients is erased for (7,3) ME, it can affect the decoding

at 3 bit locations - the RA decoder needs to consider the various erasure-related cases before computing the soft weights to initialize the iterative decoder. *Thus, a key contribution is to work through the combinatorics required to determine the soft decisions.*

(c) To reduce detectability, we use Yet Another Steganographic Scheme (YASS). Here, the hiding blocks are pseudo-randomly chosen to desynchronize the self-calibration scheme, which assumes that the hiding is done in regular 8×8 blocks of DCT coefficients, used in standard steganalysis methods [82, 83]. ME-based YASS is shown to be less detectable than QIM-based YASS for the same steganalysis schemes for the same (low) data rate.

To summarize the roles of the various modules, YASS suggests "where" to embed (randomized block locations), ME shows "how" to embed (embedding method) and the RA-based ECC framework determines "what" gets embedded (it generates code bits to be embedded, given the information bits) - this is illustrated in Fig. 4.1.

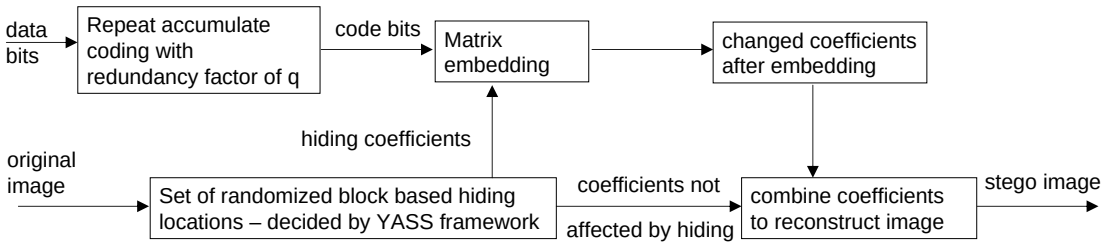


Figure 4.1: The entire hiding framework using RA coding, matrix embedding and YASS-based coefficient selection

Related work: Since the concept of matrix embedding was first introduced by Crandall [20], there have been a number of papers describing properties and improvements of ME, *for passive steganography* [35, 41, 55, 123]. The F5 algorithm [123] is a secure JPEG steganographic scheme based on ME. F5 uses $(1, n, k)$ coding based on binary Hamming codes, where $n = 2^k - 1$, and k message bits are to be embedded in n image DCT coefficients, by changing the least significant bit (LSB) of (at most) one of the n quantized DCT coefficients after rounding. This method was further generalized by Kim et al. [55] - (t, n, k) coding methodology is suggested where while embedding k-bits into n host bits ($n = 2^k - 1$), at most t ($t \geq 1$) embedding changes can be made. It is useful when the single coefficient to be perturbed in $(1, n, k)$ scheme cannot be modified due to various constraints (for example, when it is in the erasure zone). Approaches to increase the embedding rate for ME based on random linear codes of small dimension, and simplex codes, have been presented by Fridrich et al. [41]. In the context of minimal distortion steganography, an interesting approach is Perturbed Quantization (PQ) [38], which is highly secure from modern detectors [40]. Another contribution of the PQ scheme is the use of wet paper codes (WPC) which allows sending the same number of bits, on an average, as would have been possible if the receiver knew the actual set of hiding locations. In [35], the embedding impact profile of all the pixels is estimated and an optimal trade-off (to make the hiding less detectable

for similar embedding rates) is studied between the number of embedding changes and their amplitudes, while using matrix embedding.

While the ME-based methods generally focus on passive steganography, our QIM-based hiding scheme YASS [108] achieved secure (using randomized block locations) and robust (using RA-based encoding) steganography. Due to the lower embedding efficiency, QIM schemes are more detectable than ME-based schemes like F5 [123], non-shrinkage F5 (nsF5) [40] and Modified Matrix Embedding (MMx) [55]; *but all these ME-based methods are passive stego schemes.* Here, we combine the low detectability of ME with the robustness resulting from powerful ECC to obtain an active stego scheme that works for real-world channels such as JPEG compression attacks. We focus on data sets for which QIM-based YASS is vulnerable to steganalysis even at low embedding rates, and attempt to determine when our ME-based scheme will perform better (as noted earlier, we can only hope to do better than QIM at low enough embedding rates and moderate enough attacks).

Prior work that combines error correction coding with ME-based hiding includes [127]. However, the combination codes in [127] are employed over individual blocks of image coefficients, and are therefore much less powerful than the RA code framework employed here, where the codeword spans the entire set of embeddable image coefficients. Moreover, erasures resulting from coefficient-adaptive

hiding, as well as the effect of real-world channels such as JPEG compression, are not considered in [127].

Table 4.1 introduces some notations which will be frequently used in this chapter. A sequence of 7 terms $\{y_1, y_2, \ldots, y_7\}$ is referred to as $\mathbf{y}$. The complement of a bit a_i is referred to as $\overline{a_i}$.

4.2 Overall Hiding Setup

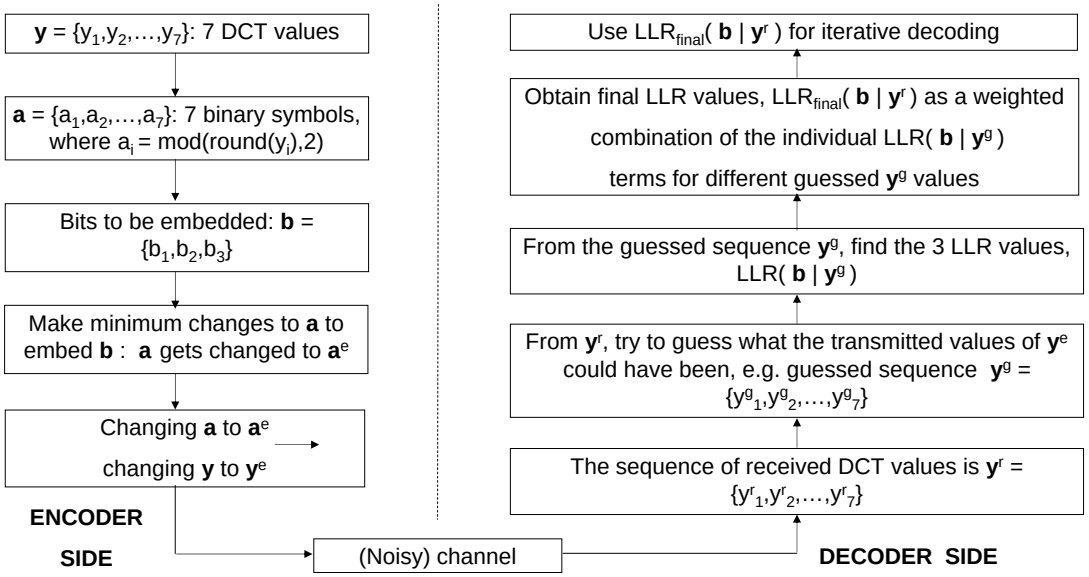


Figure 4.2: There is a 7-element sequence of DCT coefficients $\mathbf{y}$ where a 3-tuple bit sequence $\mathbf{b}$ is embedded. The encoder embeds $\mathbf{b}$ in $\mathbf{y}$ by changing the minimum number of coefficients in $\mathbf{y}$, thus modifying it to $\mathbf{y}^e$ which is transmitted through the channel and $\mathbf{y}^r$ is the corresponding received sequence. The decoder uses $\mathbf{y}^r$ to estimate the LLR values for the 3 bit locations. The sequence $\mathbf{a}^r$, which is derived from $\mathbf{y}^r$ ($a_i^r = \text{mod}(\text{round}(y_i^r), 2),\ \forall i$), is not explicitly shown at the decoder side.

Table 4.1: Glossary of Notations

Notation	Definition
$\mathbf{y} = \{y_1, y_2, \ldots, y_7\}$	set of 7 AC DCT coefficients that are considered together to embed 3 code bits for (7,3) ME
$\mathbf{a} = \{a_1, a_2, \ldots, a_7\}$	set of binary symbols obtained from $\mathbf{y}$ where $a_i = \text{mod}(\text{round}(y_i), 2),\ 1 \leq i \leq 7$
$\mathbf{b} = \{b_1, b_2, b_3\}$	set of 3 bits that are embedded in $\mathbf{y} = \{y_1, y_2, \ldots, y_7\}$
$\mathbf{b}' = \{b'_1, b'_2, b'_3\}$	syndrome obtained from LSBs of $\mathbf{y}$, i.e. from $\mathbf{a}$, thus, $(\mathbf{b}')^T = \mathbf{H}(\mathbf{a})^T$ where $\mathbf{H}$ is introduced in Section 4.1.
$\mathbf{y}^e = \{y_1^e, y_2^e, \ldots, y_7^e\}$	set of 7 AC DCT coefficients to which $\mathbf{y}$ is changed after embedding
$\mathbf{a}^e = \{a_1^e, a_2^e, \ldots, a_7^e\}$	set of binary symbols to which $\mathbf{a}$ is changed after embedding, $a_i^e = \text{mod}(\text{round}(y_i^e), 2),\ 1 \leq i \leq 7$
$\mathbf{y}^r = \{y_1^r, y_2^r, \ldots, y_7^r\}$	set of received DCT elements at the decoder side corresponding to $\mathbf{y}^e$
$LLR_{final}(\mathbf{b}\|\mathbf{y}^r)$	the set of log-likelihood ratio (LLR) values for the bit locations $\mathbf{b} = \{b_1, b_2, b_3\}$ based on $\mathbf{y}^r$
QF_h	the design quality factor selected for hiding, which is used to generate the quantized AC DCT coefficients
QF_a	the output quality factor at which the stego image, after randomized block based hiding, is JPEG compressed
B, N_B	B is the big-block size used for YASS ($B > 8$) while N_B is the number of $B \times B$ blocks in the image
λ	the number of top AC DCT coefficients, encountered during zigzag scan, used for hiding per 8×8 block
$T \in \mathbb{R}^{3\times3}$	the transition probability matrix between $\mathbf{y}^e$ and $\mathbf{y}^r$, where each term is mapped to one of $\{0, 1, e\}$, and e denotes an erasure
bpnc	ratio of the number of data bits successfully embedded in an image to the number of non-zero (after rounding) block-based AC DCT image coefficients
QIM-RA: n terms	the QIM-RA (QIM-based YASS where RA-coding is used as ECC) scheme where the first n AC DCT elements encountered during zigzag scan per 8×8 block are used for embedding, i.e. $\lambda = n$
q_{opt}	the minimum RA code redundancy factor which allows proper decoding and recovery of all the data bits

Overall System Flow: The overall system flow can be followed using Fig. 4.2. Let N_B denote the number of $B \times B$ blocks obtained from the image, with λ elements used for hiding from the DCT matrix for the 8×8 pixel block pseudo-randomly chosen from every $B \times B$ block. For $9 \leq B \leq 15$, the total number of DCT coefficients available for hiding equals $N = \lambda N_B$. When $B_{eff} < 9$ (for example, when $B = 17$, 25 or 49), $N = N_B$(no. of 8×8 blocks in a $B \times B$ big-block)$\lambda = N_B(\lfloor \frac{B}{8} \rfloor)^2 \lambda$. Using (7,3) ME, the coefficients are partitioned into $\lfloor \frac{N}{7} \rfloor$ sets, each having 7 terms - for example, $\mathbf{y} = \{y_1, y_2, \ldots, y_7\}$ is such a set where 3 bits (b_1, b_2, b_3) are embedded. The total number of code bits equals $3\lfloor \frac{N}{7} \rfloor$, and for a RA code of redundancy factor q, $\lfloor (3\lfloor \frac{N}{7} \rfloor)/q \rfloor$ data bits can be accommodated. *At the encoder side, the problem is to embed the code bits using ME while minimally changing the DCT coefficients* (for example, from $\mathbf{y}$ to $\mathbf{y}^e$). The redundancy factor q is set to q_{opt} (defined in Table 4.1) such that *the hiding rate is maximized for a given set of hiding parameters and a specified attack channel.* There is no need for a side channel for conveying the value of q to the decoder; the latter can recover q from the RA-encoded sequence as explained in [2,97]. We do not consider attacks that can potentially desynchronize the encoder and decoder (for example, warping or cropping), so that the decoder has access to the same set of N hiding locations as the encoder. A proper initialization of the soft weights (log likelihood ratios or LLRs, explained in Section 4.4) for codeword locations allows faster convergence

Table 4.2: Explanation of how $\{b_1, b_2, b_3\}$ is embedded by minimally changing $\{a_1, a_2, \ldots, a_7\}$ at the encoder side: here, we make minimum changes to $\mathbf{a} = \{a_1, \ldots, a_7\}$ to obtain $\mathbf{a}^e = \{a_1^e, \ldots, a_7^e\}$ such that $\mathbf{b}^T = \mathbf{H}(\mathbf{a}^e)^T$, where $\mathbf{b} = (b_1, b_2, b_3)$.

Syndrome obtained from LSBs of (rounded) actual image coefficients	Syndrome obtained from LSBs of (rounded) modified image coefficients
$b'_1 = a_1 \oplus a_3 \oplus a_5 \oplus a_7$	$b_1 = a_1^e \oplus a_3^e \oplus a_5^e \oplus a_7^e$
$b'_2 = a_2 \oplus a_3 \oplus a_6 \oplus a_7$	$b_2 = a_2^e \oplus a_3^e \oplus a_6^e \oplus a_7^e$
$b'_3 = a_4 \oplus a_5 \oplus a_6 \oplus a_7$	$b_3 = a_4^e \oplus a_5^e \oplus a_6^e \oplus a_7^e$

(at a lower redundancy factor) and increases the hiding rate. Various methods to compute the soft weights ($LLR_{final}(\mathbf{b}|\mathbf{y}^r)$) for 3 bit locations ($\mathbf{b} = \{b_1, b_2, b_3\}$) given 7 DCT terms ($\mathbf{y}^r = \{y_1^r, y_2^r, \ldots, y_7^r\}$, which is the noisy version of $\mathbf{y}^e$) are studied in Section 4.4.

4.3 ME-RA Embedding

The encoder embedding logic for (7,3) ME, introduced in Section 4.1, is explained in detail here. The operations at the encoder side are outlined in Fig. 4.2. The sequence $\mathbf{b} = \{b_1, b_2, b_3\}$ is to be embedded in $\mathbf{y}$, a set of 7 locations, as shown in Table 4.2. After rounding to the nearest integer and modulo 2 operation, the coefficients return the bit sequence $\mathbf{a} = \{a_1, a_2, \ldots, a_7\}$, where $a_i = \text{mod}(\text{round}(y_i), 2)$. Using the (3x7) mapping matrix $\mathbf{H}$ (introduced in Section 4.1) on $\mathbf{a}$, the following syndrome $\mathbf{b}'$ is obtained: $(b'_1, b'_2, b'_3)^T = \mathbf{H}(\mathbf{a})^T$ (Table 4.2).

Table 4.3: For (7,3) ME, there are 8 possible relations between **b**, **the 3-tuple of bits to be embedded**, and **b′**, **the syndrome obtained from y**. There is always a single coefficient y_i, whose modification to y_i^e (ensuring $a_i^e = \overline{a_i}$ where $a_i = \text{mod}(\text{round}(y_i), 2)$ and $a_i^e = \text{mod}(\text{round}(y_i^e), 2)$) ensures proper embedding. If y_i is in the erasure zone, the candidate 2-tuples/3-tuples for embedding are mentioned.

Condition	Relation between $\mathbf{b}'$ and $\mathbf{b}$	1-tuple	Possible 2-tuples
0	$b'_1 = b_1,\ b'_2 = b_2,\ b'_3 = b_3$	none	none
1	$b'_1 = \overline{b_1},\ b'_2 = b_2,\ b'_3 = b_3$	y_1	$(y_2, y_3)(y_4, y_5)(y_6, y_7)$
2	$b'_1 = b_1,\ b'_2 = \overline{b_2},\ b'_3 = b_3$	y_2	$(y_1, y_3)(y_4, y_6)(y_5, y_7)$
3	$b'_1 = \overline{b_1},\ b'_2 = \overline{b_2},\ b'_3 = b_3$	y_3	$(y_1, y_2)(y_4, y_7)(y_5, y_6)$
4	$b'_1 = b_1,\ b'_2 = b_2,\ b'_3 = \overline{b_3}$	y_4	$(y_1, y_5)(y_2, y_6)(y_3, y_7)$
5	$b'_1 = \overline{b_1},\ b'_2 = b_2,\ b'_3 = \overline{b_3}$	y_5	$(y_1, y_4)(y_2, y_7)(y_3, y_6)$
6	$b'_1 = b_1,\ b'_2 = \overline{b_2},\ b'_3 = \overline{b_3}$	y_6	$(y_1, y_7)(y_2, y_4)(y_3, y_5)$
7	$b'_1 = \overline{b_1},\ b'_2 = \overline{b_2},\ b'_3 = \overline{b_3}$	y_7	$(y_1, y_6)(y_2, y_5)(y_3, y_4)$

Table 4.4: For (7,3) ME, there are 8 possible relations between **b**, **the 3-tuple of bits to be embedded**, and **b′**, **the syndrome obtained from y**. There is always a single coefficient y_i, whose modification to y_i^e (ensuring $a_i^e = \overline{a_i}$ where $a_i = \text{mod}(\text{round}(y_i), 2)$ and $a_i^e = \text{mod}(\text{round}(y_i^e), 2)$) ensures proper embedding. If y_i is in the erasure zone, the candidate 2-tuples/3-tuples for embedding are mentioned.

Condition	Relation between $\mathbf{b}'$ and $\mathbf{b}$	Possible 3-tuples
0	$b'_1 = b_1,\ b'_2 = b_2,\ b'_3 = b_3$	none
1	$b'_1 = \overline{b_1},\ b'_2 = b_2,\ b'_3 = b_3$	$(y_2, y_4, y_7)(y_2, y_5, y_6)(y_3, y_5, y_7)(y_3, y_6, y_4)$
2	$b'_1 = b_1,\ b'_2 = \overline{b_2},\ b'_3 = b_3$	$(y_1, y_4, y_7)(y_1, y_5, y_6)(y_3, y_4, y_5)(y_3, y_6, y_7)$
3	$b'_1 = \overline{b_1},\ b'_2 = \overline{b_2},\ b'_3 = b_3$	$(y_1, y_5, y_7)(y_1, y_4, y_6)(y_2, y_4, y_5)(y_2, y_6, y_7)$
4	$b'_1 = b_1,\ b'_2 = b_2,\ b'_3 = \overline{b_3}$	$(y_1, y_2, y_7)(y_1, y_3, y_6)(y_2, y_3, y_5)(y_5, y_6, y_7)$
5	$b'_1 = \overline{b_1},\ b'_2 = b_2,\ b'_3 = \overline{b_3}$	$(y_1, y_3, y_7)(y_1, y_2, y_6)(y_2, y_3, y_4)(y_4, y_6, y_7)$
6	$b'_1 = b_1,\ b'_2 = \overline{b_2},\ b'_3 = \overline{b_3}$	$(y_1, y_2, y_5)(y_1, y_3, y_4)(y_2, y_3, y_7)(y_4, y_5, y_7)$
7	$b'_1 = \overline{b_1},\ b'_2 = \overline{b_2},\ b'_3 = \overline{b_3}$	$(y_1, y_2, y_4)(y_1, y_3, y_5)(y_2, y_3, y_6)(y_4, y_5, y_6)$

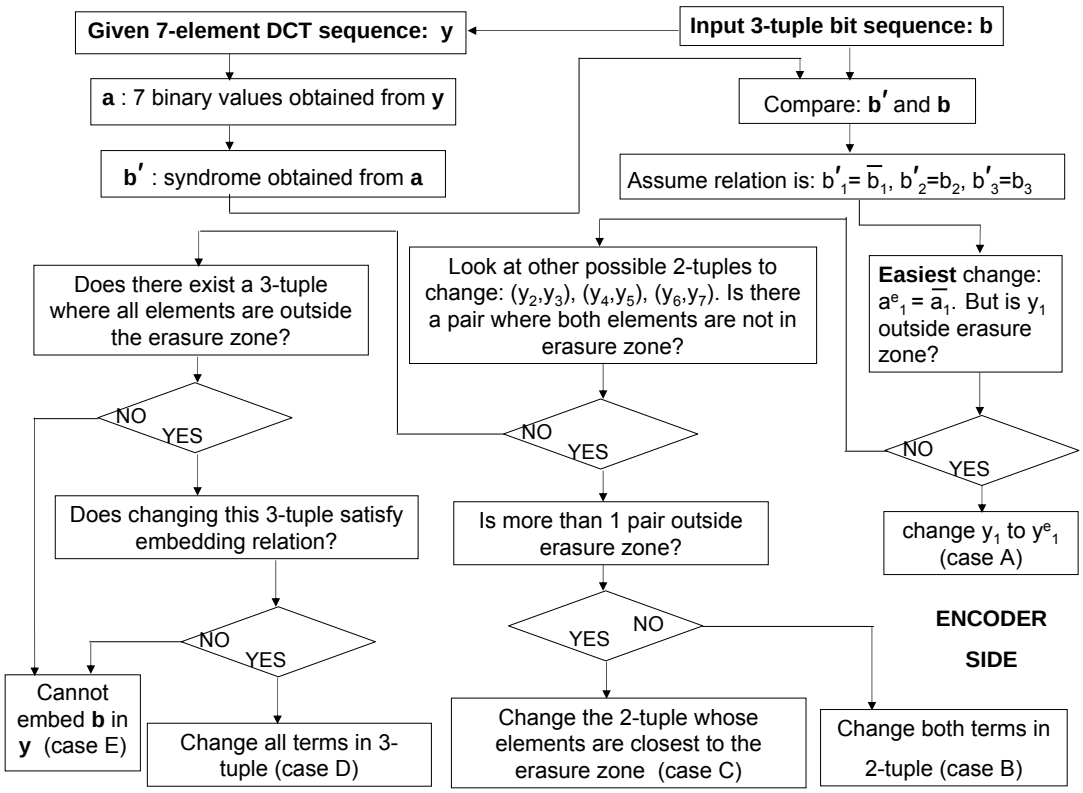


Figure 4.3: Embedding logic used at the encoder for a (7,3) matrix embedding example : cases (A)-(D) correspond to cases where embedding is possible, while case (E) is where embedding fails.

From Tables 4.3 and 4.4, for every condition (except condition 0), if y_i is the only element being changed (for the other elements, y_j^e is the rounded version of y_j, $j \neq i$), $a_i^e = \overline{a_i}$ and $a_j^e = a_j, j \neq i$. To ensure $a_i^e = \overline{a_i}$, y_i is changed to the nearest odd/even integer y_i^e, if y_i gets converted to an even/odd integer, respectively, after rounding off. It may happen that the absolute value of that single coefficient to be modified ($|y_i|$) is less than 0.5 - an erasure occurs at that location. Since we have a series of candidate locations for embedding, this erasure-induced problem can be

solved by changing two or more of the remaining coefficients in $\mathbf{y}$ (using solution proposed in [55]). Based on the embeddable bits and the candidate coefficients, 5 cases (**A-E**) are possible, as shown in Fig. 4.3.

Algorithm to decide on the minimum perturbations for embedding:

• We use Tables 4.3 and 4.4 to find the single coefficient y_i to change to the nearest odd or even integer y_i^e, if y_i rounds off to an even or odd integer, respectively. If that element (for example, y_1 in Fig. 4.3) can indeed be modified, it corresponds to **Case A**.

• If that single coefficient (y_i) lies in the erasure zone $(-0.5, 0.5)$, we look for the pairs of 2-tuples that can be perturbed for proper embedding. If there is just one suitable 2-tuple, it corresponds to **Case B**. If there are more than one suitable 2-tuple, we select that tuple whose perturbation introduces the least embedding distortion - this is **Case C**. For example, among 2 pairs (0.6, 0.7) and (0.90, 0.95), (the elements in both pairs are changed to (0, 0) after embedding), the total perturbation equals $0.6 + 0.7 = 1.3$ and $0.90 + 0.95 = 1.85$, respectively. Hence, the first pair is used for embedding - in general, among two pairs with elements in (0.5, 1), the one with elements closer to 0.5 should be selected for embedding.

• The process is repeated for 3-tuples, if we do not get a suitable 2-tuple for embedding. **Case D** corresponds to the use of a 3-tuple for embedding. If a

suitable 1/2/3-tuple cannot be found for embedding, then a higher-order tuple cannot also be found for embedding using (7,3) ME - this is referred to as **Case E**. Thus, data embedding is possible for **Cases A-D** when there are suitable 1/2/3 tuples; for **Case E**, the relevant 3 bits cannot be embedded due to too many erasures.

4.4 ME-RA Decoding

We now discuss the decoder operations depicted in Fig. 4.2. One of the main challenges involved in using ME along with RA coding is the computation of the initial LLR values provided to the decoder. It is instructive to review the LLR computations used in prior work on QIM-based embedding [107, 108] before discussing the more complicated computations for ME-based embedding.

Definition of LLR: Let a certain image coefficient be equal to y and the corresponding embedded bit be b. The LLR value $LLR(y)$ denotes the logarithm of the ratio of the likelihood that a 0 was transmitted through that coefficient ($P[b = 0|y]$) to the likelihood that a 1 was transmitted ($P[b = 1|y]$).

$$LLR(y) = \log \left\{ \frac{P[b = 0|y]}{P[b = 1|y]} \right\} \tag{4.1}$$

Hence, $LLR(y)$ is positive when 0 is the more likely bit and vice versa. $LLR(y)$ equals zero for an erasure. In our QIM-based methods [107, 108], an image coeffi-

cient was changed to the nearest even or odd integer to embed 0 or 1, respectively. If the absolute value of y were less than 0.5, $LLR(y) = 0$ (erasure); else, depending on whether the received coefficient were close to an even or odd integer, $LLR(y)$ is set to α or $-\alpha$, where α reflects the decoder's confidence in the accuracy of the decoded bit values. We have performed analysis to understand how the scaling factor for the LLR [2] values can be fixed based on the design hiding parameters.

LLR Computation for ME: For (7,3) ME, the 7-element sequence of received DCT coefficients, $\mathbf{y}^r$, decides the log likelihood for $\mathbf{b}$, a 3-tuple bit sequence. We denote it as $LLR(\mathbf{b}|\mathbf{y}^r)$ in (4.2) - it is a sequence of 3 terms, each denoting the LLR for an individual bit location. From (4.1) and (4.2), it is evident that while an LLR value ($LLR(b|y)$) for QIM-RA depends on only one coefficient value (y), there is a 7:1 mapping between the elements in $\mathbf{y}^r$ and an LLR value ($LLR(b_j|\mathbf{y}^r)$, $1 \leq j \leq 3$) for ME-RA.

$$LLR(\mathbf{b}|\mathbf{y}^r) = \left[\log\left\{\frac{P[b_1=0|\mathbf{y}^r]}{P[b_1=1|\mathbf{y}^r]}\right\}, \log\left\{\frac{P[b_2=0|\mathbf{y}^r]}{P[b_2=1|\mathbf{y}^r]}\right\}, \log\left\{\frac{P[b_3=0|\mathbf{y}^r]}{P[b_3=1|\mathbf{y}^r]}\right\}\right] \tag{4.2}$$

If we assume an ideal channel between the transmitted sequence $\mathbf{y}^e$ and the received sequence $\mathbf{y}^r$ (i.e. $\mathbf{y}^e = \mathbf{y}^r$), the 3-tuple of decoded bits, $\mathbf{b}^r$, obtained as the syndrome of $\mathbf{a}^r$, can be directly used for LLR estimation - here, $(\mathbf{b}^r)^T = \mathbf{H}(\mathbf{a}^r)^T$, where $a_i^r = \text{mod}(\text{round}(y_i^r), 2)$, $\forall i$. *In practice, the LLR values should be modified*

to account for those conditions where embedding of $\mathbf{b}$ *in* $\mathbf{y}$ *was not possible at the encoder because of erasures, or to account for errors, when* $a_i^e \neq a_i^r$.

Motivation Behind Proposed LLR Computation Methods: Since we do not have an exact mathematical model for the statistics of the data hiding channel (which varies across images), we investigate three methods (Methods **M1, M2** and **M3**) for LLR computation and empirically evaluate their performance in terms of the hiding rate (Table 4.12 in Section 4.7). Our performance metric in comparing the LLR computation methods is the overall hiding rate; thus a better LLR computation method would require a smaller redundancy factor. Both M1 and M2 compute the LLR ($LLR(b_j|\mathbf{y}^r)$) as a summation of the individual LLRs, for each of the 8 possible mappings between $\mathbf{b}^r$ (syndrome obtained at the decoder) and $\mathbf{b}'$ (syndrome at the encoder side, unknown to the decoder). For each individual LLR computation, M1 assigns an LLR value of $\{\alpha, -\alpha, 0\}$, depending on whether the bit more likely to be embedded is $\{0, 1, e\}$ (e = erasure), respectively - this is similar to the LLR allocation for QIM-RA [108]. M2 uses a more detailed analysis of the various erasure scenarios for LLR allocation. M3 computes the LLR as a ratio of probabilities, as per the definition (4.2), while considering all the 8 combinations, along with the different erasure scenarios (like M2). Computing the LLR as a ratio instead of a summation of individual LLRs eliminates the need for choosing α explicitly. However, M3 also requires some ad

hoc choices, specifically regarding weighting factors that depend on the distribution of the quantized DCT coefficients.

We initially ignore the channel effects (errors and erasures associated with the transition probability matrix between $\mathbf{y}^e$ and $\mathbf{y}^r$) in our LLR computation methods (we assume that $a_i^e = a_i^r, \forall i$) described in Section 4.4.1 and 4.4.2. The LLR computation methods are later modified to incorporate the channel effects in Section 4.5.

4.4.1 LLR Computation- Method M1 and Method M2

The final LLR values $LLR(\mathbf{b}|\mathbf{y}^r)$ are computed by summing over the individual LLRs for all the 8 possible conditions (possible mappings between $\mathbf{b}^r$ and $\mathbf{b}'$ as shown in Tables 4.3 and 4.4).

$$LLR(\mathbf{b}|\mathbf{y}^r) = \sum_{j=0}^{7} P(\text{condition } j) LLR(\mathbf{b}|\mathbf{y}^r, \text{condition } j) \tag{4.3}$$

Here, condition j is the condition where proper embedding can occur by changing only y_j (or suitable 2-3 coefficients if y_j lies in the erasure zone), and for condition 0, no term in $\mathbf{y}$ needs to be modified. The prior probability of condition j is denoted by $P(\text{condition } j)$ (4.3) and equals 1/8 since all the $2^3 = 8$ conditions are equally likely. The problem now becomes one of computing the individual LLR terms ($LLR(\mathbf{b}|\mathbf{y}^r, \text{condition } j)$). If there is a coefficient (or a set of 2-3 co-

efficients) which lies outside the erasure zone and allows proper embedding, the condition is classified as **"Not in Erasure Band" (NEB)**. If all the relevant coefficients lie in the erasure band, the condition is denoted as **"Erasure Band" (EB)**. For NEB, the LLR is computed similarly by M1 and M2 while for EB, the LLR is computed differently.

NEB and EB Conditions

Assuming a certain condition (say, condition i) to be true, if y_i is the coefficient that needs to be modified for proper embedding and $|y_i^r| > 1/2$, then the decoder is sure that proper embedding has occurred by modifying y_i (cases **(A)-(D)** in Fig. 4.3). When y_i^r rounds off to zero, it was either modified from ± 1 (rounded value of y_i) to zero or it was not modified at all, if y_i were in the erasure zone. For example, if $y_i \in (0.5, 1]$ or $y_i \in [-1, -0.5)$ and the embedding logic demands that $a_i^e = \overline{a_i}$, then it is converted to $y_i^e = 0$, after embedding. So, *a received y_i^r coefficient that rounds to zero leaves open the possibility that embedding could not be carried out in that location as happens for case* **E***, assuming that condition i is true* - this is the "shrinkage" problem (this has been countered by non-shrinkage F5 (nsF5) [40], an improvement on the F5 scheme). If $|y_i^r| \leq 0.5$, relevant 2-3 tuples are considered to check if a tuple with all the elements outside the erasure zone can be found - if yes, the condition is NEB; else it is termed as EB.

Example of NEB and EB: For a certain combination of $\mathbf{b}$ and $\mathbf{b}'$, let $b'_1 = \overline{b_1}$, $b'_2 = b_2$ and $b'_3 = b_3$. Thus, proper embedding needs that only y_1 be changed suitably (condition 1 in Table 4.3). If y_1^r equals zero after rounding, we test for suitable 2-tuples out of $\{(y_2^r, y_3^r), (y_4^r, y_5^r), (y_6^r, y_7^r)\}$, where the corresponding y_i terms could have been modified at the encoder so as to achieve the same effect as modifying y_1. Again, we are sure we have a suitable 2-tuple if both the coefficients have an absolute magnitude ≥ 1 (after rounding). If we do not find a suitable 2-tuple, we look for a suitable 3-tuple (Table 4.4). Finally, depending on whether (or not) we find a suitable 1/2/3-tuple, with all its elements outside the erasure zone, the combination is considered to be an example of NEB (or EB).

LLR Computation for NEB: For the NEB condition, if the coefficient which we assumed to have been changed for embedding were indeed the one that was modified, the 3-tuple $\mathbf{b}^r$ computed from $\mathbf{y}^r$ would equal the sequence $\mathbf{b}$ that was supposed to be embedded. The LLR values are then computed identically, for both M1 and M2, based on $\mathbf{b}^r$, using (4.4).

$$LLR(\mathbf{b}|\mathbf{y}^r, \text{condition } j) = \alpha\{(-1)^{b_1^r}, (-1)^{b_2^r}, (-1)^{b_3^r}\}, \tag{4.4}$$

if condition j is NEB, for both M1 and M2

LLR Computation for EB

For the EB condition, the two methods, M1 and M2, compute the individual LLR values $LLR(\mathbf{b}|\mathbf{y}^r, \text{condition } j)$ differently.

• **Method M1:** We assign zeros, corresponding to erasures, to all the 3 bit locations for that condition.

• **Method M2:** If a certain condition is classified as EB, we can still guess what the actual embeddable bits were. For example, consider the EB condition (condition 1 in Tables 4.3 and 4.4) where y_1 or corresponding 2-3 terms could not be modified. Then, $b'_1 = \overline{b_1}$, $b'_2 = b_2$ and $b'_3 = b_3$ (Tables 4.3 and 4.4) is the relation between $\mathbf{b}$ (bits to embed) and $\mathbf{b'}$ (default syndrome) at the embedder side. Since y_1 could not be changed, bit b_1 will be wrongly decoded. Thus, if the EB condition indeed occurred due to y_1, or corresponding 2-3 terms, lying in the erasure zone, then $\{\overline{b_1^r}, b_2^r \text{ and } b_3^r\}$ would be the actual embeddable bits. The LLR for the 3 bit locations for condition 1 can then be expressed in terms of $\overline{b_1^r}$, b_2^r and b_3^r, as shown later in (4.8).

For an EB condition, embedding may still have been possible. Thus, we have *less confidence* in the embeddable bit values suggested by an EB condition than by a NEB condition. M2 uses a weighting factor w_{EB} $(0 < w_{EB} \leq 1)$ for the EB condition LLRs, used below in (4.8). We demonstrate proper embedding under an EB condition using an example. Say, for condition 1, we observe that

$\{y_1^r, y_2^r, y_4^r, y_7^r\}$ are all in the erasure zone, while the remaining terms are outside the erasure zone. Then, if $\{y_1, y_2, y_4, y_7\}$ were all originally in the erasure zone, embedding under condition 1 using 1/2/3-tuples would not be possible. However, it may have been that one/more of these y_i coefficients were in $[-1, -0.5)$ or $(0.5, 1]$ and they got modified to values in the erasure zone after embedding. Thus, proper embedding can occur if $y_1 \in [-1, -0.5)$ or $(0.5, 1]$, or if $y_1 \in [-0.5, 0.5]$ and at least one element from $\{y_2, y_4, y_7\}$ is outside the erasure zone. The probability of proper embedding is expressed as p_{embed} (4.5) and the weighting factor w_{EB} (4.6) corresponds to the probability that embedding could not be performed, due to an EB condition:

$$\begin{aligned} p_{embed} &= (P[y_1 \in \{[-1, -0.5) \cup (0.5, 1]\}]) + \\ &\quad (P[y_1 \in [-0.5, 0.5]])(1 - (P[y_i \in [-0.5, 0.5]])^3) \qquad (4.5) \\ w_{EB} &= 1 - p_{embed} \qquad (4.6) \end{aligned}$$

Though w_{EB} varies per image as it depends on the distribution of the quantized DCT coefficients, we empirically observe that $w_{EB} = 0.5$ is a good choice across a variety of design QFs. For the experiments, we estimate the distribution of the quantized AC DCT coefficients at different quality factors and use that to compute p_{embed} and w_{EB}.

$$LLR(\mathbf{b}|\mathbf{y}^r, \text{condition } j) = \{0,0,0\}, \text{ if condition } j \text{ is EB (M1)} \quad (4.7)$$

$$= \alpha w_{EB}\{(-1)^{f_j(b_1^r)}, (-1)^{f_j(b_2^r)}, (-1)^{f_j(b_3^r)}\}, \quad (4.8)$$

$$\text{if condition } j \text{ is EB (M2)}$$

In (4.8), the functions $f_j(\cdot)$ are given by the j^{th} condition. For an EB condition, the function f_j maps the computed (b_1^r, b_2^r, b_3^r) values to the actual bit sequence which we assumed was embedded. For example, when proper embedding is not possible for condition 1, the mapping between $\mathbf{b}$ (bits to embed) and $\mathbf{b}^r$ (output syndrome) is $b_1^r = \overline{b_1}$, $b_2^r = b_2$ and $b_3^r = b_3$. Hence, $f_1(b_1^r) = \overline{b_1^r}$, $f_1(b_2^r) = b_2^r$ and $f_1(b_3^r) = b_3^r$. The computation of LLR values for M1 and M2 is explained through an example (Table 4.5).

Example 1: In this example, $\mathbf{y}^r = \{0.4, 0.2, 0.9, 1.5, 0.3, 0.1, 1.2\}$, $\mathbf{a}^r = \{0,0,1,0,0,0,1\}$ and $\mathbf{b}^r = \mathbf{H}(\mathbf{a}^r)^T = \{0,0,1\}$. Conditions $\{0,3,4,7\}$ are the NEB conditions. The LLR values for the NEB conditions are computed using (4.4) (for M1 and M2) and the EB condition LLRs are computed using (4.7) and (4.8), for M1 and M2, respectively. To show why condition 1 is classified as EB, consider y_1^r which lies in the erasure zone. Looking for higher order tuples that satisfy condition 1 (from Tables 4.3 and 4.4), we find that $\{y_2^r, y_3^r, y_5^r, y_6^r\}$ all lie in the erasure zone and hence, 2-3 tuples which are suitable for embedding cannot be

found. However, embedding fails under condition 1 only if the corresponding y_i terms were also in the erasure zone. For M2, if condition 1 is true and we assume that proper embedding has not occurred, the actual bit sequence that should have been embedded is $\{\overline{b_1^r}, b_2^r, b_3^r\} = \{1, 0, 1\}$; the resultant LLR equals $\frac{\alpha}{2}\{(-1)^1, (-1)^0, (-1)^1\}$ using (4.8).

Table 4.5: Explanation of LLR allocation for Example-1: conditions $\{0, 3, 4, 7\}$ are the NEB conditions. Here, the syndrome based on $\mathbf{y}^r$, $\{b_1^r, b_2^r, b_3^r\} = \{0, 0, 1\}$. **If condition j is NEB, the actual bit sequence that was embedded is assumed to be $\{b_1^r, b_2^r, b_3^r\}$, while for an EB condition, the bit sequence that should have been embedded is assumed to be** $\{f_j(b_1^r), f_j(b_2^r), f_j(b_3^r)\}$ for M2 (functions f_j are obtained from Tables 4.3 and 4.4). For a certain condition, "Bits" refers to the original sequence that is embedded (for a NEB condition) or the sequence that was supposed to be embedded (for an EB condition). We assume $w_{NEB} = 1/2$.

NEB or EB Condition	Computing Total LLR values			Individual LLR (M1)	Individual LLR (M2)
	Condition	Bits	Sequence		
NEB	0	$\{b_1^r, b_2^r, b_3^r\}$	$\{0, 0, 1\}$	$\alpha\{1, 1, -1\}$	$\alpha\{1, 1, -1\}$
EB	1	$\{\overline{b_1^r}, b_2^r, b_3^r\}$	$\{1, 0, 1\}$	$\{0, 0, 0\}$	$\frac{\alpha}{2}\{-1, 1, -1\}$
EB	2	$\{b_1^r, \overline{b_2^r}, b_3^r\}$	$\{0, 1, 1\}$	$\{0, 0, 0\}$	$\frac{\alpha}{2}\{1, -1, -1\}$
NEB	3	$\{b_1^r, b_2^r, b_3^r\}$	$\{0, 0, 1\}$	$\alpha\{1, 1, -1\}$	$\alpha\{1, 1, -1\}$
NEB	4	$\{b_1^r, b_2^r, b_3^r\}$	$\{0, 0, 1\}$	$\alpha\{1, 1, -1\}$	$\alpha\{1, 1, -1\}$
EB	5	$\{\overline{b_1^r}, b_2^r, \overline{b_3^r}\}$	$\{1, 0, 0\}$	$\{0, 0, 0\}$	$\frac{\alpha}{2}\{-1, 1, 1\}$
EB	6	$\{b_1^r, \overline{b_2^r}, \overline{b_3^r}\}$	$\{0, 1, 0\}$	$\{0, 0, 0\}$	$\frac{\alpha}{2}\{1, -1, 1\}$
NEB	7	$\{b_1^r, b_2^r, b_3^r\}$	$\{0, 0, 1\}$	$\alpha\{1, 1, -1\}$	$\alpha\{1, 1, -1\}$
Final LLR value $LLR(\mathbf{b}\|\mathbf{y}^r)$				$\alpha\{\frac{4}{8}, \frac{4}{8}, -\frac{4}{8}\}$	$\alpha\{\frac{4}{8}, \frac{4}{8}, -\frac{4}{8}\}$

4.4.2 LLR Computation For ME-RA - Method 3 (M3)

For M1 and M2, when bit b is embedded, the LLR value is given by $\alpha(-1)^b$, where α denotes the decoder's confidence level. The proper choice of α varies with

the hiding parameters (QF_h, λ). *Here, we propose an LLR computation method which is independent of* α. By definition (4.2), an LLR term is expressed as a ratio of two probability terms - each term (for example, $P[b_1 = 0|\mathbf{y}^r]$) can be replaced by the relative frequency of occurrence of that event, considering all 8 conditions. We express $LLR(b_1|\mathbf{y}^r)$ as a ratio, as shown below in (4.9). Of the 8 possible bit-values for $\mathbf{b}$, we determine which conditions correspond to $b_1 = 0$ and $b_1 = 1$, respectively, and also weight each condition differently depending on whether it corresponds to NEB or EB. The j^{th} condition can result in $b_1 = 0$ in the following ways: either, it is an instance of NEB and b_1^r equals 0, or it is an EB condition and $f_j(b_1^r)$, as used in (4.8), equals 0. As explained before, the EB conditions are weighted less (by $w_{EB} = 0.5$ in (4.8)) than the NEB conditions (weighted by $w_{NEB} = 1$). We denote the number of NEB and EB conditions where $b_1 = b$ by $N_{NEB,b_1=b}$ and $N_{EB,b_1=b}$, respectively.

$$LLR(b_1|\mathbf{y}^r) = \log\left[\frac{P[b_1 = 0|\mathbf{y}^r]}{P[b_1 = 1|\mathbf{y}^r]}\right] = \log\left[\frac{\text{frequency of occurrence of } b_1 = 0}{\text{frequency of occurrence of } b_1 = 1}\right]$$
$$= \log\left[\frac{N_{NEB,b_1=0}w_{NEB} + N_{EB,b_1=0}w_{EB}}{N_{NEB,b_1=1}w_{NEB} + N_{EB,b_1=1}w_{EB}}\right], \text{ where}$$

$$N_{NEB,b_1=b} = |\{j : \text{ condition } j \text{ is NEB and } b_1^r = b,\ 0 \le j \le 7\}|, b \in \{0,1\},$$

$$N_{EB,b_1=b} = |\{j : \text{ condition } j \text{ is EB and } f_j(b_1^r) = b,\ 0 \le j \le 7\}|, b \in \{0,1\},$$

where $|\{\cdot\}|$ denotes the cardinality of the set $\{\cdot\}$,

and $w_{NEB}(\text{or } w_{EB}) =$ weight assigned to a NEB (or EB) condition $= 1$ (or 0.5)

To avoid the numerator or denominator in $LLR(b_1|\mathbf{y}^r)$ becoming zero, we add 1 to both of them.

$$\text{Using M3: } LLR(b_1|\mathbf{y}^r) = \log\left[\frac{1 + N_{NEB,b_1=0}w_{NEB} + N_{EB,b_1=0}w_{EB}}{1 + N_{NEB,b_1=1}w_{NEB} + N_{EB,b_1=1}w_{EB}}\right] \tag{4.9}$$

For QIM-RA and the M1 and M2 schemes in ME-RA, the LLR is expressed in terms of the scaling factor α. *For a given set of hiding conditions (QF_h, λ), the best possible α (α_{opt}) is that value which results in the highest data rate.* We experimentally observe that the α_{opt} values used for M1 and M2 (for M2, $\alpha_{opt} = 3, 3, 2, 2$ for $QF_h = 40, 50, 60, 70$) result in similar LLR values for M3. In (4.9), we add 1 to both the numerator and denominator to avoid the LLR becoming $\pm\infty$.

We provide two examples (Tables 4.6 and 4.7) to explain the steps involved in LLR computation using M3.

Example 2: The first of these examples is the same as used in Table 4.5. Focussing on b_1, $b_1^r = 0$ at the NEB conditions $\{0, 3, 4, 7\}$, therefore $N_{NEB,b_1=0} = 4$. Of the EB conditions, $f_j(b_1^r)$ equals 0 for conditions $\{2, 6\}$ and 1 for conditions $\{1, 5\}$, respectively - hence, $N_{EB,b_1=0} = 2$ and $N_{EB,b_1=1} = 2$. The resultant $LLR(b_1|\mathbf{y}^r)$ equals $\log\left(\frac{1 + 4w_{NEB} + 2w_{EB}}{1 + 2w_{EB}}\right) = 1.0986$, as shown in Table 4.6.

Example 3: Let $\mathbf{y}^r$ be $\{1.2, 0.2, 0.9, 1.5, 1.9, 0.1, 0.2\}$. Then, $\mathbf{a}^r = \{1, 0, 1, 0, 0, 0, 0\}$ and $\mathbf{b}^r = \mathbf{H}(\mathbf{a}^r)^T = \{0, 1, 0\}$. Embedding is seen to be possible for all the 8 cases (all

NEB conditions). For b_1 and b_3, $LLR(b_i|\mathbf{y}^r)$ equals $\log\left(\frac{1+8w_{NEB}}{1+0}\right) = 2.1972$. $LLR(b_2|\mathbf{y}^r)$ equals $\log\left(\frac{1+0}{1+8w_{NEB}}\right) = -2.1972$, as shown in Table 4.7.

Table 4.6: LLR computation for Method 3 based on (4.9) - explained using Example 2, and using $w_{EB}=0.5,\ w_{NEB}=1$

Example 2	$\mathbf{y}^r = \{0.4, 0.2, 0.9, 1.5, 0.3, 0.1, 1.2\}$ NEB conditions are $\{0, 3, 4, 7\}$
$LLR(b_1\|\mathbf{y}^r) =$	$\log\left(\frac{1+4w_{NEB}+2w_{EB}}{1+2w_{EB}}\right) = 1.0986$
$LLR(b_2\|\mathbf{y}^r) =$	$\log\left(\frac{1+4w_{NEB}+2w_{EB}}{1+2w_{EB}}\right) = 1.0986$
$LLR(b_3\|\mathbf{y}^r) =$	$\log\left(\frac{1+2w_{EB}}{1+4w_{NEB}+2w_{EB}}\right) = -1.0986$

Table 4.7: LLR computation for Method 3 based on (4.9) - explained using Example 3, and using $w_{EB}=0.5,\ w_{NEB}=1$

Example 3	$\mathbf{y}^r = \{1.2, 0.2, 0.9, 1.5, 1.9, 0.1, 0.2\}$ NEB conditions are $\{0, 1, 2, 3, 4, 5, 6, 7\}$
$LLR(b_1\|\mathbf{y}^r) =$	$\log\left(\frac{1+8w_{NEB}}{1+0}\right) = 2.1972$
$LLR(b_2\|\mathbf{y}^r) =$	$\log\left(\frac{1+0}{1+8w_{NEB}}\right) = -2.1972$
$LLR(b_3\|\mathbf{y}^r) =$	$\log\left(\frac{1+8w_{NEB}}{1+0}\right) = 2.1972$

4.5 Accounting for Channel Effects

Let us consider the flow $\mathbf{y} \rightarrow \mathbf{a} \rightarrow \mathbf{a}^e \rightarrow \mathbf{y}^e \rightarrow \mathbf{y}^r \rightarrow \mathbf{a}^r$, as shown in Fig. 4.2.

For the LLR computation algorithms described in Section 4.4.1 and 4.4.2, we

assume that there are no errors or erasures in the channel between $\mathbf{y}^e$ and $\mathbf{y}^r$. We now refine the LLR computation method, accounting also for channel effects.

For a received sequence $\mathbf{y}^r$, we can compute the LLR value for the 3 bit locations corresponding to these 7 coefficients using M1/M2/M3. However, considering channel effects, the received sequence $\mathbf{y}^r$ need not be the same as the transmitted sequence $\mathbf{y}^e$. Here, we guess the value of the transmitted sequence and refer to it as $\mathbf{y}^g$. For each guessed sequence $\mathbf{y}^g$, we compute the probability that the transmitted sequence is $\mathbf{y}^g$, given that the received sequence is $\mathbf{y}^r$. The final LLR value for $\mathbf{b}$, given the sequence $\mathbf{y}^r$ and considering the channel effects for all possible $\mathbf{y}^g$ sequences, $LLR_{final}(\mathbf{b}|\mathbf{y}^r)$ is computed as shown in (4.10).

LLR expression considering channel effects:

$$LLR_{final}(\mathbf{b}|\mathbf{y}^r) = \sum_{\mathbf{y}^g} LLR(\mathbf{b}|\mathbf{y}^g) P[\mathbf{y}^g|\mathbf{y}^r], \tag{4.10}$$

where $P[\mathbf{y}^g|\mathbf{y}^r]$ = Prob($\mathbf{y}^g$ is transmitted, given that $\mathbf{y}^r$ is received sequence)

(4.11)

We express $P[\mathbf{y}^g|\mathbf{y}^r]$ (4.11) in terms of the parameters in the 3×3 transition probability matrix T, for the channel between $\mathbf{y}^g$ and $\mathbf{y}^r$, where each coefficient can be mapped to one of $\{0, 1, e\}$ (e = erasure).

$$T = \begin{bmatrix} t_{00} & t_{01} & t_{0e} \\ t_{10} & t_{11} & t_{1e} \\ t_{e0} & t_{e1} & t_{ee} \end{bmatrix},$$

where $t_{ij} = P[m(y_k^r) = s_j | m(y_k^g) = s_i]$, and symbol set $s = \{0, 1, e\}$, (4.12)

and the mapping function $m(\cdot)$ is such that

$$m(y_k^r) = \begin{cases} 0/1 & \text{if } |y_k^r| > 0.5 \text{ and } \mathrm{mod}(\mathrm{round}(y_k^r), 2) = 0/1 \\ e & \text{if } |y_k^r| \leq 0.5 \end{cases} \tag{4.13}$$

Representing the sequences in terms of the ternary symbols, we consider only those $\mathbf{y}^g$ where 0 or 1 symbols are changed to obtain $\mathbf{y}^r$ from $\mathbf{y}^g$ to compute LLR_{final} (4.14) - we assume that the channel error and erasure probabilities are small enough so that $P[\mathbf{y}^g|\mathbf{y}^r] \approx 0$ when more symbols are changed.

LLR expression using 0/1 changes:

$$LLR_{final}(\mathbf{b}|\mathbf{y}^r) = \sum_{i=0}^{1} \left\{ \sum_{\mathbf{y}^g \in B_i(\mathbf{y}^r)} LLR(\mathbf{b}|\mathbf{y}^g) P[\mathbf{y}^g|\mathbf{y}^r] \right\} \tag{4.14}$$

where $B_i(\mathbf{y}^r) = \{\mathbf{y}^g : \mathbf{y}^g$ is obtained from $\mathbf{y}^r$ by changing any i elements in $\mathbf{y}^r\}$

The elements in T depend on the JPEG compression channel and not on the distribution of the DCT coefficients in the original image. For a given set of hiding parameters (QF_h, QF_a and λ), we compute T for each image, from a set of 500 images, and the average is taken to obtain a mean estimate of T.

We compute the probability terms $P[\mathbf{y}^g|\mathbf{y}^r]$ assuming that the symbols are independent of each other. Using $m(\mathbf{y}^g)$ and $m(\mathbf{y}^r)$ to denote the sequences of ternary symbols corresponding to $\mathbf{y}^g$ and $\mathbf{y}^r$ respectively, the probability $P[\mathbf{y}^g|\mathbf{y}^r]$ equals $\prod_{i=1}^{7} P[m(y_i^g)|m(y_i^r)]$.

For the T matrix, we assume $t_{00}=t_{11}$, $t_{01}=t_{10}$, $t_{e0}=t_{e1}$, $t_{0e}=t_{1e}$ (this is also experimentally verified). Let p_n denote the value of $P[\mathbf{y}^g|\mathbf{y}^r]$, when n LSBs from $\mathbf{y}^g$ are changed (through errors/erasures) to generate $\mathbf{y}^r$. The probability terms $p_n|_{n=0} = p_0$ and $p_n|_{n=1} = p_1$ are computed as shown below in (4.15) and (4.16), respectively.

$$p_0 = (t_{00})^{7-n_1}(t_{ee})^{n_1}, \text{ when } n_1 \text{ elements in } \mathbf{y}^r \text{ are in the erasure zone} \quad (4.15)$$

$$p_1 = \begin{cases} (t_{00})^{6-n_1}(t_{ee})^{n_1}t_{e0} & \text{if an erasure term in } \mathbf{y}^g \text{ is changed to 0/1 in } \mathbf{y}^r \\ (t_{00})^{6-n_1}(t_{ee})^{n_1}t_{0e} & \text{if a 0/1 in } \mathbf{y}^g \text{ is changed to an erasure in } \mathbf{y}^r \\ (t_{00})^{6-n_1}(t_{ee})^{n_1}t_{01} & \text{if a 0/1 in } \mathbf{y}^g \text{ is changed to a 1/0 in } \mathbf{y}^r \end{cases} \quad (4.16)$$

Experimental Setup: An output quality factor QF_a of 75 is used for all the experiments in this chapter. While performing experiments to account for the channel errors, we systematically increase the values of the design quality factor QF_h, as shown below in Table 4.8. As QF_h increases, the DCT coefficients are divided by a finer quantization matrix (the elements in the JPEG quantization matrix get smaller) - the perturbation introduced by a fixed JPEG channel

($QF_a = 75$) can cause more errors/erasures if the original elements undergo finer quantization. We show results for $QF_h = 60$, 70 and 75 - the error and erasure probabilities are much smaller for lower QF_h. For QF_h of 60, 70 and 75, QF_a being fixed at 75, and using an embedding band of $\lambda = 19$ elements, T (averaged over 500 images) equals

$$\begin{bmatrix} 0.9589 & 0.0333 & 0.0078 \\ 0.0332 & 0.9592 & 0.0076 \\ 0.0043 & 0.0043 & 0.9914 \end{bmatrix}, \begin{bmatrix} 0.9108 & 0.0733 & 0.0160 \\ 0.0737 & 0.9102 & 0.0161 \\ 0.0126 & 0.0125 & 0.9749 \end{bmatrix} \text{ and}$$

$$\begin{bmatrix} 0.8776 & 0.1022 & 0.0201 \\ 0.1031 & 0.8761 & 0.0208 \\ 0.0192 & 0.0191 & 0.9617 \end{bmatrix},$$

respectively. The average hiding rate is computed in terms of *bpnc* (explained in Table 4.1). "Hiding rate" refers to the bpnc computed per image while using the minimum redundancy (q_{opt}) for RA coding that ensures perfect data recovery. *From Table 4.8 onwards, the bpnc computation is averaged over 250 images - the image dataset is explained in Section 4.8.1.* Since considering the channel transition probability matrix improves the bpnc for more noisy channels, we use (4.14) for LLR computation for QF_h of 60 and 70 in further experiments. For lower QF_h values, the LLR is computed using the erasure-only model (4.9).

Table 4.8: Variation of the hiding rate (bpnc) with different LLR computation methods for (7,3) ME-RA, with B=9, QF_a=75, and using the first 19 AC DCT terms for hiding ($\lambda = 19$), and M3 to compute individual LLRs.

LLR Model / QF_h	without using T	using p_0	using p_0 and p_1
60	0.0731	0.0741	**0.0745**
70	0.0436	0.0493	**0.0498**
75	0.0265	0.0323	**0.0329**

Experimental Results: We show the usefulness of the assumed model (individual LLR terms $LLR(\mathbf{b}|\mathbf{y}^g)$ are computed using M3) in Table 4.8. The bpnc using both p_0 and p_1, as in (4.14), for LLR computation is slightly higher than that computed using only p_0, while both are significantly higher than the erasure-only model as in (4.9), especially for channels with a higher error rate ($QF_h = 70$ and 75).

4.6 Punctured RA Codes

We have explained how LLRs can be estimated assuming erasures and with/without channel errors. Here, we show how the embedding rate can be further improved by adding more erasures using a technique called "puncturing".

RA codes are near-optimal for our application because of the high proportion of erasures, but the available rates are limited to $\frac{1}{q}$, where q is an integer. We show in this section that we can address this shortcoming by the use of punctured RA codes. Puncturing [5, 6] is a technique where the effective code rate

$\left(\frac{\text{length of codeword}}{\text{length of dataword}}\right)$ is increased by deletion of some bits in the encoder output. The bits are deleted according to some puncturing matrix. We explain how the effective data rate can be increased through puncturing using an example. Assume that there are 200 embedding locations - for a RA codeword of 200 bits, let the value of q_{opt} be 4 and hence, we can embed $\frac{200}{4} = 50$ data-bits. Now, we increase the effective codeword length using "puncturing". Let the new codeword length be 300 - since only 200 bits can be embedded, we assume that the extra 100 bits are deleted (*these deletions are regarded as erasures at the decoder output*). As the effective noise channel is the same, the error and erasure rate for the 200 code-bits is unchanged while there are 100 additional erasures. We obtain a higher data-rate if the new value of $q_{opt} \leq 5$, as $\left\lfloor \frac{300}{5} \right\rfloor > 50$. *The design choices for puncturing here are (i) the number of additional erasures (the extra bits that are deleted) and (ii) their locations in a RA codeword.* By suitably puncturing the RA codeword which is embedded using ME-RA, we obtain a higher bpnc - this new approach is called "ME-RA-puncture". We first explain the algorithm and then describe why/how it results in an improved performance.

Algorithm Description: The steps in the algorithm are outlined below.

• The embedding band remains the same for ME-RA and ME-RA-puncture schemes. We use an embedding band of top 19 AC DCT coefficients per 8×8 block ($\lambda = 19$). Let the number of $B \times B$ blocks pseudo-randomly chosen by YASS

be N_B. The total number of hiding coefficients $N = 19N_B$ (assuming $B \leq 15$).

• To create a longer codeword than ME-RA (which has $3\lfloor \frac{N}{7} \rfloor$ code bits), we assume that η ($\eta \geq 3\lceil \frac{19}{7} \rceil$, i.e. $\eta \geq 9$) bits are embedded per 8×8 block. Hence, the codeword has ηN_B bits. The problem becomes one of distributing the $(\eta N_B - 3\lfloor \frac{N}{7} \rfloor)$ erasures among the ηN_B code bits.

• One can spread the erasures pseudo-randomly or the erasures can occur at consecutive locations (*bursty erasures*). Let L denote the set of locations which correspond to the $3\lfloor \frac{N}{7} \rfloor$ code bits which are embedded out of ηN_B code bits. The four methods that we explore to obtain L are as follows :

(i) **Locally Random Erasures (LRE):** we assume that out of a set of η consecutive code bits, $(\eta - 9)$ bits are erased. Let the 9 pseudo-randomly selected bit locations, decided based on a key known to the decoder, used for embedding out of η locations be $\{\ell_i\}_{i=1}^{9}$, where $\ell_i \in \{1, 2, \ldots, \eta\}$. Thus, the set of the bit locations (out of ηN_B locations) which correspond to the RA-code bits which are actually embedded is $L = \cup_{i=1,k=0}^{i=9,k=N_B-1}\{\ell_i + \eta k\}$. As $3\lfloor \frac{19N_B}{7} \rfloor < 9N_B$, we consider the first $3\lfloor \frac{19N_B}{7} \rfloor$ of the $9N_B$ locations in L to obtain the embeddable code bits.

(ii) **Locally Bursty Erasures (LBE):** We assume that out of η consecutive code bits, locations $\{1, 2, \ldots, 9\}$ are used for embedding and $\{10, \ldots, \eta\}$ are erasure locations.

(iii) **Globally Random Erasures (GRE):** Out of ηN_B locations, $3\lfloor \frac{N}{7} \rfloor$ loca-

tions are pseudo-randomly selected as the embedding locations. For LRE and GRE, the pseudo-random locations are decided based on a key shared between encoder and decoder, so that the decoder knows the additional erasure locations. (iv) **Globally Bursty Erasures (GBE):** We assume that out of ηN_B locations, locations $\{1, 2, \ldots, 3\lfloor \frac{N}{7} \rfloor\}$ are used for embedding while the remaining are erased.

- The LLR values for the locations where code bits are actually embedded (locations specified by L) are computed using M1, M2 or M3. The LLR values at the additional erasure locations are set to zero.

Understanding how "ME-RA-puncture" works: When can increasing the codeword length, while "actually embedding" the same number of bits, increase the effective hiding rate? Suppose that the size of the RA codeword for two different choices of "number of additional erasures" be $\eta_1 N_B$ and $\eta_2 N_B$ (we assume that both $\eta_1, \eta_2 \geq 9$ and $\eta_2 > \eta_1$). Let the value of q_{opt} for the two cases be $q_{opt,1}$ and $q_{opt,2}$, respectively. The number of data bits embedded is $\lfloor \eta_1 N_B / q_{opt,1} \rfloor$ and $\lfloor \eta_2 N_B / q_{opt,2} \rfloor$, respectively. As $\eta_2 > \eta_1$, the number of erasures introduced in the second case is higher (the channel becomes more noisy) and hence, the minimum redundancy needed $q_{opt,2}$ may be equal to or higher than $q_{opt,1}$. *Thus, the key to having a higher data rate using $\eta_2 > \eta_1$ is that the rate of increase in the redundancy factor q_{opt} should be less than the fractional increase in the code length, i.e.* $(q_{opt,2}/q_{opt,1}) < (\eta_2/\eta_1)$.

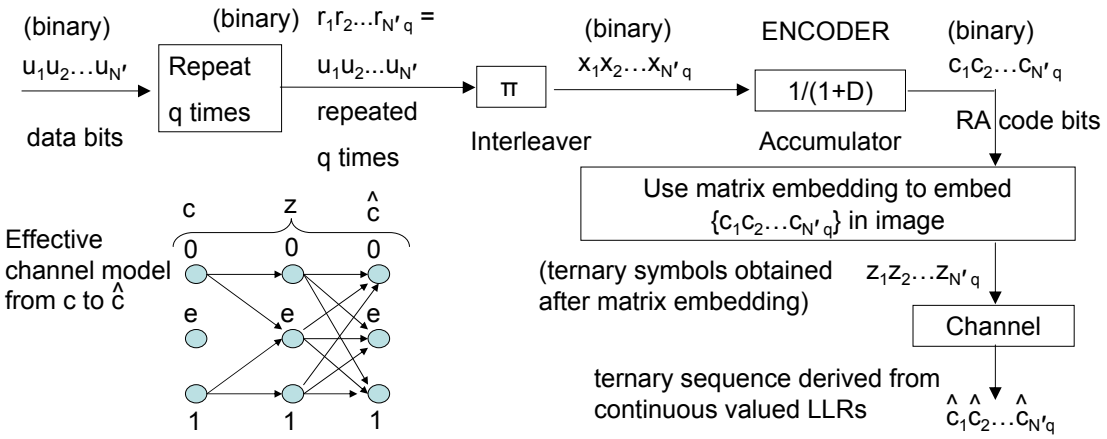


Figure 4.4: The mapping between the binary RA code bits to the ternary sequence obtained from the continuous valued LLRs at decoder output is shown here for the ME framework - "channel" refers to the JPEG compression channel that introduces errors and erasures in the mapping from $\mathbf{z}$ to $\hat{\mathbf{c}}$. The additional erasure locations for the **ME-RA-puncture** scheme are selected in the final RA encoded sequence ($\mathbf{c}$) and not in the intermediate sequences ($\mathbf{r}$ or $\mathbf{x}$). **For RA decoding, continuous valued LLRs are used - the ternary sequence ($\hat{\mathbf{c}}$) is used only to represent the channel as a 2×3 transition probability matrix**.

To understand how the redundancy varies after inserting additional erasures, we study the effective data hiding channel (the channel shown in Fig. 4.4 applies to both ME-RA and ME-RA-puncture), observe how the channel transition probability matrix changes with η and how it affects q_{opt}. The generation of the RA codeword form the data-bits has already been shown in Fig. 3.3. We repeat the RA-encoding framework here for ease of explanation.

For the QIM-RA scheme, the LLR values at the decoder side equal α, $-\alpha$, and 0 for an input symbol of 0,1 and e, respectively. For the ME-RA scheme, the LLR values are continuous valued and we empirically obtain a suitable threshold (δ) to

map the LLR values to ternary symbols. The continuous LLR values belonging to $[-\infty, -\delta), [-\delta, \delta]$ and $(\delta, \infty]$ are mapped to the 3 discrete values $-\alpha$, 0 and α, respectively. The 2×3 mapping from the binary RA code bits $\mathbf{c}$ to the ternary symbols $\mathbf{z}$ and then the 3×3 mapping between the ternary symbols ($\mathbf{z}$ and $\hat{\mathbf{c}}$) owing to the JPEG-based compression channel are shown in Fig. 4.4. The effective 2×3 mapping from $\mathbf{c}$ to $\hat{\mathbf{c}}$ is used to compute the effective channel capacity $\mathcal{C}$, which is obtained by maximizing the mutual information $I(\mathbf{c}, \hat{\mathbf{c}})$ between the sequences $\mathbf{c}$ and $\hat{\mathbf{c}}$. Recall (2.31) in Section 2.6.4 which showed how the capacity can be computed.

The inverse of the capacity ($\lceil \frac{1}{\mathcal{C}} \rceil$) provides the minimum redundancy factor needed for proper decoding for an ideal channel code - the RA code is expected to be close to the ideal channel code for channels with high erasure rates [25, 107]. The minimum q needed for RA decoding should be equal to or slightly higher than this redundancy factor. We empirically observe that using $\delta = 0.30$ provides a 2×3 transition probability matrix between $\mathbf{c}$ and $\hat{\mathbf{c}}$ that results in q_{opt} values close to $\lceil \frac{1}{\mathcal{C}} \rceil$.

Say, the overall transition probability matrix between $\mathbf{c}$ and $\hat{\mathbf{c}}$, for $\eta = 9$, is expressed as $\mathcal{P} = \begin{bmatrix} \rho_{0,0} & \rho_{0,1} & \rho_{0,e} \\ \rho_{1,0} & \rho_{1,1} & \rho_{1,e} \end{bmatrix}$. For $\eta = \eta'$, where $\eta' > 9$, out of every η' bit locations, 9 bits obey the mapping specified by $\mathcal{P}$, while the remaining $(\eta' - 9)$

bits always get erased. The modified transition probability matrix $\mathcal{P}'$ (for $\eta = \eta'$) is related with $\mathcal{P}$ as follows:

$$\mathcal{P}' = \begin{bmatrix} \rho'_{0,0} & \rho'_{0,1} & \rho'_{0,e} \\ \rho'_{1,0} & \rho'_{1,1} & \rho'_{1,e} \end{bmatrix} = \left(\frac{9}{\eta'}\right)\mathcal{P} + \left(\frac{\eta' - 9}{\eta'}\right)\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{9}{\eta'}\rho_{0,0} & \frac{9}{\eta'}\rho_{0,1} & \frac{9}{\eta'}\rho_{0,e} + (\frac{\eta'-9}{\eta'}) \\ \frac{9}{\eta'}\rho_{1,0} & \frac{9}{\eta'}\rho_{1,1} & \frac{9}{\eta'}\rho_{1,e} + (\frac{\eta'-9}{\eta'}) \end{bmatrix}$$

Let the channel capacity based on $\mathcal{P}$ and $\mathcal{P}'$ be denoted by $\mathcal{C}$ and $\mathcal{C}'$, respectively; we empirically observe that in general, $\frac{\mathcal{C}}{\mathcal{C}'} \approx \frac{\eta'}{9}$. Let $\mathcal{C}(\eta')$ denote the channel capacity using $\eta = \eta'$. The average values of $\frac{\mathcal{C}(9)}{\mathcal{C}(12)}$, $\frac{\mathcal{C}(9)}{\mathcal{C}(15)}$, $\frac{\mathcal{C}(9)}{\mathcal{C}(17)}$, $\frac{\mathcal{C}(9)}{\mathcal{C}(19)}$, $\frac{\mathcal{C}(9)}{\mathcal{C}(21)}$ and $\frac{\mathcal{C}(9)}{\mathcal{C}(23)}$ are 1.33, 1.66, 1.87, 2.11, 2.31 and 2.55 for $QF_h = 60$; for comparison, $\frac{\eta'}{\eta}$ equals 1.33, 1.67, 1.89, 2.11, 2.33 and 2.56 for η' of 12, 15, 17, 19, 21 and 23, respectively, where $\eta = 9$. *For the RA code framework, the redundancy q is constrained to be an integer.* Hence, it is seen for certain cases (numerical examples from Table 4.9) that even on inserting extra erasures, the RA code redundancy remains the same or increases at a rate lower than $\frac{\eta'}{9}$ leading to increased bpnc.

Numerical examples: For 4 sample images, η is varied from 9-23 and we observe how bpnc, $\mathcal{C}$ and q_{opt} vary, for QF_h of 60, as shown in Table 4.9. The LRE method is used to determine the embeddable bit locations. We set $B = 9$ and use M3 for individual LLR computation.

Table 4.9: For each image, the bpnc is computed using **ME-RA-puncture**, using QF_h=60 and B=9. The bpnc increases for a suitable range of η (rate of additional erasures). Here, **LRE** is used for erasure distribution.

Image 1	$\eta = 9$	$\eta = 12$	$\eta = 15$	$\eta = 17$	$\eta = 19$	$\eta = 21$	$\eta = 23$
$\mathcal{C}$	0.0976	0.0746	0.0572	0.0510	0.0450	0.0407	0.0386
$\lceil \frac{1}{\mathcal{C}} \rceil$	11	14	18	20	23	25	26
q_{opt}	17	18	21	22	25	29	31
$bpnc$	0.0588	0.0740	0.0793	**0.0858**	0.0844	0.0804	0.0824
Image 2	$\eta = 9$	$\eta = 12$	$\eta = 15$	$\eta = 17$	$\eta = 19$	$\eta = 21$	$\eta = 23$
$\mathcal{C}$	0.2610	0.1933	0.1589	0.1421	0.1266	0.1162	0.1029
$\lceil \frac{1}{\mathcal{C}} \rceil$	4	6	7	8	8	9	10
q_{opt}	6	7	8	9	10	12	14
$bpnc$	0.0801	0.0915	0.1001	0.1008	0.**1014**	0.0934	0.0877
Image 3	$\eta = 9$	$\eta = 12$	$\eta = 15$	$\eta = 17$	$\eta = 19$	$\eta = 21$	$\eta = 23$
$\mathcal{C}$	0.1128	0.0850	0.0694	0.0592	0.0532	0.0499	0.0448
$\lceil \frac{1}{\mathcal{C}} \rceil$	9	12	15	17	19	21	23
q_{opt}	11	15	17	19	20	24	27
$bpnc$	0.0816	0.0798	0.0880	0.0892	**0.0948**	0.0873	0.0850
Image 4	$\eta = 9$	$\eta = 12$	$\eta = 15$	$\eta = 17$	$\eta = 19$	$\eta = 21$	$\eta = 23$
$\mathcal{C}$	0.1301	0.0965	0.0774	0.0717	0.0630	0.0558	0.0501
$\lceil \frac{1}{\mathcal{C}} \rceil$	8	11	13	14	16	18	20
q_{opt}	10	14	16	18	20	21	23
$bpnc$	0.0736	0.0701	0.0767	0.0772	0.0777	**0.0818**	**0.0818**

Comparing Erasure Distribution Methods: In general, bursty erasures result in a lower bpnc as compared to when erasures are pseudo-randomly distributed (Table 4.10). For less noisy channels ($QF_h = 50, 60$), LRE and GRE perform much better than LBE and GBE. For more noisy channels ($QF_h = 70$), erasures located in globally consecutive positions (GBE) perform similar to/better than LRE and GRE schemes. We set $B = 9$, $QF_a = 75$ and use M3 for individual LLR computation.

Table 4.10: The bpnc values are computed using **ME-RA-puncture** for different erasure distribution methods, for varying QF_h and η, and B=9. The channel effects are considered for LLR computation for more noisy channels (QF_h of 60 and 70) using (4.14), while an ideal channel is assumed for QF_h of 50.

Method used	$QF_h = 50$			$QF_h = 60$			$QF_h = 70$		
	η=12	η=15	η=19	η=12	η=15	η=19	η=12	η=15	η=19
LRE	0.0864	**0.0935**	0.0960	0.0801	0.0867	0.0872	0.0519	0.0529	0.0488
LBE	0.0830	0.0836	0.0794	0.0758	0.0738	0.0698	0.0475	0.0440	0.0413
GRE	**0.0876**	0.0930	**0.0975**	**0.0811**	**0.0874**	**0.0916**	**0.0527**	**0.0536**	0.0485
GBE	0.0739	0.0731	0.0727	0.0728	0.0723	0.0716	0.0518	0.0520	**0.0537**

Experimental Results: Our experiments, performed on 250 images, show that as η is increased from 9, the bpnc increases significantly initially while it flattens out for η in the range 17-19, for "ME-RA-puncture" (Table 4.11). We use (4.14) for LLR computation for QF_h of 60 and 70 and use an erasure-only model for QF_h of 30, 40 and 50 (the same setup is again used in Table 4.12 and also in Section 4.8.2). We use LRE for choosing the embeddable code bits. We use M3 to compute the individual LLR values.

Table 4.11: The bpnc values are computed using **ME-RA-puncture** for different η and QF_h and **LRE** for erasure distribution; we set $B = 9$ and use M3 for individual LLR computation.

QF_h	$\eta = 9$	$\eta = 12$	$\eta = 15$	$\eta = 17$	$\eta = 19$	$\eta = 21$	$\eta = 25$
30	0.0594	0.0666	0.0756	**0.0777**	0.0776	0.0755	0.0750
40	0.0708	0.0785	0.0886	**0.0930**	0.0929	0.0895	0.0883
50	0.0766	0.0864	0.0935	0.0954	**0.0960**	0.0944	0.0923
60	0.0746	0.0801	0.0867	0.0870	**0.0872**	0.0863	0.0854
70	0.0499	0.0519	**0.0529**	0.0516	0.0488	0.0479	0.0456

Utility of ME-RA-puncture scheme: The same number of RA-encoded bits gets embedded for the ME-RA and ME-RA-puncture schemes; hence, the

effective embedding distortion and the detectability against steganalysis are identical for these methods. *Thus, for a proper choice of η, we obtain a higher data rate for ME-RA-puncture as compared to ME-RA even though both have the same detectability.*

4.7 Comparison of LLR Computation Methods

In Section 4.4, we introduced 3 methods for LLR computation (M1, M2 and M3). Here, we compare the effective hiding rate (in terms of bpnc) obtained using these methods in Table 4.12. M2 and M3 are more complicated than M1 in the way the different erasure scenarios are analyzed. Performance-wise, in general, **M1 < M2 < M3 (in terms of bpnc achieved using these methods)**. It is also seen that the performance benefits of ME-RA-puncture over ME-RA (in terms of increased bpnc) are higher for lower QF_h values, where the effect of erasures is more dominant. In Section 4.8.2, while reporting the steganalysis results using ME-RA-puncture, we also report its bpnc - for that, we use those parameters (erasure distribution method and η) which maximize the bpnc. We use GRE for erasure distribution for QF_h of 50 and 60 and GBE for QF_h of 70, and use η=19 (based on Tables 4.10 and 4.11).

Table 4.12: Table comparing the hiding rate (bpnc) obtained after using M1, M2 and M3, for LLR computation, using $B = 9$ and $QF_a = 75$. For M1 and M2, the optimum α (that results in highest bpnc for a set of hiding parameters) value for LLR scaling is empirically determined and the reported bpnc corresponds to the optimum α. For ME-RA and ME-RA-non-shrinkage, we use $\lambda = 19$, while for ME-RA-puncture, we use $\eta = 19$. The effective bpnc obtained using M3 is higher, in general, than that using M1 and M2.

Hiding Method	Decoding Method	QF_h=30	QF_h=40	QF_h=50	QF_h=60	QF_h=70
ME-RA	M1	0.0566	0.0652	0.0695	0.0690	0.0463
	M2	0.0578	0.0672	0.0728	0.0715	0.0490
	M3	**0.0592**	**0.0706**	**0.0766**	**0.0745**	**0.0498**
ME-RA-puncture (using LRE for QF_h of 30-60 and GBE for QF_h of 70)	M1	0.0744	0.0875	0.0917	0.0841	0.0492
	M2	0.0760	0.0889	0.0948	0.0847	0.0519
	M3	**0.0776**	**0.0929**	**0.0960**	**0.0872**	**0.0537**
ME-RA-non-shrinkage	M3	0.0662	0.0760	0.0789	0.0755	0.0433

Avoiding Shrinkage: The "shrinkage" concept has been explained in Section 4.4.1. The MMx algorithm [55] avoids shrinkage as follows - when the value of y_i is such that it lies in a non-erasure zone ([0.5,1] or [-1,-0.5]) but gets converted to zero after embedding, y_i is converted to 2 (or -2) depending on whether it is ≥ 0 (or < 0). Thus, shrinkage is avoided as a zero-valued coefficient can arise only due to erasure and not due to embedding; however, embedding distortion is also higher for the non-shrinkage case which leads to higher detectability. For LLR computation, we use $w_{EB} = 1$ for M3 (4.9) for the non-shrinkage case as there is no ambiguity between an erasure (coefficient is changed to zero) and an embedding (coefficient is changed to a non-zero term after embedding).

While comparing the performance of non-shrinkage ME-RA with ME-RA, we observe that the non-shrinkage version results in a higher bpnc only when the shrinkage problem is dominant, which happens when the erasure rate is high enough, i.e. at lower QF_h (of 30-60 in Table 4.12). The gain in bpnc (for ME-RA-non-shrinkage as compared to ME-RA) decreases as QF_h increases (erasure rate decreases) from 30-60. The embedding distortion of the non-shrinkage version is always higher than ME-RA, which in turn has the same embedding distortion as ME-RA-puncture. Hence, the non-shrinkage scheme is expected to be more detectable than ME-RA-puncture for the same steganalysis features. In Table 4.12, it is seen that the bpnc for ME-RA-puncture is higher than the non-shrinkage version across different QF_h.

4.8 Experiments and Results

The focus of these experiments are to demonstrate the following:

(i) The detectability of both the QIM-RA and the ME-RA-puncture schemes are compared against steganalysis, at similar hiding rates (shown later in Tables 4.13 and 4.14). The hiding rates are adjusted by varying B and the number of AC DCT coefficients used for hiding (λ).

(ii) The detection performance is also observed when hiding is performed in a

randomly selected set of AC DCT coefficients instead of always choosing the top λ coefficients (as returned by zigzag scan). This is demonstrated later through Tables 4.15 and 4.16.

(iii) *The level of noise attacks upto which ME performs better than QIM* is investigated, as shown later in Tables 4.17 and 4.18.

(iv) The steganalysis experiments are repeated using some recently proposed features, most of which were designed specifically to detect YASS (Table 4.19 and Table 4.20).

(v) We also observe how the steganalysis performance is improved on using a larger sized dataset for training, as shown in Table 4.21.

4.8.1 Setup for Steganalysis Experiments

The experiments are done on a set of 1630 high-quality JPEG images taken with a Canon S2-IS Powershot camera; the images were originally at a QF of 95 and they were JPEG compressed at a QF of 75 for the experiments [1]. The advertised QF (QF_a) is therefore kept at 75, so that both the cover and stego images, considered for steganalysis, are at the same JPEG QF.

[1]We have experimentally observed that the detectability is higher using high quality JPEG images than images taken with the same camera, but at poorer quality, i.e. JPEG compressed with lower QF. Hence, we use high-quality images for our experimental setup to show that ME-based YASS is more undetectable as compared to QIM-based YASS.

Steganalysis Performance Measures: The steganalysis results are expressed in terms of the detection probability P_{detect} (4.17) while the embedding rates are expressed in terms of the *bpnc*. We train a support vector machine (SVM) on a set of known stego and cover images. The SVM classifier has to distinguish between two classes of images: cover (class '0') and stego (class '1'). Let X_0 and X_1 denote the events that the image being observed belongs to classes '0' and '1', respectively. On the detection side, let Y_0 and Y_1 denote the events that the observed image is classified as belonging to classes '0' and '1', respectively. The probability of error P_{error} and the detection probability P_{detect} have already been defined before in Section 3.6.2. We repeat the definitions for ease of understanding. Here, we assume $P(X_0) = P(X_1) = \frac{1}{2}$:

$$\begin{aligned} P_{error} &= P(X_0)P(Y_1|X_0) + P(X_1)P(Y_0|X_1) = \frac{1}{2}P_{FA} + \frac{1}{2}P_{miss}, \\ P_{detect} &= 1 - P_{error} \end{aligned} \tag{4.17}$$

P_{detect} being close to 0.5 implies nearly undetectable hiding, and as the detectability improves, P_{detect} should increase towards 1. For the steganalysis results, we report P_{detect} as a percentage, at a precision of 2 significant digits after the decimal point.

Features Used for Steganalysis: The following features are used for steganalysis as these have generally been reported as having the best detection performance among modern JPEG steganalyzers.

1. **PF-219/324/274**: Pevny and Fridrich's 274-dim feature vector (**PF-274**) is based on the self-calibration method [83] and it merges Markov and DCT features. The extended DCT feature set and Markov features are 193-dim (**PF-193**) and 81-dim, respectively. The logic behind the fusion is that while Markov features capture the intra-block dependency among DCT coefficients of similar spatial frequencies, the DCT features capture the inter-block dependencies. For the extended DCT features [57, 83], the authors have a 219-dim implementation (**PF-219**) [2]. The Markov features (**PF-324**) are obtained based on the 324-dim intra-block correlation based feature set (**Shi-324**) proposed by Shi et al. [103] - the only difference being that the features are "calibrated" in [83].

2. **Chen-486**: Another steganalysis scheme that accounts for both intra and inter-block correlation among JPEG DCT coefficients is the 486-dim feature vector, proposed by Chen et al. [17]. It improves upon the 324-dim intra-block correlation based feature [103].

[2] **PF-219** differs from **PF-193** in the following ways: (i) in **PF-219**, there are 25 co-occurrence features for both the horizontal and vertical directions - these are averaged to give 25 features in **PF-193**. (ii) Instead of 1 variation feature in **PF-193**, there are 2 variation features (for horizontal and vertical directions, separately) in **PF-219**.

4.8.2 Discussion of Experimental Results

Comparison after Varying Big-block Size B: The detection performance, in terms of P_{detect} (4.17), and the embedding rate, in terms of bpnc, are compared for QIM-RA and "ME-RA-puncture", using $B = 9$ and 10 (Table 4.13), and 25 and 49 (Table 4.14). The ME based method has been experimented with for both the (7,3) and (3,2) encoding schemes. The "QIM-RA: n terms" scheme, to which ME-RA-puncture is compared to here, has been defined in Table 4.1.

From these tables, it is seen that P_{detect} is comparable for "QIM-RA: 2 terms" and "ME-RA-puncture (7,3)" while the latter has a higher embedding rate. The bpnc for "ME-RA-puncture (7,3)" (or ME-RA-puncture (3,2)) is higher than that of "QIM-RA: 4 terms" (or QIM-RA: 6 terms) while the latter is more detectable, for the self-calibration based features. It is seen that YASS is more detectable using the self-calibration based features, than using Chen-486. Hence, the performance improvement of ME over QIM (lower P_{detect} at similar bpnc values) is more significant for PF-219/324/274 features.

Depending on the bpnc requirements for a certain stego scheme, one can decide whether to use (3,2) or (7,3) matrix embedding - the former allows for higher bpnc while the latter is more undetectable. Using (15,4) code for ME results in very low hiding rates and hence has not been considered.

Table 4.13: Comparing detection performance (P_{detect}) and embedding rate (bpnc) using QIM-RA and "ME-RA-puncture" schemes - for B=9 and 10, QF_h=50, QF_a=75. The bpnc for "ME-RA-puncture (7,3)" (or ME-RA-puncture (3,2)) is higher than that of "QIM-RA: 4 terms" (or QIM-RA: 6 terms) while the latter is more detectable, for the self-calibration based features.

Hiding Scheme	big-block size B=9				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: 2 terms	69.45	65.52	67.73	52.39	0.0493
QIM-RA: 4 terms	80.00	77.18	79.39	56.20	0.0864
QIM-RA: 6 terms	81.84	77.67	84.05	57.55	0.1138
ME-RA-puncture (7,3)	64.79	65.40	69.45	55.95	**0.0975**
ME-RA-puncture (3,2)	74.97	72.27	78.65	61.60	**0.1200**
Hiding Scheme	big-block size B=10				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: 2 terms	68.83	65.28	67.85	52.52	0.0382
QIM-RA: 4 terms	78.53	74.97	77.91	55.09	0.0700
QIM-RA: 6 terms	78.90	79.26	79.39	55.95	0.0923
ME-RA-puncture (7,3)	63.31	68.83	67.61	54.36	**0.0805**
ME-RA-puncture (3,2)	73.87	78.77	78.77	59.02	**0.0998**

Table 4.14: Comparing P_{detect} and bpnc for **QIM-RA** and **ME-RA-puncture** - for B=25 and 49, QF_h=50, QF_a=75

Hiding Scheme	big-block size $B = 25$				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: 2 terms	71.53	66.38	71.17	53.74	0.0588
QIM-RA: 4 terms	82.70	79.26	82.45	57.55	0.1018
QIM-RA: 6 terms	84.29	80.61	86.26	59.51	0.1336
ME-RA-puncture (7,3)	72.15	73.99	75.95	59.14	**0.1106**
ME-RA-puncture (3,2)	77.30	82.82	81.35	62.54	**0.1382**
Hiding Scheme	big-block size $B = 49$				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: 2 terms	71.17	68.96	71.78	54.23	0.0606
QIM-RA: 4 terms	82.33	80.00	83.56	59.14	0.1048
QIM-RA: 6 terms	87.98	84.54	88.34	61.47	0.1379
ME-RA-puncture (7,3)	69.08	67.61	71.04	56.69	**0.1136**
ME-RA-puncture (3,2)	82.94	84.91	84.91	63.80	**0.1421**

Comparison after Further Randomization for QIM-RA: In the experiments discussed above, the top AC DCT elements, encountered after zigzag scan, are used for embedding. While the top DCT elements are generally higher in magnitude (non-erasure locations) making them suitable for embedding, most detection schemes (for example, PF-219/274) focus on these coefficients - hence, using these coefficients for hiding increases the hiding rate and also helps in detection. *To make detection more difficult, we choose a certain number of DCT terms randomly out of the top 19 coefficients for the "QIM-RA: rand-n" scheme.* In Tables 4.15 and 4.16, "QIM-RA: rand-n" refers to the QIM-RA scheme, where n randomly chosen AC DCT terms out of the top 19 are used for embedding. For this scheme, we vary n, the number of DCT coefficients in the embedding band, to make the hiding rate comparable to that resulting from the ME based scheme - the detection rates of the QIM and ME based methods are then compared. We experiment with hiding at $QF_h = 50$, 60 and 70, as shown in Tables 4.15 and 4.16.

From Table 4.15, "ME-RA-puncture(7,3)" (or ME-RA-puncture(3,2)) performs better than "QIM-RA: rand-8/10" (or QIM-RA: rand-12) - by having higher bpnc for similar P_{detect}, at QF_h of 50. From Table 4.16, "ME-RA-puncture(7,3)" performs better than "QIM-RA: rand-8" while "ME-RA-puncture(3,2)" performs better than "QIM-RA: rand-10/12", at QF_h of 60 and 70.

Table 4.15: Comparing P_{detect} and bpnc for QIM-RA and "ME-RA-puncture", for B=9 and 10, $QF_h = 50$, and using randomly chosen DCT coefficients for QIM-RA. "ME-RA-puncture(7,3)" performs better than "QIM-RA: rand-8/10" while "ME-RA-puncture(3,2)" performs better than "QIM-RA: rand-12".

Hiding Scheme	$B = 9, QF_h = 50$				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: rand-8	69.45	66.99	70.43	53.37	0.0700
QIM-RA: rand-10	71.04	73.13	74.36	55.83	0.0850
QIM-RA: rand-12	78.41	76.81	80.61	57.30	0.1115
ME-RA-puncture (7,3)	64.79	65.40	69.45	55.95	**0.0975**
ME-RA-puncture (3,2)	74.97	72.27	78.65	61.60	**0.1200**
Hiding Scheme	$B = 10, QF_h = 50$				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: rand-8	68.83	68.10	69.82	52.52	0.0530
QIM-RA: rand-10	78.16	78.28	80.25	55.34	0.0700
QIM-RA: rand-12	81.23	80.98	83.56	56.07	0.0880
ME-RA-puncture (7,3)	63.31	68.83	67.61	54.36	**0.0805**
ME-RA-puncture (3,2)	73.87	78.77	78.77	59.02	**0.0998**

Table 4.16: Comparing P_{detect} and bpnc for QIM-RA and "ME-RA-puncture", using B=9 and QF_h=60 and 70, and using randomly chosen DCT coefficients for QIM-RA. "ME-RA-puncture(7,3)" performs better than "QIM-RA: rand-8" while "ME-RA-puncture(3,2)" performs better than "QIM-RA: rand-10/12".

Hiding Scheme	$B = 9, QF_h = 60$				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: rand-8	74.72	73.13	74.72	53.99	0.0760
QIM-RA: rand-10	77.18	79.63	80.00	55.95	0.0913
QIM-RA: rand-12	79.63	85.03	85.77	64.79	0.1094
ME-RA-puncture (7,3)	63.34	64.61	66.27	55.09	**0.0916**
ME-RA-puncture (3,2)	73.55	71.81	75.12	62.82	**0.1161**
Hiding Scheme	$B = 9, QF_h = 70$				
	PF-219	PF-324	PF-274	Chen-486	bpnc
QIM-RA: rand-8	61.23	62.58	63.19	52.39	0.0430
QIM-RA: rand-10	64.79	65.64	66.50	53.37	0.0550
QIM-RA: rand-12	69.69	69.33	70.43	55.21	0.0720
ME-RA-puncture (7,3)	56.81	59.75	57.18	52.39	**0.0537**
ME-RA-puncture (3,2)	63.31	68.34	65.03	54.85	**0.0780**

Table 4.17: Comparing bpnc under various attacks - **"QIM-n"** refers to the **"QIM-RA: n terms"** method, while **(p,q)** refers to the **"ME-RA-puncture (p,q)"** method. For hiding, we use $QF_h = 50$, $B = 9$, and after the attack, the images are JPEG compressed using $QF_a = 75$. Here, the bpnc for "ME-RA-puncture(7,3)" and "ME-RA-puncture(3,2)" are compared with that of QIM-4 and QIM-6, respectively.

Gamma correction: $\gamma < 1$					Gamma correction: $\gamma > 1$				
γ	QIM-4	QIM-6	(7,3)	(3,2)	γ	QIM-4	QIM-6	(7,3)	(3,2)
0.99	0.0851	0.1125	0.0953	0.1191	**1.01**	0.0854	0.1125	0.0959	0.1194
0.98	0.0839	0.1105	0.0929	0.1171	**1.02**	0.0843	0.1114	0.0935	0.1171
0.95	0.0783	0.1013	0.0849	0.1069	**1.05**	0.0799	0.1026	0.0863	0.1095
0.90	0.0608	0.0799	0.0655	0.0845	**1.10**	0.0631	0.0827	0.0685	0.0893
0.80	0.0307	0.0401	0.0200	0.0250	**1.20**	0.0366	0.0465	0.0260	0.0400

Robustness Comparison for Various Noise Attacks: We now study how the bpnc is affected by additional noise attacks for these schemes. The YASS framework can be made robust against various global (*and not local*) attacks by adjusting the RA-code redundancy factor. We consider a wider range of attacks - gamma variation and additive white Gaussian noise (AWGN) attacks, which are followed by JPEG compression at QF_a=75. It is seen that for higher noise levels, ($|\gamma - 1| > 0.10$, for gamma variation, or SNR $\leq$ 35 dB, for AWGN) the bpnc is significantly lower for the ME based method, as compared to QIM-RA, for similar detection rates (Tables 4.17 and 4.18).

Using Recent Steganalysis Features more tuned to detect YASS: We explain the following features and then show the steganalysis performance using these features in Tables 4.19 and 4.20:

(i) **KF-548**: To improve upon the **PF-274** feature, Kodovsky and Fridrich [58]

Table 4.18: We repeat the experiments in Table 4.17 and replace the gamma variation attack by AWGN addition based attack.

AWGN attack: SNR (dB)				
SNR	QIM-4	QIM-6	(7,3)	(3,2)
50	0.0860	0.1133	0.0952	0.1190
45	0.0859	0.1125	0.0940	0.1179
40	0.0840	0.1096	0.0885	0.1126
35	0.0728	0.0912	0.0659	0.0889
30	0.0393	0.0485	0.0200	0.0260

proposed the use of a 548-dimensional feature set which accounts for both calibrated and uncalibrated features. Here, the reference feature is used as an additional feature instead of being subtracted from the original feature.

(ii) **Li-14** and **Li-2**: In [63], Li et al. propose the use of the frequency of re-quantized DCT coefficients in the candidate embedding band which round off to zero. The $(2i-1)^{th}$ and $(2i)^{th}$ features correspond to B of $(8+i)$, for $1 \leq i \leq 7$. Thus, if we are sure that $B = 9$, we use the first two dimensions of **Li-14**, i.e. **Li-2**; else when the exact value of B is not known, the 14-dim feature is used.

(iii) **YB-243**: In [126], Yu et al. propose the use of a 243-dim feature based on transition probability matrices computed using the difference matrix computed in the pixel and DCT domains.

It is seen that in the lower embedding rate regime for which ME performs better than QIM, these newer features (**KF-548** and **Li-2**) provide similar levels of detectability as that provided by features already discussed, like **PF-274**.

Table 4.19: Comparing P_{detect} for a variety of recently proposed features to detect YASS, using $QF_h = 50$. We use B=9 for the QIM schemes. The acronyms used for the various methods are the same as used in Table 4.17.

Feature	QIM-2	QIM-4	QIM-6	QIM-12	QIM-19	(7,3),B=9	(3,2),B=9	(3,2),B=10
KF-548	68.45	79.48	83.82	89.20	92.03	69.61	80.15	78.97
Li-14	54.01	55.27	56.75	59.74	67.65	52.88	59.93	55.76
Li-2	64.43	69.49	77.51	81.19	96.08	68.83	76.05	71.08
YB-243	54.64	55.70	56.75	58.12	69.89	54.23	59.68	56.12

Table 4.20: Comparing P_{detect} for a variety of recently proposed features to detect YASS, using $QF_h = 70$. We use B=9 for the QIM schemes.

Feature	QIM-2	QIM-4	QIM-6	QIM-12	QIM-19	(7,3),B=9	(3,2),B=9	(3,2),B=10
KF-548	59.85	63.97	67.65	77.21	78.70	57.83	66.18	61.52
Li-14	50.49	50.67	50.85	54.17	58.70	49.94	50.00	51.35
Li-2	53.37	58.22	71.85	76.91	81.00	58.39	69.80	53.79
YB-243	50.55	51.22	51.68	54.82	59.35	51.90	52.70	51.52

The detection results are also shown for a variety of QF_h in Tables 4.19 and 4.20. It is generally seen that the detection accuracy is higher when $QF_h = 50$ (QF_a is fixed at 75), while it decreases generally as we increase QF_h from 50 to 70.

Effect of Varying the Size of the Training Dataset: The training dataset now has 1850 images instead of (1630/2) 815 images, while the test set remains the same. The additional images are generated in the same way as the original set of 1630 images. In Table 4.21, we observe that there is marginal increase in the detection accuracy after increasing the size of the training dataset by more than a factor of 2.

Table 4.21: Comparing P_{detect} for two different sized datasets, using a variety of steganalysis methods, and different variants (embedding methods) of the YASS scheme - B=9 is used along with QF_h=50 and QF_a=75.

Hiding Scheme	815 training images				
	PF-274	Chen-486	KF-548	Li-2	YB-243
QIM-RA: 2 terms	67.73	52.39	68.45	64.43	54.64
QIM-RA: 4 terms	79.39	56.20	79.48	69.49	55.70
QIM-RA: 6 terms	84.05	57.55	83.82	77.51	56.75
ME-RA-puncture (7,3)	69.45	55.95	69.61	68.83	54.23
ME-RA-puncture (3,2)	78.65	61.60	80.15	76.05	59.68
Hiding Scheme	1850 training images				
	PF-274	Chen-486	KF-548	Li-2	YB-243
QIM-RA: 2 terms	70.31	54.60	69.57	65.65	54.80
QIM-RA: 4 terms	81.60	59.88	79.75	71.78	55.75
QIM-RA: 6 terms	85.15	61.60	84.56	78.22	57.30
ME-RA-puncture (7,3)	70.35	60.86	71.66	67.47	54.40
ME-RA-puncture (3,2)	78.80	62.79	81.00	78.00	60.05

Performance Comparison after Puncturing: We have employed puncturing for the ME-RA framework but not for the QIM-RA scheme. We now use puncturing for QIM-RA ("QIM-RA: n terms" scheme) and compare the bpnc results for ME-RA and QIM-RA, both with and without puncturing, in Table 4.22. From Tables 4.13 and 4.14, ME-RA-puncture is found to be less detectable than QIM-RA and also has higher bpnc. After using puncturing, we observe that the bpnc gain margin (of ME-RA-puncture over QIM-RA-puncture) decreases - however, in general, ME-RA-puncture is still less detectable (puncturing does not affect the detectability) than QIM-RA-puncture at similar bpnc values.

Table 4.22: The bpnc values are compared for **ME-RA** and **QIM-RA** methods, before and after puncturing, at QF_h=50.

Hiding Scheme	$B = 9$		$B = 10$		$B = 25$		$B = 49$	
	before	after	before	after	before	after	before	after
QIM-RA: 2 terms	0.0493	0.0572	0.0382	0.0438	0.0588	0.0670	0.0606	0.0683
QIM-RA: 4 terms	0.0864	0.0965	0.0700	0.0784	0.1018	0.1110	0.1048	0.1134
QIM-RA: 6 terms	0.1138	0.1206	0.0923	0.0999	0.1336	0.1392	0.1379	0.1427
ME-RA (7,3)	0.0766	0.0975	0.0634	0.0805	0.1000	0.1106	0.1050	0.1136
ME-RA (3,2)	0.1100	0.1200	0.0900	0.0998	0.1300	0.1382	0.1350	0.1421

Fig. 4.5(a) illustrates how ME outperforms QIM in the "bpnc vs P_{detect}" trade-off. Considering points along the same vertical line (equal P_{detect}), the ME-points have higher y-values than the QIM-points, indicating higher bpnc. Fig. 4.5(b) corresponds to Table 4.13 - ME (7,3) (which actually corresponds to ME-RA-puncture (7,3)) is shown to be less detectable than QIM-2 (QIM-RA: 2 terms) and QIM-4 from ROC curves while Table 4.13 shows that ME (7,3) achieves higher bpnc than these QIM-based schemes. The variation in detectability with the hiding parameters (B, λ, (7,3) ME or (3,2) ME) for ME and QIM-based schemes is shown in Figs. 4.5(c) and 4.5(d), respectively.

To conclude, *for hiding conditions where the embedding rate has to be low enough to ensure a certain level of undetectability, ME based embedding with suitable puncturing generally results in higher bpnc than QIM, for similar robustness levels against steganalysis.* However, this holds true only when the channel noise introduced by the active adversary is low enough - for more severe noise, the LLR estimation for ME is erroneous enough to result in a lower hiding rate than QIM.

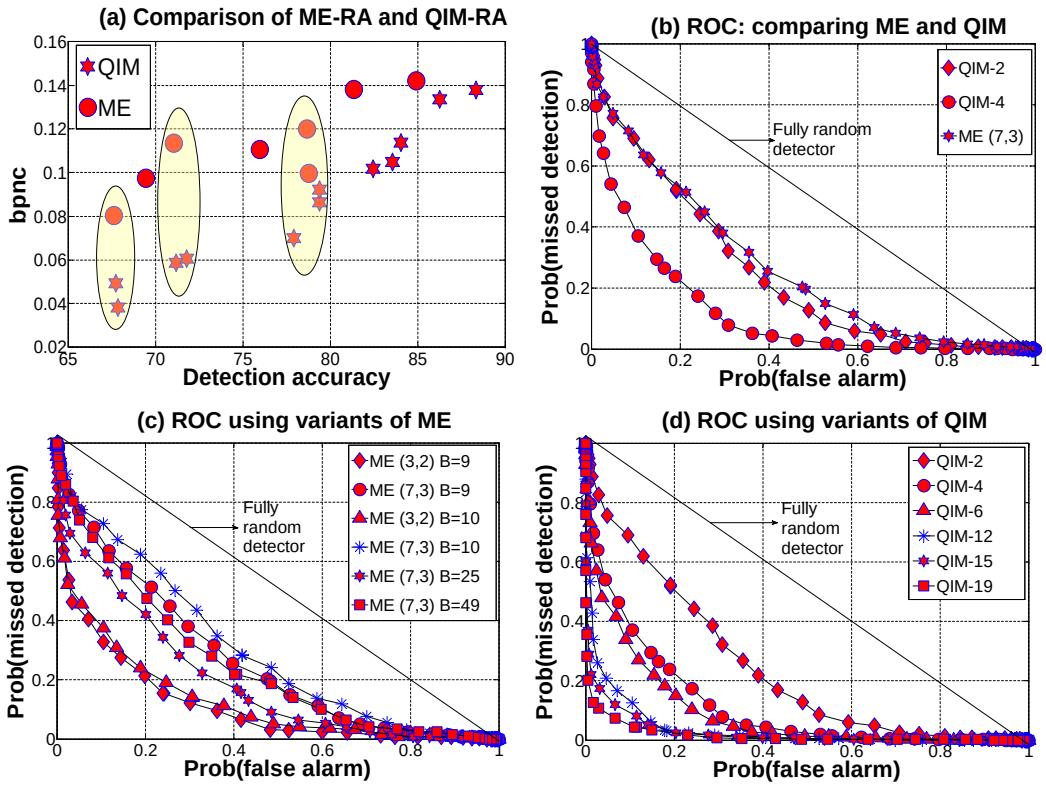


Figure 4.5: PF-274 is used for steganalysis in these plots - (a) trade-off of hiding rate vs detection accuracy is shown considering different parameter settings for ME and QIM based hiding, as used in Tables 4.13-4.14, (b-d) comparison of ROCs is done for (b) ME vs QIM at comparable bpnc, (c) variants of ME, and (d) variants of QIM (varying λ). Here, B=9 unless otherwise mentioned. The diagonal line corresponding to a fully random detector is kept as reference - the closer a ROC curve is to this line, more secure is the hiding method.

4.9 Estimating Redundancy Factor for ME-RA

We have already considered the problem of q-estimation for a RA-encoded sequence for the QIM-RA scheme in Section 3.5. Here, we extend that solution for the ME-RA scheme. The proposed q-estimation framework for ME-RA is a

modification of the approach used for the QIM-RA scheme.

System Framework:

Fig. 4.6 shows the end-to-end hiding framework except for the iterative decoding part at the RA decoder. If there are ℓ possible hiding locations, and (7,3) ME is used, the actual number of coded bits that can be embedded $\ell' = \lfloor \ell \times 3/7 \rfloor$. For a redundancy factor of q, the number of data bits for (7,3) ME, $N' = \lfloor \ell'/q \rfloor$. On inputting $\{u_n\}_{n=1}^{N'}$, the sequence of N' data bits to a q-times repeater block, the output is $\{r_n\}_{n=1}^{N'q}$ (4.18), which is then passed through the interleaver π. The resulting sequence is $\{x_n\}_{n=1}^{N'q}$ (4.19), which is then passed through the accumulator $(\frac{1}{1+D})$ to produce $\{c_n\}_{n=1}^{N'q}$ (4.20), the encoded output. The mapping from $\{u_n\}$ to $\{c_n\}$ at the encoder have already been explained in Section 3.5, and are repeated for convenience.

Steps involved in mapping from $\{u_n\}$ to $\{c_n\}$ at the encoder

$$[r_{(i-1)N'+1} r_{(i-1)N'+2} \ldots r_{iN'}] = [u_1 u_2 \ldots u_{N'}],\ 1 \leq i \leq q \tag{4.18}$$

$\{x_1, x_2, \cdots, x_{N'q}\} = \pi(\{r_1, r_2, \cdots, r_{N'q}\})$, where π is the interleaver function

(4.19)

$$c_1 = x_1,\ c_n = c_{n-1} \oplus x_n,\ 2 \leq n \leq N'q \tag{4.20}$$

The RA-encoded sequence $\{c_n\}$ is embedded in the image using matrix embedding. After embedding, we get a ternary sequence $\{z_n\}$ of $\{0, 1, e\}$ based on what is actually embedded, where e denotes an erasure (Fig. 4.6). The sequence of LLR values obtained from the hiding locations in the noisy received image is called $\{\hat{c}_n\}$.

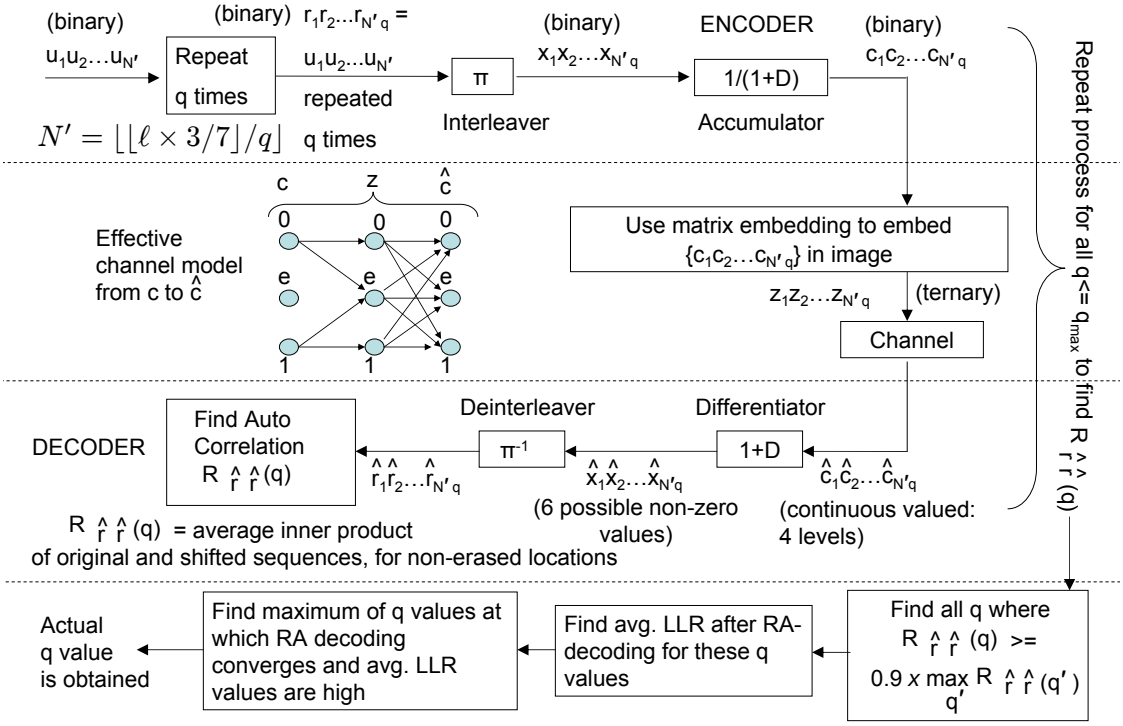


Figure 4.6: The data hiding setup for ME-RA method where the decoder has to correctly estimate the q that was independently decided upon by the encoder.

Decoder Outputs:

The LLR values, computed for ME-RA using M1, M2 or M3, are continuous valued. However, for providing the sequence of the LLR values $\{\hat{c}_n\}$ as input to

the differentiator $(1+D)$, the LLR values have to be quantized into a finite number of levels. The values in $\{\hat{c}_n\}$ are split into 4 zones $([-\infty, -\delta), [-\delta, 0), (0, \delta], (\delta, \infty])$, based on a suitably chosen threshold δ. Here, instead of a binary output for the $(1 + D)$ block, the elements in $\{\hat{x}_n\}$ are assumed to have positive (instead of 1) and negative (instead of 0) values. The magnitude of the $\{\hat{x}_n\}$ terms is quantized to 3 levels $\{\mathcal{L}, \mathcal{M}, \mathcal{H}\}$, where $\mathcal{L} < \mathcal{M} < \mathcal{H}$. These values indicate the confidence we have in the decoded $\{\hat{x}_n\}$ values based on the LLR terms $\{\hat{c}_n\}$. For example, if both $\hat{c}_n$ and $\hat{c}_{n-1}$ are positive, then $\hat{x}_n$ should be negative ($1 \oplus 1 = 0$, here "positive" $\Leftrightarrow 1$ and "negative" $\Leftrightarrow 0$). If both $\hat{c}_n$ and $\hat{c}_{n-1}$ exceed δ, we have high confidence in both these terms being positive and $\hat{x}_n$ being negative and hence, $\hat{x}_n = -\mathcal{H}$. The various possibilities are covered in (4.21), where we show how the terms in $\{\hat{x}_n\}$ are computed from $\{\hat{c}_n\}$.

$$\begin{aligned}
\hat{x}_n \;&>\; 0 \text{ if } \hat{c}_n \times \hat{c}_{n-1} < 0, && |\hat{x}_n| = \mathcal{L} \text{ if } |\hat{c}_n| < \delta, \text{ and } |\hat{c}_{n-1}| < \delta \\
&<\; 0 \text{ if } \hat{c}_n \times \hat{c}_{n-1} > 0, && = \mathcal{M} \text{ if } |\hat{c}_n| \geq \delta \text{ and } |\hat{c}_{n-1}| < \delta, \text{ or vice versa} \\
&=\; 0 \text{ if } \hat{c}_n \times \hat{c}_{n-1} = 0, && = \mathcal{H} \text{ if } |\hat{c}_n| \geq \delta, \text{ and } |\hat{c}_{n-1}| \geq \delta
\end{aligned} \tag{4.21}$$

Correlation Computation:

The next issue is computing the correlation function $R_{\hat{r},\hat{r}}(q')$, where the sequence $\{\hat{r}_n\}$ is obtained after deinterleaving $\{\hat{x}_n\}$, where $\{\hat{r}_n\} = \pi^{-1}(\{\hat{x}_n\})$. The corre-

lation between $\{\hat{r}_n\}$ and its shifted sequence is computed as a normalized inner-product.

For an assumed redundancy factor of q', we perform element-by-element multiplication between the 2 sequences, $\{\hat{r}_1 \ldots \hat{r}_{kq'}\}$ and $\{\hat{r}_{kq'-k+1} \ldots \hat{r}_{kq'} \hat{r}_1 \ldots \hat{r}_{kq'-k}\}$, (shift $k = \lfloor \ell'/q' \rfloor$ is the assumed number of data bits) to obtain the sequence $\{s_{q'}\}$ (4.22) and then the average is taken over the non-zero elements in $\{s_{q'}\}$ to compute $R_{\hat{r},\hat{r}}(q')$ (4.23). The zeroes in $\{s_{q'}\}$ correspond to those locations where at least one of the corresponding elements in the original and shifted $\{\hat{r}_n\}$ sequences are erased. Thus, a main difference between the q-estimation strategies for QIM-RA and ME-RA is that the correlation used for QIM-RA is an inner-product between binary sequences while for ME-RA, it is an inner-product between continuous valued sequences.

$$s_{q',i} = \hat{r}_i \times \hat{r}_{kq'-k+i},\ 1 \leq i \leq kq', \tag{4.22}$$

where shift $k = \lfloor \ell'/q' \rfloor$ is the assumed number of data bits

$$R_{\hat{r},\hat{r}}(q') = (\sum_i s_{q',i})/(\text{number of non-zeros in } \{s_{q'}\}) \tag{4.23}$$

$$\mathcal{Q}_{top} = \left\{ q' : R_{\hat{r},\hat{r}}(q') >= 0.9 \times (\max_{q_1 \in \{q\}} R_{\hat{r},\hat{r}}(q_1)) \right\} \tag{4.24}$$

where $\{q\} = \{1, 2, \cdots, q_{max}\}$ is the set of possible q-values, assuming a maximum q of q_{max}.

Let the actual q value used by RA coding equal q_{act}. In a noise-free scenario, the $\hat{r}_n$ values will be high or low depending on whether the corresponding values in $\{r_n\}$ are 1 or 0. The correlation $R_{\hat{r},\hat{r}}(q')$ is high when the shift ($k = \lfloor \ell'/q' \rfloor$) equals a multiple of the actual message length, i.e. $q'{=}q_{act}/m,\ m \in \mathbb{Z}^+$. Hence, we can expect peaks at q_{act} and/or its sub-multiples. In practice, due to errors and erasures, peaks can occur at other q values. *Hence, the correlation is used to prune the search space for q_{act} but is not the only deciding criterion.*

Selecting Right Value for Redundancy Factor:

After the correlation, we follow exactly the same method to find the actual q value, as was proposed in Section 3.5. The ME-RA and QIM-RA methods differ in how the correlation (4.23) is computed. Once it is computed, the other steps of obtaining a smaller set of q values $\mathcal{Q}_{top}$ (4.24) and then finding the best q value by comparing the average LLR values remain the same.

Performance Comparison:

To compare the relative performance of q-estimation, based on *only correlation*, between QIM-RA and ME-RA, we use the separation in the correlation peaks as an index. We also find the number of times there is an error in q-estimation using just the correlation (the q corresponding to the maximum correlation value

is chosen as q_{act}). To reiterate, these errors are rectified once the final q estimation is done based on the maximum average LLR values.

Let $\mathcal{A}$ and $\mathcal{B}$ denote the two sets - $\{q_{act}/m,\ m \in \mathbb{Z}^+\}$ and $\{\{q\}\backslash\{q_{act}/m\}_{m\in\mathbb{Z}^+}\}$, corresponding to "correct" and "wrong" choice of q values, respectively. When the maximum correlation value comes from an element in $\mathcal{A}$, we classify it as "correct"; otherwise, it is an error. Also, to quantify the discriminability between elements in $\mathcal{A}$ and $\mathcal{B}$ provided by the correlation based approach, we compute the difference, R_{diff} (4.25), between the topmost correlation values among elements in $\mathcal{A}$ and $\mathcal{B}$, for both the "correct" and "wrong" cases.

$$R_{diff} = \max_{q'\in\mathcal{A}} R_{\hat{r},\hat{r}}(q') - \max_{q'\in\mathcal{B}} R_{\hat{r},\hat{r}}(q') \tag{4.25}$$

For the correct/wrong cases, we would want the value of R_{diff} to be higher/lower, respectively.

We use the (7,3) ME-RA scheme, with M3 based decoding, for the q-estimation experiments. Suitable values for $\delta, \mathcal{L}, \mathcal{M}$ and $\mathcal{H}$ are empirically determined: $\delta = 0.2, \mathcal{L} = 0.10, \mathcal{M} = 0.75$ and $\mathcal{H} = 1.90$. The results are shown for 3 cases - QF_h is varied from 50 to 70 and QF_a is set to 75 (Table 4.23). Table 4.23 shows that ME-RA performs better than QIM-RA, both in terms of having lesser errors and in having better separation in the correlation peaks, between elements in $\mathcal{A}$ and $\mathcal{B}$.

Table 4.23: The results of estimating q over 50 images, where the q used for RA coding is varied from 10-43, are presented here, for (7,3) ME-RA and QIM-RA methods. Thus, the total number of cases over which q is estimated $= 50 \times 34 = 1700$. The big-block size B is set to 9, while QF_a=75. An "error" occurs when the top peak in the correlation based method does not correspond to the actual q or its sub-multiples. Based on just the correlation, ME-RA performs better than QIM-RA in q-estimation.

QF_h	Method	error fraction	avg. R_{diff} (correct cases)	avg. R_{diff} (wrong cases)
50	QIM-RA	50/1700	0.1500	-0.2972
	ME-RA	23/1700	1.0431	-0.1835
60	QIM-RA	151/1700	0.0775	-0.2664
	ME-RA	91/1700	0.5272	-0.2111
70	QIM-RA	393/1700	0.0198	-0.3083
	ME-RA	364/1700	0.1339	-0.1663

4.10 Summary

Randomized block-based hiding as in YASS provides a powerful framework for secure hiding, especially against self-calibrating steganalysis. Here, we have shown that using ME instead of QIM within the YASS framework provides improved steganalysis performance in certain regimes, specifically when avoiding detection is a high priority (so that the hiding rate is small) and attacks are moderate. The key to our approach is to combine ME-based data hiding, which has a high embedding efficiency but is relatively fragile in the face of attacks, with a powerful channel code employing soft decisions. While we use RA codes as in our prior work, *the LLR computation framework developed here, which depends only on the ME embedding logic, is equally applicable to any channel code whose decoder*

employs soft decisions (for example, turbo codes or low density parity check codes). The performance is further improved by the use of punctured RA codes in the ME-RA-puncture scheme. While punctured RA codes have been used previously for obtaining good high-rate codes for classical communication channels [84], our results demonstrate their potential benefits for low-rate data hiding channels. Thus, it would be interesting to examine their application to other steganographic schemes.

Scope for Future Work: One approach to gain further performance improvements is to address the shrinkage problem in YASS: when given a zero coefficient, the decoder is confused as to whether the zero resulted from an embedding or due to an erasure. Fridrich et al. have used wet paper codes (WPC) [38, 39] to overcome this problem, as in the "non-shrinkage F5" method [40]. Combining WPC with the ME-RA framework might lead to further improvement in the embedding rate while maintaining the undetectability of the stego scheme. Another approach is to use more sophisticated "inner codes", possibly combining error correction with hiding as in [127], with RA or other turbo-like codes used as outer codes. However, the combinatorial complexity of computing soft decisions (at least in the direct fashion considered here) for such an inner code would be excessive for larger blocklengths and a larger number of data bits. An interest-

ing topic for future research might be to explore techniques for overcoming this complexity bottleneck.

Chapter 5

Robust Key-point based Data Hiding

The earlier chapters (Chapter 2, 3 and 4) all focus on global attacks. For global attacks, the correspondence between a coefficient used for hiding at the embedder side and the same coefficient at the receiver side is not affected by the attack. Since we perform block-based hiding in statistical restoration and YASS, the correspondence between the hiding coefficients at the encoder and the decoder is essential for successful decoding. This synchronization is affected by attacks such as cropping or geometric transformations. Such attacks are indeed commonplace and we consider them in this chapter.

In this chapter, we present a robust data-hiding method which can survive a host of global, geometric and image editing attacks. The basic technique that allows synchronization after geometric transformation is the embedding of frequency domain peaks at pre-decided frequency locations. These peaks are then recovered at the decoder side and used to estimate the geometric transformation encountered. This method works for global affine transformations. The second issue is that to achieve robustness against cropping and other local attacks, we need to repeat the embedding in multiple local regions. The receiver needs to have access to the same local regions as the embedder for decoding the embedded content. This is achieved by using suitable key-points (KP) as the anchor points and selecting regions of pre-decided dimensions as the hiding regions containing the key-points at their center.

The primary contributions of the work include novel methods for introducing frequency domain synchronization information that can be recovered at the decoder without any side information. The main challenge facing such frequency domain techniques is that attacks like JPEG compression can introduce peaks owing to the JPEG-induced periodicity, which can interfere with the synchronization peaks and cause errors in estimating the geometric transformation. We propose solutions for proper synchronization even after such attacks. Another challenge is having access to the same key-points as used by the encoder at the receiver side.

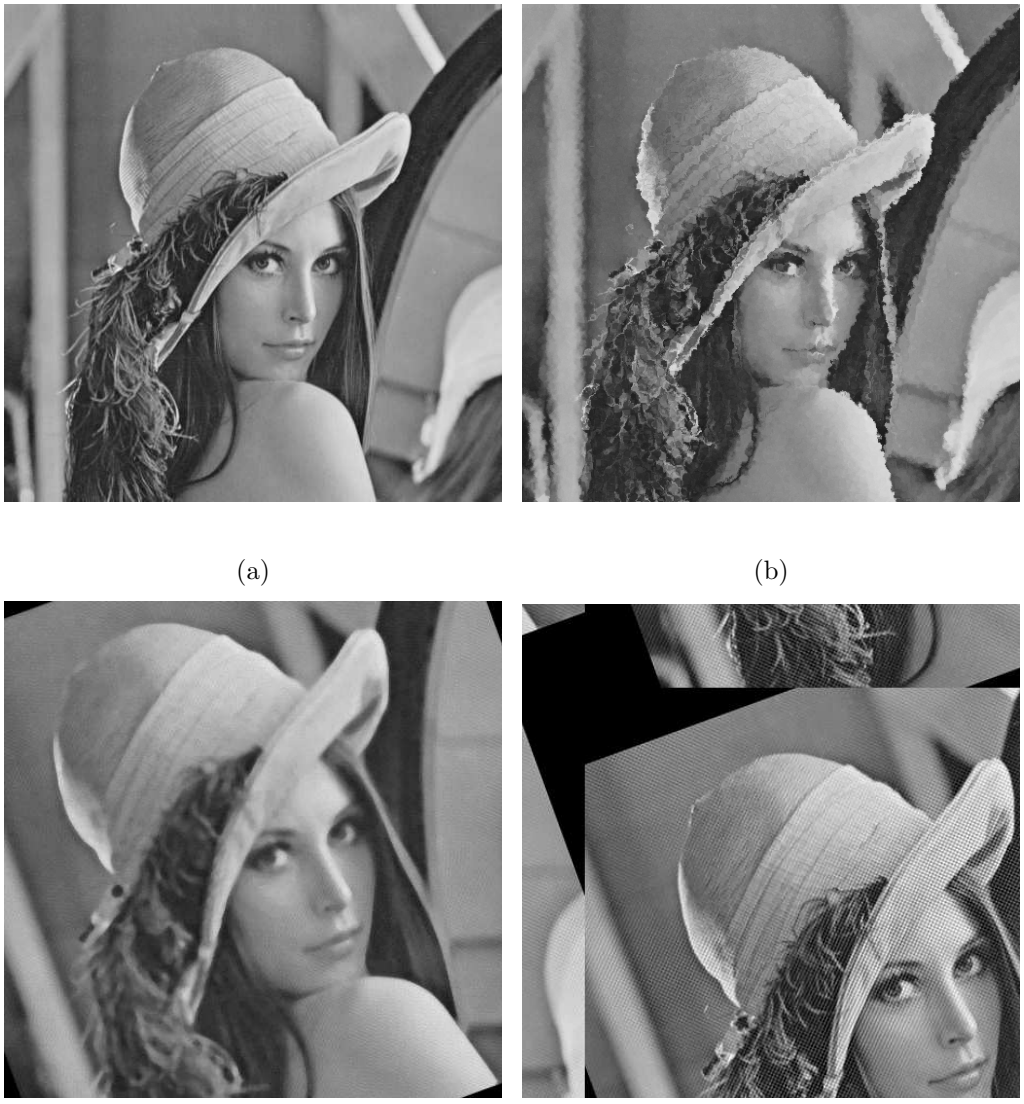
We present suitable key-point pruning methods so that even after considering a reduced number of key-points, the receiver is generally successful in identifying the same key-point locations as the embedder. Quantization index modulation based embedding and repeat accumulate code based error correction are used to obtain a good trade-off between hiding rate and error resilience. Our hiding scheme is shown to be robust against rotation angles within $(-45°, 45°)$ and scaling factors within (0.5,2).

The chapter is organized as follows. The challenges faced by data-hiding schemes that have to survive cropping and geometric transformations are described in Section 5.1. Discussion of various image watermarking methods robust to geometric transformations is presented in Section 5.2. The functionalities of the various modules involved in the encoder and decoder of our proposed hiding system are discussed in Section 5.3. The theory of computing the transformation matrix from two sets of corresponding peak locations is introduced in Section 5.4. In this section, we also discuss a suite of methods (tapered window based peak insertion, novel peak detection functions) by which the robustness and accuracy of estimation of the transformation parameters are significantly enhanced. In Section 5.5, we obtain more accurate geometric alignment in presence of JPEG compression by using prior assumptions about the form of the transformation matrix and by predicting the locations of the JPEG-induced peaks. Issues aris-

ing out of using different sized DFT grids for localizing the synchronization peaks, cropping the image and performing local transformations over small image regions (as opposed to global geometric transformations) are discussed in Section 5.6. In Section 5.7, we present a pruning method to return robust key-points. The experimental results for transformation matrix estimation, KP detection and embedded data recovery for various attacks are presented in Section 5.8. We summarize our contributions in Section 5.9 and also propose avenues for future research in the context of key-point based hiding.

5.1 The Challenges in Robust Hiding

While the design of robust watermarking methods to survive geometrical transformations and local attacks has been well studied in the literature, very little work has been done for the design of robust "data hiding" methods for surviving such attacks. *For a data hiding system*, the receiver's task is a blind decoding problem of recovering the embedded data without having a reference sequence for comparison. Here, we present a novel robust data hiding method which allows proper data recovery even after significant geometrical transformations, followed by image processing modifications, such as the ones shown in Fig. 5.1.



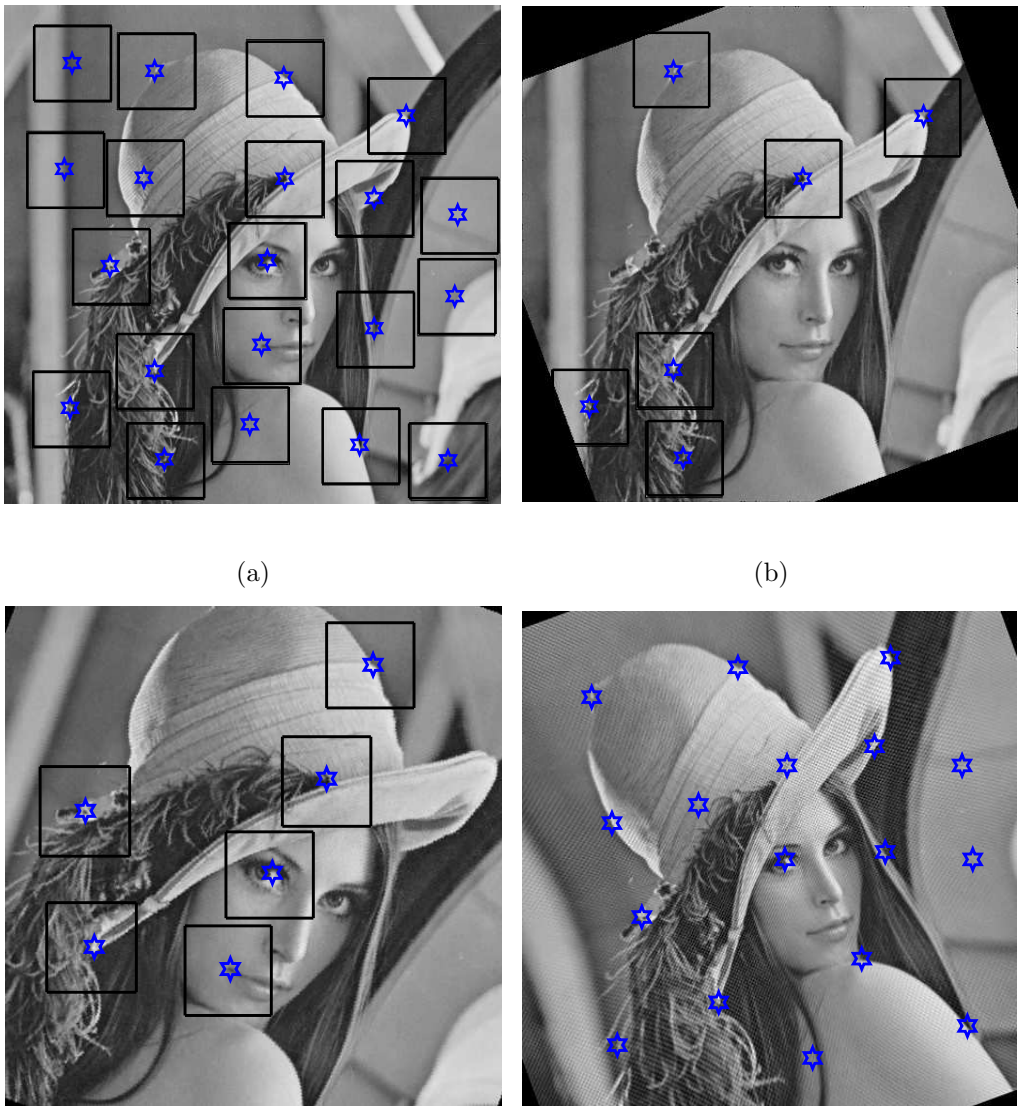
(a) (b)

(c) (d)

Figure 5.1: (a) original image, (b) image after global attack (ocean-ripple filter in Photoshop), (c) image after geometrical transformation (rotation and scale), cropping (local attack) and global attack on resultant image (dust filter in Photoshop), (d) same as (c) but dust filter is replaced by offset filter which involves translations along x and y-axes.

Key Challenges: State-of-the-art hiding systems use the entire image to embed the code bits. If an image is cropped, a part of the embedded sequence is lost and this may lead to decoding failure. One solution is to repeat the embedding in multiple regions of an image. However, this can create a synchronization problem. Without side information, how can the decoder know the embedding regions used at the encoder?

To ensure that the decoder is more likely to identify the same local regions as used by the encoder, an approach to detect salient regions for hiding is to use local regions centered around key points (KP). If the image attacks are assumed to be mild enough, the perceptual content is not changed significantly and hence, the same image regions are often identified as salient regions by both the encoder and the decoder. For example, several of the key-points in Fig. 5.2(a) are still retained after image attacks, as shown in Fig. 5.2(b) and 5.2(c). Such a scheme requires precise alignment of the embedding regions at the decoder in order for successful decoding of the embedded bits, as in Fig. 5.2(b). If the same KP are obtained but the hiding regions obtained at the decoder are not aligned with those used at the encoder, as in Fig. 5.2(c), the embedded data cannot be recovered. *To summarize, the key challenge at the decoder side is to identify the exact image region (at the original scale and orientation) at which the encoder has embedded the code bits.*



(a) (b)

(c) (d)

Figure 5.2: (a) original image with key-points (KP) marked, with the square boxes showing the embedding regions, (b) a geometrically transformed image is aligned back with the original grid - the KP that are common with (a) are shown, (c) image after geometrical transformation and cropping and it is not aligned with the original image grid - the KP that are common with the original are shown, but the hiding regions are not aligned with those used at the encoder, (d) image is created using "pinch" filter in Photoshop, and all KP are shown. The KP in (a)-(d) are obtained using Nobel-Forstner [32,77] detector followed by pruning, as discussed in Section 5.7.

A schematic of our proposed data hiding/decoding framework which embeds the code bits in salient regions chosen around key-points is shown in Fig. 5.3. Its three main components are the encoder (Fig. 5.3(a)), the channel (Fig. 5.3(b)) and the decoder (Fig. 5.3(c)). The encoder identifies key-points and embeds the code bits in image coefficients belonging to a fixed-sized region around each key-point. In the channel, we assume that the image is geometrically transformed using a global affine transform (2×2 matrix A), followed by image processing attacks such as cropping and JPEG compression. This transformation A needs to be estimated so that the received image can be suitably aligned. *If the received image is not aligned with the original grid* (as in Fig. 5.2(c)), the decoder cannot access the same image region as was used for embedding even after finding one/more common key-points and *the data cannot be recovered*. After proper alignment, the decoder identifies key-points and performs decoding in the local regions surrounding the key-points.

When the same set of code bits is redundantly hidden around each key-point, *among the multiple salient points selected by the encoder, the decoder needs to identify at least one point correctly*. Here, it is assumed that the error correction coding (ECC) framework used to encode the data bits has sufficient redundancy so that it can decode data from the transform domain image coefficients (which

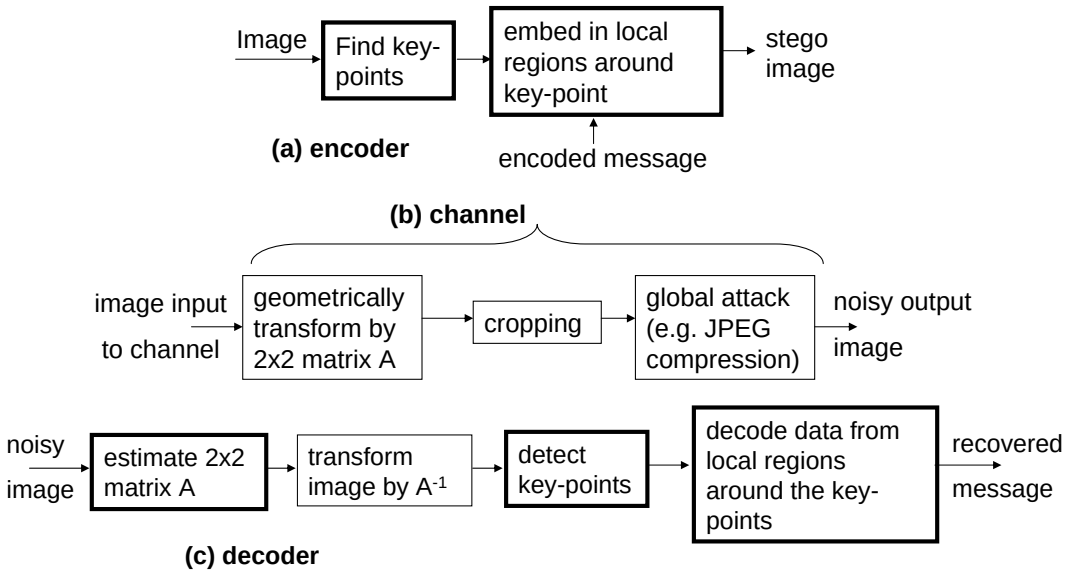


Figure 5.3: The end-to-end data hiding framework (a) **encoder**: the same encoded sequence is embedded in every local region around a key-point, (b) **channel**: the stego image can be subjected to geometrical transformations, followed by image processing attacks, (c) **decoder**: it aligns the received image with the original image grid and decodes data from the local regions around detected key-points. The boxes outlined in bold represent the main challenges/tasks at the encoder and the decoder.

are modified by noise attacks and interpolation induced errors) belonging to the local region around the correctly detected key-point.

Thus, our main contributions are: (i) robust and accurate estimation of the transformation matrix A, and (ii) robust pruning of KP such that the KP detected at the decoder match with those used at the encoder, in spite of choosing only a few key-points.

5.1.1 Specific Contributions

(1) *Robust and Accurate Geometrical Alignment:*

One robust approach to align images is to insert known patterns/peaks in the Discrete Fourier Transform (DFT) domain and detect them at the decoder. The inserted peaks are referred to as a "template" in [102]. However, such approaches run into problems while decoding after JPEG compression due to the standard 8×8 block-based induced periodicity. This periodicity introduces spurious peaks in the frequency domain which may cause improper alignment leading to decoder failure.

(i) *Novel peak detection functions:* We propose a tapered window based peak insertion (Section 5.4.2), along with novel ratio-based peakiness functions (Section 5.4.3), which results in higher peak detection accuracy. The aim is to increase the detectability of the template peaks without compromising on the perceptual quality of the images. The peak insertion and detection methods provide a trade-off between the template induced distortion and peak detectability.

To discard the JPEG induced spurious peaks, we propose the following methods:

(ii) *Using a parametric representation of the transformation matrix:* The model proposed in Section 5.5.1 assumes that either rotation or scaling or both, or only shearing, are the operations performed. For such operations, a relation exists be-

tween the elements in the transformation matrix. Detected peaks are discarded if the transformation matrix obtained using them does not follow such constraints.

(iii) *Predicting the location of JPEG induced peaks*: We experimentally predict the locations of the JPEG-induced peaks resulting from an inserted DFT peak (Section 5.5.2) and use that information to discard a JPEG-induced peak even when such a peak has a larger DFT magnitude than the actually inserted peak.

(2) *Efficient Pruning Method for Robust Key-point Selection:*

Using popular key-point detection algorithms, more than 1000 key-points are obtained (before pruning) for 512×512 images. The iterative decoding, performed for the local region around each key-point, is computationally involved and hence, a reduced number of KP needs to be returned at the decoder. Our proposed pruning method is based on spatial constraints and relative corner strengths. After pruning, about 20 key-points are returned per image. These key-points are robust enough so that about 20% of the pruned KP detected by the decoder match with those detected by the encoder (shown later in Section 5.7).

5.2 Related Work

A comprehensive survey of watermarking methods robust to a host of geometrical, local and global attacks is contained in [128]. We discuss the merits and

demerits of the following classes of methods for geometric transformation invariant image watermarking : (A) hiding in RST (Rotation, Scale, Translation) invariant domain, (B) Radon transform based, (C) template based, (D) content-based watermarking, and (E) segmentation based methods..

(A) Watermarking in domains invariant to geometric transformations

For watermarking in such domains (e.g. Fourier Mellin transform (FMT) [65, 91]) the geometric transformation need not be explicitly computed. The amplitude of FMT, computed on the Fourier transform magnitude, is RT-invariant. Thus, normalized (normalization accounts for scaling) correlation can be used for WM detection for RST attacks while using FMT. For FMT-based methods [91], it is difficult to adjust the WM strength for the hiding locations, as the spatial and frequency information which governs perceptual models is lost in this domain and the image quality also suffers due to interpolation errors. To obtain the image after WM addition in FM domain, inverse log-polar mapping (ILPM) is involved which distorts the WM, affecting the detector performance. This issue is addressed in [129], where the hiding locations are first determined in the LPM domain, then the corresponding locations in the Discrete Fourier Transform (DFT) magnitude domain are computed using the ILPM. The WM is embedded directly into the DFT magnitude domain and will not be distorted by the ILPM. The original

image is however needed for WM detection. Also, the FMT-based methods cannot survive aspect ratio change.

In [67], since the rotation and scaling result in shifts in the LPM (on the Fourier magnitude of an image) domain, the symmetrical phase only matching filter is used to compute the possible shift in the LPM domain. It uses the phase information of a matching template and the LPM of the WM image to compute the shift parameters. The template is a small part of the LPM domain of the original image - hence, it is not a completely blind detection. In [65], Lin et al. embed a context-related WM in a 1D projection domain derived from the FMT. The domain used is invariant to both translation and scaling. However, the rotation angle is computed through exhaustive search.

(B) Radon Transform based Method

The Radon transform involves projecting the 2D image along various directions - generalized Radon transforms proposed for robust watermarking [104] include the Radial Integration Transform (RIT) and the Circular Integration Transform (CIT). Scaling the original image also scales the CIT plot by the same factor. The shift in the RIT plot indicates the rotation angle. The RIT and CIT plots of a reference watermark, which suffers the same rotation and scale as the original image, are used to compute the rotation angle and the scale factor. The scale factor and orientation of this reference WM are known before the geometric transforma-

tion. The origin of the Radon transform computation is vital to the success of the detection. The modified Harris corner [47] is used to ensure that the decoder identifies the same point as the origin as the encoder. To reduce the dependence on obtaining the "single corner point" exactly, more feature points can be used and Radon transform can be computed on the individual points. This involves less data for the Radon transform computation and compromises the geometric parameter detection accuracy. This method cannot survive aspect ratio change and also does not survive cropping.

(C) Template Matching based Watermarking

We have used a template based approach for geometrical alignment in our hiding scheme. The template is a set of peaks inserted in the mid-frequency band in the DFT magnitude domain of the image [102]. By obtaining the position of the template peaks in the noisy WM image and comparing it with the initial position and orientation of the peaks, the decoder can estimate the geometric transformation. The mid-frequency band is chosen so as to maintain the image quality after template embedding. In [29], the inserted template consists of a periodical structure with a large number of repeats so as to get a large number of peaks in the auto-correlation function (ACF). A robust estimate of the affine transformation is obtained using penalized maximum likelihood or Hough transform based techniques.

By detecting the local maxima, the template peaks can be identified and hence, removed by an adversary [48, 80]. Also, sufficiently strong templates can substantially degrade the perceptual quality of the watermarked image.

The template based method, initially proposed to recover global transforms, is extended to account for local or nonlinear geometrical distortions in [116]. The watermark is repeatedly embedded into small blocks and local affine transforms are estimated per block. The embedding occurs at various wavelet decomposition levels. The periodic WM serves jointly as the embeddable message and the synchronization template. Without any geometrical transform, the message is decoded directly from the extracted WM. In presence of geometrical transforms, the peaks in the ACF of the estimated WM are used to recover the transformation parameters at a block level. In [60], a "structured" watermark (repeated several times in the spatial domain) is embedded with different offsets so as to obtain multiple peaks in the autocorrelation function of the estimated watermark. The positions of the extracted peaks with regard to the inserted peaks help in estimating the geometrical transformation.

(D) Content-based watermarking schemes

We discuss some embedding methods that embed the same information in multiple local regions selected based on the image content. The focus is on finding feature points, which can be independently obtained at the decoder even after

embedding and noise attacks, and embedding the watermark in local regions of pre-decided shape and size, which are completely specified once the feature points are known.

A popular scheme by Bas et al. [12] finds Harris corner points, and the WM is then embedded in local regions near the feature points using Delaunay triangulation [12], where if a vertex disappears, the tessellation is affected only for connected triangles. Wiener filtering based denoising is used to remove components pertaining to the original image during correlation based WM detection. To improve the tessellation based scheme in [12], Hu [50] proposed a scheme which generates four feature points using a robust intersection based point detector. Based on these points, some additional points are generated and triangulated in a key-dependent manner - hence, an attackers cannot find out the exact tessellation even if the feature points are known. Weinheimer et al. [122] use the improved Harris corner based feature points and embed the watermark in the Fourier domain of a normalized (*for a normalized image, the image origin is mapped to its centroid and the image is scaled based on the zero-order moments* [7, 79]) disk centered at a feature point. The normalized image is invariant to rotation and reduces synchronization errors, and is often used for key-point based watermarking [113, 114, 118].

For robust hiding in image regions identified around salient points, most methods have a small capacity due to the embedding area (size of local region) being

small - this can also result in a large variance for the correlation detector used for WM detection. Also, most of these methods (e.g. [12]) are *robust against only slight global transformations, e.g. small angles of rotation.* For image normalization related methods, *the precision of the normalization module is affected by the interpolation introduced by geometric transformations* [113]. Also, when the image normalization is performed for smaller sized image areas, it can lead to erroneous normalization. In our hiding framework, salient key-point (KP) detectors are used so that the encoder and decoder have access to the same local regions. Since the geometric alignment depends only on the accuracy of detection of template peaks, and not on tessellation or image normalization, our scheme is robust to significant geometric distortions.

(E) Segmentation based Methods

In [76], the largest salient regions obtained through segmentation are approximated by ellipsoids, and the bounding rectangle of each region is used for embedding. There is a local search for each embedding region parameter to ensure that the detector uses the same set of parameters as the encoder - *this parameter selection process is highly computationally intensive.* As the segmentation is image-content dependent, *image distortions severely affect the segmentation results* [61].

Description of our synchronization approach in the context of other methods: Spread spectrum is commonly used for geometrically robust watermarking and can also be used for low bit-rate data hiding. We employ quantization index modulation (QIM) [16] here (embedding 1 bit per coefficient) to increase the effective hiding rate. For proper data recovery after using QIM, the received image should be properly aligned and the key-point detector (at the encoder and the decoder) should return common points. We use the template based method [102] as it is resistant against significant geometric transformations and cropping, and we ensure robustness against JPEG compression (Section 5.5.2) and highly precise estimation of rotation and scale parameters (Section 5.5.1). A common criticism of the template based method is that the DFT domain peaks are detectable, thus indicating to an adversary that a synchronization template is contained in the image. However, it can be similarly argued that if the synchronization method is known beforehand, image processing attacks are always possible that can cause it to fail - e.g. for methods that rely on key-points, (e.g. Radon transform based, or content-based WM), slight modifications that distort the key-point locations disturb the synchronization procedure.

The discussion so far has been mainly on robust WM methods while robust data hiding methods have not been so well studied. To help understand how geometrically robust data hiding schemes operate, we include a brief description of

the data hiding method proposed by Wang and Ly [69,117]. Delaunay tessellation is done to partition the image into different triangles. The center coordinate vector of each triangle is represented as a binary sequence, which serves as the synchronization information (SI). The SI is embedded in the red channel for each triangle. Discrete Cosine Transform (DCT) domain embedding occurs in the blue channel of the image. While decoding, the SI is extracted using the fast correlation attack (FCA) [18], a common cryptanalysis method, which has powerful error-correcting analysis. Since it uses Delaunay triangulation as in [12], it is resistant against only small geometric changes. *Also, it cannot be used for data hiding in single-channel (grayscale) images.*

5.3 Robust Image Data Hiding Framework

We now explain the encoder and decoder blocks which were introduced in Section 5.1 (Fig. 5.3). Table 5.1 lists the various notations used in the following discussion.

(A) Encoder (Fig. 5.4)

(1) DFT-domain Peak Insertion for Geometric Alignment: The encoder embeds the template peaks in the DFT domain (F) at locations also known to the decoder. The DFT matrix after template insertion is F^{temp}, where $F^{temp} \xrightarrow[\text{DFT}]{\text{Inverse}} f^{temp}$. Var-

Table 5.1: Glossary of Notations

Notation	Definition
$f/f^{temp}/f^w/f'$	f: Original image/ f^{temp}: image after synchronization template addition/ f^w: watermarked image/ f': output image from the channel, after noise attacks on f^w
$F/F^{temp}/F^w/F'$	the DFT matrix corresponding to $f/f^{temp}/f^w/f'$, respectively
F_{mag}/F_{phase}	F_{mag}: the magnitude component of the DFT matrix F, F_{phase}: the phase component of F
$\{P_1, \cdots, P_4\}$	location of DFT domain peaks added to F to produce F^{temp} - inserted as self-synchronization template
$\{Q_1, \cdots, Q_4\}$	peak locations extracted from F'_{mag}, the DFT magnitude plot of f', $\{Q_i\}_{i=1}^4$ are integer valued points
$\{R_1, \cdots, R_4\}$	the ideal peak locations $\{P_1, \cdots, P_4\}$ get mapped to $\{R_1, \cdots, R_4\}$ (real numbers) in the DFT plot of the received image - if the geometric transformation is estimated "correctly enough", Q_i = rounded version of(R_i), $1 \leq i \leq 4$
X_{enc}	set of key-points obtained from f^{temp} - hiding occurs in $B \times B$ local regions around the key-points
X_{dec}	set of K_{dec} key-points obtained after geometrically aligning f', the noisy received image
f_A	image obtained after geometric transformation of f using $A \in \mathbb{R}^{2\times 2}$: $f_A\left(A[x_1\ x_2]^T\right) = f\left([x_1\ x_2]^T\right)$
A_{DFT}	if A is the geometric transformation between images f^1 and f^2, A_{DFT} is the transformation between DFT plots F^1 and F^2, i.e. $F^2_{A_{DFT}}(A_{DFT}[u_1\ u_2]^T) = F^1([u_1\ u_2]^T) \Longleftrightarrow f^2_A(A[x_1\ x_2]^T) = f^1([x_1\ x_2]^T)$
B	the size of a local region used for hiding is $B \times B$
δ_{th}	a threshold imposed on the corner strength for key-point selection while hiding
QF_h	design quality factor (QF) used for hiding
QF_a	output JPEG QF at which the noisy output image f' is advertised
λ	the first λ AC DCT coefficients obtained after zigzag scan are used for embedding for a 8×8 block

ious peak insertion methods are discussed in Section 5.4.2.

(2) Obtaining Local regions for Embedding after Key-point detection: Key-points (KP) are detected at the encoder using a suitable corner detector method and a reduced number of salient points are returned after appropriate pruning (Section 5.7). The embedding is repeated in $B \times B$ non-overlapping regions, each centered around a key-point.

(3) QIM-based Embedding in Local Regions: Embedding is done using quantization index modulation (QIM) [16] in non-overlapping 8×8 blocks in each selected $B \times B$ region. For embedding, we choose a certain low and mid-frequency band of the AC DCT coefficients computed per 8×8 block, as in [107]. The DCT coefficients are divided by a quantization matrix corresponding to the design JPEG quality factor used for hiding, QF_h, before being used for embedding. The embedding band consists of the first λ AC DCT coefficients, per 8×8 block, encountered during zigzag scan. The QIM-based **embedding rule** used is: a quantized AC DCT coefficient is converted to the nearest even/odd multiple of the quantization interval (which is unity, in our case), to embed 0/1, respectively. For perceptual transparency, quantized DCT coefficients in the range [-0.5,0.5] are mapped to zero and are regarded as *erasures* at the decoder [107].

(4) Repeat Accumulate (RA) Coding for Error Correction: The data bits to be embedded are encoded using the RA coding framework [25]. RA codes are used

because QIM-based embedding in the DCT domain generally involves a high erasure rate (the DCT histogram is generally peaky near zero and tapers off for higher magnitude coefficients) and RA codes have good robustness for high erasure channels [97, 107]. The hiding rate can be maximized by choosing the minimum redundancy factor (q_{opt}) that allows perfect data recovery for a given image and a specified attack channel.

(B) Image modifications in the channel: The watermarked image f^w is geometrically transformed using a 2×2 matrix A to obtain f^w_A, and then it is subjected to global/local attacks (Fig. 5.3(b)). It is finally advertised as a JPEG image f' where the quality factor used in the compression step is QF_a. Thus, the sequence of changes is $f^w \xrightarrow{\text{(transformed by A)}} f^w_A \xrightarrow{\text{(local/global attacks)}} f'$.

(C) Decoder (Fig. 5.5)

(1) Identifying DFT Peaks for Geometric Alignment: To compute the transformation matrix, we need to correctly identify the four peaks in the DFT magnitude plot of the received image. From the original peak locations $(P_1, \cdots, P_4)$ and the detected peak locations $(Q_1, \cdots, Q_4)$, an estimate of the DFT-domain transformation matrix ($\hat{A}_{DFT}$) is obtained, which is then used to estimate the pixel-domain transformation matrix ($\hat{A}$).

(2) Aligning the Received Image: The received image is transformed using $\hat{A}^{-1}$ to align it with the original image grid.

(3) Computing Key-points: The decoder finds key-points (X_{dec} is the set of detected KP) for the aligned received image using the same key-point detection and pruning method as the encoder.

(4) RA-decoding for Local Regions around each detected Key-point: For every key-point in X_{dec}, RA-decoding is performed on the extracted symbols corresponding to the quantized block-based DCT coefficients obtained from the local region surrounding the key-point. For turbo decoding, the soft decision values, called log-likelihood ratios (LLR), have to be properly initialized for all the embedding locations. Each DCT coefficient in the embedding band is mapped to a LLR value of 0 if its absolute value is less than 0.5 (corresponds to an erasure), else it is mapped to $\alpha/-\alpha$ (corresponds to 0/1 being embedded) depending on whether the coefficient rounds off to an even/odd integer. The scaling factor α reflects the decoder's confidence in the accuracy of the decoded bit values - it depends on the hiding parameters (QF_h, λ). For a given set of parameters, the optimal value of α (α_{opt}) ensures convergence of the RA decoding at the lowest q (compared to other α values) and allows the maximum data rate. Another issue is computing q for the RA codeword given the ternary sequence obtained at the decoder - it is computed by exploiting the structure of RA-encoded sequences as

discussed in [97].

(5) Termination Condition for Data Recovery: The termination condition (we do not consider the remaining key-points) used depends on whether it is a watermarking or data-hiding application. In watermarking, the decoder's goal is to test for the presence/absence of the WM - the remaining KP are skipped when the retrieved data corresponds to the embedded WM. For a data hiding example, we stop when the turbo decoding converges with a high LLR value averaged over the data-bits, as shown in [97]. *We have experimentally observed that for noisy channels, the RA-decoding can converge to the same sequence (which is not the desired/correct sequence) in two consecutive iterations but the average LLR is high only when when it converges to the desired sequence.*

The issues of deciding q_{opt} at the encoder (encoder module 4), recovering the q factor at the decoder and using the proper value of α (α_{opt}) (decoder modules 4 and 5) for LLR computation are valid for any scheme employing RA-based error correction and QIM based hiding. Here, we have focussed on the issues that are specific to "local KP based hiding" robust to geometrical attacks -

(i) the geometric transformation needs to be correctly and robustly estimated, and

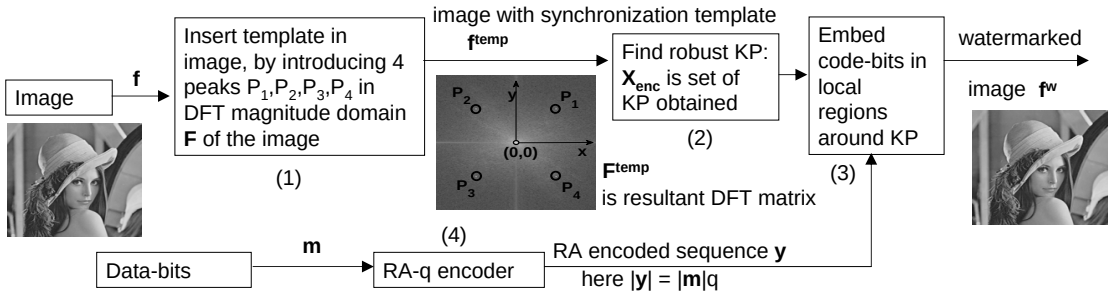


Figure 5.4: Block diagram at the encoder side - numbers (1)-(4) correspond to the encoder side modules.

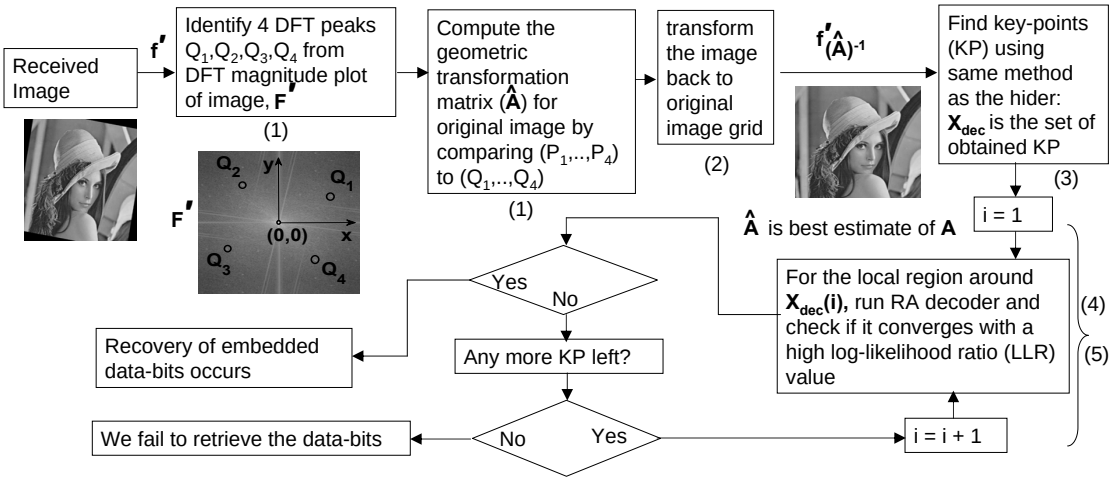


Figure 5.5: Block diagram at the decoder side - numbers (1)-(5) correspond to the decoder side modules.

(ii) there should be an efficient pruning algorithm for the KP detected at the encoder and the decoder so that the decoder can return one or more KP common with the encoder while considering only a small number of KP.

5.4 Peak based Alignment

We briefly explain how the transformation is estimated in the DFT magnitude domain and how it is used to estimate the pixel-domain transformation (Section 5.4.1). The magnitude of the inserted DFT peaks can be increased to enable better detection. However, increasing the peak strengths may introduce visible periodic striations. The perceptual distortion introduced after the template insertion is quantified through the Peak Signal-to-Noise Ratio (PSNR). *All the methods presented here are aimed at achieving this trade-off, i.e. obtaining more accurate estimation at the same PSNR.* Section 5.4.2 describes the peak insertion approaches while Section 5.4.3 describes the peak detection functions. We present the experimental results in Section 5.4.4.

5.4.1 Estimating Transformation Matrix

We show that a simple relationship exists between the spatial-domain transformation matrix and the corresponding matrix in the DFT magnitude domain - the relation is mentioned in [11, 28] and we include the proof here for completeness and ease of understanding. The transformation is computed w.r.t. the image center in the pixel domain and w.r.t. the center of the DFT grid in the DFT domain.

The synchronization template consists of 4 peaks inserted in the DFT magnitude domain of the $N_1 \times N_2$ input image - we refer to the image with the inserted template as u. After transforming the image using $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, we obtain u_A where the relation between the pixel locations in u and u_A is:

$$u([x_1 \ x_2]^T) = u_A(A[x_1 \ x_2]^T),$$

i.e. the point (x_1, x_2) in u gets mapped to $(a_{11}x_1 + a_{12}x_2, a_{21}x_1 + a_{22}x_2)$ in u_A.

Let U and U_A denote the 2D $N \times N$ DFT for u and u_A, respectively, where $N = \max(N_1, N_2)$.

$$U(k_1, k_2) = \sum_{x_1} \sum_{x_2} u(x_1, x_2) e^{-2j\pi k_1 x_1/N} e^{-2j\pi k_2 x_2/N}, \ 0 \leq k_1, k_2 < N$$

$$U_A(k_1, k_2) = \sum_{x_1} \sum_{x_2} u_A(x_1, x_2) e^{-2j\pi k_1 x_1/N} e^{-2j\pi k_2 x_2/N}, \ 0 \leq k_1, k_2 < N$$

The problem is to express A_{DFT} in terms of A, where $[k_1 \ k_2]^T$ in U gets mapped to $A_{DFT}[k_1 \ k_2]^T$ in U_A (notation wise, $U(k_1, k_2)$ is equivalent to $U([k_1 \ k_2]^T)$).

$$U([k_1 \ k_2]^T) = U_A(A_{DFT}[k_1 \ k_2]^T) \tag{5.1}$$

$$u(x_1, x_2) = u_A(a_{11}x_1 + a_{12}x_2, a_{21}x_1 + a_{22}x_2) \Rightarrow$$

$$u_A(x_1, x_2) = u((a_{22}x_1 - a_{12}x_2)/D, (-a_{21}x_1 + a_{11}x_2)/D), \text{ where } D = (a_{11}a_{22} - a_{12}a_{21})$$

$$U_A(k_1, k_2) = \sum_{x_1} \sum_{x_2} u((a_{22}x_1 - a_{12}x_2)/D, (-a_{21}x_1 + a_{11}x_2)/D\ ) e^{-2j\pi k_1 x_1/N} e^{-2j\pi k_2 x_2/N}$$

Replacing $y_1 = (a_{22}x_1 - a_{12}x_2)/D$ and $y_2 = (-a_{21}x_1 + a_{11}x_2)/D$, we get

$$U_A(k_1, k_2) = \sum_{y_1} \sum_{y_2} u(y_1, y_2) e^{-j2\pi y_1(a_{11}k_1 + a_{21}k_2)/N} e^{-j2\pi y_2(a_{12}k_1 + a_{22}k_2)/N} \Longrightarrow$$

$$U_A(k_1, k_2) = U(k_1a_{11} + k_2a_{21}, k_1a_{12} + k_2a_{22}) \Longrightarrow$$

$$U(k_1, k_2) = U_A((a_{22}k_1 - a_{21}k_2)/D, (-a_{12}k_1 + a_{11}k_2)/D) = U_A((A^{-1})^T [k_1\ k_2]^T) \tag{5.2}$$

Thus, we see that $A_{DFT} = (A^{-1})^T$, from (5.1) and (5.2)

Thus, if the transformation matrix is known in the DFT domain (A_{DFT}), it is simple to compute the transformation matrix in the pixel domain (A), for any $A \in \mathbb{R}^{2\times 2}$. The received image f', which has been geometrically transformed by A, can be inverse-transformed so that it ($f'_{A^{-1}}$, as in Fig. 5.5) corresponds with the original image grid (f or f^{temp} or f^w, as in Fig. 5.4), based on which hiding was done.

There are 4 parameters to be estimated in A. Each point in the DFT grid is 2-D and hence, 2 points are needed to solve for the 4 unknowns. By conjugate

symmetry, once we insert peaks in the first and second quadrants, the corresponding peaks in the third and fourth quadrants get fixed.

Encoder Side: The $N \times N$ DFT of the original image (f) produces the matrix F, and four peaks $\{P_i\}_{i=1}^{4}$ are then introduced, one at the center of each quadrant.

Decoder side: The DFT is computed for the received image f' and the new peak locations ($\{Q_i\}_{i=1}^{4}$) are identified. Let $G = \hat{A}_{DFT} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$ be the estimated transformation matrix in DFT domain. Here, $P_i = (p_{ix},\ p_{iy})$ and $Q_i = (q_{ix},\ q_{iy})$, are the peak locations in the original (F_{mag}^{temp}) and new DFT magnitude (F'_{mag}) grids, respectively, for the i^{th} quadrant. The elements in G are computed using (5.3).

$$\begin{bmatrix} g_{11} \\ g_{12} \end{bmatrix} = \begin{bmatrix} p_{1x} & p_{1y} \\ p_{2x} & p_{2y} \end{bmatrix}^{-1} \begin{bmatrix} q_{1x} \\ q_{2x} \end{bmatrix} \text{ and } \begin{bmatrix} g_{21} \\ g_{22} \end{bmatrix} = \begin{bmatrix} p_{1x} & p_{1y} \\ p_{2x} & p_{2y} \end{bmatrix}^{-1} \begin{bmatrix} q_{1y} \\ q_{2y} \end{bmatrix} \tag{5.3}$$

Translation along the axes affects only the phase but not the DFT magnitude. Hence, it does not affect the accuracy of the geometric transformation estimation. What is the precision to which we can estimate the elements in A? Generally, the ideal peak locations R_i in the DFT grid are not integers while the best matched point Q_i in the DFT grid is constrained to be an integer. A possible solution is to increase the size of the DFT grid. A bigger sized DFT involves a larger search space but also increases the accuracy. This trade-off is elaborated upon

in Section 5.6.1. Most of our input images are 512×512 images before geometric transformation. After rotation and scaling, we crop the images such that their individual dimension does not generally exceed 512. Hence, we use 512×512 as the DFT size. When the scaling factor is less than 0.5 or the rotation angle exceeds 45° in magnitude, the peak is shifted to a different quadrant, and hence the correspondence between the original peak P_i and the new peak R_i is lost. We assume such cases do not occur in our experiments.

Illustration of Template Based Embedding Distortion: For more accurate parameter estimation, one can increase the magnitude of the DFT peaks, so that they can be identified more precisely. The trade-off here is that increase in the peak strength may introduce visible periodic striations in the image (Fig. 5.6).

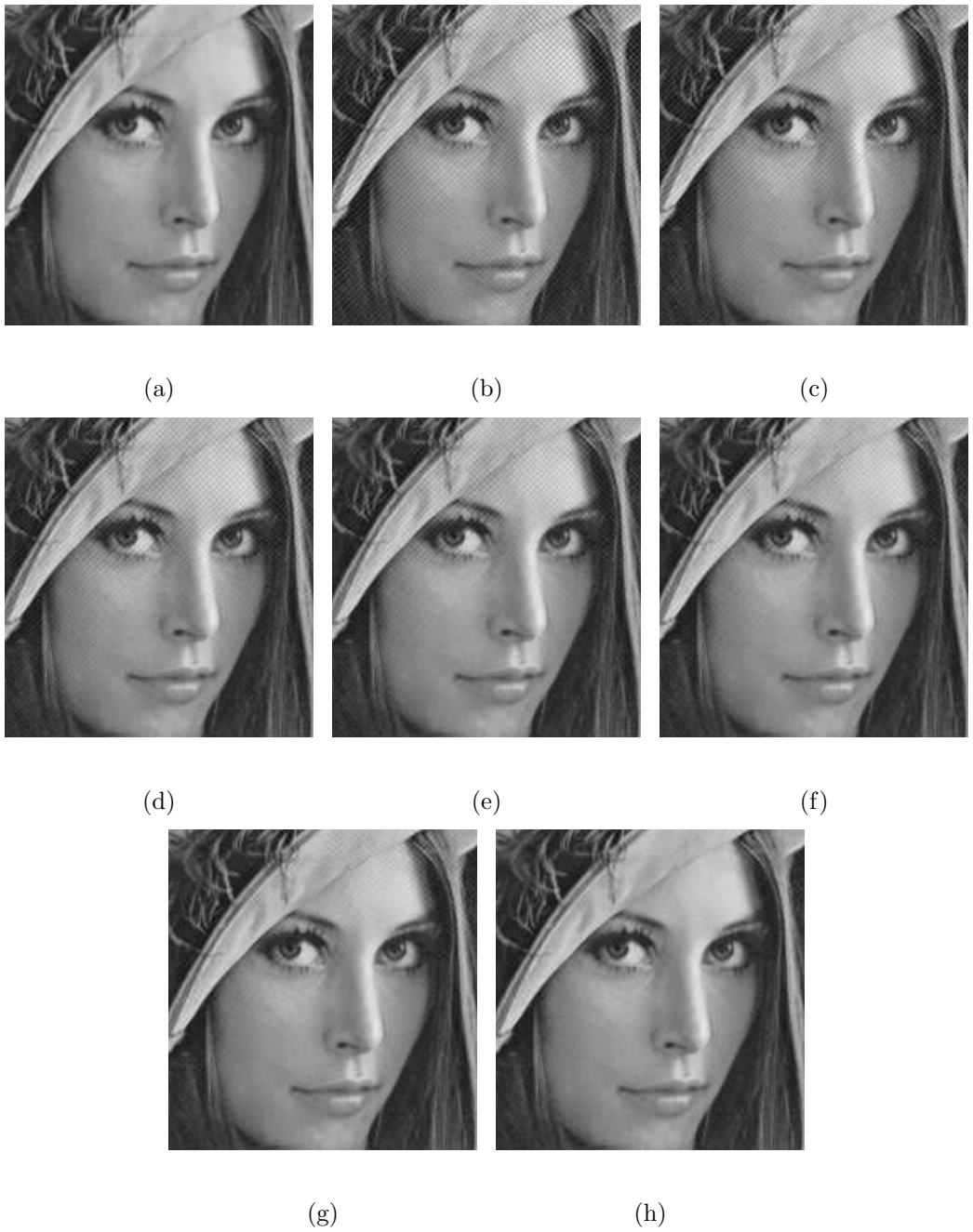
5.4.2 Peak Insertion Methods

We consider the following peak insertion methods:

(i) *single-point based method* - a peak inserted at every quadrant center serves as the reference peak,

(i) *window-based peak* - there is a tapered window of peaks with the window center corresponding to the highest valued peak,

(ii) *line-based method* - the inserted peaks belong to a certain pattern, e.g. multiple



(a) (b) (c)

(d) (e) (f)

(g) (h)

Figure 5.6: It is seen that the striations become more visible as the PSNR decreases: (a) original image; the other images are obtained after template addition in the DFT domain at different PSNRs (dB) (b) 31.34 (c) 33.46 (d) 36.17 (e) 37.91 (f) 39.68 (g) 41.59 (h) 45.85. The periodic patterns are very clearly evident in the lower PSNR images and the visibility of the periodic patterns progressively decreases with increased PSNR.

peaks inserted in a quadrant lie in a straight line passing through the origin, where the central peak is the quadrant center.

In [107], we have shown that moderate levels of JPEG compression (e.g. $QF_a = 75$) still result in practical hiding rates, after using suitable error correction codes. Here, the geometrical alignment needs to be ensured first for data recovery. JPEG compression introduces a periodicity of 8 in the image DFT as it works on 8×8 blocks. This results in spurious DFT-domain peaks for each inserted peak. The challenge is to detect the actual peaks in spite of the presence of these JPEG induced peaks.

Tapering Peak Strength for Peak Insertion: Window-based peak insertion with tapered peak strengths allows us to use peak detection functions that exploit the tapered peak property. Such peak insertion and detection methods result in more accurate peak detection for the same template induced distortion, as shown later in Tables 5.2, 5.3 and 5.4.

The tapered peak insertion method is explained considering a $(2\Delta+1)\times(2\Delta+1)$ window around (p_{1x}, p_{1y}) in the first quadrant. The local maximum $S(p_{1x}, p_{1y})$ (5.4) is computed over the window in the DFT magnitude domain. The points in the window are then weighted using tapering weights to obtain F^{temp} from F (5.5). The weights assigned to points at a "maximum-norm" distance (maximum of the distances along the two dimensions) of $0, 1, \cdots, \Delta$ from the window center

are $v_1, v_2, \cdots, v_{\Delta+1}$, respectively. This process is repeated for the other quadrants.

$$S(p_{1x}, p_{1y}) = \max_{(x,y):\max\{|x-p_{1x}|,|y-p_{1y}|\}\leq\Delta} F_{mag}(x, y) \tag{5.4}$$

$$F_{mag}^{temp}(x, y) = S(p_{1x}, p_{1y}).v_i, \forall(x, y) \in C_i, \tag{5.5}$$

where $C_i = \{(x, y) :$ maximum norm distance of (x, y) from $(p_{1x}, p_{1y}) = (i - 1)\}$

Tapered Peaks for Line-based Method: Here, all the peaks in the same quadrant lie in a straight line passing through the origin. After geometric transformation, the DFT peaks will still lie along a straight line through the origin. If we have (2Δ+1) peaks in a quadrant, the quadrant center has a peak strength of v_1 and the peaks at a distance of i from the center are weighted by v_{i+1}, and $v_1 \geq v_2 \geq v_3 \cdots \geq v_{\Delta+1}$. At the decoder, a certain number of top peaks are determined per quadrant and the dominant peak direction is identified. For the i^{th} quadrant, the average location of the peaks along the dominant direction for this quadrant is used as Q_i.

5.4.3 Ratio based Peakiness Function

We present two peakiness functions, shown below in (5.6) and (5.7). The first function $\mathbf{S}_\Delta(F'_{mag}, x, y)$ (5.6) at a point (x, y) in the DFT grid equals the sum of DFT magnitudes in a $(2\Delta + 1) \times (2\Delta + 1)$ window around the point. The second function $\mathbf{T}_\Delta(F'_{mag}, x, y)$ (5.7) accentuates the peakiness by considering the ratio

between the sum of points in the $(2\Delta{+}1) \times (2\Delta{+}1)$ window and the sum of points at a width of one unit outside the window. We find the top peak in the first two quadrants using our proposed peakiness function, and the peaks in the other quadrants get fixed by conjugate symmetry.

$$\mathbf{S}_{\Delta}(F'_{mag}, x, y) = \sum_{(x',y'):\max\{|x'-x|,|y'-y|\}\leq\Delta} F'_{mag}(x', y') \tag{5.6}$$

$$\mathbf{T}_{\Delta}(F'_{mag}, x, y) = \frac{\mathbf{S}_{\Delta}(F'_{mag}, x, y)}{\mathbf{S}_{\Delta+1}(F'_{mag}, x, y) - \mathbf{S}_{\Delta}(F'_{mag}, x, y)} \tag{5.7}$$

We refer to the two peak detection methods as **Method 1 (M1)** and **Method 2 (M2)**, where the peak points are determined based on $\mathbf{S}_{\Delta}(F'_{mag}, \cdot, \cdot)$ and $\mathbf{T}_{\Delta}(F'_{mag}, \cdot, \cdot)$, respectively. Let $(\hat{x}_j, \hat{y}_j)$ denote the peak location for the j^{th} quadrant. The detected peak locations $\{(\hat{x}_j, \hat{y}_j)\}_{j=1}^{4}$ are regarded as $\{Q_1, \cdots, Q_4\}$ based on which we compute the transformation matrix using (5.2) and (5.3).

For the peakiness features $\mathbf{S}_{\Delta}(F'_{mag}, \cdot, \cdot)$ and $\mathbf{T}_{\Delta}(F'_{mag}, \cdot, \cdot)$, we do not consider very low frequencies and points in the DFT grid, one of whose dimensions is close to zero. The image DFT is expected to have very high values in the very low frequency zone and points in such zones may show up as spurious peaks. This explains the black lines in the $\mathbf{T}_{\Delta}$ plots used later in Figs. 5.7, 5.8 and 5.17.

5.4.4 Experimental Comparison of Various Methods

Experimental Setup: We conduct all the experiments for estimating the geometric transformation matrix A over a set of 250 grayscale images [1]. The fraction of images for which A is accurately estimated is denoted by p_A. How do we define an estimate to be "accurate enough"? There is a precision error between an ideal peak location R_i (has a fractional part) and the actual detected location Q_i (integer). To effectively compare different peak insertion and detection methods, we need to use the same definition of p_A for all the methods. The transformation matrix $A = \begin{bmatrix} 1.10 & 0.20 \\ -0.15 & 0.90 \end{bmatrix}$ along with 20% cropping for both dimensions, is used for all the tables in this section. At the decoder side, we convert the elements of the estimated matrix $\hat{A}$ to nearest multiples of 0.05 before comparing them element-by-element with A. Considering numbers lower (or higher) than 0.05 will result in lower (or higher) p_A but the relative performance between the different methods will still hold. *A peak insertion method is better than another if it leads to a higher p_A, at the same (or higher) PSNR.*

At the encoder side, we wish to ensure that for every image, the inserted template peak is just strong enough to enable detection over the JPEG compression introduced peaks. We assume that the worst-case JPEG compression is known;

[1] downloaded from http://www-2.cs.cmu.edu/yke/retrieval

i.e. if we adjust the template strength assuming QF_a of 75, it will survive JPEG attacks at QF_a equal or exceeding 75. We start with relatively lower values of $\{v_i\}_{i=1}^{\Delta+1}$ (5.5) and gradually increase them by a factor of 1.2, considering a maximum of 10 steps. Here, the encoder simulates the transformation and JPEG compression based channel and estimates A at each step. The process is terminated when A is reliably estimated over a range of transformations for template strength of $\{v_i.(1.2)^s\}_{i=1}^{\Delta+1}$, where s is the minimum value in $\{1, 2, \cdots, 10\}$ that ensures correct peak detection.

The value of s is seen to be fairly consistent over a wide range of transformation angles (the DFT peaks can be affected by the smoothing introduced in the interpolation that is associated with geometric transformations) for a given image but varies with QF_a. Regarding scaling, the smoothing introduced is more severe when the image is down-scaled leading to lower p_A. We defer presenting the performance with different scale factors till Section 5.5.2 as the focus here is "relative comparison" of the peak insertion and detection methods (down-scaling affects these different methods similarly). If A is not correctly estimated after 10 steps, i.e using peak strengths of $\{v_i.(1.2)^{10}\}_{i=1}^{\Delta+1}$, we consider it as a failure in our p_A computation. The PSNR reported in Tables 5.2-5.4 is over those images for which A is correctly computed. For hiding, we use a design quality factor $QF_h = 60$ (used for all other experiments) and the hiding band size $\lambda = 5$.

Table 5.2: We compare methods M1 and M2 for estimating A correctly. **M2 performs better than M1**.

Method	[14 10 4 2] $QF_a = 85$	[14 10 4 2] $QF_a = 75$	[14 10 4 2] $QF_a = 65$	[14 10 4 2] $QF_a = 50$	[12 12] $QF_a = 75$
p_A (M1)	0.876	0.836	0.732	0.700	0.675
p_A **(M2)**	**0.940**	**0.880**	**0.800**	**0.736**	**0.745**
PSNR in dB (M1)	37.59	37.24	35.12	34.77	36.75
PSNR in dB **(M2)**	**42.25**	**40.50**	**39.12**	**37.50**	**38.29**

Table 5.3: We compute p_A and average PSNR for **various window sizes** Δ and **varying window peak strengths** $[v_1 \cdots v_{\Delta+1}]$ - these denote the starting peak strengths. QF_a of 75 is used.

Windowed peak insertion with tapered peak strengths (general)							
Δ	$[v_1 \cdots v_{\Delta+1}]$	PSNR (dB)	p_A	Δ	$[v_1 \cdots v_{\Delta+1}]$	PSNR (dB)	p_A
0	[24]	41.60	0.768	0	[30]	39.51	0.796
0	[40]	38.07	0.856	1	[12 12]	38.29	0.745
1	[20 20]	35.96	0.784	2	[6.32 6.32 6.32]	39.53	0.700
3	**[14 10 4 2]**	**40.50**	**0.880**	**3**	**[16 10 4 2]**	**40.41**	**0.876**

Comparison of Peakiness Functions: In Table 5.2, we show that $\mathbf{T}_\Delta$ (5.7) is a more effective indicator of peakiness than $\mathbf{S}_\Delta$ (5.6). Hence, M2, which uses $\mathbf{T}_\Delta$, is subsequently used to estimate A (Tables 5.3 and 5.4).

Choosing Window Size and Tapered Weights: From Tables 5.3 and 5.4, we observe that for the same PSNR, a larger window size (Δ) with tapered weights results in a higher p_A. E.g. using [14 10 4 2] results in a higher p_A than just a single peak, v_1 of 40. Also, [14 10 4 2] results in a higher p_A as compared to using non-tapered peak distributions such as [20 20] or [6.32 6.32 6.32].

From Tables 5.3 and 5.4, we also observe that the "line-based peak allocation" performs better than having a window of peaks around a single point. For the

Table 5.4: We repeat the same experiments as shown in Table 5.3 - the only difference being that we use the line-based method for peak insertion instead of the window-based method.

Tapered peaks (**line-based method**)			
Δ	$[v_1 \cdots v_{\Delta+1}]$	PSNR (dB)	p_A
1	[12 12]	41.02	0.805
2	[6.32 6.32 6.32]	41.60	0.810
3	**[16 10 4 2]**	**41.62**	**0.905**
3	**[14 10 4 2]**	**41.72**	**0.900**

line-based scheme, the fact that the top peaks (for a single quadrant) should lie in a straight line passing through the origin helps in discarding spurious peaks. In Section 5.5.1, we present a parametric approach *where if we assume some prior relation among the elements in A, some noisy candidates among the various detected peaks can be discarded.* The parametric approach needs point-based peaks as input instead of multiple peaks along a straight line. The parametric approach (Table 5.5) is seen to outperform both the non-parametric window-based and line-based methods.

5.5 Accurate Parameter Estimation

Here, we use two assumptions to refine our choice of the candidate DFT-domain peaks.

(i) If the assumed transformation involves rotation and/or scale (in any sequence) or shearing, a relationship exists between the elements in A. The noisy peak lo-

cations can be discarded if the transformation matrix obtained using them does not satisfy the assumed relationship. This is a reasonable assumption because the common image transformations are rotation, scale, crop (cropping does not create new peaks but blurs the DFT plot as discussed in Section 5.6.2), translation (does not affect the DFT magnitude plot), and shear (we assume linear shear). This is discussed in Section 5.5.1.

(ii) Since JPEG compression creates most of the spurious peaks, *if the locations of JPEG-induced peaks can be accurately predicted relative to the actual peak locations*, we can use that prediction model to discard potential JPEG peaks, as explained in Section 5.5.2.

5.5.1 Using Prior Assumption about the Geometric Transformation

Suppose, the image has undergone rotation by θ (counter-clockwise), and scaling of s_x and s_y along the x (columns) and y-axes (rows), respectively. Depending on the sequence in which rotation and scaling occur, the DFT domain transformation matrix A_{DFT} can be expressed as:

rotation followed by scaling:

$$A_{DFT} = \begin{bmatrix} \frac{1}{s_x} & 0 \\ 0 & \frac{1}{s_y} \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} \frac{1}{s_x}\cos(\theta) & -\frac{1}{s_x}\sin(\theta) \\ \frac{1}{s_y}\sin(\theta) & \frac{1}{s_y}\cos(\theta) \end{bmatrix},$$

scaling followed by rotation:

$$A_{DFT} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \frac{1}{s_x} & 0 \\ 0 & \frac{1}{s_y} \end{bmatrix} = \begin{bmatrix} \frac{1}{s_x}\cos(\theta) & -\frac{1}{s_y}\sin(\theta) \\ \frac{1}{s_x}\sin(\theta) & \frac{1}{s_y}\cos(\theta) \end{bmatrix}.$$

Let the rotation angle θ, and scale factors s_x and s_y be quantized into N_θ, N_{s_x} and N_{s_y} levels. We construct two 3D histograms H_{RS} (for rotation, followed by scale) and H_{SR} (for scaling, followed by rotation), where each histogram has $(N_\theta.N_{s_x}.N_{s_y})$ bins. E.g. for $\theta \in [-45°, 45°]$ and an angular resolution δ_θ of 1°, $N_\theta = (2\times45+1) = 91$. Similarly, for s_x(or s_y) $\in [0.5, 2.0]$ and using a resolution δ_{s_x} (or δ_{s_y}) of 0.05, N_{s_x} (or N_{s_y}) equals $(\frac{2-0.5}{0.05}+1) = 31$.

Rotation and Scale Estimation: Both the peak detection methods M1 and M2 encounter errors when the topmost peak is not the actual peak. *Here, we select multiple peaks per quadrant, and then the noisy peaks which do not result in geometrically consistent transformations are discarded.* Using either M1 or M2, let us obtain μ^2 sets of 4 peaks (top μ peaks per quadrant). For the i^{th} set, we compute $\hat{A}_{i,DFT}$, the estimate of the corresponding DFT-domain transformation matrix. If $\frac{\hat{A}_{i,DFT}(2,1)}{\hat{A}_{i,DFT}(1,1)} \approx -\frac{\hat{A}_{i,DFT}(1,2)}{\hat{A}_{i,DFT}(2,2)}$, it is an example of scaling followed by rotation. For a given $\hat{A}_{i,DFT}$, we compute θ_c (angle estimate), $s_{x,c}$ and $s_{y,c}$ (scale

estimates) using (5.8). The histogram bin corresponding to the angle index (i_θ) and scale indices (i_{s_x} and i_{s_y}) are updated as follows, given $\hat{A}_{i,DFT}$:

$$\theta_c = \tan^{-1}\left(\frac{\hat{A}_{i,DFT}(2,1)}{\hat{A}_{i,DFT}(1,1)}\right),\ s_{x,c} = \frac{\cos(\theta_c)}{\hat{A}_{i,DFT}(1,1)},\ s_{y,c} = \frac{\cos(\theta_c)}{\hat{A}_{i,DFT}(2,2)}, \tag{5.8}$$

$$i_\theta = \frac{\theta_c - (-45^\circ)}{\delta_\theta},\ i_{s_x} = \frac{s_{x,c} - 0.5}{\delta_{s_x}},\ i_{s_y} = \frac{s_{y,c} - 0.5}{\delta_{s_y}},\ \text{(all indices are rounded)} \tag{5.9}$$

$$H_{SR}(i_\theta, i_{s_x}, i_{s_y}) = H_{SR}(i_\theta, i_{s_x}, i_{s_y}) + 1,\ \text{where all histogram bins are initialized to 0.} \tag{5.10}$$

The process is repeated for all 4-tuples of peaks ($1 \leq i \leq \mu^2$). Similarly, when $-\frac{\hat{A}_{i,DFT}(1,2)}{\hat{A}_{i,DFT}(1,1)} \approx \frac{\hat{A}_{i,DFT}(2,1)}{\hat{A}_{i,DFT}(2,2)}$, it is an example of rotation followed by scaling. The corresponding angle and scale indices are computed and the H_{RS} histogram is updated. *Since we use window-based peaks, two or more sets of 4-tuples can get mapped to the same 3D bin.* We expect the 3D bin corresponding to the actual transformation to have the maximum number of 4-tuples that get mapped to it. To decide whether the transformation consists of rotation followed by scaling, or vice versa, we consider the maximum bin-count in H_{SR} and H_{RS}, computed over all μ^2 4-tuples.

$$SR_{max} = \max_{i_\theta, i_{s_x}, i_{s_y}} H_{SR}(i_\theta, i_{s_x}, i_{s_y}),\ RS_{max} = \max_{i_\theta, i_{s_x}, i_{s_y}} H_{RS}(i_\theta, i_{s_x}, i_{s_y}),$$

$$\text{If } SR_{max} > RS_{max},\ (i^*_\theta, i^*_{s_x}, i^*_{s_y}) = \arg\max_{i_\theta, i_{s_x}, i_{s_y}} H_{SR}(i_\theta, i_{s_x}, i_{s_y}),$$

$$\text{else } (i^*_\theta, i^*_{s_x}, i^*_{s_y}) = \arg\max_{i_\theta, i_{s_x}, i_{s_y}} H_{RS}(i_\theta, i_{s_x}, i_{s_y})$$

where the final indices chosen for θ, s_x and s_y are denoted by i^*_θ, $i^*_{s_x}$ and $i^*_{s_y}$, respectively. This discussion also covers the cases where only rotation, or only scaling, is performed.

Shear Factor Estimation: Among other 2×2 linear transformations, we consider shearing. For a shear factor of α (along x-axis), if $A = \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix}$, then the DFT domain matrix $A_{DFT} = \begin{bmatrix} 1 & 0 \\ -\alpha & 1 \end{bmatrix}$. For shearing along both axes (shear factor of α_1 and α_2 along x and y axes), the corresponding matrices are $A = \begin{bmatrix} 1 & \alpha_1 \\ \alpha_2 & 1 \end{bmatrix}$ and $A_{DFT} = \frac{1}{(1-\alpha_1.\alpha_2)} \begin{bmatrix} 1 & -\alpha_2 \\ -\alpha_1 & 1 \end{bmatrix}$. An $\hat{A}_{i,DFT}$ matrix that is an example of shearing is of the form: $\left(\hat{A}_{i,DFT}(1,1) - \frac{\hat{A}_{i,DFT}(2,1)\hat{A}_{i,DFT}(1,2)}{\hat{A}_{i,DFT}(2,2)} \right) \approx 1$. The shear parameters can be estimated using a 1D or 2D histogram based approach (depending on whether the model assumes shear along 1 or 2 dimensions).

In Table 5.5, we compare the accuracy using the model-based approach ($p_{A,m}$) with that obtained using the general "window with tapered peaks" based approach ($p_{A,g}$) and the line based approach ($p_{A,\ell}$) for varying QF_a. The transformation involved is a rotation (θ) of 20° followed by scaling of 1.1 (s_x) and 1.1 (s_y) along x and y-axes, alongwith 20% cropping along both dimensions (unless otherwise stated, this same transformation is used for other experimental results). We use a

Table 5.5: The geometric transformation parameter estimation results are presented for 3 different methods, for varying QF_a - here, $\mu = 10$, $\delta_\theta = 0.5°$ and $\delta_s = 0.05$. PSNR values are reported in dB.

QF_a used	general		line		model	
	$p_{A,g}$	PSNR	$p_{A,\ell}$	PSNR	$p_{A,m}$	PSNR
85	0.94	42.25	0.95	44.98	**0.99**	**45.65**
75	0.88	40.50	0.90	41.72	**0.94**	**44.25**
50	0.73	37.50	0.74	38.01	**0.82**	**41.48**
25	0.65	33.10	0.69	33.88	**0.78**	**35.55**

resolution of 0.5° and 0.05 for the angle and scale parameters. At the decoder side, we assume that the computed rotation angle and scale factors are multiples of 0.5° and 0.05, respectively, and the angle and scale values thus computed have to equal 20° and 1.1, respectively, for it to be considered a success in p_A computation. We use peak strengths of [14 10 4 2] while incrementing them by a factor of 1.2 with a maximum of 10 steps.

It is seen that in general, $p_{A,m} > p_{A,\ell} > p_{A,g}$, while comparing the methods at similar PSNR values. The performance deteriorates with lower QF_a as it smoothes the DFT matrix more severely, thus often suppressing the actual peaks. *Thus, the model-based method helps only when there are multiple candidate peaks (including the correct ones) and the spurious peaks do not satisfy the transformation model.*

5.5.2 Using JPEG Peak Location Property

We make the following observation *about the likely position of the JPEG induced peaks relative to an inserted peak location*, and the observation is also experimentally validated. If (x, y) corresponds to the geometrically transformed location of the inserted DFT peak, then the JPEG induced peak locations will be at $(x \pm 64k, y)$ and $(x, y \pm 64\ell),\ k, \ell \in \mathbb{Z}$ considering a 512×512 grid. Considering the 30 topmost peaks per quadrant, and comparing the peak locations with the actual location of the inserted DFT peak for geometrically transformed images, we found that 80% of the detected peaks were "JPEG-like neighbors" of the actual peaks. Visual examples of how the JPEG-induced peaks are spatially related with the location of the original DFT peaks are presented in Figs. 5.7 and 5.8.

We describe the steps involved in using the "JPEG peak location property" to discard spurious peaks. We first identify the rows, and then the columns, which have a larger number of peaks at multiples of 64 units apart. We consider the peak locations defined by the selected rows and columns and the higher magnitude points, which also satisfy the geometric transformation model (as in Section 5.5.1), are selected as likely candidates.

Step 1: We compute γ peaks in $\mathbf{T}_{\Delta}(F'_{mag}, \cdot, \cdot)$ for the first two quadrants. Let the peak locations detected for the j^{th} quadrant be $\{(\rho_{j,i,x}, \rho_{j,i,y})\}_{i=1}^{\gamma}$.

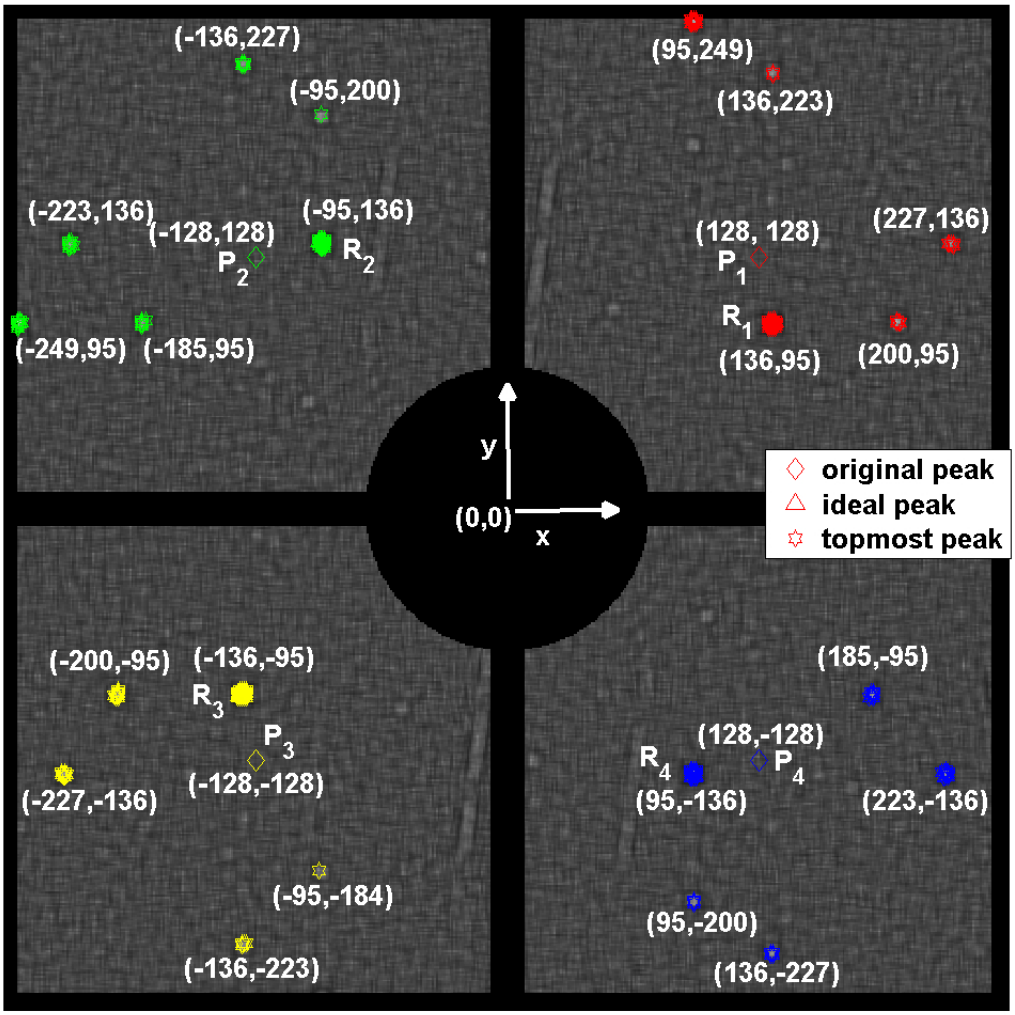


Figure 5.7: The figure shows the 512×512 $\mathbf{T}_\Delta$ plot for a transformation of $\{\theta = 10°, s_x = s_y = 1.1\}$, along with 20% cropping, and QF_a=75. P_i, the original peak location, is shifted to R_i after the geometric transformation. The 20 topmost peaks are shown per quadrant. Due to the window based peak insertion, many peak locations are clustered together; hence we see fewer peaks per quadrant. We observe that JPEG-induced peaks are generally separated at multiples of 64 units apart, horizontally and vertically.

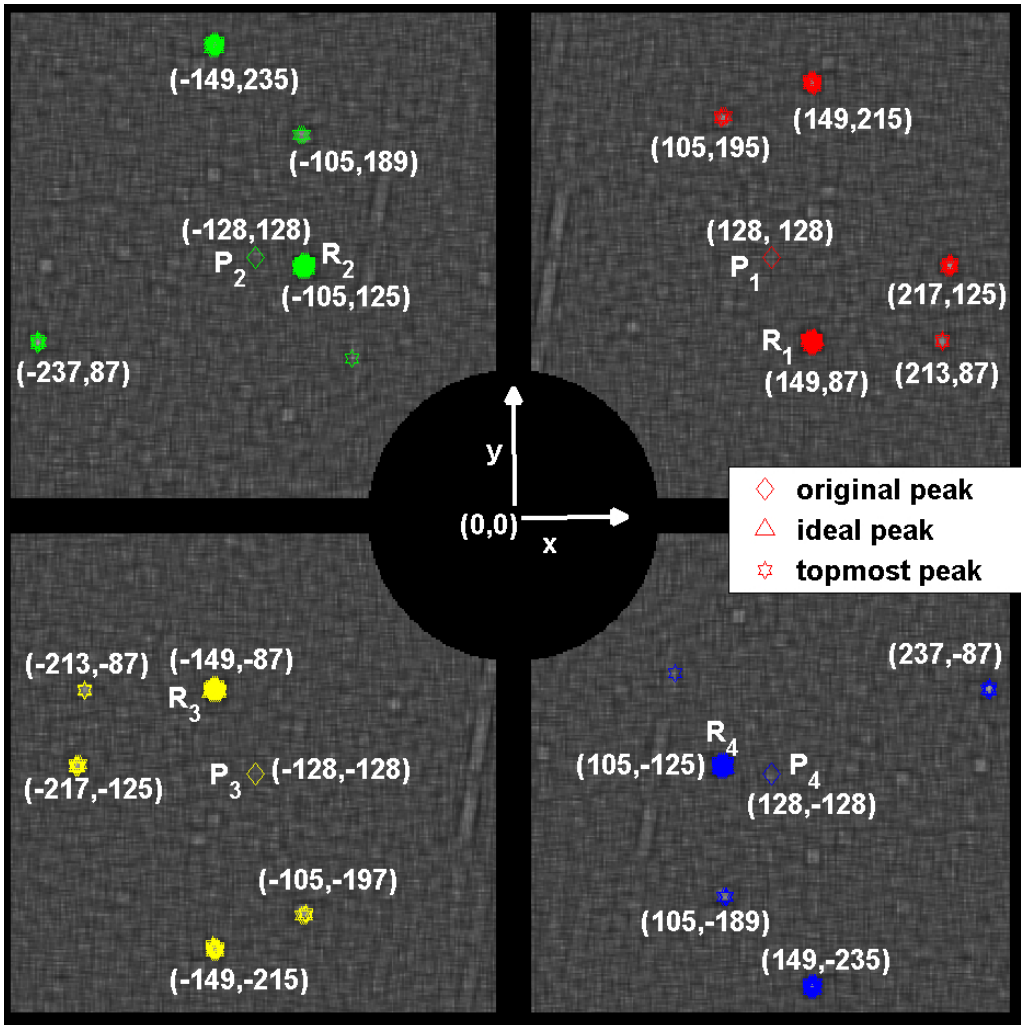
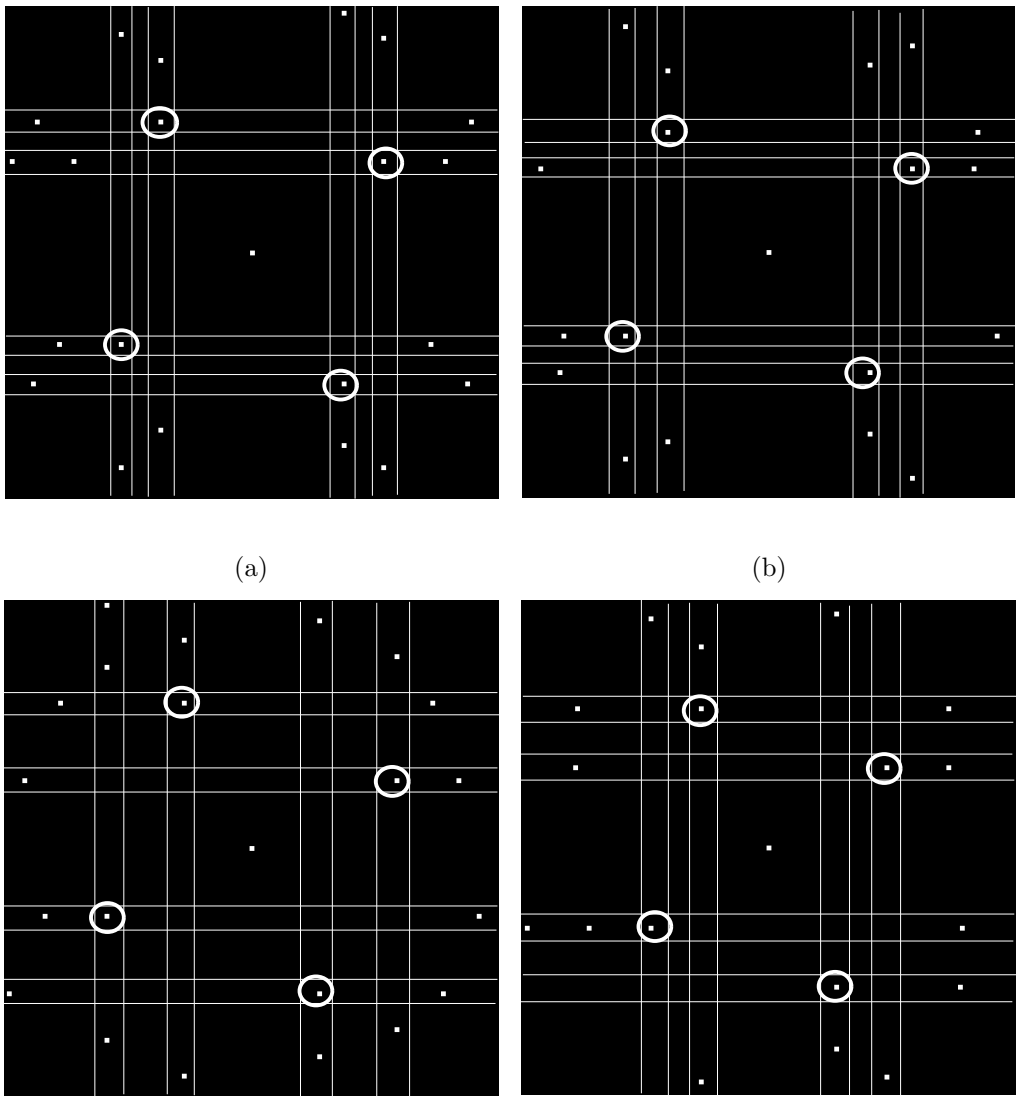


Figure 5.8: Similar plot, as in Fig. 5.7, with the transformation corresponding to $\{\theta = 30°, s_x = 1, s_y = 1.2\}$, along with 20% cropping, and QF_a=75.



(a) (b)

(c) (d)

Figure 5.9: (from left to right) (a)-(d) correspond to $\theta = 10°, s_x = s_y = 1.1$, $\theta = 30°, s_x = 1, s_y = 1.2$, $\theta = 20°, s_x = s_y = 1.1$, and $\theta = 15°, s_x = 1.1, s_y = 1.3$, along with 20% cropping and $QF_a = 75$. The circled locations denote $\{R_i\}_{i=1}^{4}$, while the horizontal and vertical lines show how the JPEG-induced peaks (white dots) are at multiples of 64 units apart from $\{R_i\}_{i=1}^{4}$.

Step 2: We find the row indices in the DFT grid which are more likely to contain JPEG-induced peaks. If a row contains the actual peak, then there will be other JPEG-induced peaks in the same row at multiples of 64 units away. We identify the rows having more instances of such peaks from the top half of the DFT grid. The corresponding rows in the bottom half will be determined by conjugate symmetry. The number of possible JPEG-induced peaks for the i^{th} row $\phi_{64}(i)$ are determined using (5.11).

$$\mathcal{B} = \{\rho_{j,k,x} : \rho_{j,k,y} = i,\ j = 1, 2\},\ \mathcal{B} = \text{ascending sort}(\mathcal{B}),$$
$$\phi_{64}(i) = |k : (\mathcal{B}(k+1) - \mathcal{B}(k)) \approx 64m, m \neq 0, m \in \mathbb{Z}| \tag{5.11}$$

We choose the entries in ϕ_{64} (5.11) having more than one peak and store them in $\mathcal{A}$ (5.12). Since we use window-based peak insertion, a row adjacent (within $\delta_{\mathcal{A},diff}$ units, as shown in (5.12)) to the row containing the actual inserted DFT peak can also have "JPEG-like neighboring peaks". We group together the closely placed rows and these rows are represented by their average value ϕ_{row} (5.13). The number of JPEG peaks associated with the rows selected in ϕ_{row} are given by ϕ_{mag} (5.14).

$$\mathcal{A} = \text{ascending sort}\{k : \phi_{64}(k) > 1\},\ \mathcal{E} = \{k : (\mathcal{A}(k+1) - \mathcal{A}(k)) > \delta_{\mathcal{A},diff}\}, \tag{5.12}$$

$$\phi_{row}(k) = \text{ mean of } \{\mathcal{A}(\mathcal{E}(k-1)+1), \ldots, \mathcal{A}(\mathcal{E}(k))\}, 2 \leq k \leq |\mathcal{E}|, \text{ and } \mathcal{E}(0) = 0. \tag{5.13}$$

$$\phi_{mag}(k) = \sum_{i=\mathcal{A}(\mathcal{E}(k-1)+1)}^{\mathcal{A}(\mathcal{E}(k))} \phi_{64}(i) \tag{5.14}$$

Step 3: The next step is to retain the significant rows out of the row indices in ϕ_{row} (5.13) such that we do not select peaks which are JPEG-induced neighbors of each other. Let the number of elements in ϕ_{row} equal T. We construct a $T \times T$ symmetric matrix Λ where if $\Lambda(i,j)$ equals 1/0, it means that if $\phi_{row}(i)$ is retained, then $\phi_{row}(j)$ will/will not be retained and vice versa.

Let the column indices that corresponds to the maximum valued element in $\mathbf{T}_{\Delta}(F'_{mag}, \cdot, \phi_{row}(i))$ and $\mathbf{T}_{\Delta}(F'_{mag}, \cdot, \phi_{row}(j))$ be χ_i and χ_j respectively. We then determine whether the peak at $(\chi_i, \phi_{row}(i))$ is a JPEG induced peak resulting from the peak at $(\chi_j, \phi_{row}(j))$, or its complex conjugate version. If yes, then $\Lambda(i,j)$ and $\Lambda(j,i)$ are set to 0; else they are set to 1.

For the i^{th} row in Λ, the rows containing peaks which serve as JPEG-like neighbors of the peak at $(\chi_i, \phi_{row}(i))$ constitute the set $\mathcal{M}$ (5.15). Of these rows, we choose that row, using (5.16), which has the maximum number of "JPEG-

like neighboring peaks", as is given by ϕ_{mag} (5.14) values. *To reiterate, a JPEG induced peak can be higher in magnitude than the original peak but the number of "neighboring JPEG-like peaks" is higher in general for the actual peak than for a JPEG induced peak.*

$$\mathcal{M} = \{k : \Lambda(i,k) = 0\}, \text{ and } k_{opt} = \arg\max_{k} \phi_{mag}(\mathcal{M}(k)) \tag{5.15}$$

$$\varphi_{row}(i) = \phi_{row}(\mathcal{M}(k_{opt})) \tag{5.16}$$

The set of unique elements in φ_{row} (5.16) is denoted by ψ_{row} and let the number of such rows be T_{row}.

Step 4: Similarly, we obtain the columns in the second and third quadrants which are more likely to contain the actual peaks. Let ψ_{col} denote the unique column indices and T_{col} is the cardinality of ψ_{col}.

Step 5: We use the selected row and column indices to obtain the potential peak locations for the first two quadrants. Considering the peak location $(\psi_{col}(j), \psi_{row}(i))$, we perform some local refinement, using (5.17)-(5.18), to obtain a more precise peak estimate in a $(2\delta_W + 1) \times (2\delta_W + 1)$ local window $\mathcal{W}$ around $(\psi_{col}(j), \psi_{row}(i))$. The precisely estimated peak location, corresponding to $(\psi_{col}(j), \psi_{row}(i))$, is given by $(\xi_{col}(T_{col}.(i-1)+j), \xi_{row}(T_{col}.(i-1)+j))$ (5.17) and the peak magnitude at that location is $\xi_{mag}(T_{col}.(i-1)+j)$ (5.18).

$$\xi_{col}(T_{col}.(i-1)+j), \xi_{row}(T_{col}.(i-1)+j) = \arg\max_{(k_1,k_2)\in\mathcal{W}} \mathbf{T}_{\Delta}(F'_{mag}, k_1, k_2), \tag{5.17}$$

$$\xi_{mag}(T_{col}.(i-1)+j) = \max_{(k_1,k_2)\in\mathcal{W}} \mathbf{T}_{\Delta}(F'_{mag}, k_1, k_2) \tag{5.18}$$

where $k_1 \in \{\psi_{col}(j) - \delta_W, \psi_{col}(j) + \delta_W\}, k_2 \in \{\psi_{row}(i) - \delta_W, \psi_{row}(i) + \delta_W\}$.

Why do we need the local refinement process to better localize the peak location? When we compute ϕ_{row} (5.13) and ϕ_{col}, we consider the mean value computed over consecutive row indices and column indices, respectively. This can cause the peak positions to be one/more units away from the actual peak locations in $\mathbf{T}_{\Delta}(F'_{mag}, \cdot, \cdot)$ over the local window. *While using the geometric transformation model to test if a set of 4 peaks satisfies the model, a slight shift in a peak location may result in errors.*

Step 6: Considering the top μ peaks in each of the first two quadrants based on ξ_{mag} (5.18) values, we obtain μ^2 4-tuples of peaks. Assuming a specific model for the transformation matrix as discussed in Section 5.5.1, we find which 4-tuples result in proper transformation matrices. Let the number of such 4-tuples be ν. Let the ν peak locations for the first 2 quadrants be $\{\eta_{1,i,x}, \eta_{1,i,y}\}_{i=1}^{\nu}$ and $\{\eta_{2,i,x}, \eta_{2,i,y}\}_{i=1}^{\nu}$. Of these ν 4-tuples of peaks, we find the sum of the $\mathbf{T}_{\Delta}(F'_{mag}, \cdot, \cdot)$ values at the peak locations, so as to constitute the vector η_{mag} (5.19). The 4-tuple

of peak locations that result in the maximum value of η_{mag} for these ν peaks is identified. Since we have chosen row and column indices which are well separated from each other (consecutive rows and columns are averaged to compute ϕ_{row} and ϕ_{col}), a nonzero 3D histogram bin (H_{RS} or H_{SR} as in Section 5.5.1) will mostly have only one element and so, the bin-count cannot be used as a criterion.

$$\eta_{mag}(i) = \mathbf{T}_{\Delta}(F'_{mag}, \eta_{1,i,x}, \eta_{1,i,y}) + \mathbf{T}_{\Delta}(F'_{mag}, \eta_{2,i,x}, \eta_{2,i,y}),\ 1 \leq i \leq \nu,$$
$$i^* = \arg\max_k \eta_{mag}(k) \tag{5.19}$$

The estimated peak locations Q_1 and Q_2 equal $(\eta_{1,i^*,x}, \eta_{1,i^*,y})$ and Q_2 is $(\eta_{2,i^*,x}, \eta_{2,i^*,y})$, respectively, using (5.19). In Table 5.6, we report $p_{A,m1}$ (obtained using prior assumption about the geometric transformation) and $p_{A,m2}$ (using the geometric transformation related assumption and also using JPEG peak location property). It is seen that the latter approach helps in more accurate transformation estimation without involving higher PSNR. While computing $p_{A,m2}$, errors will occur only when there are high magnitude (higher than the $\mathbf{T}_{\Delta}$ value at the actual peak location) peaks, which are not identified as JPEG-induced peaks, and which also satisfy the geometric transformation model. The value of various parameters used in computing $p_{A,m2}$ in Table 5.6 are: $\delta_{\mathcal{A},diff} = 4$, $\gamma = 60$, $\mu = 10$, and $\delta_W = 6$.

We also show the effect of down-scaling on the accuracy of the transformation estimation after significant JPEG compression in Table 5.7. In absence of severe

Table 5.6: The geometric transformation parameter estimation results are presented with/without using JPEG peak location property, for varying QF_a - here, $\delta_\theta = 0.5°$ and $\delta_s = 0.05$.

QF_a used	without JPEG peak location		with JPEG peak location	
	$p_{A,m1}$	PSNR (dB)	$p_{A,m2}$	PSNR (dB)
85	0.99	45.65	0.99	48.18
75	0.94	44.25	0.97	47.20
50	0.82	41.48	0.86	43.80
25	0.78	35.55	0.83	37.25

Table 5.7: The rotation angle is fixed at 20° while the scale factor ($s_x = s_y$) is varied from 1.1 to 0.70, and $p_{A,m2}$ is computed using angle and scale resolutions of 0.5° and 0.05, respectively.

scaling	1.10	1.00	0.95	0.90	0.85	0.80	0.75	0.70
$p_{A,m2}$	0.97	0.96	0.92	0.90	0.86	0.82	0.79	0.76
PSNR (dB)	47.20	46.85	45.05	43.20	40.75	38.90	37.75	36.30

compression (using QF_a of 99), the estimation is near-perfect for scale factors as low as 0.5. However, using QF_a of 75, the smoothing introduced by the down-scaling process is severe enough to suppress the actual peaks, and the smoothing becomes more severe with more down-scaling.

5.6 Experimental Results

The performance of the geometric transformation estimation is studied after varying these parameters.

Table 5.8: $p_{A,m}$ is computed for $\theta = 20°$, $s_x = 1.1$, $s_y = 1.1$ along with 20% cropping, for $QF_a = 75$, and various resolutions for rotation (δ_θ) and scale (δ_s) parameters, e.g. (0.5, 0.05) indicates $\delta_\theta = 0.5°$ and $\delta_s = 0.05$. PSNR values are in dB.

(0.5, 0.05)		(0.25, 0.05)		(1, 0.025)		(0.5, 0.025)		(0.25, 0.025)		(0.125, 0.05)	
$p_{A,m}$	PSNR	$p_{A,m}$	PSNR	$p_{A,m}$	PSNR	$p_{A,m}$	PSNR	$p_{A,m}$	PSNR	$p_{A,m}$	PSNR
0.94	44.25	0.86	42.81	0.90	43.89	0.90	43.74	0.87	43.22	0.85	42.41

5.6.1 Effects of Varying DFT Size

The accuracy of the transformation estimation depends on the resolution we set for the scaling and rotation parameters, as shown in Table 5.8. Here, a DFT size of 512×512 is used and it is assumed that the geometric transformation constitutes of rotation and/or scaling (in any sequence). The resolution used to compute the angle and scale parameters are denoted by δ_θ and δ_s, respectively. The probability of correct estimation of the transformation parameters in A, which is denoted by $p_{A,m}$, is expressed in Table 5.8 for a given set of angle and scale resolutions. E.g. a resolution of 0.5° and 0.05 is used for the angle and scale parameters, respectively.

At the decoder side, it is assumed that the computed rotation angle and scale factors are multiples of 0.5° and 0.05, respectively, and the angle and scale values thus computed have to equal 20° and 1.1, respectively, for it to be considered a success in $p_{A,m}$ computation. For all the experiments, we use a rotation angle of 20° and a scaling factor of 1.1 is used along both dimensions. Also, for all subsequent tables (after Table 5.8), we use $\delta_\theta = 0.5°$ and $\delta_s = 0.05$.

Table 5.9: p_A is computed for different DFT sizes:

JPEG QF (QF_h)	DFT size = 512×512		DFT size = 1024×1024	
	Accuracy	PSNR(dB)	Accuracy	PSNR(dB)
40	0.85	40.25	0.86	40.32
50	0.86	43.80	0.87	44.20
60	0.90	45.90	0.92	45.98
75	0.97	47.20	0.98	47.60

We have experimented with the DFT size N, as shown later in Table 5.9. Since the 250 images we experimented with were 512×512 images, N is set to 512. It is seen that slight performance improvement is obtained using N of 1024 as compared to 512 - the cost involved is searching over $(\frac{1024}{512})^2 = 4$ times as many points compared to when $N = 1024$. Here, it is assumed that the geometric transformation consists of rotation and/or scaling (in any sequence). We also use the property that the position of the JPEG-induced peaks can be predicted in terms of the actual location of the actual inserted DFT-domain peak. This combination of prior assumption about the transformation and prediction of JPEG peak locations is used for all subsequent experiments in the report.

5.6.2 Effects of Cropping

Cropping can be interpreted as multiplying an image by a rectangle, where the rectangle size determines the image size after cropping. *Multiplying by a rectangle in the spatial domain is equivalent to convolving with a 2-D sinc function in the*

DFT domain. The smaller the size of the rectangle (smaller is the number of pixels retained), the corresponding sinc in the DFT domain will have a wider variance making the DFT of the cropped image more blurred and peak picking more difficult.

Three variations of cropping are shown in Fig. 5.10. Starting off with a $N_1 \times N_2$ image, the cropped image has $0.8N_1$ rows and $0.8N_2$ columns in all the 3 cases (a)-(c). In (a), the discarded pixels come from the ends while in (b), they come from the central part. In (b), the 4 cropped regions are put together to constitute the final image. In (c), 1 row (column) out of every 5 rows (columns) is removed. Thus, on an average, the size of the individual blocks that are retained in the pixel domain is smallest for (c) and hence, the DFT peaks in (c) are maximally blurred by the corresponding higher variance of the sinc functions. Experimental results show that p_A is highest for (a) and lowest for (c).

Table 5.10: For the cropping experiments, $QF_a = 75$, the starting windowed peak strengths $[v_1\ v_2\ v_3\ v_4] = [14\ 10\ 4\ 2]$, and the results are shown after using various cropping methods - methods (a)-(c) are explained in Fig. 5.10.

crop	Method a		Method b		Method c	
	p_A	PSNR(dB)	p_A	PSNR(dB)	p_A	PSNR(dB)
0.6	0.955	46.90	0.850	44.10	0.518	40.06
0.8	0.970	47.20	0.872	44.33	0.548	42.02

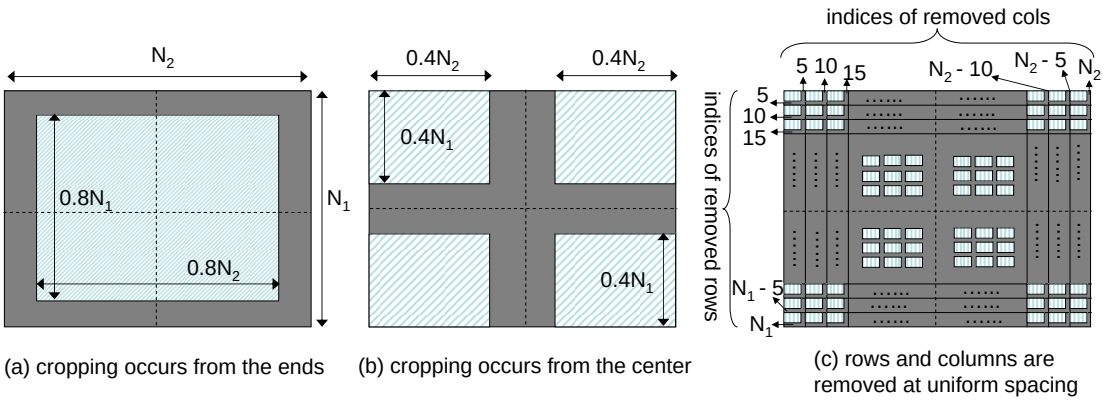


Figure 5.10: Various cropping approaches for a $N_1 \times N_2$ image - the greyish part denotes the cropped out (discarded) image part.

5.6.3 Robustness to Small Local Transformations

The template based method is useful for estimation of global affine transformations. If there are small local regions which undergo transformations different from that of the initial global transform, we cannot determine the individual transformations undergone by the local regions. However, if we can still determine the global transformation in spite of the small local transformations, then the received image can be properly aligned with the original grid and decoding is possible in those image regions which do not suffer local transformations.

When the entire image is transformed using A while there are n small local regions which are transformed using $A_1, A_2, \cdots, A_n$, A can still be recovered provided the local regions are small enough. E.g. if four regions are considered in the pixel domain, each centered around a quadrant center, and each region has

Table 5.11: Results with varying amounts of local transformations, using $QF_a = 75$

center	0.05	0.10	0.15	0.20	quadrant	0.05	0.10	0.15	0.20
p_A	0.97	0.96	0.92	0.84	p_A	0.96	0.95	0.91	0.80
PSNR	47.20	46.78	44.88	42.67	PSNR	46.76	44.30	42.10	40.91

a different geometrical transformation from that of the overall image, the global transform can still be recovered if each region is not more than 30% of the quadrant dimensions. In Table 5.11, results are presented for accurate A accurately for a variety of local region sizes. In Table 5.11, by "quadrant" (0.05), we refer to the case where we take a region of dimension 5% that of the quadrant around the quadrant center, (for all the four quadrants) and then subject it to a transformation, different from the overall global transform (A). By "center"(0.05), we refer to the scenario where a local region of dimension 5% of a quadrant is considered around the DFT center and it is subjected to a transformation different form A.

5.7 Method for Robust Key-point Selection

The robustness of a key-point selection scheme requires that if the image undergoes changes due to template insertion, data embedding, and channel attacks, it should still be possible to *exactly* recover one/more of the original key-points from the received image. A standard KP detection scheme yields a large number

of points. We present a pruning scheme to return a small number of salient points and in general, we observe that there are common KP between the encoder and decoder even after pruning.

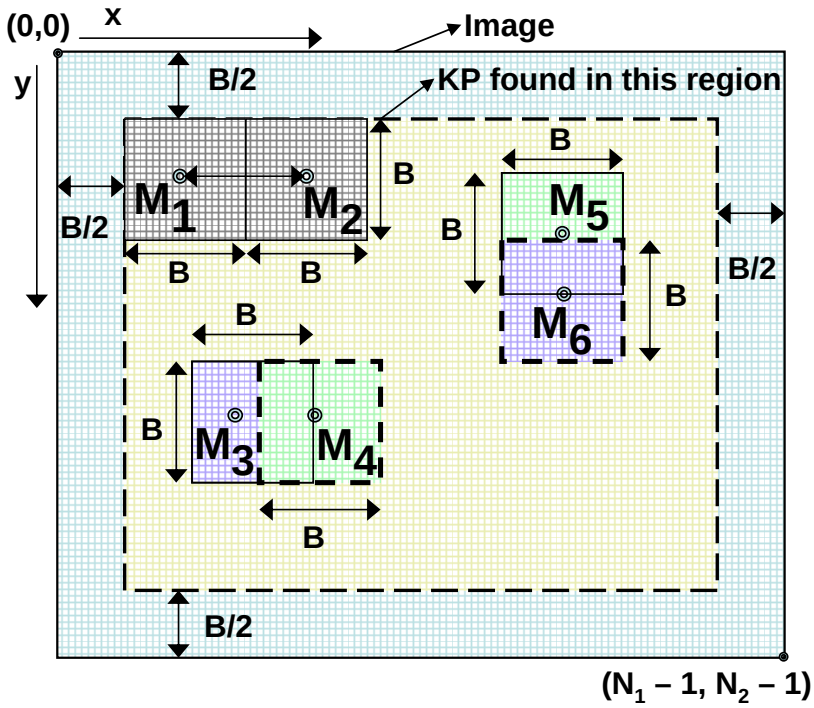


Figure 5.11: The KP pruning steps are shown. For two prospective KP ((M_3, M_4) or (M_5, M_6)) which are less than B apart from each other, we retain the one with the higher strength. When the distance between two points (e.g. (M_1, M_2)) is greater than or equal to B, both can be retained.

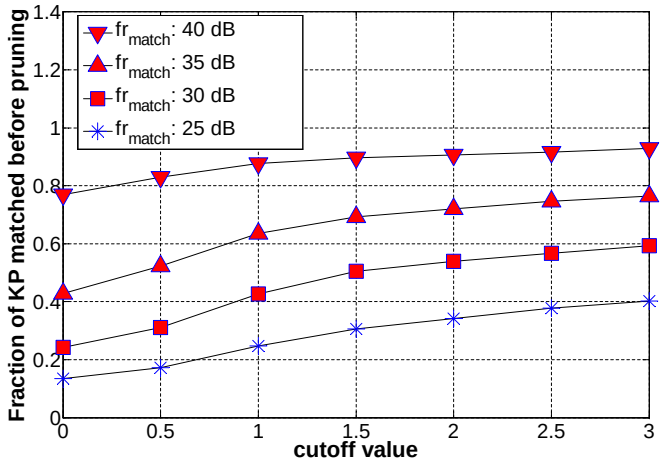
The steps in the key-point computation are outlined below.

(1) **Border effects** - the key-points are computed in $f(x, y)$, where $B/2 < x \leq (N_1 - B/2)$ and $B/2 < y \leq (N_2 - B/2)$, as shown in Fig. 5.11. This ensures that the $(B \times B)$ local regions obtained around the key-points do not extend outside

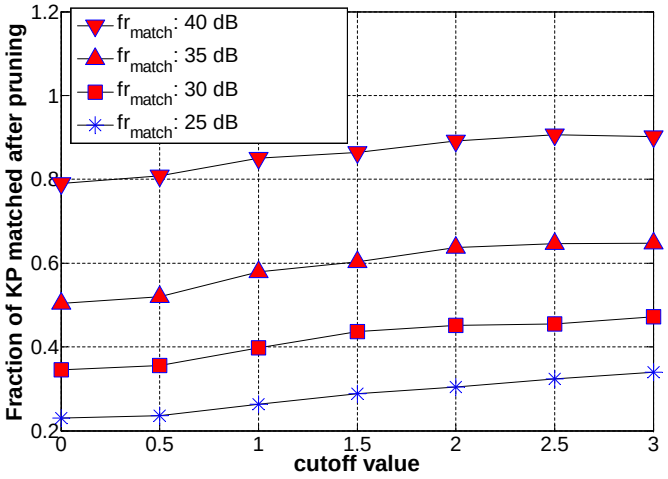
the image.

(2) **Actual key-point computation** - it is done using a variety of methods (SIFT [68]/ Harris corner [47]/ Nobel-Forstner [32, 77]). The corner strength $I(\cdot)$ is computed for all key-points. We retain only those key-points whose corner strength exceeds a certain pre-defined threshold δ_{th}. Let there be K_1 such key-points, given by $\{M_i = (x_i, y_i)\}_{i=1}^{K_1}$.

(3) **Removing Nearby Key-points** - One may end up with key-points $M_3 = (x_3, y_3)$ and $M_4 = (x_4, y_4)$, where $\min(|x_3 - x_4|, |y_3 - y_4|) < B$, as in Fig. 5.11. We cannot use both these key-points to obtain the hiding regions as we want the $B \times B$ local regions to be disjoint. The corner strength is used to discriminate between 2 (or more) points in such cases; e.g. we retain the key-point M_3 if $I(M_3) > I(M_4)$ and vice versa. This process is repeated till there are no close enough key-points. After removing the nearby key-points, let the final number of points be K_2, where $K_2 \leq K_1$. As an example of the amount of pruning achieved for SIFT key-points, the average value of K_1 using pruning step (1)-(2) (for 512×512 images) varies from 900 to 5000 as δ_{th} is decreased from 5 to 0; after pruning using steps (1)-(3), the corresponding K_2 varies from 12 to 21.



(a)



(b)

Figure 5.12: Fig.(a) and (b): The figures show the fraction of blocks that are correctly detected using SIFT key-points at the decoder side before and after pruning, respectively.

5.7.1 Design Criteria - selecting the cutoff for key-point detection (δ_{th}) and the embedding region size (B)

The design parameters that affect K_2, for a given corner detection method, are δ_{th} and B. K_2 decreases as δ_{th} or B (or both) increases. Let X_{enc} and X_{dec} denote the set of key-points at the encoder and decoder side, (after pruning) where the number of elements in these sets is K_{enc} and K_{dec}, respectively. The decoder has a better chance of proper data recovery for a higher value of K_{dec}. *The drawback of using a higher value of K_{dec} is the increased computational complexity - the turbo decoder has to be run for the local region around each key-point till the decoder successfully converges at one of them.*

We define the following quantities that are used later in this section:

• $X_{BER,0}$ denotes the set of key-points at which the data is successfully recovered (with zero BER),

• fr_{match} denotes the fraction of the K_{dec} key-points that correspond exactly to any of the K_{enc} key-points used at the encoder, i.e. $fr_{match} = |X_{dec} \cap X_{enc}|/K_{dec}$,

• $fr_{BER,0}$ denotes the fraction of successfully detected KP at the decoder for which the embedded data is also fully recovered, i.e. $fr_{BER,0} = |X_{BER,0}|/|X_{dec} \cap X_{enc}|$.

Errors can be of 3 types: (i) in the geometric transformation estimation (type 1), or (ii) in the key-point detection (type 2), or (iii) in decoding the data even if

the key-point detection is successful (type 3). In subsequent tables, the following error terms are used:

• E_A : the number of images for which A could not be successfully estimated (type 1),

• E_{KP} : the number of images for which $|X_{enc} \cap X_{dec}| = 0$ (out of those images for which A is successfully estimated) (type 2),

• E_{dec} : the number of images for which $|X_{enc} \cap X_{dec}| > 0$ but the data-bits were not successfully recovered (out of those images for which A is successfully estimated) (type 3).

Effect of Varying Threshold: In Fig. 5.12(a) and 5.12(b), we show the effect of varying cutoff (δ_{th}) on fr_{match} for AWGN channels with varying noise levels. The final stego image is JPEG compressed at QF_a of 75. "Before pruning" refers to the case where we consider all the K_1 key-points (without removing the nearby points) while "with pruning" refers to K_2 key-points ($B = 80$ is used). This shows that as δ_{th} increases and the noise introduced decreases, the key-points detected at the decoder are more likely to be the same as those detected by the encoder - hence, fr_{match} increases with δ_{th}, and with a higher SNR.

The variation of the key-point detection performance with δ_{th} for the Nobel Forstner (NF) and SIFT key-point detectors (the results outperformed Harris corner based KP) is studied in Table 5.12. When δ_{th} is low, we have "less reliable"

Table 5.12: As the threshold δ_{th} is increased, K_{enc} slightly decreases, and (E_{KP}+ E_{dec}) also increases slightly. We use $QF_h = 60$, $\lambda = 5$ and $QF_a = 75$.

Nobel Forstner					
δ_{th}	avg(K_{enc})	E_{KP}	E_{dec}	fr_{match}	data-bits
0	21.30	7	6	0.2130	162.51
0.0001	21.00	7	6	0.2159	162.51
0.001	18.46	12	2	0.2422	163.10
0.005	15.13	19	0	0.2781	165.16
SIFT					
δ_{th}	avg(K_{enc})	E_{KP}	E_{dec}	fr_{match}	data-bits
0	20.92	25	8	0.1238	156.28
0.5	19.92	26	7	0.1285	156.28
1	17.32	35	0	0.1388	157.49
1.5	15.97	41	0	0.1487	161.62

KP and fr_{match} is low; however, we also miss detecting any common KP in fewer cases. We assume that $E_A = 0$ for these experiments. Based on these results, to minimize ($E_{KP}+E_{dec}$), we use $\delta_{th} = 0.001$ for NF and 0 for SIFT based key-points in our system.

Fixing B Depending on Data Rate Requirements: The variation of the fraction of matched KP, the number of detected KP and the data-rate with B are shown in Fig. 5.13(a), 5.13(b) and 5.13(c), respectively. As B increases, the effective data-rate increases (data-rate $\propto B^2$) but the number of key-points obtained after removing nearby key-points (K_{enc} and K_{dec}) decreases. Assume that the number of data bits we need to embed is 180. We can obtain a data rate of 180 bits for $B \geq 80$ (from Fig. 5.13(c)). If we use $B = 80$, we hide just the required amount but the number of blocks correctly detected and decoded increases. If

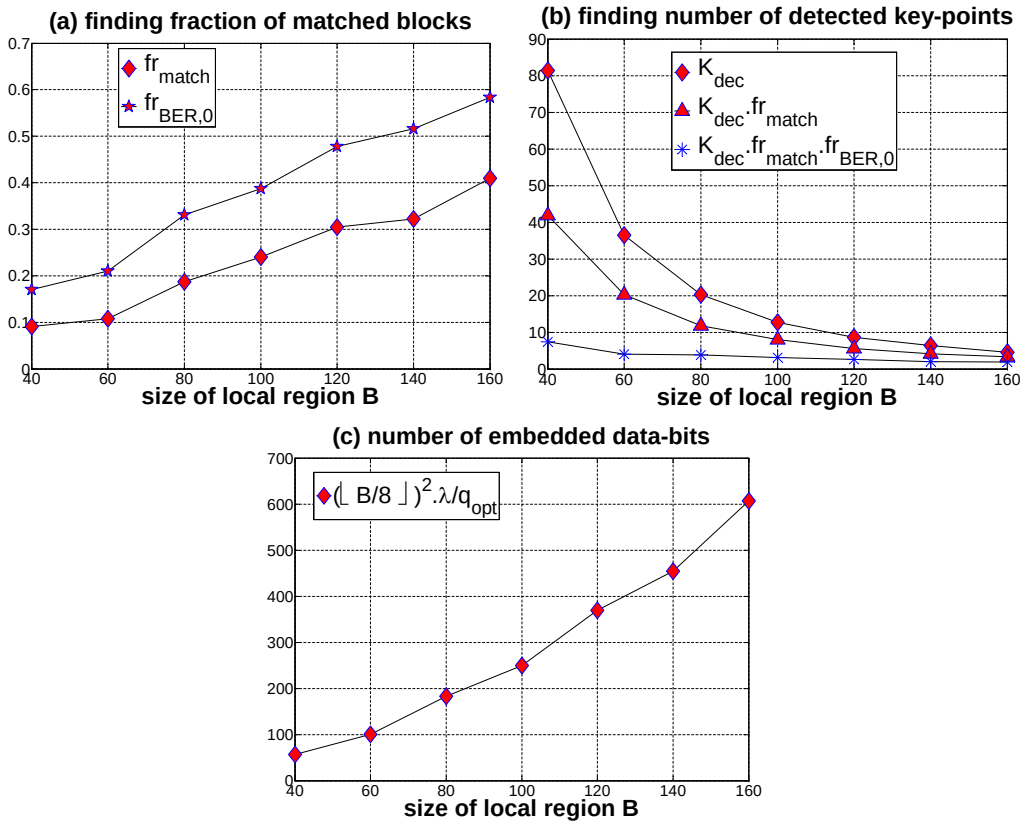


Figure 5.13: Fig. (a) shows the fraction of detected and successfully decoded blocks. Fig. (b) shows the number of key-points detected at the decoder side (K_{dec}), the number of local blocks that are correctly detected ($K_{dec}.fr_{match}$) and decoded ($K_{dec}.fr_{match}.fr_{BER,0}$). Fig.(c): The number of successfully embedded data bits is shown. The number of data bits embedded in a $B \times B$ region $= \lfloor \frac{B}{8} \rfloor^2 . \frac{\lambda}{q_{opt}}$ where λ elements are used for hiding per 8×8 block and the minimum RA-code redundancy factor needed for proper decoding is q_{opt}. We use $QF_h = 60, \lambda = 5, QF_a = 75$.

we increase B, the data-rate goes up significantly but the number of blocks correctly detected and decoded also decreases. *For maximal robustness, we use the minimum B that satisfies the data-rate requirements*, i.e. we use $B = 80$ in this case.

We compare the performance of the various key-point detectors under various attacks (Figs. 5.14 and 5.15) - it is observed that Nobel-Forstner (NF) key-points result in a higher fr_{match} and also, the embedded databits can be successfully retrieved for a higher fraction of images.

5.8 Overall Results for the Hiding Framework

We present results of (i) geometric transformation estimation, (ii) key-point detection and (iii) successful RA decoding for various attacks. To refresh the reader's mind about the relative importance of these three modules, we present the following break-up of their functionalities:

(a) Geometric transformation - the major part of the chapter is dedicated to proper synchronization through accurate estimation of the transformation matrix. The major obstacle in the path of correct estimation is JPEG compression; hence, majority of the work is on correctly identifying the

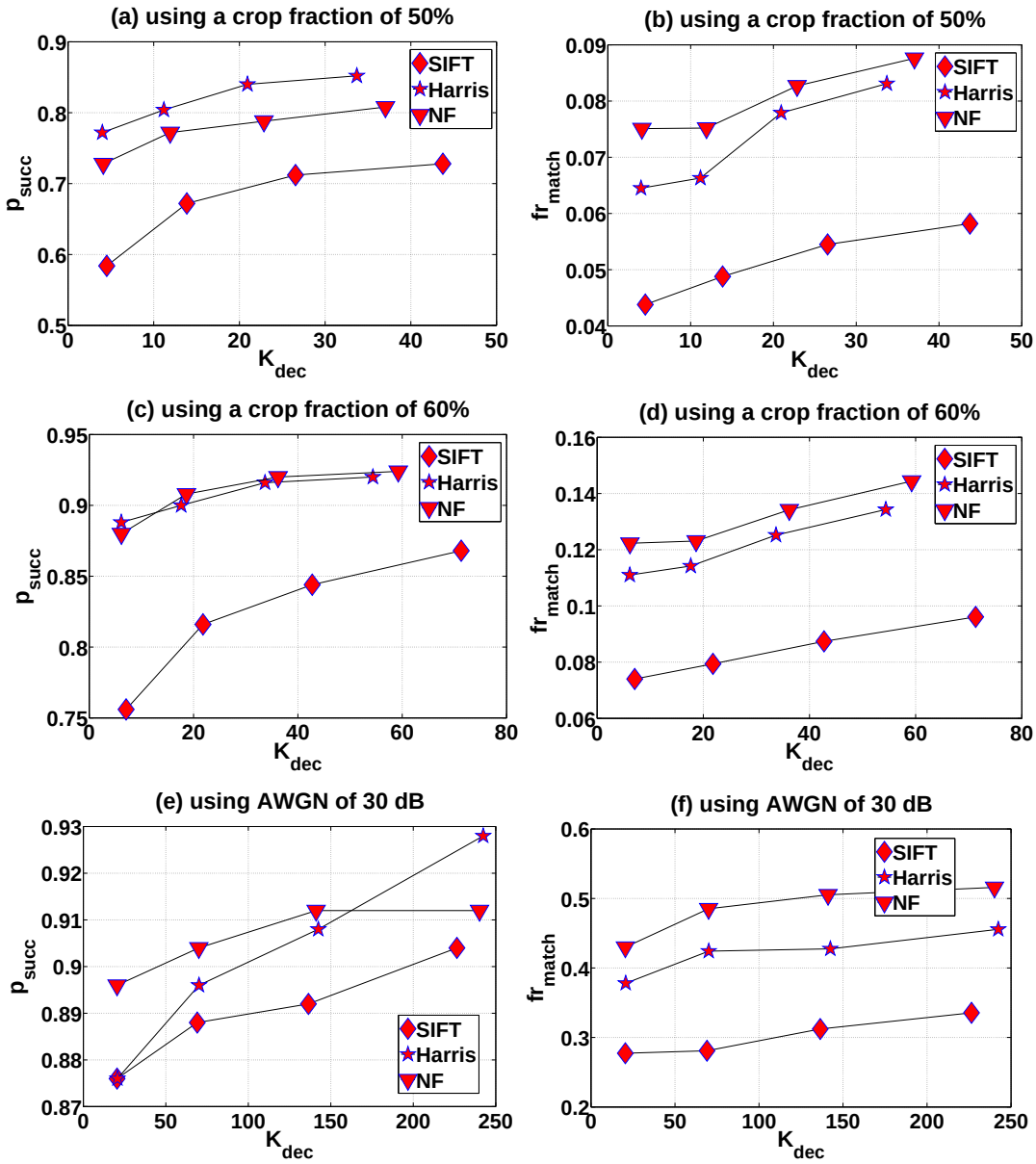


Figure 5.14: Based on the above experiments, Nobel-Forstner (NF) key-points perform better than Harris and SIFT key-points; here p_{succ} is the fraction of images for which we successfully retrieve the embedded data. The experiments are performed on 250 images and the average fr_{match} is reported. In (c)-(d), a crop fraction of 60% means 60% of the image is retained along both the axes.

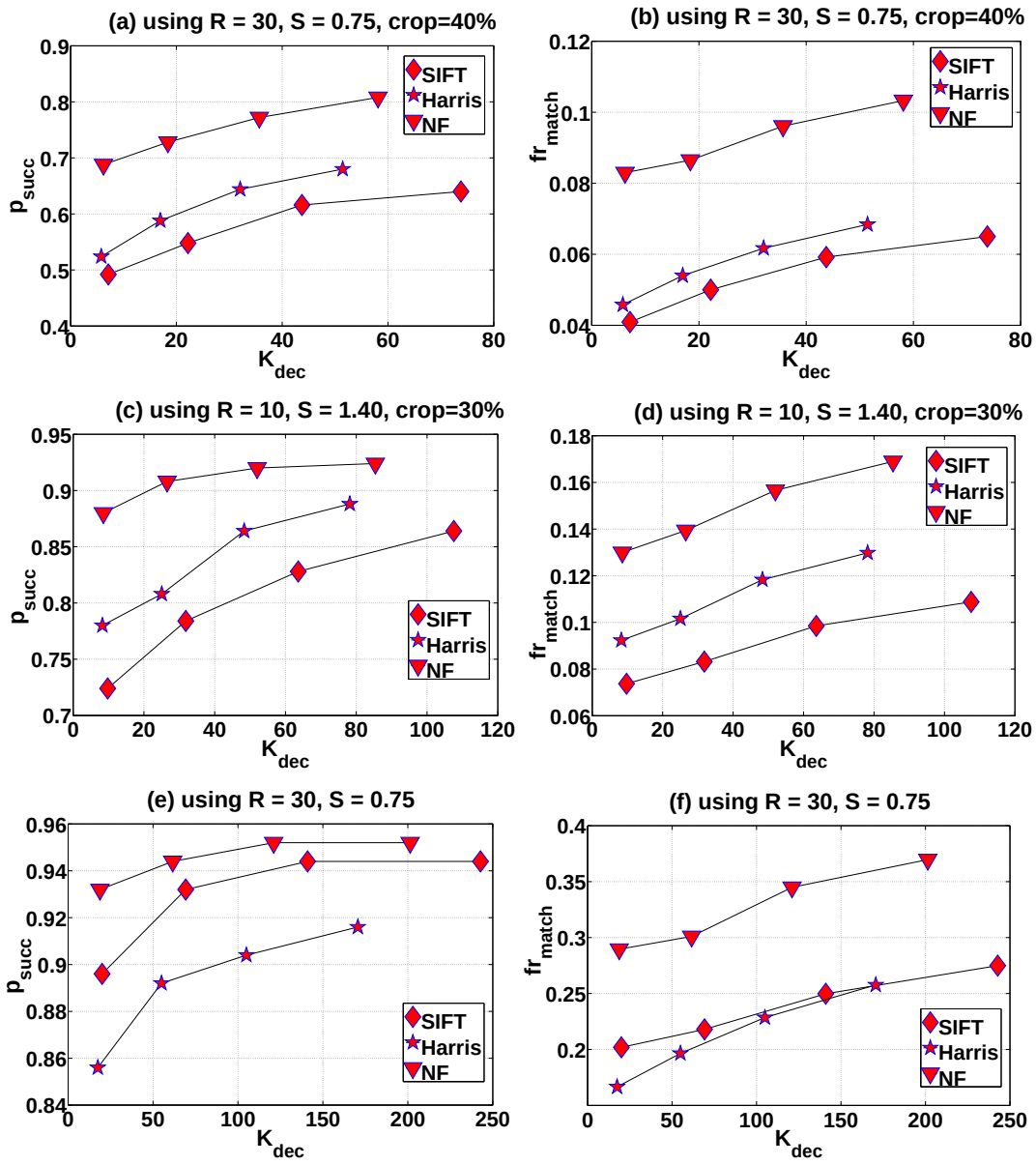


Figure 5.15: The same experiments as in Fig. 5.14 are repeated under a different set of transformations. In (g), R=30 refers to a rotation angle of 30°, S=0.75 refers to a scaling factor of 0.75 for both the axes.

template DFT peaks even in the presence of (or by intelligently discarding) the JPEG induced peaks.

(b) Accurate key-point detection: If the geometric transformation is correctly estimated and the received image is synchronized with the original, the next problem is to ensure that the decoder considers the same regions for data recovery as the encoder. This is done through accurate key-point detection after suitable pruning.

(c) RA-decoding: After geometric alignment and correct key-point identification, the embedded data can be recovered provided the redundancy for the RA code is high enough to counter the noise attack (we have experimented with AWGN addition, gamma variation and Gaussian blur attacks).

The geometric transformation parameters are estimated *using a model-based approach*, and *using JPEG peak location property*, as in Section 5.5.2. We first discuss how we fix λ, the size of the hiding band, before describing the performance against various attacks.

Deciding the embedding band size λ: For less noisy channels, E_{KP} *dominates over* E_{dec}. The hiding rate is increased using B and λ. As shown in Fig. 5.13, B is determined based on the data-rate requirement for a given λ. We fix λ to the maximum value for which data recovery is possible for a given noise channel

(varying λ normally does not affect E_{KP} for a fixed B). For more noisy channels, E_{dec} *dominates over* E_{KP}. For such channels, we reduce E_{dec} by choosing a low value of λ and the resultant data-rate decreases. For noisy channels, the errors/erasures introduced will be less if the DCT coefficients are larger in magnitude. The magnitudes of the coefficients encountered during zigzag scan generally decrease as the scan progresses. Thus, by using smaller λ (using higher magnitude DCT coefficients for hiding), the channel errors are reduced at the expense of a lower data-rate.

Seam carving attacks: We have not geometrically transformed the image - E_{KP} increases as more seams are removed. To get more matched key-points, we increase K_{dec} by removing a lesser number of key-points (e.g. if nearby key-points are removed when they are $\frac{B}{t}$ apart, then less key-points are removed for $t = 4$ compared to $t = 1$), as shown in Table 5.13. Data recovery is possible only if the removed seams are such that one/more of $B \times B$ embedding regions are left unchanged.

Gaussian blur, AWGN addition and gamma correction attacks: From Tables 5.14-5.16, we observe that $E_A \approx 0$ when QF_a=99; however, E_A is substantially higher for more severe JPEG compression at QF_a of 75. In Table 5.14, we observe that as the blurring becomes more severe (higher σ), geometric alignment becomes more difficult. The DFT matrix is then smoothed to such an extent that

Table 5.13: Effect of seam carving attacks for λ=5, $QF_h = 60$, $QF_a = 99$, B=80, the parameter t is such that nearby key-points are removed when they are $\frac{B}{t}$ apart.

# seams	t	E_{KP}	E_{dec}	data-bits	fr_{match}	t	E_{KP}	E_{dec}	data-bits	fr_{match}
10	1	0	2	241.32	0.6423	4	0	0	252.40	0.5925
50	1	0	2	175.28	0.5092	4	0	1	178.94	0.4593
100	1	1	23	121.04	0.3803	4	0	16	122.47	0.3464
150	1	4	54	81.88	0.2829	4	1	36	85.18	0.2407
200	1	6	108	57.7868	0.2123	4	1	77	60.8837	0.3764

the DFT peaks often get suppressed. We do not report results for more severe blurring ($\sigma > 1$) because decoding becomes a problem (very high E_{dec}) even if A is correctly estimated. In Table 5.15, it is seen that JPEG compression affects the geometric alignment much more than AWGN addition. However, AWGN addition affects the decoding process more severely (very high E_{dec}), especially at lower SNR.

Effect of Gaussian blur: For the Gaussian-blur based noise channel, we observe that the geometric transformation estimation is successful ($E_A \approx 0$) for $QF_a = 99$ while for the same value of the standard deviation σ, E_A is significantly higher at $QF_a = 75$. Gaussian blur, followed by JPEG compression, smoothes the DFT matrix to such an extent that the DFT peaks often get suppressed, specially for higher values of σ, as shown in Table 5.14. For more severe blurring ($\sigma > 1$), decoding becomes a problem even if A is correctly estimated.

Table 5.14: Results with 3x3 gaussian blur, with varying σ, for $\lambda = 5$, $QF_h = 60$, $B = 80$

σ	$QF_a = 75$					$QF_a = 99$				
	E_A	E_{KP}	E_{dec}	fr_{match}	data-bits	E_A	E_{KP}	E_{dec}	fr_{match}	data-bits
0.1	16	2	2	0.2422	164.50	0	14	0	0.2368	211.70
0.2	20	2	2	0.2422	164.20	1	15	0	0.2368	211.70
0.4	23	2	3	0.2295	146.40	1	15	0	0.2371	199.20
0.6	41	1	3	0.2001	69.30	1	20	1	0.2076	113.30

Table 5.15: Results with AWGN addition at varying SNR, for $\lambda = 5$, $QF_h = 60$, $B = 80$

SNR (dB)	$QF_a = 75$					$QF_a = 99$				
	E_A	E_{KP}	E_{dec}	fr_{match}	data-bits	E_A	E_{KP}	E_{dec}	fr_{match}	data-bits
25	14	5	140	0.1796	33.15	0	20	151	0.1897	48.30
30	16	4	12	0.2023	54.50	1	18	18	0.2185	73.50
35	19	5	2	0.2239	128.40	1	20	2	0.2225	169.20
40	18	2	0	0.2302	152.40	2	13	0	0.2370	205.60

Effect of AWGN addition: For the AWGN addition based noise channel, $E_A \approx 0$ for both $QF_a = 75$ and 99, as shown in Table 5.15. Though the geometric transformation is correctly estimated, there are decoding errors due to the large noise levels (for lower SNR) resulting in a high E_{dec}. It is to be noted that geometric estimation is more accurate when the AWGN noise level is higher at $QF_a = 75$.

Effect of gamma correction: For such attacks, we observe that E_A is significantly higher when the attack is followed by JPEG compression at $QF_a = 75$, as compared to $QF_a = 99$ (Table 5.16).

Table 5.16: Results with gamma correction attack at varying γ, for $\lambda = 5$, $QF_h = 60$, $B = 80$

γ	$QF_a = 75$					$QF_a = 99$				
	E_A	E_{KP}	E_{dec}	fr_{match}	data-bits	E_A	E_{KP}	E_{dec}	fr_{match}	data-bits
0.85	20	2	2	0.2276	98.15	2	1	18	0.2305	139.10
0.95	16	1	1	0.2321	139.40	1	0	15	0.2443	188.60
1.05	17	2	1	0.2364	140.20	2	0	14	0.2355	189.50
1.15	21	3	3	0.2246	98.20	0	1	19	0.2280	142.35

Image editing software based results: We geometrically transform 50 images using 20° rotation, scaling of 1.1 and 20% cropping, before subjecting them to different attacks, performed using image editing software (Adobe PhotoShop). We compute the fraction of cases for which we can successfully estimate A, as reported in Table 5.17. From Figs. 5.16 and 5.17(b)-(d), it is seen that for local non-linear filtering attacks like pinch, twirl and shear, the DFT template peaks can no longer be observed in $\mathbf{T}_\Delta(F', \cdot, \cdot)$. Therefore, p_A is very low for these attacks. The filter parameters for the attacks are described below.

The parameters used for various Photoshop attacks are as follows:

(i) diffuse glow: graininess=5, glow amount=2, clear amount=20,

(ii) film grain: grain=1, highlight area=0, intensity=1,

(iii) pinch: the pinch factor was varied from 10%-75%,

(iv) spatter: spray radius=1, smoothness=15,

(v) twirl: the twirl angle is varied from 10°-25°,

Table 5.17: Image editing software based filtering attacks: p_A **is low for local nonlinear filter based attacks**. Also, the shear filter used here performs local nonlinear distortions, and cannot be modeled as a global 2×2 matrix.

attack	p_A	attack	p_A	attack	p_A	attack	p_A
diffuse	0.84	film-grain	0.80	**pinch**	**0.10**	spatter	0.68
twirl	**0.12**	**shear**	**0.12**	unsharp	0.92	zigzag	0.84
lens-blur	0.90	ocean-ripple	0.86	dust	0.92	offset	0.92

(vi) unsharp masking: amount=20%, radius=1, pixel threshold=0,

(vii) zigzag: amount=10, ridges=1, style is pond ripples,

(viii) lens blur: iris shape is a hexagon, iris radius=5,

(ix) ocean ripple: ripple size=2, ripple magnitude=2,

(x) dust and scratches: radius=3, threshold=0,

(xi) shear: a list of points is specified and then non-linear distortions are introduced by using splines which pass through these points,

(xii) offset: the horizontal and vertical offsets are 15 and 25, respectively.

5.9 Summary

We have presented a robust key-point based hiding method where the geometric transformation parameters are estimated using a template pattern of suitable strength that is inserted in the frequency domain. Such manipulations in the DFT

Figure 5.16: (a)-(l) images after various Photoshop attacks

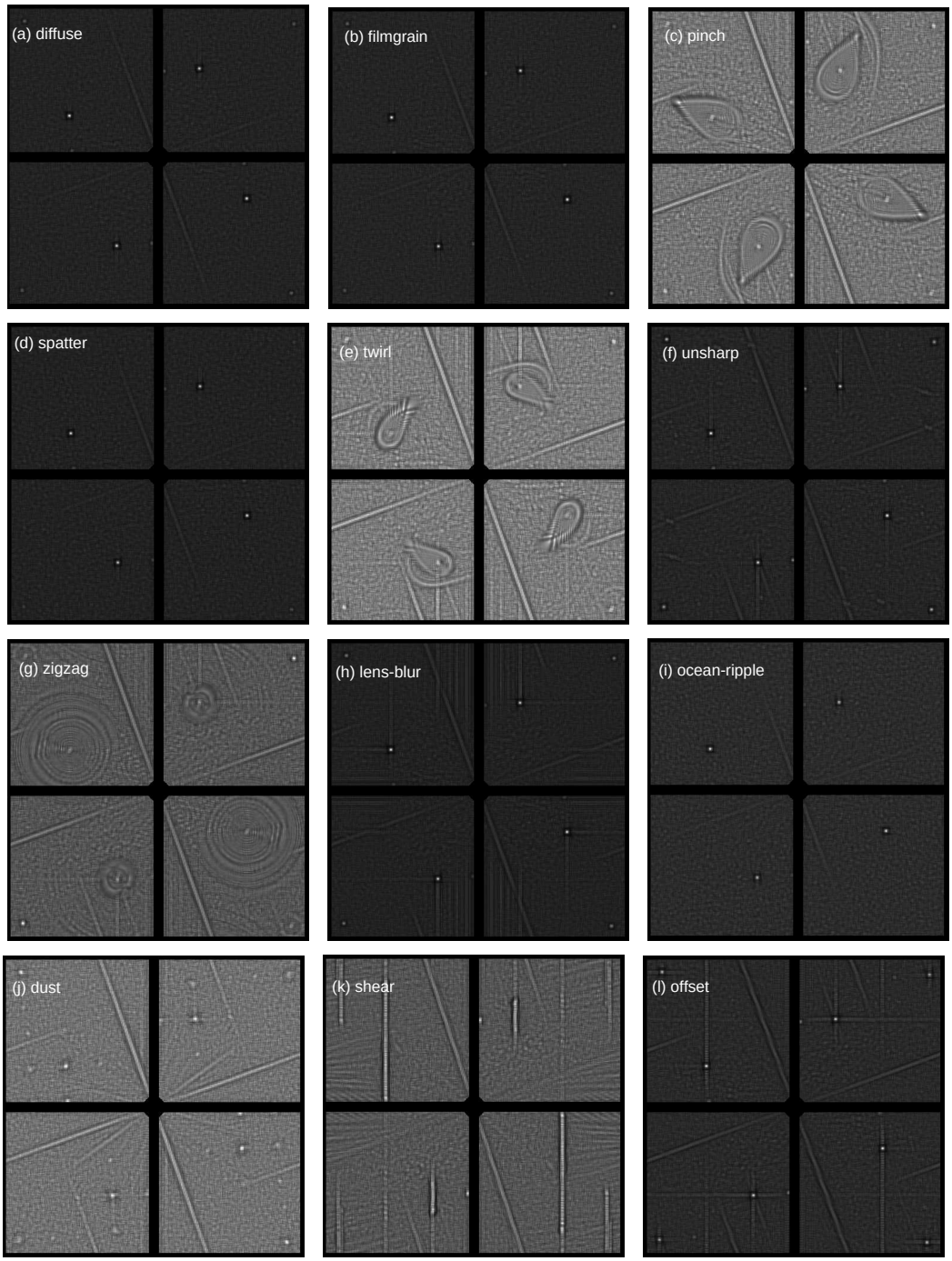


Figure 5.17: (a)-(l) corresponding DFT magnitude plots after various Photoshop attacks

domain are usually susceptible to JPEG compression and we have proposed solutions to make the template detection and geometric alignment procedure robust to JPEG. Using our key-point detection and pruning technique, the encoder and decoder return common key-points, even after pruning.

As mentioned before, masking the presence of the DFT domain peaks is of prime importance. Through the use of our proposed peak insertion and detection methods, we have obtained better detection of the template peaks without incurring additional distortions (i.e. better detection at the same PSNR).

Future Work: One possible future research is to identify a domain which further accentuates the effect of the DFT peaks so that while the peaks themselves may be perceptually undetectable in the DFT magnitude domain, they can be more clearly distinguished in the new domain. Also, for deciding on the template peak strengths, the encoder uses an iterative procedure where we also simulate the channel effects to observe the decoder performance. Another potential future research direction is to obtain a relationship between the frequency content of the image and the minimum template strength needed to ensure detection, for a given noise channel.

Chapter 6

Extensions to Image Forensics

The previous chapters have focussed on effective data-hiding and steganographic methods. In this chapter, we concentrate on another aspect of multimedia security which is authentication ("authentic" = "untampered") of the given content. This field of research, where given an image we verify whether it is real or tampered (artificially modified), is called "digital image forensics". With the widespread usage of highly advanced photo editing software, it is very simple to modify an image while still maintaining a "perceptually authentic" look. An analogy can be drawn between forensics and steganalysis. Assuming that the hiding may have taken place with a certain stego method, steganalysis solves a two-class decision problem where the detector decides whether the received image is a cover or stego. For image forensics, a forensics method decides whether or not a certain

image has been tampered using a specified image tampering method. Just as a steganalysis method is generally tuned to detect hiding by a single (or a select few) hiding method, a forensics method is generally suited to detect tampering by one (or a select few) method. Thus, to practically verify the authenticity of an image, one has to use various forensic analysis methods before coming to a final decision about its being untampered.

In this chapter, we focus on two issues in image forensics. Of the various methods used to create forgeries, one option is to copy a part of an image and paste it in another image. This copy-paste operation involves rotating, scaling or stretching the image or a part of the image to ensure the copy-paste operation does not leave any easily detectable traces. This re-sampling operation, when detected over an image window, indicates that some tampering may have occurred. Re-sampling introduces unnatural correlations between pixels due to the interpolation involved. Though there are various correlation-based approaches to detect re-sampling, they work for mainly uncompressed and mildly compressed (JPEG) images. Our first contribution is in detecting image re-sizing even under more severe compression.

We use seam carving, a newly proposed technique for content-aware image resizing, for image tampering. Our second contribution is in designing methods to detect seam carving and seam insertions. We also localize the image regions where

seam insertions have taken place. The work is extended to localize seam-carving based tampering which is associated with seam-carving based object removal.

The outline of this chapter is as follows. Common image tampering methods are described in Section 6.1. The importance of re-sampling detection for forensics purposes and the problems faced by traditional re-sampling detection methods after strong JPEG compression are presented in Section 6.2. Our solution for re-sampling detection after JPEG attacks, for the second difference method and the Expectation Maximization method, is described in Section 6.3. In Section 6.4, we propose the use of DCT-domain Markov features (previously used for steganalysis) for detecting seam carving and seam insertions in images. We also show how the relationship between pixels introduced through seam insertions can be used to localize the seam insertions. This work is further extended in Section 6.5 where the goal is the localization of the object removal, assuming that it is performed using seam carving.

6.1 Common Image Tampering Methods

Traditionally in image forensics, an "attack" is the alteration of any part of the visual content. Some methods commonly used for image tampering are as follows:

1. Re-sampling is the result of rotating, scaling or stretching an image or parts of an image. As shown in Fig. 6.1(a), tampering often involves inserting a part of an image in another image; rotating or scaling is needed to make sure the object inserted fits properly in the new image. These editing methods introduce unnatural correlation between pixels due to interpolation. Correlation-based approaches to detect such re-sampling based tampering are proposed in [85].

2. Splicing is mixing together portions of different images in a photomontage, as illustrated in Fig. 6.1(a). It is shown in [75] that this type of tampering causes changes in higher order statistics.

3. Copy-move is a very common attack (an example is in Fig. 6.1(b)). The user replaces the zone of the image to be hidden with a portion of the same image. This can be detected via block artifacts grid extraction, as proposed in [64], or by principal component analysis (PCA) on small image blocks, as shown in [86].

4. Inpainting for object removal, proposed in [2], is similar to copy-move because the source of the new information added in the image is the original image itself. The user selects the target region to be deleted and creates a hole in the image. Inpainting propagates linear structures (Fig. 6.1(c)) into

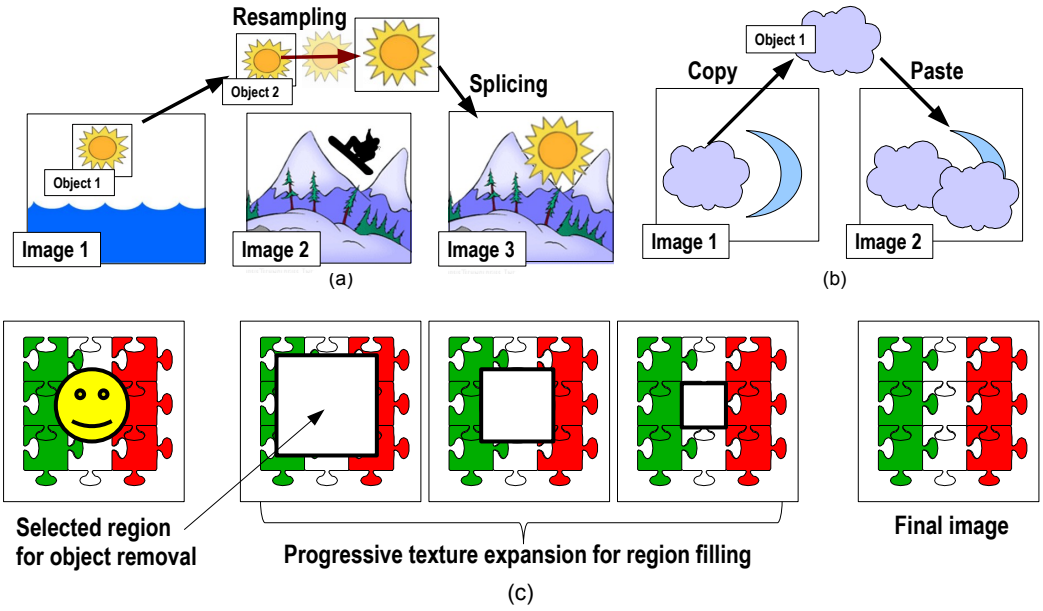


Figure 6.1: Visual illustration of methods for image tampering: (a) Object 1 is taken from Image 1. Re-sampling involves scaling the object in order to ensure that it fits properly in Image 2. Image 3 is the result of splicing Object 2 in Image 2, (b) in copy move forgery a portion of Image 1 (Object 1) is copied and pasted in the same image, (c) inpainting fills the void left from object removal with texture and structure expansion.

the target region via diffusion, and fills the hole starting from the border and progressively reaching the center. Li et al. [66] detect inpainting by analyzing the Discrete Cosine Transform (DCT) domain histograms. Wu et al. [124] use zero-connectivity feature and fuzzy membership to detect such doctoring.

Just as these methods are common tampering methods, there are various techniques commonly used as forensics tools. Apart from re-sampling detection [44, 85, 87], forensic techniques include detecting Color Filter Array (CFA)

interpolation [88], region duplication [86], JPEG compression artifacts [30], lighting changes [52], and studying variations in the camera sensor pattern noise.

6.2 Re-sampling Detection: a Forensics Tool

To motivate the importance of re-sampling detection, we repeat why re-sampling is frequently encountered in forged images. When a doctored photograph is created by digitally compositing individual images, it may be often required to re-sample (resize/rotate/stretch) the image to make it look natural.

In [85], Popescu et al. discuss how re-sampling introduces statistical correlations and describe methods to automatically detect them based on the Expectation-Maximization (EM) algorithm [24]. The EM algorithm estimates the periodic correlation (after first detecting whether such a correlation does exist) among the interpolated pixels. The specific form of the correlations indicates the exact form of the re-sampling. For images that have been resized using bilinear/bicubic interpolation, Gallagher [43, 44] has proposed techniques based on the variance of the second difference of interpolated images. This method is more robust than the correlation based method [85] though it applies only to up-sampled images. This was further improved by Mahadien et al. [71] to tackle other forms of re-sampling

using a Radon transform based approach. In [56], Kirchner presented techniques to improve upon the re-sampling detector proposed in [85].

These methods are effective for uncompressed or mildly compressed JPEG images. However, the EM-based method [85] is very susceptible to JPEG attacks, especially when the JPEG quality factor (QF) is 97 or lower. For the second difference method [44], it works for a QF higher than 80 (observed later in Fig. 6.4).

JPEG compression works on 8×8 blocks in the pixel domain and hence, there is an induced block-based periodicity. *The reason why re-sampling detection becomes difficult after JPEG compression is that the periodic JPEG blocking artifacts coincide with the periodic patterns introduced by re-sampling.*

Flexibility to use Suitable Post-processing methods for Forensics Work: *In forensics, we can perform suitable post-processing on the image without worrying about its visual quality, as the processed image is just meant for forensic analysis and not for further distribution.* Though suppression of JPEG blockiness has been well studied [62, 78] to improve the image quality, such methods have not been incorporated in forensic applications. Why is it a non-trivial problem to think of JPEG deblocking schemes that can detect re-sampling even after JPEG compression? A deblocking scheme is useful from a forensics perspective if it suppresses the JPEG patterns but still retains the re-sampling induced patterns.

We propose a novel approach to suppress JPEG artifacts by adding Gaussian noise for robust detection of image resizing.

When we add noise in the pixel domain, it affects the block-based periodicity of the entire image. After noise addition, the traces of periodicity over the entire image are reduced. The extent to which the periodicity is suppressed is controlled by the amount of noise addition. The other competing requirement remains that the re-sampling should still be detectable. When the noise added is small enough (for a certain range of the added noise), the re-sampling peaks slightly decrease in magnitude as the noise affects the inter-pixel correlation introduced by re-sampling.

In the following, we show that *there is a certain range of noise values, where for the added noise, the periodicity is affected more than the re-sampling traces.* Hence, even after JPEG compression, the re-sampling traces can be clearly distinguished. We also compare our method to median, averaging and weighted averaging based filters for suppressing JPEG induced peaks.

6.3 Detecting Resizing after JPEG Attacks

We first explain the second difference based method and then show how it is affected by JPEG compression. In [44], Gallagher has shown that the variance

(and also the mean) of the second difference of the interpolated signal has the same periodicity as the sampling rate of the original signal.

The steps involved in the second difference algorithm are as follows:

(i) Consider the image on a window-by-window basis. Since it is more likely that a certain portion of the image has been re-sampled (instead of the entire image), the re-sampling detector is run on a per-window basis.

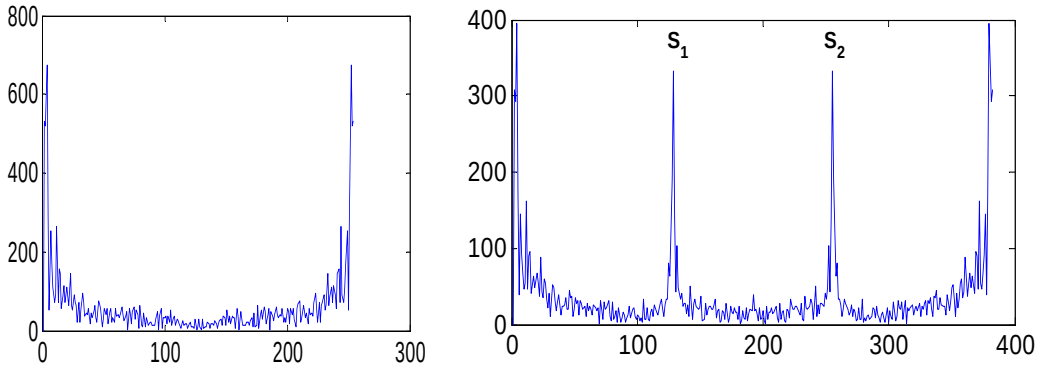
(ii) Compute the second derivative for every row.

(iii) Average the second derivative across all columns to obtain a 1-D signal.

(iv) If the above signal is periodic, then the image is interpolated. Hence, peaks in the 1D-DFT indicate tampering.

Figs. 6.2(a) and 6.2(b) show the DFT for the second difference signal for an original (uncompressed and not re-sampled) image and re-sampled (also uncompressed) image, respectively.

Referring to Fig. 6.2(b), we show how the locations of the sampling peaks correspond to the sampling factor used for image re-sampling. For example, the size of the DFT used in this figure is 378. The locations of the sampling peaks S_1 and S_2 are 128 and 255. The two possible sampling values that can lead to peaks at these locations are 378/128 (=3) and 378/255 (= 1.5). Thus, considering

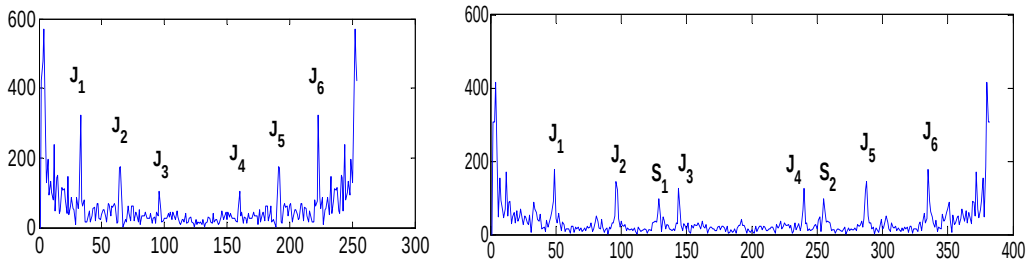


(a) (b)

Figure 6.2: The DFT for the 1-D signal obtained by taking the mean of the second difference computed per row is shown here. X-axis shows the DFT indices. (a) DFT for the original TIFF image, without re-sampling, does not show any peaks as the second difference signal does not show any periodicity. (b) DFT for a re-sampled image, re-sampling factor being 1.5, shows peaks at locations S_1 and S_2. The DFT size is changed depending on the image size. E.g., the original image was 256×256 and so the DFT size used in Fig. (a) was 256. After re-sampling by 1.5, the new image is 378×378. The DFT size used in Fig. (b) is 378. It should be noted that if we maintained a suitably larger value for DFT size (to prevent aliasing), we would still see the re-sampling peaks. Here, for ease of illustration, we maintain the DFT length as equal to the image dimension.

the DFT plot on a normalized frequency grid, the peaks are observed at integer multiples of $1/\hat{f}$, when the periodicity equals $\hat{f}$, for bilinear interpolation.

We now motivate how JPEG compression makes the sampling detection very difficult through Fig. 6.3. For an original (JPEG compressed but no re-sampled) image, the DFT of the second difference shows JPEG-induced peaks $J_1, \ldots, J_6$, in Fig. 6.3(a). For a re-sampled JPEG image, there are two sets of DFT peaks - sampling peaks S_1 and S_2, and JPEG induced peaks $J_1, \ldots, J_6$.



(a) (b)

Figure 6.3: The DFT for the mean of the second difference signal is shown for the following images: (a) DFT for the original JPEG image, without re-sampling, shows peaks $J_1, \ldots, J_6$ at an interval of $\frac{1}{8}\times$(DFT length) (b) DFT for a re-sampled image, re-sampling factor being 1.5, shows peaks at locations S_1 and S_2.

Thus, the main problem for detecting re-sampling even after JPEG compression is to suppress the JPEG peaks $J_1, \ldots, J_6$ while retaining S_1 and S_2.

Here, we have considered the performance of this second difference method for varying levels of JPEG compression. For all the results shown for this method (Fig. 6.4-6.8), the results have been averaged across 50 images.

6.3.1 Effects of JPEG Compression

In Fig. 6.4, we resize the image by a factor of 3 using bilinear interpolation, followed by JPEG compression. JPEG introduces periodicity by a factor of 8.

One approach is to zero out the JPEG peaks (since their locations are known). However, when the resize factor is a multiple of 4, some sampling peaks will coincide with some JPEG induced peaks. Hence, mere zeroing out of JPEG peaks using knowledge of the JPEG peak locations will not help in re-sampling

detection in all cases. We suggest methods to suppress the JPEG peaks while retaining the sampling peaks.

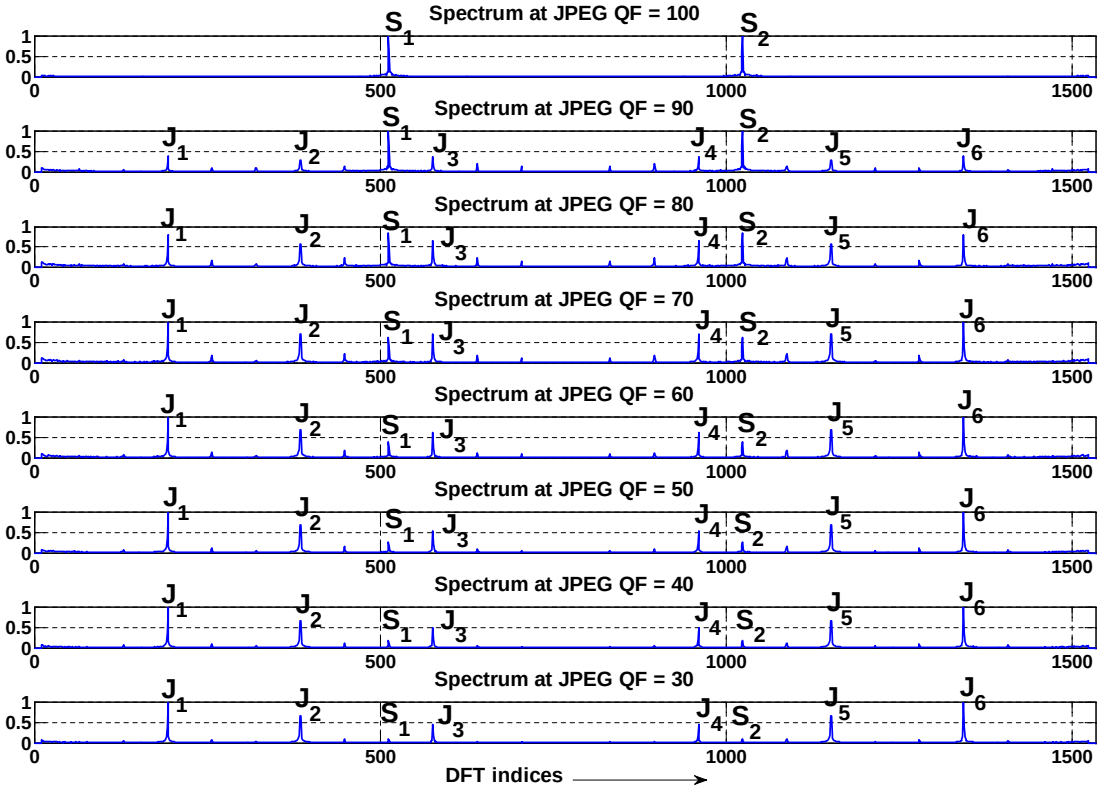


Figure 6.4: JPEG compression is performed (after bilinear interpolation by a factor of 3) at different QF, from 30-100. The size of DFT used is 1500 while the sampling peaks S_1 and S_2 lie at 500 and 1000. These sampling peaks can occur due to sampling factors of 1.5 and 3. $\frac{1}{\text{sampling factor of 3}} = \frac{500(\text{Location of } S_1)}{1500(\text{DFT size})}$. Also, $\frac{1}{\text{sampling factor of 1.5}} = \frac{1000(\text{Location of } S_1)}{1500(\text{DFT size})}$

On progressively lowering the QF (more severe compression), the JPEG peaks increase in magnitude relative to the sampling peaks. Closer to a QF of 70-80, the peaks due to re-sampling and JPEG compression are almost of the same

magnitude. For lower QF, the JPEG peaks dominate over the sampling peaks and hence, the DFT magnitude plots reveal only a periodicity of 8, due to JPEG.

6.3.2 Masking JPEG Peaks using AWGN

We add AWGN (Additive White Gaussian Noise) to a re-sampled image and observe its effect on the JPEG peaks and sampling peaks. We find that with increasing noise levels (Fig. 6.5), the magnitude of the JPEG peaks is more significantly reduced than that of the sampling peaks. Thus, AWGN suppresses the JPEG induced periodicity. At a SNR of 20 dB, the JPEG peaks are masked but the sampling peaks are still visible. However, excessive noise addition also suppresses the sampling peaks, as observed for 15 dB SNR.

We find suitable SNR ranges for the added AWGN for a host of JPEG compression factors ($40 \leq QF \leq 80$) as shown in Table 6.1.

(i) For noise values less than 15 dB, both sampling and JPEG induced peaks are suppressed (Fig. 6.5).

(ii) For noise values greater than 25 dB, the topmost JPEG peaks J_1 and J_2 are almost equal in magnitude to the sampling peaks (Fig. 6.5).

(iii) For 20 dB noise, S_1 and S_2 are significantly higher than the JPEG induced peaks.

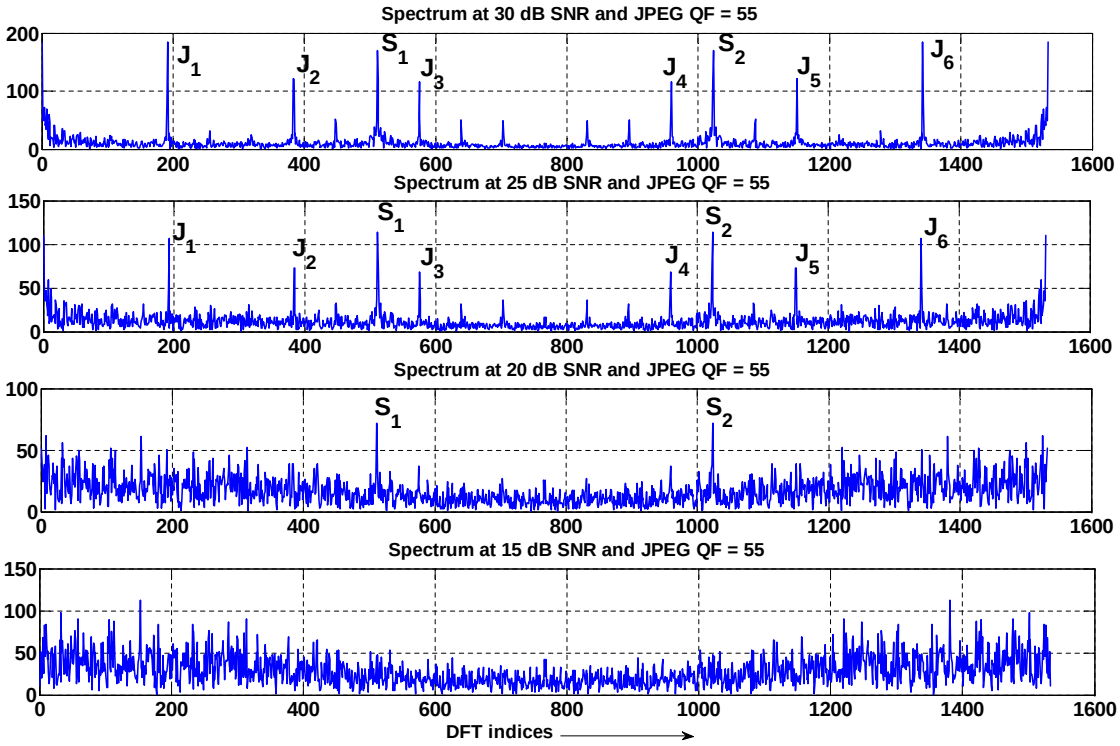


Figure 6.5: By adding suitable amount of Gaussian noise, the peaks due to JPEG compression are suppressed while still retaining the sampling peaks - as seen at 20 dB SNR. Below 20 dB SNR, both the JPEG and sampling peaks are suppressed.

It is seen that the amount of noise added needs to be higher (lower SNR $\Rightarrow$ more noise is added) for more severe JPEG compression. We have also experimentally observed that the suitable amount of noise does not depend on the re-sampling factor but mainly on the amount of JPEG compression.

We consider other filters (median filter, 3×3 and 5×5 weighted average filter) to "denoise" JPEG images, for different JPEG QF (Fig. 6.6-6.8). In Fig. 6.6, we observe that the sampling peaks can be better distinguished for the 3×3 and 5×5

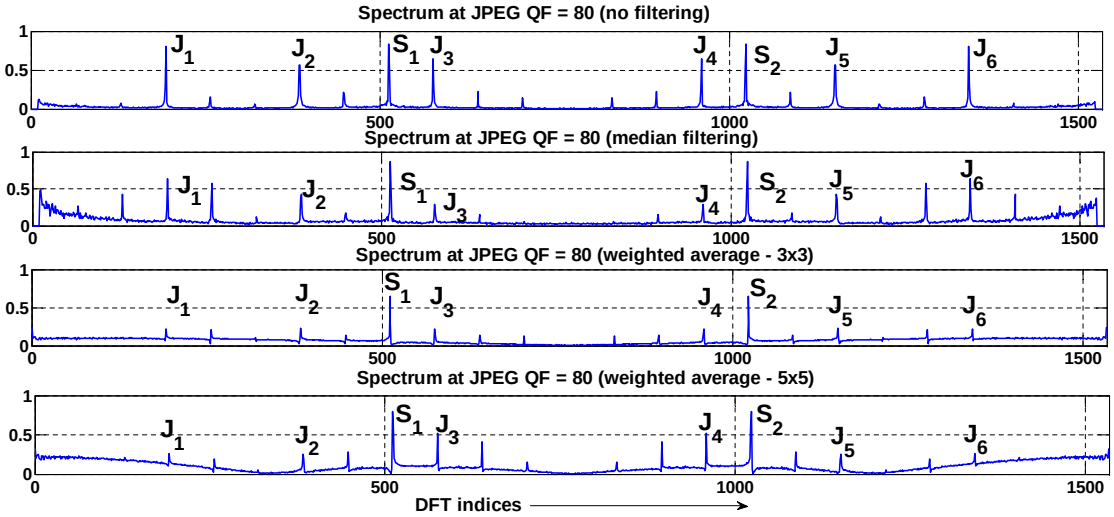


Figure 6.6: Filtering the JPEG compressed image, at QF of 80

Table 6.1: Strength of AWGN to add for a given JPEG QF

JPEG QF	SNR Range (dB)	JPEG QF	SNR Range (dB)
80	[20-50]	75	[20-50]
70	[20-40]	65	[20-30]
60	[20-30]	55	[20-25]
50	20	40	20

weighted average filters. In Fig. 6.7, S_1 and S_2 can be much better distinguished using 3×3 than 5×5 filter. In Fig. 6.8, for 3×3 filter, S_1 and S_2 are almost equal to J_1 and J_6, in magnitude. However, adding AWGN "denoises" JPEG better than all these filters.

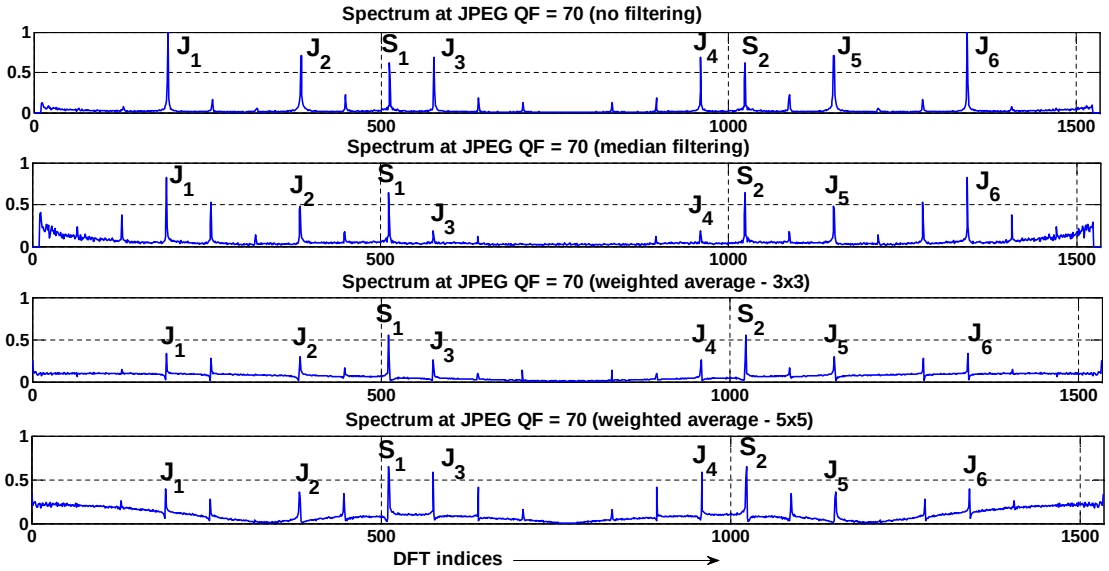


Figure 6.7: Filtering the JPEG compressed image, at QF of 70

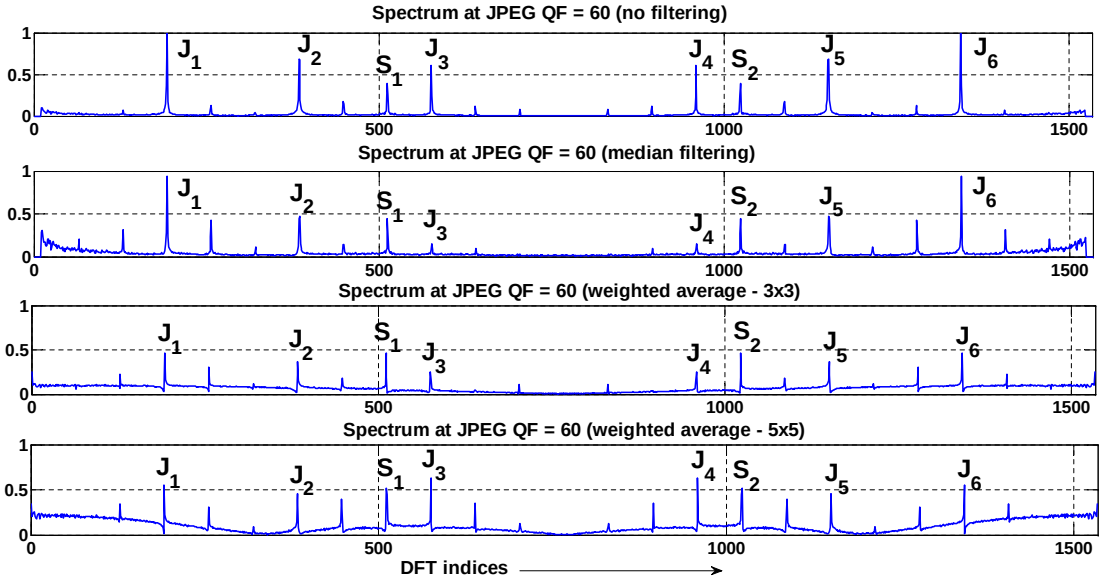


Figure 6.8: Filtering the JPEG compressed image, at QF of 60

6.3.3 Effect on the EM-based Method

In Popescu et al.'s Expectation Maximization (EM)-based algorithm [85], each re-sampled pixel is assumed to be a linear combination of its neighbors. Each pixel's probability of being a linear combination of its neighbors is then estimated using the EM-based learned weights, computed over a certain window. This matrix of probability values, called the probability map (p-map), exhibits a periodic pattern for re-sampled images.

Hence, peaks are seen in the DFT plot of the p-map only for re-sampled images, making the peaks an indicator of re-sampling. In our case, we use a 1×5 window across a row and repeat the same for all rows. In Fig. 6.9, we consider the p-map for an image re-sampled by a factor of 3. We see that the 3 peaks clearly visible in (a) get smeared due to JPEG (b). By adding AWGN, we are able to retrieve the periodicity as shown in (c) and (d).

In Fig. 6.10, we show the p-map which is used to detect rotation. In this example, the image is rotated by 5° and the peaks in the DFT of the p-map show peaks oriented at an angle of 5° from the x (or y) axis in the DFT grid (Fig. (b)). In Fig. (c), we see that after JPEG compression at QF of 75, the peaks can no longer be observed. In Fig. (d), we observe that after adding a suitable amount of noise, the sampling peaks can again be observed.

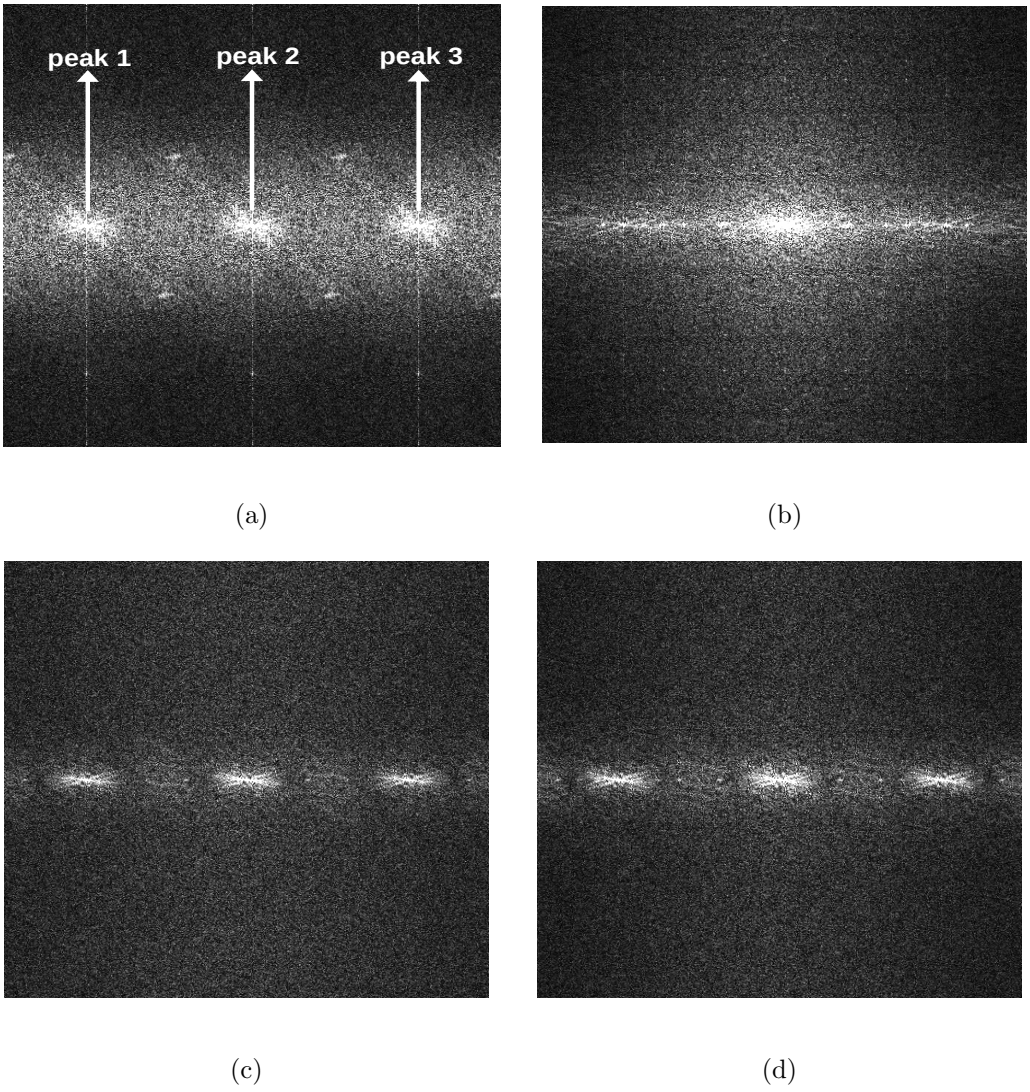
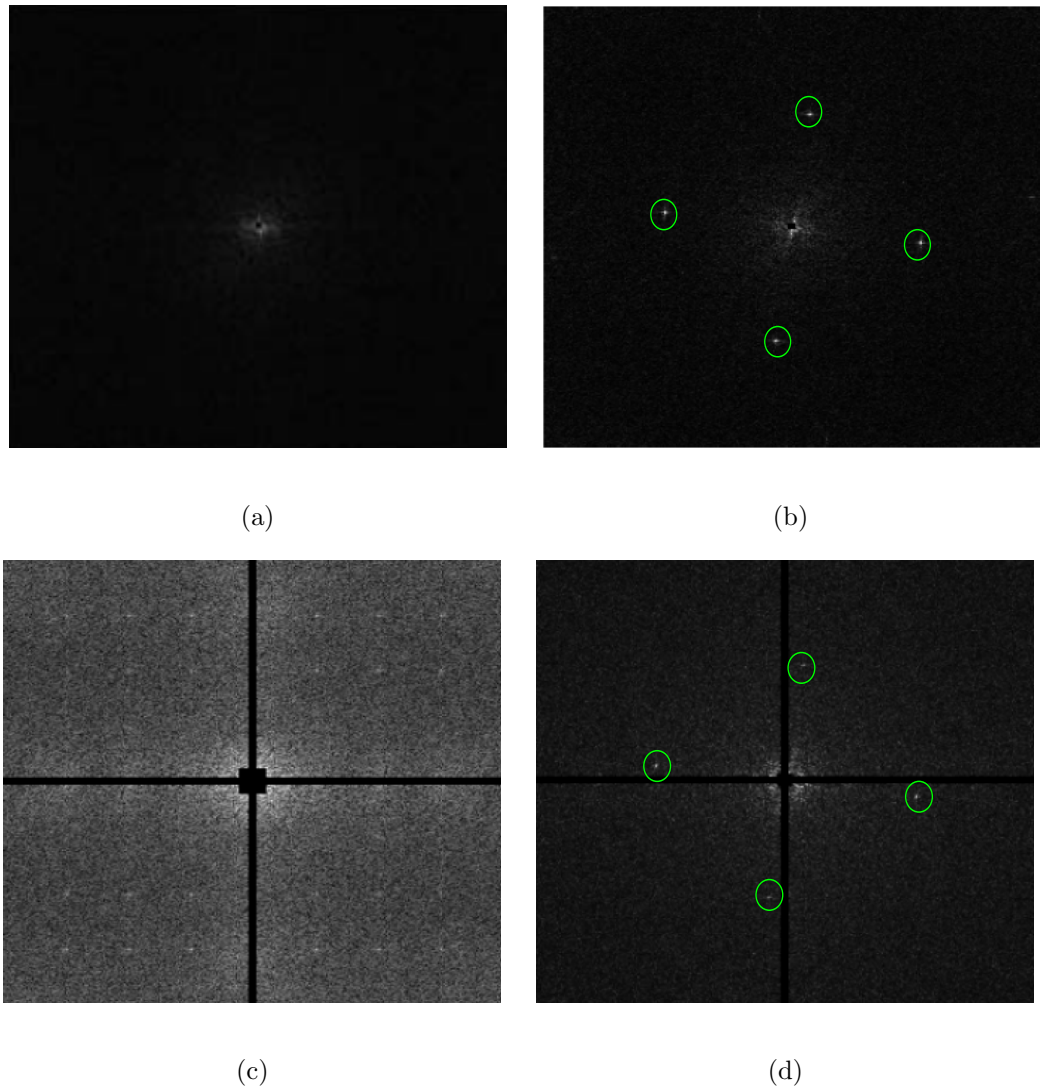


Figure 6.9: DFT of the p-map after (a) resizing the image by a factor of 3: note 3 peaks (b) JPEG on the resized image at a QF of 85: no distinct peaks can be observed (c) adding AWGN on JPEG image at 35 dB SNR, (d) 40 dB SNR: 3 peaks can again be seen in (c) and (d).



(a) (b)

(c) (d)

Figure 6.10: DFT of the p-map for the (a) original image (b) image rotated by 5° (c) rotated image is JPEG compressed at QF = 75 (d) AWGN is then added to the rotated JPEG image at 40 dB SNR

6.3.4 Summary

Various resizing detection algorithms fail after JPEG compression since it introduces significant peaks in the frequency domain that interfere with the periodicity introduced by resizing. We have shown that adding Gaussian noise is an effective "denoising" technique to mask the effects of JPEG. By varying the strength of the Gaussian noise based on the JPEG quality factor, we can suppress the JPEG peaks while still retaining the sampling peaks. Though we have focussed solely on image resizing detection, the proposed "JPEG induced peak suppression" can be applied to other methods which fail due to JPEG attacks and where the visual quality of the final image is not of interest.

Future Work: We have shown that there is a range of noise values for which this re-sampling detection method works after JPEG compression. Going beyond feasibility experiments, theoretical studies can be performed as to why re-sampling is affected less while JPEG compression based periodicity is affected more by the noise addition. In the absence of a formal proof, we present an intuitive explanation. When a signal is added to a periodic signal (JPEG compressed image shows an induced periodicity of 8), the periodic effects are reduced if that signal does not have the same periodicity; hence, the noise addition reduces the periodicity. The re-sampling effects are manifested in the relationship between nearby pixels (e.g., for bilinear interpolation, each pixel is a weighted combination of four neighbor-

ing pixels); the noise addition offsets this relationship and hence, the re-sampling peaks also decrease. It is suggested through the experimental results that for lower noise values, the inter-pixel correlation and hence, the sampling peaks are slightly affected while the periodicity based JPEG induced peaks are more significantly affected. Still, a more quantitative analysis needs to be pursued in the future. The final goal is to obtain the "best noise distribution and noise level" for a given image and a known level of JPEG compression to achieve "best possible" JPEG peak suppression while still retaining the re-sampling peaks.

6.4 Detection of Seam Carving and Localization of Seam Insertions

"Seam carving" is a recently introduced content aware image resizing algorithm [9]. This method can also be used for image tampering. In this section, we explore techniques to detect seam carving (or seam insertion) without assuming any knowledge of the original image. A machine learning based framework is employed to distinguish between seam-carved (or seam-inserted) and normal images. It is seen that the 324-dimensional Markov feature, consisting of 2D difference histograms in the block-based Discrete Cosine Transform domain [103], is well-suited for the classification task. The feature yields a detection accuracy

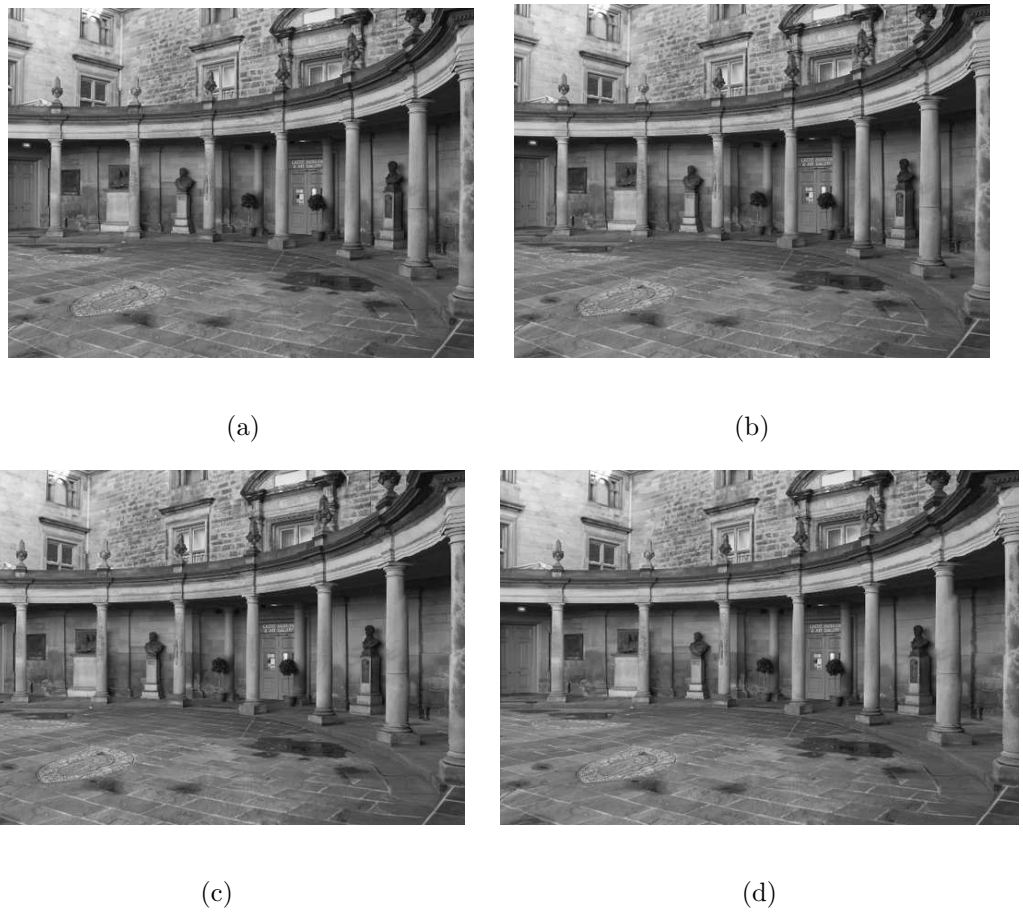
of 80% and 85% for seam carving and seam insertion, respectively. For seam insertion, each new pixel that is introduced is a linear combination of its neighboring pixels. We detect seam insertions based on this linear relation, with a high detection accuracy of 94% even for very low seam insertion rates. By exploiting the relationship between newly inserted pixels lying along a seam, we are able to accurately localize seam insertions. We also show empirically that the Markov feature is useful for scaling and rotation detection.

6.4.1 Seam Carving for Image Tampering

We give a brief overview of seam carving and then motivate how seam carving can be thought of as an image tampering technique.

In Fig. 6.11, we show how the image width is increased in Fig. (b), (c) and (d). The width is increased by adding more "seams" to the image. Though we explain seams more formally later, from the figures (Fig. 6.12), one can guess that a seam is a set of 8-connected pixels that traverses the entire image. We consider "vertical" seams in our experiments and the seams (curved red lines in Fig. 6.12(b)-(d)) traverse the entire image top-to-bottom.

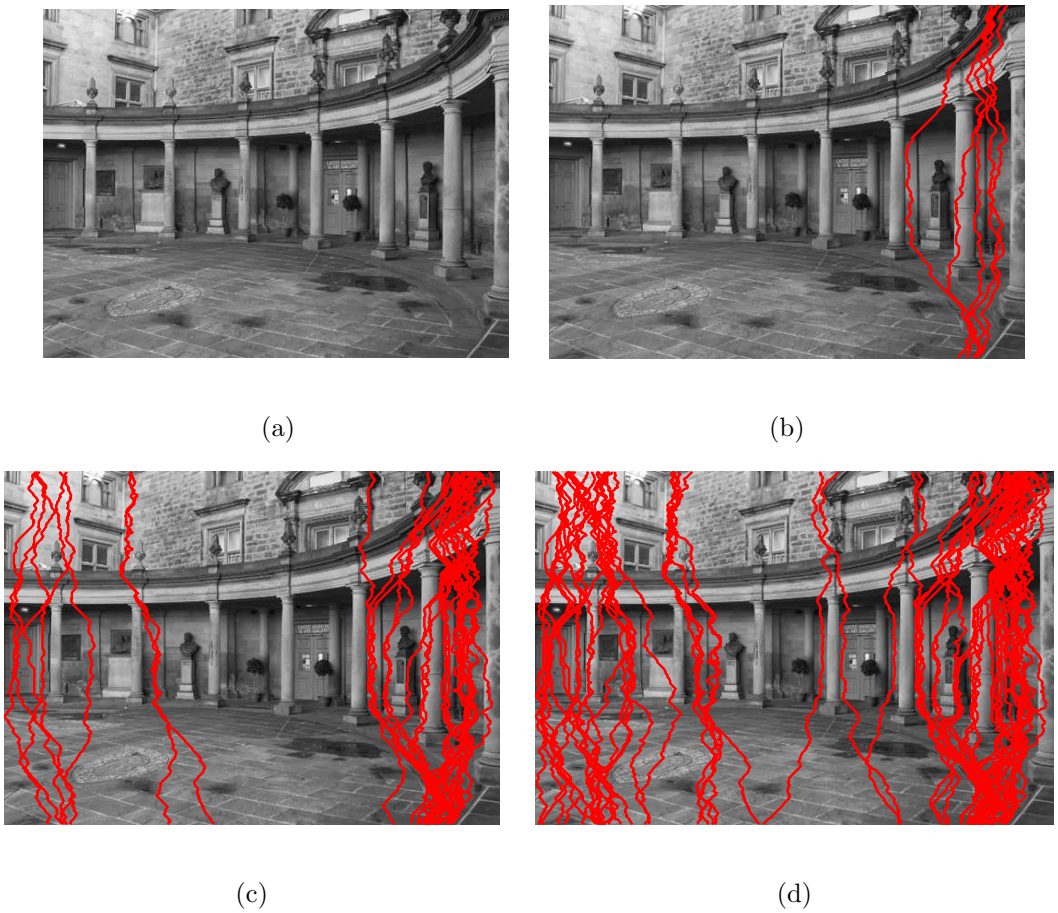
In the "content aware image resizing" technique [9], the "important content" in an image is left unaffected when the image is resized. It is assumed that the "important content" is not characterized by the low energy pixels. For resizing, a



(a) (b)

(c) (d)

Figure 6.11: (a) original image, (b), (c) and (d) are modifications of (a) with the same number of rows and 1%, 5% and 10% more columns, respectively.



(a) (b)

(c) (d)

Figure 6.12: It is similar to Fig. 6.11 except that the seams are also clearly shown for (b), (c) and (d) to explain how the image has been modified to increase its width.

series of optimal 8-connected paths of pixels (each optimal path is called a "seam") are identified which traverse the entire image vertically or horizontally. The optimality criterion is related to an energy function computed for all points along the seam - the underlying theory is briefly explained in Section 6.4.2. The optimal choice of seams maintains the image quality in this resizing process. *Since the image content and/or its dimensions are changed, we treat the seam carved/inserted image as a tampered image.* Hence, we consider the problem of detecting seam carving/insertion, which has not been considered by current forensic techniques. Interpolation kernel based methods for re-sampling detection will fail when the resizing in the doctored image is done using seam carving/insertion. Another possible scenario of using seam carving for image tampering is object removal as shown in [9] - this provides another forensic application scenario in the context of seam carving detection.

Fig. 6.13 shows the steps involved in object removal. Fig. 6.13(b) shows a black mask which shows the object (or region) which is to be removed. Since the seams pass through the low energy regions and the energy is quantified through the gradient per pixel, the gradient corresponding to the pixels contained in the mask are assigned very low (negative) values. This ensures that during seam carving, the seams will pass through the mask. Figs. 6.13(c) and (d) show the 1^{st} and 45^{th} seams that are respectively removed. The highlighted region after all the seams

that pass through the masked region have been removed is shown in Fig. 6.13(e). If the image size is to be retained, then as many seams need to be inserted in the image as were originally removed. This is shown through Figs. 6.13(f) and (g). Fig. 6.13(h) shows the final image, having the same size as the original, after seam carving (which accounts for object removal) and seam insertion. Thus, seam insertion detection suggests that operations like object removal may have also taken place.

Framework for Seam-carving Detection; We adopt a machine learning framework for the two-class (for example, seam carved and original images) classification problem. As candidate features, we have experimented with natural image statistics based features that are extensively used in steganalysis. For steganalysis, the successful features correspond to consistent changes in the image, that occur due to the same hiding method being used for all images. Similarly, to detect seam-carving, *the key is to identify common statistical changes* - even though the seam locations depend on the image content. The Markov features which depend on first order differences in the quantized Discrete Cosine Transform (DCT) domain - the 324-dimensional feature by Shi et al. [103] which is subsequently referred to as **Shi-324**, is seen to perform better than other features experimented with for the detection task.

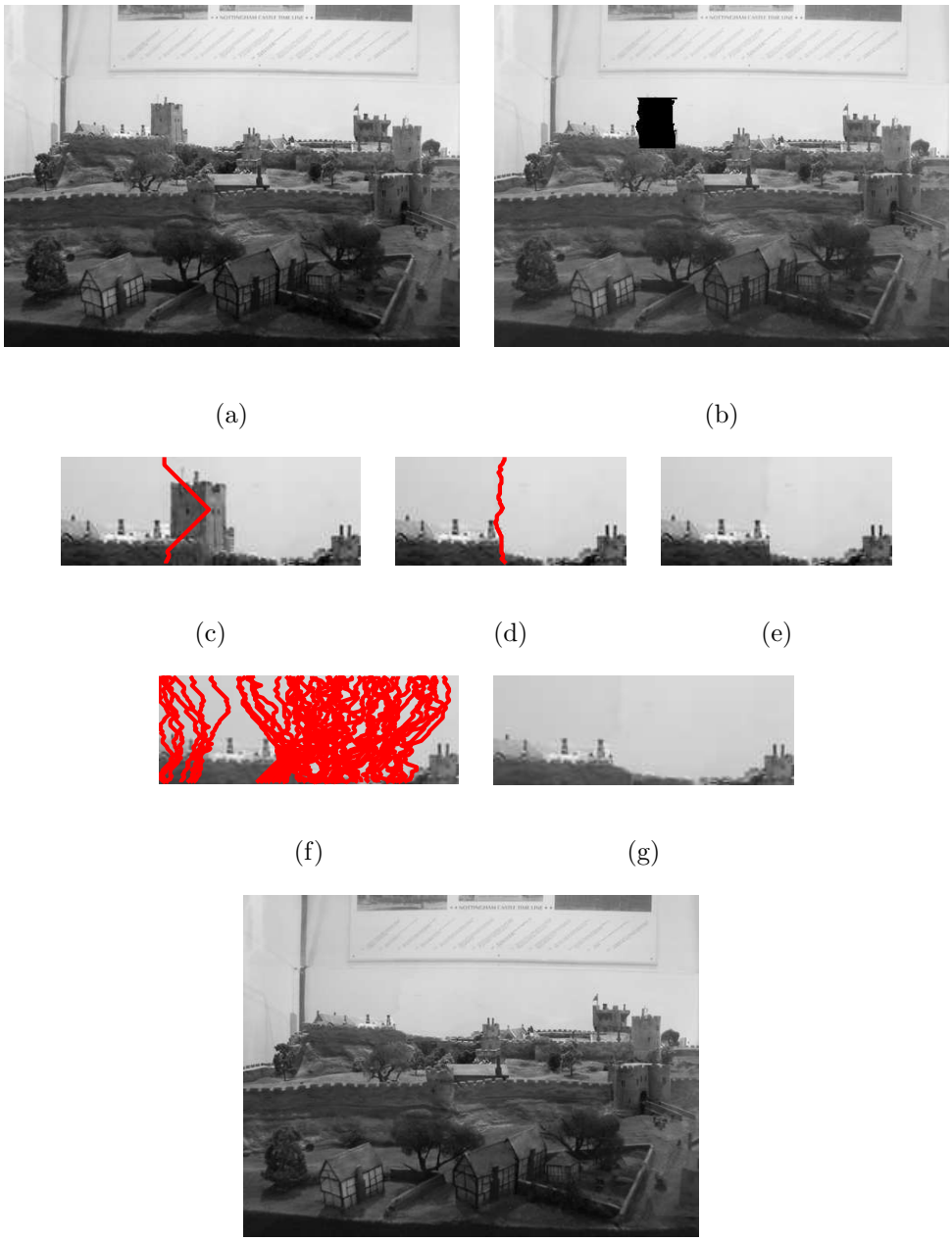


Figure 6.13: Steps (a)-(h) show how the object removal is done using seam carving and seam insertion: (a) original image, (b) the mask for object removal is shown in black, (h) the image after object removal and having the same size as the original is shown.

The intuition behind using the Markov features is briefly explained. The pixel neighbors change upon seam removal for the pixels bordering the seams. When a sufficient number of seams are removed, there is a significant change in adjoining pixel statistics (pixel domain Markov features were discussed in [111]) as well as in local block-based frequency domain statistics. We have experimented with first order difference based 2D histograms in both pixel and frequency (quantized DCT) domains - it is seen that the change in DCT domain is more consistent for proper classification. The intra-class variability in the pixel domain Markov feature is high enough to be useful for classification.

We use a Support Vector Machine (SVM) based model to train the features. As the number of seams deleted (or inserted) increases, the detection accuracy also increases. We also detect and localize seam insertions based on the linear relationship between the new pixels, introduced by seam insertion. Experiments with **Shi-324** feature show that it can also detect geometric transformations, with SVM models being trained for specific transformation parameters.

Contributions: Our contributions to seam-carving based tamper detection can be summarized as follows:

- using Shi-324 [103], a Markov feature successfully used for JPEG steganalysis, for detecting seam carving and seam insertion, using SVM for learning the

appropriate model,

- using the linear relationship between the new pixels introduced during seam insertion to localize the seam insertions,
- demonstrating that Shi-324 is also effective for rotation and scaling detection.

The rest of this section is organized as follows: Section 6.4.2 briefly describes the theory behind seam carving and seam insertion. In Section 6.4.3, we describe the Shi-324 feature, based mainly on [103], where it was originally proposed. To reiterate, this feature was previously used in the context of steganalysis, in Chapters 3 and 4. The experimental results for seam carving/insertion detection using Shi-324 are presented in Section 6.4.4. Detecting and localizing the seam insertions exploiting the linear relationship between seam-inserted pixels are shown in Section 6.4.5. The use of a probability-map based feature to detect seam-insertions, based on Popescu et al.'s EM-based algorithm (this was briefly discussed in the context of re-sampling detection in Section 6.3.3) [85], is presented in Section 6.4.6. The use of Shi-324 for rotation and scaling detection is shown in Section 6.4.7.

6.4.2 Seam Carving and Seam Insertion

Seam carving was introduced in [9] for automatic content-based image resizing. As more seams are removed, the image quality degrades gracefully. Two commonly

used methods for changing the image size are scaling and cropping. For cropping, it cannot be detected using statistical image features as it does not modify the image pixels which are retained. Cropping can however only remove image pixels from the sides or the image ends. If there is some redundant content in the image center and useful image content at the ends, cropping cannot be used. Scaling is also oblivious to the image content and is generally applied uniformly to the entire image instead of an image sub-part. Since scaling is generally performed using pixel interpolation over a certain window, it introduces some correlation between the neighboring pixels (the correlation depends on the scaling factor and the interpolation method used) which can be exploited to detect scaling [44,85,87].

Seam carving uses an energy function based on the energy of pixels that lie along a certain path, called the seam. By successively removing or inserting seams, one can reduce, or enlarge, the image size in both directions. For image reduction, seam selection ensures that mainly the low energy pixels are removed which help to preserve the image structure. The low energy pixels generally correspond to the low-frequency (smooth) image regions where minor changes are difficult to detect perceptually.

Here, we have considered mainly the deletion and insertion of vertical seams (therefore, "seam" refers to a vertical seam in this section). Seam-carving or seam-

insertion fraction refers to the number of deleted or inserted seams, expressed as a fraction of the number of columns in the original image.

Table 6.2: Table of Notations

Notation	Definition
$\mathbf{I}$	$N \times M$ image intensity matrix
$I(a, b)$	pixel corresponding to the a^{th} row and b^{th} column in the image I
$\mathbf{s}$	the set of pixels constituting a certain seam, for example, if there are N pixels in a seam (as happens for a vertical seam in an image with N rows), with locations given by $\{a_i, b_i\}_{i=1}^{N}$, then $\mathbf{s} = \{a_i, b_i\}_{i=1}^{N}$

To maintain perceptual transparency, the pixels to be removed should blend very well with their surroundings, as is the case for smooth low-frequency regions. The cost function $e_1(\cdot)$ (6.1) associated with a pixel (other cost functions, for example, using the Histogram of Gradients, have been suggested in [9]) is the sum of the gradients.

$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}| \tag{6.1}$$

The following part is based on the discussion in [9] where it is explained why the "optimal" seam corresponds to the minimum energy path and why a seam consists of D_8 connected pixels. An optimal method to remove the "unnoticeable" pixels would be to remove the pixels with the lowest energy values, after arranging the pixels in ascending order. For a $N_1 \times N_2$ image, if we remove the first N_1 pixels (N_1 pixels constitute a column) having the lowest energy, we may end up removing

a different number of pixels from each row. To retain the rectangular structure of the image, the removed seam of N_1 pixels should have one pixel from each row. If these N_1 pixels are not connected with each other (vertical or diagonal neighbors), it would greatly distort the image content by introducing a zigzag effect. *An easy solution would be to remove the seam having the least overall energy (computed over the N_1 pixels in the seam).* In the proposed cost function, the pixels lying along a seam are chosen so that they constitute a connected path from top-to-bottom (6.2), one pixel is removed per row and the seam removed corresponds to the lowest energy path (6.3). Fig. 6.14 shows how seam carving takes place for a 4×5 matrix **a** to return a 4×4 matrix **b**. Elements to the left of the seam remain unchanged while those to the right are shifted by one pixel to the left.

For a $N_1 \times N_2$ image $\mathbf{I}$, a vertical seam is defined as:

$$\mathbf{s}^x = \{s_i^x\}_{i=1}^{N_1} = \{i, x(i)\}_{i=1}^{N_1}, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1 \tag{6.2}$$

where $x(i)$ maps the i^{th} pixel in the seam to one of the N_2 columns.

The pixels in a seam $\mathbf{s}$ will be $\mathbf{I_s} = \{\mathbf{I}(s_i)\}_{i=1}^{N_1} = \{\mathbf{I}(i, x(i))\}_{i=1}^{N_1}$. After removing a vertical seam, the adjoining pixels in each row are moved left or right to compensate for the removed pixels.

The optimal seam s^* is defined as follows:

$$s^* = \min_{\mathbf{s}}\{E(\mathbf{s})\} = \min_{\mathbf{s},\ \mathbf{s}=\{\mathbf{s_i}\}_{\mathbf{i=1}}^{\mathbf{N_1}}} \{\sum_{i=1}^{N_1} e_1(\mathbf{I}(s_i))\} \tag{6.3}$$

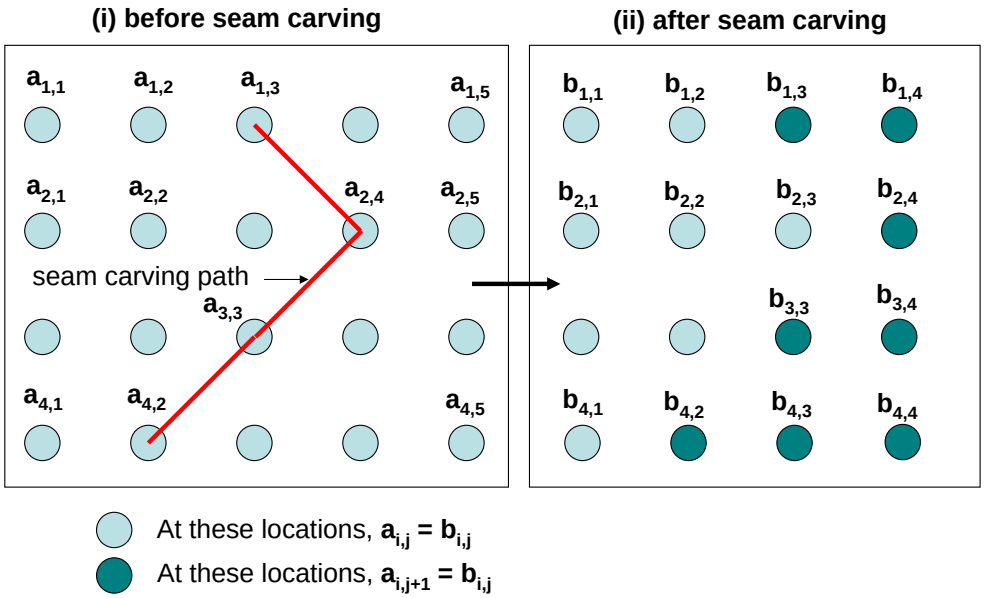


Figure 6.14: Example of seam carving for a 4x5 matrix **a**

The optimal seam is computed using dynamic programming (6.4). The image is traversed from the second row to the last row and the cumulative minimum energy M is computed for all possible connected seams for a given row (i) and column index (j).

$$M(i,j) = e_1(i,j) \; + \; \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1)) \qquad (6.4)$$

The minimum value of the last row in M indicates the ending location of the optimal vertical seam. We then back-track from this minimum entry to find the other points in the optimal seam.

The seam selection process is identical for seam carving and seam insertion. For seam insertion, for every selected seam, the corresponding pixel is removed

and is replaced by two pixels, whose values are computed as in (6.5). For example, consider three pixels $\{a, b, c\}$, which are consecutive pixels on the same row. The selected seam passes through b. After seam-insertion, $\{a, b, c\}$ is replaced by $\{a, b_1, b_2, c\}$, where the values of the new pixels (b_1 and b_2) are:

$$\text{Seam insertion: } b_1 = \text{ round}(\frac{a+b}{2}),\ b_2 = \text{ round}(\frac{b+c}{2}) \tag{6.5}$$

When the selected seam lies along the border, the pixel lying on the seam is retained and only one new pixel value is introduced. For example, when $\{a, b\}$ is replaced by $\{a, b_1, b\}$ after seam-insertion, with a (or b) being the border pixel through which the seam passes, then the new pixel value introduced, b_1, equals round($\frac{a+b}{2}$).

6.4.3 Markov Feature to Detect Seam Carving and Insertion

Seam-carving does not introduce new pixel values in the image. However, the pixels next to the seam change and hence, when sufficient seams are removed, the neighborhood change can be quite significant. This change can show up in features like inter-pixel correlation and co-occurrence matrix in the pixel and frequency domains. Local block-based DCT coefficients are also expected to reflect the change. Even if we remove only vertical seams, the neighborhood change can

affect horizontal, vertical and diagonal neighbors. For seam insertion, new pixel values are introduced (6.5) and the pixel neighborhood also changes for the pixels lying on/near the seams - hence, similar features may be effective for seam-carving and seam insertion detection.

We briefly explain the **Shi-324** feature. It assumes a JPEG image as input. Following the notation in [103], the JPEG 2D array (set of 8×8 quantized DCT coefficients, where the DCT is computed for every 8×8 image block) obtained from a given image is denoted by $F(u,v),\ u \in [0, S_u - 1]$ and $v \in [0, S_v - 1]$, where S_u and S_v denote the size of the JPEG 2D array along the horizontal and vertical directions. The first order difference arrays are expressed as:

$$\begin{aligned} \text{horizontal: } F_h(u,v) &= F(u,v) - F(u+1,v), \\ \text{vertical: } F_v(u,v) &= F(u,v) - F(u,v+1), \\ \text{diagonal: } F_d(u,v) &= F(u,v) - F(u+1,v+1), \\ \text{minor diagonal: } F_m(u,v) &= F(u+1,v) - F(u,v+1) \end{aligned} \tag{6.6}$$

where $F_h(u,v), F_v(u,v), F_d(u,v)$ and $F_m(u,v)$ denote the difference arrays in the horizontal, vertical, main diagonal and minor diagonal directions, respectively.

Since the distribution of the elements in the difference 2D arrays is similar to a Laplacian, with a highly peaky nature near 0, the difference values are considered in the range $[-T, T]$. When difference values are greater than T or less than $-T$,

they are mapped to T and $-T$, respectively. In [103], $T=4$ is used; hence, the number of histogram bins equals $(2T+1)^2=81$ along each direction.

Each of the difference 2-D arrays is modeled using Markov random process - a transition probability matrix is used to represent the Markov process. Each of the probability matrices (p_h, p_v, p_d and p_m) (6.7) used to represent the 2-D difference arrays (F_h, F_v, F_d and F_m, respectively) have $(2T+1)^2$ bins. Therefore, the total feature vector size for **Shi-324** is $81 \times 4 = 324$ (after converting each probability matrix to an 81-dim vector and concatenating the 4 vectors). The elements of these 4 matrices are given by:

$$\begin{aligned} p_h(m,n) &= \frac{\sum_{u,v} \delta(F_h(u,v)=m, F_h(u+1,v)=n)}{\sum_{u,v} \delta(F_h(u,v)=m)} \\ p_v(m,n) &= \frac{\sum_{u,v} \delta(F_v(u,v)=m, F_v(u,v+1)=n)}{\sum_{u,v} \delta(F_v(u,v)=m)} \\ p_d(m,n) &= \frac{\sum_{u,v} \delta(F_d(u,v)=m, F_d(u+1,v+1)=n)}{\sum_{u,v} \delta(F_d(u,v)=m)} \\ p_m(m,n) &= \frac{\sum_{u,v} \delta(F_m(u+1,v)=m, F_m(u,v+1)=n)}{\sum_{u,v} \delta(F_m(u+1,v)=m)} \end{aligned} \tag{6.7}$$

where $m, n \in \{-T, \cdots, 0, \cdots, T\}$, the summation range for u is from 0 to $S_u - 2$, and for v from 0 to $S_v - 2$, and

$$\delta(A=m, B=n) = \begin{cases} 1 & \text{if } A=m \ \& \ B=n \\ 0 & \text{otherwise} \end{cases}$$

For JPEG images, we can directly read off the quantized DCT coefficients to compute the transition probability terms. For uncompressed images, we initially

compress them (for both original and seam carved/seam inserted images) at a quality factor (QF) of 100, as the features are defined only for JPEG images.

6.4.4 Detection Using Markov Feature based Models

We vary the fraction of seams that is removed from (or inserted to) the image. For example, consider 20% seam carving (20% of the columns in the original image are removed) as our positive examples. We now divide the entire dataset into an equal number of training and testing images. For each set, we perform seam-carving on half of the images and keep the rest unmodified. A SVM-based model is learnt from the training images. In a practical case, we can have a random %age of the image columns corresponding to the number of seams carved. We also investigate the generality of the trained model as a seam-carved test image can have a seam carving percentage different from 20%.

Table 6.3 (or 6.4) presents the detection accuracy where train and test sets can contain same/different percentages of seam carving (or insertion). For the experiments, the dataset consists of 4500 natural images from the MM270K database[1]. We have worked with gray-scale images for all our experiments. The original images are in JPEG format at a QF of 75. The images are first decompressed, then seam-carving/insertion occurs (optimal seams are computed in the pixel domain)

[1]downloaded from http://www-2.cs.cmu.edu/yke/retrieval

Table 6.3: Seam-carving detection accuracy for different training-testing combinations: "test"/"train" refers to the seam-carving percent for the testing/training images, "mixed" refers to that case where the dataset used for training/testing consists of images with varying seam-carving percentages (images with seam-carving percentages of 10%, 20%, 30% and 50% are uniformly distributed in the "mixed" set).

test \ train	10%	20%	30%	50%	mixed
10%	**65.75**	66.54	66.26	64.91	**70.60**
20%	69.11	**70.36**	70.50	69.11	**75.72**
30%	74.00	75.54	**77.31**	77.63	**83.88**
50%	78.24	80.99	84.67	**86.72**	**91.29**
mixed	71.77	73.36	74.69	74.59	**80.37**

to create the positive training examples, and finally, images from both classes are JPEG compressed at a QF of 100. *Since the Shi-324 feature works in the quantized DCT domain, the input image has to be in JPEG format.* We have also experimented with uncompressed (TIFF) images from the UCID database [2], which are subsequently JPEG compressed at a QF of 100, and the detection results are similar to that obtained using JPEG images as the starting images.

For seam-carving detection, the "mixed" model, i.e. the maximally generalized model, (trained using images having different seam-carving fractions) results in the highest detection accuracy (Table 6.3). For seam-insertion detection, the "mixed" model works better than more specific models (trained using positive examples having a fixed seam-insertion fraction) for test-cases involving 10%-20% of seam-insertions (Table 6.4).

[2]http://vision.cs.aston.ac.uk/datasets/UCID/ucid.html

Table 6.4: Seam insertion detection accuracy for different training-testing combinations: the meaning of "test", "train" and "mixed" are same as in Table 6.3.

test \ train	10%	20%	30%	50%	mixed
10%	**68.55**	70.47	68.53	63.59	**76.95**
20%	76.36	**81.88**	84.64	81.38	**84.63**
30%	80.09	84.65	**88.49**	93.04	**85.71**
50%	82.01	87.41	91.49	**95.32**	**88.84**
mixed	76.74	81.09	83.28	83.34	**84.03**

6.4.5 Localizing Seam Insertions

In Section 6.4.2, we have described how seam insertion removes a pixel along the seam and replaces it by two pixels, which are averages of pixels lying on/near the seam, as shown in (6.5). In Fig. 6.15, the 4×5 input matrix $\mathbf{a}$ is converted to the 4×6 matrix $\mathbf{b}$ after inserting an extra seam. The seam consists of $\{a_{1,3}, a_{2,4}, a_{3,3}, a_{4,2}\}$, as shown in Fig. 6.15(a). After seam insertion, a seam pixel $a_{i,j}$ is replaced by 2 pixels - $b_{i,j}$ and $b_{i,j+1}$, as shown in Fig. 6.15(b). In (6.8), we show that a simple relation exists between the pixels bordering a seam. We create a binary matrix P where the likely seam pixels (pixels which are assumed to correspond to seam insertion locations) are set to 1 (or 0) if the condition in (6.8) is (or is not) satisfied.

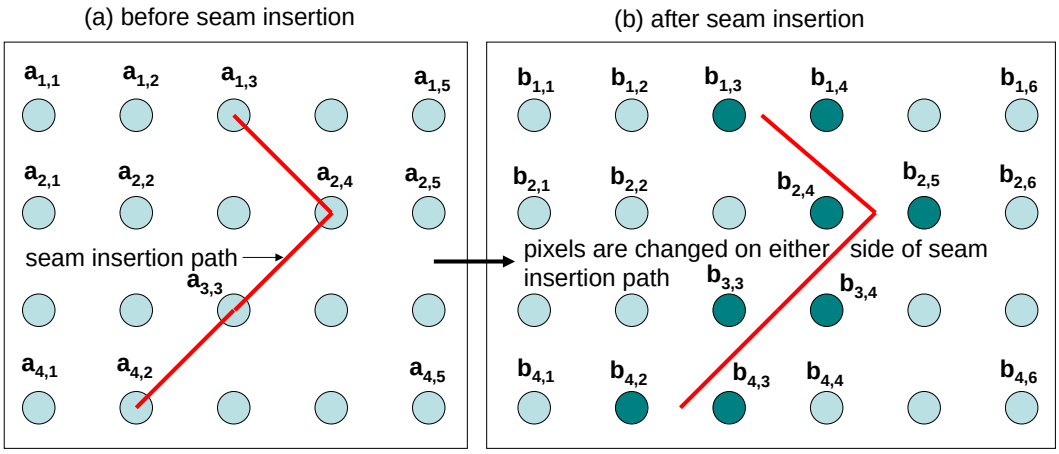


Figure 6.15: Example of seam insertion: (a) $\{a_{i,j}\}_{i=1,j=1}^{i=4,j=5}$ and (b) $\{b_{i,j}\}_{i=1,j=1}^{i=4,j=6}$ are the image matrices before and after seam insertion, respectively. For points along the seam, the values are modified as shown for the first row: $b_{1,1} = a_{1,1},\ b_{1,2} = a_{1,2},\ b_{1,3} = \text{round}(\frac{a_{1,2}+a_{1,3}}{2}),\ b_{1,4} = \text{round}(\frac{a_{1,3}+a_{1,4}}{2}),\ b_{1,5} = a_{1,4},\ b_{1,6} = a_{1,5}$.

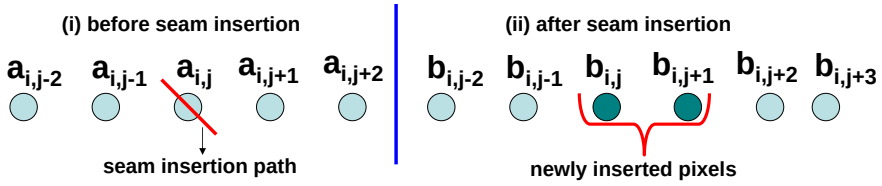


Figure 6.16: Understanding linear relationship between seam inserted pixels

Consider the image matrix $\mathbf{b}$ after seam insertion. For the new pixel values introduced after seam insertion,

$$b_{i,j} = \text{round}(\frac{a_{i,j-1} + a_{i,j}}{2})$$
$$b_{i,j+1} = \text{round}(\frac{a_{i,j} + a_{i,j+1}}{2}),$$

without rounding, and using $a_{i,j+1} = b_{i,j+2},$ and $a_{i,j-1} = b_{i,j-1},$

$$b_{i,j+1} - b_{i,j} = \frac{(a_{i,j+1} - a_{i,j-1})}{2} = \frac{b_{i,j+2} - b_{i,j-1}}{2},$$

due to rounding, the modified condition is

$$|(2.b_{i,j} - b_{i,j-1}) - (2.b_{i,j+1} - b_{i,j+2})| = 0, \text{ or } 1, \tag{6.8}$$

we set

$$P_{i,j} \begin{array}{l} =1 \quad \text{if } |(2.b_{i,j} - b_{i,j-1}) - (2.b_{i,j+1} - b_{i,j+2})| = 0, \text{ or } 1 \\ =0 \quad \text{otherwise} \end{array} \tag{6.9}$$

When the seam passes through the image border, $\{a_{i,j}, a_{i,j+1}\}$ ($a_{i,j}$ or $a_{i,j+1}$ is a seam pixel) gets modified to $\{b_{i,j}, b_{i,j+1}, b_{i,j+2}\}$, where $b_{i,j} = a_{i,j}$, $b_{i,j+1} = \text{round}(\frac{a_{i,j}+a_{i,j+1}}{2})$ and $b_{i,j+2} = a_{i,j+1}$.

For a pixel one pixel away from a bordering column,

$$\text{we set } P_{i,j} \begin{array}{l} = 1 \quad \text{if } (2.b_{i,j} - (b_{i,j-1} - b_{i,j+1})) = 0, \text{ or } 1 \\ = 0 \quad \text{otherwise} \end{array} \tag{6.10}$$

Once we have this binary matrix P, the next issue is to properly use this matrix to localize seam insertions. A vertical seam traverses the entire image from top-to-bottom. Hence, for points lying along the same seam, a '1' in a certain row should be preceded by a '1' in one of its three nearest-neighbors (NN) in the upper row (i.e. it has a parent node valued '1') and also by a '1' in one of the three NN in the lower row (i.e. it has a child node valued '1'). For example, in Fig. 6.15(b), $\{b_{1,1}, b_{1,2}, b_{1,3}\}$ are parent nodes and $\{b_{3,1}, b_{3,2}, b_{3,3}\}$ are child nodes for $b_{2,2}$. By

tracking points which have a '1'-valued parent (except pixels in the first row) and child node (except pixels in the last row) using (6.11), we obtain the binary matrix P_B from P.

$$\begin{aligned} P_B(i,j) \quad &= \quad 1 \text{ if } P_{i,j} = 1, \text{ (current node is 1)} \\ &\text{and} \quad \max\{P_{i-1,j-1}, P_{i-1,j}, P_{i-1,j+1}\} = 1, \text{ (one parent node is 1)} \\ &\text{and} \quad \max\{P_{i+1,j-1}, P_{i+1,j}, P_{i+1,j+1}\} = 1, \text{ (one child node is 1)} \\ &= \quad 0, \text{ otherwise} \end{aligned} \tag{6.11}$$

Problem Scenarios: While using (6.9) or (6.10), we may obtain '1' in locations other than the image seams due to the inherent smoothness in many image regions. For example, if the pixels in $\{b_{i,j-1}, b_{i,j}, b_{i,j+1}, b_{i,j+2}\}$ are all equal-valued, then (6.9) (or (6.10)) is satisfied. In the absence of any noise or compression attacks, are we always guaranteed to recover all the inserted seams? **The answer is no** - when two or more seams intersect, the linear relationship between the pixels adjoining the seams gets modified at the point of intersection. Also, in many cases, there are multiple possible paths for a seam, where some options correspond to smooth regions and not the actual seam. Our algorithm can identify likely seam points ('1'-valued elements in P) - due to intersections, the parent-child relationship may not hold for all seams ('1'-valued elements in P_B may not include all the seam points) and hence, some of the initially inserted seams may

not be recovered. In subsequent figures in Section 6.4.5, we have shown the likely positions of the recovered seams based on the '1'-valued elements in P_B, but these points could not be separated into distinct individual seams, due to the multiple path problem.

The possible detection (and error) scenarios are as follows:

(i) **One/more seams detected for seam-inserted images**: the matrix P_B should be non-zero for seam-inserted images as '1's are present only for likely seam points in P_B. Due to many seam intersections or due to noise (for example, compression attack), seams may not be detected leading to missed detections. Note (6.9) where the linear relationship hold exactly for a zero-noise case. Therefore, we are likely to get better seam localization for uncompressed images (JPEG compression adds noise). A possible solution is to relax the conditions for finding a seam point (discussed later in Section 6.4.5).

(ii) **No seams detected for original images**: there should not be false detections, i.e. P_B should be an all-zero matrix for a normal image. However, due to the presence of smooth regions, even un-tampered images may demonstrate the presence of seams by satisfying (6.9) (or (6.10)). In Sec-

(a) (b) (c)

(d) (e)

Figure 6.17: The five images shown here are gray-scale versions of color images obtained from the UCID database - we have used gray-scale images for all our experiments. These are the original images based on which the visual examples in Section 6.4.5 are obtained.

tion 6.4.5, we discuss methods where we reduce the false alarm by discarding the smooth regions before computing P and P_B.

Visual Illustrations of Seam Insertion Detection

Fig. 6.17 shows the original images used as examples in this section.

Identifying Seam Locations from a Given Image

We explain the computation of P_B from P using Fig. 6.18. For displaying binary images, the color convention used is "white" = 1 (seams present) and "black" = 0 (seams absent). The P matrix (Fig. 6.18(b)) is pruned to obtain P_B (Fig. 6.18(e)), which is the intersection of '1'-valued elements in parts (c) (is '1' for elements with valid parents) and (d) (is '1' for elements with valid child nodes).

Seam Localization for Varying Seam Insertion Fractions

In Figs. 6.19 and 6.20, we demonstrate the localization accuracy of inserted seams for various seam-insertion fractions, for 2 images. In Figs. 6.19 and 6.20, we have experimented with different seam-insertion fractions of 1% ((a)(d)(g)(j)), 5% ((b)(e)(h)(k)), 10% ((c)(f)(i)(l)). The first row shows the 1%, 5% and 10% seam-inserted images. The second row shows the same images with the seams marked. The third row shows the actual seam insertion locations in a binary image. The last row outputs the P_B binary matrix using (6.11), where the 1's denote the seam insertion locations obtained using our algorithm. However, we have not separated its '1'-valued elements into individual seams.

Reducing False Positives by Discarding Smooth Regions

In some images, due to presence of very smooth regions, the conditions (6.9),(6.10) are satisfied even where seams are not inserted. As a possible solution, we discard columns pertaining to smooth regions so that they are not involved in the com-

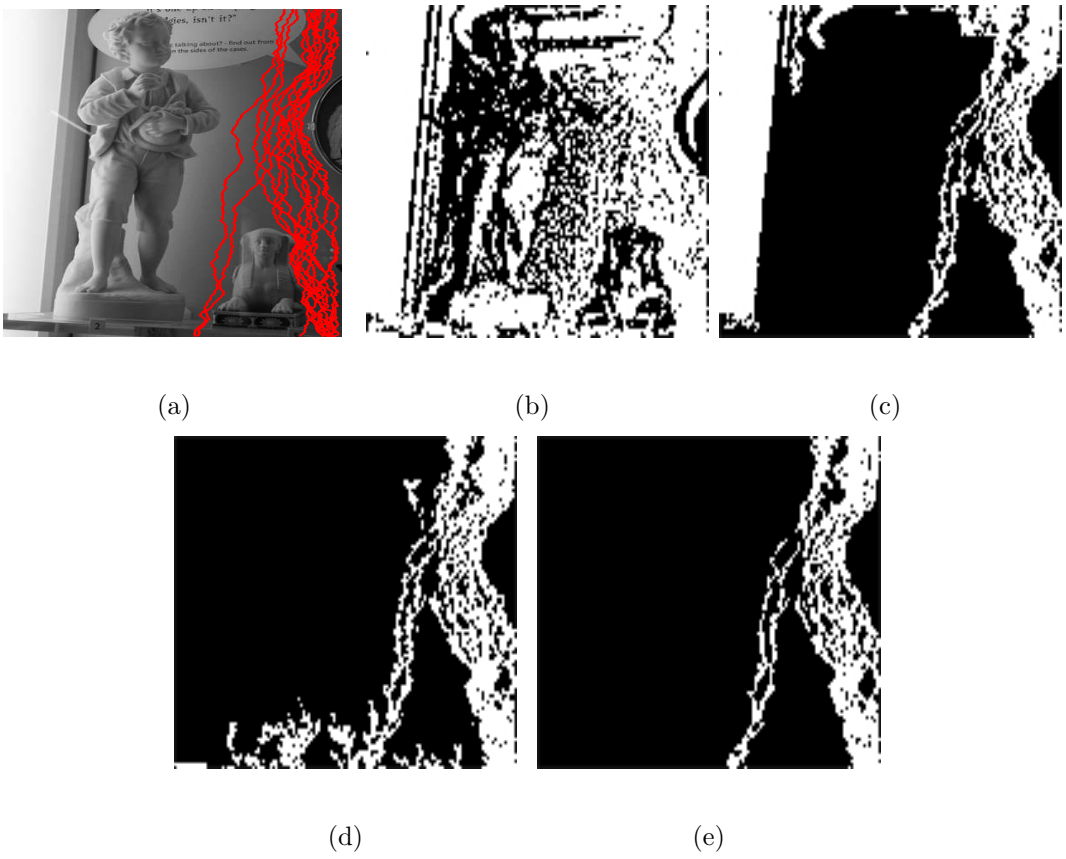


Figure 6.18: (a) the image, after 3% seam insertion, with seams marked in red, (b) P matrix for the seam inserted image, (c)/(d) pruned P matrix with points having valid parent/child nodes marked in white (e) P_B matrix, with retained points (marked in white) having valid parent and child nodes. Comparing (a) and (e), we see that the detected seams are very similar to the actual inserted seams.

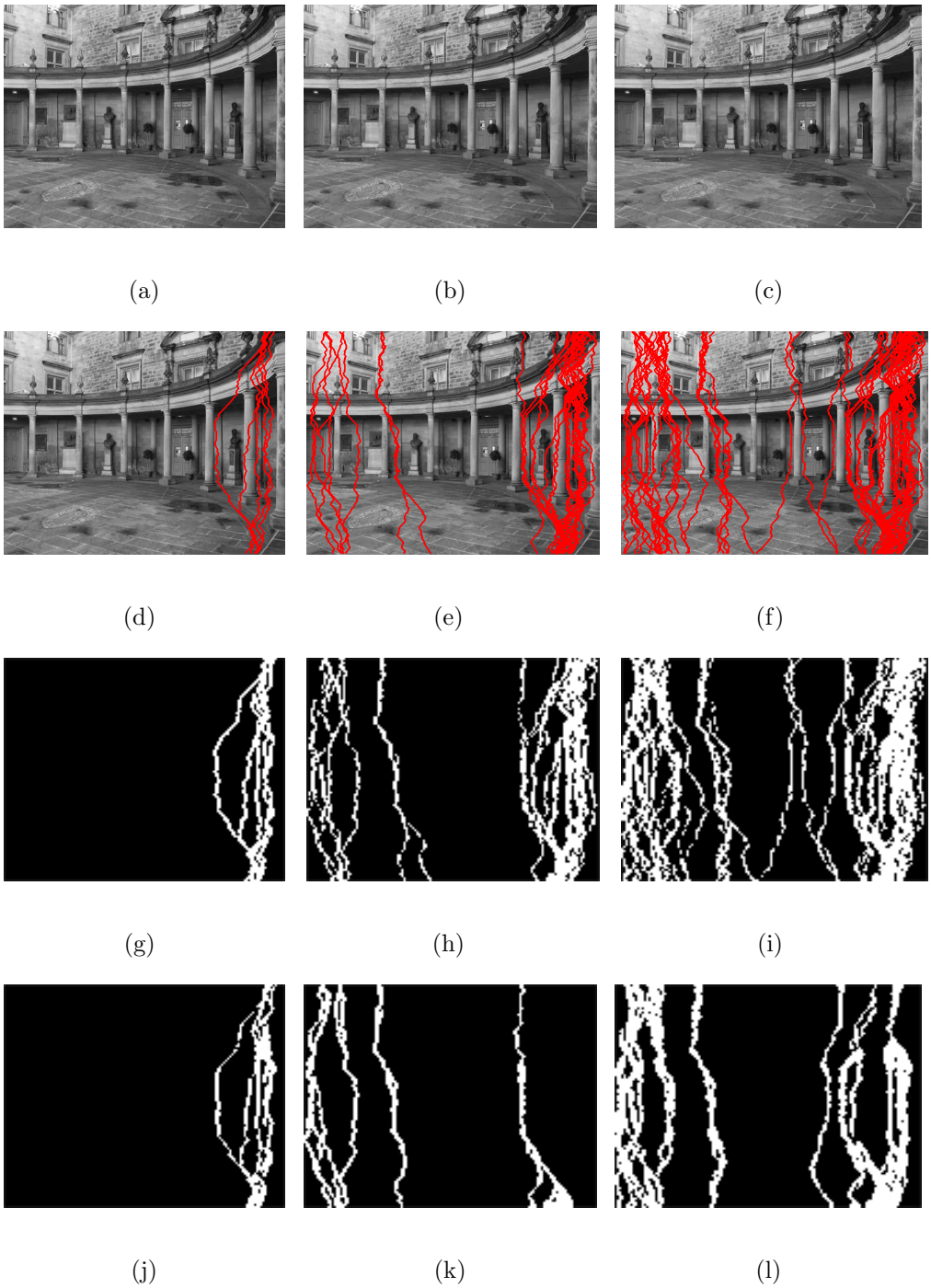
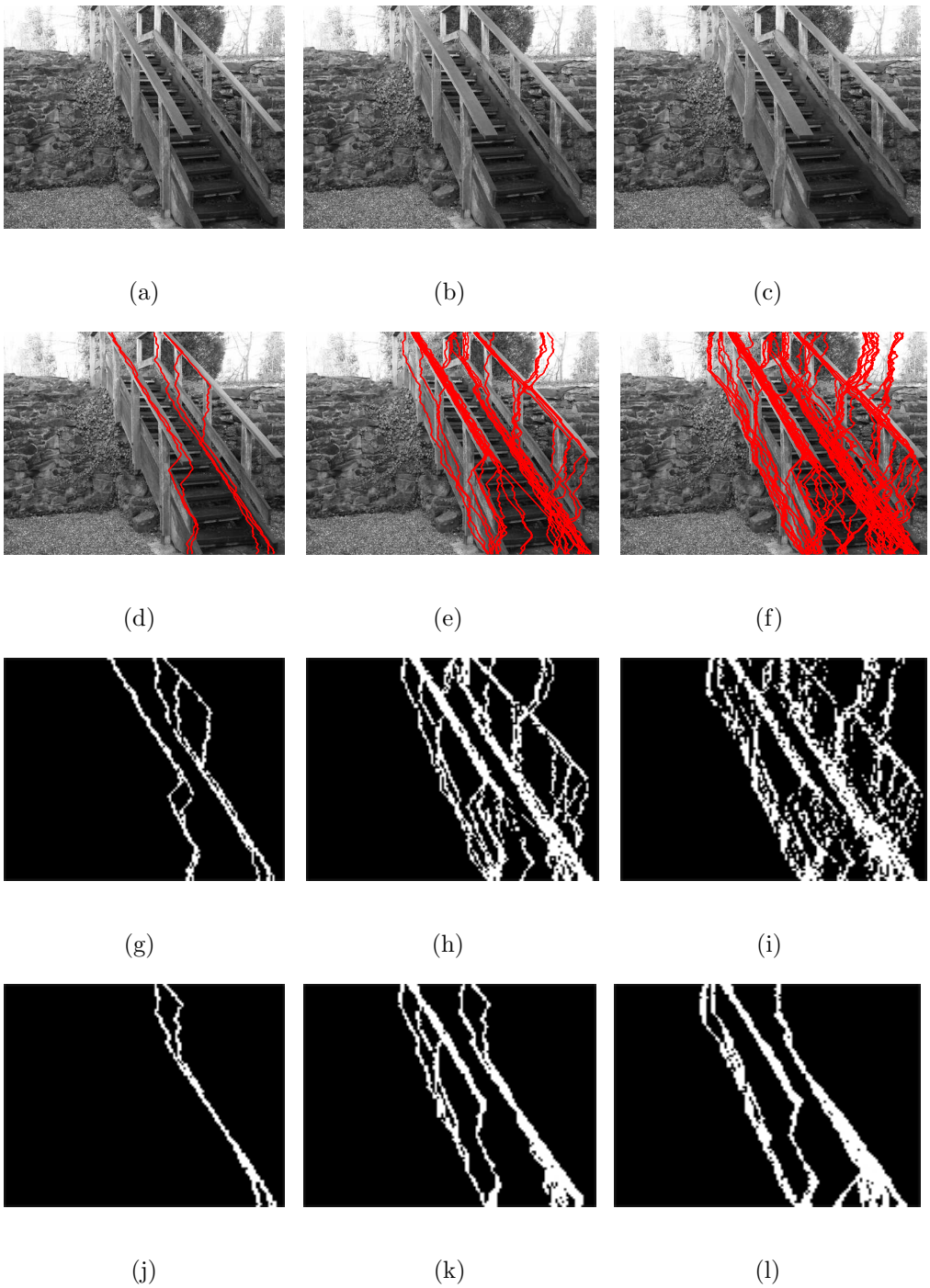


Figure 6.19: (Localizing seam insertions: parts (a,d,g,j), (b,e,h,k) and (c,f,i,l) correspond to 1%, 5% and 10% seam insertions. The seams are marked with thicker red lines in 2^{nd} row for ease of visualization. The 3^{rd} and 4^{th} rows display the actual and the detected seams, respectively.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 6.20: The seam insertion localization is performed on another image, similar to the steps in Fig. 6.19.

putation of P and P_B. We need to come up with a quantitative measure for the smoothness of a column so that we can decide which columns to consider as potential seam locations. The measure we use is the maximum separation between two consecutive edge points along a column divided by the number of pixels along a column.

For "smooth region" detection, we obtain the Canny edge-map [14] and discard columns for which the spacing between two consecutive edge-points along a certain column is quite high (or when the column has no edge point). For example, let there be m edge points for the j^{th} column in a $N_1 \times N_2$ image, denoted by $\{e_{i,j}\}_{i=1}^{m}$. The successive difference values are $\{d_{i,j}\}_{i=1}^{m-1}$, where $d_{i,j} = e_{i+1,j} - e_{i,j}$. If $\max_i d_{i,j} > s_{frac}.N_1$, we remove the j^{th} column, where s_{frac} is a tunable parameter. *A very high value of s_{frac} may result in smooth columns still being retained in the original image (leading to false detection of seams) while a very low value will remove so many columns that seams cannot be detected even in the seam-inserted image.* This trade-off is studied in Section 6.4.5.

The P matrix is then computed on the new image (after discarding smooth columns). For two sample images (Fig. 6.21 and 6.22), it is seen that this helps in reducing false positives - i.e. even after removing the smooth columns, P_B for the seam-inserted image contains '1' (shows presence of seams) while P_B for the

original image consists of '0's only (the colormap used for plotting is such that an all-zero matrix is shown as an all-green image).

A 3% seam-inserted image is shown in Fig. 6.21(a). The P matrix for the seam-inserted image (without smoothing) is shown in (b) and the pruned P matrix with valid parent nodes is shown in (c). The P_B matrices for the seam-inserted and original images are shown in (d) and (e), respectively. The spurious seams detected for the original image correspond to the smooth regions in the leftmost part of the original image. Using s_{frac} of 0.98, the columns pertaining to the smooth region are discarded, as shown in (f). The corresponding P matrix is shown in (g), and (h) and (i) display the matrix after parent and (parent + child node) based pruning. After smoothing, P_B for the original image is an all-zero matrix (j).

In Fig. 6.22, 8% seam insertion is done, and (a) and (b) show the Canny edge maps for the original and the seam-inserted images. The smooth columns identified using s_{frac} of 0.65 are shown as vertical red lines in (c). The P_B matrices for the seam-inserted and original images (for s_{frac} of 0.65) are shown in (d) and (e), respectively. The seam-inserted image is shown in (f), with the seams marked in red. The smooth columns obtained using s_{frac} of 0.45 for the seam-inserted and original images are shown in (g) and (h), respectively. It is to be noted that for the same value of s_{frac} (0.45), more columns are removed form the original image than

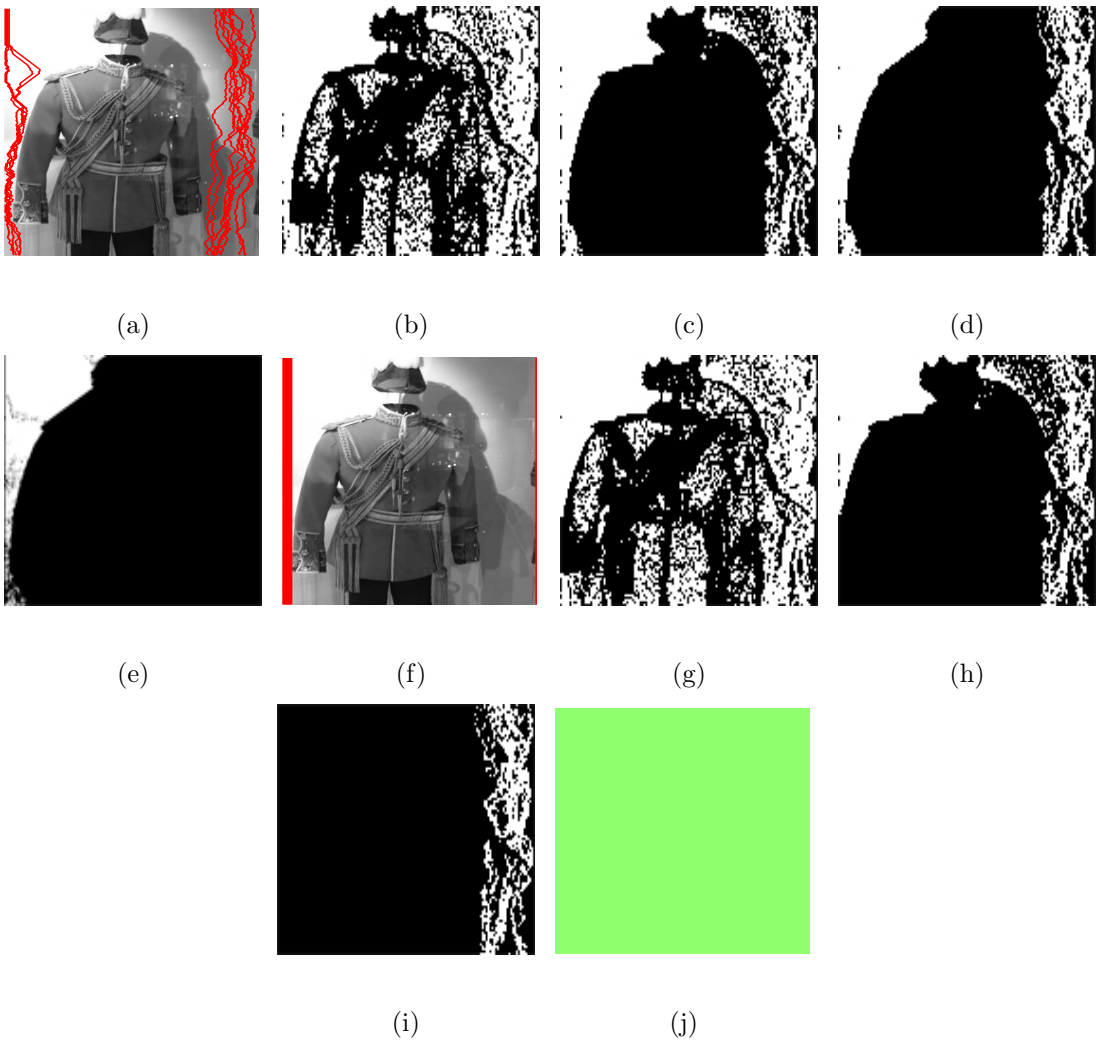


Figure 6.21: (a) 3% seam-inserted image (b) P matrix for seam inserted image (no smoothing) (c) P matrix after parent based pruning (d)/(e) P_B for seam-inserted/original image (f) seam-inserted image after smoothing using s_{frac} of 0.98 (g) P matrix after smoothing (h) corresponding P matrix after parent based pruning (i)/(j) P_B for seam-inserted/original image (using s_{frac} of 0.98). **The green image means an all-zero image**.

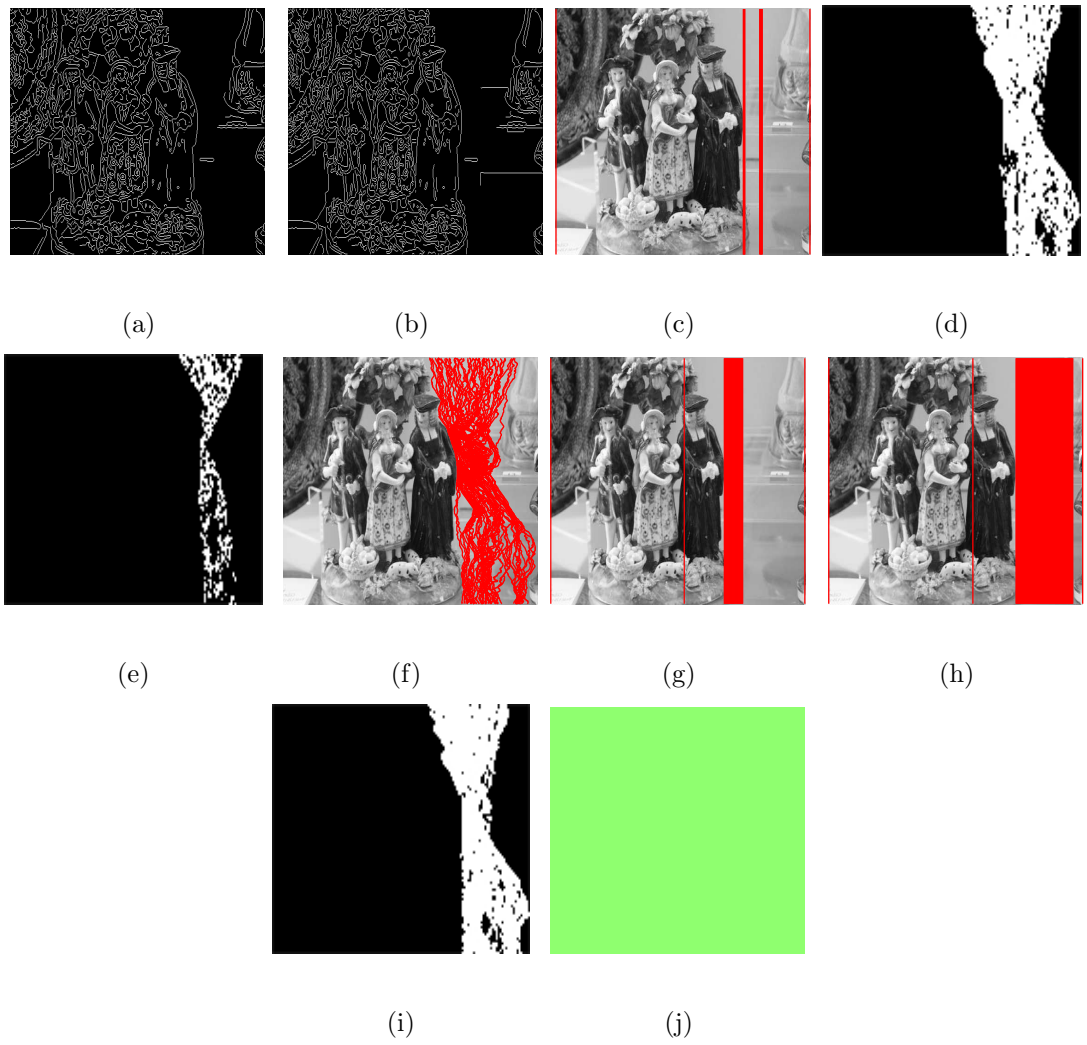


Figure 6.22: (a) Original image's Canny edge-map (b) 8% seam-inserted image's Canny edge-map (c) original image after smoothing using s_{frac} of 0.65, (d)/(e) P_B for seam inserted/original image (s_{frac} of 0.65) (f) 8% seam-inserted image (g)/(h) smoothing on seam-inserted/original image (s_{frac} of 0.45) (i)/(j) P_B for seam-inserted/original image (s_{frac} of 0.45). The columns discarded from the seam-inserted image are similar to the original for s_{frac} of 0.65 (hence, only 1 figure (c) is shown) - however, the columns removed from the seam-inserted and original images differ significantly for s_{frac} of 0.45, as shown in (g) and (h). **The green image means an all-zero image**.

from the seam-inserted image, which suppresses the detection of spurious seams from the original image. The P_B matrices for the seam-inserted and original images are shown in (i) and (j), respectively. *From these two examples, we see that the smoothing factor needed to reduce the false detections varies for different images.*

Detection Accuracy for Varying Seam Insertion Fractions and Smoothing Factors

We compute the seam-insertion detection accuracy over 1338 images from the UCID database. In Fig. 6.23, P_{MD} and P_{FA} refer to the probability of missed detection (failing to detect seams in positive examples) and false alarm (detecting seams in original images), respectively. For every set of experimental parameters (seam-insertion percent and s_{frac}), we use half of the images for seam-insertion and keep the rest unchanged, so that both errors are equally weighted. Fig. 6.23(a) shows the two types of detection errors for TIFF images, (b) shows the errors after JPEG compressing the images at QF of 100, (c) shows the errors after JPEG compression but relaxing the conditions for obtaining '1's in P. The total detection error for TIFF images, JPEG images, and JPEG with "relaxed conditions" is shown in (d), (e) and (f), respectively. As s_{frac} increases, P_{FA} increases as due to reduced pruning of smooth regions, more original images show presence of

seams. However, with increase in s_{frac}, P_{MD} decreases significantly, specially for low seam-insertion fractions. Hence, overall, the detection error decreases as s_{frac} increases from 0.4 to 0.8 (the decrease in P_{MD} is dominant over the increase in P_{FA} with increase in s_{frac}).

The "relaxed conditions" for computing $P_{i,j}$, for non-border pixels, are as follows (similar to (6.9)):

$$P_{i,j} \begin{array}{ll} =1 & \text{if } |(2.b_{i,j}-b_{i,j-1}) - (2.b_{i,j+1}-b_{i,j+2})| \leq \delta_1 \\ =0 & \text{otherwise} \end{array} \tag{6.12}$$

where δ_1 is increased to allow more pixels to be labeled as '1'. For a pixel one pixel away from a bordering column (similar to (6.10)),

$$P_{i,j} \begin{array}{ll} = 1 & \text{if } |(b_{i,j} - \frac{(b_{i,j-1}-b_{i,j+1})}{2})| \leq \delta_2 \\ = 0 & \text{otherwise} \end{array} \tag{6.13}$$

where the number of '1's increases by increasing δ_2.

For JPEG QF=100, we have used the cutoff values $\delta_1 = 3$ and $\delta_2 = 1.5$. For more severe compression, we may have to increase the cutoff values. By comparing Fig. 6.23(b)-(c) and (e)-(f), we observe that the false alarm rate increases with the "relaxed conditions"; however, the decrease in the missed detection rate is higher compared to the increase in the false alarm rate. Hence, the overall detection error is lower for the "relaxed conditions".

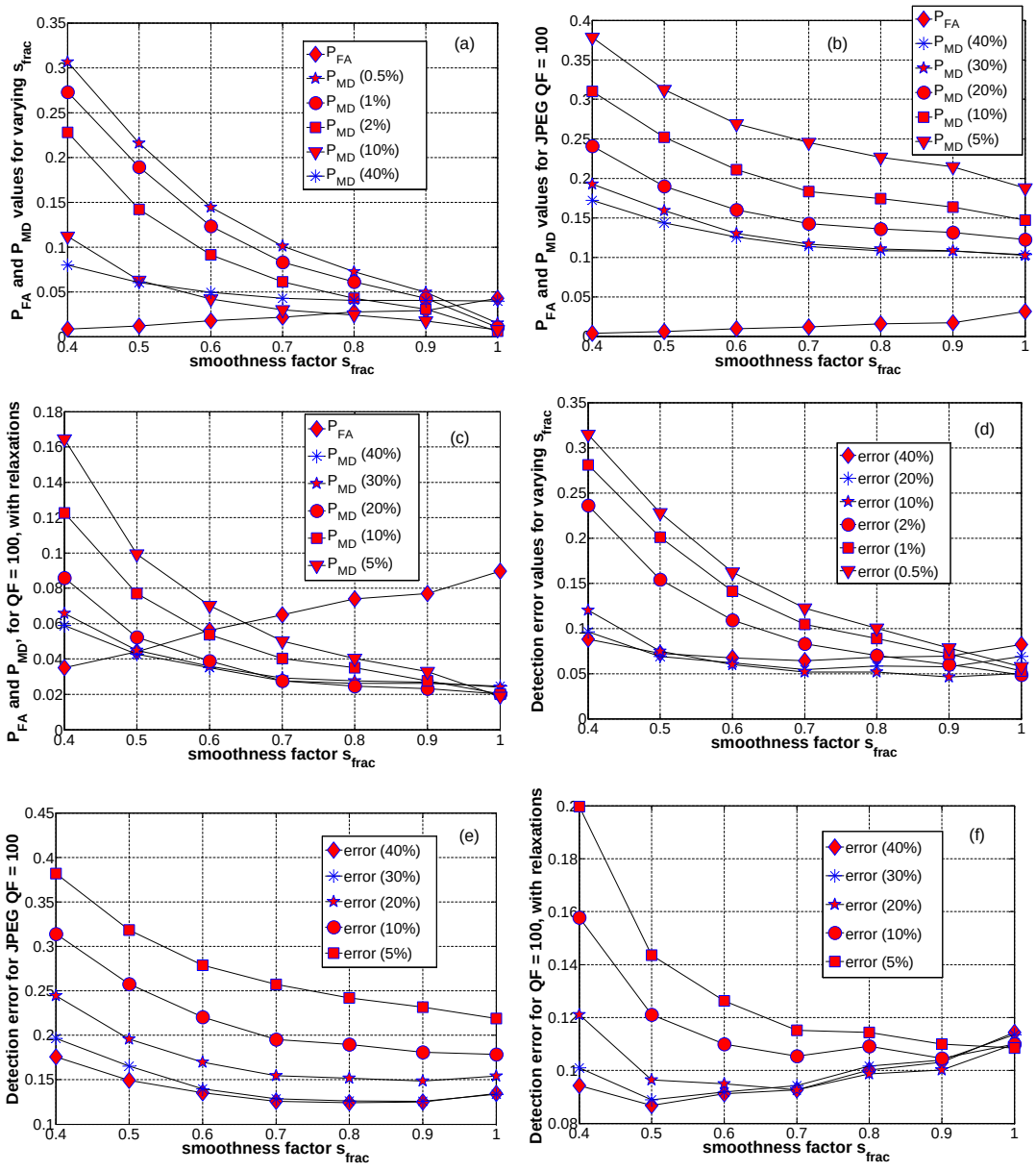


Figure 6.23: (a)/(b)/(c) False alarm (P_{FA}) and missed detection (P_{MD}) rates are computed for TIFF images/ JPEG images/ JPEG images with "relaxed conditions for obtaining positive elements in P". Similarly, (d)/(e)/(f) show the detection error rates for these three cases. The number in parentheses denotes the seam-insertion percent, for example, in (a), $P_{MD}(40\%)$ denotes the probability of missed detection for 40% seam insertion.

6.4.6 Probabilistic Approach for Detecting Seam Insertions

We provide a brief introduction to Popescu and Farid's probabilistic approach [85] for re-sampling detection, and then show how it can be used to detect seam insertions. Re-sampling introduces periodic correlations among pixels due to interpolation. To detect these correlations, they use a linear model in which each pixel is assumed to belong to two classes- re-sampled class M1 and non re-sampled class M2, with equal prior probability. To simultaneously estimate a pixel's probability of being a linear combination of its neighboring pixels and the weights of the combination, an Expectation Maximization (EM) algorithm is used. In the Expectation step, the probability of a pixel belonging to class M1 is calculated. This is used in the Maximization step to estimate the weights. The stopping condition is when the difference in weights between two consecutive iterations is very small. At this stage, the probability matrix obtained in the Expectation step for every pixel of the image is called the "Probability map (p-map)" - the p-map concept has been explained in Section 6.3.3. For a re-sampled image, this p-map is periodic and peaks in the 2D Fourier spectrum of the p-map indicate re-sampling. A probability value close to 1 indicates that a pixel is re-sampled.

We use the above method with small modifications to detect seam insertion. During seam insertion, the seam pixel is removed and then replaced by two pixels whose values are the average of the seam pixel's left and right neighbors, as shown in (6.5). This is similar to re-sampling and the pixels that are inserted are correlated with its neighbors. To detect seam insertions, we first find the p-map as described above. Since there is no periodic re-sampling, most pixels in a natural image usually have a high value in the p-map. We find a pattern to detect inserted seams by exploiting the fact that the seams form an 8-connected path.

Similar to the P matrix (6.9), we threshold the p-map to obtain a binary matrix (points greater than the threshold δ_{th} are set to 1). The 1's in the binary matrix correspond to our initial estimate of the likely seam locations. Our initial experiments suggest that if δ_{th} is properly chosen, then the pruned binary matrix (similar to obtaining P_B (6.11) from P) will show presence/absence of seams depending on whether it is a seam inserted/original image. A 5×1 window is taken and the current pixel is assumed to be linearly related to its four neighbors. Since the pixels in a smooth region are highly correlated, they have a high value in the p-map while for edges, the p-map value will be low.

For 10% seam insertion, for the best choice of the threshold, the accuracy obtained is 63%. It is to be emphasized that the knowledge of the exact weights (0.5 and 0.5, as in (6.5)) for the newly introduced pixels during seam insertion

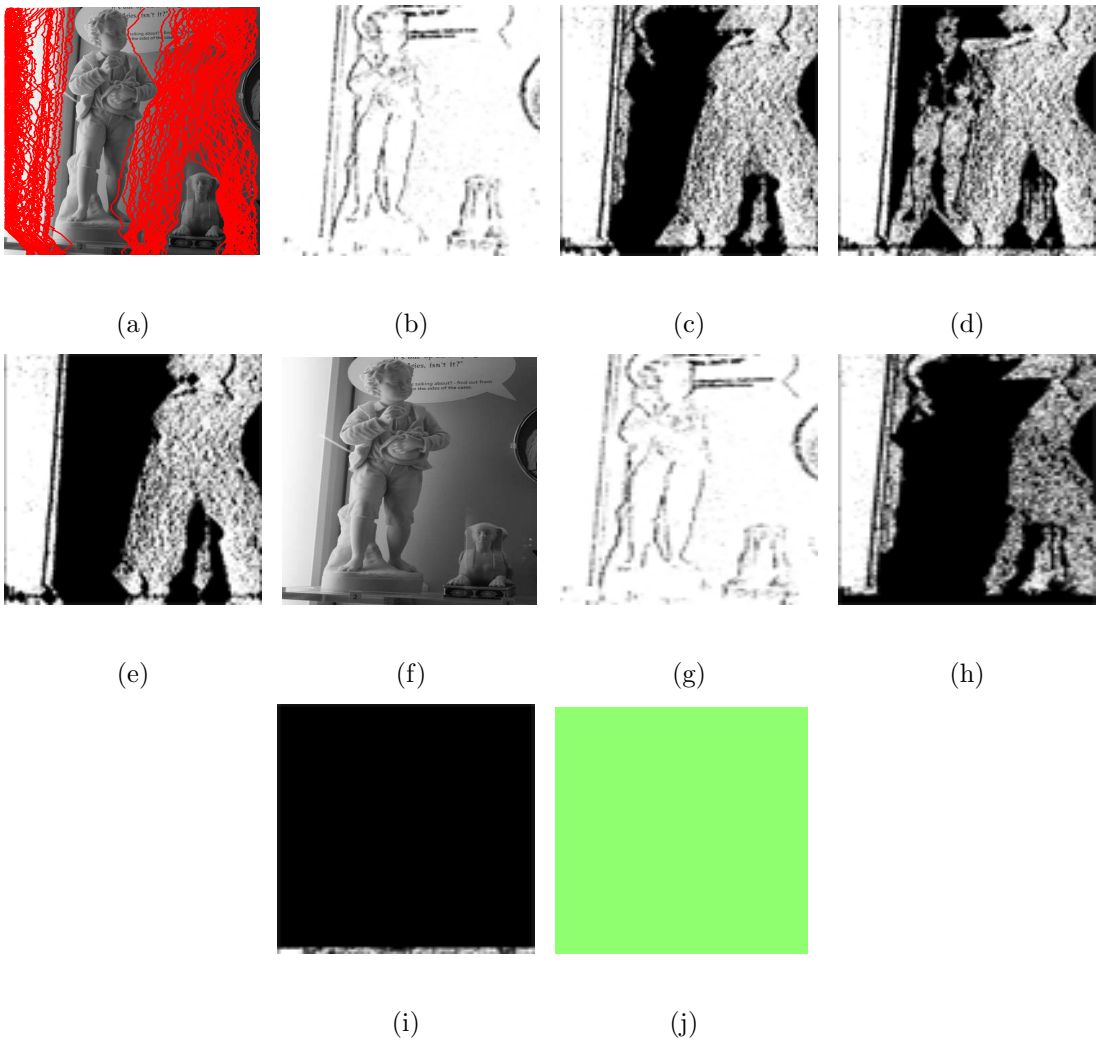


Figure 6.24: (a) image with 25% seams inserted, (b) p-map matrix for seam inserted image, (c) binary matrix obtained after using δ_{th} of 0.98 for p-map, (d) only the pixels in (c) with valid parent nodes are retained, (e) the pixels in (c) with valid parent and child nodes are shown - these indicate the seams detected, (f) original image, (g) p-map for original image, (h) corresponding binary matrix using δ_{th} of 0.98, (i) the binary matrix with parent node based pruning, (j) after (parent+child) node based pruning, the resultant binary matrix has all zeros. **The green image means an all-zero image**.

is not utilized in this probabilistic framework. In Fig. 6.24, we show that for a carefully chosen value of δ_{th}, the binary matrix obtained by thresholding the p-map can return the seams for the seam-inserted image and an all-zero matrix for the normal image.

6.4.7 Rotation/Scale Detection

The rotation and scaling detection experiments use the same dataset of 4500 images, as used for seam-carving detection in Section 6.4.4. For the scaling detection (Table 6.5), the positive examples involve scaling by a certain fraction (0.25-2) along both dimensions. From the detection results using **Shi-324** and SVM-based trained models, we observe that when the scale factor is either much less than 1(0.25 or 0.50) or much greater than 1(1.50, 2), the detection accuracy is quite high. For scaling factors of 0.95 and 1.05 (close to 1), the detection accuracy is 60% and 75%, respectively. Also, the detection rate is high when the scaling factor for the test images is close to the scale factor based on which the SVM model was trained. Only for very high scaling factors (2), the detection rate is high even when the model was trained based on a much lower scale factor (1.05).

We perform similar experiments for determining the rotation angle of a given image (Table 6.6). The positive examples consist of images which are first rotated and then cropped. It is observed that the detection rate is higher for rotation

Table 6.5: Detection of scaling using Shi-324 feature and SVM models for different train-test combinations: "test" refers to the scale-factor for the positive examples in the test set, while "train" refers to the scale-factor for the positive examples in the training set.

test \ train	0.25	0.50	0.75	0.95	1.05	1.25	1.50	2.00
0.25	**95.20**	**92.08**	80.89	40.31	39.05	49.11	49.67	48.70
0.50	**87.93**	**92.50**	**87.47**	38.58	37.05	46.69	48.09	46.97
0.75	63.56	68.45	**73.53**	44.08	44.92	46.69	48.60	47.76
0.95	49.81	48.88	51.16	**60.81**	56.10	49.72	50.37	50.28
1.05	38.77	40.63	46.41	65.24	**74.98**	60.07	56.48	55.92
1.25	47.02	44.69	39.42	60.02	66.36	**88.96**	69.25	60.90
1.50	31.83	26.37	21.11	77.68	**86.11**	**91.52**	**96.41**	**88.49**
2.00	26.93	21.71	16.17	81.92	**90.35**	**93.10**	**98.70**	**98.23**

angles much greater than 0°. The detection rate is 73%, 88%, 94% and 95% for rotation angles of 10°, 20°, 30° and 40°, respectively (when 60% of the image is retained per dimension).

For a practical setting, a variety of SVM models, based on different rotation and scale factors, can be used to detect whether an image is rotated or scaled. *Thus, we have shown that the Shi-324 feature is generalizable for a variety of tamper operations - seam carving/insertion, rotation and scaling.*

Table 6.6: Rotation detection using Shi-324 and trained SVM models: "test" refers to the (crop fraction + rotation angle combination) for the positive examples in the test set, while "train" refers to the combination for the positive examples in the training set. For example, (60%, 10°) refers to positive examples which are first rotated by 10° and then 60% of the image is retained per dimension.

test \ train	60%, 10°	60%, 20°	60%, 30°	60%, 40°
60%, 10°	**73.67**	71.48	59.65	58.71
60%, 20°	68.31	**88.12**	**87.56**	**87.05**
60%, 30°	64.82	**89.93**	**94.04**	**94.55**
60%, 40°	63.89	**90.31**	**94.45**	**95.39**
80%, 10°	73.21	71.81	59.37	58.90
80%, 20°	69.52	**87.88**	**87.14**	**87.47**
80%, 30°	65.94	**89.19**	**93.43**	**93.94**
80%, 40°	64.12	**89.61**	**94.64**	**95.29**

test \ train	80%, 10°	80%, 20°	80%, 30°	80%, 40°
60%, 10°	72.74	66.64	57.08	54.33
60%, 20°	67.94	**83.50**	75.49	65.10
60%, 30°	64.77	**87.98**	**86.53**	75.68
60%, 40°	63.65	**89.14**	**88.21**	79.36
80%, 10°	**72.93**	69.71	58.53	55.87
80%, 20°	70.08	**89.05**	**87.56**	78.19
80%, 30°	67.05	**94.04**	**95.71**	**94.69**
80%, 40°	66.08	**95.25**	**96.51**	**96.09**

6.5 Localization of Seam Carving based Object Removal

We have introduced the concept of object removal through seam carving in Fig. 6.13. We revisit the object removal problem with a goal of not only to detect seam carving but also to localize the object removal.

We describe the object removal example (again) through Fig. 6.25. Here, we remove a certain image region with the seam carving technique. The user selects the region to be removed (Fig. 6.25(b)) As the energy value computed for the image pixels generally corresponds to the gradient computed per pixel, a mask of very low negative values (lower than the minimum gradient value for all the other pixels) is applied in the gradient domain to all the image pixels to be removed. This ensures that the seams that are selected pass through the object to be removed (Fig. 6.25(c)). Seams are carved from the image till all the pixels belonging to the masked area have been removed (Fig. 6.25(d)). After seam carving, seam insertion can be done so that the size of the image is restored (Fig. 6.25(f)).

We have adapted the machine learning framework for seam carving/insertion detection to the object removal problem. Object removal concentrates all the seams to be carved in a certain portion of the image. The first intuition that

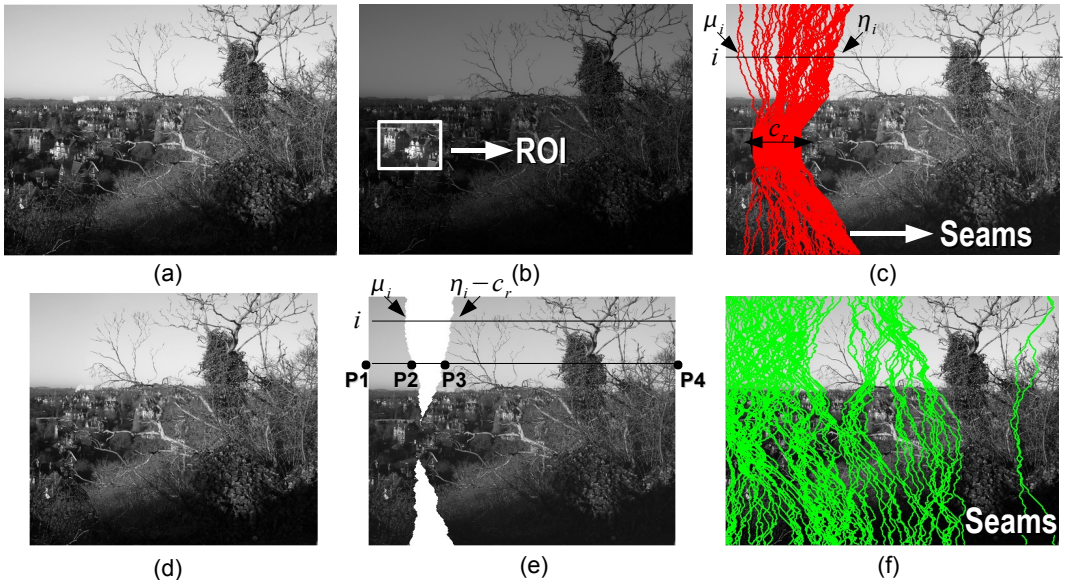


Figure 6.25: Object removal with the seam carving algorithm: (a) original image, (b) the white spot shows the region of interest, (c) seams to be removed: they are forced to pass through the region of interest (marked as ROI): we use μ_i and η_i to denote the first and last column affected by seam carving in the i^{th} row, respectively, and c_r denotes the number of seams removed, (d) is the image after seam carving, and (e) shows the zones affected by seam carving in the tampered image, (f) the seam insertions are shown which restore the original image size.

we have experimented with to localize the tampering is to employ a block-based approach: we divide the image into blocks and we use **Shi-324** [103], a 324-dim DCT domain Markov feature, computed per block as an input for the classification framework. The challenge we face here is that object removal with seam carving does not affect the image in the same way as traditional seam carving. In fact, *seams are concentrated in a single chunk where the object area is, while in regions that are far away from the object to be removed, seams are horizontally scattered. This means that intra-block statistics vary differently for different*

parts of the image. Therefore, the binary classification (untampered block/block affected by seam carving) is not efficient for localization purposes because the actual block where the object is removed does not fall in either of the two classes. We experimentally identify, depending on the variations in the **Shi-324** feature computed in different parts of the image, three different classes:

(i) untampered block: no/few seams pass from here,

(ii) seam carving: the object is not actually removed from here, but many seams pass through it,

(iii) object removal zone: seams are concentrated here (consecutive seams are vertical in this zone).

We use a two-phase approach to localize the seam carving that provides a resolution of 1 block out of 9 after dividing the image to 3×3=9 blocks.

(1) The first phase divides the image into 3 columns and identifies the column which is most probable to contain the tampered region (where object removal has occurred).

(2) In the second phase, we consider the tampered blocks individually after dividing the selected column to 3 horizontal chunks to detect the most likely chunk from where the object has been removed.

6.5.1 Object Removal

Object Removal Algorithm: Seam carving can be used to remove objects from the scene without duplicating or copy-pasting other areas of the image. The object removal algorithm performs seam carving on all the possible 8-connected paths that pass from the region to be removed, until all the pixels belonging to the selected region are deleted from the image. For ease of understanding, we present some basic notations in Table 6.7.

Table 6.7: Table of Notations used in connection with seam carving based object removal

Notation	Definition
$I_{M\times N} = \{I_{i,j}\}_{i=1,j=1}^{M,N}$	image intensity matrix
(i_1, j_1) ,(i_2, j_2)	are the left-top most and bottom-right most coordinates of the bounding box drawn around the object to be removed
A	$\{A(u,v)\}_{u=1,v=1}^{i_2-i_1+1,j_2-j_1+1} = \{I(x,y)\}_{x=i_1,y=j_1}^{i_2,j_2}$ is the region selected from the image, which corresponds to the object removal.
$u_c = \frac{i_1+i_2}{2}$, $v_c = \frac{j_1+j_2}{2}$	are the coordinates of the center of A
μ_i, $i = 1, \ldots, M$	is the left-most column carved considering all the seams passing from row i, see Fig. 6.25(c)
η_i, $i = 1, \ldots, M$	is the right-most column carved considering all the seams passing from row i, see Fig. 6.25(c)
c_r	is the number of columns spanned by the object= number of seams carved

Here is how the object removal algorithm works:

1. The input is an $M \times N$ image I.

2. The user selects a zone A from the image I as a rectangular bounding box for the object that must be removed, as shown in Fig. 6.25(b).

3. In the energy map E, which is the starting point for the computation of the optimal seam, we set $E(i, j)$ to a very low value, for pixels $(i, j) \in A$.

4. A given seam will be vertical in the object removal region, i.e. in the zone A. Therefore, if the seam passes through the j^{th} column in the original image in the object removal zone, then the seam pixels corresponding to the object removal region will be $\{I(i, j)\}_{i=i_1}^{i_2}$. The seam pixels corresponding to the regions outside the object removal region need not be vertical and the total seam I_s (when the seam passes through the j^{th} column in the original image as it traverses A) can be expressed as:
   $I_s = \{I(i, x(i))\}_{i=1}^{i_1-1} \cup \{I(i, j)\}_{i=i_1}^{i_2} \cup \{\mathbf{I}(i, x(i))\}_{i=i_2+1}^{M}$
   where $x(i)$ maps the seam pixel in the i^{th} row to one of the N columns.

5. This process iterates until all the pixels in the region of interest A are removed, i.e. all c_r seams are removed. Fig. 6.25(d) shows the image after seam carving.

6. It is possible to resize the image to the original width, inserting exactly c_r new seams, as shown in Fig. 6.25(f).

For a $M \times N$ image with c_r seams being removed, the resultant image is of size $M \times (N - c_r)$. We create a $M \times (N - c_r)$ binary matrix R where 1/0 denotes whether/or not an element in R has been affected by seam carving (6.14). To estimate the output of the algorithm, for the i^{th} row in the image I, we define μ_i and η_i as the first and last column, respectively, to be affected by the tampering (for example, first and last pixel in the i^{th} row where a seam has been removed), and c_r is the number of columns removed. We set an element in R to 1/0 as follows:

$$R(i,j) \begin{array}{ll} = 1 & \text{if } \mu_i < j < \eta_i - c_r \\ = 0 & \text{otherwise} \end{array} \tag{6.14}$$

Previous Work on Object Removal Localization: Generally, in copy-move forgeries for object removal (Fig. 6.1(b)), the portion of the image to be hidden is covered with another portion of the same image. This kind of forgery is usually detected identifying duplicated regions in the image [86, 124]. In seam carving, there is no duplicated information in the image because the object has been actually eliminated instead of being hidden with copy-move or inpainting techniques.

6.5.2 Multi-classifier based Approach to Tamper Localization

In Section 6.5.1, we explained object removal using seam carving. We briefly explain how this algorithm affects the image. Seam carving in its original version spreads the effect of the "columns removal" throughout the whole width of the image, as shown in Fig. 6.26(a). On the other hand, when a specific area of the image is cut via seam carving (as is the case for object removal), the zones affected by the resizing algorithm are concentrated in that region. In Fig. 6.26 we observe the behavior of the seams when they contribute to the object removal. For the actual region where the object was removed, all the removed seams will be concentrated in a single region due to the negative mask being applied in the energy domain in the object removal process. Progressively, while moving away from the object removal region, they start following the path that minimizes the energy function until they reach the border of the image, as shown in Fig. 6.26(c).

We know from the previous section that the Shi-324 feature helps to identify inconsistencies in transitions between neighboring pixels (6.7), because it depends on the local DCT frequencies arising out of change in the inter-pixel co-occurrence matrix. When a seam is removed, the left and right neighbors are changed for the seam pixels and new local higher frequency terms can be introduced.

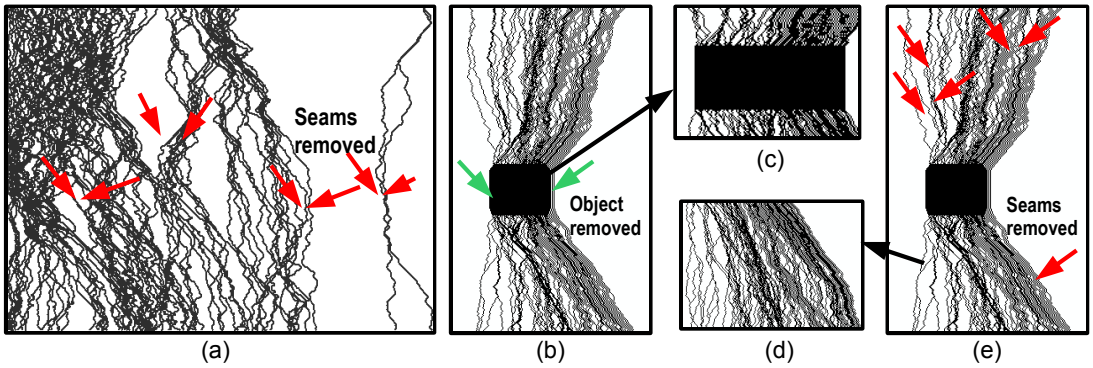


Figure 6.26: Zones to be removed by seam carving in its traditional use for content-aware image resizing are shown in (a). The seams which are to be removed for object removal are shown in (b-d). In (b) the arrows highlight how in the object area all the seams are concentrated in one chunk; see a zoomed image in (c). In (e) the arrows show the scattered seams to be removed outside the object area, and a zoomed-in version is shown in (d).

If more columns are deleted, the image will show more inconsistencies in such transitions, and the feature will change more compared to an untampered image, (see Fig. 6.26(a)). On the other hand, when a large number of consecutive columns are removed (for example, where the actual object is removed from), the pixel neighbors change for only 2 pixels for a row, as shown in Figs. 6.26(b) and (c).

Identification of the three classes: Using the Markov feature, we use a block-based approach to identify the specific area where the object has been removed. We divide the image into blocks, compute the **Shi-324** feature per block, and use them as the input for the SVM-based classifier framework. Combining the different SVM outputs, we can then find the area where it is most likely that the object has been deleted from. There are 9 zones resulting from dividing the

image grid into 3 rows and 3 columns, as shown in Fig. 6.27(a) With our previous observations, we classify a zone to be one of 3 types (an example is in Fig. 6.27(b)).

1. Class 1: it corresponds to the zone through which no/few seams pass. In Fig. 6.27(b), this area is marked with a blue cross.

2. Class 2: it is shown in Fig. 6.27(b) with a black cross. For regions lying vertically above or below the region where object removal occurred, the seams that are removed are more scattered and hence, there is a significant change in the Markov features after seam carving.

3. Class 3: it corresponds to the zone which contains the object that is removed, and the seams that pass through this object will be placed very closely. Since seams belonging to many consecutive columns are removed, the change in the Markov feature is less for such regions, see pink cross in Fig. 6.27(b).

Two-phase approach: We use two stages to identify the tampered block, as shown in Fig. 6.27.

- Consider a $M \times N$ image I where object removal was performed.

- Three vertical $M \times (N/3)$ slices are taken from the image, we call them B_1, B_2, and B_3. The **Shi-324** feature is extracted from each of the blocks.

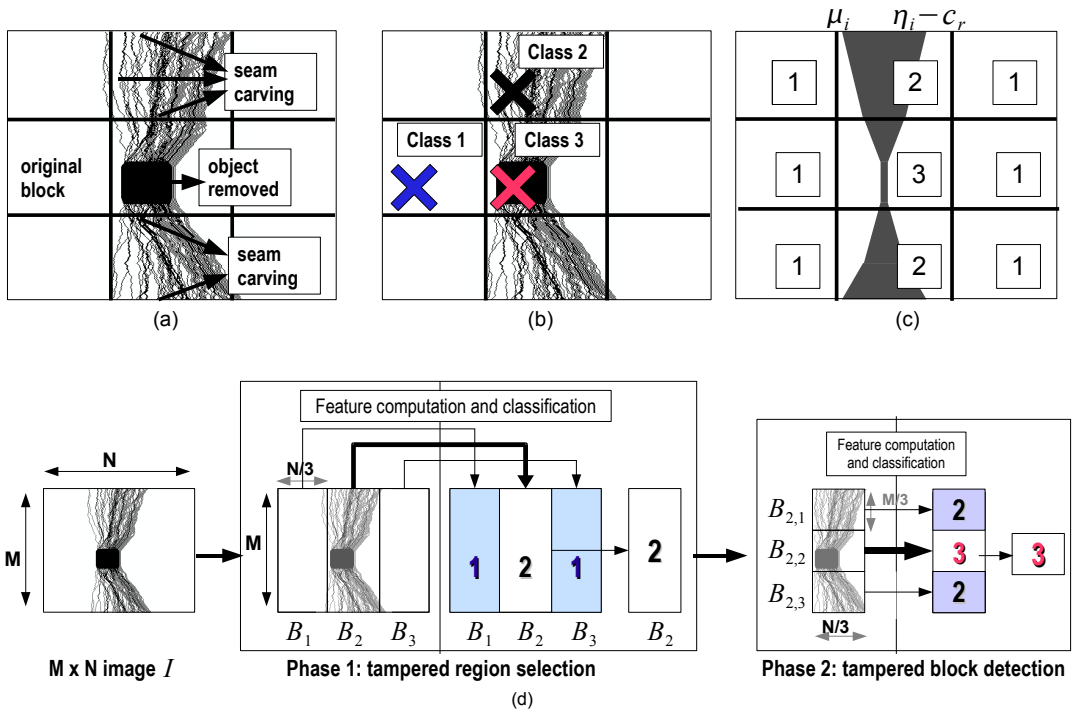


Figure 6.27: Block based approach: the division of the image into blocks of different types is shown in (a), the identification of the three classes is shown in (b), the zones affected by the seam carving and consequent labeling of the different zones for the two-phase approach is shown in (c). Figure (d) shows the two-phase approach: 1) each vertical slice is classified as tampered(containing object removal region)/untampered. 2) the vertical slice is divided into horizontal chunks for more precise localization of the object removal region.

- **First classification**: each B_b is classified as tampered/untampered block (Types 1 and 2). To properly train this classifier, we label the slice, where there is a higher concentration of zones affected by seam carving, as tampered.

- Three horizontal $(M/3) \times (N/3)$ chunks are taken from the vertical slice which was classified as tampered, i.e. considering $B_{b,1}$, $B_{b,2}$, and $B_{b,3}$.

- **Second classification**: the SVM-based classifier labels each $B_{i,j}$ block as belonging to Class 2 or 3.

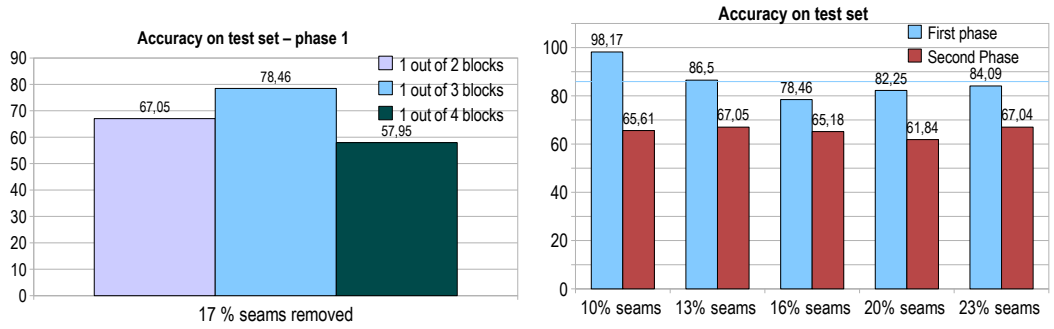
- The $(M/3) \times (N/3)$ sized block, classified as class 3 in the second phase, is then considered as tampered, i.e. where the object has been removed from.

6.5.3 Results for Object Removal Localization

The question comes up as to when we should localize the object removal region, the zone can be one out of 2×2, 3×3 or 4×4 zones in the image. If the zone is too large, then the distinction between regions having no seams or many consecutive seams may be blurred. If the zone is too small, then there may be too less information (the Markov feature learnt per zone may be inconsistent for a small enough zone) for proper classification.

As shown in Fig. 6.28(a), our experiments in the first phase showed that the efficiency increases when we consider 3×3 = 9 blocks for each image. We took 800 images from the UCID database [3] (each image is of size 384×512) and we performed object removal with seam carving in all of them. Our experiments showed that object removal with seam carving causes a degradation of the image quality when a large number of seams is removed. Therefore, we considered the removal of only small objects. We vary the fraction of seams (object width) that is

[3] http://vision.cs.aston.ac.uk/datasets/UCID/ucid.html



(a) (b)

Figure 6.28: Classification accuracy for (a) identifying the tampered vertical slice from out of 3 slices (b) identifying the tampered zone from out of 3×3=9 zones.

removed from the image. We removed 10%-23% of the columns in each image. We divide the dataset into training and testing images. We now divide each image of the training set into three vertical slices and we label as "class 2", for each image, the block that is more affected by the object removal (see (6.14)), and "class 1" the block where less seams have been removed. A SVM-based model is learnt from the training images using the **Shi-324** feature. With the same approach, we label one of the three horizontal blocks that result from the division of the slice detected as tampered in phase 1. Each image was JPEG compressed at a quality factor of 100 before the two-phase approach.

6.5.4 Summary

We have presented a machine learning based approach where Markov features in the quantized DCT domain have been shown to be useful for detecting seam carving and seam insertions. We have also proposed an algorithm which exploits the linear relationship between pixels located on/near the seam to detect seam insertions. Assuming prior knowledge of the seam insertion algorithm, we obtain highly accurate localization of the inserted seams. The Expectation-Maximization based probabilistic framework, which does not use explicit knowledge of the seam insertion algorithm, has a much reduced accuracy.

Future Work: Possible future research directions include making the seam insertion framework more general so that the newly introduced pixels can be any arbitrary combination of neighboring pixels (and not just the average). Also, the machine learning based approach needs to be further improved upon to increase the detection rate when the seam carving/insertion percentage is low enough. For object removal, the preliminary results using a nine-fold (3 horizontal and 3 vertical levels) partitioning of the image are encouraging but the object removal region can be identified more precisely through better feature selection and learning strategies.

In this chapter, we have presented results for tamper localization for uncompressed images. There is a large body of work for tamper detection in compressed

images also. Normally, tampered regions in a JPEG image show traces of double compression. The double quantization (DQ) effect is used to identify tampered regions when the first quality factor QF_1 (corresponds to the original quality factor of the image from which the tampered region was taken) is lower compared to the second quality factor QF_2 (at which the final tampered image is advertised). Farid et al. show in [87] that double compression introduces periodic artifacts in the histogram of the DCT coefficients of a given frequency. The reverse problem of tamper localization when QF_1 is greater than QF_2 has not been well-studied from a forensics perspective. Also, for the $QF_1 < QF_2$ case, the tamper localization method assumes that the image grid remains unchanged after tampering. When the image is subjected to seam carving/insertion based attacks, the image grid is modified and it becomes difficult to localize the tampering.

Chapter 7

Conclusions and Future Work

This dissertation has been mainly focussed on data hiding and steganography, with an extension into forensics. Though watermarking, the basic field based on which data hiding was developed, has been very well studied, the design of robust enough hiding methods for a wide plethora of attacks presents several challenges. For such robust hiding methods, the hiding capacity is still small enough and future research can focus on achieving higher data-rates. Also, considering the two competing fields of steganography and steganalysis, the main research thrust has been more towards accurate steganalysis methods. Hence, while we have techniques like self-calibration which can detect most of the JPEG-based hiding schemes, the concept of a universal stegangraphic scheme, or one whose security has been demonstrated across a wide class of detection methods at practical hiding

rates, is still far from reality. With YASS, we attempted to resist detection using the self-calibration method. More recently, newer methods have been proposed with both calibrated and uncalibrated features [58] - the uncalibrated features are tailored towards detecting YASS which has been shown to be secure against calibrated feature based detection. It would indeed be a challenge to come up with a stego scheme which resists calibration based detection but yet is not easily detected by the complementary set of (uncalibrated) features.

Robust Key-point based Hiding: Our hiding method relies on the accurate estimation of the 2×3 affine transformation where we estimate the 2×2 transformation terms while the shift does not affect the data extraction, if the 2×2 matrix based alignment is successful. The hiding method can be extended to account for non-linear transformations by considering the transformation between local regions (building upon a similar approach by Voloshynovskiy et al. [116]) and assuming a non-linear transformation to be a summation of different linear transformations, when considered over small enough image regions. Also, to ensure that the receiver has access to the same local regions used for hiding as the sender, we look to improve the key-point pruning strategy. There are many popular key-point detection methods and achieving proper synchronization without considering all the points requires a good pruning strategy. Also, salient region based methods (as in MSER - Maximally Stable Extremal Regions [26]) can be considered to

obtain common regions between the encoder and the decoder, instead of using key-points to locate the common regions.

One concept that can be further researched upon is embedding exactly the same data in all the local hiding regions. This was done with an aim to resist cropping. The drawback is that even if we retrieve more than one local region correctly, we end up recovering exactly the same data-bits because of the message repetition. One may think of more intelligent coding schemes where different parts of the message can be embedded in different regions and the recovery of the total message may not be needed to be able to decode the sub-parts. The issue with our error correction framework (RA code) is that it performs better for longer messages. Hence, other coding schemes may be more useful for shorter codewords, which come into play for the part-based data recovery.

YASS - Randomized Block based Hiding: YASS has been highly effective in resisting blind steganalysis by using randomized block based features. This ensures that there is de-synchronization between the steganalysis features which assume a regular JPEG block based hiding and those features resulting from randomized block-based hiding. The basic concept why YASS is undetectable is because of "randomization". Even if the steganalysis knows the randomized block based hiding method, he cannot reproduce the exact hiding scheme as the blocks are pseudo-randomly selected (based on a shared secret key) - our experiments

suggest that even if the steganalyst learns SVM models based on a training set of cover and stego (using YASS based hiding and knowing the correct value of big-block size B) images, the hiding is generally undetectable, for low enough hiding rates and on using matrix embedding like embedding strategies. The same "randomization" concept can be extended to multiple parameters - transform domains used for hiding, hiding parameters (different number of coefficients per hiding block and varying quality factor used for hiding). For the variant of YASS where we vary the quality factor used for hiding depending on the block-based variance, we do use further randomization to improve the performance.

Matrix embedding (ME): ME has been shown to be superior to quantization index modulation (QIM) based embedding strategies in the hiding rate vs detectability trade-off. The main problem that has been encounted is the shrinkage problem, where given a coefficient valued at zero, it is difficult to identify whether it corresponds to an embedding or an erasure. Wet paper codes [38, 39] can be used to further improve the hiding rate and better tackle the shrinkage problem. Also, for matrix embedding, we have used (7,4) Hamming codes as our inner codes. The LLR computation algorithm works for smaller length inner codes: the computational complexity varies as 2^k for a $(2^k - 1, k)$ code where k bits are embedded by changing at most one out of $(2^k - 1)$ image coefficients. In

the future, other LLR computation methods can be experimented with, the goal being that they should scale better for longer code-words (larger values of k).

Image Forensics: Our method of robust re-sampling detection can be extended to other tamper detection methods which work for uncompressed images but are not robust against JPEG compression. The basic philosophy of controlled noise addition also needs to be theoretically analyzed further so as to determine the noise distribution along with the associated noise levels which can lead to better suppression of the JPEG induced periodicity while maintaining the basic tamper related artifacts (for example, re-sampling induced patterns).

Another problem is detecting and localizing object removal. We have made some preliminary progress in seam carving based tamper localization, specially for the object removal problem. There is much potential for extending our tamper localization method for JPEG compressed images and also refine the precision with which the tampered area is localized.

Generic Improvements: There are certain issues which apply to data hiding in general. We have empirically observed that the DCT domain is better suited for hiding (maintains perceptual transparency and allows data recovery) than wavelet domain coefficients. We have compared the performance of matrix embedding with that of scalar QIM. It would be instructive to compare the benefits of vector or lattice QIM with that of matrix embedding. Also, we have used

repeat accumulate coding for our error correction setup. We have experimented with various low density parity check (LDPC) codes and our experiments suggest the superiority of RA codes. There has been research on designing the "optimal" channel codes for a given channel [90]. An optimal choice of the channel code will help to enhance the hiding rate (RA codes are a good choice for channel codes for high erasure-rate channels, which is the case for our data hiding channels, but they are still sub-optimal).

Overall Contributions: Our main contributions have been providing a deeper insight of steganographic and steganalysis methods, identifying the main aspects based on which hiding is detected, and then resisting detection using randomized hiding (YASS). For better trade-off between detectability and hiding rate, we use techniques like matrix embedding. While our steganographic methods are mainly robust against global image attacks, we propose key-point based hiding methods which are also robust against local and geometrical attacks. In the field of forensics, our contributions include robust re-sampling detection and detecting and localizing seam-carving based tampering.

Bibliography

[1] "http://scien.stanford.edu/class/psych221/projects/00/shuoyen/," overview of JPEG 2000 and Wavelet Compression - Stanford University.

[2] "http://vision.ece.ucsb.edu/publications/Sarkar_tech_report_estimate_q.pdf."

[3] "http://vision.ece.ucsb.edu/publications/Sarkar_tech_report_LLR_alpha.pdf."

[4] "http://www.cs.dartmouth.edu/farid/research/steg.m," code for generating wavelet-based feature vectors for steganalysis, using both coefficient and error statistics.

[5] A. Abbasfar, *Turbo-like Codes Design for High Speed Decoding.* Springer, 2007.

[6] A. Abbasfar, D. Divsalar, K. Yao, R. Inc, and C. Los Altos, "Accumulate-repeat-accumulate codes," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 692–702, 2007.

[7] M. Alghoniemy and A. Tewfik, "Geometric invariance in image watermarking," *IEEE Transactions on Image Processing*, vol. 13, no. 2, pp. 145–153, 2004.

[8] I. Avcibas, B. Sankur, and N. Memon, "Image steganalysis with binary similarity measures," in *Proc. ICIP*, 2002, pp. 645–648.

[9] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Trans. Graph.*, vol. 26, no. 3, p. 10, 2007.

[10] M. Backes and C. Cachin, "Public-key steganography with active attacks," in *Theory of Cryptography Conference Proceedings*, vol. 3378. Springer, 2005, pp. 210–226.

[11] M. Barni and F. Bartolini, *Watermarking systems engineering: Enabling digital assets security and other applications.* CRC Press, 2004.

[12] P. Bas, J. m. Chassery, and B. Macq, "Geometrically invariant watermarking using feature points," *IEEE Trans. on Image Processing*, vol. 11, pp. 1014–1028, 2002.

[13] C. Cachin, "An information theoretic model for steganography," *LNCS: 2nd Int'l Workshop on Info. Hiding*, vol. 1525, pp. 306–318, 1998.

[14] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 679–698, 1986.

[15] R. Chandramouli and N. Memon, "Analysis of LSB based image steganography techniques," in *Proc. ICIP*, Oct. 2001, pp. III–1019–22.

[16] B. Chen and G. W. Wornell, "Quantization Index Modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. on Info. Theory*, vol. 47, no. 4, pp. 1423–1443, May 2001.

[17] C. Chen and Y. Q. Shi, "JPEG image steganalysis utilizing both intrablock and interblock correlations," in *Proc. of International Symposium on Circuits and Systems (ISCAS)*, May 2008, pp. 3029–3032.

[18] P. Chose, A. Joux, and M. Mitton, "Fast correlation attacks: An algorithmic point of view," *Lecture Notes in Computer Science*, pp. 209–221, 2002.

[19] A. S. Cohen and A. Lapidoth, "The Gaussian watermarking game," *IEEE Trans. on Info. Theory*, vol. 48, no. 6, pp. 1639–1667, Jun. 2002.

[20] R. Crandall, *Some Notes on Steganography.* Posted on Steganography mailing List, http://os.inf.tu-dresden.de/∼westfeld/crandall.pdf, 1998.

[21] S. Craver, *On public-key steganography in the presence of an active warden.* Springer, 1997.

[22] O. Dabeer, K. Sullivan, U. Madhow, S. Chandrasekaran, and B. Manjunath, "Detection of hiding in the least significant bit," *IEEE Transactions on Signal Processing, Supplement on Secure Media I*, vol. 52, no. 10, pp. 3046–3058, Oct 2004.

[23] R. de Queiroz, C. K. Choi, Y. Huh, and K. R. Rao, "Wavelet transforms in a JPEG-like image coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 419–424, Apr 1997.

[24] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 99, no. 1, pp. 1–38, 1977.

[25] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for turbo-like codes," in *36th Allerton Conf. on Communications, Control, and Computing*, Sep. 1998, pp. 201–210.

[26] M. Donoser and H. Bischof, "Efficient maximally stable extremal region (MSER) tracking," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2006, pp. 553–560.

[27] J. J. Eggers, R. Bauml, and B. Girod, "A communications approach to image steganography," in *Proc. of SPIE*, San Jose, CA, 2002.

[28] R. A. Emmert and C. D. McGillem, "Multitemporal geometric distortion correction utilizing the affine transformation," in *Proc. of IEEE Conf. on Machine Processing Remotely Sensed Data*, Oct 1973, pp. 1B–24 – 1B–32.

[29] D. F., Voloshynovskiy, and P. T., "A method for the estimation and recovering from general affine transforms in digital watermarking applications," in *Proceedings of SPIE - Security and watermarking of multimedia contents IV*, vol. 4675, 2002, pp. 313–322.

[30] H. Farid, "Exposing digital forgeries from JPEG ghosts," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 4, pp. 154–160, 2009.

[31] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil, "Eliminating steganography in Internet traffic with active wardens," *Lecture notes in computer science*, pp. 18–35, 2003.

[32] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and center of circular features," in *Proc. of ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland*, June 2-4 1987, pp. 281–305.

[33] E. Franz, "Steganography preserving statistical properties," in *5th International Working Conference on Communication and Multimedia Security*, 2002.

[34] J. Fridrich, "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes," in *6th International Workshop on Information Hiding*, Toronoto, Canada, 2004, pp. 67–81.

[35] ——, "Minimizing the embedding impact in steganography," in *MM&Sec '06: Proceedings of the 8th Workshop on Multimedia and Security*. New York, NY, USA: ACM, 2006, pp. 2–10.

[36] J. Fridrich, M. Goljan, D. Hogea, and D. Soukal, "Quantitative steganalysis of digital images: Estimating the secret message length," *ACM Multimedia Systems Journal, Special issue on Multimedia Security*, vol. 9, no. 3, pp. 288–302, 2003.

[37] J. Fridrich, M. Goljan, P. Lisonek, and D. Soukal, "Writing on wet paper," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3923–3935, 2005.

[38] J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography with wet paper codes," in *MM&Sec '04: Proceedings of the 2004 Workshop on Multimedia and Security*. New York, NY, USA: ACM, 2004, pp. 4–15.

[39] ——, "Wet paper codes with improved embedding efficiency," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 1, pp. 102–110, March 2006.

[40] J. Fridrich, T. Pevny, and J. Kodovský, "Statistically undetectable JPEG steganography: Dead ends, challenges, and opportunities," in *Proc. ACM*, Sep. 2007, pp. 3–14.

[41] J. Fridrich and D. Soukal, "Matrix embedding for large payloads," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 3, pp. 390–395, Sept. 2006.

[42] D. Fu, Y. Q. Shi, D. Zou, and G. Xuan, "JPEG steganalysis using empirical transition matrix in block dct domain," in *International Workshop on Multimedia Signal Processing*, Victoria, BC, Canada, Oct. 2006.

[43] A. Gallagher, "Method for detecting for image interpolation," in *US Patent No. US 7,251,378, B2*, October 2004.

[44] ——, "Detection of linear and cubic interpolation in JPEG compressed images," in *Computer and Robot Vision, 2005. Proc. The 2nd Canadian Conference on*, May 2005, pp. 65–72.

[45] P. Guillon, T. Furon, and P. Duhamel, "Applied public-key steganography," in *Proc. of SPIE*, San Jose, CA, 2002.

[46] J. J. Harmsen and W. A. Pearlman, "Steganalysis of additive noise modelable information hiding," in *Proc. of SPIE*, Jan. 2003, pp. 131–142.

[47] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[48] A. Herrigel, S. Voloshynovskiy, and Y. Rytsar, "The watermark template attack," in *Proc. of SPIE*, San Jose, CA, Jan. 2001, pp. 394–405.

[49] S. Hetzl and P. Mutzel, "A graph theoretic approach to steganography," in *9th IFIP TC-6 TC-11 International Conference, Communications and Multimedia Security*, vol. 3677, Salzburg, Austria, 2005, pp. 119–128.

[50] S. Hu, "Geometric-invariant image watermarking by key-dependent triangulation," *Informatics in education*, vol. 32, pp. 169–182, 2008.

[51] N. Jacobsen, K. Solanki, U. Madhow, B. Manjunath, and S. Chandrasekaran, "Image adaptive high volume data hiding based on scalar quantization," in *Proc. IEEE Military Comm. Conf. (MILCOM)*, vol. 1, 2002, pp. 411–415.

[52] M. Johnson and H. Farid, "Exposing digital forgeries by detecting inconsistencies in lighting," in *7th ACM Multimedia and Security Workshop*, New York, NY, 2005. [Online]. Available: www.cs.dartmouth.edu/farid/publications/acm05.html

[53] S. Katzenbeisser and F. A. P. Petitcolas, "On defining security in steganographic systems," in *Proc. of SPIE*, Jan. 2002, pp. 50–56.

[54] M. Kharrazi, H. Sencar, and N. Memon, "Image steganography: Concepts and practice," *Lecture Note Series, Institute for Mathematical Sciences, National University of Singapore*, 2004.

[55] Y. Kim, Z. Duric, and D. Richards, "Modified matrix encoding technique for minimal distortion," in *Lecture notes in computer science: 8th International Workshop on Information Hiding*, July 2006, pp. 314–327.

[56] M. Kirchner, "On the detectability of local resampling in digital images," vol. 6819, no. 1. SPIE, 2008, p. 68190F.

[57] J. Kodovský and J. Fridrich, "Influence of embedding strategies on security of steganographic methods in the JPEG domain," in *Proc. of SPIE*, San Jose, CA, Jan. 2008, pp. 2.1 – 2.13.

[58] ——, "Calibration revisited," in *MM & Sec '09: Proceedings of the 11th ACM workshop on Multimedia and security*. New York, NY, USA: ACM, 2009, pp. 63–74.

[59] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Info. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[60] M. Kutter, "Watermarking resisting to translation, rotation, and scaling," in *Proceedings of SPIE- The International Society for Optical Engineering*, vol. 3528, 1999, pp. 423–431.

[61] H. Y. Lee, H. Kim, and H. K. Lee, "Robust image watermarking using local invariant features," *SPIE, Journal of Optical Engineering*, vol. 45(3), pp. 1–11, 2006.

[62] Y. Lee, H. Kim, and H. Park, "Blocking effect reduction of JPEG images by signal adaptive filtering," *Image Processing, IEEE Tran. on*, vol. 7, no. 2, pp. 229–234, Feb 1998.

[63] B. Li, Y. Shi, and J. Huang, "Steganalysis of YASS," in *Proceedings of the 10th ACM workshop on Multimedia and security*. ACM New York, NY, USA, 2008, pp. 139–148.

[64] W. Li, Y. Yuan, and N. Yu, "Detecting copy-paste forgery of JPEG image via block artifact grid extraction."

[65] C. Lin, M. Wu, J. A. Bloom, I. J. Cox, M. L. Miller, and Y. M. Lui, "Rotation, scale and translation resilient watermarking for images," *IEEE Trans. Image Processing*, vol. 10, pp. 767–782, 2001.

[66] Z. Lin, J. He, X. Tang, and C. Tang, "Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492 – 2501, 2009.

[67] Y. Liu, D. Zheng, and J. Zhao, "A rectification scheme for RST invariant image watermarking," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, no. 1, pp. 314–318, 2005.

[68] D. Lowe, "Distinctive image features from scale-invariant keypoints," in *International Journal of Computer Vision*, vol. 20, 2003, pp. 91–110. [Online]. Available: citeseer.ist.psu.edu/lowe04distinctive.html

[69] P. Lu and D. Wang, "A Geometrical Robust Image Data Hiding Scheme Using FCA-Based Resynchronization," *Lecture Notes in Computer Science*, vol. 4567, pp. 267–278, 2007.

[70] S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," in *5th Int. Workshop on Info. Hiding*, 2002, pp. 340–354.

[71] B. Mahdian and S. Saic, "Blind authentication using periodic properties of interpolation," *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 3, pp. 529–538, Sept. 2008.

[72] M. Mese and P. Vaidyanathan, "Optimal histogram modification with MSE metric," in *Proc. ICASSP*, May 2001, pp. III–1665–68.

[73] P. Moulin and J. A. O'Sullivan, "Information-theoretic analysis of information hiding," *IEEE Trans. on Info. Theory*, vol. 49, no. 3, pp. 563–593, Mar. 2003.

[74] L. Nataraj, A. Sarkar, and B. S. Manjunath, "Adding Gaussian Noise to "Denoise" JPEG for detecting Image Resizing," in *International Conference on Image Processing, 2009*, Nov 2009, pp. 1493–1496.

[75] T. Ng, S. Chang, and Q. Sun, "Blind detection of photomontage using higher order statistics," in *IEEE International Symposium on Circuits and Systems*, vol. 5. Citeseer, 2004, pp. 688–691.

[76] A. Nikolaidis and I. Pitas, "Region-based image watermarking," *IEEE Transactions on Image Processing*, vol. 10, no. 11, pp. 1726–1740, 2001.

[77] A. Noble, "Descriptions of image surfaces," Ph.D. dissertation, Department of Engineering Science, Oxford University, 1989.

[78] A. Nosratinia, "Denoising of JPEG images by re-application of JPEG," *Journal of VLSI Signal Processing*, vol. 27, pp. 69–79, 2001.

[79] S. Pei and C. Lin, "Image normalization for pattern recognition," *Image and Vision computing*, vol. 13, no. 10, pp. 711–723, 1995.

[80] S. Pereira, S. Voloshynovskiy, M. Madueno, S. Marchand-Maillet, and T. Pun, "Second generation benchmarking and application oriented evaluation," in *Information Hiding: 4th International Workshop, IH 2001, Pittsburgh, PA, USA, April 25-27, 2001: Proceedings*. Springer, 2001, p. 340.

[81] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding — A survey," *Proceedings of the IEEE, special issue on Identification and Protection of Multimedia Information*, vol. 87, no. 7, pp. 1062–1078, 1999. [Online]. Available: citeseer.nj.nec.com/petitcolas99information.html

[82] T. Pevny and J. Fridrich, "Multi-class blind steganalysis for JPEG images," in *Proc. of SPIE*, San Jose, CA, 2006, pp. 257–269.

[83] ——, "Merging Markov and DCT features for multi-class JPEG steganalysis," in *Proc. of SPIE*, San Jose, CA, 2007, pp. 3.1 – 3.14.

[84] S. Planjery and T. Gulliver, "Design of Rate-Compatible Punctured Repeat-Accumulate Codes," in *IEEE Global Telecommunications Conference, 2007. GLOBECOM'07*, 2007, pp. 1482–1487.

[85] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of re-sampling," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 758–767, 2005. [Online]. Available: www.cs.dartmouth.edu/farid/publications/sp05.html

[86] A. Popescu and H. Farid, "Exposing digital forgeries by detecting duplicated image regions," Department of Computer Science, Dartmouth College, Tech. Rep. TR2004-515, 2004. [Online]. Available: www.cs.dartmouth.edu/farid/publications/tr04.html

[87] ——, "Statistical tools for digital forensics," in *6th International Workshop on Information Hiding*. Springer, 2004, pp. 128–147.

[88] ——, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, 2005. [Online]. Available: www.cs.dartmouth.edu/farid/publications/sp05a.html

[89] N. Provos, "Defending against statistical steganalysis," in *10th USENIX Security Symposium*, vol. 10, Washington DC, USA, 2001, pp. 323–336.

[90] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-checkcodes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[91] J. K. O. Ruanaidh and T. Pun, "Rotation, scale and translation invariant spread spectrum digital image watermarking," *Signal Processing*, vol. 66, no. 3, pp. 303–317, May 1998.

[92] Y. Rubner, C. Tomassi, and L. J. Guibas, "The Earth Mover's Distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[93] P. Sallee, "Model-based steganography," in *IWDW 2003, LNCS 2939*, Oct. 2003, pp. 154–167.

[94] ——, "Model-based methods for steganography and steganalysis," *International Journal of Image Graphics*, vol. 5, no. 1, pp. 167–190, 2005.

[95] A. Sarkar, U. Madhow, and B. S. Manjunath, "Matrix embedding with pseudorandom coefficient selection and error correction for robust and secure steganography," *submitted to IEEE Trans. on Information Forensics and Security*, July 2009.

[96] A. Sarkar and B. S. Manjunath, "Estimating steganographic capacity for odd-even based embedding and its use in individual compensation," in *IEEE International Conference on Image Processing (ICIP)*, vol. 1, San Antonio, TX, Sep 2007, pp. 409–412.

[97] A. Sarkar, L. Nataraj, B. S. Manjunath, and U. Madhow, "Estimation of optimum coding redundancy and frequency domain analysis of attacks for YASS - a randomized block based hiding scheme," in *Proc. of ICIP*, Oct 2008, pp. 1292–1295.

[98] A. Sarkar, K. Solanki, U. Madhow, S. Chandrasekaran, and B. S. Manjunath, "Secure steganography: Statistical restoration of the second order dependencies for improved security," in *Proc. ICASSP*, vol. 2, Apr. 2007, pp. II–277–II–280.

[99] A. Sarkar, K. Solanki, and B. S. Manjunath, "Further study on YASS: Steganography based on randomized embedding to resist blind steganalysis," in *Proc. of SPIE: Security, Steganography, and Watermarking of Multimedia Contents X*, vol. 6819, San Jose, CA, Jan. 2008, pp. 16–31.

[100] A. Sarkar, K. Sullivan, and B. S. Manjunath, "Steganographic capacity estimation for the statistical restoration framework," in *Proc. of SPIE*, vol. 6819, San Jose, CA, Jan. 2008, pp. 681 916–1–681 916–11.

[101] A. Sarkar, L. Nataraj, and B. S. Manjunath, "Detection of seam carving and localization of seam insertions in digital images," in *11th ACM Workshop on Multimedia and Security*, Sep 2009, pp. 107–116.

[102] P. Shelby and P. Thierry, "Robust template matching for affine resistant image watermarks," *IEEE Transaction on Image Processing*, vol. 9, no. 6, pp. 1123–1129, 2000.

[103] Y. Q. Shi, C. Chen, and W. Chen, "A Markov process based approach to effective attacking JPEG steganography," in *Lecture notes in computer science: 8th International Workshop on Information Hiding*, July 2006, pp. 249–264.

[104] D. Simitopoulos, D. Koutsonanos, and M. Strintzis, "Robust image watermarking based on generalized Radon transformations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 732–745, 2003.

[105] K. Solanki, O. Dabeer, U. Madhow, B. S. Manjunath, and S. Chandrasekaran, "Robust image-adaptive data hiding: Modeling, source coding and channel coding," in *42nd Annual Allerton Conf. on Communications, Control, and Computing*, vol. 41, no. 2, Oct. 2003, pp. 829–838.

[106] K. Solanki, N. Jacobsen, S. Chandrasekaran, U. Madhow, and B. S. Manjunath, "High-volume data hiding in images: Introducing perceptual criteria into quantization based embedding," in *Proc. ICASSP*, vol. 4, Orlando, FL, USA, May 2002, pp. 3485–3488.

[107] K. Solanki, N. Jacobsen, U. Madhow, B. S. Manjunath, and S. Chandrasekaran, "Robust image-adaptive data hiding based on erasure and error correction," *IEEE Trans. on Image Processing*, vol. 13, no. 12, pp. 1627 –1639, Dec 2004.

[108] K. Solanki, A. Sarkar, and B. S. Manjunath, "YASS: Yet Another Steganographic Scheme that resists blind steganalysis," in *9th International Workshop on Information Hiding*, Jun 2007, pp. 16–31.

[109] K. Solanki, K. Sullivan, U. Madhow, B. S. Manjunath, and S. Chandrasekaran, "Statistical restoration for robust and secure steganography," in *Proc. ICIP*, 2005, pp. II–1118–21.

[110] ——, "Provably secure steganography: Achieving zero K-L divergence using statistical restoration," in *Proc. ICIP*, 2006, pp. 125–128.

[111] K. Sullivan, U. Madhow, S. Chandrasekaran, and B. Manjunath, "Steganalysis for markov cover data with applications to images," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 275–287, Jun 2006. [Online]. Available: http://vision.ece.ucsb.edu/publications/sullivan_TIFS06.pdf

[112] K. Sullivan, K. Solanki, U. Madhow, B. S. Manjunath, and S. Chandrasekaran, "Determing achievable rates for secure, zero-divergence, steganography," in *Proc. ICIP*, 2006, pp. 121–124.

[113] C. Tang and H. Hang, "A feature-based robust digital image watermarking scheme," *IEEE transactions on signal processing*, vol. 51, no. 4, pp. 950–959, 2003.

[114] M. Tone and N. Hamada, "Affine invariant digital image watermarking using feature points," in *International Workshop on Nonlinear Circuit and Signal Processing*, 2005, pp. 117–120.

[115] R. Tzschoppe, R. Bauml, and J. Eggers, "Histogram modifications with minimum MSE distortion," Dec 2001, tech. Rep., Telecom. Lab., Univ. of Erlangen-Nuremberg.

[116] S. Voloshynovskiy, F. Deguillaume, and T. Pun, "Multibit digital watermarking robust against local nonlinear geometrical distortions," in *International Conference on Image Processing*, vol. 3, 2001, pp. 999–1002.

[117] D. Wang and P. Lu, "A Novel Geometrical Robust Image Data Hiding Scheme," in *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS'07. Eighth International Workshop on*, 2007, pp. 59–62.

[118] X. Wang and J. Wu, "A feature-based robust digital image watermarking against desynchronization attacks," *International Journal on Automation and Computing*, vol. 4, pp. 428–432, 2007.

[119] Y. Wang and P. Moulin, "Steganalysis of block-DCT image steganography," in *IEEE workshop on Statistical Signal Processing*, St Louis, MO, USA, Sep. 2003.

[120] ——, "Steganalysis of block-structured stegotext," in *Proc. of SPIE*, vol. 5306, San Jose, CA, 2004, pp. 477–488.

[121] ——, "Optimized feature extraction for learning-based image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, pp. 31–45, March 2007.

[122] J. Weinheimer, X. Qi, and J. Qi, "Towards a robust feature-based watermarking scheme," *Image Processing, 2006 IEEE International Conference on*, pp. 1401–1404, Oct. 2006.

[123] A. Westfeld, "High capacity despite better steganalysis (F5 - a steganographic algorithm)," in *4th International Workshop on Information Hiding*, vol. 2137, April 2001, pp. 289–302.

[124] Q. Wu, S. Sun, W. Zhu, G. Li, and D. Tu, "Detection of digital doctoring in exemplar-based inpainted images," in *Machine Learning and Cybernetics, 2008 International Conference on*, vol. 3, 2008.

[125] G. Xuan, Y. Q. Shi, J. Gao, D. Zou, C. Yang, C. Yang, Z. Zhang, P. Chai, C. Chen, and W. Chen, "Steganalysis based on multiple features formed by statistical moments of wavelet characteristic functions," in *Lecture notes in computer science: 7th International Workshop on Information Hiding*, vol. 3727, 2005, pp. 262–277.

[126] X. Yu and N. Babaguchi, "Breaking the YASS Algorithm via Pixel and DCT Coefficients Analysis," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 2008, pp. 1–4.

[127] X. Zhang and S. Wang, "Stego-Encoding with Error Correction Capability," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 3663–3667, 2005.

[128] D. Zheng, Y. Liu, J. Zhao, and A. Saddik, "A survey of RST invariant image watermarking algorithms," *ACM Computing Surveys (CSUR)*, vol. 39, no. 2, pp. 5–95, 2007.

[129] D. Zheng and J. Zhao, " LPM-based RST invariant digital image watermarking," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, vol. 3, 2003.