INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Robust Image Segmentation Towards an Action Recognition Algorithm

Shawn Arseneau

Centre for Intelligent Machines
Department of Electrical Engineering
McGill University, Montreal

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Masters of Engineering

Acknowledgements

I would like to thank my research supervisor, Jeremy Cooperstock, for his invaluable advice and guidance throughout my Masters. I found his energy and encouragement were key in the quality of my research.

I would also like to thank Jean-Guy Nistad for his help in the French translation of the abstract.

Table of Contents

Abstract (English) 1

Abstract (French) 2

Chapter 1

1.1 Introduction 3

Chapter 2 – Filters and Kernels

2.1 Introduction to Filters 6

2.2 Low Pass Filter 6

2.3 Median Filter 8

2.4 High Pass Filter 9

2.5 Dilation and Erosion 9

2.6 Edge Detection 10

2.6.1 First Derivatives 10

2.6.2 Second Derivatives 11

2.7 Conclusion 12

Chapter 3 - Background Removal Schemes and Noise Reduction Techniques

3.1 Background Removal Techniques 13

3.1.1 Chroma-Keying 13

3.1.2 Background Differencing 15

3.1.3 Background Primal Sketch 17

3.2 Ghosting 21

3.3 Motion Information 25

3.4 Noise Reduction Techniques 25

3.4.1 8-Connected Isolated Pixels 26

3.4.2 Otsu Thresholding Method 28

3.5 Conclusions 32

Chapter 4 – Color Detection

4.1 Color Models 33

4.1.1 RGB 34

4.1.2 HSV 37

4.1.3 YUV 40

4.1.4 YIQ 42

4.1.5 Normalized RGB 43

4.2 LUT vs. Gaussians 45

4.3 Results 45

4.4 Conclusions 53

Chapter 5 – Contour Extraction

5.1 Contours 55

5.2 Action Feature Vector Elements 56

5.2.1 Size 56

5.2.2 Position 57

5.2.3 Orientation 57

5.3 Regions 59

5.4 Boundary Algorithms 60

5.4.1 Boundary Following Algorithm 60

5.4.2 Dilate and Compare 61

5.4.3 Pixel Sweep 62

5.5 Skeletonization 63

5.5.1 Distance Metric 63

5.5.2 Medial Axis 64

5.5.3 Thinning 65

5.6 Simplifying Data Set Information 66

5.6.1 Chain Codes 66

5.6.2 Curve Fitting 67

5.6.3 Recursive Subdivision and Polygonal Approximation 69

5.6.4 Hop Along Recursive Subdivision 69
5.6.5 Hough Transform 70
5.6.6 Active Contours 71
5.7 Summary 71

Chapter 6 – Action Recognition Segmentation

6.1 Steps Towards Action Recognition 72
6.2 Background Removal Scheme 74
6.3 Skeletal Attraction 77
6.4 Conclusion 80

References 82

List of Figures

Figure 2.1 Low-Pass Filters 7

Figure 2.2 Median Filters 8

Figure 2.3 High-Pass Filters 9

Figure 2.4 Sobel Edge Detection 10

Figure 2.5 Second Derivative Edge Detection 11

Figure 3.1 Chroma-keying Background Removal 14

Figure 3.2 Background Differencing 16

Figure 3.3 Background Primal Sketch Method 20

Figure 3.4 Ghosting Effect 21

Figure 3.5 Edge-Detected Background Method 23

Figure 3.6 Noise Removal – Remove Isolated Pixels 26

Figure 3.7 Noise Removal – Effects of Erosion 27

Figure 3.8 Histogram for Otsu's Thresholding Technique 29

Figure 3.9 Results of Otsu's Thresholding Technique 31

Figure 4.1 Wavelengths of Color 35

Figure 4.2 Skin Tone in RGB Color Space 36

Figure 4.3 Skin Tone in HSV Color Space 38

Figure 4.4 Skin Tone in YUV Color Space 42

Figure 4.5 Skin Tone in Normalized RGB Color Space 44

Figure 4.6 Results of Skin Detection – Test Case 'A' 47

Figure 4.7 Results of Skin Detection – Test Case 'B' 50

Figure 5.1 Example of Orientation 58

Figure 5.2 Boundary Algorithm 60

Figure 5.3 Pixel Sweep Technique 63

Figure 5.4 Distance Metrics 63

Figure 5.5 Medial Axis Transformation 64

Figure 5.6 Results of Medial Axis Transformation 65

Figure 5.7 Slope Representation of Chain Code 67

Figure 6.1 'Lowering Volume' Action 73

Figure 6.2 Background Removal Scheme 75

Figure 6.3 Noise Removal using Otsu 76

Figure 6.4 Contour Tracing – Pixel Sweep 76

Figure 6.5 Temporal Differencing 78

Figure 6.6 Skeletal Attraction Scheme 79

Abstract

To facilitate proper recognition of a human's action from a video sequence, several key features must first be determined. Initially, the person performing the action must be isolated from the background scene. This information is then used to decipher pertinent action attributes that may include the center of mass, contours, and regions of motion. It is these characteristics that will become the feature elements in the recognition of a person's actions.

This thesis will investigate the various image processing tools available to obtain the aforementioned action attributes. The applicability of filters, background removal techniques, skin-tone matching, and contouring schemes will all be investigated. A thorough comparison with both existing and novel approaches to action recognition is then discussed. Overall, the temporal based algorithm is best suited for an action recognition application as the spatially based approaches rely too heavily on *a priori* knowledge of the background scene.

Sommaire

Afin de faciliter la reconnaissance des actions d'un être humain dans une séquence vidéo, certains paramètres doivent être déterminés. Tout d'abord, l'être humain doit être isolé de la scène de fond. Cette information est par la suite utilisée pour discerner des caractéristiques importantes relevant des actions de la personne. Ces caractéristiques comprennent le centre de masse, les contours et les régions en mouvement.

La présente thèse analysera les divers outils de traitement d'images permettant l'obtention des caractéristiques mentionnées ci-haut. Ces outils comprennent des filtres, des techniques d'isolement de scène de fond, de corrélation de couleur de peau et de détection de contour. Une comparaison détaillée des techniques classiques et plus récentes de reconnaissance d'action sera également présentée. Enfin, un algorithme se servant de données temporelles plutôt que spatiales est plus adapté à une application de reconnaissance d'action. En effet, ce-dernier nécessite une connaissance trop détaillée de la scène de fond.

CHAPTER 1

1.1 Introduction

Although human gesture recognition has been studied for quite some time [Torige and Kono, 1992; Imagawa et al., 1998], action recognition is an area of research still at its infancy. The difference between these two terms can be best differentiated as follows: A gesture is a stationary pose of a user that may be classified within a single image. For example, pointing to an object requires only an outstretched arm along with a single finger extended towards the object in question. An action, however, requires a series of gestures or poses over time to properly identify what the user wishes to convey. This is perhaps best exemplified by a conductor of a symphony orchestra, who communicates effectively with the musicians through the use of pre-defined actions. Both the volume and tempo are communicated through actions that must be inferred by a sequence of images. A single image of the conductor at any one time would be insufficient to deduce either of these variables.

Perhaps the simplest way of deciphering an action is to observe how a person accomplishes such a task. For example, if one were to describe the action of waving 'hello,' they would not begin by describing the angle of the knee. The logical approach of describing this action may start with noting that the hand moves back and forth. This shows not only that the need to identify the location of the user is noteworthy, but also the relationship of the individual body parts. This encompasses two areas of computer vision: tracking and object recognition. These are both widely studied and many algorithms exist for such applications, however, few exist that specifically address the

field of action recognition. It is the goal of this thesis to review both existing and novel approaches to computer vision to better formulate an image segmenting scheme suitable for action recognition. We also wish to develop an algorithm that works not only in the laboratory, but also in a *real* world environment. If action recognition is ever to evolve into the mainstream, it must remain robust in any type of environment. The variables that are most commonly encountered in these scenarios are noise, occlusion, light intensity changes, and the overall unpredictability of the environment.

Noise in the computer vision domain is the introduction of erroneous pixel color values to the original image. Pixels resulting from noise are often referred to as *outliers* [Yang and Levine, 1992]. It may be a result of many different factors such as inherit camera noise, image transmission [Liebe, 1993], or compression-decompression techniques. Whatever the case, noise poses a significant challenge, as there is often no specific way of distinguishing color values introduced by noise, from true pixels. Some interesting techniques exist to reduce or eliminate outliers at the various stages of processing an image and will be addressed in more detail in Chapter 2.

When developing a robust tracking algorithm, one must consider the problem of occlusion. For instance, if a skin-based tracker is used to track a person's face while teaching a class, the algorithm must account for the scenario in which they turn their face towards the board to write notes. Partial and total occlusion of an object is a common occurrence in tracking scenarios. Solutions include movement bounding boxes, and optical flow to *predict* where the tracked object may be headed [Piaggio et al, 1998].

In the real world environment, light intensity can be expected to change frequently and randomly. As a person being tracked during a sunny day moves into a

shaded area, features such as color, texture, and even shape appear to change dramatically. Algorithms have been developed to deal with such situations through the use of histogram equalization, as well as edge-detection in concert with a low-thresholding scheme. This will be examined in further detail in Chapter 2.

Finally, the unpredictability of the real world gives rise to a substantial obstacle to overcome in vision techniques. For example, if tracking by shape alone, background objects may also resemble the contour of a person, thus forcing the algorithm to choose. Worse still, trying to predict motion of a person is at best, a guess. Some methods have been studied to better predict the motion of objects with the use of Markov models and Kalman filters, but as stated before, it is merely an estimation [Imagawa et al, 1998; Vogler and Metaxas, 1999].

Developing an action recognition algorithm requires investigation of two major fields of computer vision: tracking and object recognition. As these are vast fields of research in themselves, we confine our study to algorithms that are well suited for action recognition. Chapter 2 deals with possible pre-processing methods that are commonly used to aid in both the reduction of noise and to expose relationships between pixels, such as common regions or edges. Chapter 3 discusses segmentation algorithms that are based on knowing what is not being tracked in order to deduce where the object in question resides. This is commonly referred to as *background removal*. Chapter 4 investigates color-based tracking methods to ascertain their robustness for skin tone tracking. Chapter 5 reviews contour techniques to isolate areas of interest and their relationship to one another. Finally, Chapter 6 reviews the overall results to determine the best possible combination of techniques towards a robust, action recognition algorithm.

CHAPTER 2

FILTERS AND KERNELS

2.1 Introduction to Filters

Several preprocessing steps are often included in computer vision algorithms to help isolate and expose the appropriate data. [Yang and Levine, 1992; Arseneau, and Cooperstock, 1999a] Perhaps the most common such step involves the convolution of the original image with a small mask, known as the *kernel* or filter. Mathematically speaking, this process calculates the correlation between the image and the reversed kernel. In a discrete domain such as image processing, convolution of the image, f(x) with the kernel g(x) is calculated as follows:

$$h(x) = f(x) \cdot g(x) = \sum_{-\infty}^{\infty} f(u)\, g(x-u) \qquad (1)$$

In essence, processing is performed on a pixel by pixel basis, extracting the local characteristics of the pre-determined neighborhoods of each pixel. The most commonly used filters are low-pass or *smoothing* filters, high-pass or *sharpening* filters, and median filters. These will be described in the following sections.

2.2 Low-Pass Filter

This filter takes the local average of pixel values within a given neighborhood. In the global scheme, this has the effect of reducing outliers due to noise by producing a *smudging* effect. Although many outliers are eliminated, smoothing reduces the accuracy of the edges in the image. A 3x3 kernel reveals the optimum tradeoff by reducing noise,

yet not smoothing the edges to such a degree where they become negligible. (See figure 2.1)

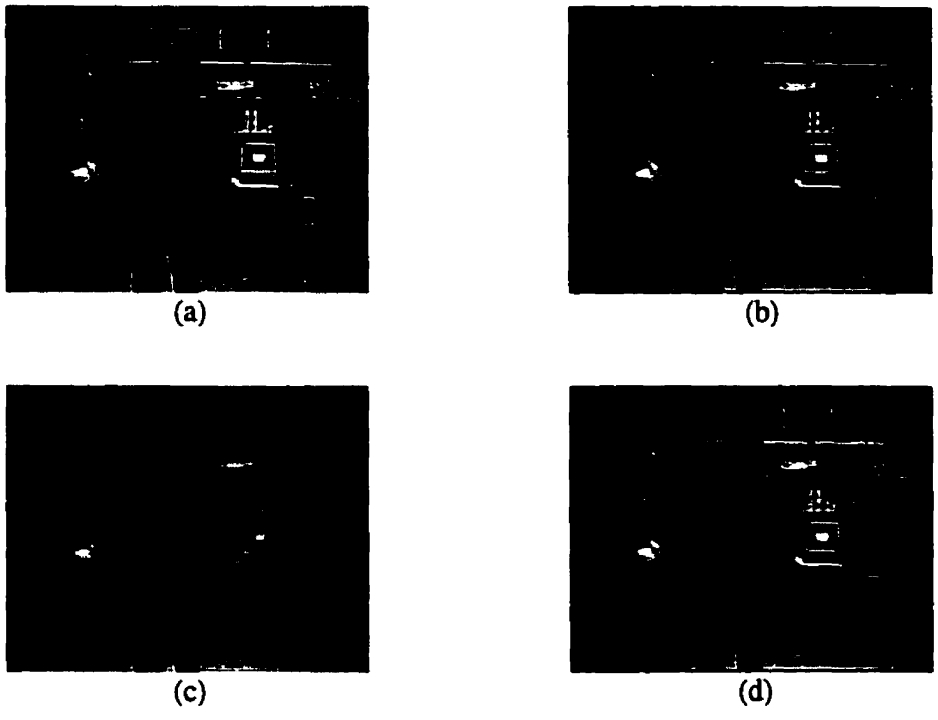


Figure 2.1, (a) grayscale image, (b) lowpass (average) 3x3, (c) lowpass 7x7, (d) Gaussian filter

The general purpose of low-pass filtering is to reduce the occurrence of large pixel variations within the image, hence reducing the gradient values throughout the scene. By increasing the kernel's size, the image becomes far more blurred. (Note figure 2.1c) It should be also noted that new pixel color values are introduced into the image due to the nature of this averaging scheme.

2.3 Median Filter

As the name implies, this particular filter determines the median pixel value within a given pixel neighborhood. This tool is also used to reduce the effects of noise, however, unlike the low-pass filter less gradient information is lost in the process (See Figure 2.2). By using the median, as opposed to the mean, the pixel value is less susceptible to spurious noise values that have occurred within the neighborhood.

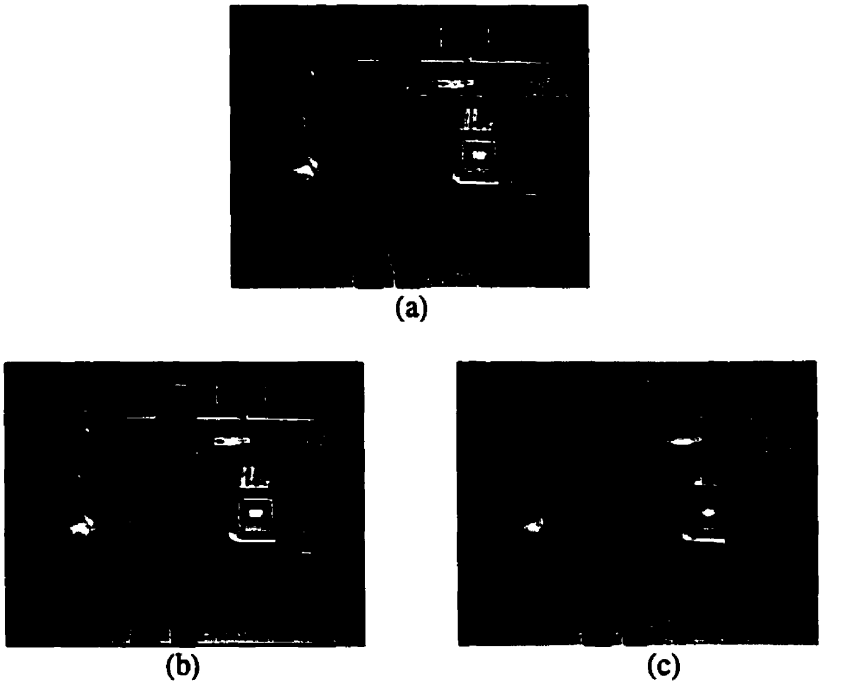
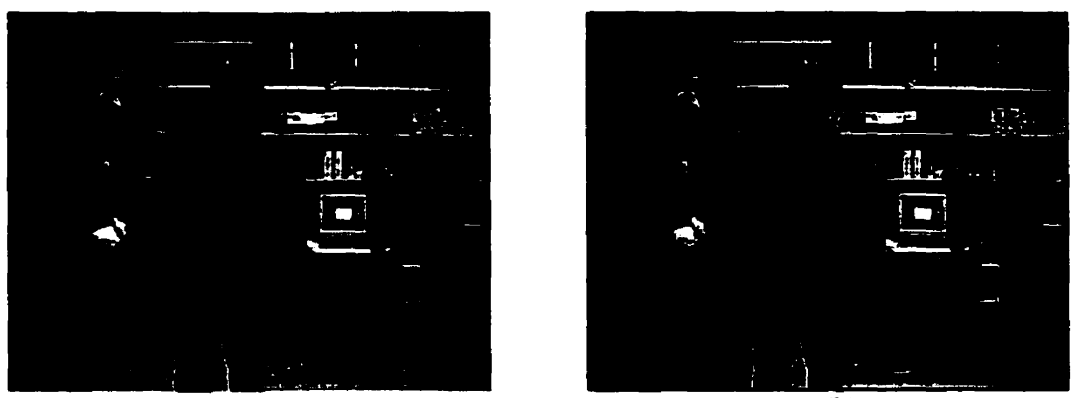


Figure 2.2 - (a) original grayscale image, (b) median 3x3, (c) median 7x7

This procedure avoids the tradeoff of low-pass filters and provides the researcher with a valuable tool to eliminate noise while maintaining edge information helpful in determining the general outline of the user. Furthermore, if color matching is to be performed, no new color values are introduced that were not present in the original image.

2.4 High-Pass Filter

High-pass filtering of an image *sharpens* the gradient information within the scene. Convolving with a kernel containing both positive and negative elements whose sum is one, results in an edge-enhanced image (See figure 2.3). While high-pass filtering tends to be pleasing to the eye, this process is also sensitive to noise, making it unsuitable as a pre-processing step for action recognition.



(a) (b)

Figure 2.3 – (a) Original image, (b) High-pass (3x3)

2.5 Dilation and Erosion

The next filtering technique to be discussed is normally used on binary images to vary the thickness of edges. *Dilation* amounts to replacing a pixel value with the maximum value among its neighbors. For color scenes, the resulting image becomes lighter in intensity while edges become thicker. This technique is most often used to merge edge segments in a binary image, as will be discussed further in Chapter 3.

Erosion is the same idea, but replaces the pixel value with the minimum value among its neighbors. This darkens color images and thins edges in binary images. This

serves as an efficient noise filtering technique when combined with other steps, as discussed in section 3.2.

2.6 Edge Detection

A common preprocessing tool applied to extract pertinent gradient information is known as an *edge detector*. This filter transforms the scene into a grayscale or binary image where the value of each pixel denotes the magnitude of the gradient. Also, the phase image is sometimes used to denote the direction of the gradient as a function of pixel intensity.

2.6.1 First Derivative Edge Detection

The most common form of an edge detecting filter is the first derivative of a pixel's neighborhood. Using a Sobel, Roberts, or Prewitt operator, the image is convolved with a kernel that transforms the scene into its first derivative equivalent [Levine, 1985]. An example using the Sobel operator is shown in figure 2.4.

(a) (b)

Figure 2.4 – (a) Original image, (b) Sobel edge detection

The advantages of a first derivative approach are its low computational cost and insensitivity to noise. Edges are preserved if the gradient value is above a pre-determined threshold, while edges with a gradual slope are ignored.

2.6.2 Second Derivative Edge Detection

By examining the second derivative of an image, edges are now identified by observing the *zero crossings*. The advantage of this method is that thinner edges are more likely to be identified through the technique, as opposed to the first derivative case. The graphical equivalent of this method can be seen in figure 2.5.

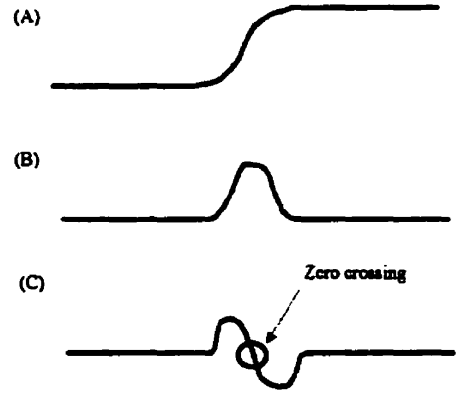


Figure 2.5 – Graphical equivalent of gray level gradients, (a) original, (b) first derivative of (a), (c) second derivative of (a)

The disadvantages of this scheme are that it is computationally more expensive and susceptible to noise within the scene. An interesting offshoot of this approach is known as the *Laplacian of Gaussian* [Marr and Hildreth, 1980]. This implementation involves convolving the scene with a Gaussian filter, followed by a Laplacian kernel. The

stipulation for an acceptable edge is that the zero crossing in the second-order must also have an equivalent first-order crossing above some pre-determined threshold. This method is often referred to as the *Mexican Hat Operator* due to the resulting kernel's shape.

2.7 Conclusion

A closer look will reveal that only a few of the filters discussed prove useful as a preprocessing step towards action recognition. The low-pass filter does not serve well as a pre-processing step for action recognition due to its combined blurring effect, and introduction of new color values into the image. The median filter however proves quite effective for reducing noise while retaining edge information. High-pass filters do not seem appropriate for this application due to their sensitivity to noise. Both dilation and erosion prove quite useful for extracting more coherent edge information, however using them as a pre-processing step to a color image yields little benefit. Finally, edge detectors prove quite useful to determine the gradient that could be used for contour extraction, which will be further discussed in Chapter 5.

Chapter 3

Background Removal Schemes and Noise Reduction Techniques

3.1 Background Removal Techniques

With a goal such as action recognition, it is vital to properly locate and identify specific parts of the user's body. This in itself is the most common challenge in computer vision, hence the plethora of literature devoted to the problem. One of the most widely used methods to deal with this challenge is known as *background removal* [Wren et al., 1997; Davis and Bobick; 1998 Arseneau and Cooperstock, 1999a]. In its simplest form as *chroma-keying*, the ease of computation has increased its popularity in such realms as special effects and television production, thus being a testimony to its efficacy.

3.1.1 Chroma-Keying

Background removal uses *a priori* knowledge of the background scene in an attempt to isolate the person in the image. In its simplest form, the background is made of a solid color or texture [Davis and Bobick, 1998]. The algorithm ignores all occurrences of a particular color within the scene to properly isolate the user. This technique is known as *chroma-keying*, or more commonly as blue-screening, as blue was the predominant color of backgrounds used in the past. It is used most often to overlay weather maps into the background for meteorologists, and proves quite effective. Having the distinct advantage of being one of the least computationally expensive methods, it is still plagued with fatal flaws. For instance, if the user is wearing a shirt that is similar in color to the pre-determined background, that part of the body is ignored in the resulting difference image.

The consequence of this is an image with a disembodied head and legs. Another flaw surfaces when changes in light intensity saturate the background to such a degree where it no longer falls within the acceptable background color range. This results in large blobs of false positives appearing in the difference image.

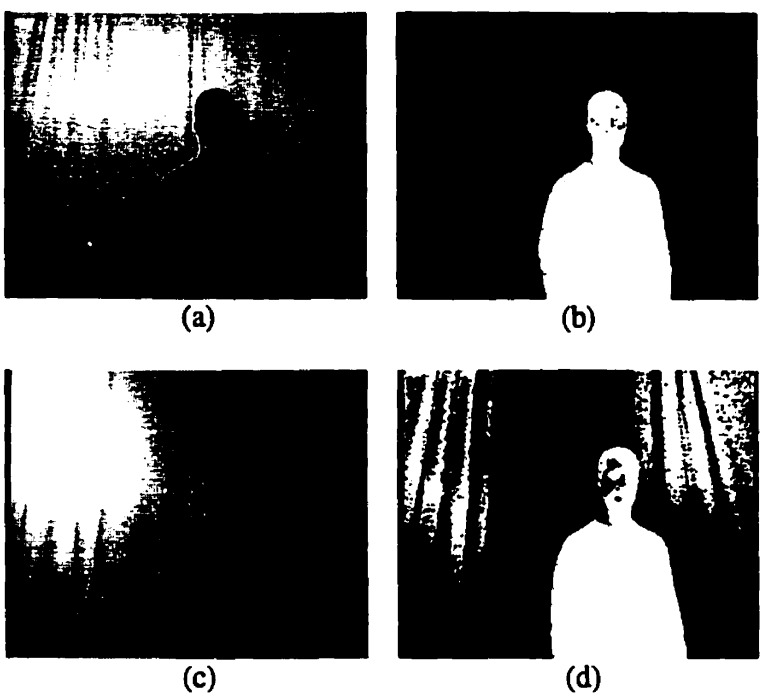


Figure 3.1 - Chroma-keying/Fixed Threshold - (a) original image against solid background, (b) difference image for (a), pixels within color range removed to show user (in white), (c) another scene with light intensity change, (d) difference image for (c)

To demonstrate this method, a solid white background was constructed to test the validity of chroma-keying. (Figure 3.1) After setting the color and lighting intensity, the technique was performed on an image, (figure 3.1a) revealing fantastic results outlining the user in the center (figure 3.1b). However, when the lighting changes such that the intensity becomes higher than the pre-determined threshold, many false positives are

created. (See Figure 3.1d) Note also that parts of the user are labeled black indicating background, however since the light reflects off of the face, the algorithm is fooled. One might suggest an increase in the threshold to such a point that the noise is removed, but this would remove correctly identified user pixels as well. In summary, this method is the least computationally demanding, however, it is highly restricted in its application domain requiring uniform lighting conditions, and restricting the color of the user's clothes.

3.1.2 Background Differencing

The next stage in the evolution of background removal, is to take a snapshot of the background without the user in it to obtain the *a priori* knowledge of the scene. This information, namely the pixel color values, is compared against the incoming video sequence in order to remove the background. Known as *background differencing*, this method is analogous to constructing a chroma-keying scheme on a pixel by pixel basis. Without the restriction of having to create a mono-colored backdrop, background differencing is becoming the basis of many researchers' vision algorithms. [Huang et al. 1986, Kahn and Swain, 1995; Davis and Bobick, 1997] By performing the calculation on a pixel by pixel basis, the background has fewer constraints as to what it may include. If F_{xy} is the pixel value at (x, y) in the original image, and B_{xy} is the pixel value at (x, y) in the background image, the binary, difference image D is constructed using equation 1, where T is the threshold value.

$$D_{xy} = \begin{cases} 0, \text{ if } | B_{xy} - F_{xy} | < T \\ 1 \text{ otherwise} \end{cases} \qquad (1)$$

If all objects in the background remain static, the resulting difference image reveals promising results. However, note that little is mentioned about the threshold value *T*. Unfortunately, there is no universal threshold as a value too low will include too many false positives, or *outliers*, in the difference image. At the other extreme, if the value is too high, many false negatives will result in parts of the user disappearing. One restriction that applies to background differencing, but not to chroma-keying, given that the background always remains uniformly colored, is that the camera must remain stationary. Even the slightest movement would render the difference image useless as all of the pixel values would be shifted, resulting in a false background. Thus, the pixel by pixel calculation would result in a pixel falsely being identified as the user wherever a high frequency edge occurs in the background scene. Also, background differencing is still susceptible to lighting changes as a result of the fixed threshold for the entire image.

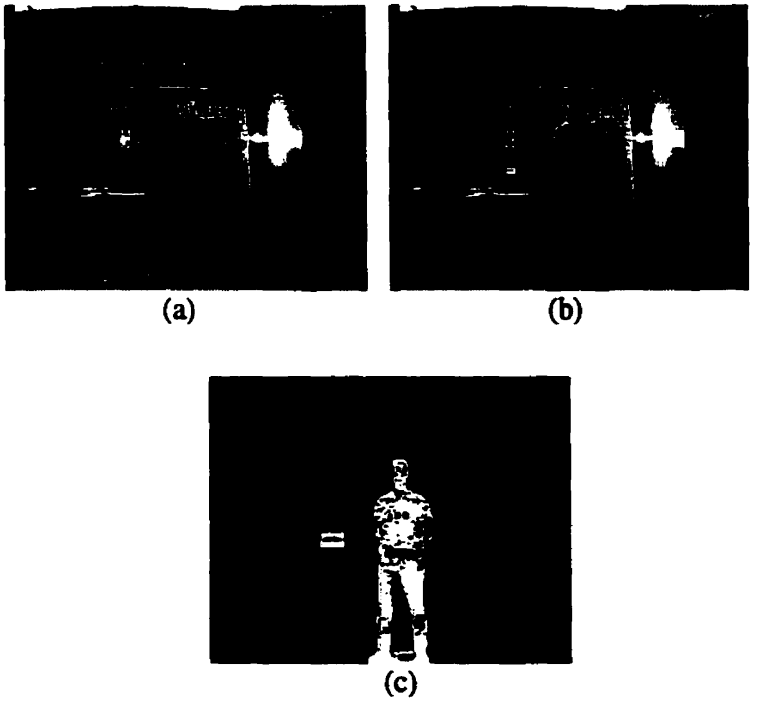


Figure 3.2 - Background Differencing - (a) background image, (b) scene with user, (c) difference image

Figure 3.2 shows an example of background differencing with a multi-colored background scene. Figure 3.2a shows the background image used, i.e. excluding the user, which is then subtracted from the incoming scene (figure 3.2b). If the difference exceeds a constant threshold, a white pixel is drawn, otherwise, it is black, denoting what is believed to be the background. The resulting difference image, (figure 3.2c), shows the user clearly outlined, however the white square to the left is a result of the display frequency of the monitor. A closer look reveals that the scan line is near the bottom of the monitor's screen in the background image, whereas it is near the top in the incoming scene, resulting in falsely identified user pixels. Although it cannot be seen at this scale, the difference image also results in a few stray pixels that were triggered as a result of the lamp in the background. Both the scan line and the lamp vary the lighting intensity of the background, hence producing additional *candidate pixels* or pixels denoting sufficient change in the difference image.

3.1.3 Background Primal Sketch

In an attempt to overcome the limitations of the previously discussed algorithms, the *background primal sketch* was introduced [Yang and Levine, 1992]. The background primal sketch, B_{xy}, is constructed by taking the median value of each pixel over a sequence of N background images $F^1_{xy}, F^2_{xy}, \ldots F^N_{xy}$, without the user:

$$B_{xy} = \text{median}\{F^1_{xy}, F^2_{xy}, \ldots F^N_{xy}\} \qquad (2)$$

The median value rather than the mean is used as it is more robust to spurious outliers [Rousseeuw and Leroy 1987]. For instance, if the sampled pixel values for given point over a sequence of five images was {80, 82, 83, 85, 130}, one would likely infer that 130 is an outlier and thus the median, (83), would be a far better estimate than the mean, (92).

Robustness is normally referred to as the property of insensitivity to *stray* data points. To measure robustness, the term *breakdown* was introduced as, "...an estimator of the smallest fraction of the data that has to be replaced to carry the estimator over all bounds" [Hampel et al. 1986]. The breakdown point of the mean of n samples is $1/n$. Thus the replacement of only one sample by an outlier can greatly affect the mean. However, the median has a breakdown point of 50%, in other words, at least half of the samples must be replaced by outliers before serious corruption results.

Yang and Levine decided to incorporate outliers into the calculation as they are not always due to noise, but may result from a legitimate scene change. The practice of leaving the outliers in the calculation is categorized as an *accommodation-based* method [Yang and Levine 1992]. An example of this would be the least-median-square approach. It is also common to attempt to identify which of the points in the data set are outliers, eliminate them, and continue using a less robust technique, such as the least-mean-square. For the most part, it is preferable to use an accommodation-based approach as points that are far from the mean may be correct and should exude influence, be it less than those near the mean, on the final results. Other methods attempt to lessen the effects of outliers by imposing a weighting scheme to the data such that all points nearer the mean will have a greater influence [Hampel et al. 1986].

The second step of the background primal sketch is to account for minute changes in the camera position. This is accomplished with a slight modification to the method used for chroma-keying by comparing the pixel value in the incoming image not only to its positional equivalent in the background scene, but also to those pixels in the $2n$-neighborhood of the background pixel, as indicated in the following equation:

$$D_{xy} = \begin{cases} 0 & \text{if } | B_{ij} - F_{xy} | < T, \text{ for any (i,j)} \\ 1 & \text{otherwise} \end{cases} \qquad (3)$$

Note that F_{xy} is now compared to B_{ij} where $i \in \{i\text{-}n, \ldots, i, \ldots, i\text{+}n\}$ and $j \in \{j\text{-}n, \ldots j, \ldots, j\text{+}n\}$ as opposed to equation 1. This implies that each pixel of the incoming frame is compared against a neighborhood of ($2n$ x $2n$). If any of the differences are less than a pre-determined threshold, then the pixel is not a candidate pixel of the person in the scene. As the value of n increases, the camera may shift its view by larger angles. However, fewer pixels will be identified as candidates in this case. The first condition of equation 3 excludes any pixels with at least one difference value less than the threshold.

The third step is thresholding to account for areas in the background that may be more susceptible to lighting changes, such as windows or computer monitors [Yang and Levine, 1992]. Again, this is accomplished on a pixel by pixel basis. Gathering the set of pixel values over N images, a sorting is performed using a method similar to a bucket-sort technique. Next, these values are grouped into smaller sequences of length $N/2$. The sequence with the smallest *span*, or the difference between the maximum and minimum

value within the sequence, is then chosen as the threshold range for that particular pixel location.

The result of these steps is a background removal method that is far more robust than background differencing, as the dynamic threshold deals with areas in the scene that may be more susceptible to lighting changes.

For example, in the difference image of figure 3.3, the candidate pixels resulting from the video monitor are now eliminated as compared to figure 3.2c, due to the dynamic thresholding technique. Although there are still outliers, the improvement over a simple background difference method is dramatic.

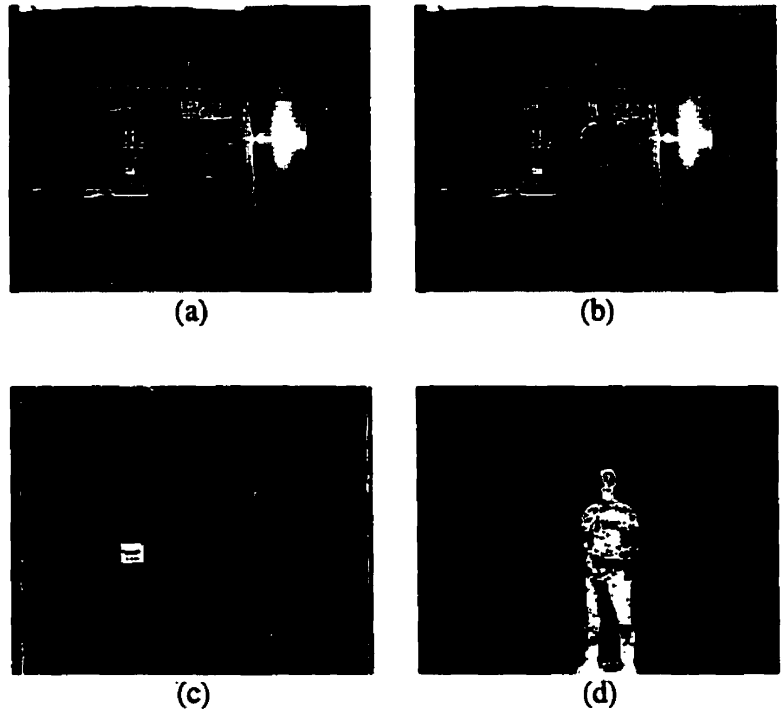


Figure 3.3 - Background Primal Sketch Method – (a) Background primal sketch, (b) scene with user, (c) threshold image where intensity denotes threshold value at that pixel location, (d) difference image

3.2 Ghosting

One of the constraints for simple background removal techniques is that background objects must remain stationary, otherwise a *ghosting* effect will ensue (See figure 3.4). This limitation discourages the use of this method as an action recognition tool since fully static backgrounds are relatively uncommon in the real world. However, in the case that the background scene is stationary, this method proves very accurate. This observation motivates a slight variation of technique.

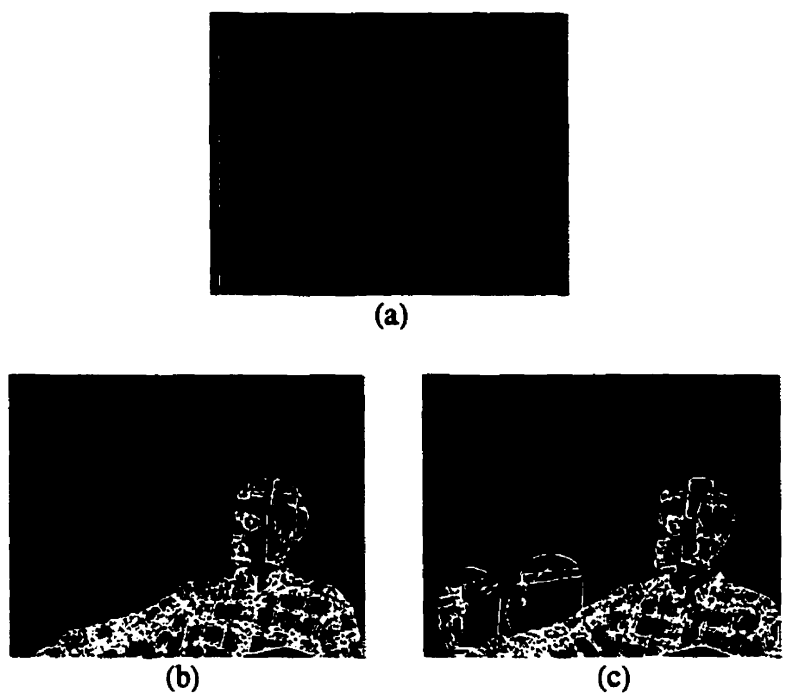


Figure3. 4 – (a) edge-detected background, (b) outlier with user, (c) outlier with user and ghosting due to chair

One novel approach to reduce the effects of ghosting is to update the background primal sketch periodically. For instance, suppose there was a closed door in the background scene when the sketch was constructed. If the door is opened, the resulting difference image will then contain both a moving user and the region denoting changes in the door

position. By updating background images in the manner of a *shift register*, the reconstructed primal sketch will eliminate the ghosting due to the door. Once there are n images of $(2n\text{-}1)$ in the register with the new door position, the median property will ensure that the new background primal sketch is updated properly to account for the new door position.

Another method to reduce ghosting considers only those pixels within a smaller, cropped region around the user known as the *region of interest* (ROI), while all of the pixels outside of this region denote background. This has the advantage of both reducing the complexity of the calculation, as well as inherently reducing the effects of ghosting. If the user can be identified throughout a range of images, the location information can be used to create, for example, a rectangular ROI about the user. Again, as in the shift register method, the pixels that lie outside the ROI can be updated to reflect the most current state of the background. Any scene changes that occur outside the ROI will not lead to ghosting, provided the sketch is updated sufficiently often so that the user's movement does not cross into a background area that has changed since the last update.

Although these techniques significantly reduce the effects of ghosting, they cannot eliminate it entirely. For instance, if the light intensity changes dramatically, candidate pixels will result from both the light source and the user. Thus, it may be helpful to employ a variety of methods in concert to deduce the specific action being performed.

One such approach, illustrated in figure 3.5, preprocesses the images using an *edge-detector* (see Section 2.6) such that only the outlines of objects in the scene are subtracted [Yang and Levine 1992].

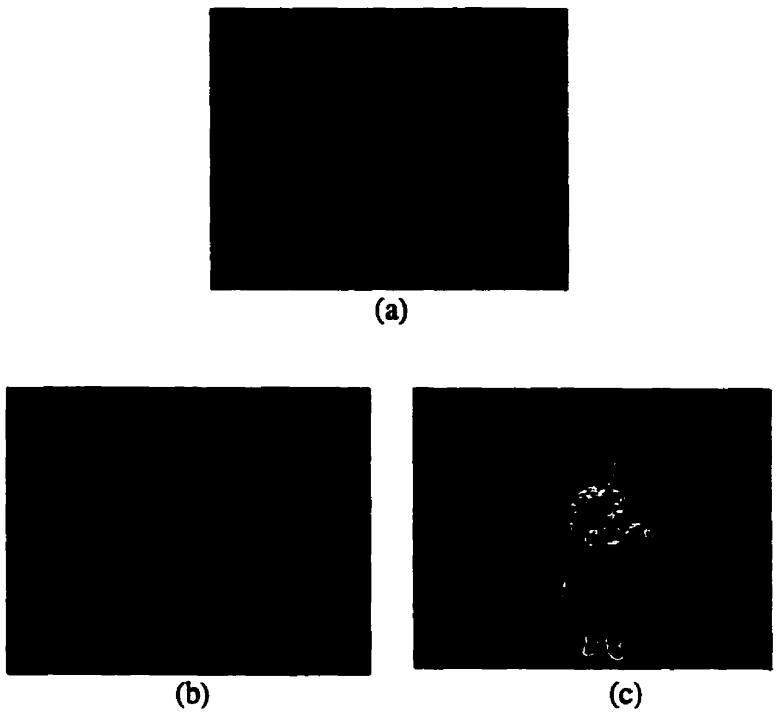


Figure 3.5 – (a) Background (edge-detected), (b) original image (edge-detected), (c) difference image

This added step greatly reduces the candidate pixels due to ghosting while retaining pertinent edge information. In the example of figure 3.5, only the pixels along the aura of the light source would contribute towards ghosting, therefore eliminating many of the false positives. One potential drawback of edge-detected images however, is that they provide less information about the center of mass of the person, which is sometimes useful for simple tracking algorithms [Arseneau and Cooperstock 1999b]. However, in the action recognition realm, the contour of the user often proves quite important. For instance, to recognize a pointing gesture, the outline of the arm and its location with respect to the rest of the body is of greatest importance, while the center of mass is of little interest.

In general, gradient information with contour methods proves useful in action recognition. Further discussion of outlining or *contouring* techniques is discussed in Chapter 5.

Another approach to person tracking could use *histogram differencing*. With the challenge of varying light intensities, it is conceivable to compare a normalized histogram of the background against a normalized incoming scene. Whereas the primal sketch technique attempts to account for areas in the background scene that may be more susceptible to varying light intensities by taking the median over a sequence of images, it relies on the assumption that any relevant variations will occur during that specific span of time. Taking advantage of the fact that the shape of the histogram remains fairly constant as the global light intensity varies in RGB space, preprocessing the image through normalization helps eliminate false positives in the difference image. For instance, if a background primal sketch were created in a dimly lit room, the introduction of a bright, global light source would drastically increase the number of false positives in the difference image. However, noting that objects retain their respective values in HSV space, varying mainly in saturation due to changes in lighting intensities, a more robust estimation can be made as to whether a pixel value belongs to the user or background. While this technique has not yet been tested it provides a unique perspective to the relationship between color values over a range of light intensities, as will be discussed further in the following chapter.

3.3 Motion Information

A newer train of thought that seems to be catching on in the action recognition community is that of *motion detection*. The goal is to locate the change in the scene from image t to image $t+1$. Closely related to background removal, it is a kin to continuously updating the background scene with the present image once the difference image has been obtained. Davis and Bobick have implemented this technique using *temporal templates* to identify actions of a user, with a high degree of success. [Davis and Bobick, 1997] This approach has many advantages over background removal techniques. Firstly, there is no dependence on a priori knowledge of the background scene prior to function properly. Also, if there is a sufficiently high frame rate, the light intensity changes over time will not adversely affect the outcome due to its temporal characteristics. This technique has been used to successfully track people in a classroom environment by noting the shape of the resulting motion, [Arseneau and Cooperstock, 1999b], as well as directing an avatar while interacting with performers on stage [Pinhanez and Bobick, 1998].

3.4 Noise Reduction Techniques

Once a reasonable difference image is constructed, there is still the challenge of reducing candidate pixels due to noise, while retaining as many of the correctly identified pixels as possible. Outliers resulting from noise often appear as *isolated* pixels in the difference image. As this is the prime reason for errors in the final result, a few techniques for removing outliers are evaluated.

3.4.1 8-Connected Isolated Pixels

Isolated in this context refers to a labeled candidate pixel whose 8-connected neighborhood consists entirely of non-candidate pixels. A common technique of removing these pixels is to *erode* the image, as discussed in Chapter 2, such that the image contains fewer *stray* pixels (See figure 3.6c). Unfortunately, while many of the outliers due to noise are removed, this technique also eliminates many of the candidate pixels denoting the user. A better alternative eliminates candidate pixels that have only non-candidate pixels in their four or eight-connected neighborhood. Adding this constraint removes isolated candidates while ignoring others that are part of a larger connected region, most likely denoting the user (See figure 3.6d).

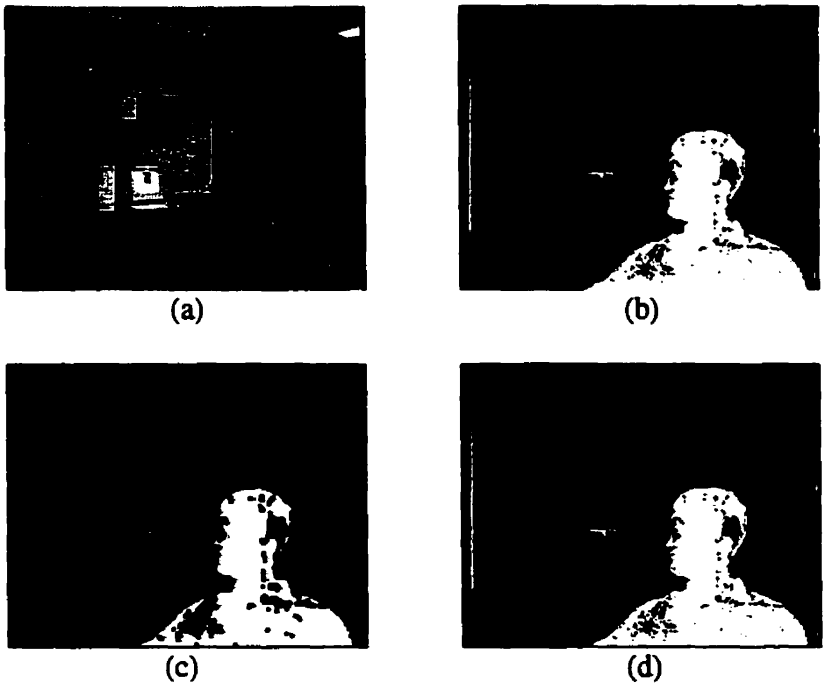


Figure 3.6 – Color differencing - (a) Original image, (b) difference image, (c) eroded difference image, (d) difference image with isolated pixels removed

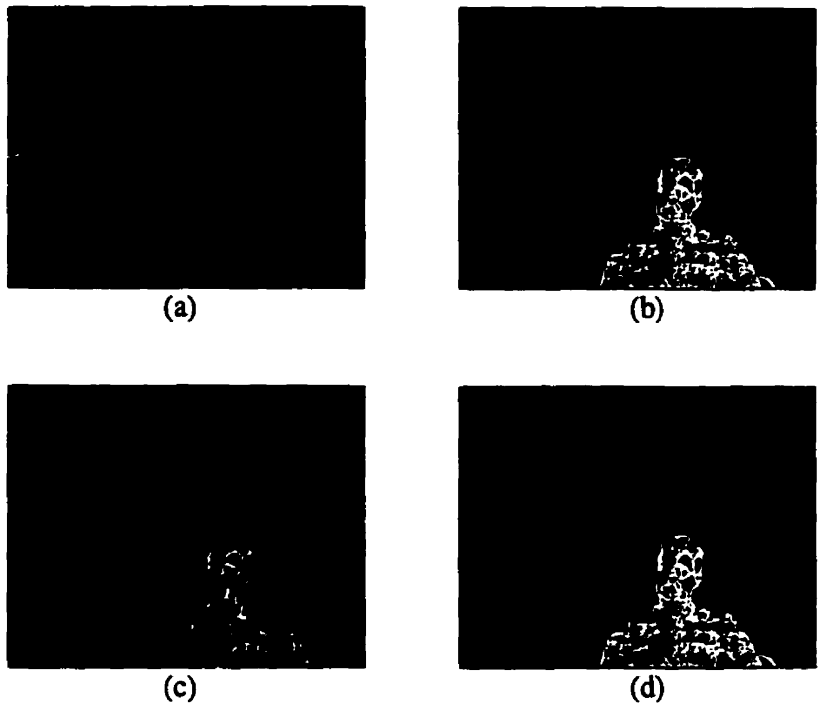


Figure 3.7 – Edge differencing - (a) Original image, (b) difference image, (c) eroded difference image, (d) difference image with isolated pixels removed

The summation of the $2n$ neighborhood is calculated to determine if a pixel is isolated, as shown in equation 4.

$$\text{If } D_{xy} = 1, \text{ (denoting a candidate pixel)}$$

$$F_{xy} = \begin{cases} 0 \text{ if } \sum_{i=x-n}^{x+n} \sum_{j=y-n}^{y+n} D_{ij} = 1 \\ 1 \text{ otherwise} \end{cases} \qquad (4)$$

This assures that given a neighborhood of $2n$ x $2n$, outliers that are connected to a larger region are not eroded, while at the same time, isolated pixels in the difference image D are removed.

It is worth noting that erosion and isolated pixel removal have very different results on images that were derived from gradient methods, as opposed to color

differencing. Since the color-based difference image results in a dense blob-like scene, *eroding* successfully removes many of the outliers, while preserving most of the shape of the region.

The isolated pixels technique is less successful at eliminating outliers but preserves both the size and shape of the user. When these same two techniques are performed on a difference image derived from an edge-detected scheme, the results are quite different. Eroding the image now removes far too many candidate pixels as the gradient image consists primarily of thin lines denoting edges. The eroding scheme risks eliminating these edges due to the neighborhood style calculation. (For details, see Chapter 2). In this case, the isolated pixel removal proves the better of the two schemes as it successfully removes many of the outliers, while ignoring chains of pixels, normally denoting edges in the scene. It is evident that for maximum effectiveness, the choice of noise removal technique should depend on the various methods used prior to this step.

3.4.2 Otsu Thresholding Method

Another noise reduction technique employed eliminates the outliers that form as a result of low-gradient edge values. This assures that only candidate pixels denoting sharp edges are kept, however, choosing a threshold is a challenge in itself. There exist many different methods of histogram manipulation in order to choose a reasonable threshold such that the true *valley* is found, as opposed to a local minima. Otsu's method [Otsu 1979], a non-parametric, unsupervised approach, based on the characteristics of the gray level histogram proved ideal. This approach is far more robust when the resulting *peaks* are of different magnitudes. As can be seen in figure 3.8, the histogram has a distinct

valley between two peaks, however the heights of these peaks as well as the distribution can vary widely from image to image.

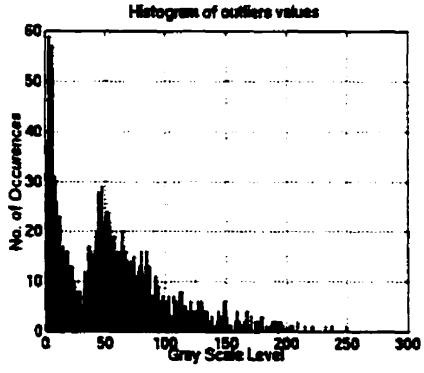


Figure 3.8 - Histogram of difference image, Otsu's threshold at gray level 33

Some researchers have tried to manipulate the histogram such that only pixel values also exhibiting high gradient values are used [Weszka 1974]. This type of approach uses neighboring pixels to ascertain a reasonable threshold, while other approaches deal specifically with the shape of the histogram itself. Through the use of parametric techniques, the histogram is fitted to a Gaussian type distribution. The mean and variance are then used to choose a threshold [Fukunage 1972]. The problem with this technique, like so many others based on a Gaussian distribution, is that data does not necessarily fit such a distribution.

Otsu's approach employs the zeroth and first-order cumulative moments of the grayscale histogram. He noted that many images with a single object and background could be categorized by finding the *valley* in the histogram and using this value as the threshold. The pixel values denoting the foreground were most likely on one side of the threshold with the background on the other.

The first step of Otsu's method is to construct the gray level histogram consisting of L levels. The number of occurrences of gray level n is denoted as n_i, and the total number of pixels S is equivalent to the sum of $\{ n_1, n_2, \ldots, n_L \}$. The histogram is then normalized and treated as a probability distribution function:

$$p_i = n_i / S, \quad \text{where} \sum_{i=1}^{L} p_i = 1 \tag{5}$$

The problem now is to identify the class of pixels C_0 as those with a value less than or equal to the threshold k, and those above k as C_1. The probabilities of each class occurring, W_0 and W_1, as well as the class means, u_0 and u_1, are then calculated as follows:

$$W_0 = \Pr(C_0) = \sum_{i=1}^{k} p_i \tag{6}$$

$$W_1 = \Pr(C_1) = \sum_{i=k+1}^{L} p_i \tag{7}$$

$$u_0 = \sum_{i=1}^{k} i \Pr(i \mid C_0) = \sum_{i=1}^{k} i\, p_i / W_0 = u(k)/W_0 \tag{8}$$

$$u_1 = \sum_{i=k+1}^{L} i \Pr(i \mid C_1) = \sum_{i=k+1}^{L} i\, p_i / W_1 \tag{9}$$

The zeroth-order (equations 6 and 7), and the first-order u(k) (See equation 8) cumulative moment are used to calculate a measure of class separability. Discriminant analysis is then used to find the optimum threshold [Fukunage 1972]:

$$\sigma_B^2 = W_0 W_1 (u_1 - u_0)^2 \qquad (10)$$

The problem then simplifies to finding the optimum threshold k^* that maximizes σ_B. This technique was performed on the histogram in figure 3.8 and successfully chose a threshold of 33, which falls within the *valley* region, thus logically separating the histogram in two.

All pixels that fell below this threshold were discarded, resulting in an image in which all of the candidate pixels denoted high gradient values in the original scene (See Figure 3.9).

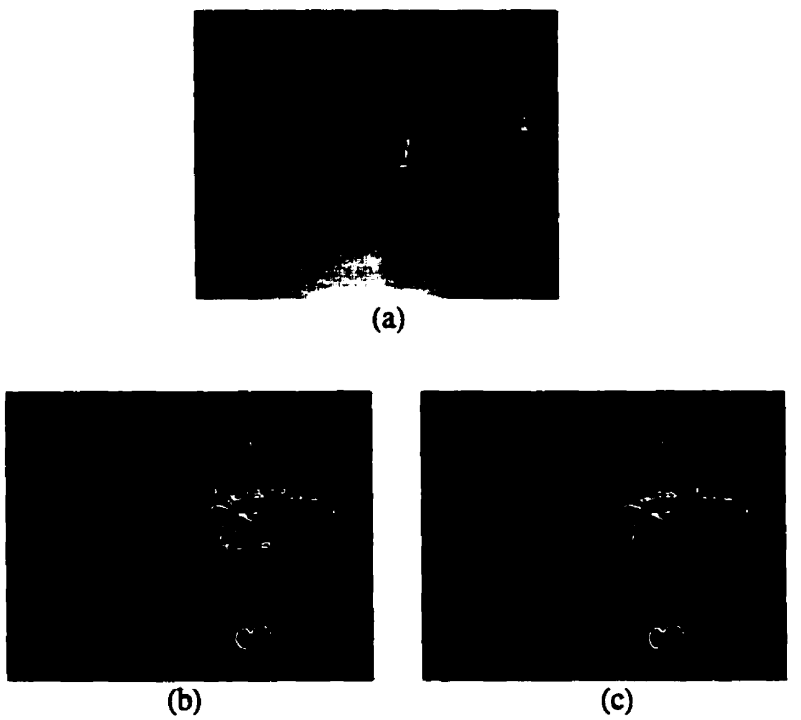


Figure3. 9 – (a) Original image, (b) difference image before Otsu, (c) after Otsu

As an experiment, Otsu's thresholding method was tested with color values instead of gradient values. In RGB space, eliminating all pixel values below a threshold translates

to eliminating dark colors. With a difference image constructed from color rather than edge-detected scenes, a person's shadow often resides in the lower region of the resulting histogram. Otsu's method eliminated most of the undesirable shadows, but had the unfortunate side-effect of also removing parts of the people if they were wearing dark colors.

3.5 Conclusions

Background removal techniques are able to produce valuable information about a user in the scene. Examples include the center of mass, contour, and location of body parts. This information can then be used to ascertain many different variables to facilitate an action recognition feature vector, as will be discussed in Chapter 5. The resulting difference image, however, is highly dependent on the environmental variables at play during the image sequence. For instance, the camera must remain stationary throughout the sequence to produce a satisfactory image and the background should remain static in order to avoid ghosting effects. Since computer vision algorithms must be robust in all types of scenarios to succeed outside of the laboratory environment, background removal is perhaps too overly restrictive. Under proper conditions, this technique proves to be a useful tool for action recognition, but due to its inherit restrictions it is best used in conjunction with other techniques to identify candidate pixels properly.

Chapter 4

Color Detection

4.1 Color Models

The study of colors has been a realm of great debate for hundreds of years. In scientific terms, different colors arise as a result of the spectral content of radiant energy that emanates from a given object. Sir Issac Newton performed the first recorded study of colors in 1704 with his prism experiment [Levine, 1985]. Newton successfully proved that light is formed from different, monochromatic (single-wavelength) colors. This became the foundation upon which trichromatic theories are based today. It is interesting to note that Newton actually chose red, yellow, and blue to represent the three chromatic portions upon which any other color made be made, however, the more popular red, green and blue combination emerged as a result of its similarity to characteristics of the human eyes'. [Levine, 1985] The cones on the inner wall of our eyes react specifically to stimulus of wavelengths denoting red, green and blue, hence our natural tendency towards this combination of colors.

Many computer vision applications such as face recognition and person tracking can be simplified due to the common characteristic of skin tone. Since this type of application relies upon human characteristics, one can attempt to discern the color information from a scene to simplify calculations. This approach is one of the most popular for identifying human characteristics due to its simple *compare and contrast* evaluation. The alternative classification of contours and shapes is discussed in detail in Chapter 5. A typical color matching scheme that produces a color-identified result *S*, often takes the form of equation 1.

$$S_{xy} = \begin{cases} 0 \text{ if } |C - F_{xy}| < T \\ 1 \text{ otherwise} \end{cases} \qquad (1)$$

where each pixel location (x, y) in frame F is compared against a pre-determined color C as to whether or not it exceeds some heuristic threshold T.

Many researchers have made the color matching scheme the basis of their algorithms [Yang and Waibel, 1995; Bregler, 1997; Ayers and Shah, 1998], as the range of possible skin tones is limited. However, color alone is insufficient. Some researchers argue that there exists a *universal* skin tone in which all tones are accounted, from the darker shades of Africans, to the lighter shades of Caucasians [Chai and Ngan, 1999]. This amounts to stating that there exists a region in RGB space that encapsulates all possible values of skin tone. This is not a terribly profound statement as it simply expresses the fact that a specific color exists in a certain region of RGB space, as do all other colors! Of course, it is the shape of this region that is important. Whether the RGB values are used directly [Birchfield, 1998] or indirectly, by converting to HSV, YUV, or some other transformation [Bregler, 1997; Ayers and Shah, 1998], specific values to define the region must be determined *a priori*.

4.1.1 RGB

The most basic color scheme is red, green, and blue (RGB). As mentioned earlier, these three, monochromatic colors in combination can produce any other color in the visible spectrum. Figure 4.1 shows these colors as a function of wavelength [Lammens, 1994].

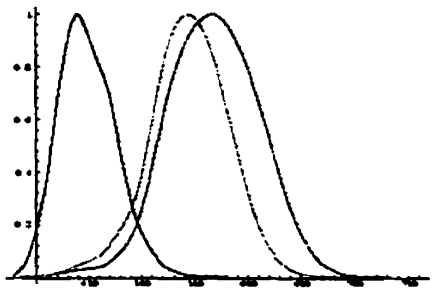


Figure 4.1 – Color as a function of wavelength, (nm) [Lammens, 1994]

In the field of computer vision, this model is seldom used as a method to detect skin tone directly. This is mainly due to its inherit inseparability of colors into easily distinguishable regions. For instance, most skin tone based algorithms begin by accumulating a data set of known skin colors at various lighting intensities [Campbell et al. 1997]. Next, either a lookup table is implemented [Ishibuchi et al, 1992], the mean and variance are calculated directly from the data [Yang and Waibel, 1995] or an unsupervised learning approach is implemented [Campbell et al. 1997]. This translates into a three-dimensional region in RGB space in which all pixel values are labeled as skin tone candidates, while those exterior to the space are not. It is vital that the region denoting skin tone is small enough in proportion to the entire color space such that a lookup table (LUT) is minimized to reduce processing time. However, it must include enough tones to account for skins of varying pigment and different lighting conditions. Figure 4.2 exhibits the RGB values of varying skin tones, grouped by pigment as Asians, Caucasians, Latin Americans, and Africans. As is evident from the graphs, all of the different skin pigments evaluated occur along a common diagonal region protruding from the origin, and stretching towards full white, (RGB=255,255,255). This to be expected due as to account for most skin tones scenarios, the swatches were taken using different

lighting intensities. Images of humans can occur in any lighting condition, from in shadows to an intense sunlight or electronic flash. The final outcome produces swatches ranging from near-black values (RGB=0,0,0) to full white, with the proportions of R, G, and B being roughly equal.

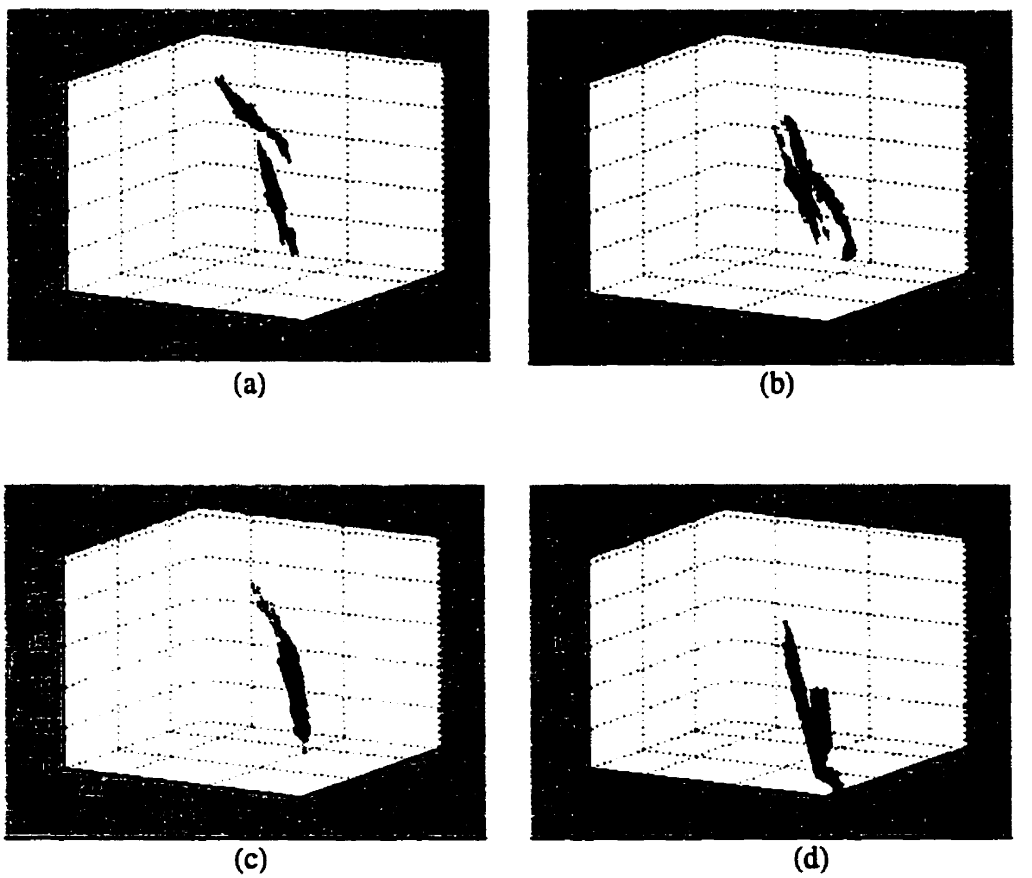


Figure 4. 2 – RGB color space denoting different racial skin tones deriving from, (a) Asian, (b) Caucasian, (c) Latin American and (d) African descent

Even though the region denoting various skin tones seems sufficiently clustered, applying a color matching scheme using RGB space does not always yield good results [Zarit et al. 1999]. What is needed is a color space that is less susceptible to variances in lighting

intensity, and deals specifically with color, not shade. This requirement led to the development of hue, saturation, and value color space (HSV).

4.1.2 HSV

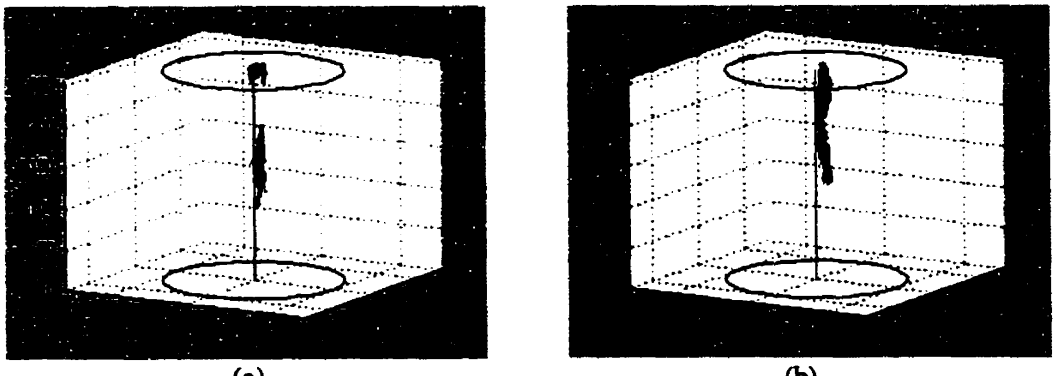
The shape of the HSV space allows for a better understanding of colors as the pixel location moves about its cylindrical coordinate system. The *hue*, for instance, spans from 0 to 360 degrees, and normally denotes the *family* of color. This value is most closely related to how humans distinguish between colors, by referring to the general color type, such as red or yellow. The distance from the center of the cylinder, or *saturation*, refers to the strength of the color. A saturation of 0 signifies a color in the grayscale range. As this number increases, the contribution of the color family associated with the angle has a greater effect. For example, travelling from the center along the green hue will grow from a dark green to an intense green. Finally, height within the cylindrical space denotes the *value* component. This translates to how dark or bright a color family becomes. The value is synonymous with brightness or grayscale.

The calculation of HSV from RGB is more complex than most other color spaces, as it must transform a Cartesian space into cylindrical coordinates (see table 1).

Using HSV color space to amalgamate skin tone values produces a closer knit region than RGB space, and hence is much faster as a lookup table scheme, as can be seen from figure 4.3. More importantly, there exists a specific hue region that encapsulates many pigments of skin under varying lighting conditions.

Table 1 – Steps to transform a given value of RGB to HSV

H – (0-360 degrees), S – (0, 1), V – (0, 1)	
1	Normalize RGB values
2	Determine the global minimum and maximum value of RGB
3	***V = maximum value***
4	Delta = maximum – minimum
5(a)	If (maximum != 0), then ***S = Delta/maximum***
5(b)	Else, ***S = 0, and H = undefined*** (i.e. Since this occurs at the center of the cylinder, no angle is appropriate
6(a)	If(R = maximum), then ***H = (G – B) / delta***
6(b)	Else if (G = maximum), then ***H = 2 + (B – R)/delta***
6(c)	Else, ***H = 4 + (R – G) delta***
7	H = H * 60 (to convert to degrees)
8	If (H is negative), H = H + 360

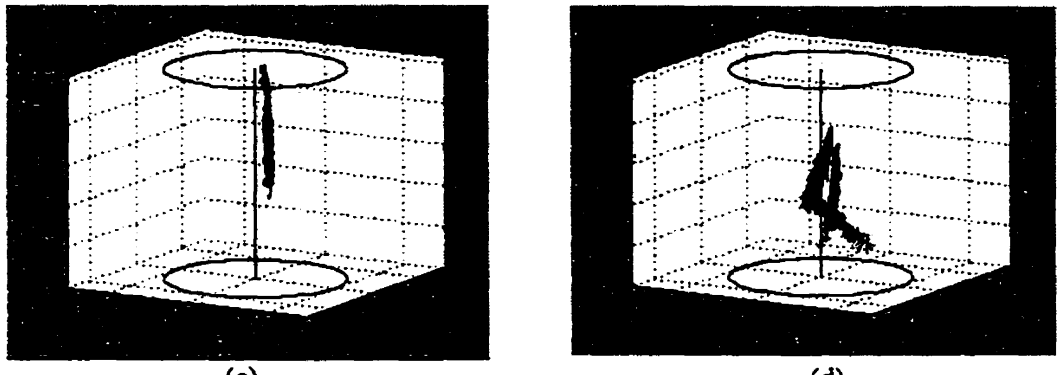


(a) (b)

Figure 4.3 – Skin tone values denoted in HSV for people of

(a) Asian and (b) Caucasian descent

* The vertical line denotes the grayscale pixel values within this cylindrical coordinate system



(c) (d)

Figure4. 3 (cont'd) – Skin tone values denoted in HSV for people of (c) Latin American and (d) African descent.

* The vertical line denotes the grayscale pixel values within this cylindrical coordinate system

The intuitive architecture of the HSV space has inspired many skin tone detection algorithms. Bregler found that the data set identifying skin tone within HSV space was very effective [Bregler, 1997], and used this to narrow down the possible poses of a user's legs. However, there is no mention as to how the data set was compared against the incoming scene, whether it uses a lookup table or statistical variables.

HSV has also been adopted for the recognition of hand gestures [Ishibuchi et al. 1992]. In this case, only hue and saturation were employed in the detection of skin tone, while the value data was used in a separate background differencing step. Excluding the *value* component from the detection of skin tone is appropriate as this value is most closely related to shading, or light intensity. While the published results are promising, the test scenes were basic in the sense that a monochromatic background was used, with only a few books in the scene. However, Bregler's work was tested on people in *real*

world backgrounds, such as a runner at a track meet with many different colors appearing in the background and was far more encouraging.

4.1.3 YUV

Another widely applied color model is the luminance and chrominance space, also known as YUV. Computationally less expensive than HSV, YUV has gained widespread popularity in the area of skin tone recognition [Wren et al. 1997]. Its color space is Cartesian thus accounting for its ease of calculation:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (2)$$

Like HSV, it extracts the shading or luminance component from the color, leaving pure chromatic components. This model is the basis of the European standard for television broadcasting known as PAL.

Many researchers have turned to YUV over HSV due to its ease of computation, and hence, ability to process at faster frame rates. For instance, Wren et al. employed YUV to identify skin tones for several applications including American Sign Language recognition and vision-driven avatars [Wren et al. 1997]. Their algorithm uses the mean and variance of *a priori* data set of skin tones to identify possible skin tone matches. This approximating technique risks identifying tones that do not occur in the test set, as the simplified estimate of the skin tone region is inexact, compared to a lookup table.

Reducing the variance to cut down on false positives leads to more false negatives. This is a typical tradeoff in such models. It is interesting to note that Wren et al. incorporated a slight mutation of YUV space in order to eliminate pixel values due to shadows. Normalizing the chrominance components by the luminance as in equation 3, they claim that the resulting *true* color, (U*, V*), is independent of illumination.

$$U^* = U / Y \qquad (3)$$

$$V^* = V / Y$$

Ayers and Shah have also been successful in detecting skin tone using YUV [Ayers, and Shah 1998]. Their goal is to identify general actions that take place in a static environment. By tracking the position and velocity of the hands and face, actions such as picking up a phone, or exiting a room are determined. Again, as Wren et al., they utilize the Gaussian variables of mean and variance in order to detect skin tone matches. While the results were promising for static environments, it is difficult to ascertain whether the background included objects such as beige colored corkboards, which may be identified falsely as skin in YUV space. Unfortunately, as with most other color matching papers, the authors fail to describe any experiment with such background distracters. The pitfalls of color matching are addressed further in section 4.3.

Again, the region denoting various skin tones is more refined in YUV compared to RGB, as can be seen in figure 4.4. More importantly, using only UV components provide a reasonable trade-off in data versus the error rate, as will be discussed in section 4.3.

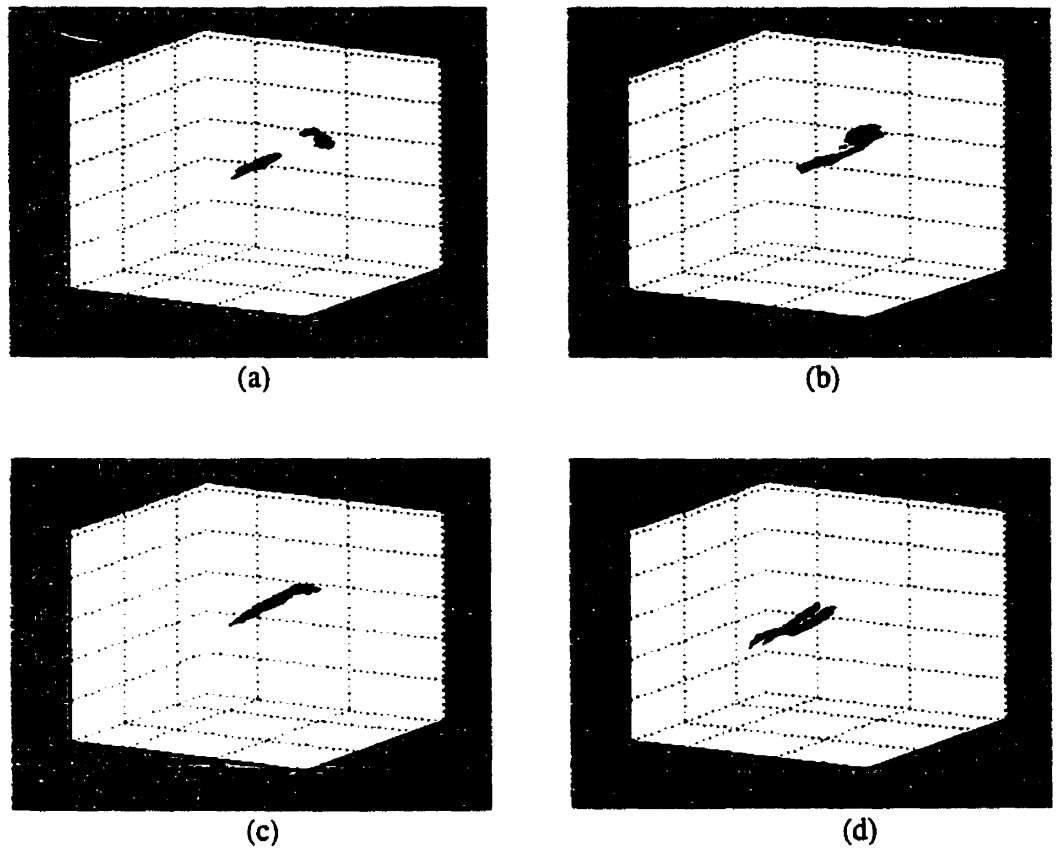


Figure 4.4 – YUV color space of skin tones of people from
(a) Asian, (b) Caucasian, (c) Latin American and (d) African descent

4.1.4 YIQ

Another important color model is YIQ, which exploits particular features of the human visual system in order to minimize bandwidth [Levine, 1985]. Closely related to HSV, the YIQ model has been adopted as the North American standard for television broadcast known as NTSC (National Television System Committee). The *Y* component serves the same definition as the luminance factor in YUV. However the *in-phase or I* component determines the orange-cyan content, while the *Q* or *quadrature* component determines the green-magenta content of the color. This scheme evolved as researchers noted that 64

levels of V in HSV space, with only 32 levels of hue and 8 levels of saturation, are sufficient to denote colors for typical human observers. (See equation 4)

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (4)$$

4.1.5 Normalized RGB

The final color space that we discuss is normalized RGB space. As the name implies, the values of red, green, and blue are converted such that they represent the percentage of total contribution to the color:

$$\begin{aligned} nR &= R / (R + G + B) \\ nG &= G / (R + G + B) \\ nB &= B / (R + G + B) \end{aligned} \qquad (5)$$

where nR, nG, and nB represent the normalized red, green, and blue components respectively. It is important to note that this particular representation is unique in its ability to reconstruct the entire color with the use of only two of its components i.e. nR + nG + nB = 1. This further restricts the region, hence allowing for a more condensed lookup table and facilitating faster processing. The range of skin tone data for various pigments is shown in figure 4.5.

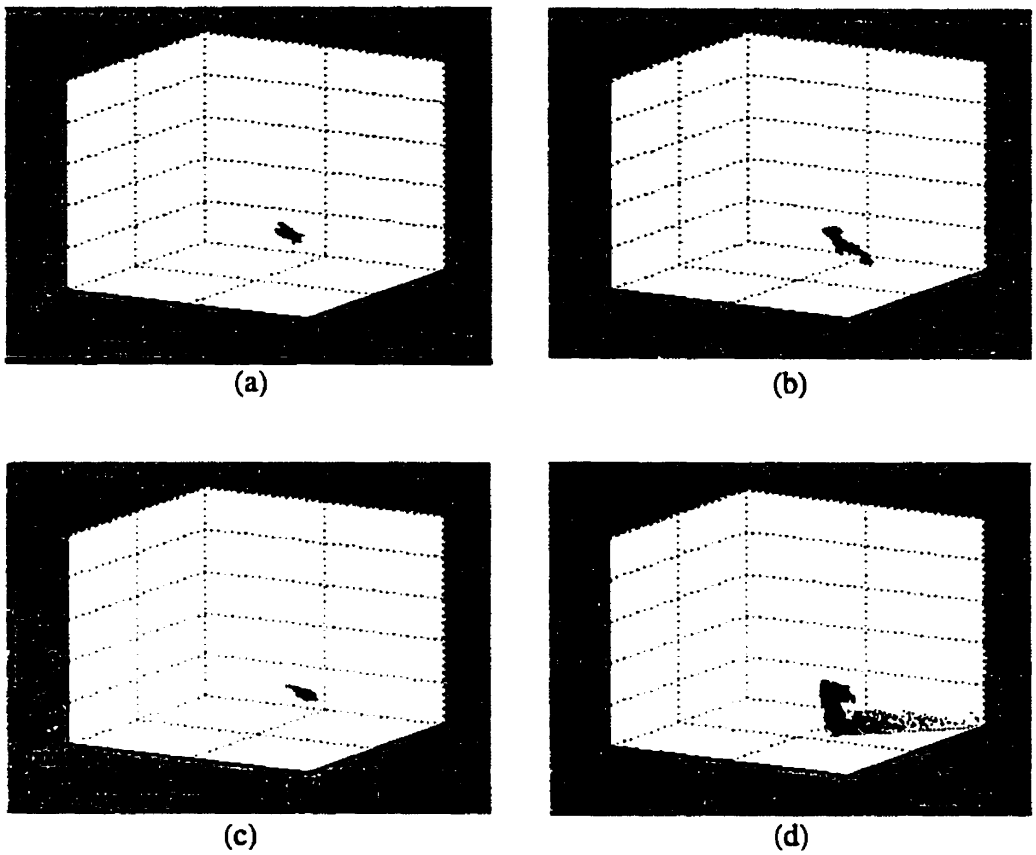


Figure 4.5 – Normalized color space for people of
(a) Asian, (b) Caucasian, (c) Latin American and (d) African descent

This color space has been employed in a real-time face tracking algorithm with notable success [Yang and Waibel, 1995]. The authors claim that any skin tone can be discerned from the background using the normalized RGB color space in concert with motion detection. Using the mean and variance approach, as opposed to a lookup table, they were able to track humans at a frame rate of 30 Hz. Unfortunately, no data is provided to indicate its robustness when it is in the presence of objects of similar color to skin, or under extreme lighting conditions such as shadows or high illumination.

4.2 LUT vs. Gaussians

In implementing the various color space matching schemes, the most important choice is whether to use a lookup table (LUT) or Gaussian variables. As mentioned before, the advantage of a LUT is that it categorizes pixels based on *a priori* knowledge of skin tone rather than an approximation of the enclosed region. Since the appropriate region of most color spaces does not conform to an easily approximated shape, (i.e. figure 4.2), this results in fewer false positives (and negatives). However, due to the LUT's massive size, it does not lend well to real-time application, as the process of comparing each pixel value to every cell of the table is very expensive.

Due to the heavy computational requirements of such a scheme, the most popular alternative is to approximate the skin tone data using a Gaussian function. Comparison based on the mean and variance of this data, can then be performed efficiently. However, this strategy suffers from potential over-simplification of the data space and risks increasing the number of false positives.

4.3 Results

Figures 4.6 and 4.7 display the results of applying six different color spaces to the skin data set and finding the appropriate values in the image based on a lookup table. The table was created from swatches of skin from 25 different individuals of different skin pigmentation, and under a variety of lighting conditions. From these results, two conclusions can be drawn. First, the color segmentation is highly dependent on the lighting conditions and imaging characteristics of the system used to produce the LUT. Second, differentiating skin tone from background objects is by no means robust, at least,

not when applied in isolation. However, color matching does serve as a useful tool in finding *possible* areas of interest in the scene.

In figure 4.6b, RGB color matching successfully identified the general outline of five out of the six people in the scene. However, it mislabeled the leftmost person's pants as skin and entirely missed the face of the individual kneeling. With this color space, densely packed candidate pixels would not result in finding all six people, not even a shape detecting scheme like Birchfield's elliptical head tracker would be sufficient to extract the person's elliptical face with the number of candidate pixels present for the kneeling individual [Birchfield, 1998].

The YUV scheme, illustrated in figure 4.6d did not perform well either. While a few more pixels were correctly labeled for the face of the man kneeling, the various regions denoting pants at both sides of the image introduced far too many false positives to be of use. Furthermore, contrary to the claims of a number of papers, the effects of shadows were not completely eliminated. This can be seen in terms of the differing proportion of correctly identified pixels for faces and hands for the gentleman on the far right.

Isolating the search to match pixel values only of common UV components proved beneficial at correctly identifying more of the skin pixels. At the same time, it also introduced a much higher density of false positives, as the LUT criteria were broadened. Following the UV color matching with an elliptical or circular face detector, such as Birchfield's scheme [Birchfield, 1998], may prove effective. However, extracting the location of hands is likely to be far more difficult. Using knowledge of the head location may be a good starting point as to possible search areas for hands.

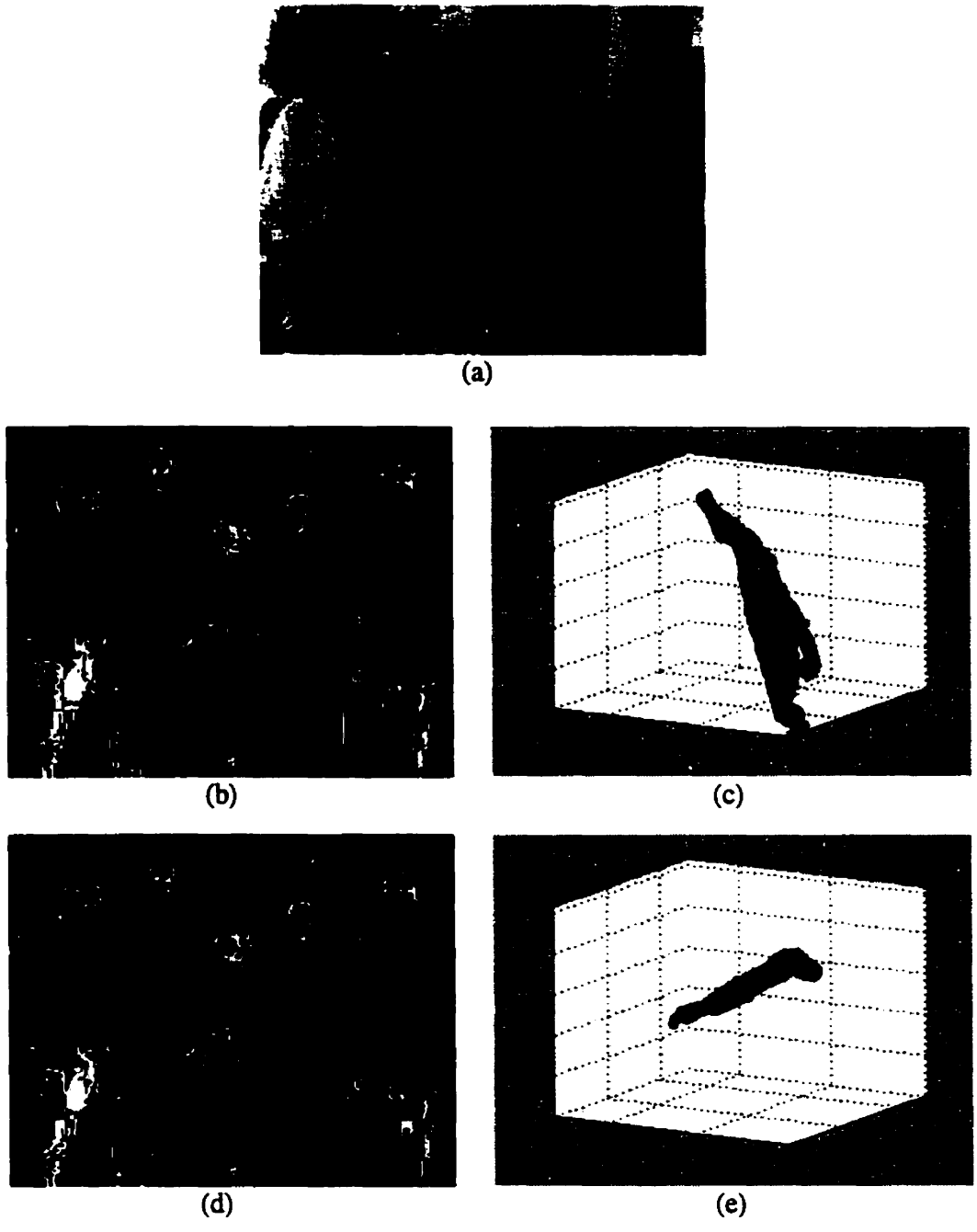


Figure 4.6 – Resulting skin tone detected images

(a) Original image, (Courtesy of: http://www.execpc.com/~lam/startrek.html)

(b) Pixels found in RGB space, (c) Distribution of skin tone in RGB

(d) Pixels found in YUV space, (e) Distribution of skin tone in YUV

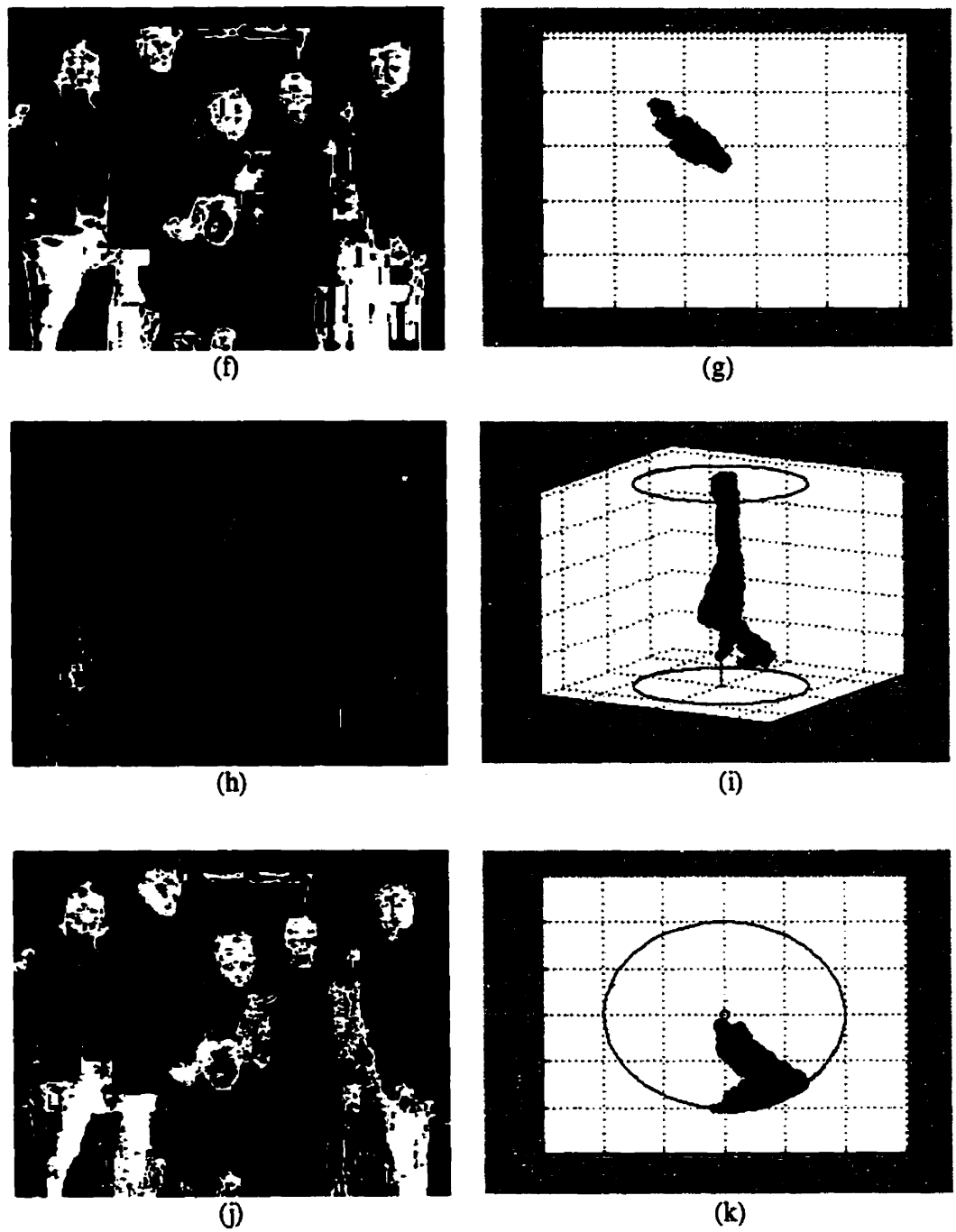
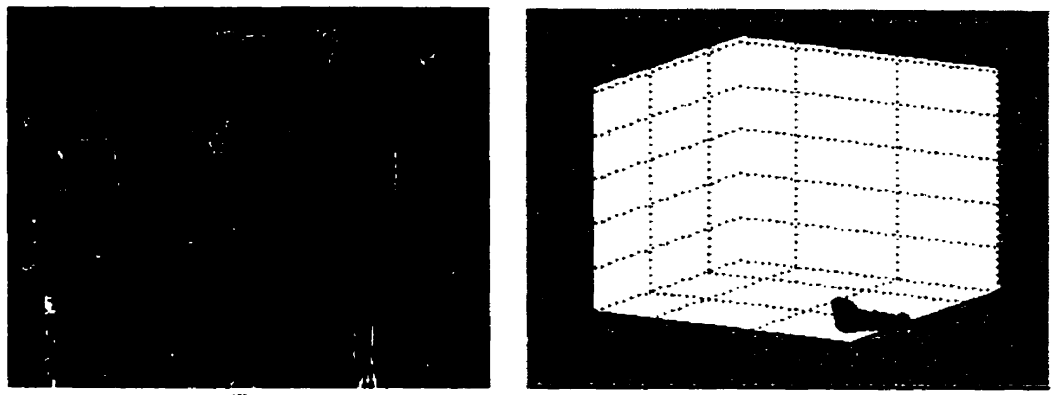


Figure 4. 6 (cont'd) - Resulting skin tone detected images

(f) Pixels found in UV space, (g) Distribution of skin tone in UV

(h) Pixels found in HSV space, (i) Distribution of skin tone in HSV

(j) Pixels found in HS space, (k) Distribution of skin tone in HS



(l) (m)

Figure 4.6 (cont'd) - Resulting skin tone detected images
(l) Pixels found in Normalized RGB space,
(m) Distribution of skin tone in Normalized RGB

HSV color matching, shown in figure 4.6h performed well under various lighting conditions. The five people standing all exhibited a high density of identified pixels for their faces, while the kneeling gentleman had far fewer candidate pixels. Again, the pants of the woman on the extreme left introduced many false positives. However, this may be addressed by later applying an elliptical or circular face detector as mentioned before.

Limiting the search to matches of hue and saturation as shown in figure 4.6j, yielded the best results of all the color spaces tested in this experiment. Nearly all of the face pixels were identified in the image. Even the outline of the kneeling man's face was recognized as skin tone. Incorporating a circle-finding mechanism would certainly lead to six humans in the scene, despite the added noise introduced by some of the individuals' pants.

Finally, normalized RGB color space, shown in figure 4.6*l* was by far the worst at differentiating skin tone from background colors. Normalization of the optimized region

of skin tone may have been responsible for the poor performance. As with the other schemes, the color space may be expanded by adding a few skin swatches to the lookup table. In its defense, it could be said that the normalized RGB space provides a conservative approach to color matching such that any faces found with this method have a higher probability of being valid matches than those resulting from other schemes.

The same LUT's were then compared against another scene to ascertain whether the results were consistent. Improved results were observed overall, but in particular for the UV space, possibly as a result of biased data. As can be seen in figure 4.3d, data collected for the African skin pigment was quite spread out. In contrast, the density of identified pixels for the Caucasian boxer is quite low in comparison.

(a)

Figure 4.7 – Resulting skin tone detected images
(a) Original image, (Courtesy of: http://www.espn.com)

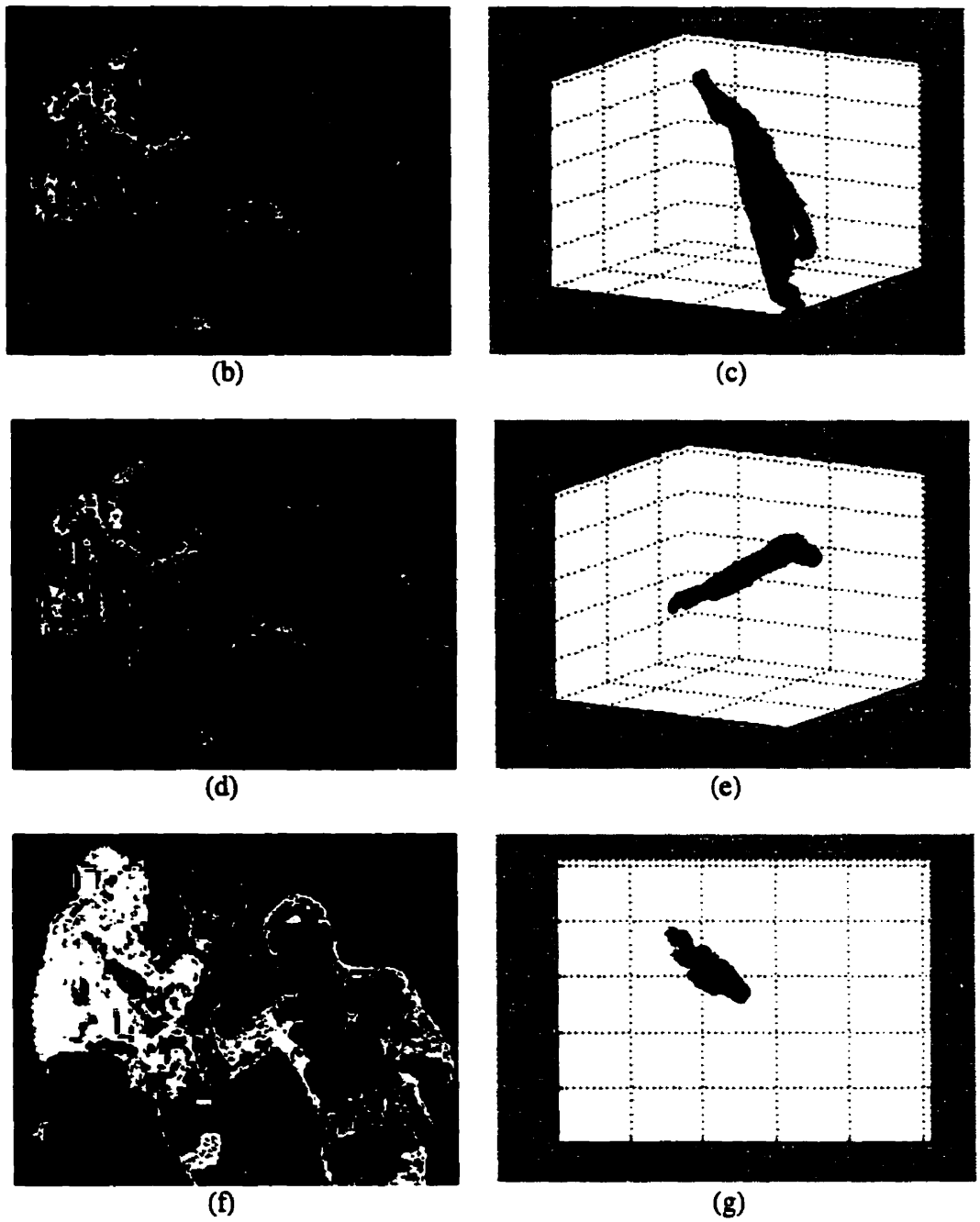


Figure 4.7 (cont'd) - Resulting skin tone detected images

(b) Pixels found in RGB space, (c) Distribution of skin tone in RGB

(d) Pixels found in YUV space, (e) Distribution of skin tone in YUV

(f) Pixels found in UV space, (g) Distribution of skin tone in UV

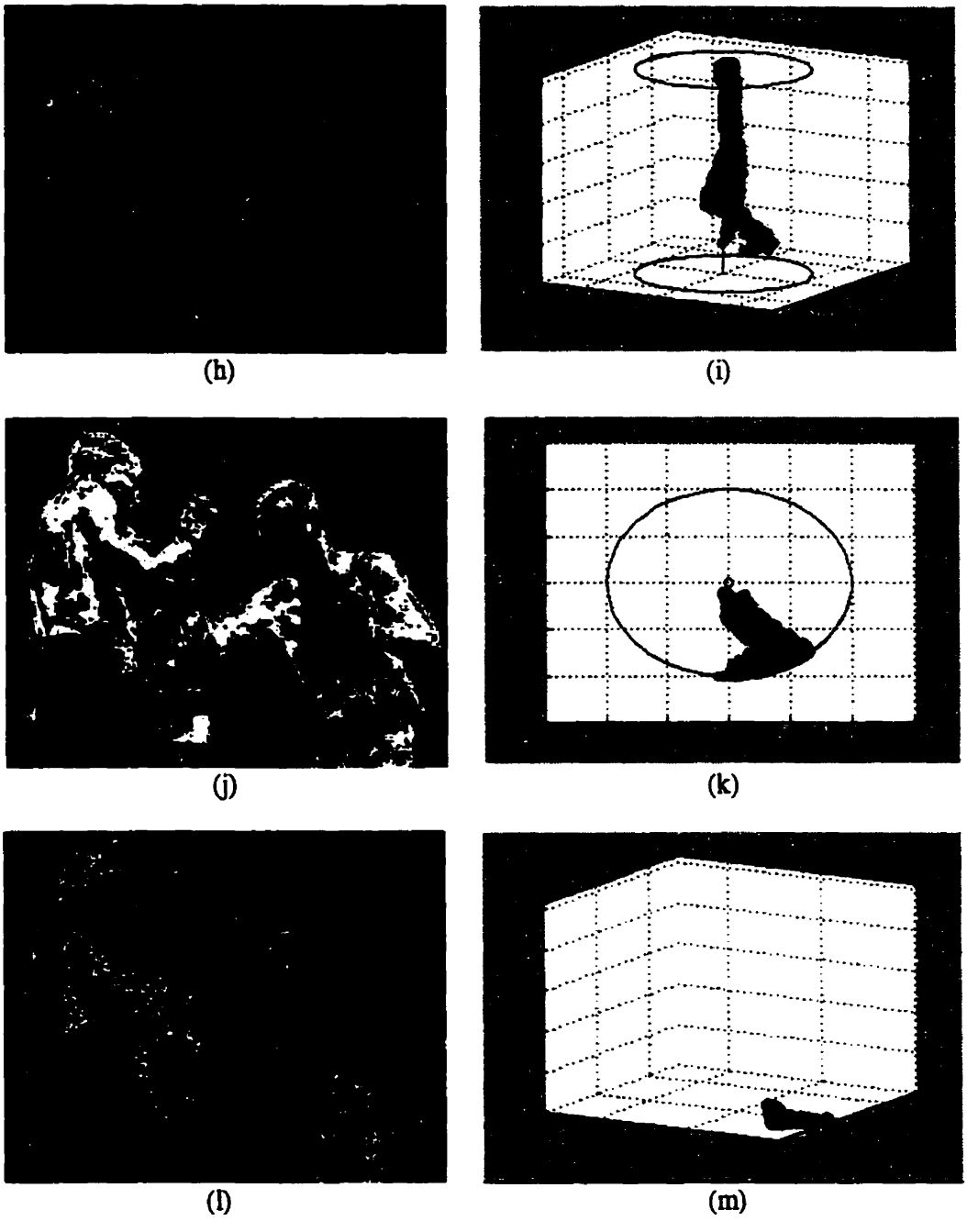


Figure 4.7 (cont'd) - Resulting skin tone detected images

(h) Pixels found in HSV space, (i) Distribution of skin tone in HSV

(j) Pixels found in HS space, (k) Distribution of skin tone in HS

(l) Pixels found in Normalized RGB space,

(m) Distribution of skin tone in Normalized RGB

One possible direction for further research is to examine methods of optimizing the data selected for the LUT. In order to obtain the most appropriate data, a common method of reducing the size of the table is to ignore data for skin tones that have a saturated value for any of the red, green or blue components. These colors are normally due to very bright lighting, and while useful in identifying skin under bright lighting, the same values can result from bright lighting on **any** surface. Another possible method of reducing the LUT would be to eliminate outliers or values that are sufficiently separated from clusters of the data. This could be done through a number of pattern recognition techniques such as simple partitioning using a minimal spanning tree [Zhan, 1971] or a variant of the nearest neighbor technique with a threshold. Once the data has been properly optimized, the choice of technique depends heavily on the application. If an unmodified lookup table is employed the processing demands will reduce the achievable frame rate but no approximation is required, thus the identified pixels are certain to have a skin tone from the data set. On the other hand, simplifying the data allows much higher frame rates. Simplifications can be made by noting the Gaussian distribution variables or by approximating the contour of the data by a simple geometric shape.

4.4 Conclusions

After testing color matching schemes in six different color spaces, several outcomes emerged. First, it was found that there do exist certain color spaces that yield satisfactory results in finding pixel values related to human skin tone. Most importantly, though, these results indicate that color matching schemes, alone, are insufficient as a person

locating algorithm due to the high density of false positives and the potential for false negatives.

Now that we have considered various strategies for localizing an individual within a scene, we now turn to the task of extracting information relevant for action recognition. The next chapter considers one such strategy based on contouring methods.

Chapter 5

Contour Extraction

5.1 Contours

After an image has been segmented sufficiently such that the user's whereabouts are known, information must then be extracted to facilitate action recognition. If one examines how humans interpret actions, some key features come to mind, notably, joint angles, the velocity and acceleration of movements, and the context of the environment. For instance, consider the actions of the conductor of a symphony orchestra. Observing the *position* or *pose* of the hands and the *velocity* and *acceleration* of their movement determine such factors as tempo and volume. Furthermore, the *context*, in this case a concert hall, allows one to, for example, discern that the conductor is trying to convey musical instructions as opposed to flagging down a taxi cab! While this latter factor plays a role in action recognition, it is beyond the scope of this chapter. Instead, our focus will rest in locating key points, as well as their relative characteristics, in pursuit of an action recognition algorithm.

This chapter discusses various computer vision tools to extract the contour of a given object, as well as possible methods of utilizing this information to develop an *action feature vector*. First, a general review of object properties useful in obtaining feature vector elements is offered, followed by a discussion of boundary following techniques. Next, region segmentation is considered for multiple objects, and skeletonization methods and active contours are presented. Finally, some preliminary

conclusions are drawn concerning the utility of these techniques to the field of action recognition.

5.2 Action Feature Vector Elements

In the area of object recognition, three key variables are normally considered: size, position, (denoted by the center of the object), and orientation. Together, these features determine whether or not the object matches a preconceived model or template. We begin this section with a brief overview of these features.

5.2.1 Size

Since the realm of image processing lies in the discrete domain, many of the calculations are simplified versions of their continuous domain counterparts. One such example is the calculation of size, or *area* of an object. Given a binary image[1], the area may be found as follows:

$$\text{Area} = \sum_{i=1}^{N} \sum_{j=1}^{M} B(i, j) \qquad (1)$$

where the region or image is of size N x M, and B(i, j) denotes the pixel value at location (i, j) in the binary image. This calculation, which also refers to the zeroth-order moment, is commonly used in situations where the camera remains at a fixed distance from the object in question, for example on an assembly line conveyer belt.

[1] The term *binary image* refers to an image in which a pixel value of 1 denotes a potential user pixel and 0 denotes the backgrcund

5.2.2 Position

One of the more important feature elements in action recognition is position. The center location of various regions, such as those denoting hands is vital in determining actions. In image processing, *position* is often used to denote the center of mass of an object. This calculation, also known as the first-order moment, is easily determined for discrete space:

$$\underline{X} = \sum_{i=1}^{N} \sum_{j=1}^{M} (i)\,(B\{i, j\}) \qquad (2a)$$

$$\underline{Y} = \sum_{i=1}^{N} \sum_{j=1}^{M} (j)\,(B\{i, j\}) \qquad (2b)$$

Given that we are dealing with a binary image, the object is assumed to have uniform mass throughout, i.e. each pixel has an equal weighting of 1.

5.2.3 Orientation

Another important feature element to action recognition is orientation. This element serves as the basis of angular velocity and acceleration, which are often fundamental to discerning an action. For example, if someone is pointing to an object, the orientation is vital to determine which object the user wishes to identify. Without the orientation value, no distinction could be made as to whether the user is pointing at the book on the top, or the bottom shelf. The most commonly used tool to obtain this information is the second-order moment, which determines the orientation along the elongated center line of the object (See figure 5.1).

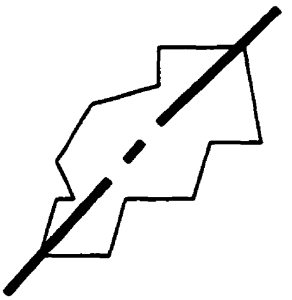


Figure 5.1 - The dashed line denotes orientation along the most elongated region of the object, forming the second-order moment.

This helps to interpret actions as these normally occur along the skeletal equivalent of the user. For instance, pointing direction can be established directly by noting the hand-to-elbow vector, typically the most elongated portion of the arm.

The equation for the second-order moment is calculated by minimizing the sum of the squared distances between object points and the centerline. The centerline that provides the least second moment is the line that denotes the orientation of the object. Firstly, second-order moment is calculated as noted in equation 3:

$$Z^2 = \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} B(i, j) \qquad (3)$$

where d_{ij} denotes the distance of object pixel at (i, j) to the axis. Using polar coordinates to account for axes that exhibit a slope of 90°, the problem now becomes a minimization equation to find the minimum value of Z^2. (For full details, see [Jain et al, 1995])

5.3 Regions

Once a reasonable binary image is created identifying the possible location of the user, it is often useful to segment and label the connected regions. Grouping connected regions allows for the efficient elimination of those clusters smaller in area than a pre-determined threshold (See section 3.2). In general, regions may be labeled recursively or sequentially. The recursive algorithm follows the steps outlined below:

Table 1: Recursive Region Labeling Method

Recursive Labeling Algorithm	
1	START position, (top-left of image) and perform raster style search
2	Find an unlabeled pixel, if none then stop
3	If any neighbors have no labels, label them and goto (1)

While this algorithm is easy to visualize, its recursive nature makes it computationally expensive. Fortunately, the same result can be achieved using the following, more efficient, sequential algorithm:

Table 2: Sequential Region Labeling Method [Jain et al. 1995]

Sequential Labeling Algorithm	
1	Scan image in raster format, (top left, to bottom right)
2	If pixel is value '1' (denoting potential user pixel, not label)
2a	If the upper or left pixels are labeled, copy either label
2b	If both upper and left have same label, copy label
2c	If upper and left have different labels copy upper label, and add both labels to equivalence table
2d	Otherwise, assign new label
3	Goto 2 until all pixels have been labeled
4	Sweep entire image again re-labeling as per the equivalence table

This method requires only two sweeps through the image, whereas the cost of the recursive method is dependent upon the number of regions. It should also be noted that the sequential labeling algorithm can be easily modified to label 8-connected regions, simply by noting the northwest pixel in addition to the northern and western neighbors.

5.4 Boundary Algorithms

Once the regions in the binary image are properly segmented, their boundaries can be found. As a general definition, a *path* exists between (x_1,y_1) and (x_N, y_N) if, for all k, $1 \leq k \leq (N\text{-}1)$, (x_k, y_k) and (x_{k+1}, y_{k+1}) are 4 or 8-connected neighbors. A *boundary* can then be defined as the maximum length closed path consisting entirely of pixels with at least one background pixel as a neighbor. From this definition, it is clear that only the dashed line of Figure 5.2, constitutes a boundary of the enclosed region.

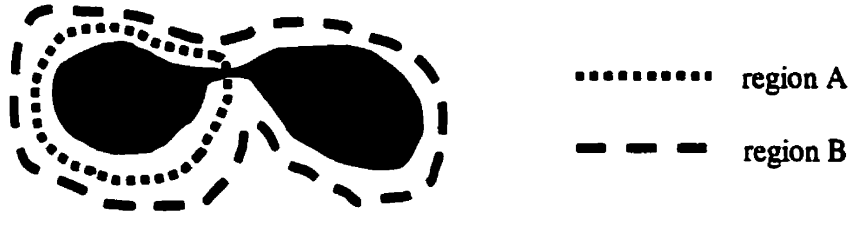


Figure 5.2 – A valid (dashed) and invalid (dotted) boundary of the shaded region.

5.4.1 Boundary Following Algorithm

One of the simplest methods for extracting boundary information is the Boundary Following Algorithm [Rosenfeld and Melter, 1989]. Also known as *Moore Tracing*, this technique first identifies a pixel that is part of the object and has at least one background neighbor, denoted as *P1*. Following a clockwise direction, each of this pixel's eight

neighbors is checked to see whether this condition also applies. If so, the corresponding pixel is labeled successively as *P2*, *P3*, etc., until the following two conditions are met:

1. *P1* is found again; and
2. The next pixel found after *P1* is *P2*.

These conditions ensure that the algorithm will continue until a maximum length path is identified, thus preventing an erroneous identification of path A as the boundary of the region in Figure 5.2. The final outcome is a connected boundary in the form of a list of pixel location. In terms of its effectiveness, this algorithm works well for binary images with densely connected regions. However, for highly fragmented regions, other techniques such as dilation must be used to obtain a reasonable contour (see Section 2.5).

5.4.2 Dilate and Compare

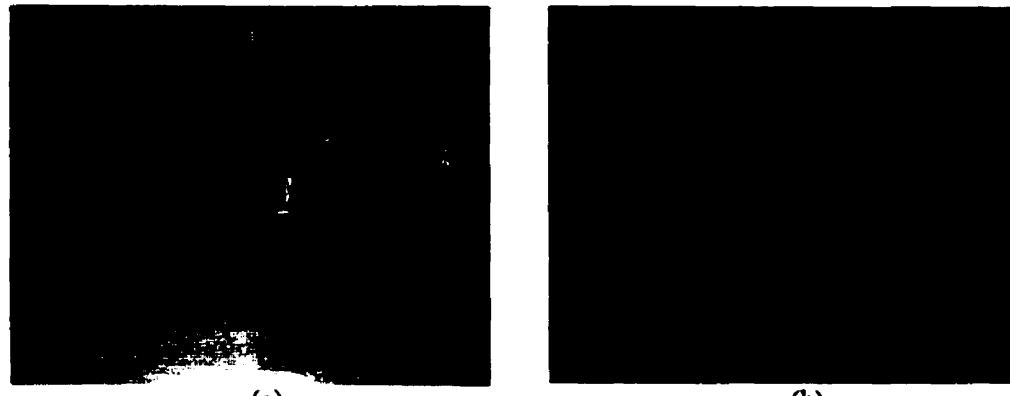
One method that is quite effective in dealing with highly fragmented edges is the dilate and compare algorithm [Yang and Levine, 1992]. The first step, as the name implies, is to perform a number of dilations on a copy of the binary image. Assuming that only one region is found, i.e. corresponding to a person in the scene, the outermost boundary layer of the dilated image is then labeled *B1*. This boundary is then removed and the second outermost boundary is labeled *B2*. The process continues until there are no more boundaries to label. The second step calculates the number of candidate pixels in the original binary image that match locations in each of the boundaries $\{B_1,..,B_N\}$. The layer with the maximum number of matches is chosen as the boundary for the binary image.

It is interesting to note that because of dilation, this method works quite well for several independent regions of edges throughout the image as they *fill* the broken regions. Since the pixels are expanded from known edges, the *middle*, or median boundary is usually the most likely candidate. This algorithm also works well as a global technique as opposed to a neighborhood or local technique as a dilation may be performed for the entire image in one step which significantly reduces the computational requirements compared to the local approaches.

The drawback of dilation and compare is that the first step may join regions that are truly non-contiguous, thus adversely affecting the final result. The advantages, however, are that it eliminates small fragments in the edges, and offers an inexpensive computational approach to locating the boundary.

5.4.3 Pixel Sweep

Another method to extract contour information from a binary image is to perform a *pixel sweep* [Arseneau and Cooperstock, 1999a]. Essentially, a row of pixels *sweeps* from one side of the image to the other until it contacts a candidate pixel. The result is analogous to dragging a flexible snake from one side to the other, however no energy values are optimized. This technique, illustrated in Figure 5.3, is computationally inexpensive and provides a reasonably accurate assessment of the most likely contour. To reduce noise-related errors, an *impulse detector* can be implemented. This detector rates the likelihood of each pixel belonging to the user by noting the slope of the preceding and successive N candidate pixels. If its value fails to meet a pre-determined threshold, it is recalculated as the average value of its immediate neighbors (i.e. interpolated).



(a) (b)

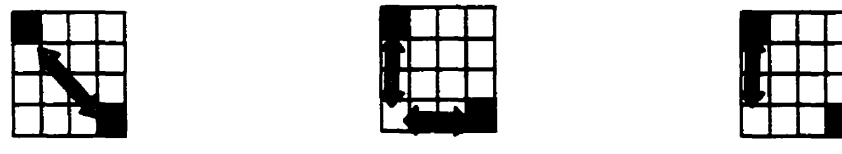
Figure 5.3 – (a) original image, (b) pixel sweep denoted in white

5.5 Skeletonization

Once the boundary information is obtained, it is sometimes useful to extract the basic or skeletal form of the object. Skeletonization shrinks the boundary shape until it forms a series of curves and lines that are of unity width. The resulting *skeleton* easily allows one to identifies key elements relevant to action recognition such as joint angles. Before the various strategies are examined, it is important to define the distance metric.

5.5.1 Distance Metric

The three most common methods of determining the distance between two pixels are the Euclidean metric, Minkowski metric, and the chessboard metric, as illustrated in figure 5.4.



(a) (b) (c)

Figure 5.4 - (a) Euclidean metric, (b) Minkowski metric, and (c) chessboard metric

The Euclidean metric provides the most accurate distance between two pixels, but unlike the Minkowski and chessboard metrics, does not guarantee an integer valued result. Of course, the result can be rounded up or down as needed. The Minkowski, or city-block metric, is well suited to finding boundary skeletons while the chessboard metric, which measures only in a single dimension has a much smaller application base. In the following sections, the Minkowski metric shall be used to demonstrate the medial axis transform.

5.5.2 Medial Axis

The medial axis algorithm re-labels object pixels according to their distance to the nearest background pixel.

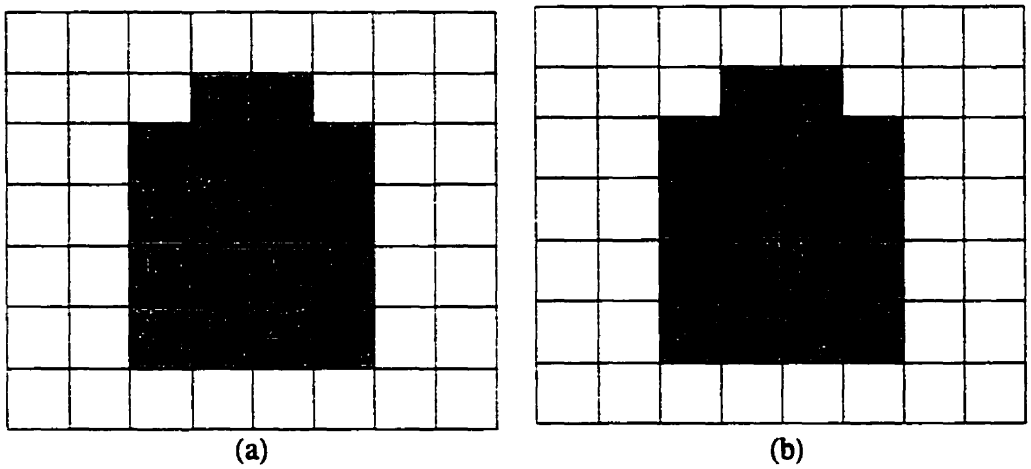


Figure 5.5 – Medial Axis, numbers denote distance from the object pixel (gray), to nearest background pixel (white cell). (a) All object pixels with accompanying distance to background, (b) Medial axis. *The Minkowski metric was used in the example

Next, pixels are eliminated if this distance is less than that of their 4-connected neighbors. The results of the algorithm are illustrated in figure 5.6.

(a) (b)

Figure 5.6 - (a) Medial Axis Transformation, (b) Effect of noise on the Medial Axis

The resulting skeletonization of a noise-free boundary forms a good approximation, however the presence of noise can have a significant impact, as illustrated in Figure 5.6.

5.5.3 Thinning

A more robust approach to skeletonization, less susceptible to noise, is to *thin* the object. This involves stripping successive layers from the object until it has reduced to a line or curve. In order to maintain end points, a series of checks are performed to ensure that the appropriate skeletal form is maintained. (See table 3)

Table 3: Conditions for Thinning [Jain et al. 1995]

1	Connected regions must thin to connected structures
2	The final connected structure should be minimally 8-connected
3	End points should be maintained
4	The final connected structure should approximate the Medial Axis
5	Short branches caused by thinning should be eliminated

This method provides a good approximation of the skeletal structure of a given object, although the threshold for eliminating short branches (criterion 5) must be chosen carefully so as not to eliminate important branches.

5.6 Simplifying Data Set Information

Once a reasonable boundary or skeletonization is attained, the next step toward action recognition is to transform this discrete set of points into lines, quadratic, or cubic curves that approximate their general shape. [Bharatkumar et al. 1994; Birchfield, 1998] Beyond simplifying later processing, this serves the added benefit of reducing memory requirements. Though the various algorithms available for such transformations, surveyed in the following pages, are quite effective, they are only as accurate as the information passed to them. The presence of noise can have a significant impact on the curves chosen to represent the contour list.

5.6.1 Chain Codes

Chain codes describe a shape by noting the direction of coinciding pixels over its boundary. This provides a quick method of comparing object shapes that may differ only in scale, translation or rotation. Unlike other techniques, chain codes do not *approximate* the boundary, hence errors introduced due to improper approximation are eliminated. However, no data reduction results.

The first step is to label the neighborhood of a boundary pixel as per figure 5.7:

2	3	4
1	X	5
8	7	6

Figure 5.7 - Slope representation for chain codes

The contour boundary is followed in a clockwise direction and each binary pixel is replaced by the appropriate label (1 to 8) corresponding to the slope of its neighbor. The resulting chain code is a general *shape* description, of equal length to the original boundary list.

Since chain codes do not reduce the data set, the contour of complex shapes may prove less efficient that those created from an approximation technique. On the other hand, by not approximating the data set, less error is introduced as to the exact shape of the object in question. Therefore, the final decision will rely heavily on the application needs.

5.6.2 Curve Fitting

Polyline approximation, or *curve fitting*, is widely used in computer vision due to its inherit simplification of data. [Birchfield, 1998; Benjamin, 1990; Lipardi et al., 1989] To simplify the following discussion, we will consider the boundary to be reduced to an open path to facilitate simpler curves.

This approach tests different curves and notes how well each approximates the general path of existing pixels. This is reminiscent of the *dilate and compare* method, discussed in section 5.4.2, in which the number of matching locations with the boundary

is compared to the original binary image. However, with polyline approximation, the shortest distance of each of the N boundary pixels to the curve is noted, and the curve is updated until the *maximum absolute error* is minimized. (See equation 4)

$$E_{max} = \max(d_i) \qquad \text{for } (1 \le i \le N) \tag{4}$$

where the distance d_i refers to the minimum distance from the point to the curve. For instance, approximating only with lines, equations 5 and 6 describe the calculation of distance of each point, (u, v) from the line with endpoints, (x_1, y_1) and (x_2, y_2).

$$r = u(y_1 - y_2) + v(x_2 - x_1) + y_2x_1 - y_1x_2 \tag{5}$$

$$d = r / ((x_2 - x_1)^2 + (y_1 - y_2)^2)^{1/2} \tag{6}$$

where equation 6 normalizes the absolute distance, r, by the length of the line. If there are no sign changes over the set of r values, this indicates that the chosen curve should be translated closer to the data set. Jain et al. propose that the number of sign changes of r should be used to choose the next curve to fit, for example, selecting a quadratic if there are two sign changes or a cubic for three sign changes [Jain et al., 1995]. If there are no sign changes, it is a good indication that the curve chosen should be translated closer to the data set.

While there are many situations for which this heuristic fails, it appears to be a reasonable approach in determining the complexity of the curve needed to approximate the data set.

5.6.3 Recursive Subdivision and Polygonal Approximation

If either a connected sequence of lines or a polygon can reasonably approximate the data then recursive subdivision or polygonal approximation prove useful. [Zhu and Poh, 1988; Laumond et al., 1994] These techniques follow the same general approach outlined in Table 3 with polygonal approximation adding the constraint that the polyline must form a closed loop.

Table 4: Recursive Subdivision

1	Construct a line joining the end points of the data setand add these to the vertex list
2	Calculate the maximum absolute error (equation 4) with respect to the vertex list
3	While the error is above a given threshold, add the point corresponding to this error to the vertex list, and goto 2
4	Otherwise, stop

A restriction on the number of recursive subdivision may be desirable in order constrain the polyline to a maximum number of segments. For example, five segments would seem a reasonable breakdown of a human's arms, in order to discern the two forearms, upper arms and shoulder region. The angles formed at the vertices could then be used to interpret the action being performed.

5.6.4 Hop Along Recursive Subdivision

A slight refinement of this technique, known as hop-along recursive subdivision obtains the same results with less computation [Jain et al., 1995]. The first-step of the algorithm remains unchanged. However, after the first subdivision, the algorithm considers only

those points between the first two vertices in order to determine the maximum error and select a new point for the next subdivision. This continues recursively, as before, until the maximum error falls below the chosen threshold. The process is then repeated on the second half. This reduces the calculations needed for each subdivision by concerning itself only with data points close to the line segment currently being examined. While the authors do not explain **how** to determine these points without explicitly performing the distance calculations, a nearest neighbor type of algorithm might be used to group the data set.

5.6.5 Hough Transform

Another commonly used method of finding lines in a binary image is the *Hough Transform* [Chan and Sandler, 1992], which maps the (x, y) pixel coordinates to the slope and y-intercept in another domain, known as the *Hough space*. The intersections of the resulting lines are used in a voting scheme to estimate the best approximation. If multiple lines occur in the initial binary image, a number of clustered intersections typically appear in the Hough space. Various statistical pattern recognition schemes such as a *K*-means approach, or clustering techniques based on Gestalt clusters [Zhan, 1971], could be used to determine the exact number of lines, while minimizing noise. Although the Hough transform works reasonably well for images with multiple straight lines, the presence of curves may lead to an overly complicated representation, making it difficult to select appropriate thresholds and intersections.

5.6.6 Active Contours

Another class of contour extraction methods is the physics-based approach of *snakes*. These use a model where a pre-defined deformable line is attracted towards regions of high energy, while escaping areas of low energy. The effect is a-kin to a snake wrapping its body in such a manner as to overlay the most pixels, while avoiding areas denoting background [Blake and Isard, 1998]. With this approach, the initial placement of the snake is vital to its success. Snakes are also computationally expensive, and at the present, are not a feasible option for real-time action recognition.

5.7 Summary

This chapter has surveyed various tools that perform operations of importance to action recognition. Previous chapters investigated filters, background removal techniques, and color matching. This chapter examined a number of contour extraction techniques, which provide key information concerning object shapes. Note that this review is by no means complete, as there exist many other feature elements such as symmetry [Reisfeld et al., 1995] and other matching techniques including templates and correlation methods [Sawasaki et al., 1996] that were not explicitly mentioned but may warrant consideration.

Now that a review of the image processing tools is complete, the task now becomes one of choosing the appropriate algorithms to effectively extract key action feature elements from a video sequence to facilitate an action recognition scheme.

Chapter 6

Action Recognition Segmentation

6.1 Steps Towards Action Recognition

Now that the various tools have been described and investigated, we can begin the task of assembling them into a framework to facilitate action recognition. The general outline proposed consists of three steps:

1. Segment the user from the scene;
2. enhance the binary image; and
3. locate points of interest (features) and quantify these over a period of time.

A fourth step, outside the scope of this thesis, would then use this information to classify or *recognize* specific actions. As a starting point, consider the action that represents *lowering the volume*. This action begins with one hand extended high in the air, followed by lowering the hand along an imaginary, vertical line, until the hand is pointed downwards (See figure 6.1).

This particular action exhibits features in common with most everyday human gestures. There is a starting pose, indicating the beginning of a gesture, followed by a movement, which *imitates*, to some degree, the idea being expressed. In this case, the lowering of the arm is reflective of the lowering of volume. Finally, there is an ending pose denoting the completion of the action.

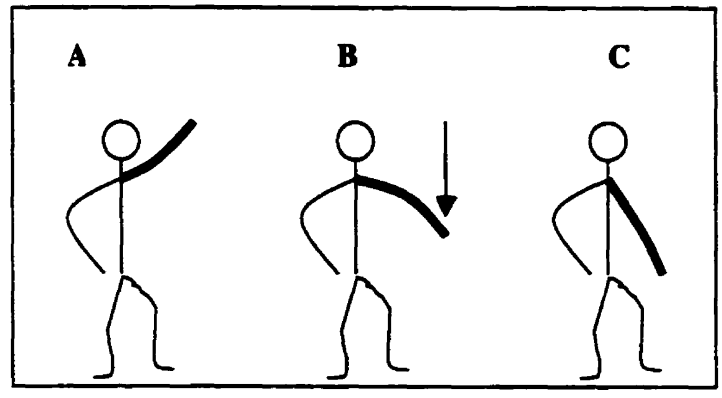


Figure 6.1 – Lowering volume action – (a) starting pose, (b) movement period, (c) ending pose

In order to interpret the action being performed, there are several key elements at work. First, the person performing the action (the *actor*) must be located within the observer's field of view. Next, key points on the actor, such as elbows, hands and feet, called *feature elements*, must also be located. The specific set of points relevant to the action in any particular frame is known as the *feature vector*. As actions span a period of time, the collection of feature vector will form the action's *feature set*. In addition to identification of the feature elements, it is important to note the relationship between these points. For our previous hand-lowering example, the downward movement of the right hand should be interpreted differently if the upper body is simultaneously lowered, which may indicate another type of action.

An important issue, but one outside the scope of this work, is the problem of recognizing which frames of a video sequence contain the start and end poses. We are primarily concerned with image segmentation to retrieve the salient features of the person, not the translation of these features into actions. However, some preliminary efforts to tackle this problem using probabilistic models [Bregler and Omojundro, 1995],

as well as an examination of poses over time [Emering et al., 1996], are worth noting. Among the more intuitive approaches, temporal templates, statistical data formed from the motion within a scene, have been implemented to recognize actions over a fixed period of time [Davis and Bobick, 1997].

Returning to the task of segmentation for action recognition, two strategies have evolved. The first follows the approach adopted by Arseneau and Cooperstock [Arseneau and Cooperstock, 1999a], based on a variation of the background primal sketch, discussed in Chapter 3 [Yang and Levine, 1992]. The second strategy, not yet implemented, is based on motion and a virtual skeleton.

6.2 Background Removal Scheme

As this approach is based on the background primal sketch, restrictions of a fixed camera as well as a relatively static background should be observed for best performance.

To avoid noise due to lighting changes in the scene, the incoming images (Figure 6.2a) are pre-processed by a Sobel edge-detector (Figure 6.2b). This step will help two fold by reducing the number of candidate pixels, while providing the appropriate pixel locations denoting contours. These edge-detected images are then used to construct a *background* primal sketch (Figure 6.2c), containing only high frequency edges. Note that this reduction in the number of candidate pixels restricts the applicability of later image processing operations. For example, it would be very difficult to determine which pixels belong to the largest connected region, or calculate the exact center of mass, without knowing what is *inside* the contour.

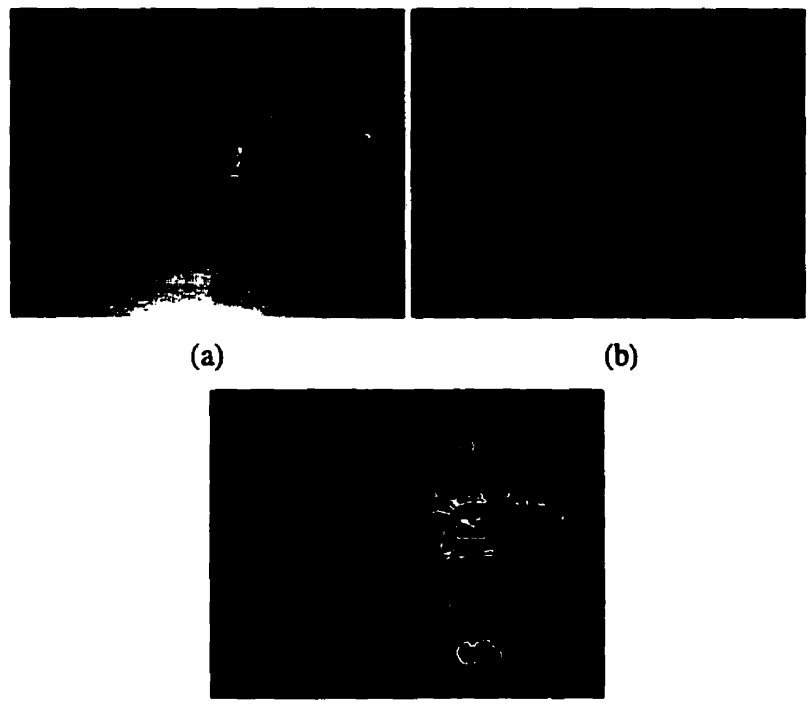


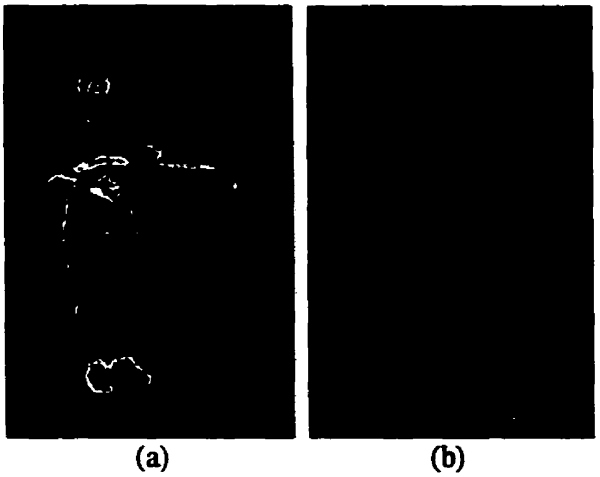
Figure 6.2 – (a) Original image, (b) Sobel edge-detection,
(c) Difference image created from edge-detected Background Primal Sketch

Once the user has been segmented from the background, the difference image is processed to reduce noise pixels while retaining the maximum number of true candidate pixels. As observed in Chapter 3, removing isolated pixels yields better results on edge-detected images than does erosion, due to the *thinning* nature of edge-detection.Also, pixel locations that denote a low gradient value can be eliminated using Otsu's adaptive thresholding technique [Otsu, 1979]. The resulting binary image, illustrated in Figure 6.3, satisfies our requirements of eliminating many of the false positives while retaining most of the true user pixels.

(a) (b)

Figure 6.3 – Noise removal, (a) Original difference image, (b) post-noise removal

The final and most difficult step is to transform pixel locations into useful feature elements for action recognition. Following edge detection, the logical approach is to extract the user's contour, which we perform by the pixel sweep technique discussed in Chapter 5. The resulting contour information, shown in Figure 6.4, is sufficient to interpret the action of *lowering the volume*. However, a more complex scheme may be required for actions involving a more complex set of feature elements.



(a) (b)

Figure 6.4 – Contour tracing, (a) cropped version of difference image, (b) pixel sweep from top to bottom. *Note that these images have been cropped manually so that the pixel sweep is clearly visible.

Using the pixel sweep contour, pertinent feature information including points such as the angles formed at the elbows, can be obtained through polygonal approximation. Tracking endpoints such as the hands and feet could also be accomplished by first noting their location during a *robust* pose, for example, with the user's arms, legs, and body forming an "X" shape. Keeping a record of the position and velocity of these features over time is important, in order to cope with tracking problems, for instance, when the user's hand is swept in front of the body. In such a case, the contour-tracing algorithm would lose one of the user's arms. This problem helped inspire the second segmentation algorithm, described in the following section.

The background removal segmentation scheme described here proves efficient at locating the user, segmenting, and extracting feature information to be used for action recognition. However, the restriction of a static background is a severe drawback that precludes a general purpose application. As discussed in Chapter 3, there are various methods for reducing ghosting effects due to the movement of small objects. However, if the scene change is dramatic, this algorithm will breakdown. Ideally, we seek an algorithm that needs minimal *a priori* knowledge of the scene and imposes no restrictions on the background. This leads to the second algorithm, based on skeletal attraction.

6.3 Skeletal Attraction

To avoid the need for background construction and remove the restriction of a stationary camera, we begin with a temporal rather than spatial approach to segmentation. Thus, subtracting image (t) from image $(t+1)$ produce the binary image, as shown in Figure 6.5.

(a) (b)

Figure 6.5 Temporal Differencing, (a) Original image,
(b) Difference image resulting from small movement of the user.
The horizontal and vertical lines represent the maxima of the horizontal and vertical histograms, respectively.

The candidate pixels in the difference image now denote a change in the scene over a small period of time, as opposed to a difference from the background. Since all actions involve a sequence of different poses over time, this approach seems well suited for action recognition. Next, the binary image is enhanced by an erosion to eliminate stray pixels. At that point, we are ready to isolate key features.

This step requires the use of a *virtual skeleton*, possibly initialized by the user standing in a pre-determined pose (i.e. with the hands and feet extended in the shape of the letter "X"). Once initialized, the virtual skeleton would be maintained by deforming the joints in accordance with the user's movements (see Figure 6.6). This could be accomplished in a manner akin to *snakes*, in which the joints are attracted to areas of high energy, corresponding to motion in the binary image.

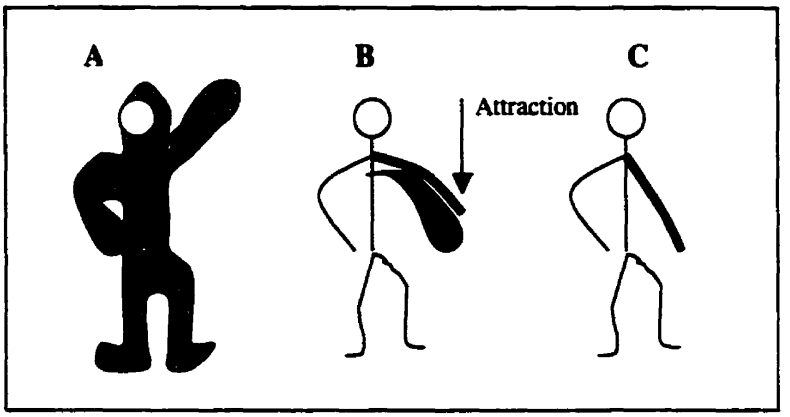


Figure 6.6 – Skeletal Attraction (a) Gray region denotes initial user position, (b) Gray region denotes motion, (c) no motion

In this case, feature elements would be much easier to quantify, as the shape and proportions of the skeleton are pre-defined. This method would also assist in determining the location of feature elements when the user's hand is swept in front of the body. While the earlier background removal algorithm would fail in this situation, the skeletal attraction scheme would create a potential motion region in front of the user's body, thus allowing the virtual skeleton to follow the hand's position.

In order to implement this algorithm successfully, two important challenges must be addressed. First, some mechanism is needed to differentiate motion due to the user from that due to background changes. Certainly, simple segmentation techniques offer a reasonable approximation, but more accurate methods are called for. Second, a scheme is required for controlling the movement about each of the joints, while restricting their movement to physically realizable poses. It is believed that progress can be made fairly quickly and foresee this algorithm proving itself to be of significant relevance and applicability due to its simplicity and lack of restrictions on the background.

6.4 Conclusions

In reviewing both existing and new image processing tools for segmentation, a few key points have been discovered. In order to construct a reasonable action feature vector, specific parts of the body must be tracked, both spatially and temporally. Using a combination of filters and contouring techniques appears logical to aid in the location of the user. However, the choice of algorithms must be chosen with utmost care. If vision algorithms are to be applied successfully outside of the laboratory environment, they must not be encumbered by unrealistic restrictions. Furthermore, the algorithms must be sufficiently robust enough so as to cope with a wide variety of scenarios without manual tuning.

The background removal scheme proposed provides a high degree of robustness. The feature vectors are quite easily calculated from the resulting binary image, which is the prime goal of the algorithm. However, the restrictions that accompany such an algorithm may be too detrimental for use outside the laboratory environment. Requiring both a stationary camera and more importantly a static background, this approach is significantly limited. Also, adding the constraint on the user as to the color of clothing they may wear further hinders this approach.

The second algorithm proposed seems to hold promise, providing both robustness in dynamic environments, as well as accurately identifying key features of the user. Combining this method with a color matching step, identifying skin tones in UV space, may provide an added level of stability to this algorithm in situations where the binary image becomes saturated due to camera motion, or sudden changes to the background.

In the future, multimodal interfaces incorporating gesture and speech are likely to augment, if not replace the keyboard and mouse. With speech recognition well on its way to maturity, action recognition must make up for lost time, and provide algorithms that may be adopted in real world, general-purpose settings.

REFERENCES

Arseneau, S., and Cooperstock, J. 1999a. Real-Time Image Segmentation for Action Recognition. *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*. 86-89.

Arseneau, S., and Cooperstock, J. 1999b. Presenter Tracking in a Classroom Environment. *IEEE, Industrial Electronics Conference*. San Jose, U.S.A. 145-148.

Ayers, D., and Shah, M. 1998. Recognizing Human Actions in a Static Room. *Fourth IEEE Workshop on Applications in Computer Vision*. 42-47.

Benjamin, R. 1990. Object-surface-based Extraction and Manipulation of 3-D Medical Imaging Data. *IEE Colloquium on Low Bit Rate Image Coding*. 12:1-6.

Bharatkumar, A., Daigle, K., Pandy, M., Cai, Q., and Aggarwal, J. 1994. Lower Limb Kinematics of Human Walking with the Medial Axis Transformation. *Proceedings of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*. 70-76.

Birchfield, S. 1998. Elliptical Head Tracking Using Intensity Gradients and Color Histograms. *IEEE Computer Vision and Pattern Recognition*. 232-237.

Blake, A., and Isard, M. 1998. Active Contours. Springer-Verlag, Great Britain.

Bregler, C. 1997. Learning and Recognizing Human Dynamics in Video Sequences. *IEEE Conference on Computer Vision and Pattern Recognition.* 568-574.

Bregler, C., and Omohundro, S. 1995. Nonlinear Manifold Learning for Visual Speech Recognition. *Proceedings of the Fifth International Conference on Computer Vision.* 494-499.

Campbell, N., Thomas, B., and Troscianko, T. 1997. A Two-stage Process for Accurate Image Segmentation. *Sixth International Conference on Image Processing and Its Applications.* 2:655-659.

Chai, D., and Ngan, K. 1999. Face Segmentation Using Skin-Color Map in Videophone Applications. *IEEE Transactions on Circuits and Systems for Video Technology.* 9-4:551-564.

Chan, C., and Sandler, M. 1992. A Complete Shape Recognition System Using the Hough Transform and Neural Network. *International Conference on Pattern Recognition.* 2:21-24.

Davis, J. and Bobick, A. 1997. The Representation and Recognition of Action Using Temporal Templates. *IEEE Conference on Computer Vision and Pattern Recognition.* 928-934.

Davis, J. and Bobick, A. 1998. Virtual PAT: A Virtual Personal Aerobics Trainer. *Workshop on Perceptual User Interfaces.*

Emering, L., Boulic, R., and Thalmann, D. Interacting with Virtual Humans through Body Actions. *IEEE Computer Graphics and Applications.* 18:8-11.

Fukunage, K. 1972. Introduction to Statistical Pattern Recognition. New York, Academic Press. 260-267.

Huang, T. Blostein, S., Werkheiser, A., McDonnell, M., and Lew, M. 1986. Motion Detection and Estimation from Stereo Image Sequences. *IEEE Proceedings of the Workshop on Motion: Representation and Analysis.* 45-46.

Hampel, F., Ronchetti, P., Rousseeuw, P., and Stahel, W. 1986. Robust Statistics – The Approach Based on Influence Functions. Wiley, New York.

Imagawa, K., Shan Lu, and Igi, S. 1998. Color-Based Hands Tracking System for Sign Language Recognition. *The Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition.* 462-467.

Ishibuchi, K., Takemura, H., and Fumio, K. 1992. Real Time Hand Shape Recognition for Man-Machine Interfaces. *IEEE International Conference on Systems, Man and Cybernetics.* 2:1407-1412.

Jain, R., Kasturi, R., and Schunck, B. 1995. Machine Vision. McGraw-Hill, New York.

Kahn, R., and Swain, M. 1995. Understanding people pointing: the Perseus system. *International Symposium on Computer Vision*. 569 –574.

Laumond, J., Jacobs, P., Taix, M., and Murray, M. 1994. A Motion Planner for Nonholonomic Mobile Robots. *IEEE Transactions on Robotics and Automation*. 10-5:577-593.

Lemmens, J. 1994. A Computational Model of Color Perception and Color Naming, Ph.D. dissertation, State University of New York at Buffalo, Computer Science department.

Levine, M. 1985. Vision in Man and Machine. McGraw-Hill, Montreal.

Liebe, C. 1993. Pattern Recognition of Star Constellations for Spacecraft Applications. *IEEE AES Systems Magazine*. 31-39.

Lipardi, C., Trivedi, M., and Harlow, C. 1989. Geometric Modeling and Recognition of Elongated Regions in Aerial Images. *IEEE Transactions on Systems, Man, and Cybernetics*. 19-6:1600-1612.

Marr, D., and Hildreth, E. 1980. Theory of Edge Detection. *Proceedings of R. Soc. Lond. B*, 207:187-217.

Otsu, N. 1979. A Threshold Selection Method from Gray Level Histograms, *IEEE Trans. on Systems, Man, and Cybernetics*. SMC-9:62-66.

Piaggio, M., Fornaro, R., Piombo, A., Sanna, L., and Zaccaria, R. An Optical-Flow Person Following Behaviour. *The Proceedings of the IEEE Joint Conference of ISIC/CIRA/ISAS*. 301-306.

Pinhanez, C., and Bobick, A. 1998. "It/I": A Theater Play Featuring an Autonomous Computer Graphics Character. *M.I.T. Media Laboratory Perceptual Computing Section.* Technical Report No. 455.

Reisfeld, D., Wolfson, H., and Yeshurun, Y. 1995. Context Free Attentional Operators: The Generalized Symmetry Transform. *International Journal of Computer Vision.* 14:119-130.

Rosenfeld, A., and Melter, R. 1989. Digital Geometry: The Mathematical Intelligence. 11-3:69-72.

Rousseeuw P., and Leroy, A. 1987. Robust Regression and Outlier Detection. Wiley, New York.

Sawasaki, N., Morita, T., and Uchiyama, T. 1996. Design and Implementation of High-Speed Visual Tracking System for Real-Time Motion Analysis. *Proceedings from IEEE Computer and Pattern Recognition.* 96:1015-4651.

Torige, A. and Kono, T. 1992. Human-interface by recognition of human gesture with image processing-recognition of gesture to specify moving direction. *IEEE International Workshop on Robot and Human Communication.* 105 –110.

Vogler, C. and Metaxas, D. 1999. Parallel Hidden Markov Models for American Sign Language Recognition. *The Proceedings of the Seventh IEEE International Conference on Computer Vision.* 1: 116 –122.

Weszka, J., Nagel, R., and Rosenfeld, A. 1974. A Threshold Selection Technique. *IEEE Transactions on Computers.* C-23:1322-1326.

Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. 1997. Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol.19-7:780-785.

Yang, J., and Waibel, A. 1995. A Real-Time Face Tracker. Doc:CMU-CS-95-210, School of Computer Science, Carnegie Mellon University, Pittsburg, U.S.A.

Yang, Y., and Levine, M. 1992. The Background Primal Sketch: An Approach for Tracking Moving Objects. *Machine Vision and Applications*. 5:17-34.

Zarit, B., Boaz, S., and Quek, F. 1999. Comparison of Five Color Models in Skin Pixel Classification. *International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*. 58 –63.

Zhan, C. 1971. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*. C-20-1:68-86.

Zhu, Q., and Poh, L. 1988. A Transformation-Invariant Recursive Subdivision Method for Shape Analysis. *9th International Conference on Pattern Recognition*. 2:833-835.