SPATIAL HISTOGRAMS FOR HEAD TRACKING

A Thesis

Presented to

the Graduate School of

Clemson University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Electrical Engineering

---

by

Sriram Rangarajan

December 2005

Advisor: Dr. Stan Birchfield

December 16, 2005

To the Graduate School:

This thesis entitled “Spatial Histograms for Head Tracking” and written by Sriram Rangarajan is presented to the Graduate School of Clemson University. I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical Engineering.

Dr. Stanley T. Birchfield, Advisor

We have reviewed this thesis
and recommend its acceptance:

Dr. Ian D. Walker

Dr. Adam W. Hoover

Accepted for the Graduate School:

ABSTRACT

This work aims to analyze the use of spatial information and the effectiveness of such information for tracking a person's head in a video sequence. This thesis introduces new concepts that make use of color information along with a limited amount of spatial information on a global scale. Also, existing concepts like co-occurrence matrices are adapted to color images to provide a concept that uses color and spatial information on a local scale. The work done also involves the introduction of two texture-based algorithms for tracking. To determine the efficiency of each tracker, the results are compared with a standard algorithm, namely the color histogram based tracker. Experimental results demonstrate the advantage of using a combination of color and spatial information. In addition, other plots and error measures show that the use of a limited amount of spatial information greatly improves tracking results, while also showing that color information is also needed for successful tracking.

Dedication

To my beloved mom and dad.

ACKNOWLEDGMENTS

I am indebted to my advisor Dr. Stan Birchfield. His patience, encouragement and guidance have played a crucial part in helping me develop my research skills.

I would also like to thank Dr. Ian Walker and Dr. Adam Hoover for gracing my committee with their presence.

I would also like to thank my lab mates who have provided me with encouragement and support whenever I needed them.

TABLE OF CONTENTS

Page

TITLE PAGE i

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

1 Introduction 1

1.1 Tracking framework 4

1.1.1 Overview of thesis 7

2 Color histograms 8

2.1 Tracking using color histograms 10

3 Texture histograms 12

3.1 Gabor and log-Gabor filters 12

3.1.1 Computing log-Gabor histograms 14

3.1.2 Tracking using log-Gabor histograms 15

3.2 Haar histograms 16

3.2.1 Computing Haar histograms 17

3.2.2 Tracking using Haar histograms 17

3.3 Edge-orientation histograms 18

3.3.1 Computing edge-orientation histograms 19

3.3.2 Tracking using edge-orientation histograms 21

4 Spatiograms 23

4.1 Spatiograms and histograms 23

Table of Contents (Continued)

Page

4.2 Tracking using spatiograms 25

5 Co-occurrence matrices 29

5.1 Color co-occurrence matrices 30

5.1.1 Spatiograms and co-occurrence matrices 31

5.2 Tracking using co-occurrence matrices 32

6 Experimental results 34

6.1 Log-Gabor histograms 34

6.1.1 Sequence 1 34

6.1.2 Sequence 2 35

6.2 Haar histograms 37

6.2.1 Sequence 1 37

6.2.2 Sequence 2 37

6.3 Edge-orientation histograms 39

6.3.1 Sequence 1 39

6.3.2 Sequence 2 41

6.4 Spatiograms 41

6.4.1 Sequence 1 41

6.4.2 Sequence 2 43

6.5 Co-occurrence matrices 45

6.5.1 Sequence 1 45

6.5.2 Sequence 2 45

7 Conclusion 49

BIBLIOGRAPHY 50

LIST OF TABLES

Table Page

6.1 Mean errors in tracking in x and y for Sequence 1. 47

6.2 Mean errors in tracking in x and y for Sequence 2. 48

LIST OF FIGURES

Figure Page

3.1 Image and its log-Gabor histogram 15

3.2 An image and its edge-orientation histogram. The histogram contains 8 bins for orientation and 3 levels. Each level is a DoG kernel with different σ values. 20

4.1 Three different poses of a person (top), with images generated from the histogram (middle) and spatiogram (bottom). The spatiogram captures spatial relationships among the colors, whereas the histogram discards all spatial information. 25

6.1 Tracking results for color histograms (blue) vs log-Gabor histograms (red) for Sequence 1. Shown are frames 95, 99, 125, 128, 269, 400, 411, 418. . . 35

6.2 Tracking error in *x* and *y* using histograms (blue, dashed) versus log-Gabor histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1. 36

6.3 Tracking error in *x* and *y* using histograms (blue, dashed) versus log-Gabor histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2. 36

6.4 Tracking results for color histograms (blue) vs log-Gabor histograms (red) for Sequence 2. Shown are frames 5, 11, 15, 19, 23, 25, 28, 30. 37

6.5 Tracking results for color histograms (blue) vs Haar histograms (red) for Sequence 1. Shown are frames 122, 125, 129, 146, 400, 407, 413, 465. . . . 38

6.6 Tracking error in *x* and *y* using color histograms (blue, dashed) versus Haar histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1. 38

6.7 Tracking results for color histograms (blue) vs Haar histograms (red) for Sequence 2. Shown are frames 1, 12, 15, 21, 24, 25, 28, 30. 39

List of Figures (Continued)

Figure Page

6.8 Tracking error in *x* and *y* using histograms (blue, dashed) versus Haar histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2. 40

6.9 Tracking error in *x* and *y* using color histograms (blue, dashed) versus edge-orientation histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1. 40

6.10 Tracking results for color histograms (blue) vs edge-orientation histograms (red) for Sequence 1. Shown are frames 99, 123, 127, 168, 200, 311, 403, 413. 41

6.11 Tracking error in *x* and *y* using color histograms (blue, dashed) versus edge-orientation histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2. 42

6.12 Tracking results for color histograms(blue) vs edge-orientation histograms(red) for Sequence 2. Shown are frames 1, 3, 8, 10, 22, 24, 25, 30. 42

6.13 Tracking error in *x* and *y* using color histograms (blue, dashed) versus co-occurrence matrices (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1. . . . 43

6.14 Tracking results for color histograms (blue) vs spatiograms (red) for Sequence 1. Shown are frames 93, 99, 123, 125, 128, 367, 400, 407, 411, 463, 472, 474. 44

6.15 Tracking error in *x* and *y* using color histograms (blue, dashed) versus spatiograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2. 44

6.16 Tracking results for color histograms (blue) vs spatiograms (red) for Sequence 2. Shown are frames 1, 5, 10, 12, 15, 25, 28, 30. 45

6.17 Tracking error in *x* and *y* using histograms (blue, dashed) versus co-occurrence matrices (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1. 46

6.18 Tracking results for color histograms (blue) Vs co-occurrence matrices (red) for Sequence 1. Shown are frames 97, 123, 143, 404, 410, 462, 472. . 46

List of Figures (Continued)

Figure Page

6.19 Tracking error in *x* and *y* using color histograms (blue, dashed) versus co-occurrence matrices (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2. . . . 47

6.20 Tracking results for color histograms (blue) vs co-occurrence matrices (red) for Sequence 2. Shown are frames 1, 5, 12, 15, 22, 25, 28, 30. 48

Chapter 1

Introduction

Histograms have been a popular tool in image processing and computer vision, widely used for detection, recognition and tracking tasks. Computational simplicity and efficiency have been the prime reasons for their popularity in these fields. Over the years as computer vision and image processing have progressed, histograms have become increasingly popular and their ability to represent image data has been put to great use especially in tracking. Various algorithms have been developed to take advantage of the performance of histograms. The robustness of histograms to change in pose and shape has been made use of in various tracking algorithms. With powerful concepts such as histogram intersection and color indexing [26] already in use, new concepts such as the Bhattacharyya coefficient have also been developed to further improve the performance of histograms.

Recent developments in texture recognition and analysis have lead to texture being used widely in combination with color for a variety of pattern recognition tasks. While texture has been widely used for detection, very few forays have been made in the area of tracking using texture. This is mainly due to a fact that the use of spatial information tends to require specific and detailed models of the object to be tracked. This is primarily due to the fact that spatial information contained by pixels changes drastically under such conditions, causing

a tracker to fail. To overcome such a problem, an adaptive technique to update the model maybe required.

While texture cannot be directly used for tracking due to the above problems and considering the popularity and robustness of color histograms, the concept of computing histograms based on texture becomes an innovative and interesting area of exploration with respect to tracking. Such an approach eliminates complex models and also provides a cue for tracking that is relatively robust to changes in object shape and pose. Texture histograms can be easily computed once texture is extracted from an image. Computing texture histograms removes the problem of changes to object shape and pose, since they capture enough spatial information while not taking into account explicit spatial arrangement of pixels. For texture extraction, two different approaches are followed, giving two different head trackers. The first approach uses a bank of log-Gabor filters, while the second approach makes use of Haar wavelets. The reason for the choice of filters are based on the perception of texture by primitive mammalian cortexes (log-Gabor) and simplicity (Haar). To show the robustness and efficiency of such a texture histogram based tracker, the tracking results achieved using both log-Gabor and Haar histogram based trackers are compared with a standard color-histogram based tracker.

Another approach with using spatial information for tracking would be the use of edge-based information. A tracker that computes the orientation of a pixel based on edge information is also introduced here. Gradient-based trackers [2, 1], which use edge-based information in an indirect manner, have been successfully used to track an object or a person and hence the motivation to use edge-based information to track an object arises. Edge-orientation histograms have been used in tasks such as gesture recognition [12] and face recognition [21]. However, use of edge-based information in tracking has disadvantages due to frequent changes in edge information contained by pixels which caused by changes in lighting conditions and other factors. Such changes would have a pronounced effect in cases where template-based methods are used. By creating histograms from edge-

based information, the effects due to variations in lighting are reduced to a great extent, if not eliminated. The edge-histograms are designed to show that edge-based information can be reliably used not only on the boundary of the object, but also inside the boundary of the object to be tracked.

Color has played a pivotal role in computer vision and the use of color histograms has given rise to various successful tracking algorithms [2, 24, 27, 8, 7, 6, 28, 15]. Another approach is to represent the image as a template window of pixel intensities and the window registered with the previous frame to determine the displacement of the object [23, 16]. In such an approach, the spatial arrangement of pixels in the window is expected not to deviate from a low-order parametric motion model. This approach lies at the opposite end of the spectrum with respect to histograms due to the fact that color histograms ignore information about the spatial domain.

Adding a limited amount of spatial information to histograms has the potential of providing the robustness and invariance of histograms while accounting for a richer description of an object. Recently, several approaches have been proposed for doing this. Hager et al. [15] use multiple spatially-weighted histograms in a kernel-based framework, and Elgammal et al. [10] present a probabilistic framework for considering feature values (e.g., color) and feature locations. In both formulations, the asymptotic behavior of the algorithm approaches either template-based tracking or traditional histogram-based tracking as a parameter is varied between two extremes. Another approach using spatial information was proposed by Jepson et al. [19], who use phase information for tracking. Such an approach requires the model to be updated regularly to prevent the tracker from failing when out-of-plane rotations occur. The use of color information along with a limited amount of spatial information, introduced by Birchfield and Rangarajan [3], showed improved results when compared to color histograms. The use of the mean-shift algorithm, introduced by Comaniciu et al. in [7], for the new technique also showed that such an attempt could be achieved without significant loss of computational efficiency.

A single histogram in which each bin is spatially weighted by the mean and covariances of the locations of pixels that contribute to that bin is called a spatial histogram or *spatiogram*. Such a concept gives a richer representation by capturing not only the values of the pixels, but also their spatial relationships. Using the means and covariances of pixel locations provides a way of capturing a limited amount of spatial information on a global scale, and it can be shown that spatiograms are indeed higher order histograms [3].

Instead of using the global mean and covariance of the location of pixels in a histogram bin, spatial information can be captured by making use of the relationship between pairs of pixels. Such an approach makes use of the concept of co-occurrence matrices[9]. Co-occurrence matrices describe spatial information based on repeated occurrence of a configuration of pixels. In other words, a co-occurrence matrix captures information about how frequently two pixels with particular values, such as gray-level values or intensities, appear next to each other in an image. Such a method, when used for representing color information instead of intensities, provides an ideal way of capturing a limited amount of spatial information. While co-occurrence matrices have been popularly used to describe gray-level texture, we adapt co-occurrence matrices to capture spatial information for color images and demonstrate efficient tracking results on an image sequence of a person's head.

1.1 Tracking framework

A few years ago, a successful real-time head tracker [2] was presented that uses two modules that are orthogonal in operation, in the sense that when one module fails, the other module still keeps the tracker on the target. The first module is the intensity gradient module, which has two variants - gradient magnitude and gradient dot-product, that works by matching gradients around the boundary of a person's head. The second module is a color histogram-based module that develops a model histogram of the head using an elliptical

mask from the first frame and compares it with target histograms obtained from the current frame.

The head is modeled as a vertical ellipse,with an aspect ratio of 1.2, with the co-ordinates of the center of the ellipse given by (x, y) and minor axis σ. Thus, the location of the head or the state is denoted by $S = (x, y, \sigma)$ and for every frame, the *state* is updated by the location whose values best match the values in the model. The goodness of this match depends on the intensity gradients around the object (head) and the color histogram of the interior of the object (face + hair).

The intensity gradient module works by computing the magnitude of gradients around the head under the constraint that the direction of the gradient should be perpendicular to the perimeter of the ellipse. The gradient score or goodness of match is then converted into a percentage score to facilitate easy combination with the color histogram module. The color histogram module works by computing a model histogram from the first frame, where the tracker is manually initialized. The module then produces a likelihood by comparing target histograms from current frames with the model histogram that is obtained from the first frame. The histograms are compared by histogram intersection [26] to provide the likelihood measure. This measure is then converted into a percentage score and combined with the percentage score from the intensity gradients module to provide an overall likelihood location of the head. A constant velocity prediction is also employed to enable the handling of a moving target, eliminating the need for complex motion models.

While a robust tracker is described above, it also provides a framework, where one module maybe replaced by another efficient module to overcome the shortfalls of a particular module. With such a framework already established, various modules or “cues” can be explored to improve the color histogram module’s performance.

The base for such a framework is the intensity gradient module from the tracker described in [2]. The intensity dot product module produces a likelihood score by matching

the gradients on the boundary of the object to be tracked. The likelihood score thus obtained is shown in Equation 1.1.

$$\Phi_g = \frac{1}{N_\sigma} \sum_{i=1}^{N_\sigma} \mid n_\sigma . g_s(i) \mid \tag{1.1}$$

where N_σ is the number of points on the ellipse, n_σ is the normal to the ellipse, and $g_s(i)$ is the gradient score at a point *i* on the ellipse.

From this equation it can be seen that the goodness of match is based not only on the gradient magnitude, but also on the requirement that the direction of the gradient be perpendicular to the perimeter of the ellipse. While the intensity gradient works on the boundary of the object or the perimeter of the ellipse, it may be combined with other modules that work on the inside of the object boundary, thereby providing a complementary system, where one module would work when the other fails. To facilitate such an addition, the likelihood score obtained in Equation 1.1 should be normalized. The normalized likelihood shown in Equation 1.2 is a percentage score which is done to provide an easy combination with other modules.

$$\bar{\Phi_g}(S) = \frac{\Phi_g(S) - min_{S_i \in S} \Phi_g(S_i)}{max_{S_i \in S} \Phi_g(S_i) - min_{S_i \in S} \Phi_g(S_i)} \tag{1.2}$$

If $\bar{\Phi_M}(S_i)$ is the normalized likelihood from another module, the final likelihood or the state maybe computed as

$$S^* = \arg\max_{S_i \in S} \{\bar{\Phi_g}(S_i) + \bar{\Phi_M}(S_i)\} \tag{1.3}$$

The advantage of such a framework is that any number of modules may be added to provide a final likelihood by converting the likelihood obtained from those modules into percentage scores. Thus, a framework is defined that facilitates the addition of one or more modules to the intensity gradient module to provide an efficient, robust tracker.

1.1.1 Overview of thesis

The work done in this thesis focuses on modules that contain varying ranges of spatial information that are designed to fit into such a framework. Presented in this thesis are trackers such as color histograms, which contain no spatial information, color spatiograms and color co-occurrence matrices, which contain a mix of spatial information and color information, edge-orientation histograms, log-Gabor histograms and Haar histograms, all of which rely only on spatial information. The motivation behind such an effort is to show that the use of a limited amount of spatial information along with color information helps improve tracking results drastically.

In this thesis, Chapter 2 deals with the implementation of a color histogram module. The chapter describes the re-implementation of the color histogram module from [2]. Chapter 3 describes the implementation of two types of texture histograms, one obtained using a bank of log-Gabor filters and another obtained using Haar wavelets, and the concept of edge-orientation histograms. Chapter 4 introduces the concept of spatiograms, which is a histogram that contains a limited amount of spatial information. Chapter 5 deals with the concept of co-occurrence matrices, where limited information about the spatial domain is captured on a local scale along with color information. The last two chapters deal with experimental results and conclusions drawn from this thesis.

Chapter 2

Color histograms

A successful tracker that uses color histograms, combined with an intensity gradient module was introduced by Birchfield [2]. This chapter describes a reimplementation of the color histogram module used in that head tracker. Color has been an important cue in the field of tracking. Various trackers that exploit the uniqueness of skin color have been introduced in the past [11, 14, 18, 20]. Such an approach provides reliable tracking as long as there are no skin-colored objects in the background. The approach also fails when the head undergoes an out-of-plane rotation. This is due to the fact that a person's head contains more than a single color and at the least, bimodal due to the presence of skin and hair. Hence, any tracker that aims to address out-of-plane rotation has to take into account the fact that the head looks completely different from its frontal view, which in most cases, is used to generate a model.

This problem can be handled with the use of color histograms, since they are invariant to complicated non-rigid motion due to the fact that they ignore geometric or spatial information. This property means that histogram intersection [26] between the model histogram,which contains information about both skin color and hair color, and a histogram from a region containing both hair and skin colors would provide a better match when com-

pared to the match between the model and a histogram from a region containing only skin color or only hair color.

Another reason for the popularity of color histograms in the field of tracking is the fact that they are invariant to translation, rotation and only change due to occlusion, scaling or a change in the angle of view. This means that a relatively small amount of data can be used to represent even a complex three-dimensional object. Computing color histograms, in addition to these advantages, is also computationally efficient.

Given a particular color space and an image, a color histogram of the image can be obtained by discretizing the image colors and then counting the number of times each discrete color appears in the image. Thus, a color histogram captures the frequency of occurrence of a particular color value over an image. In other words, let an image be considered as a mapping of pixels **x** to values **v** such that

$$I : \mathbf{x} \rightarrow \mathbf{v}$$

where, **x** represents pixels given by $[x, y]^T$ and **v** represents values of pixels which can be either gray level values or color values. A *histogram* of I is given by

$$h_I : \mathbf{v} \rightarrow \mathcal{Z}^* \tag{2.1}$$

where **v** represents the color values and $\mathcal{Z}^*$ is a non-negative integer. Since an image contains three color channels, the color histogram is given by

$$H(i_1, i_2, i_3) = n \tag{2.2}$$

where i_1,i_2 and i_3 are the indices for each color channel and n is the number of pixels in a bin. Each index is computed as

$$i_k = C_k(x,y)\frac{nbins}{\max(C_k)} \tag{2.3}$$

where $C_k(x,y)$ is the color value of the pixel at location (x,y) for the color channel C_k and *nbins* is the number of bins corresponding to that color channel and $k = 1, 2$ or 3.

2.1 Tracking using color histograms

Tracking a person's head using color histograms requires the creation of a model histogram, which is generated at the start of tracking and maintained throughout the tracking period, and then comparing target histograms generated from each frame in a sequence to generate a likelihood map. The likelihood map gives the probability of the person's head or the object which is being tracked to be in a particular place. The size of the likelihood map is determined by the size of the local search window which is used for tracking. For computing color histograms, the color channels are binned into 8, 8, and 4 bins respectively. The model histogram is computed when the tracker is initialized by the user. At initialization, the user defines the initial "state" of the tracker i.e. the center of the ellipse defined by co-ordinates (x,y) and the minor axis of the ellipse defined by σ. After the initial state is defined, an ellipse mask is applied around the region defined by the user during initialization and the model histogram is computed as shown in equation 2.2. To provide reliable tracking, it is imperative that the region contains a three-quarters view of the person who is to be tracked. This is done to ensure that the model captures information about hair and skin color. To produce a likelihood map of possible locations of the target in the next frame, candidate histograms are obtained from the current frame and compared with the model histogram. The model histogram, h, and the target histogram, h', are compared using histogram intersection [26] to obtain the likelihood using histogram intersection as

shown.

$$\Phi_c(S) = \frac{\min(n_b, n'_b)}{\sum_{b=1}^{N} n'_b}. \tag{2.4}$$

where n_b represents the bins in the model histogram h and n'_b represents the bins in the target histogram h' and S is the state from which the target histogram is obtained.

Since the tracker incorporates scaling, histogram intersection could potentially provide erroneous values for similarity in a case when the target histogram is a scaled version of the model. Hence, to avoid such a problem, the target histograms are normalized with respect to the model as shown.

$$\hat{n'_b} = n'_b \frac{\sum_{b=1}^{N} n_b}{\sum_{b=1}^{N} n'_b} \tag{2.5}$$

where n_b represents the bins of the model histogram, n'_b represents the bins of the target histogram and N is the number of bins in the histogram. Once a likelihood map is obtained, it is converted into a percentage score, as shown in equation 2.6 to enable an easy combination with the other modules that may be used in equation 1.3.

$$\bar{\Phi_c}(S) = \frac{\Phi_c(S) - min_{S_i \in S}\Phi_c(S_i)}{max_{S_i \in S}\Phi_c(S_i) - min_{S_i \in S}\Phi_c(S_i)} \tag{2.6}$$

where $\Phi_c(S)$ represents the likelihood obtained at the state S from which the target histogram was obtained.

Chapter 3

Texture histograms

Texture has been used for pattern recognition tasks like face recognition, iris recognition and image retrieval [22, 17, 25]. Although popular for such tasks, very few tracking systems have been based on texture and those that use texture for tracking use complex models for tracking such as modeling the head as a 3-D cylinder [4]. The idea of using texture histograms though, has not been widely explored. One major problem that arises with using texture for tracking is the requirement of complex models to represent a target. This is due to the fact that texture varies widely with changes in lighting or scale. However, eliminating templates and obtaining histograms from texture extracted from an image would mean that complex models are not required to model an object. The idea of using texture histograms is based on the fact that texture is similar to color and computing a texture histogram is computationally less expensive compared to existing texture-based methods for tracking.

3.1 Gabor and log-Gabor filters

Gabor filters have been used in texture analysis due to the fact that the real part of complex Gabor functions fits the receptive field weight functions found in simple cells in the cat

striate cortex. The Gabor function in space domain is given by

$$g(x,y) = s(x,y)w_r(x,y) \tag{3.1}$$

where $s(x,y)$ is a complex sinusoidal, known as the carrier,and $w_r(x,y)$ is a 2-D Gaussian known as the envelope. For a given image I(x,y) with size $P \times Q$ its discrete Gabor wavelet transform is given by a convolution:

$$G(x,y) = \sum_s \sum_t I(x-s, y-t)\Psi^*_{mn}(s,t) \tag{3.2}$$

where, s and t are the filter mask size variables, and Ψ^*_{mn} is the complex conjugate of Ψ_{mn}, where m and n are the scales and orientations of the Gabor wavelet and Ψ_{mn} is a class of self-similar functions obtained from dilating and rotating the Gabor wavelet.

Gabor filters are a traditional choice for obtaining localized frequency information. They offer the best simultaneous localization of spatial and frequency information. However they have two main limitations. The maximum bandwidth of a Gabor filter is limited to approximately one octave and Gabor filters are not optimal if one is seeking broad spectral information with maximal spatial localization.

An alternative to the Gabor function is the Log-Gabor function proposed by Field [1987]. Log-Gabor filters can be constructed with arbitrary bandwidth and the bandwidth can be optimized to produce a filter with minimal spatial extent. Field suggests that natural images are better coded by filters that have Gaussian transfer functions when viewed on the logarithmic frequency scale. On a linear frequency scale, the log-Gabor function has a transfer function of the form

$$G(w) = e^{(-\log(w/w_0))/(2(\log(k/w_0)^2)} \tag{3.3}$$

where w_0 is the filter's center frequency. The Log-Gabor filter by definition does not have a DC component and has an extended tail at the high frequency end. Field suggested that log Gabor functions, having extended tails, should be able to encode natural images more efficiently than ordinary Gabor functions, which would over-represent the low frequency components and under-represent the high frequency components in any encoding.Another point in support of the log Gabor function is that it is consistent with measurements on mammalian visual systems which indicates that the cell responses are symmetric on the log frequency scale.

3.1.1 Computing log-Gabor histograms

A log-Gabor histogram is generated by convolving an incoming image with the log-Gabor filter bank. Before convolving with the filter bank, an ellipse mask is generated such that the resulting image contains only the head of the person.The histogram is then generated for a fixed number of scales and orientations, after convolution with the filter bank. The number of scales and orientations for the filter bank is set at the start which defines the size of the log-Gabor histogram.The log-Gabor histogram is computed as shown.

$$h_{S,O} = \sum_{x} \sum_{y} I_{S,O}(x,y) \tag{3.4}$$

A single bin in the log-Gabor histogram is represented by $h_{S,O}$, S represents the scale, O represents the orientation, and $I_{S,O}(x,y)$ is the result of convolving the image with the filter bank at scale S and orientation O. The process is then repeated over all scales and orientations to obtain the final histogram. From the equation, it is clear that the histogram for each scale and orientation is the sum of the resulting filter outputs for that particular scale and orientation. For reliable tracking, only the real part of the filter outputs are taken into consideration. To provide a visual representation of the log-Gabor histogram, an image and its histogram are shown in Figure 3.1.

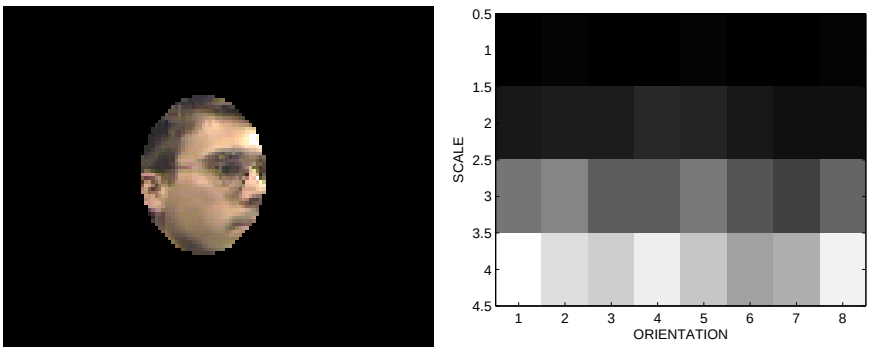

Figure 3.1: Image and its log-Gabor histogram

3.1.2 Tracking using log-Gabor histograms

Tracking using log-Gabor histograms follows an approach that enables the module to be plugged into equation 1.3. To enable such a fit, a percentage likelihood score is obtained as a result of comparing a model histogram with several target histograms, thereby providing an estimate for a candidate location of the target in the next frame. A three-quarters view of a person's head is used to construct the model histogram after initialization. The initialization is done manually by the user and the model histogram is constructed from the initialization point, i.e., the ellipse coordinates specified by the user. The user also specifies the number of scales and orientations over which the histogram is to be computed and it remains constant till the end of tracking. Once the ROI is specified by the user, the model histogram is computed as shown in Equation 3.4, after an elliptical mask is applied around the ROI. The target histograms is obtained in the same way as the model histogram, with the only difference being the fact that the ROI is determined by the state which is updated by the tracker. As in the case of computing the model histogram, an ellipse mask is created around the ROI in the current image and the target histogram is then computed as shown in Equation 3.4. To counter the problem of variations in scale between the target and the model histogram, the target histogram is normalized as shown.

$$\hat{n'_b} = n'_b \frac{\sum_{b=1}^{B} n_b}{\sum_{b=1}^{B} n'_b} \tag{3.5}$$

Once the target histogram is normalized, a likelihood measure is obtained based on the similarity between the model and the target histogram, which is computed using histogram intersection, as shown.

$$\Phi_l(S) = \frac{\min(n_b, \hat{n}'_b)}{\sum_{j=1}^{B} n_b}. \tag{3.6}$$

where n_b represents the bins in the model histogram h and n'_b represents the bins in the target histogram h' and S represents the state from which the target histogram is obtained.

The likelihood obtained from Equation 3.6 is then normalized to obtain a percentage score to enable a smooth combination with the framework suggested in equation 1.3.

$$\bar{\Phi}_l(S) = \frac{\Phi_l(S) - min_{S_i \in S}\Phi_l(S_i)}{max_{S_i \in S}\Phi_l(S_i) - min_{S_i \in S}\Phi_l(S_i)} \tag{3.7}$$

where $\Phi_l(S)$ represents the likelihood obtained at the state S from which the target histogram was obtained.

3.2 Haar histograms

Haar wavelets have commonly been used in various image processing algorithms including image compression. Haar wavelets are the oldest and simplest wavelets and are capable of uniformly approximating any continuous function. The idea of using Haar wavelets is based on their simplicity and ability to describe basic texture, thereby making them computationally efficient. The Haar wavelet works in such a way that it first operates on adjacent horizontal elements in the image matrix and then on the vertical elements of the image matrix. For the sake of computational efficiency and simplicity, a 4×4 Haar matrix is used for convolution with the images.

3.2.1 Computing Haar histograms

For our algorithm, we use a 2-D Haar wavelet pyramid, with 3 levels for texture extraction. At the end of each level, the image is sub-sampled but a factor of 1/2, i.e. reducing it to half the size of the image given to the current level, and passed on to the next level. Though it is common practice to use a 4-level Haar wavelet pyramid, we observed that the images after the third level do not make a difference while computing histograms.The histogram is calculated from the image after convolving with the Haar kernel over 3 levels, with four orientations each. In theory, while all four images at each level are used to compute the histogram for that level and scale, the low-pass image can be neglected, since the information contained is redundant, without affecting performance. The Haar histogram of an image, n_b, is calculated as shown.

$$n_b(S,O) = \sum_x \sum_y |(H_{S,O}(x,y))| \tag{3.8}$$

where, n_b represents the Haar histogram at a particular scale and orientation, S represents the number of scales, which is fixed by setting the number of levels to 4 specified for the filter bank,O represents the number of orientations, and $H_{S,O}(x,y))$ is the result of convolving the image with the Haar pyramid.

3.2.2 Tracking using Haar histograms

Tracking a target follows a similar approach to that of log-Gabor histograms. A model histogram is computed using a three-quarters view of a person's head, after being initialized manually. The model histogram is computed as shown in equation 3.8 after an ellipse mask is applied around the ROI, which is defined by the initial state. A likelihood map is obtained by comparing target histograms, obtained from the current frame. The updated state and the search range determine the number of target histograms to be compared and hence the size of the likelihood map. The target histograms are computed as shown in equation 3.8,

with the ROI determined by the the state, which is updated at the start of the current frame, based on the likelihood obtained from the tracker. Since scaling is a factor, it is essential that the target histogram be scaled with respect to the model to ensure an accurate measure of similarity between the model and the target histograms. The target histograms are scaled as shown in equation 3.9.

$$\hat{n'_b} = n'_b \frac{\sum_{b=1}^{N} n_b}{\sum_{b=1}^{N} n'_b} \tag{3.9}$$

Once the target histograms are normalized, a likelihood measure is obtained based on the similarity between the model and target histograms. This likelihood measure is obtained, using histogram intersection, as shown.

$$\Phi_h(S) = \frac{\min(n_b, \hat{n'_b})}{\sum_{j=1}^{B} n_b}. \tag{3.10}$$

where n_b represents the bins in the model histogram h and n'_b represents the bins in the target histogram h' and S represents the state from which the target histogram is obtained. The likelihood obtained from Equation 3.10 is then normalized as shown to obtain a percentage score to enable the module to be plugged into equation 1.3.

$$\bar{\Phi_h}(S) = \frac{\Phi_h(S) - min_{S_i \in S}\Phi_h(S_i)}{max_{S_i \in S}\Phi_h(S_i) - min_{S_i \in S}\Phi_h(S_i)} \tag{3.11}$$

where $\Phi_h(S)$ represents the likelihood obtained at the state S from which the target histogram was obtained.

3.3 Edge-orientation histograms

A natural approach to include spatial information would be to account for edges in an image. One method would be to compute the orientation of a pixel based on its edge information. Although such an approach is primitive when compared to texture extraction, it is computationally less intensive compared to texture extraction methods that use com-

plex filters. Computing histograms based on the orientation of a pixel captures a great deal of spatial information and yet, is not as complex as template-based methods. The reason for such an approach is due to the wide-spread use of gradients, which essentially contain information about edges, in the field of tracking.

Edges in an image carry vital information compared to features such as color, texture etc. Edges and Lines define structures, boundaries of objects among others in an image. An edge maybe defined as a jump in intensity from one pixel to another. Edges may also be defined as discontinuities in the image intensity due to changes in scene structure. Such discontinuities may originate from different scene features and can describe the information that an image of the external world contains.In most cases, a pixel location is declared as an edge location if the gradient value at that location exceeds a threshold. Edge detection is the process of identifying such sharp discontinuities that exist in an image.

3.3.1 Computing edge-orientation histograms

Edge detection is considered by many as the first task in image processing and as a result, numerous methods have been proposed for edge detection. A few of the popular edge detection algorithms are the Canny edge detector [5], Prewitt edge detector, Sobel edge detector, Laplacian of Gaussian and Difference of Gaussian [13]. Edge detection is achieved using a DoG (difference of Gaussian) kernel. Since a smoothing operation is required to remove noise and prevent the tracker from being sensitive to small edges, the Difference of Gaussian kernel is chosen as the edge detector for the module. The DoG kernel is shown in equation 3.12. For computing the edge-orientation histograms a three-level DoG kernel is used. “Level” in this term refers to the value of σ in the DoG kernel and hence, the outputs of three DoG kernels, each with different σ values are used to compute the histograms.

$$\nabla I = I * \frac{1}{\sqrt{2\pi}} \left[\frac{1}{\sigma_1} e^{-\left(\frac{x^2+y^2}{2\sigma_1^2}\right)} - \frac{1}{\sigma_2} e^{-\left(\frac{x^2+y^2}{2\sigma_2^2}\right)} \right] \tag{3.12}$$

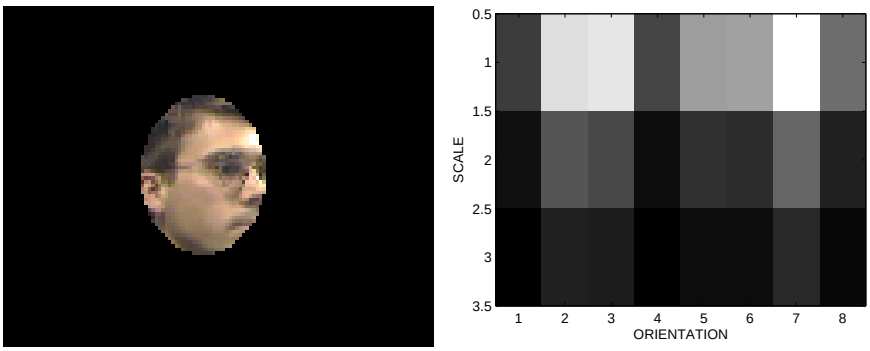

Figure 3.2: An image and its edge-orientation histogram. The histogram contains 8 bins for orientation and 3 levels. Each level is a DoG kernel with different σ values.

The gradient images are obtained by convolving the image with DoG along the horizontal and vertical directions. Once the gradient images along *x* and *y* are computed, edge-orientation histograms can be obtained by computing the orientation of pixels from the two images. Once the orientation at a particular pixel is obtained, a histogram maybe computed, with the index for the bin in which the pixel is to be assigned obtained from the orientation of the pixel itself. The orientation θ is computed as shown in equation 3.13.

$$\theta = atan2\left(\frac{\frac{\partial I}{\partial Y}}{\frac{\partial I}{\partial X}}\right) \tag{3.13}$$

From Equation 3.13, it can be noticed that an edge along the vertical direction will have an orientation of 0 degrees. The edge-orientation histogram can be computed as shown in equation 3.14.

$$n_b(S, \frac{2\pi}{\theta}) = \sum_{1}^{x}\sum_{1}^{y} abs\left[\frac{\partial I}{\partial x} + \frac{\partial I}{\partial y}\right] \tag{3.14}$$

where, $\frac{2\pi}{\theta}$ is the bin in which a pixel is to be classified, *S* is the scale or the level. As mentioned earlier, 3 levels, each comprised of a DoG kernel with different σ values is used to compute the edge-orientation histograms. Figure 3.2 shows an image and its edge-orientation histogram.

3.3.2 Tracking using edge-orientation histograms

The edge-orientation histogram module is designed in such a way that it can fit into equation 1.3. A three-quarters view of the person's head is used to create a model histogram as shown in equation 3.14, after an ellipse mask is applied around the ROI specified by the user during initialization. The model histogram and the image with the ellipse mask is shown in figure 3.2. To obtain a likelihood map, target histograms obtained from the current frame are compared with the model histogram and the state of the tracker is then updated based on the likelihood map. Target histograms are computed the same way as the model histogram, as shown in equation 3.14. The difference is that the target histograms are obtained from the updated state from the previous frame and the ROI, hence is defined by the local search window and the updated state. One problem that arises when comparing target histograms with the model histogram is the issue of scaling. When scaling is implemented for tracking, two histograms with different scales, but from the same ROI appear dissimilar when compared, even though they are just scaled versions of each other. To handle this problem, the target histograms are normalized before being compared with the model histograms as shown in equation 3.15.

$$\hat{n'_b} = n'_b \frac{\sum_{b=1}^{N} n_b}{\sum_{b=1}^{N} n'_b} \tag{3.15}$$

Once the target histograms are normalized, the histograms are then compared to produce a likelihood map using histogram intersection as shown.

$$\Phi_l(S) = \frac{\min(n_b, \hat{n'_b})}{\sum_{j=1}^{B} n_b}. \tag{3.16}$$

where n_b represents the bins in the model histogram h and n'_b represents the bins in the target histogram h' and S represents the state from which the target histogram is obtained. After a likelihood map is obtained, the likelihood is then normalized to a percentage score

as shown

$$\bar{\Phi_e}(S) = \frac{\phi_e(S) - min_{S_i \in S}\Phi_e(S_i)}{max_{S_i \in S}\Phi_e(S_i) - min_{S_i \in S}\Phi_e(S_i)} \tag{3.17}$$

where $\Phi_e(S)$ represents the likelihood obtained at the state S from which the target histogram was obtained. This normalized likelihood is computed to enable a simple and easy combination with the intensity gradient module in the framework given by equation 1.3.

Chapter 4

Spatiograms

While histograms ignore all spatial information, a *spatiogram* takes into account both occurrence information about the range of the function, as in a histogram, and information about the (spatial) domain as well.

The concept of a single histogram in which each bin is spatially weighted by the mean and covariance of the locations of the pixels that contribute to that bin is introduced in [3]. This concept known as the spatiogram may be thought of as a geometric model bridging the gap between histograms, which allow for arbitrary transformations, and more specific models such as translation, similarity, affine, projective, or B-splines. Like histograms, spatiograms are efficient to compute, and they enable comparison between corresponding image patches without specifically calculating the actual geometric transformation between them. Nevertheless, like the more specific models, spatiograms retain some information about the geometry of the patches.

4.1 Spatiograms and histograms

Spatiograms are simply histograms with higher-order moments, with histograms being zeroth-order spatiograms. Spatiograms are a richer representation, capturing not only the values of the pixels but also to a limited extent,their spatial relationships as well. The in-

formation regarding the domain is retained by using higher-order moments of the binary function g, where the ith-order moment is given by

$$h_f^{(i)}(v) = \sum_{x \in \mathcal{X}} x^i g_f(x, v) \tag{4.1}$$

The kth-order spatiogram is defined as a tuple of all the moments up to order k: $\langle h_f^{(0)}(v), \ldots, h_f^{(k)}(v) \rangle$. A histogram, thereby, is just a zeroth-order spatiogram.

An image is a two-dimensional mapping $I : \mathbf{x} \rightarrow v$ from pixels $\mathbf{x} = [x, y]^T$ to values v. The meaning of these values is arbitrary and they may represent raw gray-level intensities or component colors, or the result of preprocessing (quantization, color space transformation, wavelet coefficients, etc.). This gives a mapping from pixels to values.

The second-order spatiogram of an image is represented as

$$h_I^{(2)}(b) = \langle n_b, \mu_b, \Sigma_b \rangle, \qquad b = 1, \ldots, B, \tag{4.2}$$

where n_b is the number of pixels whose value is that of the bth bin, and μ_b and Σ_b are the mean vector and covariance matrices, respectively, of the coordinates of those pixels. The number $B = \mid \mathcal{V} \mid$ is the number of bins in the spatiogram. Notice that

$$h_I^{(0)}(b) = n_b, \qquad b = 1, \ldots, B$$

is just the histogram of I.

Being a higher order histogram, all concepts applicable to histograms, such as histogram intersection and Bhattacharyya coefficients, are applicable to spatiograms as well. In the coming sections, the term spatiogram shall refer to the second-order spatiogram unless mentioned otherwise. Figure 4.1 shows a visual representation, with the original image and the quantized images obtained from histograms and spatiograms. The images from the histogram and spatiogram were obtained by using them as a generative model and sampling

Figure 4.1: Three different poses of a person (top), with images generated from the histogram (middle) and spatiogram (bottom). The spatiogram captures spatial relationships among the colors, whereas the histogram discards all spatial information.

the PDF. A closer look at the image generated from spatiograms shows a faint outline of the person's head and the hairline.

4.2 Tracking using spatiograms

Tracking a person's head follows an approach that enables the user to plug the module into equation 1.3, described in section 1.1. A likelihood score for the location of the object being tracked may be obtained from the module by comparing two spatiograms, one being a model histogram and another being a target histogram. To obtain the model spatiogram, the object to be tracked is presented to the camera, either at the first frame, or the model maybe generated offline. In the case of tracking a person's head, a three quarters view of

the person's head is used to obtain the model spatiogram. Such a view is chosen to enable the spatiogram module to capture color information of both hair and skin. This is critical in a case where the head undergoes a 180 degree rotation and only the back of the head is visible to the camera. In such a case, without information about hair color, the spatiogram module would fail. This is due to the fact that the target spatiogram, which would contain very little information about skin color and more information about hair color, will be completely different from the model spatiogram, where the color information would be reversed. Thus, a limited amount of information about hair color helps the module track the head efficiently even when it undergoes a 180 degree rotation.

The model spatiogram is obtained by applying an ellipse mask,whose center is manually specified by the user when "initializing" the tracker in the first frame. The minor axis of the ellipse or the "scale" is also fixed in the first frame, thereby creating a ROI that is specified by the user. The spatiogram is then computed for the initial state $S_{init} = (x, y, \sigma)$, where (x, y) is the center of the ellipse and the minor axis is given by σ. The model thus obtained is used till the end of tracking, to be compared with target histograms obtained from the current frame to provide a likelihood estimate of the position of the target in the next frame. The target histogram is obtained from the current frame by using a local search window around the state from the previous frame.

The likelihood map of the target being present in the current frame is obtained by comparing various target spatiograms, obtained using local search, with the model spatiogram. The similarity between the model spatiogram, denoted by h, and the target spatiogram, denoted by h' can be computed as the weighted sum of similarities between the two histograms as shown to produce a likelihood measure,

$$\Phi_s(S) = \sum_{b=1}^{N} \psi_b \rho_n(n_b, n'_b). \tag{4.3}$$

where n_b represents the bins in the model histogram h, n'_b represents the bins in the target histogram h' and S represents the state from which the target histogram is obtained.

For a zeroth-order spatiogram, ψ_b is set to 1. For a second-order spatiogram, the ψ_b is computed as

$$\psi_b = \eta \exp\left\{-\frac{1}{2}(\mu_b - \mu'_b)^T \hat{\Sigma}_b^{-1}(\mu_b - \mu'_b)\right\} \tag{4.4}$$

where η is the Gaussian normalization constant and $\hat{\Sigma}_b^{-1} = (\Sigma_b^{-1} + (\Sigma'_b)^{-1})$. It is clear that $\rho_n(n_b, n'_b)$ is the similarity between the histogram bins and this similarity can be obtained by using two methods - histogram intersection and Bhattacharyya coefficient [7]. The similarity measure obtained using histogram intersection is given by

$$\phi(s) = \frac{\min(n_b, n'_b)}{\sum_{j=1}^{B} n_b} \tag{4.5}$$

The similarity between the bins of the histograms can also be computed using the Bhattacharyya coefficient as shown.

$$\phi(s) = \frac{\sqrt{n_b n'_b}}{\sqrt{\left(\sum_{j=1}^{B} n_j\right)\left(\sum_{j=1}^{B} n'_j\right)}}. \tag{4.6}$$

One problem that arises with the use of scaling is that the scales of the target spatiogram and the model spatiogram are not always the same. The result of such a problem is that even when two spatiograms are similar and with the only difference being the scaling factor, their similarity when computed using either of the methods suggested would produce an erroneous value. To overcome this problem, the target spatiogram is normalized with respect to the model spatiogram as shown.

$$\hat{n'_b} = n'_b \frac{\sum_{b=1}^{N} n_b}{\sum_{b=1}^{N} n'_b} \tag{4.7}$$

To compute the likelihood map, the normalized target spatiogram is compared with the model spatiogram using either histogram intersection or Bhattacharyya coefficient. Once the likelihood is obtained, it is then normalized to obtain a percentage score and the final likelihood is given by

$$\bar{\Phi}_s(S) = \frac{\phi_s(S) - min_{S_i \in S}\Phi_s(S_i)}{max_{S_i \in S}\Phi_s(S_i) - min_{S_i \in S}\Phi_s(S_i)} \tag{4.8}$$

where $\Phi_s(S)$ represents the likelihood obtained at the state S from which the target histogram was obtained. This normalized likelihood is computed to enable a simple and easy combination with the intensity gradient module in the framework given by equation 1.3.

Chapter 5

Co-occurrence matrices

Spatial information can be added to color histograms by two ways. One way is to incorporate the spatial information globally, using the mean and variance of pixels in a histogram bin, such as in a spatiogram [3]. Another way of incorporating spatial information is to create a histogram-like matrix that not only captures information at a single pixel, but also includes information about a neighboring pixel. Such a concept, known as the co-occurrence matrix, where spatial information is captured locally can be used successfully to track a given target. By definition, a co-occurrence matrix is a two-dimensional array in which the rows and columns are represented by a set of possible image values. The co-occurrence matrix of an image I(r,c)is given by

$$C[x,y] = j \tag{5.1}$$

where x = I[r,c] and y = I[r+dr,c+dc] and j is the number of pixel pairs with the value x and y. A simpler way of defining the co-occurrence matrix is shown below

$$C[i,j] = \sum_{x=1}^{N} \{f(x), f(x+dx)\} \tag{5.2}$$

On the other hand, the histogram of an image I, given by H(I), where x denotes a pixel is given by

$$H(I) = \sum_{x=1}^{N} f(x) \tag{5.3}$$

This means that the co-occurrence matrix captures not only the value of the current pixel x, but also captures the value of a neighboring pixel of x denoted by x_n. The neighbor maybe any one of eight possible pixels, surrounding the current pixel. The choice of the neighbor to be included in the co-occurrence matrix is made by the user at the start of tracking and remains constant throughout the experiment. For ease of use and reference, we use the neighbor immediately before the current pixel, say x-1. Comparing equations 5.3 and 5.2, it is clear that co-occurrence matrices are very similar to histograms.

5.1 Color co-occurrence matrices

The concept of co-occurrence matrices can be applied to a color image to store spatial information, to provide a robust cue for tracking a target, say a person's head. The color co-occurrence matrix, henceforth referred to as co-occurrence matrix unless mentioned otherwise, consists of three matrices, one for each color channel. The co-occurrence matrix for an image with three color channels. is computed as follows. The color values of a pixel in the first color channel of an image, given by I(x,y) is used as the index to increment a counter. The index is always a pair, based on the current pixel and its neighbor, thereby providing a unique way of capturing spatial information locally. Repeating the same for the other two channels provides a matrix containing three co-occurrence matrices, one for each channel. The co-occurrence matrix for a single color channel is shown in equation 5.4.

$$C(I(r,c), I(r+dr, c+dc)) = j. \tag{5.4}$$

Each color channel has one such matrix associated with it, and hence, a color image would be represented by a set of three co-occurrence matrices. From equation 5.4, it is clear that the indices for incrementing the counter are provided by the color values of a pixels at a particular location. The terms “dr” and “dc” represent an “offset” from the current pixel location and are effectively the “direction” for the neighbor. For practical purposes, any one of the eight connected neighbors to the current pixel is used to compute the co-occurrence matrix. Henceforth, the term co-occurrence matrix will refer to this set of co-occurrence matrices, constructed such that each color channel has a co-occurrence matrix associated with it. Being similar to histograms, various concepts that are applicable to histograms such as histogram intersection and Bhattacharyya coefficients are applicable to co-occurrence matrices as well. With the use of these methods, co-occurrence matrices can be used as a reliable cue for tracking a target.

5.1.1 Spatiograms and co-occurrence matrices

Co-occurrence matrices are similar to spatiograms, since they essentially capture not only range information, but also limited information about the spatial domain. However, the similarity ends there, with spatiograms capturing the global relationship between pixels in a bin of the histogram, while co-occurrence matrices capture pair-wise relationships between pixels. Spatiograms use the global mean and variances of pixels in a bin, while co-occurrence matrices use direct information about a pixel and its neighbor, thus capturing spatial information on a local scale, as compared with spatiograms. Like spatiograms and histograms, the similarity between two co-occurrence matrices may be computed either using histogram intersection or Bhattacharyya coefficient due to the fact that they are similar to both spatiograms and histograms.

5.2 Tracking using co-occurrence matrices

Tracking a target is achieved by the use of co-occurrence matrices in a way similar to the methods introduced in previous sections. Similar to previously introduced modules, the co-occurrence matrix module produces a likelihood map that is converted into a percentage score to facilitate a combination with the framework given by equation 1.3. A model co-occurrence matrix is constructed from a three-quarters view of a person's head after initialization by the user. In addition to initializing the tracker, the direction from which the neighboring pixel is to be chosen is also decided by the user at the start of tracking. Once initialized, the direction from which the neighboring pixel is to be chosen is set as a constant till the end of tracking. As in other modules, the model is also maintained till the end of tracking. The model co-occurrence matrix is a combination of three co-occurrence matrices, each computed for one color channel as shown in equation 5.4. To obtain a likelihood map from the module, target histograms obtained from current frames are compared with the model histogram obtained from the first frame. A measure of similarity may be obtained either using histogram intersection,

$$\phi(s) = \frac{\min(n_b, n'_b)}{\sum_{j=1}^{B} n_b} \tag{5.5}$$

or using Bhattacharyya coefficient as,

$$\phi(s) = \frac{\sqrt{n_b n'_b}}{\sqrt{\left(\sum_{j=1}^{B} n_j\right)\left(\sum_{j=1}^{B} n'_j\right)}}. \tag{5.6}$$

where n_b represents the bins of the model co-occurrence matrix C and n'_b represents the bins of the target co-occurrence matrix C'. As in the case of color histograms, texture histograms and spatiograms, normalization of the target co-occurrence matrix is crucial to

obtaining a correct measure of similarity.The target histograms are normalized as shown

$$\hat{n'_b} = n'_b \frac{\sum_{b=1}^{N} n_b}{\sum_{b=1}^{N} n'_b} \tag{5.7}$$

Once a likelihood measure is obtained, it is then converted into a percentage score, as shown in equation 5.8, to enable an easy combination with either the intensity gradient module or other modules that could be used in the framework suggested in equation 1.3.

$$\bar{\Phi_s}(S) = \frac{\phi_s(S) - min_{S_i \in S}\Phi_s(S_i)}{max_{S_i \in S}\Phi_s(S_i) - min_{S_i \in S}\Phi_s(S_i)} \tag{5.8}$$

where $\Phi_s(S)$ represents the likelihood obtained at the state S from which the target histogram was obtained. This normalized likelihood is computed to enable a simple and easy combination with the intensity gradient module in the framework given by equation 1.3.

Chapter 6

Experimental results

6.1 Log-Gabor histograms

To test the efficiency of Log-Gabor histograms, the log-Gabor histogram tracker was run on two sequences. To provide a reliable method of comparison, the results obtained were compared with the color histogram based tracker and also with manually marked ground truth.

6.1.1 Sequence 1

The log-Gabor sequence was run on the first sequence using an exhaustive search method with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The color histogram module was also run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale to provide a fair comparison. Both trackers were manually initialized at the same point at the start of tracking to ensure impartiality and were run along with the intensity gradient module. Figure 6.2 shows the absolute error computed for every tenth frame using manually computed ground truth for log-Gabor histograms and color histograms. Table 6.1 gives the mean error in x and y calculated using manually determined ground truth. Figure 6.1 shows the tracking results of log-Gabor histograms (red ellipse) compared with tracking results of color histograms

Figure 6.1: Tracking results for color histograms (blue) vs log-Gabor histograms (red) for Sequence 1. Shown are frames 95, 99, 125, 128, 269, 400, 411, 418.

(blue ellipse). The log-Gabor histogram module is not distracted by skin-colored background, tracking the target successfully while color histograms fail. However, the presence of background clutter distracts the log-Gabor histogram module from the target and it fails to track the target when it moves into a cluttered background.

6.1.2 Sequence 2

For the second sequence, the log-Gabor histograms were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.4. The log-Gabor histogram successfully tracks the target for most of the sequence but is distracted by the presence of another person in the background, as seen in frame 25. Figure 6.3 shows the absolute error computed for every tenth frame using manually computed ground truth for log-Gabor histograms and color histograms.

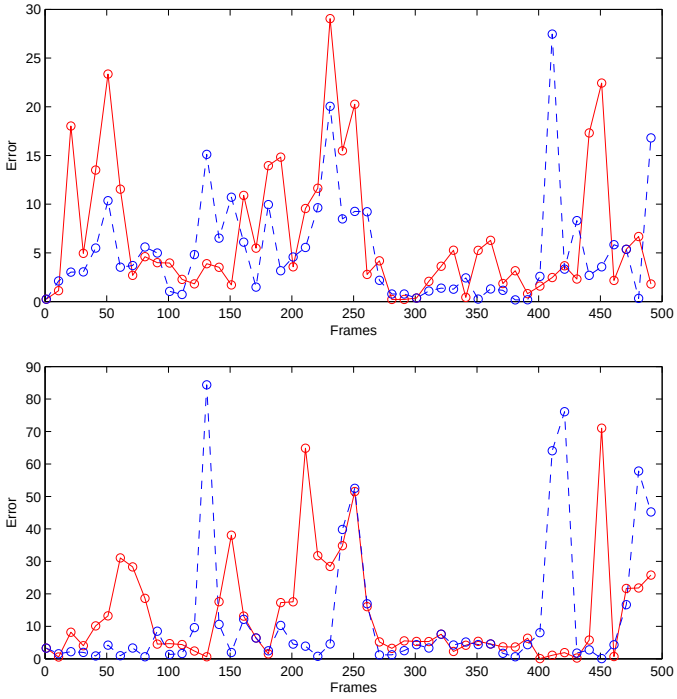

Figure 6.2: Tracking error in x and y using histograms (blue, dashed) versus log-Gabor histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1.

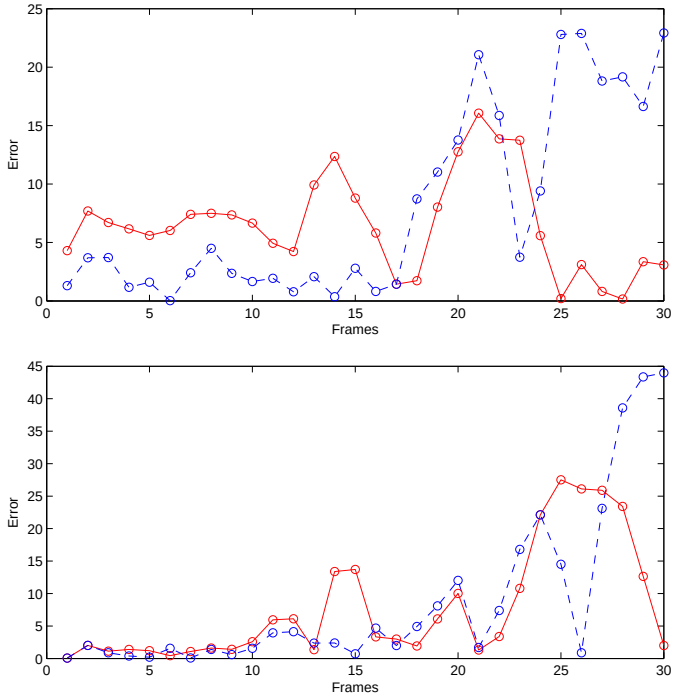

Figure 6.3: Tracking error in x and y using histograms (blue, dashed) versus log-Gabor histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2.

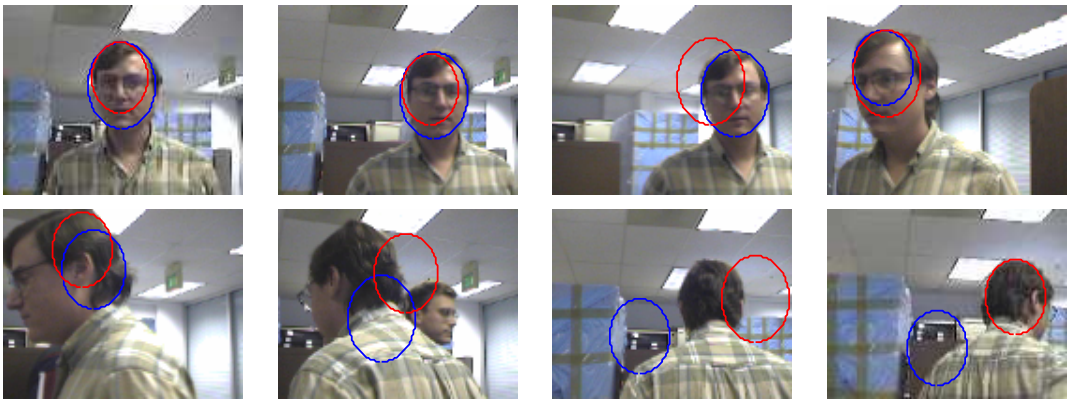

Figure 6.4: Tracking results for color histograms (blue) vs log-Gabor histograms (red) for Sequence 2. Shown are frames 5, 11, 15, 19, 23, 25, 28, 30.

6.2 Haar histograms

6.2.1 Sequence 1

For Sequence 1, Haar histograms were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.5. Similar to log-Gabor histograms, the Haar histogram module successfully tracks the target when it moves into a skin-colored background, but is distracted by a cluttered background and fails to track the target, as seen in frame 146. Figure 6.6 shows the absolute error computed for every tenth frame using manually computed ground truth for Haar histograms and color histograms.

6.2.2 Sequence 2

For Sequence 2, Haar histograms were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.7.

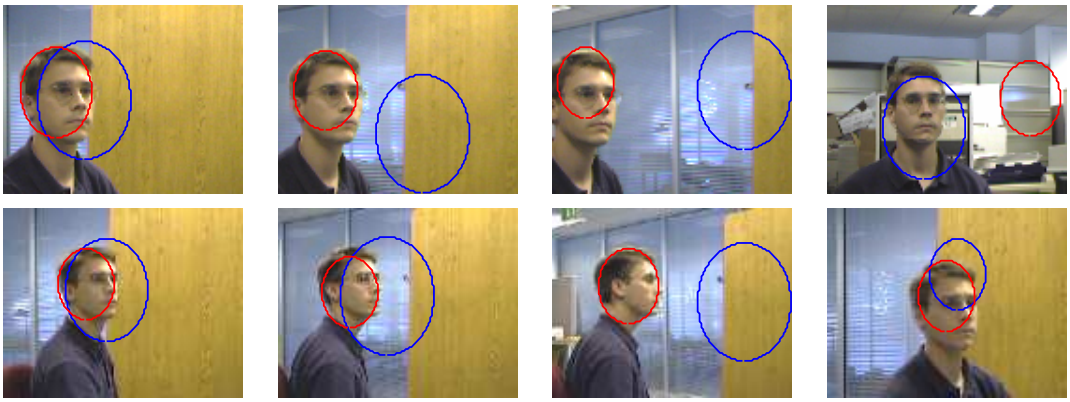

Figure 6.5: Tracking results for color histograms (blue) vs Haar histograms (red) for Sequence 1. Shown are frames 122, 125, 129, 146, 400, 407, 413, 465.

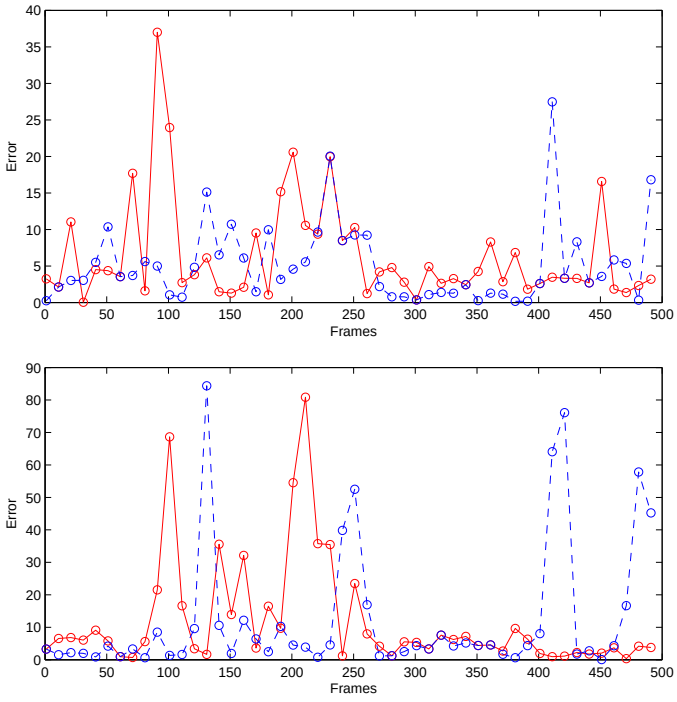

Figure 6.6: Tracking error in x and y using color histograms (blue, dashed) versus Haar histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1.

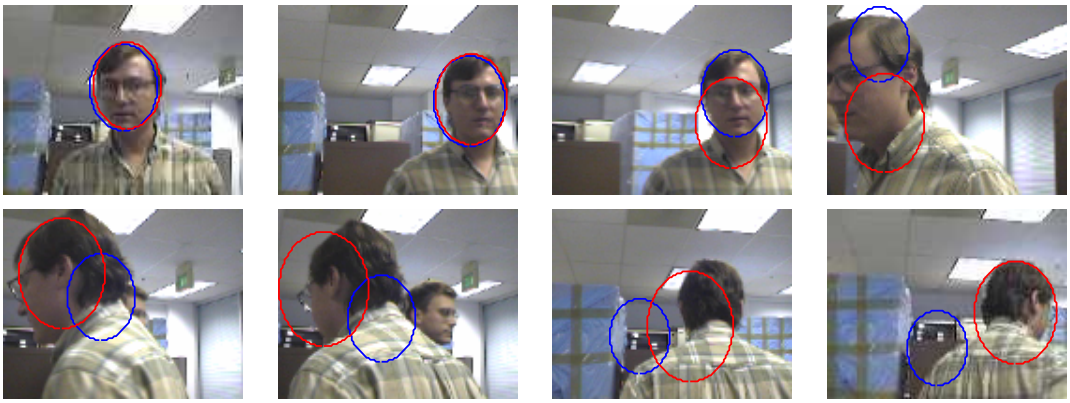

Figure 6.7: Tracking results for color histograms (blue) vs Haar histograms (red) for Sequence 2. Shown are frames 1, 12, 15, 21, 24, 25, 28, 30.

The Haar histogram module successfully tracks the target throughout the sequence and is not distracted by the presence of another person in the background. Figure 6.8 shows the absolute error computed for every tenth frame using manually computed ground truth for Haar histograms and color histograms.

6.3 Edge-orientation histograms

6.3.1 Sequence 1

For Sequence 1, edge-orientation histograms were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.10. While successfully tracking the target under a skin-colored background, edge-orientation histograms are also distracted by the presence of clutter in the background, as seen in frame 168. As the target continues to move in a cluttered background, edge-orientation histograms lose track of the target, as seen in frame 200. Figure 6.9 shows the absolute error computed for every tenth frame using manually computed ground truth for edge-orientation histograms and color histograms.

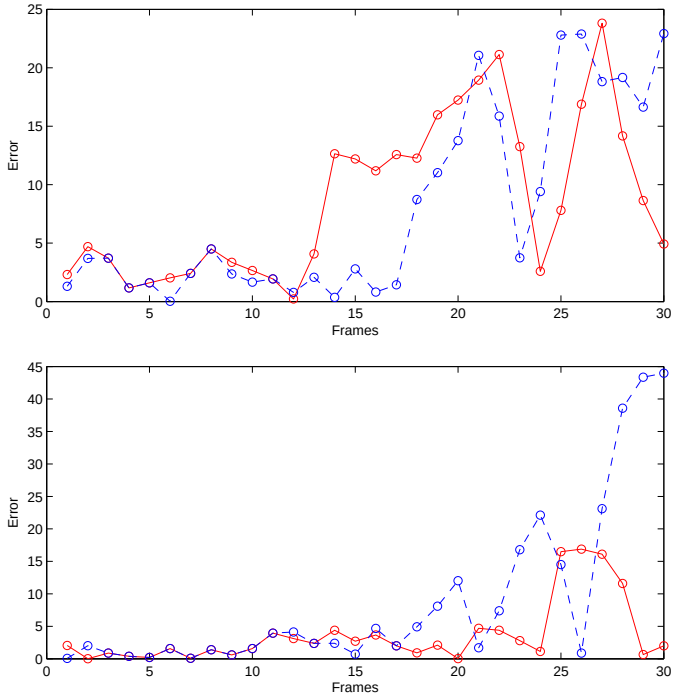

Figure 6.8: Tracking error in *x* and *y* using histograms (blue, dashed) versus Haar histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2.

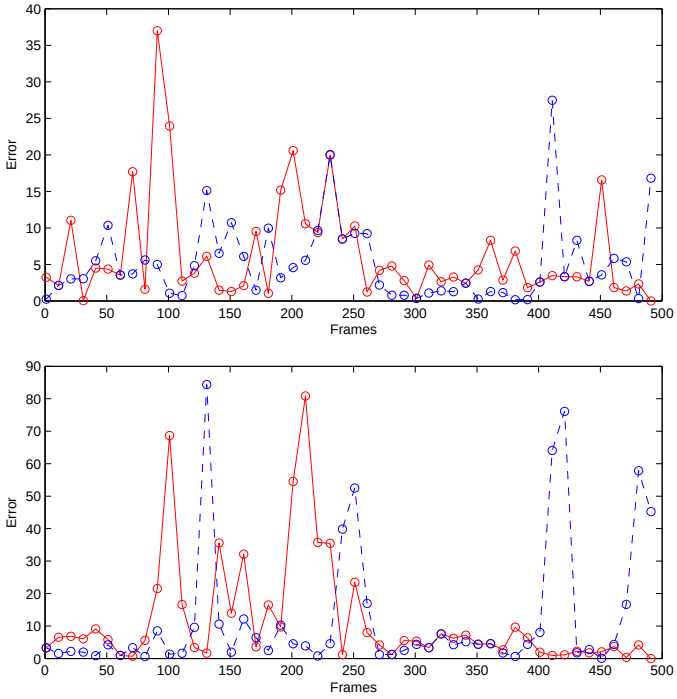

Figure 6.9: Tracking error in *x* and *y* using color histograms (blue, dashed) versus edge-orientation histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1.

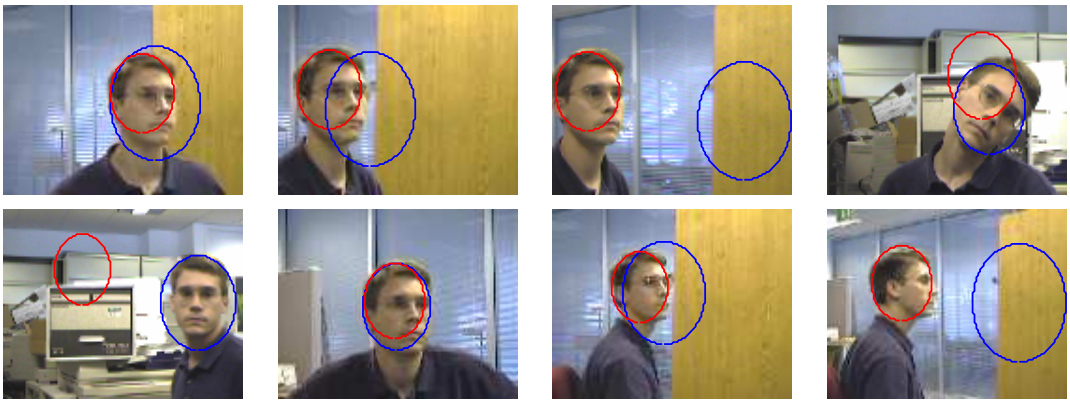

Figure 6.10: Tracking results for color histograms (blue) vs edge-orientation histograms (red) for Sequence 1. Shown are frames 99, 123, 127, 168, 200, 311, 403, 413.

6.3.2 Sequence 2

For Sequence 2, edge-orientation histograms were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.12. From frame 30, it can be clearly seen that the edge-orientation histograms are distracted by objects causing clutter in the background, thus casuing the tracker to fail. Figure 6.11 shows the absolute error computed for every tenth frame using manually computed ground truth for edge-orientation histograms and color histograms.

6.4 Spatiograms

6.4.1 Sequence 1

For Sequence 1, spatiograms were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.14. From the frames shown, it can be observed that spatiograms successfully track the target and are not

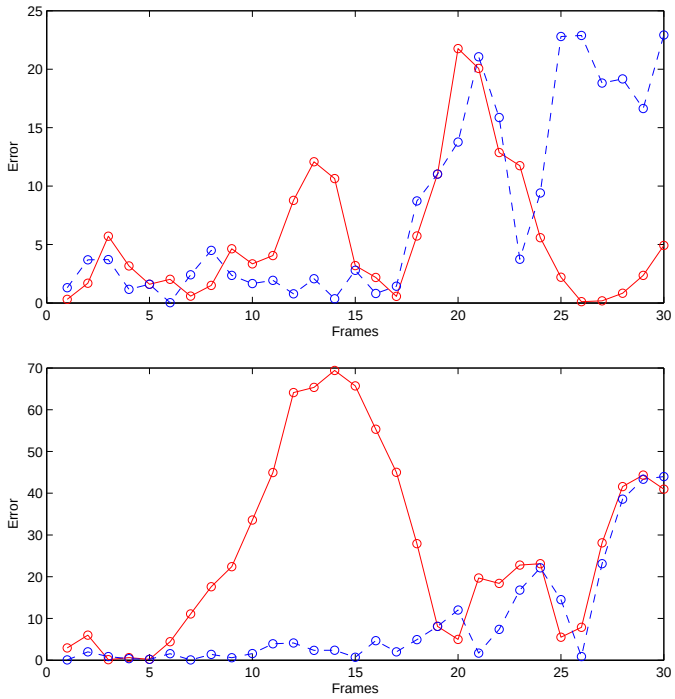

Figure 6.11: Tracking error in *x* and *y* using color histograms (blue, dashed) versus edge-orientation histograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2.

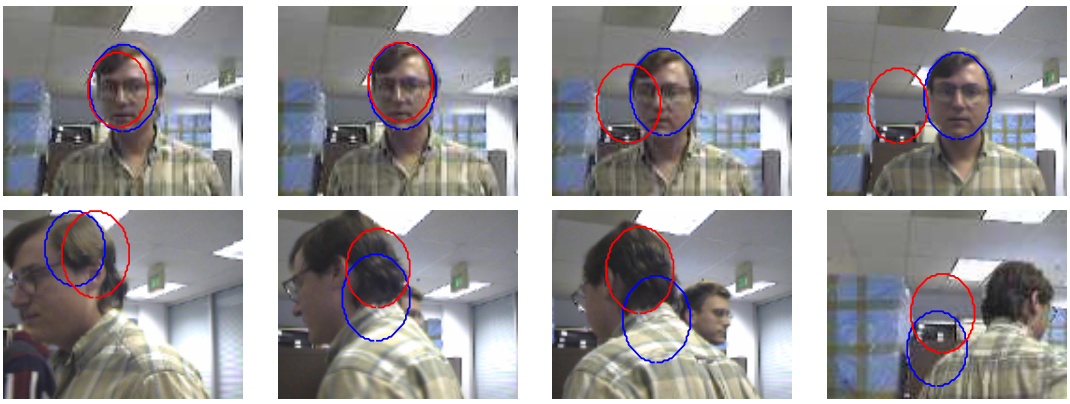

Figure 6.12: Tracking results for color histograms(blue) vs edge-orientation histograms(red) for Sequence 2. Shown are frames 1, 3, 8, 10, 22, 24, 25, 30.

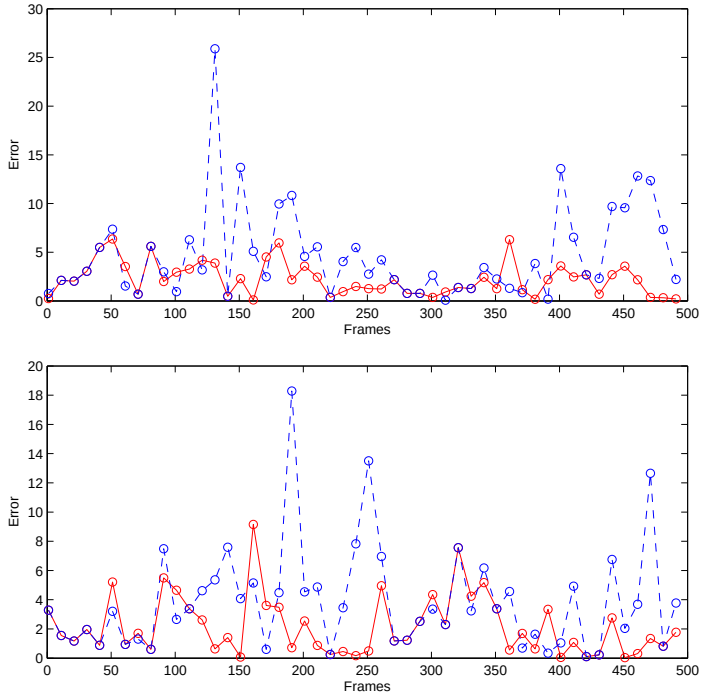

Figure 6.13: Tracking error in *x* and *y* using color histograms (blue, dashed) versus co-occurrence matrices (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1.

distracted by skin-colored background or a cluttered background. Figure 6.13 shows the absolute error computed for every tenth frame using manually computed ground truth for spatiograms and color histograms.

6.4.2 Sequence 2

For Sequence 2, spatiograms were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.16. Frames 25-30 show that the spatiogram module successfully tracks the target under a cluttered background, which undergoes a sudden motion causing color histograms to fail. Figure 6.15 shows the absolute error computed for every tenth frame using manually computed ground truth for spatiograms and color histograms.

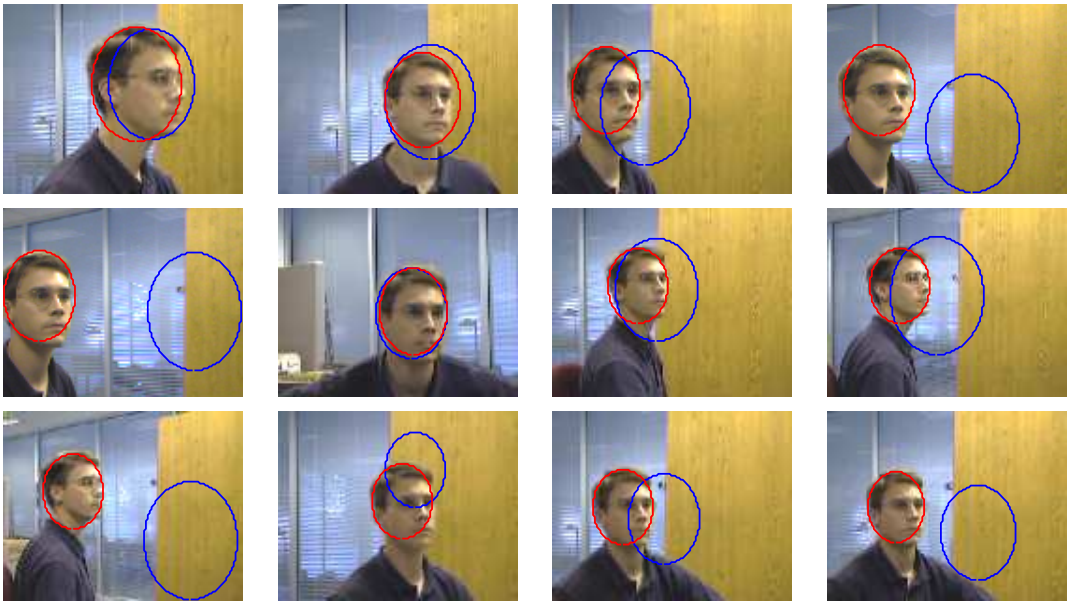

Figure 6.14: Tracking results for color histograms (blue) vs spatiograms (red) for Sequence 1. Shown are frames 93, 99, 123, 125, 128, 367, 400, 407, 411, 463, 472, 474.

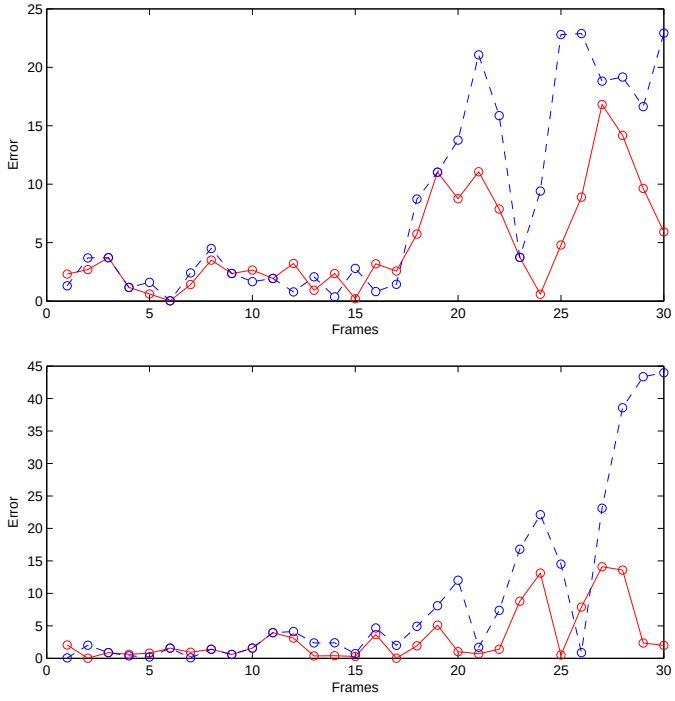

Figure 6.15: Tracking error in x and y using color histograms (blue, dashed) versus spatiograms (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2.

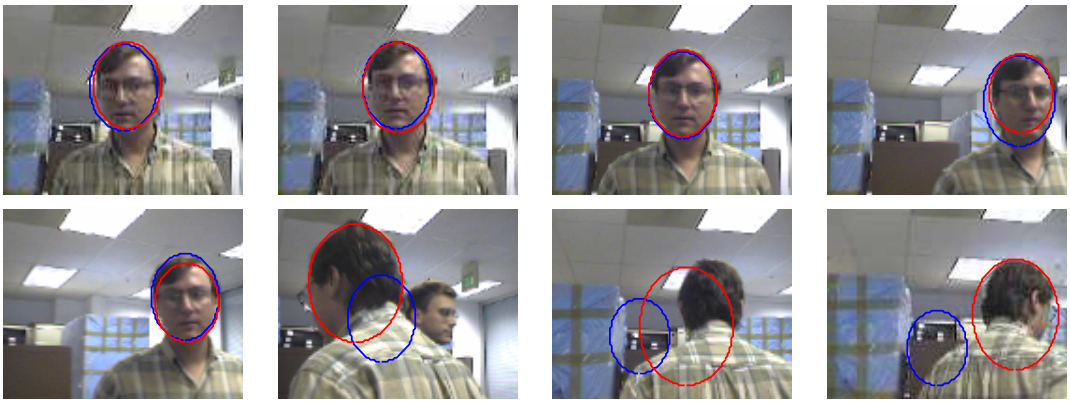

Figure 6.16: Tracking results for color histograms (blue) vs spatiograms (red) for Sequence 2. Shown are frames 1, 5, 10, 12, 15, 25, 28, 30.

6.5 Co-occurrence matrices

6.5.1 Sequence 1

For Sequence 1, color co-occurrence matrices were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.18. Frames 123 and 143 show that co-occurrence matrices are able to successfully track the target under different backgrounds, similar to spatiograms. Figure 6.17 shows the absolute error computed for every tenth frame using manually computed ground truth for color co-occurrence and color histograms.

6.5.2 Sequence 2

For Sequence 2, color co-occurrence matrices were run with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale along with the intensity gradient module. The color histogram module was also run on the same sequence, combined with the intensity gradient module, with a $\pm 6 \times \pm 6 \times \pm 1$ search window in *x*, *y*, and scale. The tracking results are shown in figure 6.20. Similar to the previous sequence, co-occurrence matrices successfully track

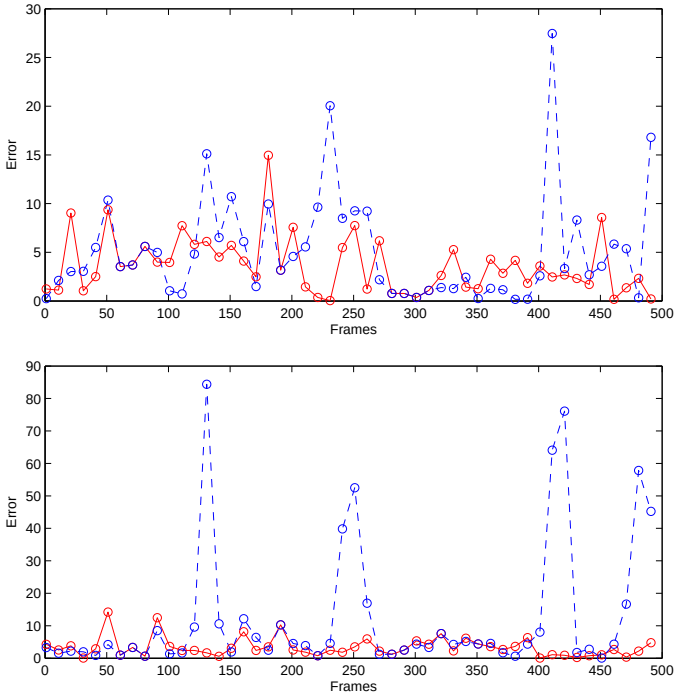

Figure 6.17: Tracking error in *x* and *y* using histograms (blue, dashed) versus co-occurrence matrices (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 1.

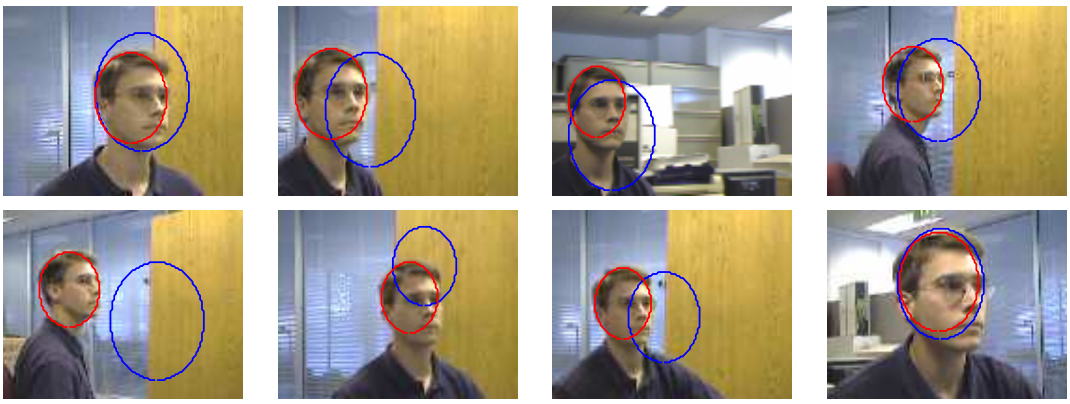

Figure 6.18: Tracking results for color histograms (blue) Vs co-occurrence matrices (red) for Sequence 1. Shown are frames 97, 123, 143, 404, 410, 462, 472.

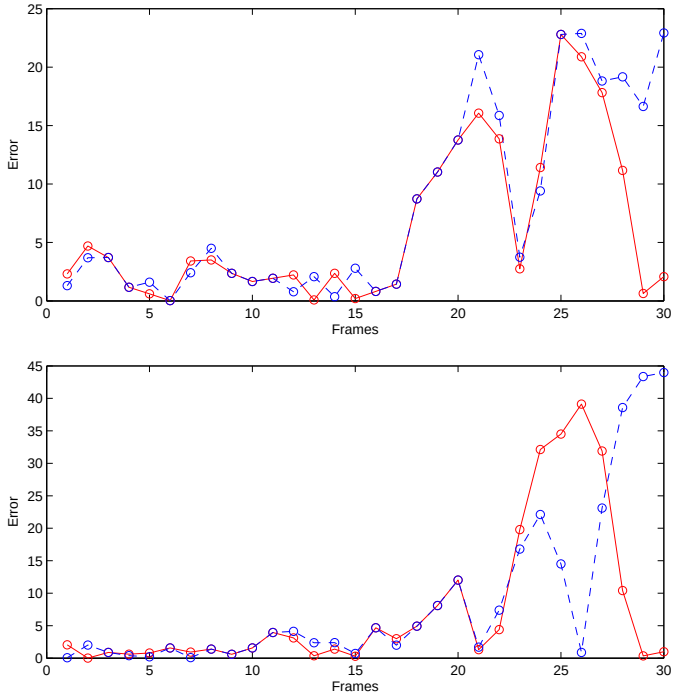

Figure 6.19: Tracking error in *x* and *y* using color histograms (blue, dashed) versus co-occurrence matrices (red, solid) with exhaustive local search using a search window size of $6 \times 6 \times 1$ in x,y and scale respectively for Sequence 2.

the target even when the target undergoes sudden motion. Figure 6.19 shows the absolute error computed for every tenth frame using manually computed ground truth for color co-occurrence and color histograms.

Algorithm	Mean Error in	
	x	**y**
Color histograms	7.58	24.03
Log-Gabor histograms	9.76	21.33
Haar histograms	9.71	21.07
Edge-orientation histograms	6.36	17.91
Color co-occurrence matrices	4.74	4.52
Color spatiograms	**4.33**	**4.37**

Table 6.1: Mean errors in tracking in x and y for Sequence 1.

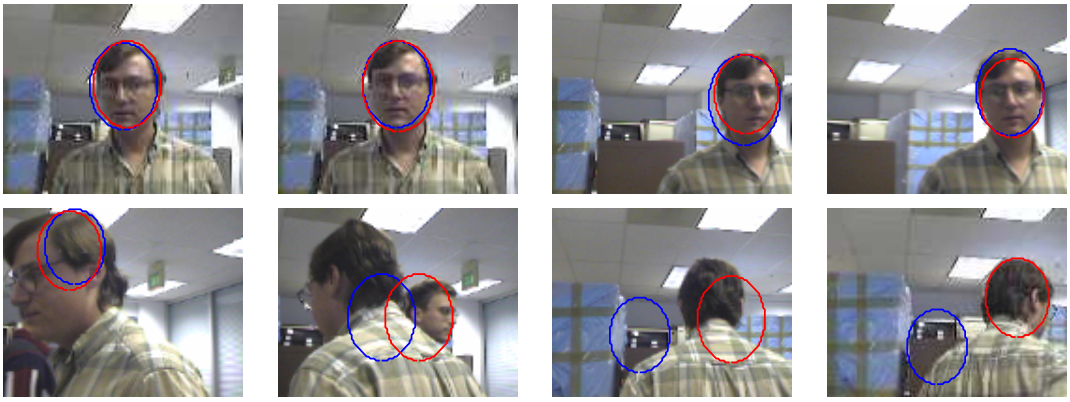

Figure 6.20: Tracking results for color histograms (blue) vs co-occurrence matrices (red) for Sequence 2. Shown are frames 1, 5, 12, 15, 22, 25, 28, 30.

Algorithm	Mean Error in	
	x	**y**
Color histograms	11.31	15.48
Log-Gabor histograms	7.69	11.64
Haar histograms	10.97	6.05
Edge-orientation histograms	7.87	34.47
Color co-occurrence matrices	9.07	13.66
Color spatiograms	**6.42**	**5.14**

Table 6.2: Mean errors in tracking in x and y for Sequence 2.

Chapter 7

Conclusion

Various algorithms have been put forth to make use of spatial information in the recent past. However, the use of such information comes with its own disadvantages and overcoming them reduces the computational efficiency of the algorithm using the information and makes the algorithm more complex. Hence, complete reliance on spatial information provides a disadvantage when compared to algorithms that make use of color information only. The work in this thesis shows that a combination of spatial and color information produces robust results without sacrificing computational efficiency, while also retaining the simplicity of algorithms that use only color information.

From the results of experimental comparisons done in section 6, it can be seen that the use of a limited amount of spatial information combined with color information greatly improves tracking results. Such an approach easily outperforms other approaches that rely on using either only color information or only spatial information and evidence for such a conclusion can be obtained from Tables 6.1 and 6.2. A combination of spatial and color information makes a tracker robust and provides the tracker with advantages that rise out of using each method separately, which is shown by the successful results obtained from using color spatiograms and color co-occurrence matrices when compared with color histograms, edge-orientation histograms, log-Gabor histograms and Haar histograms.

Bibliography

[1] Stan Birchfield. An elliptical head tracker. In *Proceedings of the 31st Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1710–1714, 1997.

[2] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 232–237, 1998.

[3] Stanley T. Birchfield and Sriram Rangarajan. Spatiograms versus histograms for region-based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1158–1163, 2005.

[4] L. Brown. 3D head tracking using motion adaptive texture-mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 998–1005, 2001.

[5] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[6] Robert T. Collins. Mean-shift blob tracking through scale space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[7] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.

[8] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 142–149, 2000.

[9] L. Davis, S. Johns, and J.K Aggarwal. Texture analysis using generalized co-occurrence matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):251–259, 1979.

[10] Ahmed Elgammal, Ramani Duraiswami, and Larry S. Davis. Probabilistic tracking in joint feature-spatial spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[11] Paul Fieguth and Demetri Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–27, 1997.

[12] William T. Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pages 296–301, 1995.

[13] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[14] H. P. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan. Multi-modal system for locating heads and faces. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 88–93, 1996.

[15] G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with SSD. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[16] Jeffrey Ho, Kuang-Chih Lee, Ming-Hsuan Yang, and David Kriegman. Visual tracking using learned subspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 782–789, 2004.

[17] Jing Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[18] Martin Hunke and Alex Waibel. Face locating and tracking for human-computer interaction. In *Proceedings of the 28th Asilomar Conference on Signals, Systems and Computers*, pages 1277–1281, 1994.

[19] Allan D. Jepson, David J. Fleet, and Thomas F. El-Maraghi. Robust online appearance models for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 415–422, 2001.

[20] Stephen J. McKenna, Yogesh Raja, and Shaogang Gong. Object tracking using adaptive colour mixture models. In *Proceedings of the 3rd Asian Conference on Computer Vision*, 1998.

[21] Chahab Nastar and Matthias Mitschke. Real-time face recognition using feature combination. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 312–317, 1995.

[22] Yossi Rubner and Carlo Tomasi. Texture-based image retrieval without segmentation. In *Proceedings of the International Conference on Computer Vision*, pages 1018–1024, 1999.

[23] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[24] Lronid Sigal, Stan Claroff, and Vassilis Athitsos. Estimation and prediction of evolving color distributions for skin segmentation under varying illumination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[25] John R. Smith and Shih-Fu Chang. Automated image retrieval using color and texture. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[26] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[27] Ying Wu and Thomas S. Huang. A co-inference approach to robust visual tracking. In *Proceedings of the 8th International Conference on Computer Vision*, pages 26–33, 2001.

[28] Zoran Zivkovic and Ben Kröse. An EM-like algorithm for color-histogram-based object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.