VISUAL OBJECTS AND ENVIRONMENTS: CAPTURE, EXTRACTION, AND REPRESENTATION

BY

KAR-HAN TAN

B.S., National University of Singapore, 1994
M.S., University of California at Los Angeles, 1996

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

To Sandra

Acknowledgements

I would like to thank my advisor Professor Narendra Ahuja for sharing his wisdom with me, and the members of the Beckman Institute Computer Vision and Robotics Lab for creating the stimulating and supportive environment in the group. I am also very grateful to Professors David Kriegman, Michael Garland, and Yizhou Yu for their suggestions and criticisms that helped me tremendously in many ways. I owe a great debt of thanks to my fiancee Sandra, for being my biggest fan and primary source of inspiration. Lastly, my deepest gratitute goes to my parents and two sisters, for their unwavering and unconditional support that gave me the courage to embark on this journey.

Table of Contents

List of Figures

Chapter 1

Introduction

The availability of affordable computational power and graphics rendering capabilities is enabling the creation of realistic imagery that are widely used today in special effects and animation. New forms of visual media with a higher degree of interactivity have also emerged as a result of these technological advances. For example, virtual reality can be considered a form of visual media that is controlled by the position of the viewer to give a sense of immersion. Video games are another form where the viewer can give feedback through input devices like buttons and joysticks. Some of the most popular video games can be considered interactive movies, where the player gets to control the events in action sequences and sometimes determine the story line, resulting in a more engaging experience than with traditional media.

While there are many factors that contribute to the effectiveness of a VR or video game presentation, one of the most important is visual realism, and the use of images as textures is often the key. Figure 1.1 shows a screenshot taken from a popular video game that is rendered predominantly with texture mapped 3D models. The main character and his vehicle are fully articulated, detailed models. The mountains and ground are static objects, while the distant backdrop is a panoramic image. Figure 1.2 shows a screenshot of a second game that makes more extensive use of images: except for the main characters, the entire background is a prerendered image. While the first game allows dynamic camera movement, the second relies on fixed cameras to give the illusion that the 3D characters are interacting

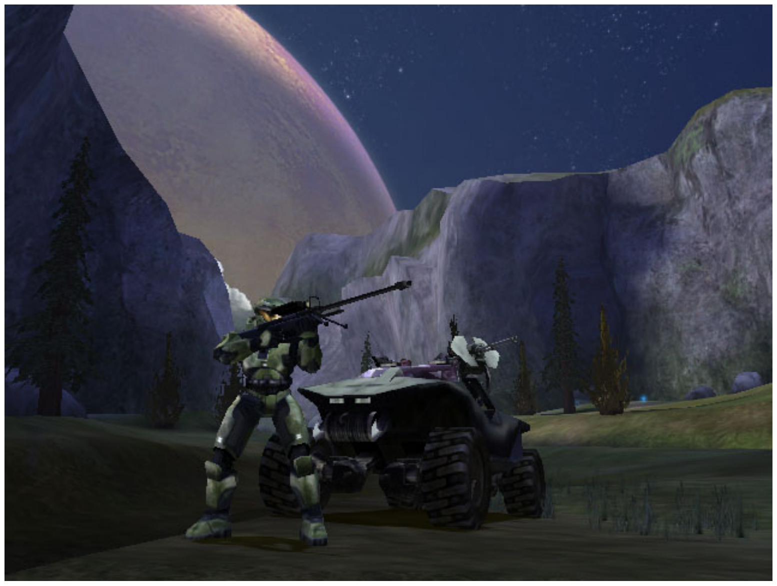


Figure 1.1: A typical scene in a video game rendered with texture-mapped 3D models. (Scene taken from *Halo*, image courtesy of Bungie Studios.)

with the environment. On the other hand, the second game clearly is richer in visual detail and is more realistic. These examples illustrate a tradeoff between the two techniques: while image-based methods allows a higher level of visual realism and detail, freedom in viewing angle placement is lost.

Researchers in the computer graphics community have attempted to bridge this gap by introducing image-based rendering methods that essentially capture all possible views of objects with a large collection of images and retrieves the appropriate frames for display according to viewing angle. Typically this requires the object being modeled to be placed on a specialized apparatus with a large array of cameras. Great success has been achieved with these methods when modeling rigid objects that can be placed on turntables. The capture of complex, articulated, and moving objects (such as human beings) in an image-based representation remains a challenge. Sometimes the object is simply unavailable for careful setup and modeling. For example, it may be desirable to capture an athlete's moves during a special moment in a game that cannot be reproduced afterwards, and the only source of data could be video footage.

Figure 1.2: A typical scene in a video game rendered predominantly with image-based techniques. (Scene taken from *Onimusha 1 and 2*, images courtesy of Capcom.)

1.1 Thesis Overview

Ultimately, we would like to be able to capture a dynamic scene in an image-based representation, and allow reproduction of the scene in a realistic fashion, allowing a higher degree of freedom in view point placement than is allowed with current methods. In this thesis, we present research work that attempts to address some of the challenges and make progress towards the ultimate goal. In particular, we show how objects can be extracted from images and video streams, and how a novel camera can be constructed to capture dynamic environments in the form of a panoramic video stream.

1.1.1 Extracting Visual Objects

Object extraction from images and video streams is an area that at first glance has seen very little research work done. However the fields of computer graphics and computer vision are actually rich sources of ideas and inspiration for the problem. In subsequent chapters we will examine a number of subjects and work towards an algorithm that integrates many of these ideas.

Low Level Boundary Extraction with Fast Probabilistic Matting In this chapter we examine a low level algorithm for boundary extraction, and a number of existing algorithms that address this problem. We then describe a new algorithm we call *Fast Probabilistic Matting* that produces results comparable with the existing methods, but uses a closed-form solution that runs much faster than the existing exhaustive search and iterative solutions.

Interactive Object Extraction from Images In chapter 3, we present a method for extracting objects with complex and diffused boundaries from single images. The algorithm uses an alpha channel estimation algorithm to allow a user to 'select' an object from an image.

1.1.2 Representing Visual Objects

Appearance-based Object Modeling and Tracking One of the most important issues in object extraction from video is that an object may appear completely different when viewed from a different direction. In this chapter we study methods for modeling the appearance of objects under varying pose. We propose a new representation called *faceted models* and tested it in an appearance-based object tracking algorithm,

Appearance-based Eye Gaze Estimation Using faceted models, we can also estimated object pose when a sufficiently dense set of samples is available. The pose estimation capability is tested in an appearance-based eye gaze estimation algorithm.

1.1.3 Capturing Visual Environments

We also present a method for capturing the environment using a novel camera that is capable of simultaneously capturing multiple panoramic video streams each taken from different viewpoints. The camera can be configured to give stereoscopic (two-view) video streams and in theory is capable of an unlimited number of view points. The captured panoramic images can then be used as environmental backdrops in a 3D scene.

Chapter 2

Probabilistic Alpha Channel Estimation for Low Level Object Boundary Extraction

The *alpha channel*, usually represented as a fourth component in each pixel of an image alongside the RGB channels, was first introduced in [80] as a construct for compositing multiple independently rendered computer-generated visual objects into a single image. Today the alpha channel is widely used in computer graphics, film and video production for fusing both synthetic and conventional visual digital media content. As it is a digital substitute for *mattes* [87] in film and video production, extracting an alpha channel is sometimes referred to as *matting*.

Fundamentally, an alpha channel captures the idea that the color value in an image pixel may in general be the result of mixing the colors from multiple objects. This can arise due to various factors commonly found in image capture and synthesis scenarios such as transparency, the finite resolution of imaging sensors, optical defocus, and camera-object relative motion during image formation, and antialiasing, among others. In order to naturally extract an object from an image, say to perform a cut-and-paste operation, it is very important to take into account the color blending across object boundaries, and attempt to recover the pure colors for the object extracted. This problem of estimating the pure object and background colors, and the mixing coefficient is called the *alpha channel estimation* problem [83].

The basic equation for alpha channel compositing is as follows:

$$C = \alpha F + (1 - \alpha)B$$

where F represents the pure foreground color, and B represents the pure background color. C represents an observed color value that is formed by a linear blending of F and B, controlled by the alpha channel value α. The alpha channel estimation problem can then be stated as follows:

> We are given an image, with pixels that are labeled as pure foreground, pure background, and mixed. The alpha values for the foreground pixels are given to be 1, and those for the background pixels are given to be 0. Compute alpha channel values for the rest of the pixels.

As stated, the problem is inherently underconstrained. For each observation C, we need to find its corresponding B and F, and the blending factor α. Geometrically, in a three dimensional color space, for a given color C, the problem is that of finding a straight line segment FB that passes through C. The ratio $\frac{BC}{BF}$ is equal to the value α.

2.1 Previous Work on Alpha Channel Estimation

Blue screen matting can be considered a special case of the alpha channel estimation problem, where the background is assumed to be of a single color. An excellent summary of conventional blue screen matting algorithms can be found in [87]. [111] refers to the general alpha channel estimation problem as *Natural image matting*, and has a good survey of some of the representative techniques, including algorithms used in existing commercial products such as KnockOut and Primatte. Recently, [44] proposed a PCA-based method for use in high-resolution images and video sequences. While the solutions discussed thus far are predominantly geometric in nature, the most promising solutions to the problem adopt a probabilistic framework. Currently there are two known probabilistic methods: [83] models the ‘pure’ foreground and background colors with gaussian distributions, and assumes that a mixed color is drawn from gaussians created by interpolating the parameters of the pure distributions. It was proposed that the alpha parameter be estimated by an exhaustive search. [111] formulates the problem in a maximum-likelihood framework and proposes an iterative solution. [105] uses a similar framework for extracting layers from video sequences.

While both [83, 111] produced impressive results, they are computationally intensive. The exhaustive search in [83] makes it especially expensive to compute. [111] is more economical, but still takes several minues to process a one-megapixel image. In both cases there is also an undesirable tradeoff between quality and computation time. In [83], the computation time is inversely proportional to the discretization step size. As a result, cutting down on computation time by increasing the step size results in a graininess in the estimated alpha channel (As observed by the authors of [111]). In [111], a noise variance parameter determines how close the estimated line in color space is to the mixed pixel color. The smaller the noise level allowed, the longer it will take the iteration to converge. With a large noise variance, however, the estimated pure foreground and background colors do not interpolate to faithfully reproduce the original mixed pixel color.

Given that one of the key application areas of alpha channel estimation is in video production, fast real time implementations that produce high-quality mattes are highly desirable. This is our motivation for investigating faster solutions to the problem without compromising the quality of the estimated result.

2.2 Fast Probabilistic Alpha Channel Estimation

In this section, we present our algorithm which solves the maximum-likelihood estimation problem in closed form. Subsequent sections describe experiments that evaluates the performance of the proposed algorithm and how it compares to existing algorithms.

2.2.1 Probabilistic Color Modeling

Like [83, 111] we assume that we are given samples of ‘pure’ foreground and background pixel colors, and a set of ‘mixed’ pixels for which the alpha channel values are to be estimated. We first group the pixel colors in the foreground and background samples into a number of clusters. For example, a foreground object that is red-and-blue could be represented by a red cluster and a blue cluster. Each color cluster is modeled with a multivariate Gaussian distribution characterized by its centroid μ and covariance matrix Σ. An observed mixed color C is formed by linear blending of colors F and B drawn from respective clusters with centroids μ_1, μ_2 and covariance matrices Σ_1, Σ_2. The blending is represented by a line through C (Point P in Figure 2.1). Different points along the line correspond to different mixes of F and B which are themselves points on the line at minimum Mahalanobis distance to μ_1 and μ_2 respectively. The relative distance of P from F and B will then yield the alpha value.

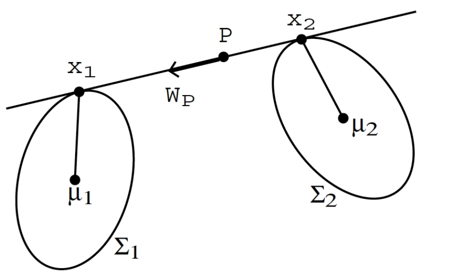
2.2.2 A Closed-form Solution

We start by revisiting some basic facts about Mahalanobis distances and covariance matrices. It is well known that covariance matrices, being positive semidefinite, can be written in the form $\Sigma = Q\Lambda Q^T$ where Q is orthonormal and Λ is diagonal. Also $\Sigma^{-1} = Q\Lambda^{-1}Q^T = Q\Lambda^{-\frac{1}{2}}\Lambda^{-\frac{1}{2}}Q^T$. The squared Mahalanobis distance (also known commonly as the covariance-weighted distance) between two points x, y is defined as

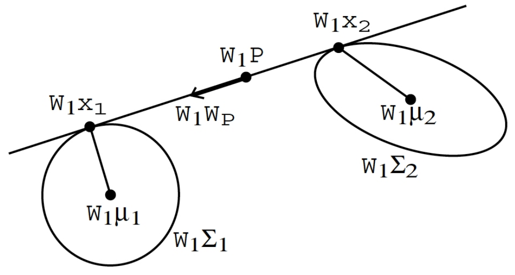
$$distance_{mahalanobis} = (x-y)^T\Sigma^{-1}(x-y)$$

If we write it as

$$\begin{aligned} &(x-y)^T Q\Lambda^{-\frac{1}{2}}\Lambda^{-\frac{1}{2}}Q^T(x-y) \\ &= (\Lambda^{-\frac{1}{2}}Q^T(x-y))^T(\Lambda^{-\frac{1}{2}}Q^T(x-y)) \end{aligned}$$



(a)



(b)

Figure 2.1: Mapping onto the covariance-weighted space. (a) Original geometric entities. (b) Mapped into covariance-weighted space. Notice that the isocontour for cluster 1 is now a circle.

We can see that the term $\Lambda^{-\frac{1}{2}}Q^T$ can be thought of as a transformation that maps a vector into a covariance-weighted space in which the spectrum of eigenvalues is uniform, and the euclidean distance is equal to the mahalanobis distance. Figure 2.1 illustrates the geometric relationship between the entities involved in the following derivation. As the derivation is largely independent of the problem domain, we chose to use a generic set of variable names that are easier to remember. In the diagram, we have two clusters represented by the two centroids μ_1, μ_2 and the two covariance matrices Σ_1, Σ_2. We want to find a line passing through the point P that minimizes the Mahalanobis distance from each of the centroids to the line. The points x_1, x_2 are the points on the line that are nearest to the respective centroids. Note also that the distance between the centroid μ_1 and the point x_1 is weighted by Σ_1, and the distance between the centroid μ_2 and the point x_2 is weighted by Σ_2. Let the vector W_p be parallel to the line x_1x_2.

The first observation about the relationship between the line and each of the clusters is that at the points x_1 and x_2, the line is tangential to the respective isocontours for the minimum distance from the two clusters. That is to say if point x_1 is at a (covariance-weighted) l units away from μ_1, then the line is tangential to the isocontour of points at a distance l units away from μ_1. If the line cuts the distance-l isocontour twice, it means that there is a point on the line that touches another isocontour at distance l', such that $l' < l$. If the line does not touch the isocontour at distance l, it means that all points on the line are at a distance greater than l from μ_1, and there must be another isocontour at distance $l'' > l$ that is tangential to the line.

In order to express this constraint, we make a second observation that the isocontours for a cluster, that are in general ellipsoids, become spheres when transformed in to the covariance-weighted space. If we transform both the vector W_p and the centroid μ_1 into the space weighted by the covariance matrix Σ_1, the isocontours of cluster 1 become spheres, and the vector $(x_1 - \mu_1)$, now a radius vector, should be perpendicular to the transformed vector W_p. If we let $W_1 = \Lambda_1^{-\frac{1}{2}} Q^T$, and $\Sigma_1 = W_1^T W_1$, we can write down the constraints for the first cluster as

$$(W_1 W_p)^T (W_1 (x_1 - \mu_1)) = 0,$$

and thus

$$W_p^T \Sigma_1^{-1} (x_1 - \mu_1) = 0 \tag{2.1}$$

and similarly for the second cluster

$$W_p^T \Sigma_2^{-1} (x_2 - \mu_2) = 0 \tag{2.2}$$

We also know that the points x_1 and P lie along the line represented by vector W_p. Thus

$$W_p = \frac{1}{k_1} (x_1 - p),$$

where k is a constant, so

$$x_1 = P + k_1 W_p \tag{2.3}$$

Substituting equation 2.3 into equation 2.1 to eliminate x_1, we obtain for cluster 1

$$W_p^T \Sigma^{-1}(P + k_1 W_p - \mu_1) = 0$$

$$k_1 W_p^T \Sigma_1^{-1} W_p + W_p^T \Sigma_1^{-1}(P - \mu_1) = 0$$

$$k_1 = \frac{W_p^T \Sigma_1^{-1}(\mu_1 - P)}{W_p^T \Sigma_1^{-1} W_p} \tag{2.4}$$

and analogously for cluster 2

$$k_2 = \frac{W_p^T \Sigma_2^{-1}(\mu_2 - P)}{W_p^T \Sigma_2^{-1} W_p} \tag{2.5}$$

Now the squared Mahalanobis distances d_1^2 and d_2^2 can be found by applying the pythagoras theorem in the covariance-weighted space, as follows:

$$d_1^2 = (\mu_1 - P)^T \Sigma_1^{-1}(\mu_1 - P) - k_1^2 W_p^T \Sigma_1^{-1} W_p \tag{2.6}$$

$$d_2^2 = (\mu_2 - P)^T \Sigma_2^{-1}(\mu_2 - P) - k_2^2 W_p^T \Sigma_2^{-1} W_p \tag{2.7}$$

To obtain the maximum likelihood estimates of W_p, we would like to minimize $d_1^2 + d_2^2$. Since the $(\mu_i - P)^T \Sigma_i^{-1}(\mu_i - P)$ terms are independent of W_p, we can drop the terms and instead maximize

$$k_1 W_p^T \Sigma_1^{-1} W_p + k_2 W_p^T \Sigma_2^{-1} W_p$$

$$= (\frac{W_p^T \Sigma_1^{-1}(\mu_1 - P)}{W_p^T \Sigma_1^{-1} W_p}) W_p^T \Sigma_1^{-1} W_p +$$

$$(\frac{W_p^T \Sigma_2^{-1}(\mu_2 - P)}{W_p^T \Sigma_2^{-1} W_p}) W_p^T \Sigma_2^{-1} W_p$$

$$= \frac{1}{W_p^T \Sigma^{-1} W_p}[(W_p^T \Sigma_1^{-1}(\mu_1 - P)) + (W_p^T \Sigma_2^{-1}(\mu_2 - P))]$$

Since $W_p^T \Sigma^{-1} W_p$ is a scalar quantity and we can constrain the length of W_p, we only need to maximize

$$(W_p^T v_1) + (W_p^T v_2) \tag{2.8}$$

where

$$v_1 = \Sigma_1^{-1}(\mu_1 - P), v_2 = \Sigma_2^{-1}(\mu_2 - P)$$

The expression derived in (8) is the sum of the weighted distances of P from the centroids μ_1 and μ_2. Suppose the point P is a lot closer to μ_1 than μ_2, implying $|\mu_1 - P| << |\mu_2 - P|$. It makes sense then that d is dominated by the second term, since the line will be close to μ_1 anyway.

We now show how d can be maximized by introducing a new constraint. If we assume that the vector W_p is coplanar with v_1 and v_2, it can be seen that we effectively want to maximize the following

$$|v_1| cos\theta_1 + |v_2| cos(\pi - (\theta + \theta_1))$$

with respect to θ, the angle between v_1 and v_2. Differentiating the expression, we obtain

$$\begin{aligned} &-|v_1| sin\theta_1 + |v_2| sin(\pi - (\theta + \theta_1))) \\ &= -|v_1| sin\theta_1 + |v_2| sin(\theta + \theta_1)) \end{aligned}$$

Setting it to zero to find the stationary point, we obtain

$$\begin{aligned} &0 = -|v_1| sin\theta 1 + |v_2| sin(\theta + \theta_1)), \\ &sin\theta_1 = \tfrac{|v_2|}{|v_1|} sin(\theta + \theta_1), \\ &\tfrac{|v_2|}{|v_1|}(sin\theta cot\theta_1 + cos\theta) = 1, \end{aligned}$$

which gives

$$\theta_1 = cot^{-1}((-\frac{|v_1|}{|v_2|} - cos\theta)\frac{1}{sin\theta})$$

θ is the angle between v_1 and v_2, and can be computed as

$$\theta = cos^{-1}(\frac{v_1^T v_2}{|v_1||v_2|})$$

which gives W_p, and consequently k_1 and k_2, which in turn allows F, B and α to be computed.

2.3 Experiments

We created a prototype of the algorithm in MATLAB, and used it to compute the alpha channel for several images. The results are shown in Figure 2.2 and Figure 2.3. While Chuang's method took minutes to compute, our method took only 1 second for the lighthouse image (480 by 320), and 2.2 seconds for the lion image (576 by 864). As can be seen, the algorithm was able to estimate the alpha channel and extract the foreground object correctly.

2.4 Discussion

We have presented a new closed-form solution to the alpha channel estimation problem that yields fast implementations for maximum likelihood estimates of the alpha channel. These claims are empirically validated by comparing the algorithm with existing methods (we implemented all the algorithms in MATLAB) in computation time and quality. In our experiments, our algorithm produced results comparable to those of existing algorithms while requiring only a small fraction of the time needed by the fastest probabilistic algorithm.

The efficiency and quality of the new algorithm make it feasible to incorporate alpha channel estimation into a wide range of computer vision and image processing algorithms. In particular, complex and fuzzy boundaries have already been shown to be amenable to extraction using alpha channel estimation.

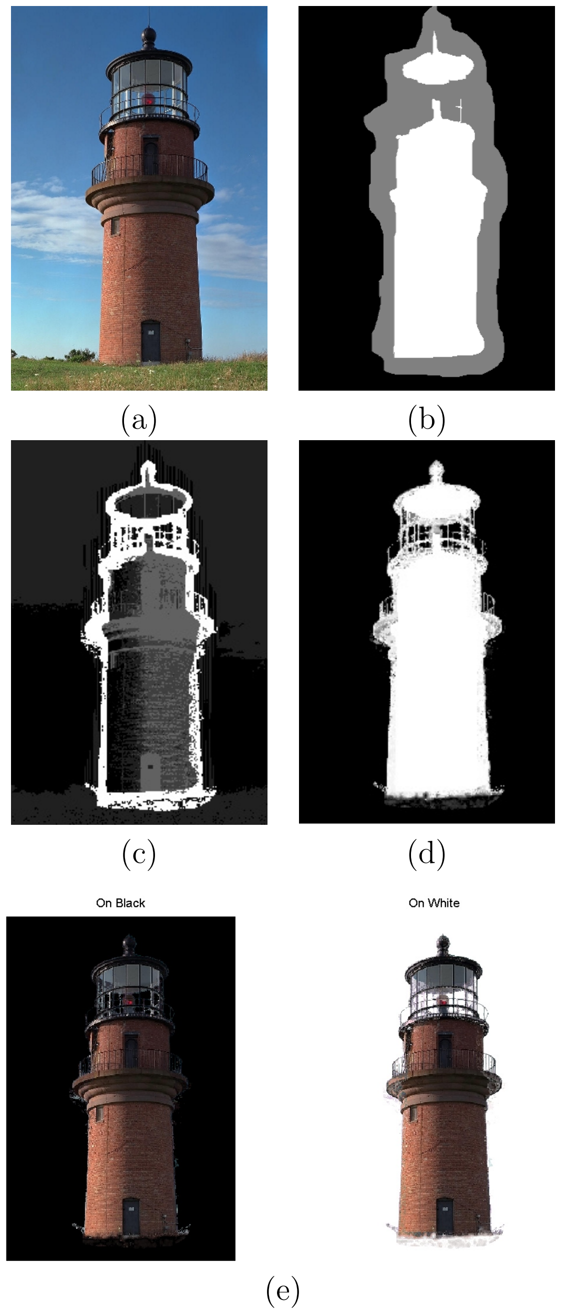


Figure 2.2: Alpha channel estimation experiment with the “lighthouse” image. (a) Original image. (b) Foreground/Background/Mixture map. (c) Estimated alpha channel. (d) Color quantization map. (e) Extracted foreground, composited against a black and a white background.

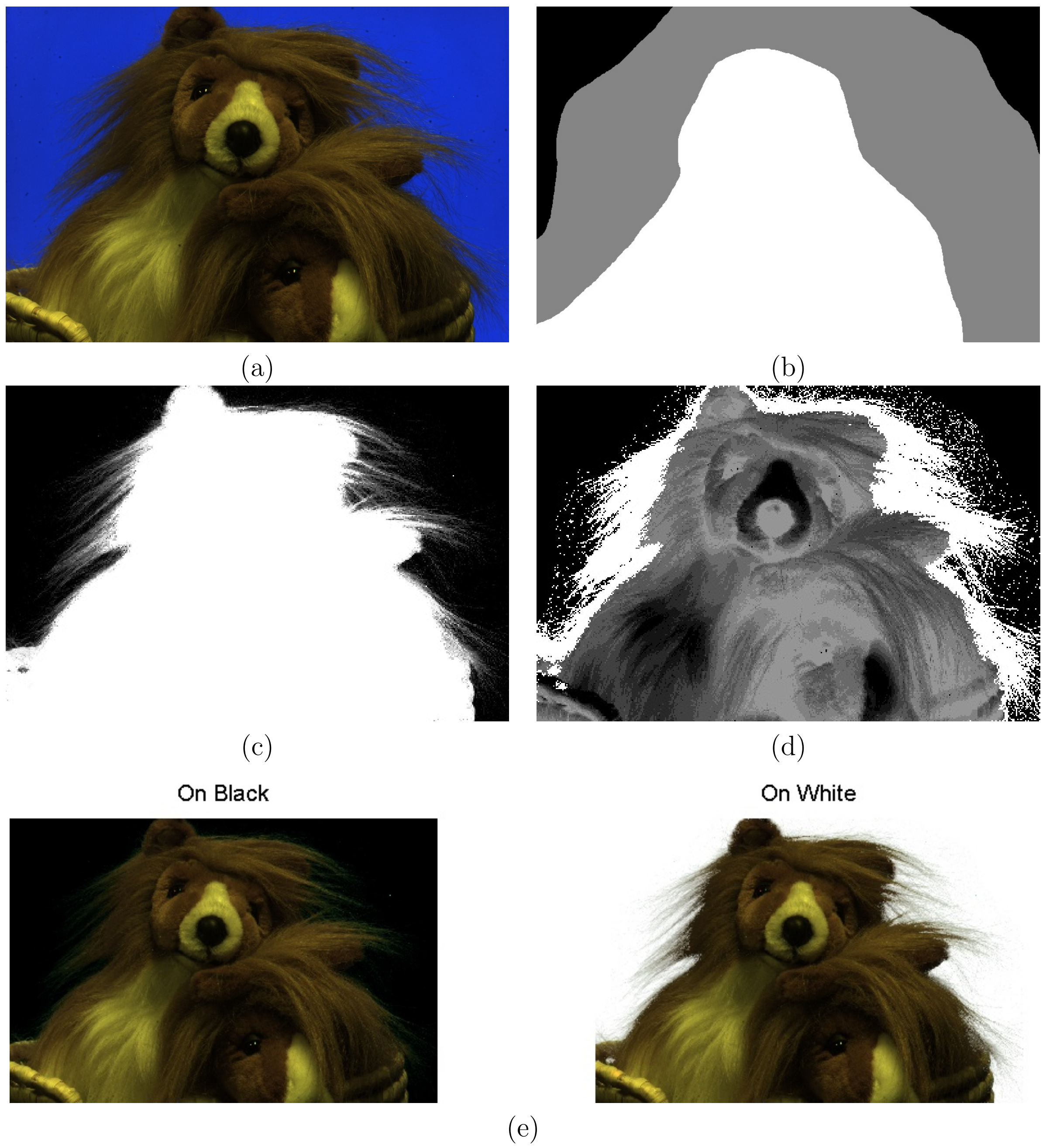


Figure 2.3: Alpha channel estimation experiment with the “lion” image. (a) Original image. (b) Foreground/Background/Mixture map. (c) Estimated alpha channel. (d) Color quantization map. (e) Extracted foreground, composited against a black and a white background.

Chapter 3

Object Extraction from Images using a Sketching Interface

The task of extracting an object embedded in an image or a video stream is performed frequently in many visual content creation applications. For example, artists creating magazine covers routinely extract people or products from photographs to remove unwanted background and compose the new images such that magazine titles appear to be occluded by the object naturally. This operation is commonly called *selection* in digital image editing. Similar to the act of highlighting portions of text in a wordprocessor so that editing operations such as cut-and-paste may be carried out, selection is one of the basic image editing operations that is frequently required. However, unlike text editors, selection in image editing remains a difficult and tedious task for human users.

We would like to address this problem, and build tools to make image and video editing easier. In chapter 3, I will present a new selection tool that not only allows high-quality selections to be created, but makes it possible to use roughly drawn sketches to make the selection. Figure 3.1 illustrates the application of the tool to a non-trivial selection task. Figure 3.1(a) shows the original image from which the head of the crowned crane is to be extracted. Also shown is a sketch drawn by a user that gives a rough indication of the desired object. Figure 3.1(b) shows the resulting selection. It can be seen that the profile of the crane's head is fairly complex, and it would take a human user a fair amount of

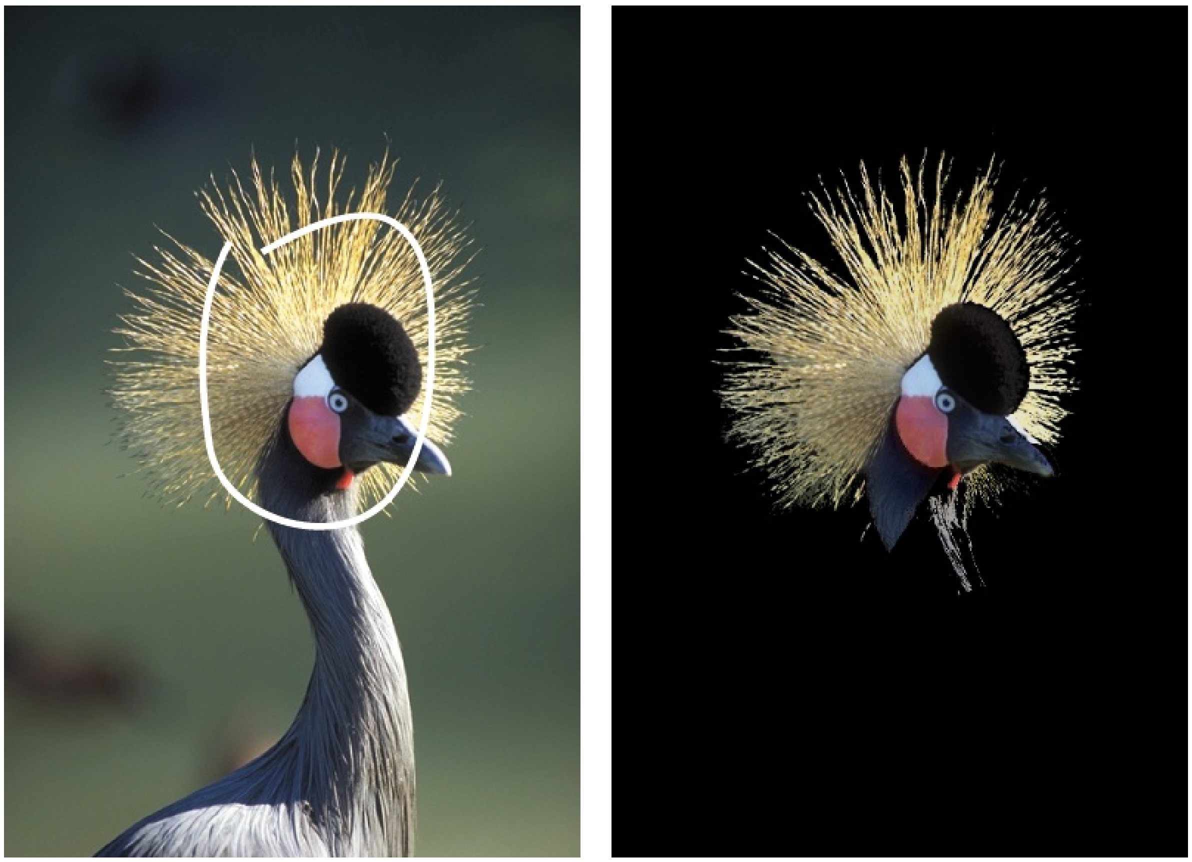

(a) (b)

Figure 3.1: Example of objection selection. (a) Original image, shown with sketch. (b) Resulting selection. (Original photography by Gerard and Buff Corsi, California Academy of Science)

time and effort to delineate the boundaries manually. The sketch shown in Figure 3.1(a) however can evidently be drawn quickly and without much skill. A semi-automatic tool that computes the complex selection using a simple sketch, as shown in the example, thus provides an improvement in ease-of-use over a fully manual tool while still producing high-quality selections. This work first appeared in [91, 92].

Currently, a number of semi-automatic selection tools are commercially available. The primary focus of these tools is the quality of the end result while providing maximum control to the user. As a result, these tools usually require skillful user guidance and cannot be used when the users do not have much time, such as during a live sports broadcast, or do not have much dextrous control, such as on a mobile handheld device, or simply want to avoid the monotony and tedium of selection, without compromising the quality of the selection. To enable the use of sophisticated and high-fidelity graphical selection and manipulation operations in these situations, we have introduced a semi-automatic selection tool that was used to create the selection in Figure 3.1. The tool allows objects to be selected using rough freehand sketches. The sketch-based interface is very natural for use in the emerging class of *tablet* computers for a range of editing applications.

3.1 Previous Work on Selection

In this section we survey the existing body of work related to the object selection problem, and discuss some of the motivation and objectives for our investigation. Traditionally, image editing applications can be broadly classified by their underlying representation for graphical content into two types: vector-based and image-based. Vector-based editors are most often used to create abstract figures such as graphs and stylized illustrations, and represent visual content with a collection of geometric primitives such as line segments and polygons. Image-based editors typically deal with images captured from optical devices such as cameras and scanners, and represent an image with a pixel array. In both cases, selection is a key operation: one has to pick out or specify the portions of a picture that are of interest before manipulation operations like cut-and-paste can be performed. The problem is easier in the case of vector-based systems since an explicit representation of the graphical content is available. Image-based systems in general do not have explicit representations of the visual content, and in many image-based editors the user has to manually delineate objects embedded in images. In our survey of selection tools, we start with a discussion of vector-based tools, followed by a survey of image-based tools.

3.1.1 Selection in Vector-based Editors

Given that selection in vector-based editors is more tractable, it is not surprising that the first intelligent selection tools were created for vector-based editors. We examine two examples, GRANDMA and PerSketch, as editors that went beyond the basic click-to-select way of picking geometric objects.

GRANDMA

An early example of vector-based systems with a non-trivial selection tool is GRANDMA [**?**], a toolkit that uses a marking interface to select objects and specify transformations. An example of image editing with GRANDMA is shown in Figure 3.2.

The primary means of selection is the PACK operation, shown in Figure3.2(d), in which a set of objects of interest are encircled by a stroke. The system then allows the user to manipulate the selected group as a single unit. The user may also click-and-drag objects to specify transformations (Figure3.2(e)-(f)).

PerSketch

PerSketch [76, 84, 85] is an augmented simulation of whiteboard sketching which operates on line drawings. Figure 3.3 shows an example of the kind of operation supported by PerSketch. In the example, a rectangle and a circle are first created, and the circle is then move so that it

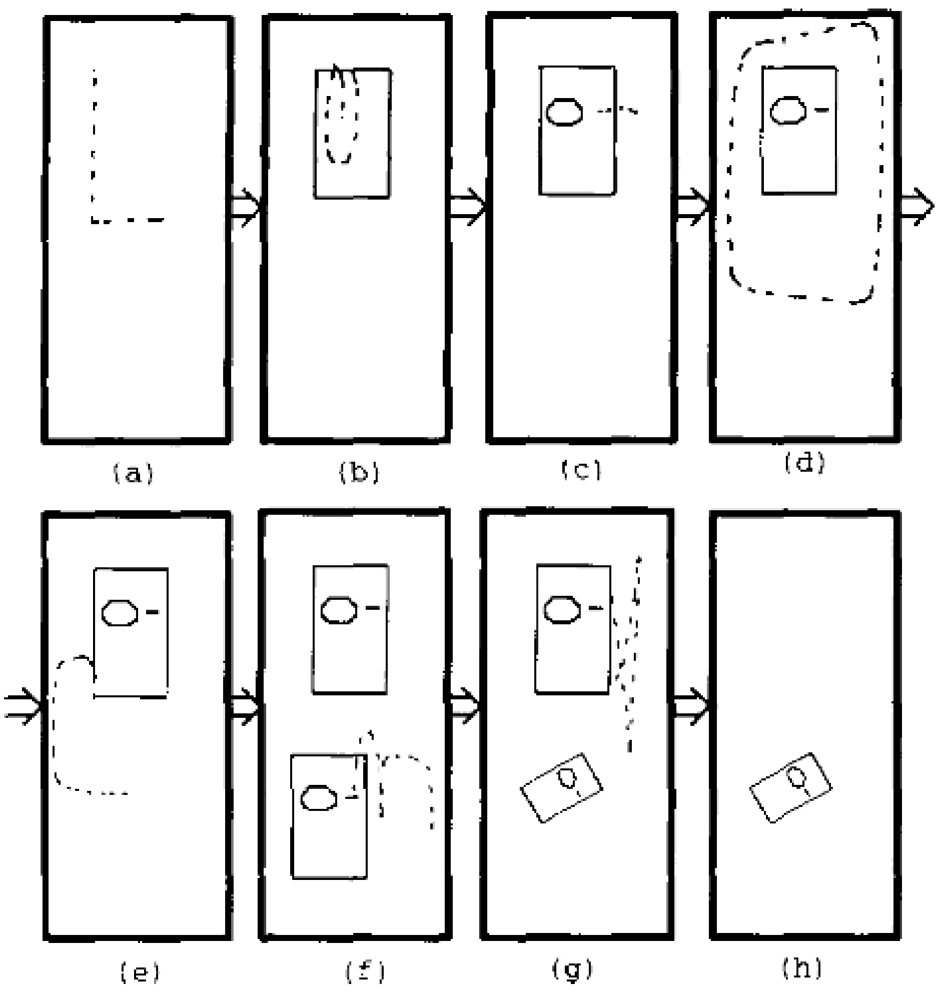


Figure 3.2: An example of vector graphics editing in GRANDMA. (a)-(c) A series of gestures for creating a number of geometric objects, including a rectangle, an ellipse, and a line segment. (d) Shows the selection operator that involves encircling the objects desired. (e)-(f) Manipulation operations on the selected group of objects: duplication(e), rotation and scale(f), and deletion(g).

overlaps the rectangle, operations that can be done with conventional vector-based editors as well. For a human user viewing the picture, new interpretations emerge from the interaction between the circle and the rectangle. For example, the circle can now be viewed as its two halves as one of the rectangle edges appear to cut the circle into two pieces. A conventional editor would still consider the picture as being composed of the original two objects: the rectangle and the circle. PerSketch, on the other hand, would attempt to find these new interpretations and allow the user access to these new interpretations. As can be seen in the third frame in Figure 3.3, the user is able to move half of the circle to the left edge of the rectangle. In the forth frame, the two vertical edges from the original rectangle is removed, resulting in a new elongated shape formed with pieces of the two original objects. Figure 3.4 shows how visual objects in a picture is represented as a collection of atomic PRIME objects, and a collection of COMPOSITE objects assembled by grouping PRIME objects. Given that any particular picture can have a large number of possible interpretations and groupings of PRIME objects, the selection of COMPOSITE objects becomes an interesting problem. The solution proposed in PerSketch is illustrated in Figure 3.5. The user simply draws a sketch that is similar to the COMPOSITE object desired, and the system chooses the object in its current database that best matches the sketch in terms of a number of geometric features such as position, orientation, and bounding rectangle size.

While PerSketch provided a very natural interface for editing line drawings, the underlying perceptual grouping routines do not apply to pixel-array images. In order to create similar selection tools for image-based editors, techniques for extracting visual object representations from raw images are needed. This is however a difficult problem because fully automatic analysis of an image and its decomposition into corresponding primitive elements remain a subject of current research.

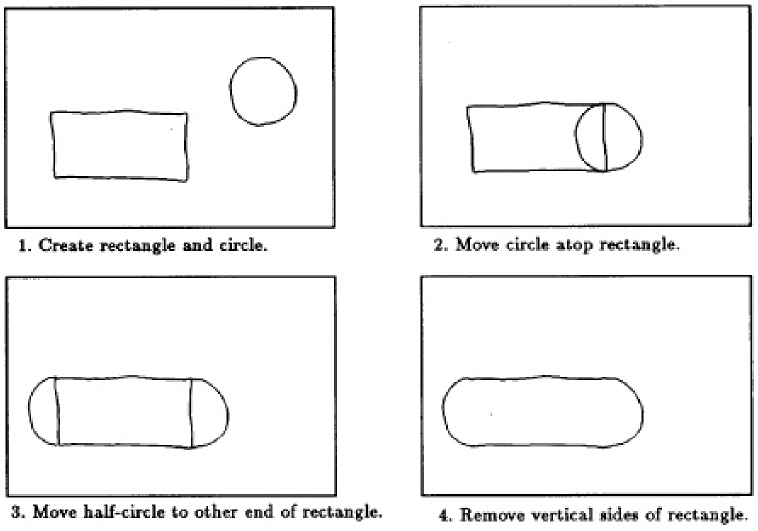

1. Create rectangle and circle.
2. Move circle atop rectangle.
3. Move half-circle to other end of rectangle.
4. Remove vertical sides of rectangle.

Figure 3.3: An example of line drawing editing in PerSketch.

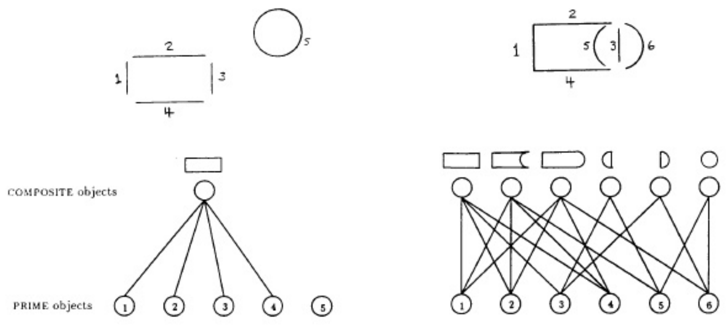


Figure 3.4: PRIME and COMPOSITE line drawing objects in PerSketch.

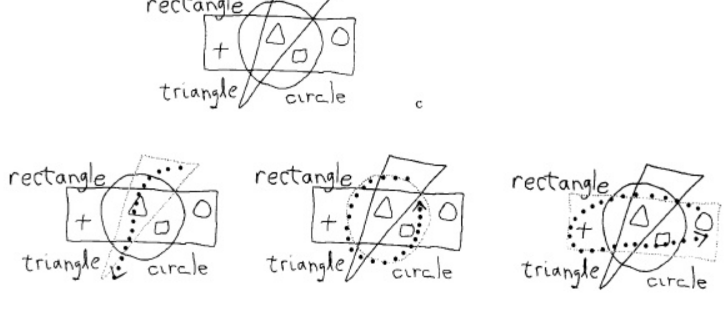


Figure 3.5: An example of selection in PerSketch.

3.1.2 Selection in Image-based Editors

In our context, the key difference between image-based and vector-based systems is the availability of exact representations for the visual objects in vector-based systems. With image-based systems, such descriptor of visual content is not available, and needs to be extracted from the array of raw pixels. There are two kinds of low-level representations for the content of an image that are widely used by researchers and practitioners in image analysis: edge maps and segmentation maps. Edges are linear features, and typically represent the shape and location of object boundaries. A segmentation of an image assigns to each pixel an id, and neighboring pixels with identical ids form regions that represent the spatial extent of objects. The final selection in an image-based editor can be represented by a bitmap, where each pixel has a binary value of 1 if it is a part of the selection, and 0 otherwise. A more general representation allows the value to range continously between 0 and 1, and is called the alpha channel [80]. In this section we shall examine a number of tools that produce bitmap selections. Alpha channel-based methods will be discussed in the next section.

Intelligent Scissors

The Intelligent Scissors [67] and Image Snapping [40] techniques are tools that are based on edges. Figure 3.6 shows how the tool allows a user to select an object by guiding a pointer around the object boundary. In the figure, the pointer is shown as a cross, and its trajectory (called the pointer trail) is shown as a white curve. It can be seen that as the pointer moves along its trail, the tool identifies the salient contours in the image and places a second curve that closely follows the actual object boundary. In Figure 3.6(c) it can be seen that the second curve has formed a closed loop, and the resulting selection is shown in Figure 3.6(d).

Active Contours

Snakes, or active contours [53, 106, 12, 25] are also widely used techniques for identifying salient objects in images. An active contour is a curve that evolves under two kinds of forces,

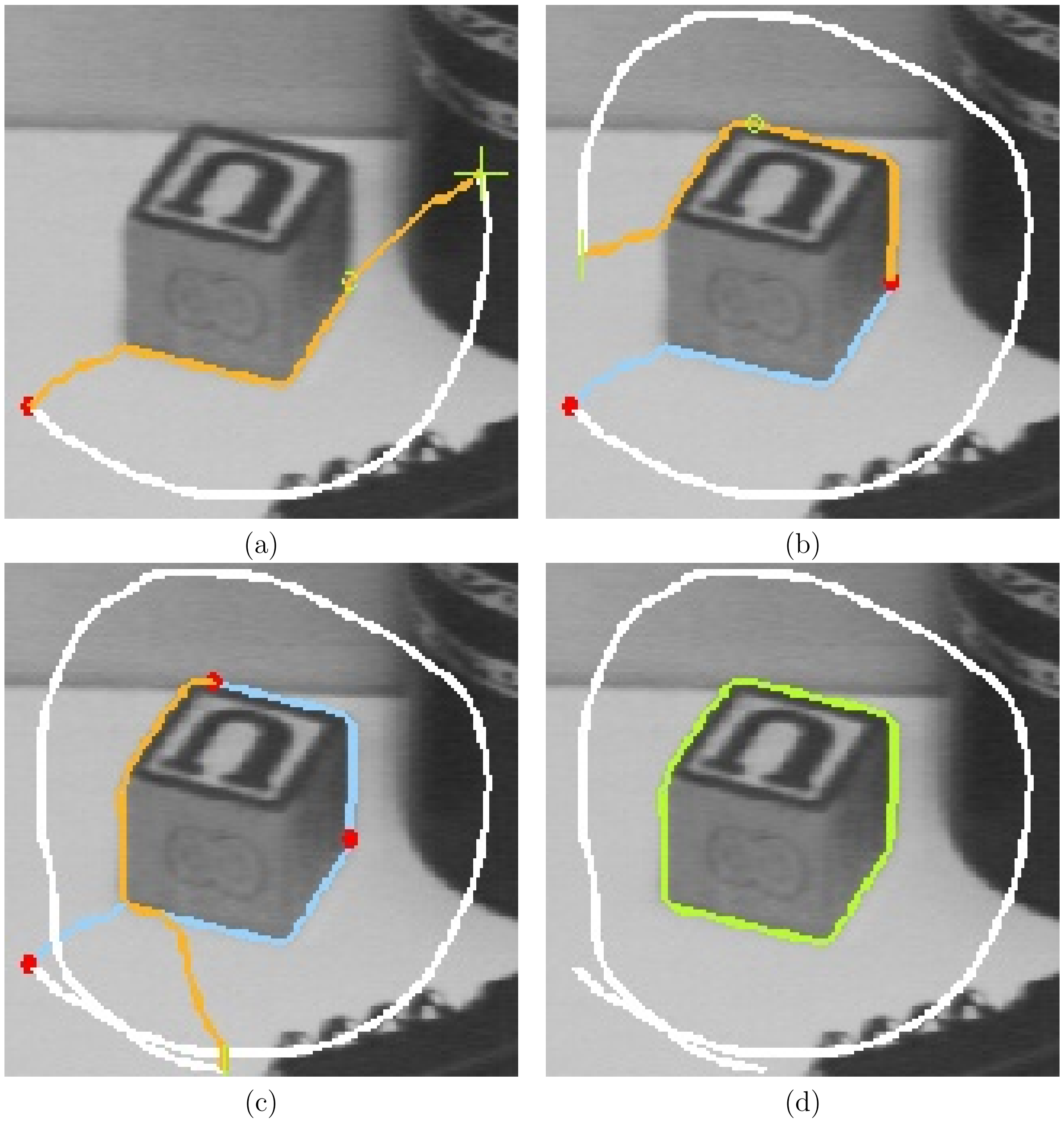


Figure 3.6: Example of selection with Intelligent Scissors.

internal and external. The internal forces make the curve smooth, while the external forces attract the curve to salient edges in the image. Interactive tools implemented with snakes typically have the user draw an initial curve, and then allow the curve to fit the object contours automatically.

Automatic Image Segmentation

As mentioned above, the segmentation of an image is also a commonly used representation for the contents of an image. It is also a natural representation for the object selection problem, since image segments generally correspond to parts of objects, and a selection of an object can be obtained by choosing a number of image segments and forming their union. The problem has been studied widely [90, 104, 6, 61], and remains an area of active research.

3.1.3 Alpha Channels and Diffused Boundaries

A common feature of the selection methods we have seen so far is that the final representation of the selection is a bitmap, such that every pixel may either belong completely to the selected object or be completely unselected. In most photographic images however, boundaries are not as sharply defined. This is true even in high-quality stock photographs such as those from the Corel Stock Photography collection [1]. Examples of diffused object boundaries can be seen in Figure 3.7. In general, object boundaries are not sharp step edges, possibly due to the following factors in the image formation process:

1. The object is not in focus (intentionally or otherwise).
2. The object is in motion.
3. Some objects, such as wispy strands of hair, are simply too small to be represented fully by a finite-resolution digital image.

It is thus important for selection tools to be able to capture this diffusion of the boundaries. In fact the ability to produce an alpha channel is one of the basic requirements of commercial

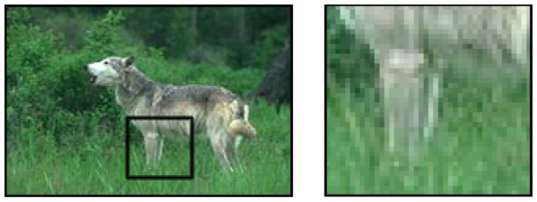


Figure 3.7: Examples of diffused object boundaries. Image is taken from the Corel Stock Photography Collection. (a) Original image, the boxed portion of which is shown in detail in (b), revealing examples of diffused edges. For high-fidelity image editing, the object boundary details need to be fully captured with an alpha channel.

selection tools [23]. Unfortunately, the techniques used by these commercially available selection tools are not published.

Alpha Channel Estimation

Recently an algorithm that estimates the alpha channel in the vicinity of object boundaries was proposed [83]. With this "Alpha Estimation" algorithm, the colors of pixels in the vicinity of object boundaries are modeled as mixtures of colors from the foreground object and those of the background object. It has been shown to be able to extract objects with detailed boundaries, given samples of "pure" foreground and background, and boundary pixels. The algorithm use a mixture model to estimate the alpha channel value at all the boundary pixels.

3.2 Algorithm Overview

A flow chart of the algorithm is shown in Figure 3.2. The inputs to the algorithm are the sketches drawn by a human user and the image containing the object to be selected. The output is the selection in the form of an alpha channel, with each pixel having a real value ranging from zero (completely not selected) to one (fully selected). The algorithm we propose involves solving two problems to allow high-quality object selection with freehand

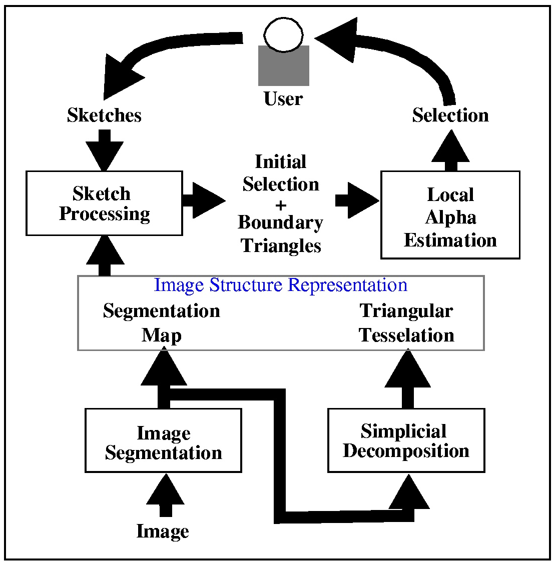


Figure 3.8: Schematic of the object selection algorithm.

sketches: mapping the sketches to the appropriate objects in the image, and computing the alpha channel representation for the object. These two problems are solved using a representation for the image structure that is extracted automatically from the given image. This representation consists of a segmentation map and segmentation-guided triangulation of the image into triangles whose vertices and edges reflect the shapes and spatial adjacency of the segments.

First, we consider the problem of mapping freehand sketches to objects in the image. We allow a user to draw sketches consisting of points, lines, and closed loops (or simply "loops"). Points are typically mouse clicks, and are typically used to select small objects. Lines are non-self-intersecting curves used to indicate object boundaries. Loops are curves that start and end at the same point, and are used to indicate the spatial extent of objects. We allow the user to specify the selection with different degrees of precision, according to the complexity of the image, so that where there is no ambiguity the user should be able to be less precise while drawing the sketches. Our solution to this first problem is shown in the flow chart as sketch processing, and its output is a set of segments whose union forms an initial selection, and a set of triangles whose vertices straddle the boundary of the selection. These triangles that we refer to as the set of boundary triangles also completely cover the boundary of the selection, and thus decompose the spatial vicinity of the boundary into triangles.

The initial selection and the boundary triangles is then passed to the algorithm shown in the flow chart as local alpha estimation. The objective of this stage is to compute the final selection using the alpha estimation algorithm [83], a method for factoring out the foreground objects contribution to a pixels value when the pixel value is a mixture of the foreground object and the background. The alpha estimation algorithm, as originally proposed, assumes that pure samples of the foreground and background are given by a user. We automate this process by using the initial selection to get the pixel samples. The problem of estimating the alpha channel over the entire boundary is also broken into small, local subproblems using the boundary triangles. The final selection is then obtained by combining the alpha channel computed within each boundary triangle. In the following sections, we present the details of the algorithms used.

3.3 Image Segmentation

We use a segmentation of the image to represent groups of pixels that form objects or parts of objects. The algorithm we use is binary-split vector quantization in color space. For each pixel in the image, we create a three-tuple *(r,g,b)*, with one component for each color space component. Initially the data points will all be placed in a single cluster. We split this cluster into two at its mean along the direction of largest variation, and recursively split the resulting clusters until the number of data points in each is below a given threshold. Typically we let the threshold be one half, quarter, eighth, or sixteenth of the number of pixels in the image. The cluster label for the pixels thus forms a raw segmentation map. We then apply a morphological “majority” filter that replaces the label of a pixel by the value that occurs most frequently within a square window centered at the pixel. We then relabel the pixels so that each 4-connected component in the segmentation map has a unique label. The segmentation computed by this procedure is an approximate representation for the spatial extent of objects in the original image, although shape details are lost due to the morphological filters employed. However, as we will see, this form of segmentation is adequate for our purposes.

3.4 Simplicial Decomposition

The segmentation map provides a pixel-resolution approximation of the objects in the image. While this form of representation for image structure is already useful for many applications, it is still not sufficient for mapping the sketches to the segments. For example, it is still difficult, in general, to answer the question: "is a segment to the left side or right side of a line ?" when the line cross the segment boundary and parts of the segment is on the right side, while other parts of the segment fall on the left side. This is, of course, the type of question we need to answer in order to map freehand sketches onto image segments. Figure 3.9 illustrates the way we answer the above question. The two regions A and B have a curved shared boundary, which intersects the thick black line, so that no segment is strictly on one side of the line. However if we can represent the two segments by two points, then there will be no ambiguity no matter how the line is drawn. Obviously, such a representation is only applicable in the local vicinity of the points. In order to fully represent the shape and spatial configuration of a segmentation, we also need to decompose it into small, manageable pieces such that the geometric queries necessary for the object selection problem can be answered. In this section, we will describe how such a representation can be computed. We propose that a simplicial decomposition of the image into triangles be used. In order for the triangles, edges and vertices to reflect the spatial characteristics of the segmentation map, we require the triangulation to satisfy the following constraints:

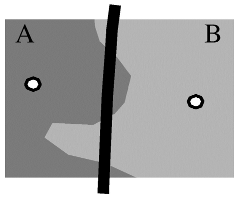


Figure 3.9: Using points to represent the relative positions of regions.

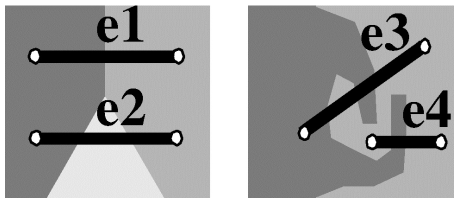


Figure 3.10: Examples of valid and invalid edges in the desired triangle decomposition: Edges *e1* and *e3* are valid, while edges *e2* and *e4* are invalid.

1. All vertices lie on the medial axes of the segments.

2. For an edge between two vertices, one in region A and one in region B, the edge must lie entirely in the union of the two regions.

3. Each triangle must be contained in at most three regions.

The first property ensures that vertices are always inside the respective segments they are representing and are not positioned too near to boundaries. The second property ensures that the triangles formed reflect the adjacency relationship well, in the sense that an edge is always between two vertices representing adjacent segments. The third constraint ensures that segment boundaries are always enclosed by vertices representing the respective segments. Figure 3.10 illustrates the second property. It is worth noting that the example edge *e3* cuts the region boundary more than once, but is still considered a valid edge because it lies entirely in the union of the two segments. Edge *e4* is invalid because its two vertices lie in the same region, but the edge passes through two different regions.

The algorithm for triangulating the segmentation map is essentially a procedure for systematically covering the segment boundaries while choosing points and adding triangles that

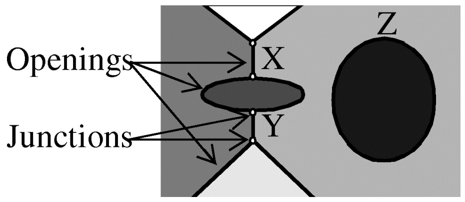


Figure 3.11: Examples of openings and junctions. The diagram shows a segmentation map with six segments, shown in different shades of gray.

satisfy the constraints. We need to define a few terms to facilitate the description: a pair of adjacent regions meet at their shared boundaries. We call each continuous piece of the shared boundary an *opening*. The word opening is used because in a triangulation there will be at least one edge passing through each opening, and these edges can only pass through this piece of shared boundary between the two regions-an opening. Junctions are points in the segmentation map where three or more regions meet. In the case of a digital image, where pixels are on a rectangular grid, at most four regions can meet at a point. Examples of openings and junctions are shown in Figure 3.11. Openings are shown as solid black lines and junctions white dots. Openings X and Y are on the share boundaries of the same regions, but the two are distinct. Opening Z forms a closed loop. Openings that do not form loops will have two end points, each one either a junction or a point at the edge of the image. Each opening thus can have two, one, or no endpoints. The main steps of the decomposition algorithm are as follows:

- Triangulation Algorithm

  1. Cover each junction with a triangle.
  2. Merge vertices within each segment as much as possible.
  3. Cover openings with 2,1 or 0 endpoint(s).
  4. Cover the remaining area in the image.

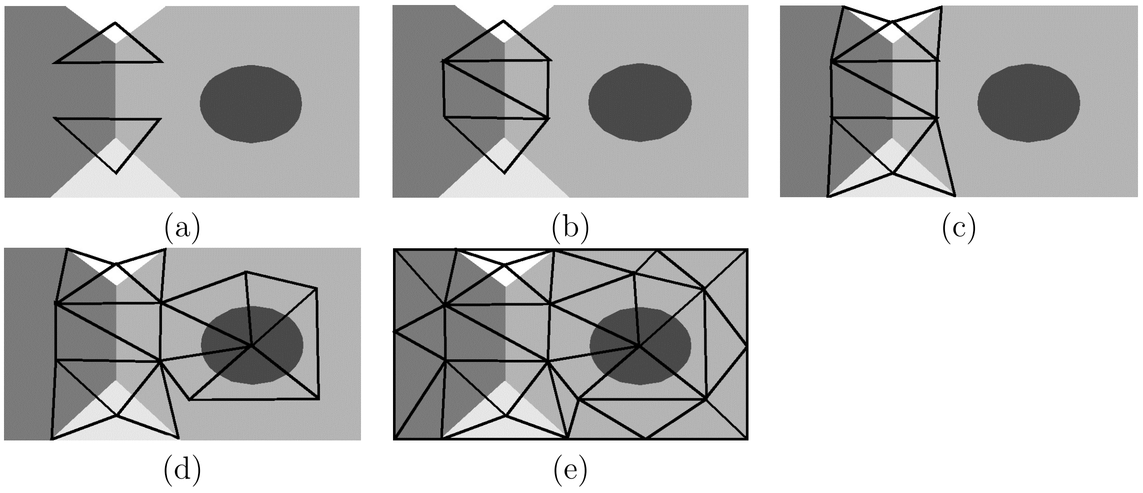


Figure 3.12: Steps in the triangulation algorithm. (a) Cover the junctions. (b) Cover the opening that has two covered end points. (c) Cover the openings that has one covered end point. (d) Cover the openings with no end points (in this case a loop). (e) Cover the remaining parts of the image between the hull of the triangles and the border of the image.

Figure 3.12 illustrates the steps in the algorithm. We now discuss the individual steps in more detail.

3.4.1 Covering Junctions

Triangle placement starts at the junctions. Where three regions meet, we pick one vertex from each of the three region skeletons corresponding to the junction such that they form a triangle with edges that are valid. At junctions where four regions meet, we place two triangles. The two cases are shown in Figure 3.13. We use the following goodness measure for a triangle:

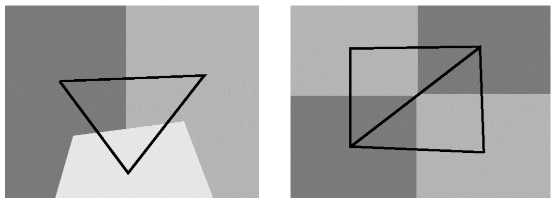


Figure 3.13: Placing initial triangles at junctions. (a) At a three-region junction, one triangle is placed. (b) At a four-region junction, two triangles are placed. In a digital image, these two cases are exhaustive.

$$\frac{\sum_{i=1}^{3} w_i}{C+P}\theta_{min}$$

where

w_i	is the *depth* of a vertex i, its distance from the nearest boundary of its containing segment
θ_{min}	is the minimum of the three internal angles of the triangle
P	is the perimeter of the triangle
C	is a constant

Thus we favor small triangles with large internal angles, and vertices that are positioned deep inside their respective segments. Search for the triangle proceeds in the following manner: we rank all points on the respective segment medial axes by their depths, so that vertices with large depths are highly-ranked. The search is then constrained by using only the points in the 90 percentile of each of the segment medial axes involved. With this small set of points, we examine all possible triangles and pick the best. If no triangle is found due to the vertex weight constraint, we drop to a lower percentile and redo the search. Figure 3.14(a) illustrates a typical set of triangles satisfying the tesselation constaints found with this search procedure.

Figure 3.14(b) shows another set of triangles satisfying the same constraints, but use fewer distinct vertices. We obtain this latter result by a procedure that merge all possible vertices within a segment. For each vertex v, we call the two vertices on the same triangle

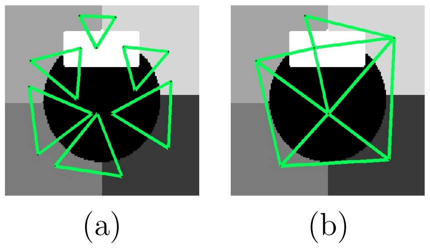


Figure 3.14: Vertex merging. (a) The segmentation map, shown with the initial set of triangles placed at the junctions. (b) The result of vertex merging.

its neighbour vertices. We say that two vertices can be connected if there is a valid edge between the two vertices. Two vertices can be merged if they each can be connected to the neighbours of the other vertex. Generalizing, a vertex can be added to a set of vertices with a neighbour set formed by the union of all the neighbours of the vertices of a set if the vertex can be connected to all vertices in the neighbour set and all vertices in the set can be connected to the two neighbours of the new vertex. We use a greedy procedure to identify a number of maximal merge sets by starting with a single vertex and then incrementally added to the merge set vertices that can be merged with the vertices in the set. A merge set is maximal when no vertices from the same segment can be added to it. We then merge the vertices in the set and start growing the next merge set by a remaining vertex not in the merge set. We repeat this until all vertices have been included in a merge set. In the worst case, all merge sets contain only one vertex, and no merging occurs.

3.4.2 Openings with Two End Points Covered

Since the initial set of triangles are placed at junctions, and junctions are endpoints for openings, each initial triangle thus cover the endpoints for either three or four openings. This property is retained by the new set of triangles obtained by the vertex merging operation. Unless an opening ends at the border of the image or forms a closed loop, both its end points would be covered by these initial triangles. We identify such openings and place additional triangles between the two initial triangles at its two ends to completely cover these openings.

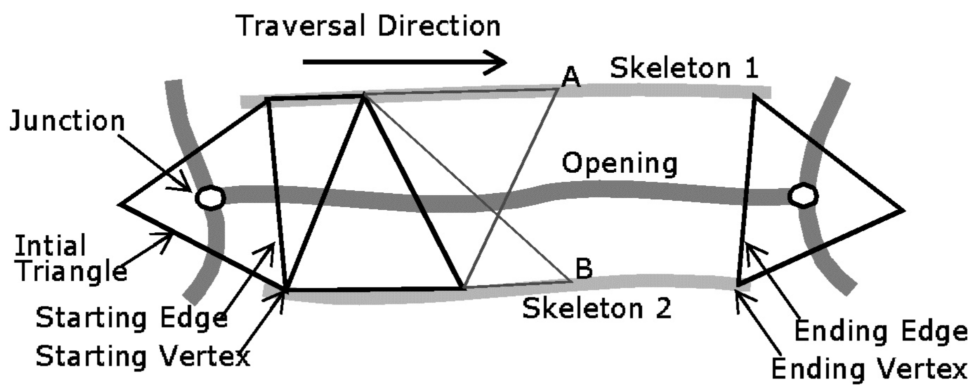


Figure 3.15: Traversing an opening and covering it with triangles. Triangles that have been placed are shown with thick edges, while the candidate triangles are shown with thin edges.

Figure 3.15 shows such an opening, with its two end points covered, leaving the middle portion still uncovered. It can be seen from the diagram that the area in which the triangles need to be placed is bounded by the two skeletons, and the two triangle edges at the two ends. This suggests that we can cover the opening by traversing along the opening from one end to the other, placing new triangle vertices on the two opposite skeletons. The traversal algorithm we use is as follows:

- Adding Triangles by Skeleton Traversal

  1. Given an opening with its two end points covered by two triangles, identify the two skeletons, and the starting and ending triangle edges.
  2. The starting and ending edges each have a vertex on the two skeletons. These are the starting and ending vertices for each of the skeleton. Use Dijkstras shortest path algorithm to find a path on each skeleton from the starting vertex to the ending vertex.
  3. On each of the skeleton paths, traverse the path. At each point along the traversal, while the following conditions are true:
     (a) The point is connected to the two vertices of the starting vertex (and thus is able to form a candidate triangle using the starting edge and the new point).

   (b) In the new candidate triangle, the angle at the new point is larger than a preset value θ_{min}. This avoids the creation of thin "sliver" triangles.

   (c) The new point has not reached the end of the path.

4. Given the two new candidate triangles, pick the one that covers more of the opening. Add the new triangle to the existing set. Add new triangles on the other side of the skeleton on which the new point was added, using the new point.

5. Let the new starting edge be the new edge between the two skeletons that is added with the new triangle.

6. Repeat steps 3-5 until the starting edge coincides with the ending edge

3.4.3 Openings with One End Point Covered

If an opening starts at a junction and ends at the edge of the image, it would have only one end point covered. We can treat this end point as a junction by having a virtual, infinitely-thin segment at the edge of the image, running around the frame of the image. This is illustrated in Figure 3.16. Then opening end points at the edge of the image can be thought of as a junction involving the virtual segment. In order to place a triangle covering this junction, a vertex would need to be placed at the point where the opening meets the edge of the image since that is the only point on the virtual segment that can be connected to both of the skeletons for the opening. Now we can choose a triangle in the same manner as before. The portion of the opening between the new triangle and the triangle at the other end of the opening can then be covered by vertex merging and skeleton traversal.

3.4.4 Openings with No End Point Covered

In the case where both opening endpoints are at the edge of the image, we can place two triangles at the two ends as described above, and then covering the rest of the opening by vertex merging and skeleton traversal between the two new triangles.

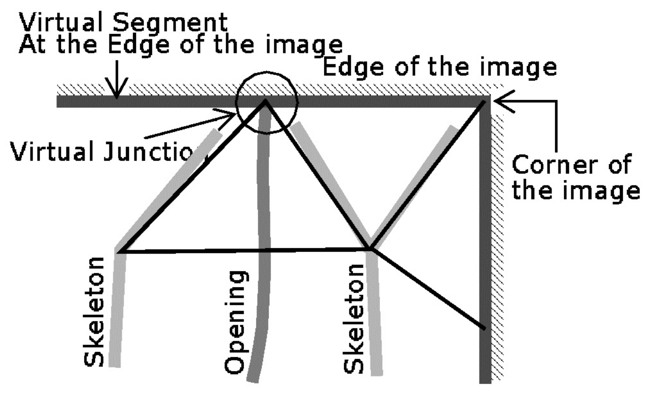


Figure 3.16: Using a virtual segment at the edge of the image.

Openings forming loops do not have endpoints, and as such do not have initial triangles. We simply choose an edge between the two respective region skeletons, and add triangles to cover the loop formed by the opening by skeleton traversal. If there are vertices already placed on the skeleton, we can use the existing vertex to place the edge. We simply use the chosen edge as the starting edge and also the ending edge.

3.4.5 Covering Borders

Finally, we need to place triangles in the remaining portion of the image, between the image boundary and the triangles placed thus far, to ensure that the entire image is covered with triangles. All the regions that need to be covered in this step is by now bounded by the edge of the image and the hull formed by the triangles already placed. We just need to identify these uncovered regions, and again use the skeleton traversal method to cover these regions, using the "virtual skeleton" of the virtual segment at the edge of the image and the hull of the existing triangles as the opposite skeleton. During this traversal, we use the area covered to choose between candidate triangles. As a result of using the virtual segment reasoning, there will always be a vertex at the corners of the image, since those are the only points along the virtual skeleton that can be connected to points on the two edges meeting at the corner. This is illustrated in Figure 3.16.

3.5 Sketch Processing

Once we have the segmentation and the triangulation, we can use them to map the input sketches to image objects. This section describes the mapping procedure. Recall that the three sketch elements are as follows:

1. Points for specifying small objects.
2. Lines for describing boundaries.
3. Loops for describing image/object regions.

Recall also that the end result of all three types of sketch processing is to classify the set of image segments into two sets: foreground and background. The union of all the foreground segments forms the initial selection. For all three types of sketch elements, the entire set of segments are initially labeled as belonging to the background, and we thus start with an empty selection.

Of the three types of sketch elements, the segmentation map is used in the processing for points and loops, while the triangles are used to process lines. Points are the simplest to process: for each point, we pick the segment label for the pixel under the point and add the corresponding segment to the foreground. Loops indicate the approximate spatial extent of the object. For each segment in the coarse segmentation, we find its intersection with the region enclosed by the loop. Segments with a significant portion of its pixels in the loop are then chosen as the foreground object segments. Lines are processed as follows:

1. Find all triangles that touch the line.
2. For each such triangle, classify the vertices (and thus the centroids) as either foreground or background depending on the side of the line they fall on (predetermined, using the right-hand rule).

3. Form the union of all the sets of foreground vertices from each triangle and use it as the set of foreground centroids selected by lines. All other centroids then are the background centroids. The definition of this polarity is an arbitrary decision, though it is important for all relevant stages of the algorithm to adopt the same convention.

The specific meaning of points and lines with respect to the object selection task is now well-defined. The benefit of using the image structure to provide a context for sketch processing is that it allows variations in the sketch while providing an unambiguous meaning of the sketch. For example, a point can fall onto any part of a segment to select the segment centroid. Lines indicating a boundary between two segments can be located anywhere between the two corresponding centroids, and the triangulation establishes an unambiguous correspondences between each line and the relevant segments in the image.

3.6 Local Alpha Estimation

At this point, we have labelled each segment as being in the foreground or the background. We also have the triangulation of the segmentation. Now we can compute the final selection, in the form of an alpha channel map and an estimate of the foreground pixel colors factored out from the image. We use the triangulation to partition the computational task into smaller pieces, and apply the object extraction procedure independently in each piece. We partition the problem as follows: identify all triangles consisting of both foreground and background vertices. We call these the boundary triangles since they define the vicinity of the object boundary. The entire subsequent object extraction computation is performed within these triangles independently. The rest of the triangles are either completely contained in the foreground or in the background. The alpha channel value for pixels inside these triangles are uniformly one for pixels in foreground triangles and uniformly zero for pixels in background triangles. We now describe the algorithm that applies to each boundary triangle (illustrated in Figure 3.17). We apply the alpha estimation algorithm [83] to the pixels inside each

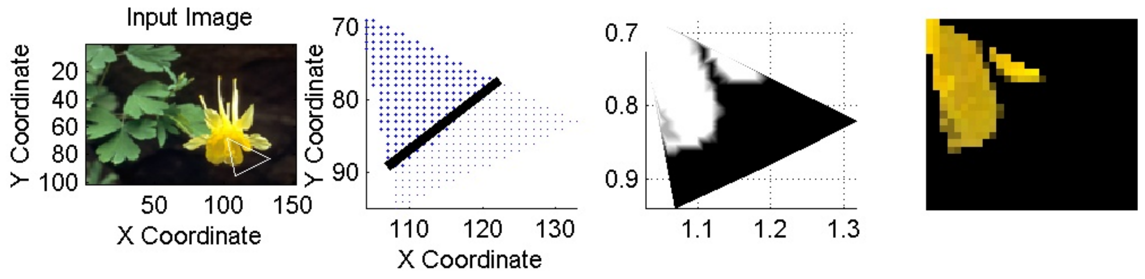


Figure 3.17: Local alpha estimation. (a) Image with a boundary triangle. (b) Linear discriminant. (c) Alpha channel estimated. (d) Object extracted.

triangle. Instead of having user-provide pure sample pixels, we automatically provide these samples as follows: we first perform a finer-scale segmentation on the pixels within each triangle. To ensure that this local segmentation is of a finer scale than the original, global segmentation (so as to capture more details locally), we set the size threshold to a quarter of the number of pixels in the triangle. This ensures that the pixels will be split into at least four clusters. Given, from the triangulation constraints, that each triangle will contain pixels from at most three global segments, the local segmentation is guaranteed to be more detailed. We then find the spatial centroid for each of these clusters, and classify these new cluster centroids into foreground and background using a linear discriminant that approximates the initial selection boundary in the triangle. We use Fisher's Linear Discriminant [35] , and let the discriminant pass through the following point:

$$w = \tfrac{n_b}{n_f+n_b}\mu_f + \tfrac{n_f}{n_f+n_b}\mu_b$$

where

n_f, n_b	are the number of pixels in the foreground and background groups respectiely, and
μ_f, μ_b	are the corresponding centroids for these local pixel populations

We then sample pixels from the respective cluster in the vicinity of the centroids and use them as the pure samples, and estimate the alpha channel value for every pixel in the triangle.

The alpha channels for the boundary triangles combine to yield the final selection. A question one may ask is whether these independently computed selections will produce a seamless result. To answer this question, recall the manner in which the selection problem is divided into subproblems of selection within triangles. Each pair of adjacent triangles share two vertices, and therefore, the color population of the segments corresponding to those vertices. These segments remain unaffected by the triangulation, thus ensuring continuity across the edges of the triangle.

3.7 Experiments

Our first results illustrate the use of points and loops for selection, and they are shown in Figure 3.18. In the case of points, the user simply clicked on the object, adding object parts until the desired object is selected entirely. Loops are more intuitive, where the user would simple enclose the desired object with a loop. Selection with lines offfers the most freedom to the user, and some examples are shown in Figure 3.19(a)-(b). The same figure also shows how the same selection can be obtained using points and loops. This shows how the tool allows a reasonable degree of freedom in making the sketch. Figures 3.20 to 3.24 illustrates the operation of the algorithm on a high-resolution image. We can also use the triangulation to further decompose the original coarse segments into smaller pieces, such that each piece corresponds to a vertex. The new decomposition is easily found from the triangulation: examine each triangle with two vertices that fall in the same segment. Identify the edge between these two vertices and find its midpoint. Draw a line from the midpoint to the third vertex in the triangle. If the edge is shared by a second triangle on the other side, do the same for the second triangle. Now cut the segment containing the two vertices along these lines. An example segmentation induced by the triangulation is shown in Figure 3.25 . The resulting segments each represented by a vertex in the triangulation, has the property that if two segments are adjacent, then there is a valid edge between the two representative

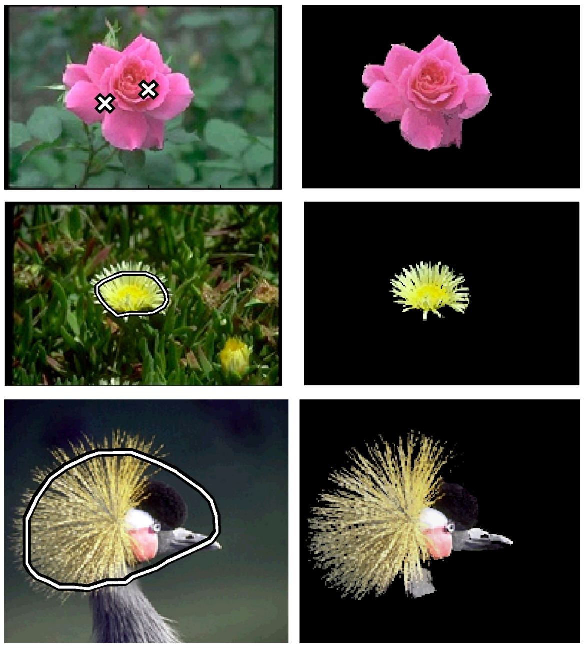


Figure 3.18: Examples of selection by points and loops. (a) Original image shown with sketch. (b) Resulting selection composited against a black background. Crane example cropped from an image by Gerald and Buff Corsi, California Academy of Science.

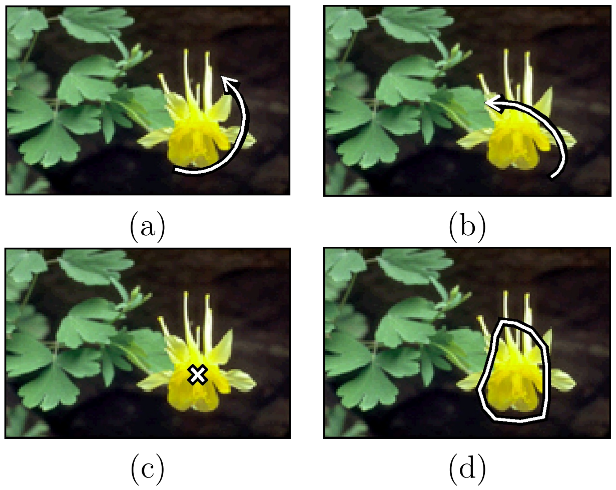


Figure 3.19: Different sketches that yield equivalent results. (a)-(b) Lines. (c) Point. (d) Loop.

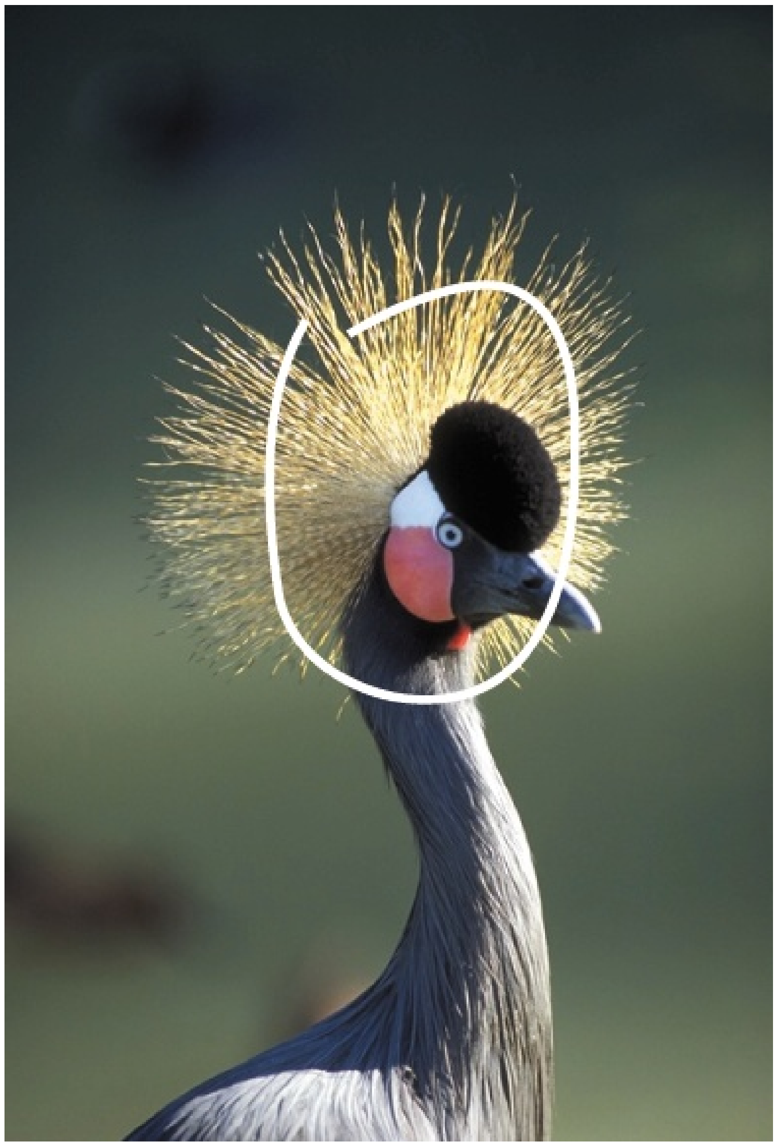


Figure 3.20: A high-resolution image, shown with a line drawn by a user.

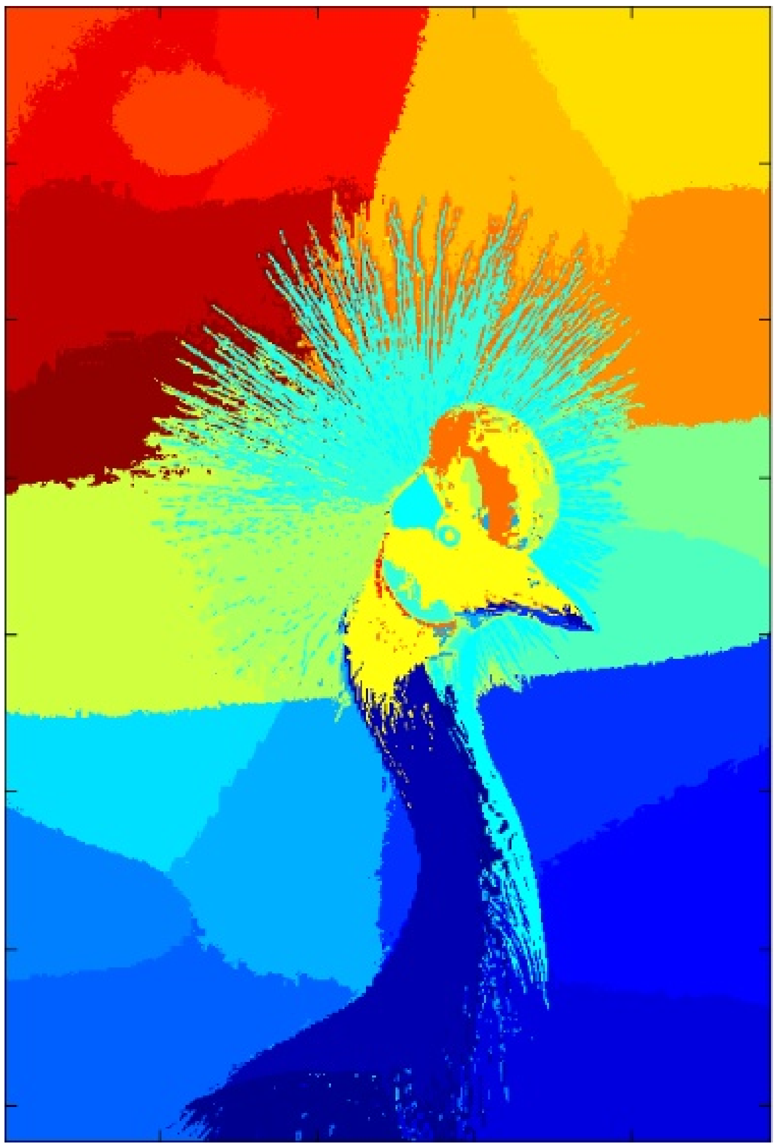


Figure 3.21: The segmentation used for the selection.

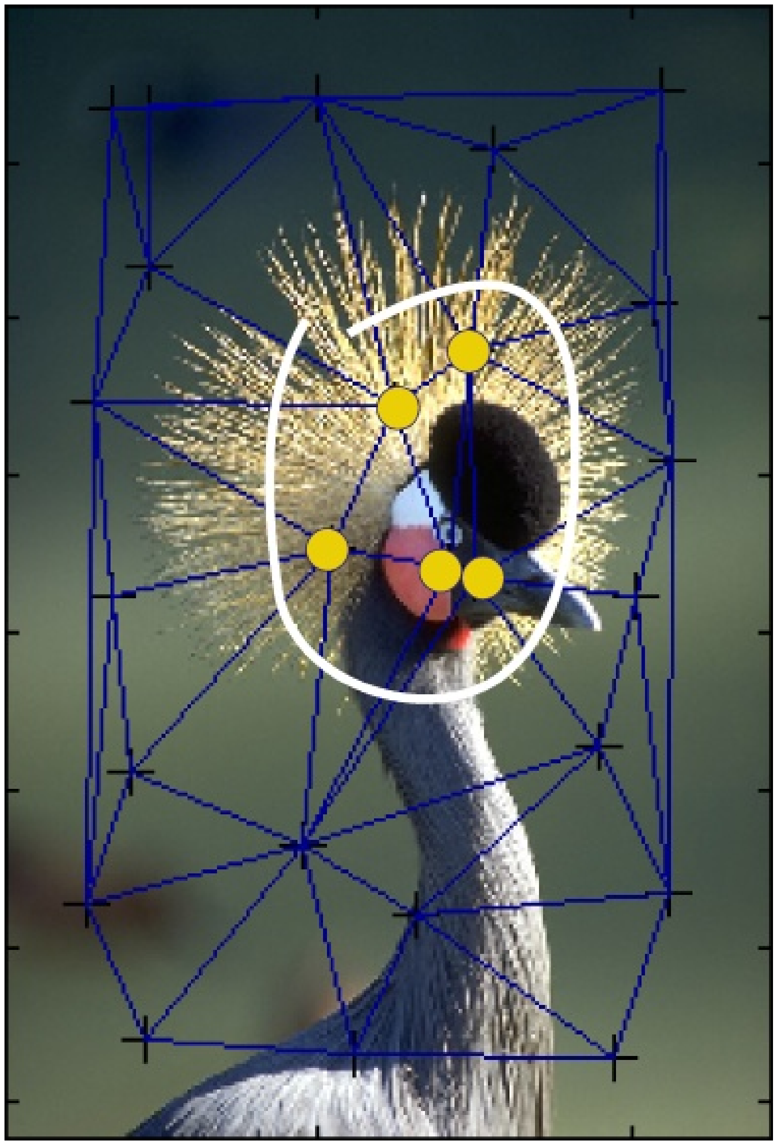


Figure 3.22: The Delaunay triangulation of the segment centroids, shown with the sketch. Light-color discs indicate the selected foreground centroids.

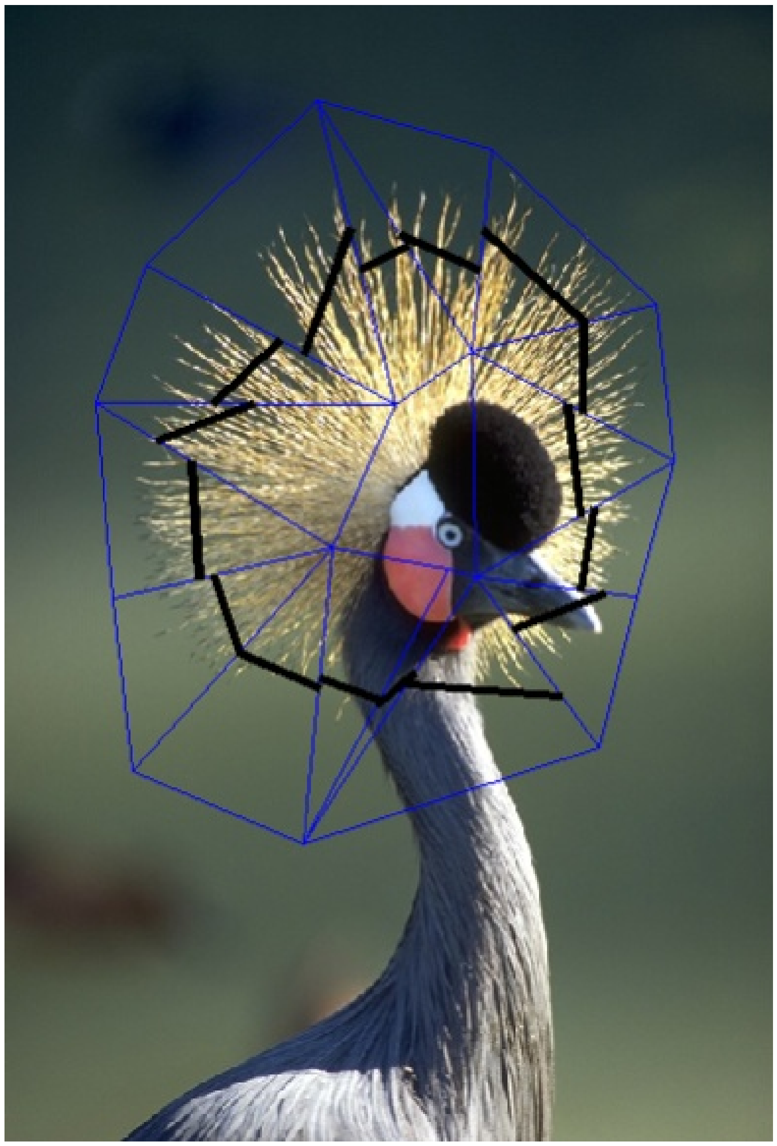


Figure 3.23: The boundary triangles, shown with the local linear discriminants.

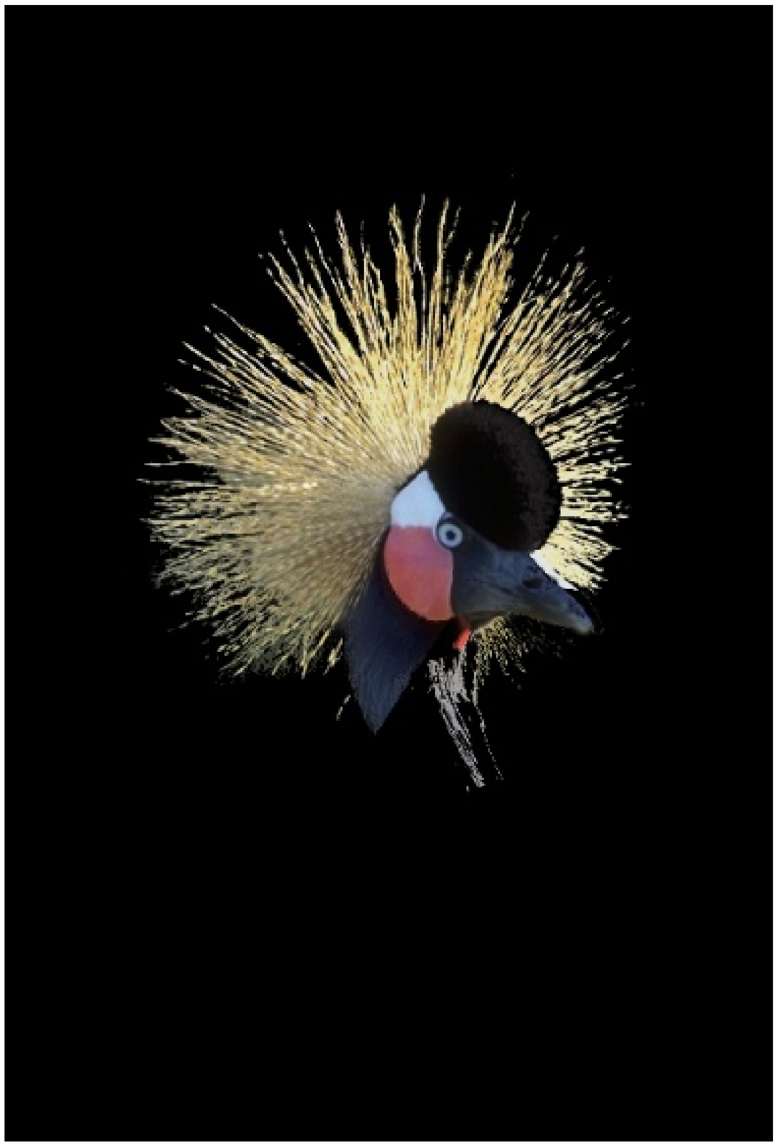


Figure 3.24: Extracted object, composited against a black background.

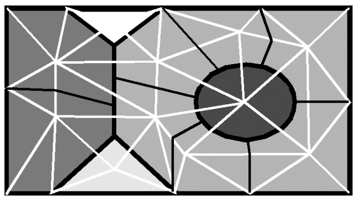


Figure 3.25: Structure-induced Segmentation. The original segment boundaries are shown in thick black lines, the triangulation is shown in thin, lightly-shaded lines. The new segment boundaries induced by the triangulation is shown with thin black lines.

vertices. This is a reasonable way to decompose an image into small pieces, and is useful in a number of applications. For example, one can use these segments in matching and recognition tasks, among others.

3.8 Discussion

We have presented a method by which simple freehand sketches may be used to select objects with complex boundaries. We have demonstrated with experimental results that the method is able to extract complex objects with a minimal amount of user input.

Ongoing work include improving the computational complexity of the simplicial decomposition stage and studying additional constraints on the triangles. For example, it might be interesting to place the vertices such that the edges can be found automatically by computing the Delaunay triangulation of the vertices. It may also be interesting to place the vertices such that the voronoi diagram of the vertices approximate the actual segment boundaries.

Chapter 4

Object Tracking with Faceted Appearance Models

The modeling of object appearances is an area that has been well-studied in recent years, and is important for solving many problems in computer vision, such as object recognition and pose estimation. It is also a key issue in object tracking, since the appearance of the object being tracked can vary dynamically. For example, a human head can appear very different when viewed from the front and the back. In order to track objects over a video sequence, it is thus necessary to take their appearance variation into account.

A popular approach for object appearance modeling is to treat an image as a point in a high-dimensional space. For example, a 20 pixel by 20 pixel intensity image can be considered a 400-component vector, or a point in a 400-dimensional space. Algorithms using this representation are often referred to as being *appearance-based* or *view-based*, and they have been successfully applied to object recognition and detection [10, 68, 101]. In [68, 73], Nayar showed that a set of images of an object taken under varying viewing parameters forms a continuous set of points in the high-dimensional space, commonly called an *appearance manifold*. If the appearance manifolds of a number of objects are available, then object recognition is simply the problem of finding the manifold that is nearest to a given sample point. If the pose parameters are available for the point on the manifold that is nearest to the sample point, they can also be used as a pose estimate for the sample point. It should be noted that although the term ‘appearance manifold’ is used commonly in the literature, the set of images of an object under varying viewing parameters in general do not satisfy all the properties of a manifold as defined in differential geometry. In the rest of the paper we shall use the term ‘appearance set’ to refer to such a set of images, and the term ‘appearance model’ to refer to representations for the set.

In order to use appearance sets in computational algorithms for recognition, it is necessary to have a representation that supports nearest-point querying efficiently. It is also essential that an appearance model can be built from a finite set of image samples. In this paper we propose the use of a construct we call faceted models to represent an appearance set. The key distinguishing feature of faceted appearance models from existing representations is that the adjacency information in the sample set is explicitly encoded, and used to enforce a topological constraint that improves representational accuracy without incurring excessive costs in terms of computation time and storage space.

4.1 Related Work

The tracking of point features across video frames is probably the most widely-used low-level motion cue. Feature tracking is most commonly achieved by following an optic flow field [60, 14, 45, 89], Once a set of points are reliably tracked, camera motion, scene structure [97], and even flexible and deformable object shape [18, 19, 99, 98] can be recovered, typically by applying a rank constraint.

In many applications, it is also important to have an estimate for the boundaries of objects in a video stream. For example, in medical imaging it is often useful to be able to track the shape of internal organs as they move. In these cases *Active Contours*, or Snakes [53, 49] are widely used. While the earlier embodiments of Snakes are essentially intensity gradient-descent methods, contour tracking techniques based on different cues [100], or different computational strategies [24] have also been proposed.

In blob tracking [13, 26, 52, 79], typically a window of fixed size and shape is used to represent the spatial support of the object being tracked. Tracking is then performed to minimize the variation of a number of (typically) statistical measures of the image pixels within the window. Blob tracking techniques are particular useful when the image is noisy or cluttered, for example, when tracking commuters on a subway platform.

While blob tracking techniques are robust and usually allow variations in object appearance, smoothness constraints on statistical measures may not be sufficient in certain cases. In such cases, blob tracking with an object-specific appearance model has been proposed. The *Eigentracking* approach uses a PCA-based appearance model, coupled with a robust matching framework [15, 16, 17, 59], while the *Active Appearance Model* approach allow general shape deformation in the matching process [28, 30, 29].

4.2 Appearance Modeling

In this section we review the basic ideas in appearance modeling, and present the motivations for using faceted models. Typically, an appearance model is constructed from a finite set of samples, and supports a nearest-point query: given an arbitrary new test sample, return the sample from the model that is nearest to the test sample. The simplest example of an appearance model is simply the original set of samples, and the nearest-point query is simply nearest-neighbour search.

Perhaps the most widely used appearance model is the principal components of the test samples. If a set of p appearance samples consists of $n \times m$ images, the set can be written as a $nm \times p$ matrix A, with each column vector being an image scanned in some standard order (for example, row-major order). The principal components of A can be found by SVD, which gives $A = U\Sigma V^T$. The columns of U then contains the principal components of A, corresponding to the singular values arranged in decreasing order. The first t columns of U can be used to form the basis for a subspace, and a new $nm \times 1$ vector e can be approximated as $e^* = \sum_{i=1}^{t} c_i U_i$. If each c_i is computed by $c_i = e\dot{U}_i$, then e^* is the least squares approximation of e [68], and is used as the point returned in a nearest-point query. Typically, the number of principal components is predetermined, and is chosen empirically to balance computational and classification performance. This PCA approach allows for storage compactness and increased generalization power, and has been successfully applied to a number of problems [101, 10].

In PCA appearance modeling, essentially an entire unbounded linear subspace is used to represent the appearance of each object. However this is not always desirable, as the subspace would also contain points that are very far away from the original sample set. [68, 73] proposed fitting a spline to the samples after projecting them onto a lower-dimensional subspace. The spline can then be densely sampled and the resulting set of points can be used in the nearest-neighbour search, thus allowing the manifold to support nearest-point queries at arbitrary precisions. Clearly if we estimate object pose parameters with this approach, the precision is directly related to how densely the spline is sampled. Of course, this can result in a large number of sample points, and consequently a large computational cost in the nearest-neighbour search. However performing PCA helps to lessen the impact, and also clever data structures can be used to speed up the search process [73].

Among the appearance models reviewed thus far, the PCA approach offers the most compact representation, but may suffer from over generalization. While the nearest point query is economical computationally, computing the principal components can be expensive especially with a large sample set. This also discourages dynamic updates to the appearance model, since the principal components need to be recomputed as new samples are added. In contrast, the simple nearest-neighbour approach allows new samples to be added and removed dynamically without any penalties, although the entire set of image samples need to be stored. The precision of the model is also directly related to the sampling density, resulting in an undesirable tradeoff between representational precision and storage requirements. The spline approach addresses the problem with over generalization while also offering better storage efficiency using PCA. The fitted spline also allows the model to support queries at arbitrary precisions. Using PCA however also means that dynamic updates to the appearance model can be expensive, and that the number of principal components need to be predetermined.

In the next section, we will introduce an alternative appearance model that is a hybrid of the PCA approach and the nearest-neighbour approach. The key component of the new model is the explicit specification of adjacency information among samples, which frequently is available in appearance samples. For example, for the set of images of an object placed on a turntable (such as those in the COIL database), successive images are adjacent to one another, with the last image also adjacent to the first to form a ring. A facet grouping rule is also specified that forms multiple subsets of the samples set, each one consisting of samples that are connected via the adjacency relationship. We call these subsets *facets*. Nearest-point querying is then carried out in a two-step procedure: first select a facet that is close to the test sample (by a specified distance measure), and project the test sample onto the linear subspace spanned by the facet samples to give the nearest model point. The subspace projection is similar to the PCA approach, except that the model has a different subspace for each facet. Also, the implicit dimensionality of the facet subspace can also vary among the facets, resulting in a richer representation. There is also no need to pick the number of principal components for use in the model: since a facet is typically a small subset of the entire sample set, we can directly use the facet samples as basis vectors. It is also easy to update the appearance model dynamically since a new sample point would typically be adjacent to only a small number of existing samples, and only a small number of facets are affected.

4.3 Faceted Models

We now give a more formal description of faceted models. A faceted model $M = (F, G, X)$ consists of a set of sample points $X = \{x_1, \dots x_n\}$ and a graph $G = (V, E)$ where $V = \{v_1, \dots, v_n\}$ is the set of vertices, where there is a one-to-one relationship between sample points and vertices. E, the set of edges, has an element for each pair of image samples that are *adjacent* in the appearance set. $F = \{f_1, \dots, f_m\}$ is the set of facets, where each facet is a connected subgraph of G. The facets are formed by specifying a grouping rule. For example, facets can be defined as the individual sample themselves, in which case the faceted model reduces to the nearest-neighbour model. The PCA model can be thought of as a faceted model where the graph consists of a single connected component and there is only one facet which is made up of all available samples. A more interesting example is that each edge in the graph G is a facet, or the edges in G along with samples that are adjacent to the two samples on the edge. In the case where the graph is a triangle mesh, a facet can be a triangle, or a triangle and adjacent samples. It should be noted that facets are not mutually exclusive and do not partition the graph into disjoint pieces. Instead neighbouring facets are usually overlapping.

During the nearest point query, a facet needs to be selected from the model before subspace projection can take place. We use a minimax distance measure for selecting the facet, i.e. for the test sample y, the facet f_k is chosen if

$$k = \underset{i}{\arg\min} \left(\underset{x_j \in \text{facet } f_i}{\arg\max} (\|y - x_j\|) \right)$$

The nearest point is then obtained by projecting the test sample onto the subspace spanned by the samples in the selected facet.

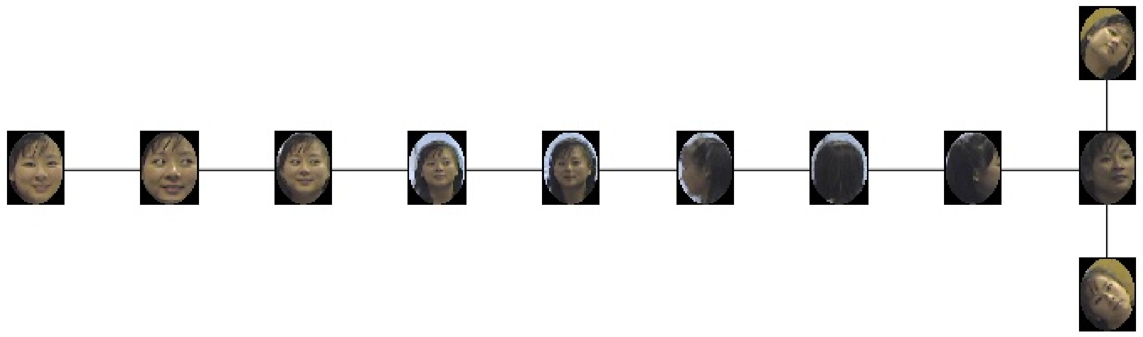


Figure 4.1: Faceted Appearance model for “girl” sequence. The ‘T’ junction captures the appearance variations when the subject tilts her head from side to side.

4.4 Object Tracking Experiments

In order to test the ability of faceted models in appearance modeling, we used it to perform view-based object tracking. In all our experiments, the grouping rule is to form one facet for each edge in the graph, and also adding samples that are immediately adjacent to the two samples on the edge.

In our first object tracking experiment, we use a sequence available from the Stanford Computer Vision Lab [13]. We picked the “girl” sequence as it shows a person moving around and exhibits a wide range of appearance variation. We picked a number of representative appearance samples from the sequence and set up a facet appearance model with topology as shown in Figure 4.1. The samples chosen capture the appearances of the girl at different distances from the camera, and also the views when her back was facing the camera. As can be seen in the figure, the topology of the model has a ‘T’ shape, with the short vertical portion capturing the view variations as the subject tilts her head from side to side.

Figure 4.2 shows the operation at each step. A sliding window is placed over a range of positions, and at each position the pixels within the window is used to form an appearance sample that is then used to perform the nearest point query with a given faceted appearance model. The euclidean distance between the test sample and the model's nearest point is then

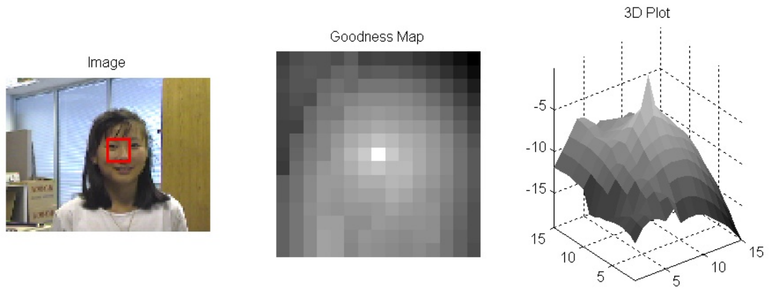


Figure 4.2: Model matching in each step of tracking algorithm. Left image shows the original frame, with a red rectangle indicating the region over which the matching window was centered. The middle image shows the model matching goodness values for the window centered at each pixel position. Lighter values indicate better match. The right image shows the same goodness values, rendered as a 3D plot. It can be seen that the matching goodness peaks at a specifiy point, indicating that in the current frame the object is in the window centered at that point.

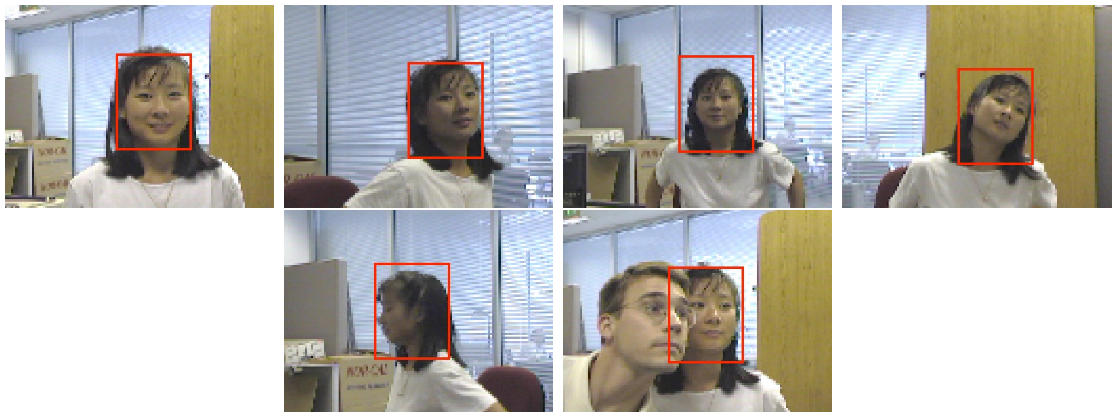


Figure 4.3: Tracking results for “girl” sequence.

a measure of how well the model describes the pixels in each window. The goodness map is then high when the distance is small. As can be seen in the figure, the goodness map peaks at the position where the window encloses the correct object being tracked. This position of best goodness is then returned as the position of the window for the current frame. A search window is then placed in the next frame, centered at the current position, and the same matching procedure is repeated.

The tracking algorithm is initialized by hand with the window position in the first frame. Some of the frames from the tracking results can be seen in Figure 4.3. The algorithm successfully tracked the head of the girl throughout the entire sequence of 500 frames. One of the interesting aspects of using a facet model for tracking is that the chosen facet along with the interpolation coefficients of the facet's samples computed during the nearest-point query provides some pose estimate for the object being tracked. We created an animation (shown in the movie file attached with the paper), where we show a marker indicating the specific part of the faceted model that matched the object's appearance in each coresponding frame by using the coefficients computed to interpolate the coordinates of the nodes of a graph depicting the model topology.

We also tested the algorithm on a number of video sequences captured from broadcast television sporting events, and the results are shown in Figures 4.4, 4.5, and 4.6. In each case, object appearance samples are again chosen manually, and here the faceted model's topology is simply a chain of image samples, in the order in which they appeared in the video sequence. As can be seen, the algorithm successfully tracked the objects even though many of these sequences had targets that exhibit a very wide range of motion and undergo large appearance changes.

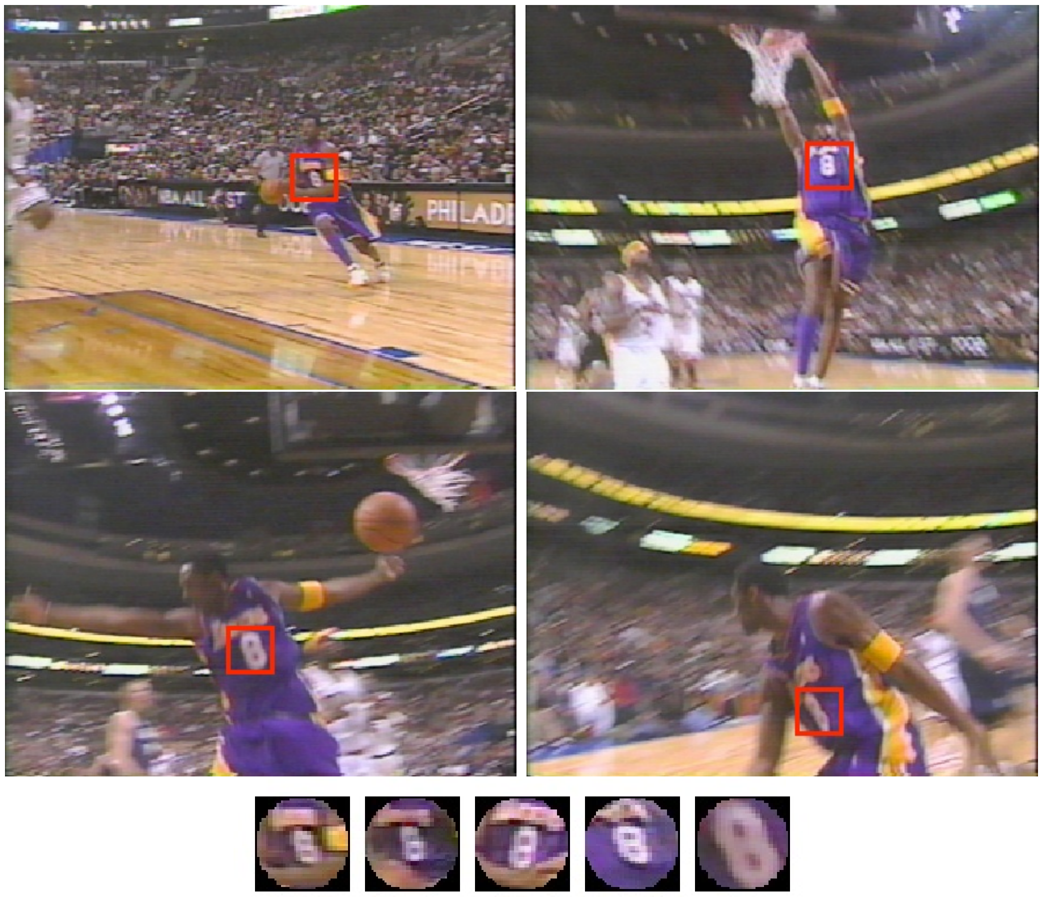


Figure 4.4: Faceted Appearance Model and tracking results for the "dunk" video sequence. The five appearance samples used for tracking are shown at the bottom.

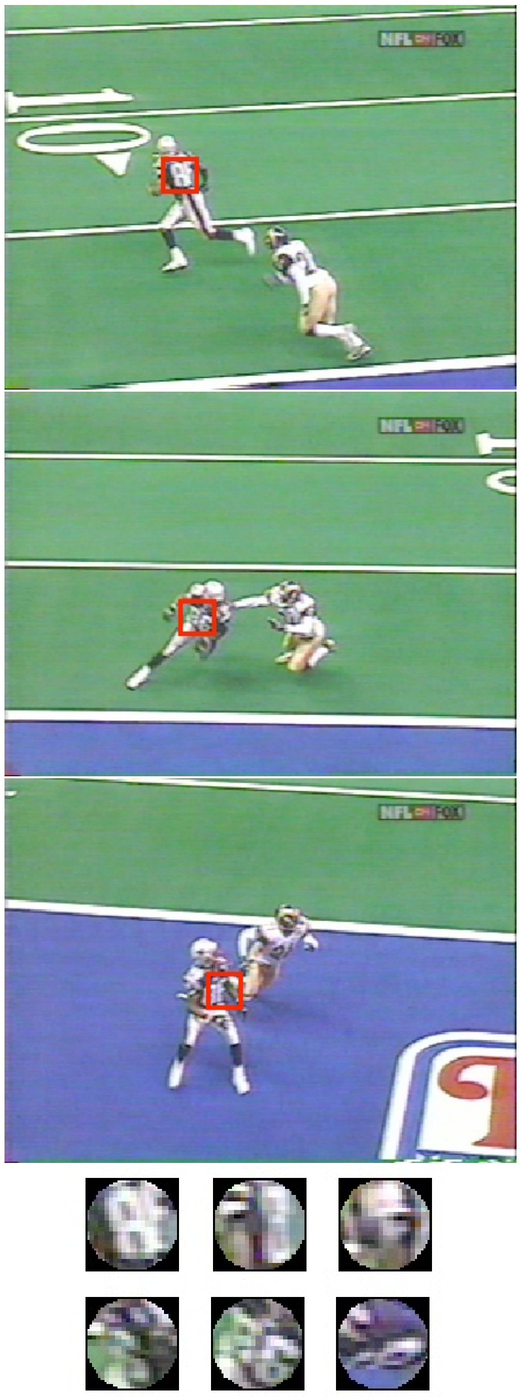


Figure 4.5: Tracking results for the “football” video sequence. The appearance samples used for tracking are shown at the bottom.

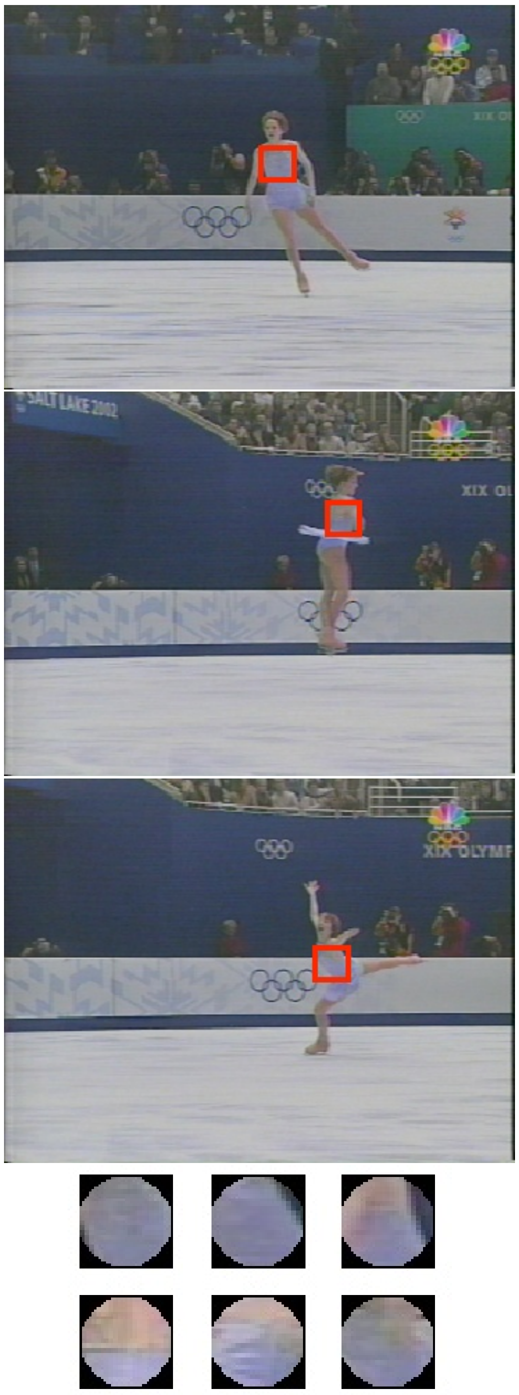


Figure 4.6: Tracking results for the “skating” video sequence. The appearance samples used for tracking are shown at the bottom.

4.5 Conclusions

We have proposed a new, alternative model for object appearances, and tested it by using it to track objects that exhibit appearance variations in video sequences. Based on the experimental results, it is clear that faceted appearance models are capable of capturing the kinds of appearance variations that can be observed in real video sequences. In all the experiments, even though a fairly sparse set of appearance samples were used, the objects were still successfully tracked over fairly long sequences.

Using faceted models to model object appearance offers a number of advantages by exploiting adjacency information that is frequently available in appearance samples. Instead of using a global dimensionality for the model's linear subspace, faceted models take into account the fact that the model subspace dimensionality may vary across local neighbourhoods in the appearance set. Faceted models thus provide a richer representation for the appearance set, and obviates the need to specify the number of principal components to use. Faceted models are also easy to update dynamically, as new samples added will only be adjacent to a small subset of the appearance model, and thus have bounded impact on the existing model.

Chapter 5

Eye Gaze Estimation with Faceted Appearance Models

The ability to *detect* the presence of visual attention from human users, and/or determine what a human user is looking at by *estimating* the direction of eye gaze is useful in many applications. For example if a graphics renderer knows which part of the display the user is looking at, it can adapt itself such that more details are shown where the visual attention is directed. In behavioural studies gaze detection and estimation are also invaluable. For example, the frequency with which a pilot looks at a panel display, and the length of time required to read off the information could be used to measure the effectiveness of the display. Perhaps the application area that would benefit the greatest with the maturation of vision-based gaze estimation techniques would be in human-computer interaction [103, 51, 41]. With the availability of affordable imaging hardware and computational power, eye trackers may well become standard issue on personal computers in the near future.

5.1 Previous Work in Eye Gaze Estimation

Not surprisingly, eye tracking has attracted the interest of many researchers and engineers, and eye trackers have been commercially available for many years. While the most widely used methods today are based on early ideas, newer methods are being enabled by the increase in computational power. For a comprehensive coverage of the earlier literature the reader is referred to [110]. In this survey we would like to cover the methods that have remained popular for completeness, and place more emphasis on promising methods that may lead to the next generation of eye trackers. We thus organize the works studied into the following three categories:

1. **Methods Requiring Physical Contact.** This part of the survey briefly covers some of the earliest eye tracking works, and includes all the eye trackers requiring physical contact with the eye, including contact lens-based techniques.

2. **Non-contact Optical Tracking.** This portion covers most of the techniques that are in widespread use today in commercially-available eye trackers. They typically involve delicate illumination and measurment instrumentation uses a certain specific optical phenomenon to estimate the eye gaze direction. Though more acceptable than the methods that require physical contact with the eye, usually the user's eye and/or head movement is severely constrained when optical tracking methods are employed.

3. **Vision-based Tracking.** This part of the survey covers the more recently proposed computer vision-based approaches to eye tracking that typically attempt to build non-intrusive eye trackers that allow a greater degree of freedom in eye and head movement than are allowed in optical tracking systems, and may lead to eye trackers that are robust enough to be used by casual users, children, and in assistive devices.

We will also examine some proposed applications for eye trackers, and conclude with a discussion of possible novel approaches to gaze estimation.

5.1.1 Methods Requiring Physical Contact

Many of the first techniques for eye tracking requires some device to be in physical contact with the eye or the face. For example, in [34], mechanical linkages to recording pens were attached to the eye directly by a plaster of paris ring. Artificial landmarks have also been

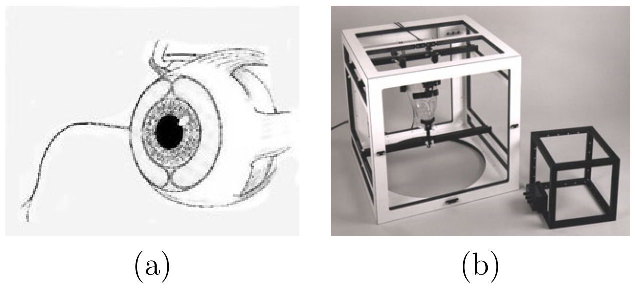


Figure 5.1: Scleral search coil-based eye tracker made by Skalar Medical, a commercially available eye tracker that requires the subject to put on a contact lens with a search coil. (a) Search coil embedded in contact lens. (b) The outer frame that generates the electromagnetic field.

placed on the eye to act as tracking features: a globule of mercury [9], chalk, and egg membrane have all been used for optical tracking. A small piece of metal imbedded in the sclera has also been used for magnetic tracking.

Contact Lens-based Methods

Contact lenses are probably the most acceptable form of attachments to the eye, and there has been contact lens designs with planar mirror surfaces ground onto the lens or attached on a stalk protruding from the lens. The principle non-optical contact lens method is based on the search coil technique [82], which is shown in Figure 2. The human subject wears contact lenses with embedded wire coils, as shown in Figure 1(a), and is surrounded by large electromagnetic coils, as shown in Figure 1(b). The induced voltage in the embedded coils varies with the eye angle relative to the magnetic field and is independent of head position.

Corneo-retina Electrical Potential

Another phenomenon that was used is the potential difference between the cornea and the retina, due to electrical activities in the retina. When the eye moves, current is induced in electrodes placed in a plane perpendicular to the line of sight (typically on the subject's face) can be used to measure eye movement.

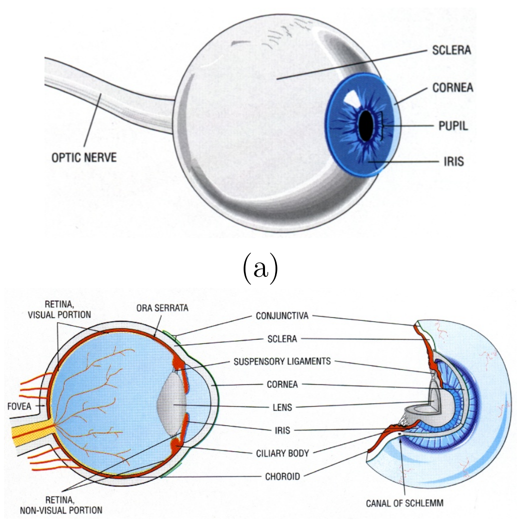


Figure 5.2: Anatomy of the eye. (a) External view. The boundary between the sclera and the iris is called the limbus. (b) Cross section view. (Images from [81])

5.1.2 Non-contact Optical Tracking

In this section we examine the class of techniques that are most widely used in commercially available eye trackers today. These eye trackers typically rely on the precise measurement of certain specific features of the eye. We briefly review the physical characteristics of the eye that we will refer to frequently in subsequent discussions. Figure 2 shows the anatomy of the eye. In Figure 2(a) an external view of an eye is shown. The white portion that forms the majority of the external surface is called the *sclera*, and the transparent bulging portion is called the *cornea*. Behind the cornea is the *iris*, which is of a darker color than the sclera, and in the center of the iris is an opening known as the *pupil*, which allows light to enter the eye. The boundary between the sclera and the iris is called the *limbus*. Figure 2(b) shows a cross sectional view of the eye, and we can see that after light enters the pupil it passes through the *lens* and the *vitreous humor* before reaching the *retina*, which senses the light.

The eye is embedded in a socket and is able to rotate with three degrees of freedom. For most applications it is acceptable to assume that the eye rotates about a fixed center [75].

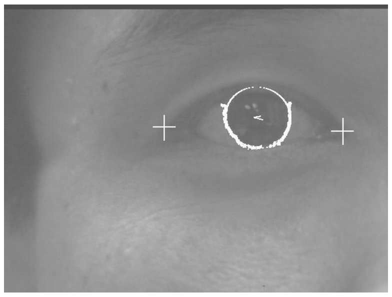


Figure 5.3: Example of eye tracking using the limbus. Image is taken from [102].

The Limbus

The limbus, sometimes called the iris-scleral boundary, is usually the most salient edge in an image of an eye, and is relatively easy to localize. As the limbus can be assumed to approximate a circle, one way to estimate the rotation of the eye using the limbus is to use the shape of the limbus as it appears in an image [102, 55]. If it is observe along the line of sight, the limbus will appear to be a circle. As the observation point moves away from the line of sight, the image of the limbus can be approximated by fitting an ellipsoid and the rotation can be recovered from the parameters of the fitted ellipsoid. Figure 3 shows an ellipsoid successfully fitted to the edge pixels corresponding to the limbus. One problem with limbus tracking is that the limbus can be occluded by the eyelids and usually only portions of it will be visible. This makes the technique sensitive to facial expression variations. Also, as the eye looks away from the camera the cornea itself will start to occlude the limbus.

The Pupil

While the pupil is harder to see under normal lighting conditions than the limbus, especially when the iris is dark-colored, it turns out that it is possible to exploit the "red eye" effect to localize the pupil. This effect is due to the retroreflectivity of the retina: it reflects light most strongly back in the direction where the light came from. As a result if the illumination is near to the axis of measurement, the pupil will appear to be bright, and appears darker than

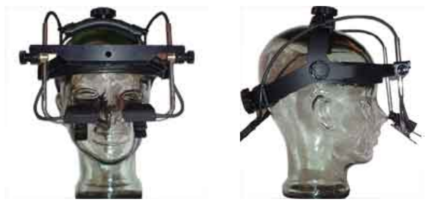
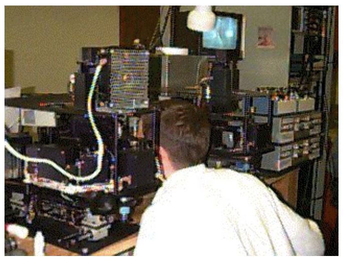


Figure 5.4: The Eyelink II eye head-mounted eye tracker uses corneal reflections to enhance its accuracy.

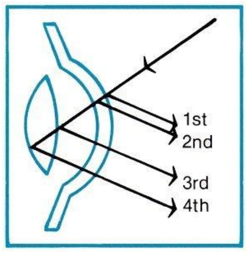
the iris when the light is not coming directly along the optical axis of the camera. Rotation information of the eye can be estimated in a manner similar to that used in limbus tracking. In addition, using the pupil has the advantage that the pupil has a smaller radius than the limbus and is less likely to be occluded by the eye lids. A method utilizing the bright pupil phenomenon was used to create a system that is able to detect multiple faces in real time using active illumination [66].

Corneal Reflection and Corneal-pupil Tracking

The "glint" or the highlight in an eye can also be used for tracking purposes. By observing the position of the reflection of a light source in the front surface of the cornea, it is also possible to estimate the orientation of the eye [38]. More commonly, the corneal reflection is used in conjunction with the location of the pupil to estimate the rotation of the eye [48, 46, 64, 36, 96, 37, 74]. Figure 4 shows a commercially available eye tracker that uses corneal reflections to improve its accuracy. One issue with the use of the corneal reflection is that as an eye rotates, it is possible that the corneal reflection falls on the sclera, that has a surface that is rougher than that of the cornea, resulting in a specular high light that is much larger than the corneal reflection. In addition, as the curvatures of the cornea and sclera are different, the geometric calculations would need to take into account whether a reflection is on the cornea or the sclera.



(a)



(b)

Figure 5.5: Dual Purkinje Image Tracking. (a) The SRI Dual Purkinje Image eye tracker. (b) Cross section view showing how the four Purkinje images are formed.

Purkinje Images

While light entering the eye is reflected to give the corneal reflection, it is also reflected by a series of surfaces as the light travels deeper into the eye. Light is not only reflected from the front surface of the cornea - its rear surface also reflects light to give a second image of the light source. Similar reflections also occur at the front and rear surfaces of the lens, as shown in Figure 5(b). These reflections form what is referred to as the four Purkinje images, and can be used for high-precision tracking [31, 32].

Of the four images, the first (the corneal reflection) and the fourth are the brightest. These two images are useful for deducing the rotation of the eye as they move together when the eye translates, and different when the eye rotates. Currently the most accurate eye tracker that is commercially available is based on Purkinje image tracking. Figure 5(a) shows the the eye tracker in operation. Although the manufacturers claim that the machine is capable of up to 1 minute of arc accuracy, in practice the accuracy is around 0.5 degree [63]. One of the problems with using the fourth Purkinje image is that when the pupil is small, the light ray used to produce the reflections may not be able to reach the lens of the eye. Also, the machine requires the head to be stationary and users have to use a bite board while using the eye tracker.

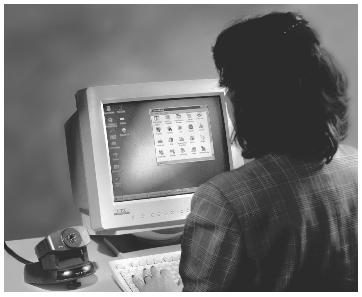


Figure 5.6: The ASL Model 504 eye tracker.

5.1.3 Vision-based Tracking

The third category of eye trackers we study typically aims to allow the user more freedom of movement, and a typical setup is shown in Figure 6: a desk-mounted camera tracks the eye of the user from a distance, and does not require the head to be stationary. The particular model shown, the ASL 504, uses a ring of infrared LEDs to produce a bright pupil image and uses it for tracking. The major advantage is that the user is not required to wear any headmounted equipment, and the tracker allows for one cubic foot's head movement. This makes the tracker suitable for use with a wider range of subjects, including children for example. We believe that ultimately eye trackers will be of this form, and computer vision techniques will be developed that enables accurate and robust eye tracking without depending on delicate optical phenomena such as the Purkinje images.

Simple Models

A simple example of a model-based approach to eye tracking can be seen in Daugman's work [33]. In this application, the machine used for identifying persons needs to search the input image to localize the iris. The algorithm essentially performs a coarse-to-fine searches for a circular contour in the image corresponding to the limbus, and then search for the pupil. In this case the model is simply a circle and a specification of the grayscale variation around the contour. For example, while searching for the pupil (without using the bright pupil method), the algorithm normalizes the grayscale values around the contour so as to bring out the pupil contour.

Deformable Templates

A more elaborate model for the eye was proposed in Yuille's deformable template work [112]. The proposed model explicitly models

1. The limbus as a circle,
2. the eye lids as two parabolic sections,
3. the centroids of the two visible portions of the sclera beside the iris, and
4. the two region of the sclera between the eye lids and below and above the iris.

The limbus circle is attracted to contours that are dark on the interior and light on the outside, and the four scleral centroids are attracted to bright portions of the image. Gradient descent is then used to fit the model to images of eyes. Although gaze estimation was not a goal of the paper, one could use the parameters thus derived to estimate the direction of the eye gaze.

This template was augmented in [95] to account for eye blinks. This eye model is able to transition between the open and close state according to the given input image to produce the best fit.

Other Facial Features

Trackers have also been built that look for other facial features, like the corners of the mouth, corners of the eyes, and corners of the eye brows. In [62], the tracking process runs in two stages, where the facial features are first matched with the input video stream, and using the head pose information derived in the first stage, a second stage then estimates the eye gaze using an eye model to track the limbus. A simpler model was also proposed in [39], where only the facial features were used to compute a *facial normal*, which is then used as a substitute for the gaze direction.

Neural Network-based Methods

A representative work on eye gaze estimation using a neural network-based technique is Baluja and Pomerleau's neural network-based method [8]. In this eye tracker, cropped images of the eyes are used as inputs to a neural network. Training data is collected by requiring the user to look at a given point on a computer monitor and taking a picture of the eye looking at the given point. Thus each labelled training sample consists of the image of the eye and also an (x, y) label of the location on the screen that the user is looking at. In their experiments, 2000 training samples were used to train a multilayer neural network, and the authors reported an accuracy of about 1.5 degrees. Their tracker runs at 15Hz, and allows some head movement. A similar method is documented in [108], which also achieved an accuracy of 1.5 degrees, using 3000 training samples.

5.2 Eye Gaze Estimation with Faceted Appearance Models

Eye gaze estimation can also be treated as a pose estimation problem, which we can solve using an appearance-based approach using Faceted Appearance Models [94]. We build an appearance model with a set of images of one of the user's eyes, and label each image with a ground-truth 2D coordinate of points on a planar display surface. The physical setup for data collection is described in the following section. We also need to specify the topology for the Faceted Appearance Model. In this case, we can use a Delaunay triangulation of the set of 2D coordinate labels to represent the topology: if there is an edge in the Delaunay triangulation between two of the points, then the two corresponding eye images can be considered neighbours on the manifold. For each new image of the eye for which its point-of-regard is to be determined, the nearest point on the Faceted Appearance Model is first estimated. The set of weights used to interpolate the facet images are then used to interpolate the corresponding 2D coordinate labels to obtain the estimated gaze point.

5.3 Experiments

To test the performance of our algorithm [94] in eye gaze estimation, we constructed an image capture system to collect a data set consisting of images of an eye that are labeled with the coordinates of a point that the eye is looking at. The system consists of a computer with a monitor, a monochrome camera, and a framegrabber. The image capture process is as follows: the computer displays a crosshair marker on its screen, and the user is asked to align (by moving a mouse) a cursor (also displayed as a crosshair) with the marker shown, and click on the marker. The next marker is then shown in a different location, and the user repeats the task. Since the user's eye will be looking at the position of the marker on screen when he/she is trying to align the marker and the cursor, images of the user's eye are captured when the cursor and marker are almost aligned. Note that the image capture takes place before the user clicks on the marker, since it is likely that upon completing the task the eye may start looking away in anticipation of the next marker. To further ensure that the user is indeed looking at the marker positions displayed, we pick the positions of the markers randomly while ensuring that at some point the extreme corner coordinates are chosen. If the marker positions can be reliably predicted by the user, it may be possible for the user to align the markers without looking at the markers directly, which would result in erroneous labels for the eye images.

We also mounted an infrared illuminator next to the camera which has a sensor that is sensitive to infrared as well as visible light. This not only ensures that the eye is well lit, it also provides a certain amount of control over the illumination. Typical images can be seen in Figure 5.7(a). It can be seen that the facial region around the eye appears very bright.

One should realize, however, that since the illumination was with infrared light, the user does not see a bright light, which would be very distracting. It should also be noted that the pupil does not exhibit the "red eye" effect since the infrared light was not close to the optical axis of the camera. In fact, the eye turns out to be much darker than the brightly illuminated facial region. We use this effect to find the eye in the image by thresholding the intensity image and identifying the connected dark region that is closest to a predetermined size. For the images shown, the intensity threshold is set at 200 (out of 255), and the region size parameter is chosen to be 4000 pixels. The eye image is then cropped from a rectangular region centered at the spatial centroid of this dark region. An example of the cropped image is shown in Figure 5.7(b). It can be seen that the cropping is different for the three users shown, in that the cropping algorithm also included the eye brows for the second user, and did not include the eye completely in the image. It turns out that this is still adequate for gaze estimation.

5.3.1 Measuring Accuracy

We used a set of 252 images each from three users, and evaluated our system by "Leaving one out": using each one of the eye images as the test image, and the rest of the set to form the appearance manifold. The mean error is computed as

$$\text{Mean error} = \frac{1}{n}\sum_{i=1}^{n} |P_e - P_t|$$

where

$$\begin{aligned} P_e &= \text{estimated position} \\ P_t &= \text{true position} \\ n &= \text{number of test samples} \end{aligned}$$

In this experiment, the eye is 18-24 inches away from the monitor, which displays the markers in an approximately planar, rectangular region of about 12 inches wide and 9 inches tall.

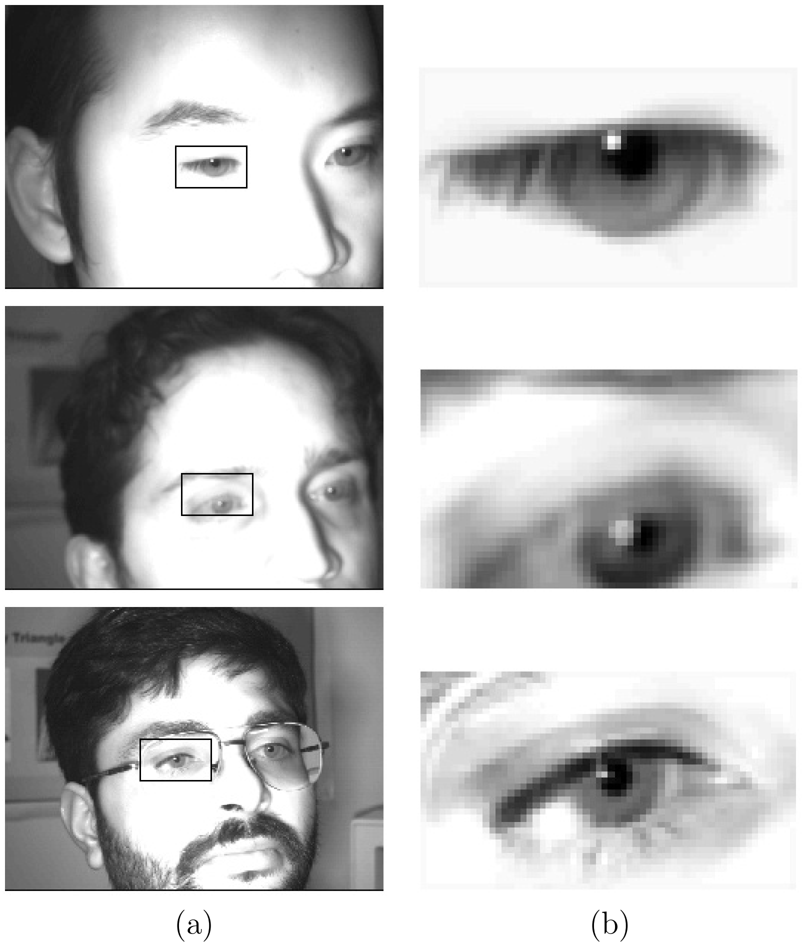


Figure 5.7: Samples of labeled data set collected from three subjects (from top) X, Y, and Z, who are at distances 18, 20, and 24 inches away from the display respectively. (a) Raw captured image. Cropped eye region is shown as an overlaid rectangle. (b) Cropped and scaled image of eye used as appearance sample.

We also assume that the line of sight is perpendicular to the planar region when the eye is looking at its center. We can estimate the mean angular error as follows:

$$\text{Mean angular error} = \tan^{-1} \frac{\text{Mean error}}{\text{Distance from Screen}}$$

Figure 5.9(a) shows the results for Subject X with the entire set of 252 images. The crosses show the real marker positions, and the round dots show the estimated positions, with a line segment joining corresponding crosses and dots. From the scatter plot, we can see that there were some large errors around the periphery. This is to be expected, as the positions were estimated by interpolating marker positions from the appearance manifold, and the test samples on the periphery are essentially approximated by *extrapolating* points that are near the periphery, which yields lower accuracy. We can identify the peripheral test samples as those whose true marker positions form the convex hull of the complete set of marker positions. We also discard the convex hull vertices of the set of points left after removing the first convex hull. For the data set shown, it amounts to removing the set of points on the rectangular border. The average angular errors for the three subjects are tabulated in Figure 5.8. As can be seen, the angular error averaged over all subjects is 0.3838 degrees. The error magnitude distribution for subject X is plotted in Figure 5.9(b). As can be seen, the maximum error is 0.45 inch. From this observation, it is clear that using this gaze estimation algorithm, one can reasonably expect to use eye gaze to guide a cursor to graphical user interface widgets on the monitor that is about a square inch in size.

In general, one would expect the accuracy of the tracking method to improve with more training samples. To validate this, Figure 5.9(c) shows how the angular error decreases as the number of calibrated samples is increased.

Figure 5.10 shows that the accuracy achieved is comparable to those reported by commercially available optical trackers, and is superior to the existing appearance-based methods using neural networks. The same table also shows the number of calibration samples required

Subject	Complete	Non-peripheral
X	0.47853	0.44551
Y	0.43685	0.30534
Z	0.47112	0.4004
Mean	0.4622	0.3838

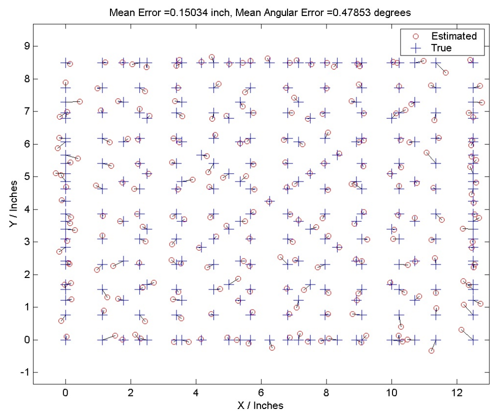
Figure 5.8: Average angular errors for the three subjects, measured in degrees.

for each of the methods. While our method still requires considerably more samples than optical trackers like the Eyelink II, it requires dramatically fewer samples than the neural network-based methods while still achieving superior accuracy.

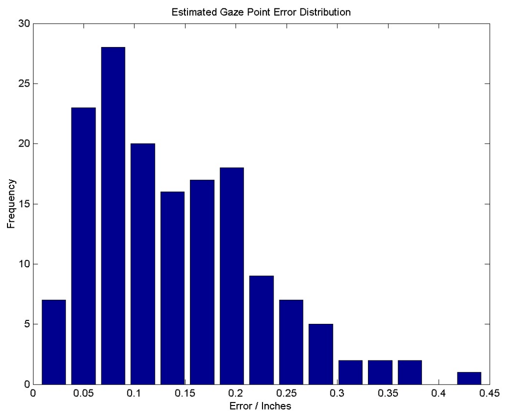
5.4 Discussion

Given that there was no explicit feature extraction and geometric modeling, the 0.38 degree accuracy achieved by the algorithm is rather surprising. The Dual Purkinje Image eye tracker, widely considered the most accurate eye tracker on the market, claims to have an impressive "less than 1 minute of arc resolution" accuracy (http://www.fourward.com). However this is measured with an artificial eye. Vision researchers who have used the Dual Purkinje Image eye tracker reported that its accuracy with human subjects is on the order of 0.5 degree. To explain this apparent discrepancy, we can consider the geometry and resolution of the human eye. Even though visual acuity in the fovea is on the order of 30 arc seconds, the fovea actually covers about 1 degree of the human visual field. When a human eye fixates on a point, the point is projected onto some point within the fovea. It seems unlikely that the point will be projected consistently onto precisely the same spot on the fovea, to within 30 arc seconds. If we assume that a point being fixated upon only needs to fall on the fovea, the eye would only need to come within 0.5 degrees of the 'ideal' gaze angle, which would be consistent with the error margins reported with human subjects on optical eye trackers.

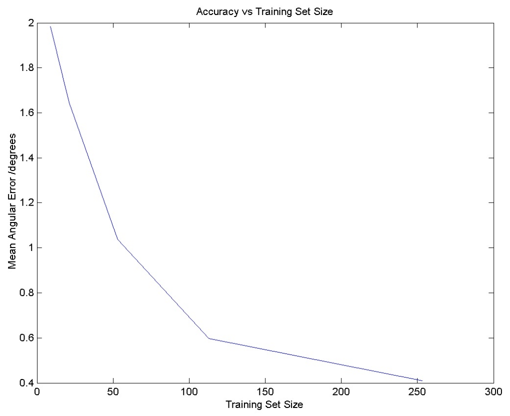
As such, even though it may be possible to measure the orientation of the human eyeball



(a)



(b)



(c)

Figure 5.9: Gaze estimation results analysis for Subject X. (a) Crosses are the ground truth marker locations, and the round dots are the estimated location that the eye is fixated upon. (b) Error magnitude distribution. (c) Average angular error decreases as sampling density increases.

	Accuracy (degrees)	Calibration Samples
Xu-Machin-Sheppard [108]	1.5	3000
Baluja-Pomerleau [8]	1.5	2000
ASL Eyelink II	0.5	9
Appearance Manifold	0.38	252

Figure 5.10: Comparing eye trackers. (Also see main text for a remark on the Dual Purkinje Image eye tracker.)

to a high degree of precision, there is still an error margin between the eyeball orientation and the direction of the true point-of-regard. We thus believe that eye tracker performance with human subjects will be on the order of 0.5 degrees, and the performance of our gaze estimation algorithm is probably near optimal. We attribute the accuracy achieved to a good fit between the problem domain and the appearance manifold model. Recall that one of the fundamental assumptions of the manifold model is that the appearance of the object being modeled should vary continuously with the given parameter. The experimental results we collected indicates that this is indeed the case for the appearance of an eye as it varies its gaze direction.

5.5 Conclusion and Future Work

We have presented a new method for estimating eye gaze direction based on appearance-manifolds, and the degree of accuracy achieved is very encouraging. We have also suggested an enhancement to the appearance manifold method by proposing a nearest manifold point search technique that exploits the topological information inherently present in the manifold model. The degree of accuracy achieved in our experiments is comparable to existing eye trackers, indicating that appearance-based parameter estimation is at least a viable solution to the eye gaze estimation problem.

While the results are very encouraging, there is more work to be done with respect to the eye gaze estimation problem. We would like to model variations in head orientation, and we believe that the current technique should in theory be able to handle head orientation variation. However it would be more difficult to collect a good set of training data that captures variations in head orientation, since it would require the user to repeat the calibration procedure using various different orientation of his/her head. Not only is it strenuous for the user, it is also difficult to measure the true head pose reliably. One possibility is to use a set of cameras mounted on a grid, and capture the appearance of the eye simultaneously from multiple viewpoints.

We are also investigating the integration of the eye tracker into head-mounted displays. Since the appearance-based method does not rely on any specific optical phenomenon such as the corneal reflection or the bright pupil, it allows for more freedom in the placement of the eye tracking camera, and may result in designs that are more compact.

Chapter 6

Video-rate Environment Capture with Mirror Pyramid-based Multiview Panoramic Cameras

Panoramic images and video are useful in many applications such as special effects, immersive virtual environments, and video games. In recent years, the subject has been actively investigated by a number of researchers [11]. Among the numerous devices proposed for capturing panoramas, mirror pyramid-based camera systems are a promising approach for video rate capture, as they offer single-viewpoint imaging, and use only flat mirrors that are easier to produce and introduce less optical aberration than curved mirrors. To date, the designs proposed typically capture panoramas from a single viewpoint. Capturing panoramas from multiple viewpoints with these designs would require either sequentially relocating a single camera system at different viewpoints or employing multiple systems located at all viewpoints which could operate in parallel. Obviously the sequential solution captures inconsistent panoramas when the scene is not static. On the other hand, the parallel solution results in bulky designs as there would need to be one mirror pyramid per viewpoint, but with adjacent viewpoints will need to be separated by a sufficiently large distance. In this paper, we show that it is possible to create multiple viewpoints with only a single mirror pyramid while retaining the ability to capture panoramas at video rates.

6.1 Previous Work

Techniques for constructing panoramic cameras can be classified into two categories: dioptric methods, where only refractive elements (such as lenses) are employed, and catadioptric methods, where reflective components (such as mirrors) are used in combination with refractive elements. Dioptric systems include camera clusters [7, 88], fish eye lens-based systems [65, 107, 115], and rotating cameras [20, 56, 58, 57, 2, 4, 3, 77]. Catadioptric systems include sensors that use curved mirrors and a single camera [21, 22, 42, 43, 50, 70, 72, 109, 114, 78], and sensors that employ planar mirrors and multiple cameras [69, 54, 47, 93, 5].

Dioptric camera clusters, in which multiple cameras point in different directions to achieve a large FOV, are capable of achieving high resolution panoramic video rate capture. However, cameras in these clusters typically do not share a unique viewpoint due to physical constraints, which makes it impossible to mosaic individual images to form a true panoramic view. Although apparent continuity across images may be achieved by ad hoc image blending, panoramas produced as such are not suitable for some machine vision tasks that need images to be captured through a single viewpoint. Systems using a fisheye lens are able to deliver large FOV images at video rate, but have limited sensor resolution as the entire FOV is covered by a single sensor. Fisheye lenses also introduce irreversible distortion for close-by objects and may have different viewpoints for different portions of the FOV. Rotating cameras, in which a conventional camera rotates about its viewpoint to acquire panoramic images, deliver high-resolution wide FOV but are not capable of video rate panoramic capture.

Catadioptric systems that use a curved mirror to map a panoramic view onto a single sensor are able to achieve a single viewpoint at video rate, but have the same limitation on sensor resolution as fisheye lens-based systems. Furthermore, the resolution varies significantly with the viewing direction across the FOV. Similar to the dioptric case, this resolution limitation can be overcome at the expense of video rate capture capability by panning the

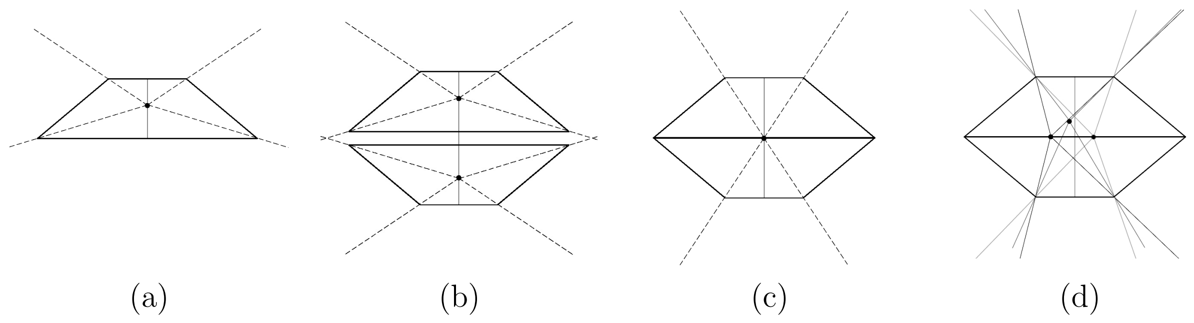


Figure 6.1: The evolution of mirror pyramid cameras. (a) Original design by Nalwa. (b) Stereo design by Kawanishi, vertically stacking two of the cameras from (a). (c) Double vertical FOV design by Hua and Ahuja. (d) Our generalized multiview design, shown with three viewpoints. In general, the design is able to accomodate an arbitrary number of viewpoints placed in arbitrary configurations.

camera system [27, 71, 86] .

A mirror pyramid camera system, first described in [69], consists of a number of flat mirror surfaces arranged in the form of a pyramid together with a set of conventional cameras each associated with a face on the mirror pyramid. These cameras are strategically positioned such that the mirror images of their viewpoints are located at a single point within the mirror pyramid. Effectively this creates a virtual camera with a wide FOV that is capable of capturing panoramas at video rates. The first mirror pyramid camera design locates the viewpoints for the conventional cameras at a point on the main axis of the pyramid, between the apex and the base plane. A recently-proposed camera system [47] uses a double mirror pyramid (two mirror pyramids sharing a common base plane), and locates the viewpoints at the intersection point of the main axis and the base plane. This doubles the vertical FOV. An attempt at creating a stereoscopic (two-view) mirror pyramid camera [54] uses two vertically-stacked mirror pyramids and locates two viewpoints, one in each pyramid, effectively duplicating the arrangement in the first design [69]. Although this creates two panoramic viewpoints, the two views are vertically displaced, i.e., displaced in a direction orthogonal to the panoramic strip. In many applications, it is more useful to have the camera displacement aligned with the direction of the panoramic strip, i.e., conforming to the commonly encountered mode of stereo vision. This type of configuration would be necessary, for example, when the stereo video stream captured is meant to be viewed by a human user.

Figure 6.1 illustrates the previous designs in the evolution of mirror pyramid cameras, and compares them with our multiview design. As far as we know, there do not appear to be any existing mirror pyramid camera designs that allow the capture of horizontal panoramas from multiple horizontally displaced viewpoints. One possible solution would be to have two single-viewpoint mirror pyramid cameras located side-by-side. Alternatively, a single mirror pyramid camera could be relocated to sequentially capture the panoramas at each viewpoint if the scene is stationary. Obviously, the second solution will not be capable of video rate capture, and the first would result in bulkier designs since there would be two mirror pyramids next to each other. More importantly, each mirror pyramid would occlude a part of the other mirror pyramid's FOV.

6.2 Multiview Mirror Pyramid Cameras

In this paper, we propose a mirror pyramid camera design that allows two or more viewpoints to be located horizontally within one mirror pyramid. In addition, the viewpoints can be placed in arbitrary spatial configurations within the mirror pyramid so that, for example, three viewpoints lie in a plane inclined at an arbitrary angle to the base plane, or four viewpoints lie at the vertices of an irregular tetrahedron with arbitrary orientation, or two viewpoints displaced horizontally. Essentially, each viewpoint within the mirror pyramid dictates the positions of a set of conventional cameras around the pyramid. A designer can thus start with the desired spatial configuration of the viewpoints and work out the required configuration of the set of conventional cameras. Video rate imaging can then be achieved simultaneously from all viewpoints. This design was first described in [93].

In the following sections, we will describe the proposed class of mirror pyramid cameras.

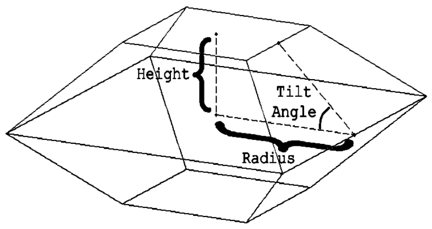


Figure 6.2: The geometry of a mirror pyramid.

We start with a description of mirror pyramids. We then examine the relation between a desired viewpoint inside a mirror pyramid and the positions of the corresponding set of conventional cameras around the pyramid. Subsequent sections show how, for each conventional camera, the focal length and orientation can be chosen to maximize the spatial utilization of each camera's optical sensor. We then discuss the tradeoffs and limitations and show the results obtained from an experimental prototype that uses four conventional cameras to form two viewpoints.

6.2.1 Properties of Mirror Pyramids

We now describe the class of symmetric mirror pyramids considered in this paper, and used either alone or as a part of double mirror pyramids. Any such mirror pyramid can be fully characterized by the following parameters: radius, tilt angle, height, and the number of faces. Radius refers to the perpendicular distance from the main axis to the line of intersection of each planar mirror face with the base of the pyramid. Tilt angle refers to the angle between each mirror face plane and the base plane. If the pyramid is not truncated, all the mirror faces will intersect at the apex of the pyramid. If the pyramid is truncated, then the distance between the truncation plane and the base plane is called its height. Finally, the number of faces refers to the number of mirror faces in a single pyramid (twice as many in a corresponding double pyramid). Figure 6.2.1 illustrates the geometry involved.

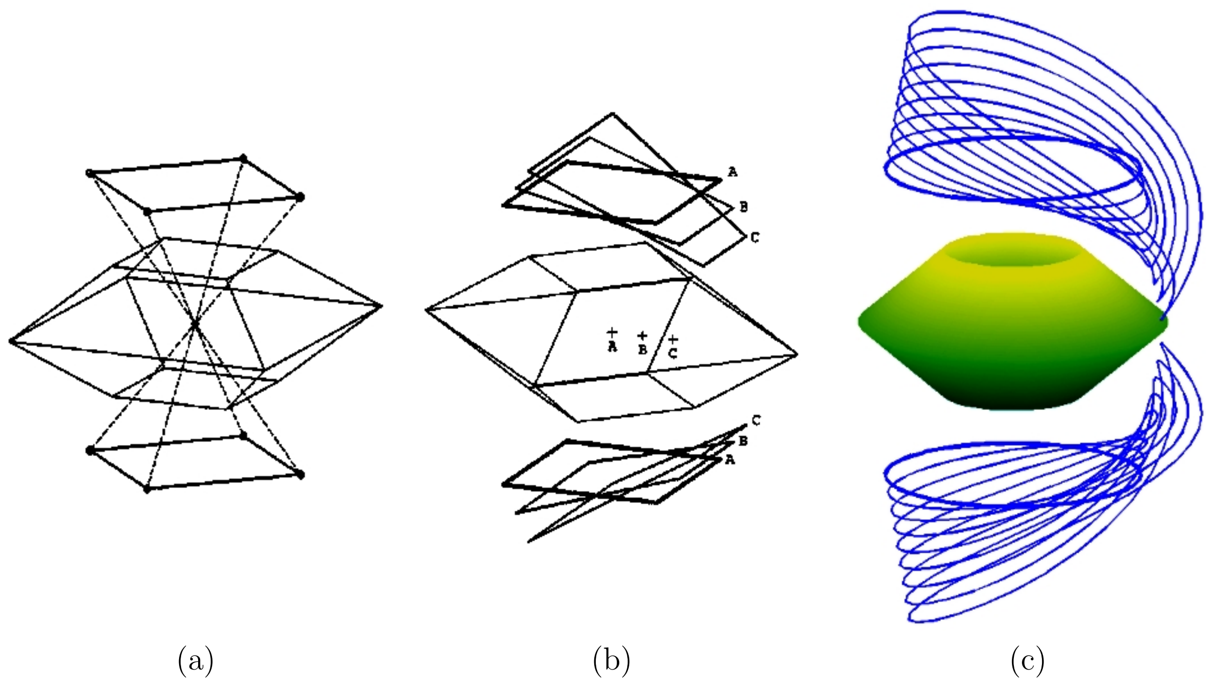


Figure 6.3: Variation in the physical camera position with viewpoint position. (a) Viewpoint is centered in a four-sided pyramid. There are eight mirror faces, and thus four cameras each for the upper and lower pyramids. (b) When the viewpoint shifts from the center, the geometric configuration of the physical cameras changes accordingly. The figure shows camera positions for viewpoints positioned at points A, B, and C. (c) Same as (b), but with a mirror pyramid with a large number of faces, to show how the shape changes as the viewpoint translates from the center towards the edge of the mirror pyramid.

6.2.2 Individual Viewpoint Placement

As mentioned earlier, previous designs of mirror pyramid cameras locate the viewpoint on the axis of symmetry of the pyramid. This viewpoint is placed at the base of the pyramid in a double mirror pyramid, and at a distance away from the base in a single mirror pyramid. In this section we will show how a viewpoint can be placed at an arbitrary location within the pyramid.

This can be easily accomplished by starting with a viewpoint and projecting its image into the physical world by finding the reflections of the viewpoint in the planes containing each respective mirror face. Each such projection is the location of the viewpoint of the physical camera associated with the corresponding pyramid face. An example is shown in

Figure 6.3(a) , in which a four-sided double mirror pyramid is used to create a viewpoint at its center. In the figure, dotted lines join the viewpoint and its corresponding physical camera positions for each mirror face.

When the viewpoint is on the main axis of the pyramid, symmetry causes the positions of the cameras for each of the upper and lower pyramids to form the vertices of a regular polygon. As the viewpoint is shifted away from the center, the polygonal shape changes. Figure 6.3(b) illustrates the effect with the same mirror pyramid as in Figure 6.3(a) , and Figure 6.3(c) illustrates the effect of a shift in the viewpoint for a pyramid with a very large number of faces, and showing how the shape deforms as the viewpoint approaches the outer edge of the pyramid. It can be seen from this last diagram that the initial, almost circular (approaching a circle for an arbitrarily large number of faces) shape smoothly deform into an irregular non-planar shape as the viewpoint shifts away from the center. The practical implication of this observation for camera designers is that if it is necessary for a mirror pyramid camera to change the position of a viewpoint on-the-fly, the camera mounting mechanism would have to take into account this irregular deformation.

6.2.3 Physical Camera FOV Determination

After placing the physical cameras at the locations dictated by the viewpoint in a given mirror pyramid, we need to determine the orientation and focal length of each physical camera, which together with the size of the camera CCD sensor determine the FOV of each camera. The minimum requirement here is for each camera to be able to capture a complete image of its corresponding mirror face. These mirror face images from the cameras can then be mosaiced to form a panoramic image. This requirement however does not ensure optimal use of the available sensors area. The image of a mirror face on a sensor is in general smaller than the sensor, and thus not fully utilize the sensor. In this section, we present a method for finding an optimal orientation that maximizes sensor usage, and the maximum allowed focal length for each camera. The procedure first determinines the camera orientation and

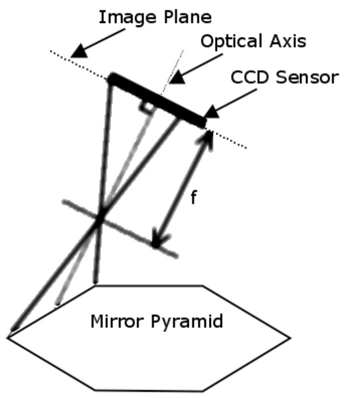


Figure 6.4: The projection geometry of mirror faces onto optical sensors.

then uses the orientation found to determine the maximum allowed focal length. Before we proceed, it is important to note that in this section we are concerned only with the shape and size of the mirrors, and their projection onto camera sensors under perspective viewing transformation. Therefore when we refer to a 'mirror face' and its image on the camera sensor, we are refering only to the shapes and sizes of the mirrors, and not the photometric variations captured on the camera sensors.

Focal Length

Figure 6.4 illustrates the projection geometry of a mirror face onto the sensor plane of a physical camera. The CCD sensor is shown to be a thick black line that is perpendicular to the camera optical axis and parallel to the image plane which coincides with the sensor plane. It can be seen that the sensor captures only a portion of the infinite image plane. In commercially available cameras, the sensor typically covers a rectangular region that is approximately centered at the point where the optical axis intersects the image plane. The focal length, together with the size of the sensor, then determines the effective field of view of the camera. Given a particular orientation and position of a camera and the size of its sensor, we can then find the largest focal length such that the mirror face image is contained in the sensors capture area. Assuming that the sensor is rectangular and its sides are aligned

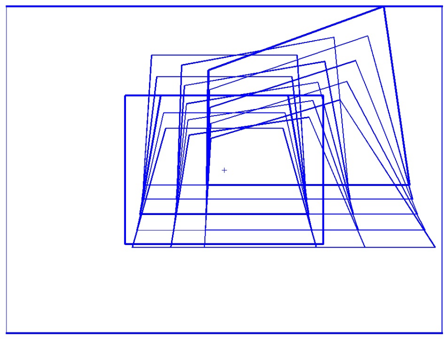


Figure 6.5: Variations in the mirror face image as the camera orientation changes.

with the axes of the frame of reference, the procedure to determine the focal length is as follows:

1. Given a camera orientation, find the image of the corresponding mirror face on the image plane by projecting the four vertices of the mirror face.

2. Find the smallest axis-aligned rectangle on the image plane centered at the optical axis that contains the mirror face image.

3. Find the smallest axis-aligned rectangle with the same aspect ratio as the sensor that contains the rectangle found in the previous step.

4. Find the focal length that makes the rectangle of Step 3 coincide with the sensor.

Figure 6.5 shows an example of the range of mirror face images projected on the sensor plane as a camera's orientation varies. The figure also shows two of the bounding rectangles corresponding to the maximum allowed focal length for the smallest and largest mirror face image. It can be seen that changes in orientation affect the shape, size, and location of the face image within the sensor capture area.

Orientation

It should be noted that the solution for the focal length is uniquely determined for each given orientation and position of the camera, and the size of the sensor. Thus for each orientation and position, we can effectively estimate the maximum possible usage of the sensor by finding the image of the mirror face on the sensor using the maximum possible focal length. Given that each mirror face is a quadrilateral, we can find the projection of the four vertices of each mirror face and compute its area. Now we can search for a set of orientation parameters that makes maximum utilization of the sensor in each camera. We define utilization as

$Utilization = \frac{F}{S}$.

where

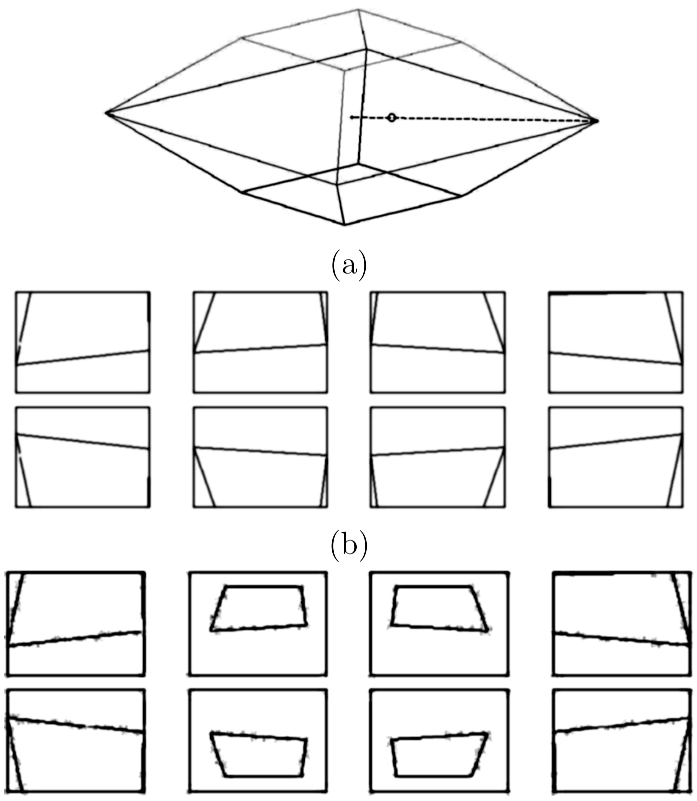
F = area of mirror face image on sensor

S = area of sensor

We then find the orientation for each camera that maximizes its utilization. The results of this optimization is illustrated in Figure 6.5. In Figure 6.6(a) , a mirror pyramid is shown with a viewpoint that is shifted from the center. Figure 6.6(b) shows the results of optimizing the FOV for each camera: each rectangle represents the effective sensor capture area, and contains a quadrilateral which is the image of the corresponding mirror face.

The Uniform-Resolution Constraint

While the sensor utilization in each camera can be maximized by varying the focal length, it is not always desired. One of the drawbacks of optimizing the focal length is that the sensor resolution per unit solid angle now varies among cameras. When the uniform-resolution property is desired, we can simply find the minimum among all the optimal focal lengths found, and use the same focal length for all cameras. The result of imposing this uniform-resolution constraint is shown in Figure 6.6(c) . It can be seen that some of the sensors are not fully utilized. However, now the resolution is constant across the entire panoramic image captured. Alternatively, we can choose the maximum focal length among all the optical focal lengths



(a)

(b)

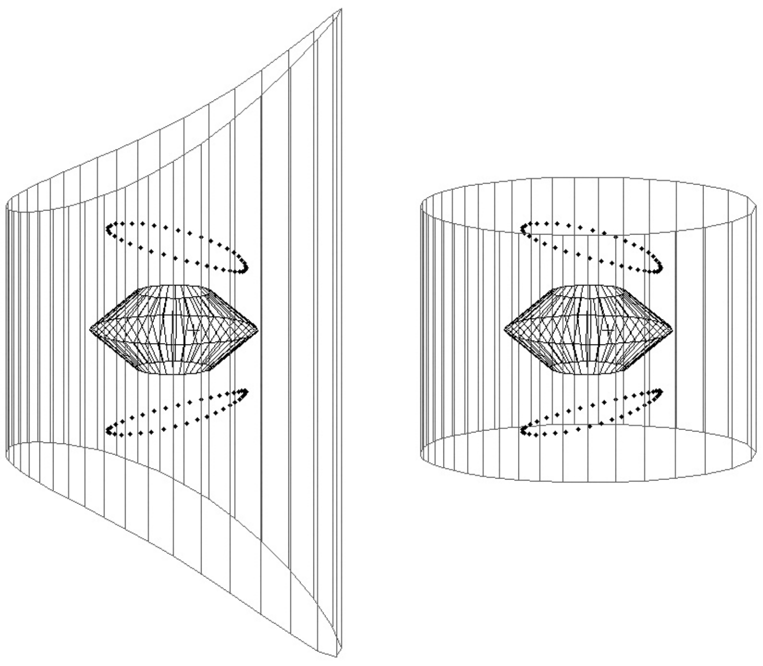
(c)

Figure 6.6: Optimizing the FOV of each camera. (a) Mirror pyramid and viewpoint. (b) FOV optimized for each camera. (c) Enforcing the uniform-resolution constraint by using a constant focal length for all cameras.

found. This results in not only a uniform resolution, but it also imposes a uniform vertical FOV across the panoramic image captured, as shown in Figure reffig:uniform-resolution.

6.2.4 Multiview Setup Considerations

Having described the method for placing individual viewpoints at arbitrary locations within a mirror pyramid, we now discuss the issues involved in the placement of multiple viewpoints within the same pyramid. Each additional viewpoint adds a new set of physical cameras configured by the method discussed in the previous section. A fact of interest to the camera designer is that the locations of the physical cameras corresponding to different viewpoints and a given mirror face are simply the mirror images of the locations of the viewpoints in



(a) (b)

Figure 6.7: The effect of camera focal length choices on vertical FOV, which is shown projected onto a cylinder at a fixed radius from the mirror pyramid's main axis. (a) Variable focal lengths results in a variable vertical FOV. (b) Uniform focal lengths results in uniform resolution and vertical FOV.

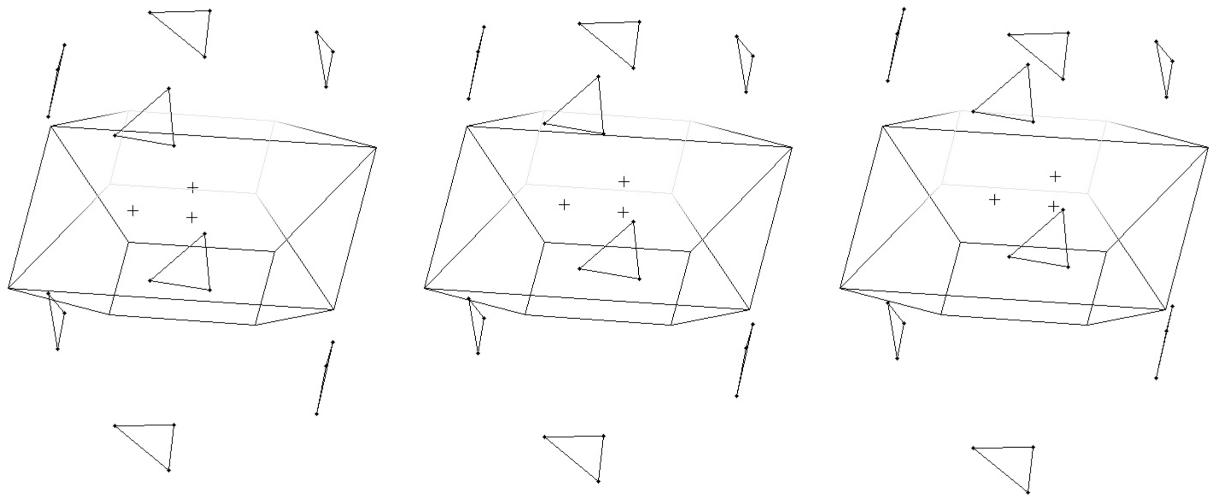
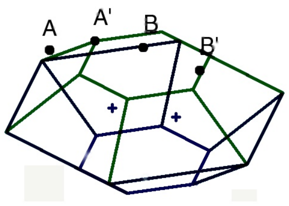


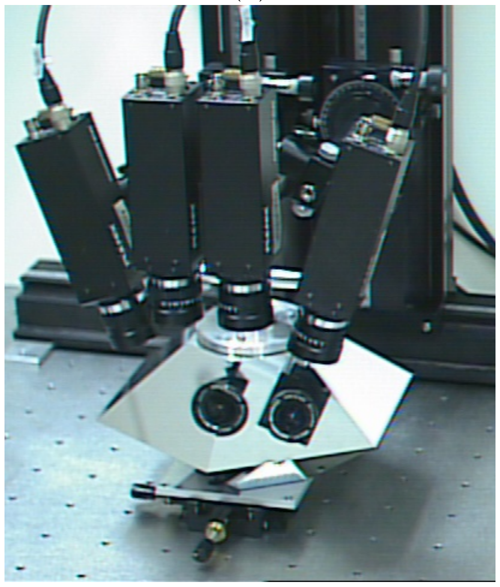
Figure 6.8: Configuration of groups of cameras corresponding to each mirror face remains rigid. The three images shows the effect of translating a group of viewpoints on the positions of the set of physical cameras.

the face. This means that the relative distances and angles among the physical cameras corresponding to a given mirror face are invariant across the different mirror faces. Further, if a set of viewpoints within the pyramid undergos a rigid transformation the corresponding camera configuration also undergoes a rigid transformation which is given by reflections of the viewpoints into the corresponding faces. This is illustrated in Figure 6.8. It should be noted, however, that this invariance property applies only to the camera positions and not the FOV-maximizing orientations.

The number of viewpoints and their spatial configuration will also have another constraint arising from the need to place the physical cameras around the mirror pyramid, which of course will depend on the physical sizes, shapes, orientations and locations of the cameras, and the size and shape of the mirror pyramid.



(a)



(b)

Figure 6.9: (a) A design showing a 6-face pyramid and only two pairs of cameras (AA', BB'). Each pair is associated with one face and the two viewpoints shown as crosses(+). (b) The experimental setup implementing the design in (a).

6.3 Implementation

We implemented a stereo (two-view) mirror pyramid camera system that utilizes two conventional monochrome cameras for each viewpoint, as shown in Figure 6.9. We used an experimental setup that does not allow all the degrees of freedom required for optimized performance, as discussed in previous sections. We had limitations on the achievable camera orientations, and also employed lenses with equal focal lengths on all the cameras. The most significant implication of these limitations is that the usage of the individual sensors may not be optimized, as described in the previous sections. However, the setup proves that it is

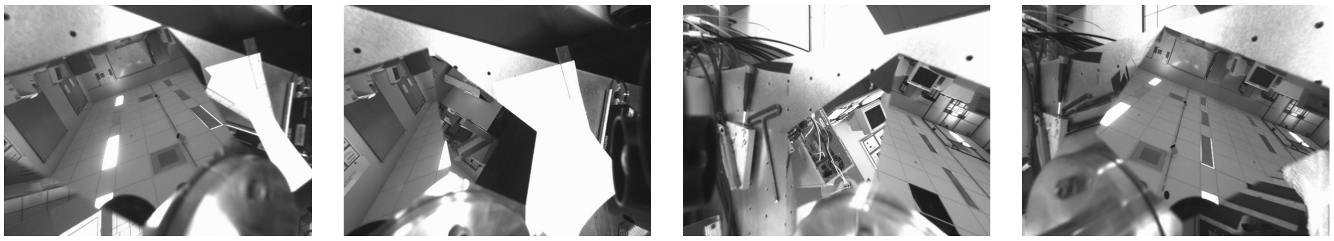


Figure 6.10: The raw images captured by the four cameras, after correcting for radial distortion.

possible to create a mirror pyramid camera with more than a single viewpoint, each located at an arbitrary position within the mirror pyramid. In contrast, all previous mirror pyramid camera designs have only one viewpoint which is limited to the axis of symmetry. As a result, the techniques described in this paper enables the design of a new class of multiview mirror pyramid cameras.

To see that we have achieved a stereo configuration in Figure 6.9, note that the images of the four cameras in the figure form a pair of cameras pointing outwards from within the mirror pyramid. In our experiments, the intrinsic parameters and the radial distortion are estimated and compensated for using the camera calibration software described in [113]. Figure 6.3 shows the images captured by each individual camera (after radial distortion compensation). The mosaiced images are shown in Figure 6.11. It may appear from the experimental results that the setup shown makes suboptimal utilization of the sensors and it might even be possible to obtain the same results using a pair of conventional cameras. However, one should remember that the experimental setup utilizes only two faces of the mirror pyramid and is not fully optimized in the manner described in section 6.2.3. If a full set of cameras were used, even this suboptimal setup would still be able to capture 360-degree panoramas, which is beyond the capabilites of a conventional camera.

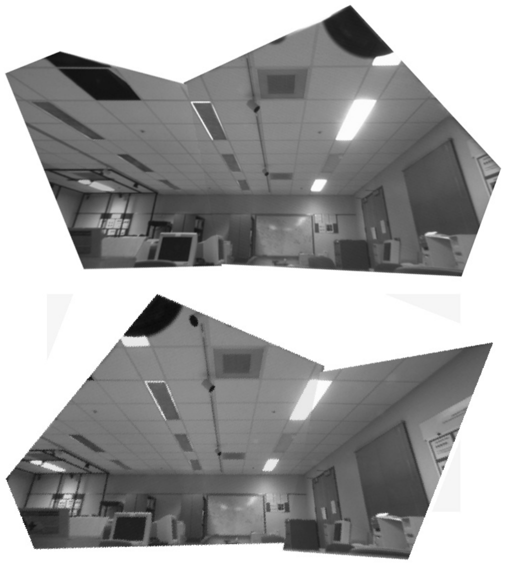


Figure 6.11: The stereo pair of panoramic views captured with the experimental setup.

6.4 Discussion and Future Work

By observing that it is possible to place viewpoints inside a mirror pyramid in arbitrary positions, we have shown how a new class of multiview mirror pyramid cameras can be designed. We have studied the impact of viewpoint shifting on the placement of the conventional cameras around the pyramid, and experimentally demonstrated the feasibility of a two-view mirror pyramid camera. In our ongoing efforts, we are investigating the use of these multiview mirror pyramid cameras in areas like robot navigation and immersive telepresence. In addition, an optical artifact noted in [47] also needs to be addressed.

References

[1] Corel stock photography collection. http://www.corel.com. (the images may also be viewed at http://elib.cs.berkeley.edu/photos.shtml).

[2] M. Aggarwal and N. Ahuja. On generating seamless mosaics with large depth of field. In *International Conference on Pattern Recognition*, 2000.

[3] M. Aggarwal and N. Ahuja. High dynamic range panoramic imaging. In *IEEE International Conference on Computer Vision*, 2001.

[4] M. Aggarwal and N. Ahuja. A new imaging model. In *IEEE International Conference on Computer Vision*, 2001.

[5] M. Aggarwal and N. Ahuja. Split aperture imaging for high dynamic range. In *IEEE International Conference on Computer Vision*, 2001.

[6] Narendra Ahuja. A transform for multiscale image segmentation by integrated edge and region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1211–1235, 1996.

[7] P. I. Anderson. From telepresence to true immersive imaging: into real-life video-now! *Advanced Imaging*, 10(7):48–50, 1995.

[8] Shumeet Baluja and Dean Pomerleau. Non-intrusive gaze tracking using artificial neural networks. *CMU CS Technical Report*, CMU-CS-94-102, 1994.

[9] H. B. Barlow. Eye movements during fixation. *Journal of Physiology*, 116:290–306, 1952.

[10] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman. Eigenfaces vs fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[11] R. Benosman, S. B. Kang, and O. Faugeras, editors. *Panoramic Vision*. Springer-Verlag, 2001.

[12] Alberto Del Bimbo and Pietro Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, 1997.

[13] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1998.

[14] Michael J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1993.

[15] Michael J. Black, David J. Fleet, and Yaser Yacoob. Robustly estimating changes in image appearance. *Computer Vision and Image Understanding*, 78, 2000.

[16] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1), 1998.

[17] Michael J. Black and Yaser Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric model of image motion. In *IEEE International Conference on Computer Vision (ICCV)*, 1995.

[18] Matthew Brand. Morphable 3d models from video. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[19] Matthew Brand and Rahul Bhotika. Flexible flow for 3d nonrigid tracking and shape recovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[20] A. Castano and N. Ahuja. Omnifocused 3d display using the nonfrontal imaging camera. In *Workshop on Computer Vision for Virtual Reality Based Human Comms.*, 1998.

[21] J. S. Chahl and M. V. Srinivasan. Reflective surfaces for panoramic imaging. *Applied Optics*, 36(31):8275–8285, 1997.

[22] J. S. Chahl and M. V. Srinivasan. A complete panoramic vision system, incorporating imaging, ranging, and three dimensional navigation. In *Workshop on Omnidirectional Vision*, pages 104–111, 2000.

[23] Sue Chastain. Knock it out! tools and techniques for removing backgrounds. http://graphicssoft.about.com/compute/graphicssoft/library/weekly/aa000607a.htm, 2000.

[24] Yunqiang Chen, Yong Rui, and Thomas S. Huang. Jpdaf based hmm for real-time contour tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[25] Christophe Chesnaud, Philippe Refregier, and Vlady Boulet. Statistical region snake-based segmentation adapted to different physical noise models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1145–1157, 1999.

[26] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature

space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 2002.

[27] S. Coorg, N. Master, and S. Teller. Acquisition of a large pose-mosaic dataset. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998.

[28] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *European Conference on Computer Vision (ECCV)*, 1998.

[29] T. F. Cootes, G. V. Wheeler, K. N. Walker, and C. J. Taylor. Coupled-view active appearance models. In *British Machine Vision Conference*, 2000.

[30] Timothy F. Cootes, Gareth J. Edwards, and Christoph J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6), 2001.

[31] T. N. Cornsweet and H. D. Crane. Accurate two-dimensional eye tracker using first and fourth purkinje images. *Journal of The Optical Society of America*, 63(8), 1973.

[32] H. D. Crane and C. M. Steele. Accurate three-dimensional eyetracker. *Applied Optics*, 17(5):691–705, 1977.

[33] John G. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1148–1161, 1993.

[34] E. B. Delabarre. A method of recording eye movements. *American Journal of Psychology*, 9:572, 1898.

[35] Richard O. Duda and Peter E. Hart. *Pattern classification and scene analysis.* John Wiley and Sons, Inc., 1973.

[36] Yoshinobu Ebisawa. Improved video-based eye-gaze detection method. *IEEE Transactions on Instrumentation and measurement*, 47(4):948–955, 1998.

[37] Yoshinobu Ebisawa, Koichi Kaneko, Shoji Kojima, Takashi Ushikubo, and Tatsuo Miyakawa. Non-invasive eye-gaze position detecting method used on man/machine interface for the disabled. In *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications*, pages 374–380, 1991.

[38] M. Eizenman, R. C. Frecker, and P. E. Hallett. Precise non-contacting measurement of eye movements using the corneal reflex. *Vision Research*, 24(2):167–174, 1984.

[39] A. H. Gee and R. Cipolla. Determining the gaze of faces in images. *University of Cambridge Department of Engineering Technical Report*, CUED/F-INFENG/TR 174, 1994.

[40] Michael Gleicher. Image snapping. In *SIGGRAPH*, 1995.

[41] Kristen Grauman, Margrit Betke, James Gips, and Gary R. Bradski. Communication via eye blinks – detection and duration analysis in real time. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[42] R. A. Hicks. Reflective surfaces as computational sensors. In *Workshop on Perception for Mobile Agents*, 1999.

[43] R. A. Hicks and R. Bajcsy. Catadioptric sensors that approximate wide-angle perspective projections. In *Workshop on Omnidirectional Vision*, pages 97–103, 2000.

[44] P. Hillman, J. Hannah, and D. Renshaw. Alpha channel estimation in high resolution images and image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[45] B. K. P. Horn and B. G. Schunk. *Artificial Intelligence*, 17, 1981.

[46] Baosheng Hu and MingHua Qiu. A new method for human-computer interaction by using eye gaze. In *IEEE International Conference on Systems, Man, and Cybernetics*, 1994.

[47] H. Hua and N. Ahuja. A uniform-resolution panoramic camera. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[48] Thomas E. Hutchinson, Jr. K. Preston White, Worthy N. Martin, Kelly C. Reichert, and Lisa A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1527–1534, 1989.

[49] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1), 1998.

[50] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):257–262, 1992.

[51] Robert J. K. Jacob. The future of input devices. *ACM Computing Surveys*, 28(4), 1996.

[52] Allan D. Jepson, David J. Fleet, and Thomas F. El-Maraghi. Robust online appearance models for visual tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[53] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. In *IEEE International Conference on Computer Vision (ICCV)*, 1987.

[54] Takahito Kawanishi and et al. Kazumasa Yamazawa. Generation of high resolution stereo panoramic images by omnidirectional imaging sensor using hexagonal pyramidal mirrors. In *International Conference on Pattern Recognition*, 1998.

[55] Kyung-Nam Kim and R. S. Ramakrishna. Vision-based eye-gaze tracking for human computer interface. In *IEEE International Conference on Systems, Man, and Cybernetics*, 1999.

[56] A. Krishnan and N. Ahuja. Range estimation from focus using a non-frontal imaging camera. In *National Conference on Artificial Intelligence*, 1993.

[57] A. Krishnan and N. Ahuja. Panoramic image acquisition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1996.

[58] A. Krishnan and N. Ahuja. Range estimation from focus using a nonfrontal imaging camera. *International Journal of Computer Vision*, 20(3):169–185, 1996.

[59] Fernando De la Torre and Michael J. Black. Robust parameterized component analysis: Theory and applications to 2d facial modeling. In *European Conference on Computer Vision (ECCV)*, 2002.

[60] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[61] Jitendra Malik, Serge Belongie, Jianbo Shi, and Thomas Leung. Textons, contours and regions: Cue combination in image segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 1999.

[62] Yoshio Matsumoto, Tsukasa Ogasawara, and Alexander Zelinsky. Behavior recognition based on head pose and gaze direction measurement. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2127–2132, 2000.

[63] George McConkie. Personal communication. Spring 2002.

[64] John Merchant. Fixation point measurement by the oculometer technique. *Optical Engineering*, 13(4):339–342, 1974.

[65] K. Miyamoto. Fish eye lens. *Journal of Optical Society of America*, 64:1060–1061, 1964.

[66] Carlos H. Morimoto and Myron Flickner. Real-time multiple face detection using active illumination. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, 2000.

[67] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH*, 1995.

[68] H. Murase and S. K. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[69] V. Nalwa. A true omnidirectional viewer. *Bell Laboratories Technical Report*, 1996.

[70] S. K. Nayar. Catadioptric omnidirectional camera. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[71] S. K. Nayar and A. Karmarkar. 360x360 mosaics. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000.

[72] S. K. Nayar and V. Peri. Folded catadioptric cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[73] Sameer A. Nene and Shree K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):989–1003, 1997.

[74] Masao Ohtani and Yoshinobu Ebisawa. Eye-gaze detection based on the pupil detection technique using two light sources and the image difference method. In *IEEE Engineering in Medicine and Biology Society Annual International Conference*, 1995.

[75] R. S. Park and G. E. Park. The center of ocular rotation in the horizontal plane. *American Journal of Physiology*, 104, 1933.

[76] Elin R. Pedersen, Kim McCall, Thomas P. Moran, and Frank G. Halasz. Tivoli: An electronic whiteboard for informal workgroup meetings. In *Proceedings INTERCHI*, 1993.

[77] S. Peleg. Panoramic mosaics by manifold projection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[78] S. Peleg, M. Ben-Ezra, and Yael Pritch. Omnistereo: Panoramic stereo imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):279–290, 2001.

[79] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision (ECCV)*, 2002.

[80] Thomas Porter and Tom Duff. Compositing digital images. *Computer Graphics*, 18(3), 1984.

[81] Thomas R. Quackenbush. *Relearning to See.* Frog, Ltd., 1997.

[82] David A. Robinson. A method of measuring eye movement using a scleral search coil in a magnetic field. *IEEE Transactions on Bio-medical Electronics*, BME-10:137–145, 1963.

[83] Mark Ruzon and Carlo Tomasi. Alpha estimation in natural images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[84] Eric Saund and Thomas P. Moran. A perceptually-supported sketch editor. In *Proceedings UIST*, 1994.

[85] Eric Saund and Thomas P. Moran. Perceptual organization in an interactive sketch editing application. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1995.

[86] H.-Y. Shum and R. Szeliski. Panoramic image mosaics. *Microsoft Research Technical Report*, MSR-TR-97-23, 1997.

[87] Alvy Ray Smith and Jim Blinn. Blue screen matting. In *SIGGRAPH*, 1995.

[88] R. Swaminathan and S. K. Nayar. Nonmetric calibration of wide-angle lenses and poly-cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10), 2000.

[89] Richard Szeliski and Heung-Yeung Shum. Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1199–1210, 1996.

[90] Mark Tabb and Narendra Ahuja. Multiscale image segmentation by integrated edge and region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(5):642–655, 1997.

[91] Kar-Han Tan and Narendra Ahuja. A representation for image structure and its application in object selection with freehand sketches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[92] Kar-Han Tan and Narendra Ahuja. Selecting objects with freehand sketches. In *IEEE International Conference on Computer Vision (ICCV)*, 2001.

[93] Kar-Han Tan, Hong Hua, and Narendra Ahuja. Multiview panoramic cameras using a mirror pyramid. In *IEEE Workshop on Omnidirectional Vision*, 2002.

[94] Kar-Han Tan, David J. Kriegman, and Narendra Ahuja. Appearance-based eye gaze estimation. In *IEEE Workshop on Applications of Computer Vision*, 2002.

[95] Ying-Li Tian, Takeo Kanade, and Jeffrey Cohn. Dual-state parametric eye tracking. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, pages 110 – 115, 2000.

[96] Kiyoaki Tokunou and Yoshinobu Ebisawa. Automated thresholding for real-time image processing in video-based eye-gaze detection. In *Proceedings of the 20th Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society, number 2, 1998.

[97] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2), 1992.

[98] Lorenzo Torresani and Christoph Bregler. Space-time tracking. In *European Conference on Computer Vision (ECCV)*, 2002.

[99] Lorenzo Torresani, Danny B. Yang, Eugene J. Alexander, and Christoph Bregler. Tracking and modeling non-rigid objects with rank constraints. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[100] Kentaro Toyama and Andrew Blacke. Probabilistic tracking in a metric space. In *IEEE International Conference on Computer Vision*, 2001.

[101] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.

[102] Jian-Gang Wang and Eric Sung. Study on eye gaze estimation. *IEEE Transactions on Systems, Man, and Cybernetics*, 32(3), 2002.

[103] Colin Ware and Harutune H. Mikaelian. An evaluation of an eye tracker as a device for computer input. In *Proceedings of the ACM Conference on Human Factors in Computing Systems and Graphics Interface (CHI/GI) 1987*, pages 183–188, 1987.

[104] Yair Weiss. Segmentation using eigenvectors: a unifying view. In *IEEE International Conference on Computer Vision (ICCV)*, 1999.

[105] Y. Wexler, A. Fitzgibbon, and A. Zisserman. Bayesian estimation of layers from multiple images. In *European Conference on Computer Vision (ECCV)*, 2002.

[106] Donna J. Williams and Mubarak Shah. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, 55(1):14–26, 1992.

[107] Y. Xiong and K. Turkowski. Creating image-based virtual reality using a self-calibrating fisheye lens. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[108] Li-Qun Xu, Dave Machin, and Phil Sheppard. A novel approach to real-time non-intrusive gaze finding. In *British Machine Vision Conference*, 1998.

[109] K. Yamazawa, Y. Yagi, and M. Yachida. Omnidirectional imaging with hyperboloidal projection. In *International Conference on Intelligent Robots and Systems*, 1993.

[110] Laurence R. Young and David Sheena. Methods and designs: Survey of eye movement recording methods. *Behavior Research Methods and Instrumentation*, 7(5):397–429, 1975.

[111] Yung yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[112] Alan L. Yuille, Peter W. Hallinan, and David S. Cohen. Feature extraction from faces using deformable templaces. *International Journal of Computer Vision*, 8(2):99–111, 1992.

[113] Zhengyou Zhang. A flexible new technique for camera calibration. *Microsoft Research Technical Report*, MSR-TR-98-71, 1998.

[114] Z. Zhu, K. D. Rajasekar, E. M. Riseman, and A. R. Hanson. Panoramic virtual stereo vision of cooperative mobile robots for localizing 3d moving objects. In *Workshop on Omnidirectional Vision*, 2000.

[115] S. Zimmerman and D. Kuban. A video pan/tilt/magnify/rotate system with no moving parts. In *IEEE/AIAA Digital Avionics Systems Conference*, 1992.

Vita

Kar-Han Tan was born in Singapore, where he grew up and attended the National University of Singapore. He graduated with a B.S. degree from the School of Computing, was placed on the Deans List in his final year and selected for the Honors class.

Prior to joining UIUC, he received his M.S. degree from the University of California at Los Angeles, where he worked at the Cooperative Mobile Robots Laboratory and co-invented the Virtual Structures paradigm for high-precision robot formation control that has since been employed by groups at the Jet Propulsion Laboratory in multi-body spacecraft experiments. He has also contributed to the graphics rendering engine of a flight simulator for the solar-powered glider project at UCLA.

While pursuing his doctorate at the University of Illinois, Kar-Han was a member of the Beckman Institute Computer Vision and Robotics Laboratory, and was responsible for designing and building the control system which enabled the Beckman Institute Hexapod robot to walk for the first time. The system is used to study biologically-inspired legged locomotion. Kar-Han was awarded the Beckman Institute Graduate Fellowship for his thesis work on object selection from images and video, and has received numerous inquiries and requests for source code from Kodak R & D, Microsoft Research, and Adobe. In between academic years, Kar-Han has worked as a Software Engineer at Teledyne Controls (West Los Angeles, CA), and as a Visiting Research Scientist at Vision Technology (Champaign, IL) and Sarnoff Corporation (Princeton, NJ).