PRECISE LOCALIZATION OF KEY-POINTS TO IDENTIFY LOCAL REGIONS FOR ROBUST DATA HIDING

L. Nataraj, A. Sarkar and B. S. Manjunath

Department of Electrical and Computer Engineering,
University of California at Santa Barbara
Santa Barbara, CA 93106

ABSTRACT

We propose a novel data hiding system where data is embedded in local non-overlapping regions in an image. To survive cropping, the encoder embeds the same data in multiple regions of fixed dimensions, while the decoder's challenge is to independently retrieve the local regions. Salient feature points are computed on an image and the local regions are centered around them. To obtain non-overlapping regions, the points are pruned based on their corner strength and the size of the region. The decoder can retrieve the data only if it can precisely identify one or more of the same key-points. We present suitable key-point pruning methods such that even after considering a reduced number of key-points, the receiver is successful in identifying the same key-point locations as the encoder. We perform experimental comparison of various corner detectors and also study the performance of segmentation methods to obtain robust key-points.

Index Terms— data hiding, corner detectors, robust segmentation, keypoint pruning

1. INTRODUCTION

Data embedding in digital multimedia finds many applications ranging from digital watermarking, secret communications,copyright protection, content authentication and others. Recent watermarking methods focus on embedding the watermark in local blocks or regions rather than the entire image in order to survive attacks like cropping. Most of these methods find salient points in an image to identify the local regions. If the image undergoes a geometric transformation, the transform parameters are computed using different techniques and the image is usually realigned back to its original grid to extract the watermark. The data hiding problem is more challenging because *the receiver has to blindly decode the data without having access to the original watermark*. Most of the research is focused on hiding data in the entire image, while very few works address the challenge of robust data hiding in local image regions. Although, the amount of data that is hidden is very less, local region based data hiding can survive attacks such as cropping which cannot be survived by current data hiding methods.

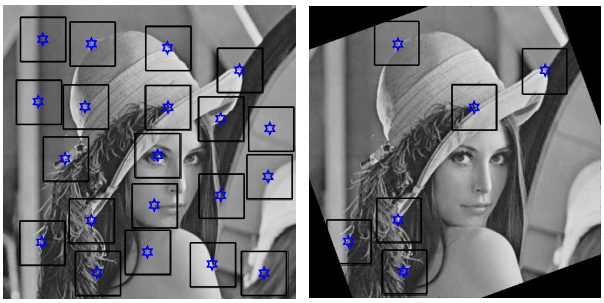

Fig. 1. (from left to right) (a) original image with key-points (KP) marked, with the square blocks showing the embedding regions, (b) a geometrically transformed image is aligned with the original grid - the KP common with (a) are shown

We propose a robust data hiding system by hiding the same data in different local regions in an image. To make the system robust to geometric transforms, synchronization peaks are inserted in the mid-frequency band of the Fourier magnitude domain whose positions are shared between encoder and decoder [1]. Although peak based methods are resistant against geometric transforms, they lack security since the peaks can be easily detected and smoothed out by an adversary. Here, our aim is to achieve *robust* rather than *secure* data hiding.

In our hiding scheme, the hiding occurs in non-overlapping square regions of fixed dimensions (Fig. 1). We use various corner point detectors to obtain the key-points(KP) and then prune them based on relative corner strengths and spatial constraints (non-overlapping) to obtain the hiding regions. We compare the performance of various corner point detectors and also see the effectiveness of the salient region based methods. We now explain the three main modules in our hiding system - encoder, channel and decoder.

Encoder: (Fig. 2(a)) Before encoding, peaks are inserted in the Fourier magnitude domain of the image. In the encoder, salient points are computed and data embedding is re-

This research is supported in part by a grant from ONR # N00014-05-1-0816 and # N00014-10-1-0141.

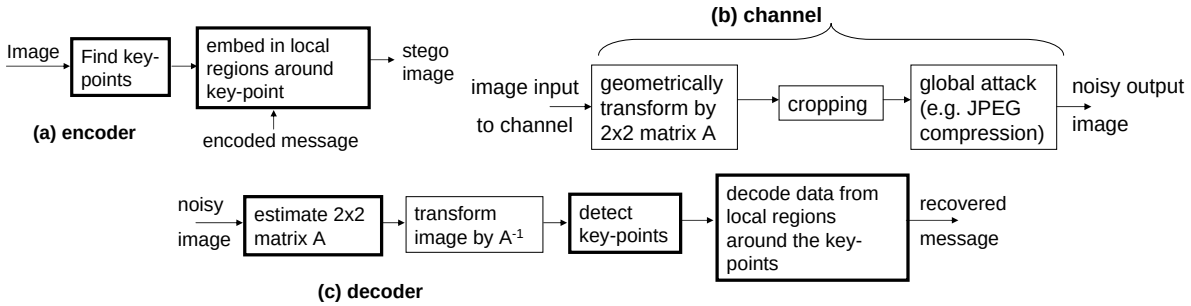

Fig. 2. The end-to-end data hiding framework (a) **encoder**: the same encoded sequence is embedded in every local region around a key-point, (b) **channel**: the stego image can be subjected to geometrical transformations, followed by image processing attacks, (c) **decoder**: it aligns the received 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.

peated in $B \times B$ square regions centered around a pruned set of key-points (the pruning method is described in Sec. 3). Embedding[2] is done using quantization index modulation (QIM) in non-overlapping 8×8 blocks in each $B \times B$ region. For embedding, we choose a certain low and mid-frequency band in the AC DCT domain. Since QIM is used, the decoder needs to access the same set of coefficients as the encoder - *a shift of one pixel will cause desynchronization* between the set of hiding coefficients considered at the encoder and the decoder. The data-bits are encoded using a turbo encoder.

Channel: (Fig. 2(b)) The stego image can be geometrically transformed using a 2×2 matrix A and then can be subjected to global image attacks and cropping.

Decoder: (Fig. 2(c)) The noisy stego image is first aligned with the original image grid by estimating the positions of DFT-domain peaks. Here, we do not describe how we estimate the peak positions. Then, the decoder obtains key-points from the aligned image (using the same pruning method as at the encoder). *A question arises as to how the decoder knows if it has correctly identified a key-point*. Turbo decoding is run on the extracted symbols corresponding to the block-based DCT coefficients obtained from the local region around that key-point. When the turbo decoding converges with a high average log-likelihood ratio (soft confidence values), then it is assumed that proper decoding has occurred. Our decoding method is robust to attacks such as JPEG, noise addition and the data can be decoded even after the received image has been re-aligned to its original grid after geometric attacks.

Paper Organization: The main contribution of this paper is the key-point pruning methods for different key-point detectors. Various salient point and salient region detectors are discussed in Sec. 2. Key-point pruning methods are discussed in Sec. 3. The detection results for the pruned key-points after various noise attacks are presented in Sec. 4.

2. CORNER POINT DETECTORS

When the geometrical transformations are compensated for (transformation by A in Fig. 2(b) followed by A^{-1} as in Fig. 2(c)), the interpolation involved introduces noise among the pixel values. The factors affecting the accuracy of the KP detector are the interpolation noise and global noise introduced by the channel (Fig. 2(b)).

Key-point detection methods:

(1) Scale Invariant Feature Transform (SIFT) [3] - Here, local maxima are computed in Difference of Gaussian (DoG) images obtained from a scale space representation of the image. Taylor's series expansion is used for more precise KP localization. The strength of a SIFT KP is the corresponding intensity value in the DoG scale space, for a given pixel location and a given scale.

(2) Harris [4] - The corner points are obtained by computing the Hessian of the auto-correlation matrix in the gradient domain for every image pixel, comparing the eigenvalues and then discarding flat areas and edges based on the relative magnitudes of the eigen values. Denoting the autocorrelation matrix per pixel as A, the corner strength m_h is expressed as $m_h = \det(A) - k(\text{tr}^2(A))$ (k is a tunable parameter) and pixels with m_h significantly greater than 0 are identified as Harris corners.

(3) Nobel-Forstener (NF) [5, 6] - The corner strength is determined using $m_n = (\text{tr}(A) + \epsilon)^{-1}\det(A)$ (ϵ is a tunable parameter).

(4) Harris-Laplace (HL) [7] - It uses the scale-adapted Harris function to localize points in scale-space and then selects the point with a characteristic scale, which is the extremum of the Laplacian over different scales.

(5) Laplacian of Gaussian (LoG) detector [8] - This algorithm is mainly used for blob detection where the number of detected key-points equals a pre-assigned number of blobs. For HL and LoG, the scale factor associated with a KP is regarded as its corner strength.

Segmentation Methods: Maximally Stable Extremal Regions (MSER): While obtaining the MSER regions, an image is thresholded into a set of binary images, where the thresholds are in ascending order. The blobs/regions obtained are such that the boundary pixels (one pixel thick) have a higher or lower intensity than the pixels inside the region. The extremal regions are affine invariant for both geometri-

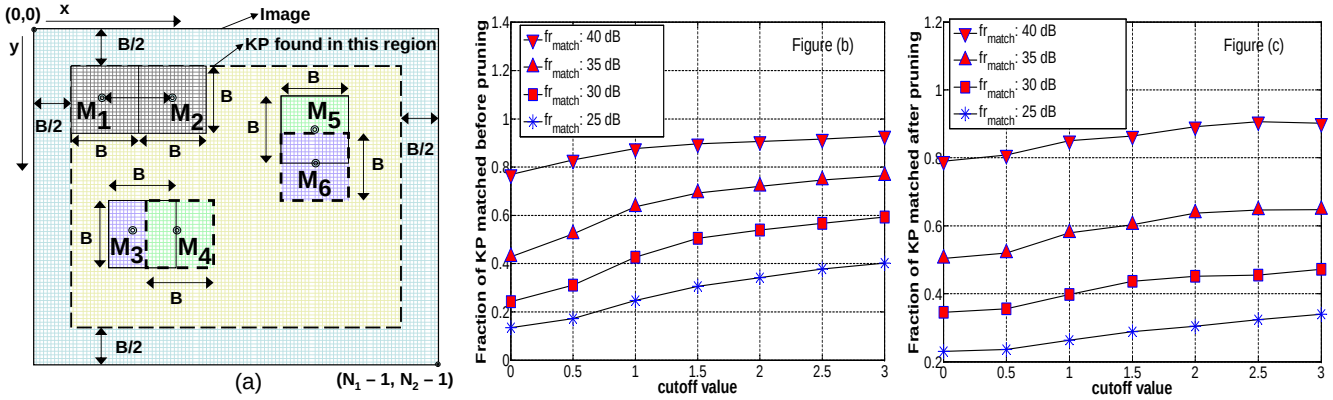

Fig. 3. Fig.(a): The KP pruning steps are shown. For two prospective KP ((M_3, M_4) or (M_5, M_6)) which are less than B apart from each other, we retain the one with the higher strength. When the distance between two points (e.g. (M_1, M_2)) is greater than or equal to B, both can be retained. Fig.(b) and (c): The figures show the fraction of KP that are correctly detected using SIFT key-points at the decoder side before and after pruning, respectively. As an attack, we add various levels of Gaussian noise before decoding.

cal and photometrical transformations [9]. The centroids of the segmented regions are used as the key-point locations.

3. METHOD FOR PRUNING KEY-POINTS

We first explain some notations that are used to explain the KP-based pruning method:

• B - the size of a local region used for hiding is $B \times B$.

• X_{enc}, K_{enc} - X_{enc} is the set of K_{enc} key-points obtained from the image at the encoder side where hiding occurs in $B \times B$ local regions around the key-points.

• X_{dec}, K_{dec} - X_{dec} is the set of K_{dec} key-points obtained after geometrically aligning the noisy received image with the original image.

• fr_{match} - it 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. $fr_{match} = |X_{dec} \cap X_{enc}|/K_{dec}$.

In our hiding scheme, we embed the same data in non-overlapping $B \times B$ blocks centered on various key-points found on the image. The robustness of a key-point selection scheme requires that if this image undergoes changes due to peak insertion, data embedding, and channel attacks, it should still be possible to *exactly* recover some of the original key-points from the received image. A standard KP detection scheme yields a large number of points. One way to decode the data is through brute force by centering a square block on every obtained key-point and decode data wherever the turbo decoder converges. 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.

The steps in the key-point computation are outlined below.

(1) **Border effects** - The key-points are computed in the $N_1 \times N_2$ image where a point (y, x) is considered if $B/2 < y \leq (N_1 - B/2)$ and $B/2 < x \leq (N_2 - B/2)$, as shown in Fig. 3(a). This ensures that the $(B \times B)$ block around the key-points do not extend outside the image.

(2) **Actual key-point computation** - It is done using a variety of methods, as discussed in Sec. 2. 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}, which is made low such that key-points are obtained in almost all regions of an image. Let there be K_1 such key-points, given by $\{M_i = (y_i, x_i)\}_{i=1}^{K_1}$.

(3) **Removing Nearby Key-points** - The key-points are first sorted according to their corner strengths and their corresponding positions $\{(y_i, x_i)\}_{i=1}^{K_1}$ are stored in a list with the top most position being the highest corner strength position. Starting from the first point, the chessboard distance is computed with all other points. Those points whose distance is lesser than B is removed. This process is repeated for all points till there are no more close enough key-points. For eg. one may end up with key-points $M_3 = (y_3, x_3)$ and $M_4 = (y_4, x_4)$, where $\max(|x_3 - x_4|, |y_3 - y_4|) < B$, as in Fig. 3(a). We cannot use both these key-points to obtain the hiding regions since we want the $B \times B$ blocks to be disjoint. The corner strength is used to decide 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. After removing all the nearby key-points, let the final number of remaining 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 steps (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. The fraction of matching KP before (after) pruning is shown in Fig. 3(b) (Fig. 3(c)) for varying cutoffs for the SIFT key-points.

Table 1. The average value of fr_{match} is computed over 250 images for different KP detectors and various noise attacks. B=80 is used to obtain the local hiding regions, along with 5 AC DCT coefficients in the hiding band.

Method	Avg. K_{enc}	Gamma-1	Gamma-2	AWGN	R+S+crop	Blur	JPEG	unsharp	median
		fr_{match}	fr_{match}	fr_{match}	fr_{match}	fr_{match}	fr_{match}	fr_{match}	fr_{match}
SIFT	20.4	0.288	0.272	0.251	0.110	0.242	0.123	0.139	0.168
Harris	20.0	0.397	0.333	0.295	0.093	0.205	0.127	0.259	0.070
NF	19.3	0.457	0.413	0.400	0.127	0.365	0.200	0.305	0.123
HL	**12.5**	**0.456**	**0.481**	**0.790**	**0.322**	**0.834**	**0.566**	**0.716**	**0.656**
LoG	11.1	0.416	0.343	0.665	0.250	0.675	0.477	0.530	0.548
MSER	16.3	0.217	0.203	0.137	0.077	0.172	0.105	0.122	0.147

4. EXPERIMENTAL RESULTS

To reiterate, one source of error for the KP detector module (between the encoder and decoder) is the interpolation involved during the geometric realignment process. KP detection errors are also caused by the various global image attacks that are assumed to be part of the channel noise. All these global attacks can be followed by cropping.

Our dataset consists of 250 JPEG images (quality factor (QF) of 75). The key-point detection results for various attacks are presented in Table 1. *Since we embed the same data in multiple blocks, our system is successful when there is exact localization of at least one region, i.e. at least one key-point match*. The attacks we performed are given below. Except the fourth attack, all other attacks are followed by a JPEG compression at an output QF of 60.

(1) Gamma-1: γ = 1.5, Gamma-2: γ = 0.6,

(2) AWGN: AWGN is added at SNR of 25 dB,

(3) R+S+crop: Rotation (R) at 30°, scaling (S) along both axes at scale factor of 0.75, and then 70% of the image is retained along each dimension (the received image is properly aligned first before KP extraction),

(4) blur: Gaussian blur at σ = 2,

(5) JPEG: JPEG compression at a QF of 10,

(6) unsharp masking: using a 3-by-3 unsharp contrast enhancement filter,

(7) median filtering: using a 5x5 window.

From Table 1, we see that even after severe noise attacks, there is still a match(one or more) between the encoder and decoder key-points. The Harris-Laplace has the best overall performance when compared with other detectors.

5. CONCLUSION

We have presented a key-point pruning method which combines both the corner strength magnitude and the spatial constraints (depending on block size) to obtain a reduced set of key-points distributed in different parts of an image. We hide data in non-overlapping blocks centered on each point. Using the same pruning method, we extract almost the same key-points from the received image and decode data by centering the blocks on those points. Since we embed same data in all blocks, we will be able to decode data when there is a match of at least one key-point with the original. Here, we assume that our data hiding system can accurately decode data even after various image attacks and geometric realignment and all it needs is the exact key-point to get the exact block. Although our pruning method outputs key-points in different parts of the image, in future, we will consider methods where almost all image regions are covered yet maintaining the magnitude and spatial constraints. Since cropping generally affects the image borders, key-points near the image center can be weighed more than points near the border, thus modifying the pruning method.

References

[1] P. Shelby and P. Thierry, "Robust template matching for affine resistant image watermarks," *IEEE Trans. on Image Processing*, vol. 9, no. 6, pp. 1123–1129, 2000.

[2] 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.

[3] D. Lowe, "Distinctive image features from scale-invariant keypoints," in *IJCV*, 2003, vol. 20, pp. 91–110.

[4] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[5] 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, Interlaken, Switzerland*, June 2-4 1987, pp. 281–305.

[6] J. Noble, "Finding corners.," *Image and Vision Computing*, vol. 6, no. 2, pp. 121–128, 1988.

[7] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *IJCV*, vol. 60, no. 1, pp. 63–86, 2004.

[8] T. Lindeberg, "Feature detection with automatic scale selection," *IJCV*, vol. 30, no. 2, pp. 79–116, 1998.

[9] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.