

# Real-Time Color Image Guidance System

[Daryl Robert Fortney](#)

Department of Electrical and Computer Engineering

University of California

Santa Barbara, CA 93106-9560

**February 2000**

# Abstract

This paper describes an implementation of a real-time color image guidance system. Areas of research include image acquisition, color-space transformation and classification, statistical object segmentation, predictive object tracking, and camera control. This system is operated on a PC platform including a color video camera, a digital signal-processing card containing a Texas Instruments C4x DSP and color image capture, and a serial port interface to a robotic tripod mechanism. This system was first used as a commercial instrument-tracking interface to a medical robotics application, and was later modified for use as a research tool and front-end to a larger system for face detection and recognition.

# Acknowledgements

It is with great pleasure that I thank my advisor, Professor B.S. Manjunath, for all the help and very patient guidance that he provided throughout the course of this research. I especially want to thank Computer Motion Inc. for providing me with both the tools and the time to explore the concepts presented in this research, amongst a tight schedule of commercial projects and deadlines.

Much of the research presented was performed while working as an intern at Computer Motion Inc. of Goleta, CA. I would like to thank Darrin Uecker for providing many useful and constructive suggestions, and Dr. Yulun Wang for his continued support.

# Table of Contents

1	Introduction.....	1
1.1	Motivation and Objectives.....	1
1.2	Principals of Image Guidance.....	2
1.3	Outline of Paper.....	3
2	System Setup .....	4
2.1	Hardware Setup .....	4
2.1.1	Intel 80386 PC .....	4
2.1.2	ISA-Bus TI320C40 DSP Card .....	5
2.1.3	Color Video Camera .....	5
2.1.4	Pan-Tilt Unit .....	5
2.1.5	Results Monitor.....	5
2.2	Software Setup.....	5
2.2.1	AVI_PC Component.....	6
2.2.2	AVI_C40 Component.....	7
2.2.3	AVISION MATLAB Tool.....	7
3	System Description.....	9
3.1	Image Acquisition.....	9
3.1.1	Pixel Format .....	10
3.1.2	Gamma Correction .....	10
3.2	Transformation and Classification .....	12
3.2.1	RGB Gamut .....	12
3.2.2	HSV Gamut .....	13
3.2.3	RGB to HSV Transformation.....	13
3.2.4	Bayesian Classification .....	14
3.2.5	Tag Color Determination .....	16
3.2.6	Results and Discussion.....	17
3.3	Segmentation.....	20
3.3.1	Connectivity Approach.....	20
3.3.2	Profile Histogram Approach.....	21
3.3.3	Results and Discussion.....	21
3.4	Tracking.....	22
3.4.1	Macro Box Configuration .....	22
3.4.2	Micro Box Configuration.....	23
3.4.3	Motion Prediction.....	23
3.4.4	Results and Discussion.....	24
3.5	Camera Control .....	25
4	System Usage .....	26
4.1	System Application.....	26
4.2	Descriptor Files .....	26
4.3	File Image IO.....	28
4.4	Program Run Control .....	29
4.5	Example 1: Tracking Medical Instrument Tags.....	29
4.6	Example 2: Tracking Human Faces .....	29
5	Conclusions.....	30
5.1	Performance .....	30
5.2	Areas of Future Research.....	30

# List of Figures

Figure 1 System Hardware Configuration.....	4
Figure 2 System Software Configuration .....	6
Figure 3 Pixel Data Format .....	10
Figure 4 Gamma Correction Image Histogram .....	11
Figure 5 RGB Gamut.....	12
Figure 6 HSV Gamut.....	13
Figure 7 Tag Color Determination.....	16
Figure 8 Tag Colors.....	17
Figure 9 Color Tag Classification Results.....	18
Figure 10 Camera Control Curves .....	25
Figure 11 PAI Image File Format .....	28

# List of Tables

Table 1 User Interface Adjustable Parameters .....	27
--	----

# 1 Introduction

This paper describes a real-time implementation of a color image guidance system, initially developed for use as a commercial instrument-tracking interface to a medical robot, and later modified for use as a research tool. Key components of this system include color image acquisition, color-space transformation and classification, statistical object segmentation, predictive object tracking, and camera control. This system provides a flexible platform for continued research and development of more complete and robust real-time color image guidance systems. It may also be used as a front-end component of a more complex system for object detection and recognition.

## *1.1 Motivation and Objectives*

Traditional surgical procedures involve the opening of a relatively large area of the patient's body for direct access and viewing of the operative site. Because of the direct access, these procedures are relatively easy to perform; however, for the same reason they are highly invasive to the patient, and cause a high risk of infection, increased post-operative trauma, and prolonged recovery times. Recently, the trend in surgery is a move toward more minimally invasive surgical techniques.[1]

Endoscopy is a minimally invasive surgical technique, where the surgical procedure is performed using long slender instruments inserted through pencil-sized holes in the patient's abdomen.[1] For visualization of the operative site, the surgeon relies on an endoscopic camera attached to a thin fiber optic tube inserted along side of the surgical instruments in the abdomen. Until recently, the task of positioning and holding the

camera steady during the length of the surgery was the job of an assistant, who with an unsteady hand would often times make it difficult for the surgeon to perform the work.

The AESOP (Automated Endoscopic System for Optimal Positioning) robot produced by Computer Motion Inc. of Goleta, CA was developed to solve this problem.[1] AESOP is a 6 degree of freedom robotic arm that attaches to the operating room table, and is able to accurately position and steadily hold the endoscopic camera. Studies show that the use of AESOP has lead to increased patient care and quality of service.[1]

With the first version of AESOP, surgical commands were issued to the robot through a foot switch or hand type controller, allowing a joystick like control over the camera position. Eventually, it became apparent that this man-machine interface was too cumbersome for the operating surgeon. A more intelligent interface was desired, an interface that would track the motion of the surgical instruments visually, automatically adjusting the camera position.

### ***1.2 Principals of Image Guidance***

Image guidance is a complex closed-loop control system with the goal of maintaining a target object within the bounds of the image region. In order to perform image guidance with a computer, many tasks are required of the system. First, the image must be digitally acquired from the imaging source, usually a color or monochrome video camera, by a sampling procedure to form pixels. After sampling is complete the pixels must be transformed and analyzed by an appropriate algorithm that extracts the features of interest, and classifies them based on their likelihood of being part of the desired object or not. The classification results are then segmented to form objects, which are then

further analyzed to determine if they are representative of the desired target. This information is then used to drive a control mechanism, thus repositioning the camera to maintain the object within the image boundaries. Before defining the appropriate algorithms to successfully perform these tasks, one must have a thorough understanding of the features and properties of the target object.

### ***1.3 Outline of Paper***

Chapter 2 covers the system setup. It details all hardware and software components of the system and identifies the interconnectivity of these components.

Chapter 3 covers the system description. It details each of the steps of image guidance as researched for implementation into this real-time system. In order, these steps are: image acquisition, color-space transformation and classification, object segmentation and tracking, and camera control. Where appropriate, the text identifies areas of previous work, discusses the decisions that were made, and presents the novel techniques employed by our system and resulting contributions to the field.

Chapter 4 covers the system usage. It details the various user and file level interfaces required to operate the system. It concludes with example configurations for using the system to track both medical instrument tags and human faces.

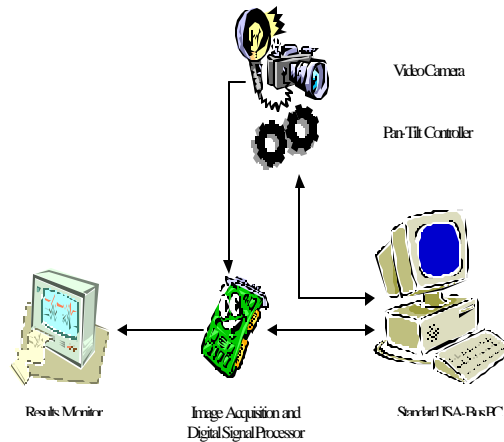
## 2 System Setup

This chapter covers the system setup. It details all hardware and software components of the system and identifies the interconnectivity of these components.

### 2.1 Hardware Setup

The hardware setup consists of a standard Intel 80386 PC, an ISA-Bus TI320C40 Digital Signal Processing Card, a Color Video Camera, a Pan-Tilt Unit, and a Results Monitor.

This hardware configuration is diagrammed in the figure below:



**Figure 1 System Hardware Configuration**

Following is a description of each component.

#### 2.1.1 Intel 80386 PC

At the heart of the system is a standard Intel 80386 PC running the DOS operating system. The purpose of the PC is to house the image processing hardware, while presenting the user interface software that allows real-time adjustment of various image guidance parameters.

### ***2.1.2 ISA-Bus TI320C40 DSP Card***

The image processing hardware consists of an ISA-Bus TI320C40 Digital Signal Processing Card with color image acquisition front-end produced by Oculus. This card contains the image acquisition front-end, dual-ported color image buffer memory, and a digital signal processor for executing the required image guidance algorithms. It communicates control and parameter information with the PC through a Link Interface Adapter protocol over the ISA-Bus. It is plugged in to the ISA-Bus of the PC.

### ***2.1.3 Color Video Camera***

The color video camera delivers a non-gamma corrected NTSC video image signal to the DSP card for acquisition and analysis. This signal is gamma corrected in the front-end acquisition hardware. It is connected to the NTSC RGB Video In of the DSP Card.

### ***2.1.4 Pan-Tilt Unit***

The Pan-Tilt Unit is a 2-degree of freedom robotic platform for x and y dimension camera control, produced by Directed Perceptions Inc. Control of the camera is possible through a serial port interface to a shell console resident on the controller. It is connected to the COM1 serial port of the PC.

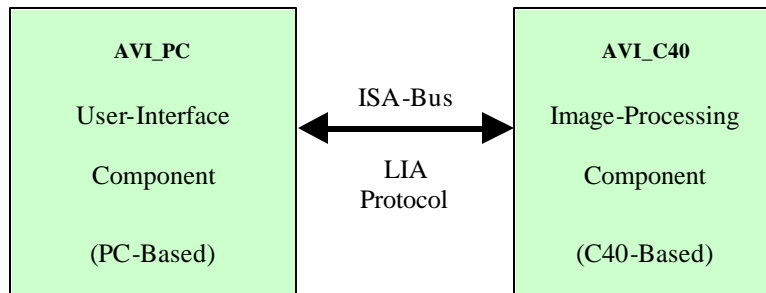
### ***2.1.5 Results Monitor***

The results monitor is a standard NTSC television with RGB inputs used for displaying the image being processed with any result from image processing overlaid onto the image. It is connected to the NTSC RGB Video Out of the DSP Card.

## ***2.2 Software Setup***

The software setup consists of a PC-based user-interface component termed **AVI\_PC** and a C40-based image-processing component termed **AVI\_C40**. These components

communicate through an ISA-Bus based LIA (Link Interface Adapter) Protocol. This software configuration is diagrammed in the figure below:



**Figure 2 System Software Configuration**

Following is a description of each component.

### ***2.2.1 AVI\_PC Component***

The AVI\_PC component is responsible for initializing and controlling the entire tracking system. This is accomplished by first loading the image processing component onto the ISA-Bus based C40 Digital Signal Processing card. This component is specified in the Descriptor File. Second, the user-interface is presented. Third, commands are sent to the image-processing component for process control and resulting control data is forwarded to the pan-tilt unit for camera control.

The user interface software was developed using the Borland C++ V3.11 development environment. The source code is located in the [\CODE\SYSTEM\PC](#) folder. The main project file is named AVI\_PC.PRJ. The components of this project were written in C++, and include the main program AVI\_PC.CPP which is supported by the files: IOL.CPP a generic IO Library, PTU.CPP a Pan-Tilt Unit driver, UART.CPP a serial port driver, and LIA.CPP and FASTLIA.ASM Link Interface Adapter Protocol drivers. Also provided is

SLU.CPP for controlling an optional servo lens unit. The resulting executable file is **AVI\_PC.EXE**.

### ***2.2.2 AVI\_C40 Component***

The **AVI\_C40** component is responsible for executing the image processing algorithms in real-time. It achieves this by first communicating with the PC-based component through the LIA Protocol for updated user-interface parameter and control information, second processing the image frame according to the image guidance algorithms as researched, and third sending the image processing results to the PC for camera control. Details of the image guidance algorithms as researched during the course of this paper are presented next.

The image processing software was developed using the Code Composer V3.05 development environment. The source code is located in the [\CODE\SYSTEM\C40](#) folder. The main project file is named **AVI\_C40.MAK**. The components of this project were written in mixed C and assembly language, and include the main program **AVI\_C40.C** which is supported by the files: **CLS\_HSV.ASM** the color space transformation and classification code, and **VIDEO.C** and **CHARSET.C** for overlaying and displaying the image processing results. The output file is **AVI\_C40.OUT** and is loaded by the PC-Based component to the DSP during system initialization.

### ***2.2.3 AVISION MATLAB Tool***

Throughout this project, MATLAB was used extensively as a test-bed for the development of the image processing algorithms. The result is a GUI based MATLAB tool termed AV for Active Vision image analysis. References to this application are

made throughout the contents of this paper. Using this tool in conjunction with hours of NTSC medical video camera footage allowed the determination and setting of the various *Descriptor File* parameters as specified in Chapter 8.

The source code for this tool is located in the [\CODE\MATLAB](#) folder. In order to run the tool this folder must be referenced by the **PATHDEF** variable of the MATLAB configuration file [\MATLAB\TOOLBOX\LOCAL\PATHDEF.M](#).

## 3 System Description

This chapter details each of the steps of image guidance as researched for implementation into this real-time system. In order, these steps are: image acquisition, color-space transformation and classification, object segmentation and tracking, and camera control.

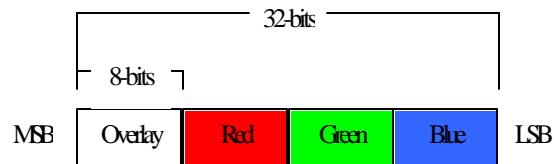
Where appropriate, the text identifies areas of previous work, discusses the decisions that were made, and presents the novel techniques employed by our system and resulting contributions to the field.

### 3.1 *Image Acquisition*

Image acquisition is the process of sampling the analog image into digital pixels for processing. In North America, video signals are broadcast in a standard display format termed NTSC, being 480 interlaced lines of analog color image data at 60 fields per second or 30 complete frames per second. Since NTSC video is interlaced, served as a field of odd lines followed by a field of even lines, memory is required to convert the image into a single progressive scan frame for processing. There are many different configurations for transmitting NTSC video, these being: color composite, RGB, and S-Video. In Color Composite, all the color information and frame synchronization information are multiplexed together and delivered on a single wire. In RGB, the three color components and synchronization information are separated and delivered on four separate wires. In S-Video, the luminance, chrominance, and synchronization signals are carried on separate wires. For our research we chose to use the RGB format, which has the highest SNR and is widely supported by quality off-the-shelf image acquisition hardware.

### 3.1.1 Pixel Format

After considerable research, we chose to develop the system using the Oculus Tci Color Image Processing Card. With this card, a number of parameters for sampling the video signal are selectable. For our work these settings were a sample resolution of 512 x 480 square pixels of 24-bit color depth, 8-bits each of the three primary colors: red, green, and blue. This data is stored into 32-bit dual ported memory, with the most significant 8-bits used to store image overlay data. This representation is shown in the figure below:



**Figure 3 Pixel Data Format**

### 3.1.2 Gamma Correction

During sampling, a non-linear gamma correction to each of the primary colors is performed by the ‘front-end’ hardware, allowing compensation for the color characteristics of the particular video camera and acquisition hardware. Gamma correction adjusts the intensity to voltage response curve through the equation below:

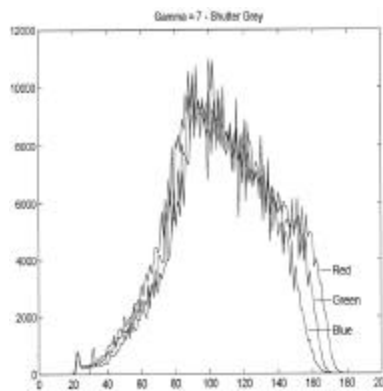
$$Intensity = Voltage^{(1/c)} \quad (0.1)$$

Where  $\chi$  represents the gamma correction value

Gamma correction therefore changes the ratios of red to green to blue for a particular pixel color. This correction is very important in that it directly affects the color qualities that are being measured for further classification by our system. Ideally, a system with

$\gamma=1.0$  preserves and gives a linear color space representation and is thus color-balanced. If the system is not properly color-balanced, the performance of a color-based classifier degrades. Typically, a computer monitor has  $\gamma=2.5$ .

Many different methods have been used to determine the gamma curve for camera and display devices.[2][3][4] In order to correct for the gamma curves of both the video camera, and the acquisition hardware, the method we chose was to sample photographic gray-scale reference papers under diffuse lighting using various gamma corrections between 1.0 and 8.0. These images were then analyzed in MATLAB using the AVISION tool by plotting and comparing image histograms such as the one shown below for  $\gamma=7.0$ .



**Figure 4 Gamma Correction Image Histogram**

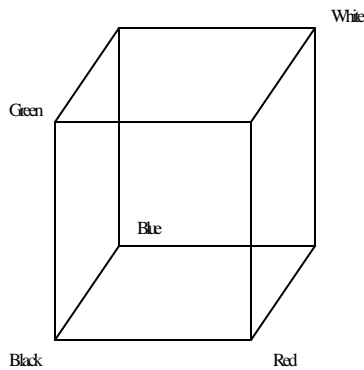
The gamma correction was then chosen as the value which best aligns the normalized intensity responses over the entire range of intensity.

### 3.2 Transformation and Classification

A color's identity may be represented in terms of a set of color space parameters termed a gamut. Many gamuts have been researched for color image processing including RGB, YUV, YCrCb, HSI and the related HSV gamuts.[5][6][7][8] After extensive MATLAB evaluation, we determined that the HSV gamut was a clear leader in both classification accuracy and computational simplicity. In this chapter we discuss the RGB and the HSV gamuts in particular and declare the implemented color space transformation algorithm.

#### 3.2.1 RGB Gamut

The most common color gamut for image acquisition and display is RGB. As identified this is the representation acquired from the video camera. In the RGB color space, color is divided into 3 primary color components: Red, Green, and Blue. Using this representation, all colors of light are represented as an intensity of each of the primary colors that combine to create the resulting color. Geometrically, these components form the orthogonal axis of a 3-dimensional cube as shown below.



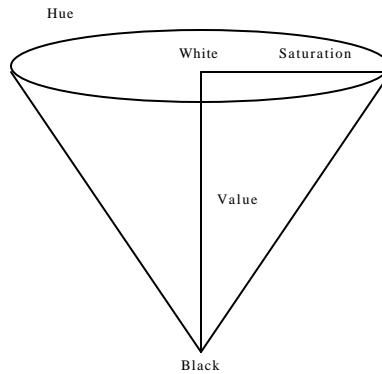
**Figure 5 RGB Gamut**

This color representation is often convenient for use by computer displays and video cameras, where photo emitters and receptors sensitive to each of these three primary

colors are used. While convenient for use in computer displays and video cameras, this color representation is not very convenient for perceptually based image analysis, where changes in lighting causes a color displacement along a non-orthogonal axis.

### 3.2.2 HSV Gamut

An alternate color gamut is HSV. In the HSV color space, color is divided into 3 perceptual components: Hue, Saturation, and Value. Geometrically these components form a cone, as shown in the diagram below:



**Figure 6 HSV Gamut**

In the HSV gamut, changes in lighting result in a translation along the Saturation and Value axes, with little effect on the Hue axis. It is this light-invariance that makes both HSV and HSI gamuts robust and computationally economical for color-based classification. As a result of this characteristic, accurate classification based on color may be performed very quickly using simple look-up table techniques once the image is represented in the HSV gamut.

### 3.2.3 RGB to HSV Transformation

The non-linear transform between the RGB and HSV gamuts is: [8]

$$M = \max(R, G, B) \quad (0.2)$$

$$m = \min(R, G, B) \quad (0.3)$$

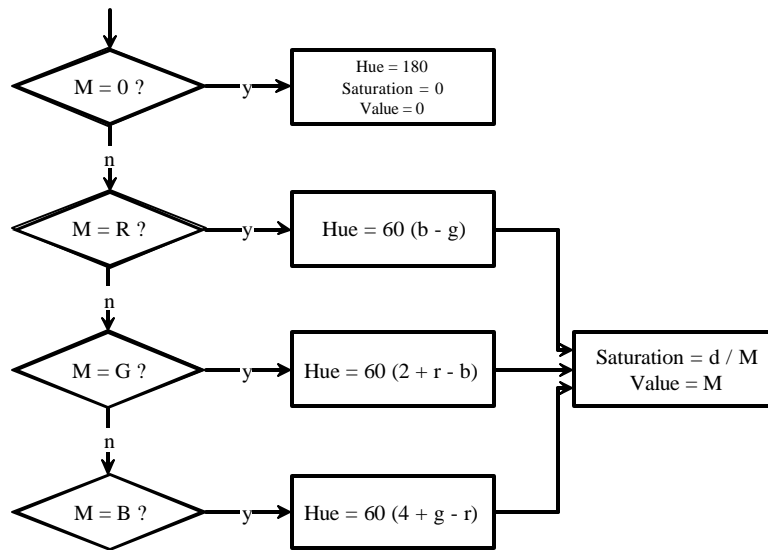
$$d = M - m \quad (0.4)$$

$$r = (M - R) / d \quad (0.5)$$

$$g = (M - G) / d \quad (0.6)$$

$$b = (M - B) / d \quad (0.7)$$

From these values we calculate the resulting Hue, Saturation, and Value using the non-linear algorithm as diagrammed below:



Since this non-linear color space transformation must be performed on each pixel within a bounding region, for highest performance the RGB to HSV gamut transformation was coded in assembly language.

### 3.2.4 Bayesian Classification

We used a Bayesian classifier to determine the classification thresholds for Hue, Saturation, and Value of the desired object. Bayesian classifiers are optimal in that they give the most probable classification assuming apriori knowledge of the probability

distribution function.[9] This is based on Bayes' Theorem, which states, given some known probability density function,

$$f(x)$$

Then the conditional distribution of that density function given some event is,

$$f(x|A) = \frac{P(A|X=x)}{P(A)} f(x) \quad (0.8)$$

Where the probability of the event is indicated as,

$$P(A) = \sum_0^i P(A|x_i)P(x_i) \quad (0.9)$$

Based on this theorem, an optimal classifier will choose the distribution with highest resulting probability, written as,

$$\max(P(x|A)) \quad (0.10)$$

Assuming the distribution is stationary, these calculations may be performed offline.

For endoscopic medical imaging, the probability density function of the non-tagged image was found to be significantly normal, stationary, and gaussian. The probability density function of a normal Gaussian distribution is described as,

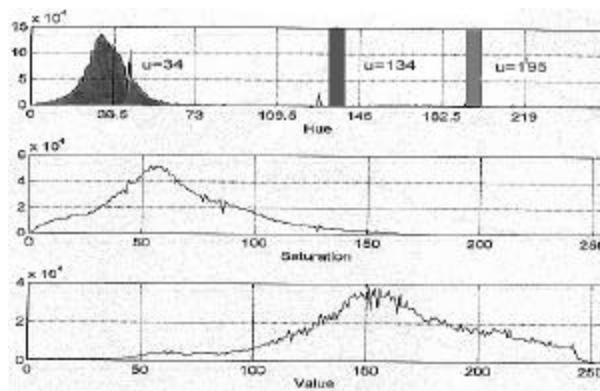
$$f(x) = \frac{1}{s\sqrt{2p}} e^{-\frac{(x-h)^2}{2s^2}} \quad (0.11)$$

Applying this distribution to the Bayesian classifier, we get a closed form solution for a distance function. We can see from this that best classification performance occurs when the distribution function of the object of interest is also Gaussian with a mean that is far from the background as possible and a very small variance.

### 3.2.5 Tag Color Determination

From this result we would conclude the in determining a color tag to offset and clearly identify an object apart from its background, logically we should choose a consistent color tone that is well removed from the represented background color spectrum as expected.

To automate this process we analyzed 250 medical images and 24 hours of medical video footage using the AVISION development tool written in MATLAB. Analysis consisted of feeding in images from the camera, and from pre-recorded VCR tapes of surgery to the tool, and hand selecting regions of both human tissue, and medical instrumentation using a mouse. From this data a pdf of human tissue color space was determined. The results of this study are shown in the figure below:

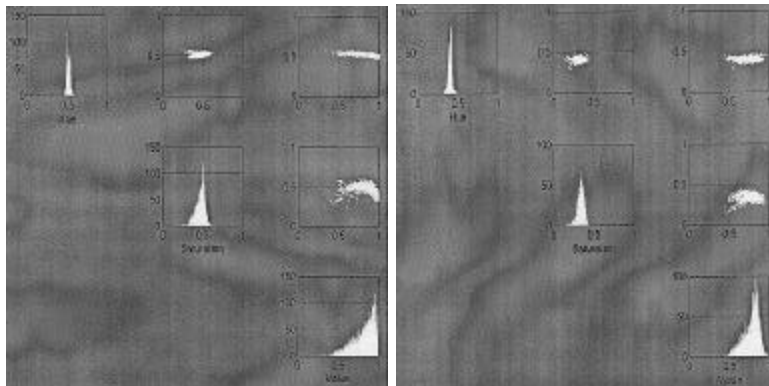


**Figure 7 Tag Color Determination**

From this we see that the mean of the human tissue samples occurs at a Hue of 34 on a scale of 256 or a normalized Hue of .13. This falls within the red through yellow region of the spectrum, which is as expected. We also see the variance is relatively small, giving a great deal of under represented colors to choose our tag from. Also shown in the figure are the ideal colors bars for a two-tone color tag, allowing determination of not

only instrument position but also direction. These colors have a mean normalized Hue of .52 which is an aqua marine, and .76 which is a deep blue. This follows our intuition as to easily discriminate colors with the human body. This result agrees with research that has been done showing green as a complimentary color within medical environments.[1]

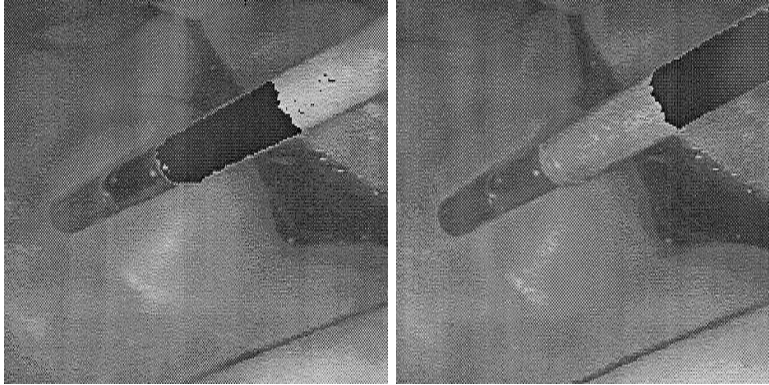
Unfortunately, we were not able to obtain accurate samples of these colors in a material that could be applied to a medical instrument. However, rough approximations of these colors with Hue of .4 and .5 were secured and testing was done to assure the color correctness as seen in the MATLAB plots below of a cross-sectional histogram of the resulting two-tone color tags that were non-uniformly illuminated.



**Figure 8 Tag Colors**

### ***3.2.6 Results and Discussion***

This two-tone tag was then applied to the end of a medical instrument, and studies were done verifying the classification performance in an animal trial setting. The images below show the classification results as analyzed using MATLAB.



**Figure 9 Color Tag Classification Results**

In these images pixels classified as part of the instrument tag are colored black. As can be seen, classification of the resulting two-tone tags is extremely precise.

Continued testing using MATLAB, we found sharpness has no effect on color pdf, and therefore the color space transformation and classification algorithms are invariant to camera focus.

We also found that as saturation decreases, noise in Hue increases, thus for best performance the target object should not be under or over saturated which occurs at the extremes of illumination. In a medical setting this condition is termed specular reflection, and occurs frequently as a result of light reflections off of the mucous lining of the organs.[1] Currently, we have no definitive solution to this problem, but suggest modeling the specular reflection and basing the classification on the classification results of the surrounding non-specular regions.

Since the classification decision occurs for each pixel, the approach taken was to perform the classification using a fast and simple look-up table technique. This technique involves a separate lookup table for each of the Hue, Saturation, and Value dimensions. These tables are seeded by the user and may operate in one of two modes: static with no feedback, and dynamic with feedback. In static mode the values in the lookup table do not change from frame to frame, whereas in dynamic mode, the values with the greatest coverage are preserved and the infrequent values are discarded based on a user-defined threshold. This allows the classifier to tighten its boundaries based on the results of previous classification. For processing efficiency, this algorithm was implemented as an extension of the transformation code in assembly language. Since many parameters of acquisition, transformation, and classification are selectable through the user interface, it is easy to tune the system to respond to different requirements.

Results from the real-time implementation were compared to results from MATLAB to validate the accuracy of implementation.

### **3.3 Segmentation**

In developing an appropriate segmentation algorithm that is both accurate and efficient, we experimented with both previously researched and novel approaches, two of which are: a connectivity approach and a profile histogram approach.

#### **3.3.1 Connectivity Approach**

Many attempts have been made to segment images based on pixel connectivity.[10][11][12][13] This is a multi-pass approach over the entire classified image. In the first pass, the classified image is scanned from top to bottom left to right and pixels are labeled based on their 2-dimensional proximity to other classified pixels. When connections are found between disparate groups of pixels, a table is updated to represent this connectivity. This table is then compiled, and during the second pass, the labels are renamed and the connectivity between these groups is resolved. Also, during the second pass basic object statistics such as mass, positional mean and variance are often computed. This information is often used to guide the tracking mechanism, or to augment more detailed object analysis.

This approach results in very well defined objects; however, in the presence of classification noise, requires additional filtering such as 2-dimensional median, or object thresholding. If properly implemented this approach shows considerable promise, resulting in accurate and well-defined object boundaries, unfortunately this approach is impractical for real-time systems because of the computational complexity of analysis.

### ***3.3.2 Profile Histogram Approach***

A second approach to segmentation we evaluated is a novel approach based on profile histograms. Using this approach we first flatten the 2-dimensional classification image into two 1-dimensional profile histograms along the x and y axes. Median filtering may then be performed very quickly on these histograms, and the grouping task is greatly simplified. From these histograms, basic object statistics may also be easily computed.

Though this approach alleviates the computational complexity of a higher dimensional approach, it is far from perfect in that objects have the possibility of merging during the flattening process, giving inaccurate segmentation results. One method of minimizing this problem is to perform the segmentation on smaller sub-image regions and linking the results using a 2-dimensional connectivity scheme as described. In the limiting case, this becomes the same as the pixel-wise connectivity approach as discussed.

### ***3.3.3 Results and Discussion***

In our system implementation we have made a number of significant simplifications to the segmentation task. We have disabled sub-image region segmentation and linking, and assume one object per image. This was done to streamline the application for use in the medical robotics application where the environment is controlled and well known.

As with the transformation and classification algorithms, the statistical segmentation operation was also written in assembler to achieve real-time performance.

### **3.4 Tracking**

In tracking objects in images, much research has been devoted to the use of bounding boxes.[11][13] Bounding boxes rely on the principle that the processing of the entire image is typically both computationally expensive and unnecessary, since objects seldom occupy the entire image space. The term bounding box implies the use of a rectangular region, but the terminology is also used to represent an image sub-region of any shape and size. In theory, by confining expensive image processing operations to such a sub-image region improves performance.

Appropriately, as the region size increases, so do the number of computations, resulting in decreased performance on systems with limited resources. For this reason, various sizes and configurations of bounding boxes have been researched and evaluated for performance. We have identified and evaluated two bounding box configurations: Macro Box Configuration and Micro Box Configuration.

#### **3.4.1 Macro Box Configuration**

Macro Box is a term we use to indicate a bounding box configuration that is larger than and fully contains the object of interest. In general, a Macro Box is not fixed in size. Since the goal of a Macro Box is to encompass the entire object for analysis, it must scale with the object. In practice, a Macro Box must be sized somewhat larger than the object to account for motion of the object.

Traditionally, most research has concentrated on the use of Macro Boxes. An advantage of using a Macro Box, is that determination may be made of the objects exact mass,

position, orientation and other statistical properties. A disadvantage of using a Macro Box; however, is that as the object scales, so does system performance. Thus, as the object moves toward the camera taking up more of the image frame, processing of the image must slow down. One solution to this problem, which has been used with success, is region sub-sampling based on scale. A second and more complicated solution is to control the camera zoom, thus maintaining the object at a fixed size within the image frame.

### ***3.4.2 Micro Box Configuration***

Micro Box is a term we use to indicate a bounding box configuration that is smaller than the object of interest. Since a Micro Box is not required to encompass the desired object, it may be fixed in size and therefore does not scale with the object.

An obvious advantage of using a Micro Box, is that system performance is unaffected by object scale. A disadvantage is accurate determination of certain object properties is not possible, such as mass and position. A side effect of this condition is that as long as the bounding box is contained completely within the object, no tracking is required.

### ***3.4.3 Motion Prediction***

In order to improve tracking performance motion prediction of the bounding region is performed. A novel approach to predicting motion as implemented is based on the physics of mass in motion. The formula for the position of a mass given its acceleration, velocity, and initial position is,

$$x = \frac{at_d^2}{2} + vt_d + x_0 \quad (0.12)$$

Integrating, in general given the N-order positional history  $x_n$  of the mass we have,

$$x = \sum_{n=0}^N \frac{1}{n!} (A_n t^n) \quad (0.13)$$

Where,

$$\begin{aligned} A_{0t} &= x_n \\ A_{1t} &= A_{0t} - A_{0(t-1)} \\ A_{2t} &= A_{1t} - A_{1(t-1)} \\ &\dots \\ A_{nt} &= A_{(n-1)t} - A_{n(t-1)} \end{aligned} \quad (0.14)$$

From this we see that the higher order terms are given much lower weight, and thus have a small effect on the result. Also, we see that as the order is increased, so is the effect of sample quantization error through the accumulated term differencing.

#### **3.4.4 Results and Discussion**

Through experimentation we have found the use of a Micro Box is appropriate for color-based classification and tracking systems, when tracking the exact mass, position or orientation of an object is not required. As well, we found that the implicit ‘tremor filtering’ behavior is actually a beneficial quality in a medical robotic instrument tracking environment, where unnecessary camera motion is eliminated, resulting in a stable image.

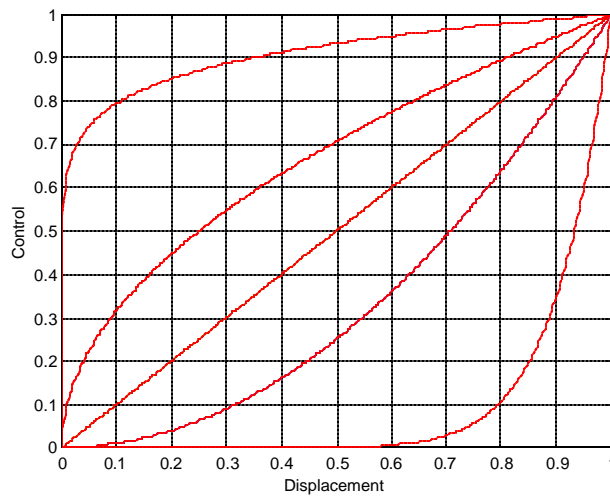
We also found that improved tracking performance occurs for motion prediction based on the physics of mass in motion, but for a predictor over 4<sup>th</sup> order, the benefits of the prediction are outweighed by the practical aspects of quantization error.

### 3.5 Camera Control

Once tracking data has been determined, data must be sent to the pan-tilt unit to reposition the camera effectively centering the target object in the image frame. This is performed through the relationship,

$$control = \alpha \square displacement^{1/b} \tag{0.15}$$

where  $\alpha$  represents the motor gain, and  $\beta$  represents a sensitivity curve response, as shown in the figure below:



**Figure 10 Camera Control Curves**

## 4 System Usage

This chapter covers the system usage. It details the various user and file level interfaces required to operate the system. It concludes with example configurations for using the system to track both medical instrument tags and human faces.

### 4.1 System Application

This system is initialized and executed by the **AVI\_PC.EXE** application running on the PC.

The command line syntax of this application is as follows:

**AVI\_PC**     *[filename.dsc]*

Where, *[filename.dsc]* is an optionally specified *Descriptor File*.

### 4.2 Descriptor Files

A *Descriptor File* is a text-based file used to specify both the target application and the default values of each of the target application recognized operating parameters. It is a text file listing first, the name of the DSP target application, second the ISA port addresses of the DSP card, third a descriptive comment, and finally an ordered list of each application recognized parameter name followed by the minimum, default, and maximum parameter values. Refer to the default *Descriptor File* named [AVISION.DSC](#) for an example.

The ordered and complete list of recognized parameters of the **AVI\_C40** application are listed in the table below:

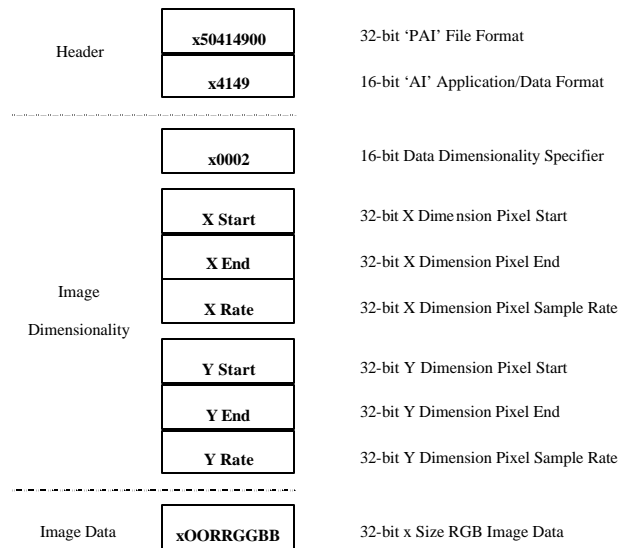
**Table 1 User Interface Adjustable Parameters**

Parameter Name	Min Value	Max Value	Description
<b>Image Acquisition Parameters</b>			
IMG_MODE	0	1	Perform Acquisition 0=Off 1=On
IMG_OVR	0	1	Store Image w/ Overlay 0=Off 1=On
IMG_X_S	0	511	X Dimension Start Pixel
IMG_X_E	0	511	X Dimension End Pixel
IMG_X_R	1	511	X Dimension Sample Rate
IMG_Y_S	0	479	Y Dimension Start Pixel
IMG_Y_E	0	479	Y Dimension End Pixel
IMG_Y_R	1	479	Y Dimension Sample Rate
<b>Classification Parameters</b>			
CLS_MODE	0	1	Perform Classification 0=Off 1=On
CLS_X_R	1	511	X Dimension Sample Rate
CLS_Y_R	1	479	Y Dimension Sample Rate
CLS_V_LO	0	255	Value Threshold (Low)
CLS_S_LO	0	255	Saturation Threshold (Low)
CLS_H_LO	0	255	Hue Threshold (Low)
CLS_H_HI	0	255	Hue Threshold (High)
<b>Object Tracking Parameters</b>			
TRK_MODE	0	2	Perform Tracking 0=Off 1=On (Single Object) 2=On (Multiple Object)
TRK_Zi	0	999	Required Mass To Begin Tracking
TRK_Zo	0	999	Required Mass To Continue Tracking
TRK_SRCt	0	999	Time (ms) To Search For Lost Object
TRK_XSIZE	0	511	X Dimension Bounding Box Maximum Size
TRK_YSIZE	0	479	Y Dimension Bounding Box Maximum Size
<b>Camera Control Parameters</b>			
CNT_MODE	0	1	Perform Camera Control 0=Off 1=On
CNT_SENS	0	10	$\beta$ - Sensitivity Curve Selector
CNT_GAIN	0	999	$\alpha$ - Gain Curve Multiplier

After initializing the Pan-Tilt Unit, AVI\_PC loads and initializes the specified DSP target application AVI\_C40 to the DSP card using the default parameter values as specified in the Descriptor File. These parameters are then listed on the console where the user may select and adjust their values using the number pad.

### 4.3 File Image IO

In order to efficiently store and retrieve multi-dimensional image frames and sequence data including overlay information along side tracking data, a unique file format was developed. This format is termed the Program Application Interface (PAI) format. Using this format many different types of information may be stored. The specific format for Color Images is diagrammed below:



**Figure 11 PAI Image File Format**

Image data from the camera may be output either to the results monitor or to a specified PAI file. The default setting is to output the images to the results monitor. To redirect the output to a specified file, select O for output and then enter F for file, followed by the path and filename. Be aware that a great deal of data is generated very quickly using this method.

Image data for processing may be input either from the camera directly, or from a specified PAI file. The default setting is to input images from the camera. To redirect

the input from a specified file, select **I** for input and then enter **F** for file, followed by the path and filename.

#### **4.4 Program Run Control**

The image processing algorithms may be run continuously or a specified number of times. The default setting is to run continuously. To run the image processing loop specified number of times, select **R** for run mode and then enter **S** for step followed by the number of steps. This feature is useful for storing a limited number of images to file as described above.

While the processor is running, the user may adjust any parameter values. Adjustments are made by either using the number pad to scroll the arrow through the parameter list and selecting new values, or by editing the *Descriptor File*. Next we present two example *Descriptor Files* which preset the system to track medical instrument tags and human faces.

#### **4.5 Example 1: Tracking Medical Instrument Tags**

An example Descriptor File setup for tracking medical instrument tags is

[\CODE\SYSTEM\INST.DSC](#). In this file, the Hue parameter has been set to identify colors between the range of 170 through 210. This represents a shade of blue with exceptional results for the chosen medical instrument tag as previously discussed.

#### **4.6 Example 2: Tracking Human Faces**

An example Descriptor File setup for tracking human faces based on Caucasian skin color is [\CODE\SYSTEM\FACE.DSC](#). In this file the Hue parameter has been set to

identify colors between the range of 60 through 80. This represents a shade of pink akin to Caucasian skin color.

## **5 Conclusions**

In this report we outlined a real-time system for color image guidance. First, the system setup was described, identifying the core hardware and software components and their connectivity. This was followed by a discussion of the various steps of image processing required of color image based guidance. References were made where appropriate to current work in the field, and novel techniques employed by our system were developed. Finally, the usage of the system was described, and examples were given for configuring the system to the specific tasks of both medical instrument tag tracking and human face tracking.

### ***5.1 Performance***

This system is fast, flexible, and accurate. It is capable of executing the image processing algorithms discussed at rates of up to 30 frames per second, while presenting a complete user interface allowing on the fly parameter adjustments, which allow the system to be configured to operate under a wide range of requirements. The accuracy of implementation of the various algorithms has been verified against tools developed using MATLAB.

### ***5.2 Areas of Future Research***

Though the system as presented is complete and fully functional, the implementation serves only as a starting point for continued research. It is anticipated that this system will

find many possible uses. Foremost, as a learning tool for future students of image processing, as well as a front-end processor for larger and more involved image tracking and identification systems.

Through the course of this research I have learned a great deal about image processing.

However, more important than any material information I have absorbed, I have learned that there is no substitute for getting your hands dirty and applying the knowledge.

Seldom does theory meet practice and everything works as expected. This is where the real education begins.

## References

1. <http://www.computermotion.com> – On Advanced Minimally Invasive Surgical Techniques
2. [http://www.cgsd.com/papers/gamma\\_intro.html](http://www.cgsd.com/papers/gamma_intro.html) – On Gamma Correction
3. CONFERENCE PAPER. Rovatti, R.; Franchii, E.; Manaresi, N.; Bellini, A.; Tartagni, M. **Analog implementation of gamma-correction for CMOS cameras.** 1998 IEEE International Conference on Electronics, Circuits and Systems. Surfing the Waves of Science and Technology (Cat. No.98EX196), (vol.1)
4. CONFERENCE PAPER. Lauziere, Y.B.; Gingras, D.; Ferrie, F.P. **Color camera characterization with an application to detection under daylight.** Proceedings Vision Interface '99, (Proceedings Vision Interface '99, Proceedings of the Vision Interface Conference, Trois-Rivieres, Que., Canada, 18-21 May 1999.) Toronto, Ont., Canada: Canadian Image Process. & Pattern Recognition Soc, 1999. p.280-7. vi+634 pp. 9
5. I. Pitas, Fundamentals of colour image processing. In: *Digital Image Processing Algorithms* Prentice-Hall, Englewood Cliffs, NJ (1993), pp. 23-40.
6. R.C. Gonzalez and R.E. Woods, Image transforms. In: *Digital Image Processing* Addison-Wesley, New York (1992), pp. 81-128.
7. Zarit, B.D., Super, B.J., Quek, F.K.H., Dept. of Electr. Eng. & Comput. Sci., Illinois Univ., Chicago, IL, USA, Comparison of five color models in skin pixel classification., (Proceedings International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems. In Conjunction with ICCV'99 (Cat. No.PR00378), Proceedings of the International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'99), Corfu, Greece, 26-27 Sept. 1999.) Los Alamitos, CA, USA: IEEE Comput. Soc, 1999. p.58-63.
8. Video Demystified – On Color Space Conversions
9. Probability, Random Variables, and Stochastic Processes. Athanasios Papoulis. Mc Graw-Hill, Inc. New York. 1991.
10. Herrmann, S., Mooshofer, H., Dietrich, H., Stechele, W., Tech. Univ. of Munich, Germany, A video segmentation algorithm for hierarchical object representations and its implementation., IEEE Transactions on Circuits and Systems for Video Technology, IEEE Trans. Circuits Syst. Video Technol. (USA), vol.9, (no.8), IEEE, Dec. 1999. p.1204-15
11. Pauwels, E.J., Frederix, G., Dept. of Electr. Eng., Katholieke Univ., Leuven, Belgium, Colour segmentation and shape extraction for image interpretation., (Proceedings of Fifth International Conference. PRIP '99. Pattern Recognition and Information Processing. Vol.1, Proceedings of PRIP'99 International Conference on Pattern Recognition and Information Processing, Minsk, Byelorussia, 18-20 May 1999.) Szczecin, Poland: Wydawnictwo i Drukarnia Inst. Inf. Politech. Szczecinskiej, 1999. p.85-9. 288
12. Tsougarakis, C., Panchanathan, S., Dept. of Comput. Sci. & Eng., Arizona State Univ., Tempe, AZ, USA, Content-based object segmentation in video sequences., Proceedings of the SPIE - The International Society for Optical Engineering, Proc. SPIE - Int. Soc. Opt. Eng. (USA), vol.3653, pt.1-2, (Visual Communications and Image Processing '99, San Jose, CA, USA, 25-27 Jan. 1999.) SPIE-Int. Soc. Opt. Eng. 1998. p.1269-76.
13. Jiann-Der Lee, Yu-Lin Hsiao, Zhong-Xian Huang, Dept. of Electr. Eng., Chang Gung Univ., Tao-Yuan, Taiwan, Color image segmentation for identification of the bladder cancer., Biomedical Engineering, Applications Basis Communications, Biomed. Eng. Appl. Basis Commun. (Taiwan), vol.11, (no.3), Biomed. Eng. Soc. Republic of China, 25 June 1999. p.149-57.

## [Additional Interesting References](#)