# UI-Wand Related Theories and Approaches Investigation

**Literature Survey**

Authors: Wei Huang, Qing Xia[1]



[1] Authors: W. Huang and Q. Xia are with the Department of Media and Knowledge Engineering, Technology University of Delft, Netherlands. This literature survey is a part of their master thesis project, UI-Wand, in Philips Lab, Aachen, Germany.

Their Email address: {w.huang-ti, q.xia}@student.tudelft.nl

# Acknowledgements

It is always true that people need some others' help when they are facing some troubles. As students, the biggest trouble they suffered is glaring at the problem definition and a white empty word file, and do not know how to input the first word.

It happened to us without an exception.

When we got our thesis topic, our minds are in blank. Even for this literature research work, we have no idea how to start it, which kinds of paper should we study in this phase. Not only because of our limited knowledge on this filed, but also because the project is a totally new idea for this world and no experience to follow. This is a bad news for us, but it is a good news on the other hand, because this means we are touching the real research work, and that means we have a chance to create something.

After several months of hard work and papers study, we found a solution for our task and summarized some useful papers in this literature survey.

During this time, besides the hard tries of ourselves, we will appreciate our supervisor, Mr. Jan Kneissler, who is very patient and gave us many suggestions and guides from practical aspect and theory aspect. Without his help, maybe we are still struggling in reading some deep theory paper or wondering among huge number of algorithms.

Another person, we would like to thank is our professor Leon Rothkrantz. He is a very nice man. Almost in every party of our classmates, we were talking about his enthusiastic and friendly help to us. He gave us lots of suggestions on writing this literature survey and many hints on the real system. His advices let us write the first word of this paper, so we must say "Thanks" to him.

# Abstract

UI-Wand is a new project in Machine Interface Group of Philips. Its concept for future is to use a stick attached with a camera on the top to recognize the objects in your home and control them by doing some gestures with it. But for its current version, Philips want to make it control some applications running on some devices with a screen, like Computer, Laptop, or TV. So the basic requirement for current UI-Wand control is the screen and point positioning and gesture recognition.

This literature survey is an investigation to the theories and approaches that might be used in UI-Wand. Due to that UI-Wand is a new project and its idea is also new, there were no previous tries that we can study. Therefore, this paper does not focus directly on the topic or some model theory, instead we figured out and studied a set of possible theories or approaches that different with each other, but could be used solely or combined together later in the system.

So the paper we studied and summarized in this survey consists of several parts talking about different theories, they are corner detection approaches, classification and regression models, tracking and gesture recognition methods. In every part of this paper, we summarized the basic idea of each paper, and give a comment below it. And for the theory or algorithm to solve the same problems, we did some comparisons between them so that we can get a rough idea about their performances.

**Index Terms** – Corner Detection Algorithms, Relevance Vector Machine, Support Vector Machine, Hidden-Markov Model, Tracking Algorithm, Gesture Recognition

# Contents

# Chapter 1

# 1 Problem Definition

## 1.1 What is UI-Wand?

Did you notice the wand in the hand of magician? It is a fantastic stick that can control everything he wants as his will. Philips's new project, UI-Wand, with full name "User Interface Wand", just came from such an idea. Its concept is to use a camera as a control and let users control every object in their home just by aiming at it, and then doing a gesture. For example, you can control the TV volume by just aiming at TV and turning it or you can change the current channel to next one by moving the wand a little right and back, and more you can control the menu displaying on TV, even play some games. For the future, it can control other objects like windows, to open and close, your digital camera, to download pictures, and etc. UI-Wand is a powerful control using different technology to be adaptive for various objects. Since it just starts, the real UI-Wand has a long way to go. The UI-Wand mentioned in our papers is just for the screen detection and positioning.

## 1.2 Problem statement

Considering the current requirements and feasibility, Philips defined the current UI-Wand as a control for personal computer, laptop, or television set. It should recognize where the users are pointing at and should recognize the gestures the users are doing and execute the corresponding action related to those gestures.

The possible applications for UI-Wand is a Philips medical application control, a presentation slides control or a TV control. As a medical application, it should control the menu and can rotate, move or do the other adjustment for the medical model in the application. For the presentation slides control, it can control the slides to go next or go back and some other functions like drawing a curve or dragging an icon from one place to another. As a TV control, the UI-Wand has some functionalities such as to control menu, change channel, volume, mosaic choices and so on. Although the specific applications are different, the kernel technical requirements for UI-Wand are the same, which are screen positioning and gesture recognition, which are also our tasks to be finished in our master thesis.

## 1.3 Requirement of UI-Wand application

According to above problem statement, there are some requirements of current UI-Wand listed below:

- Recognize screen only
- Realize screen positioning and simple gesture recognition
- It can be used for some specific applications
- Operate easily
- The speed should be in real time level
- The technique embedded in should be extensive and easy to integrate into real system.

So this literature research paper is an investigation to the current techniques and approaches that might be used in the current UI-Wand.

## 1.4 The organization of this investigation

Because UI-Wand is a new project in PHIILPS and its idea to recognize screens by camera also did not be tried before by others. Even though some companies have made similar mouse products that can control presentation slides remotely by doing some gestures, but that is by using mechanical sensors methods, which essentially is different with UI-Wand.

So the difficult of this system is that there are no existing papers for the similar project that can be referred directly. Considering this fact, we did not put our literature research strictly and directly on our topic, UI-Wand. We analyzed the whole system, got a possible solution, and studied the possible theories, approaches that might be used later. The rough UI-Wand system structure we designed is shown right side (See Fig.1.1). From the figure, you can briefly get our approaches to detect screen and position. That is to use corner detection method to detect 4 screen corners in a 2 dimensional space and then figure out the real 3D pointer positioning. Finally, according to the positioning track history it works out the possible gestures the user is doing.

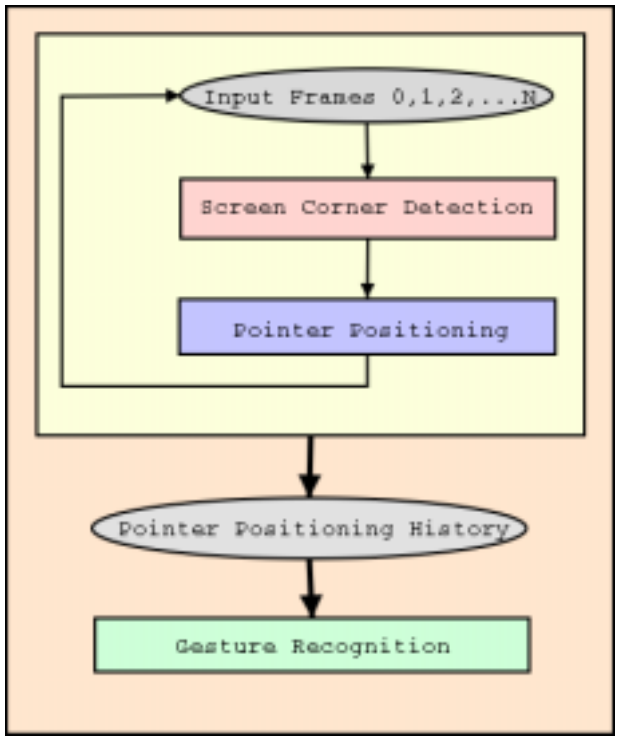



Figure 1.1 Rough structure of UI-Wand

Because of the specific models we are in charge of study are screen positioning and gesture recognition, so our literature survey just focuses on two models, “Screen Corners Detection” and “Gesture Recognition”.

We organized our survey as following:

There are many different approaches to recognize corners, like image processing methods, to analysis gradient values curves or geometric properties. Or the pattern recognition methods, like classification. The latter techniques can be used in gesture recognition too. So in Chapter 2, we find out and compare some image processing techniques on corners detection and edges detection that can detect screen directly by gradients values in an image. In Chapter 3, many pattern recognition techniques like classifications and regression models are investigated and compared, which will be very useful for the corner and gesture recognition models. In Chapter 4, we give comparison to some existing gesture recognition methods and tracking approaches,

which could be helpful for UI-Wand gesture model and screen corners tracking problem. Considering the requirements of this system, this literature survey includes different type of papers like theoretical papers, literature survey, algorithm papers, application papers, or even some books. Every paper has its features that give us many hints for our future study. So In the last chapter, besides summary of this literature survey, in order to let users get a brief overview of these papers we make a table to summarize them.

# Chapter 2

# 2 Corner Detection Approaches

With respect to its practical applications, the feature detection has already been studied insensitively for more than three decades. Among many kinds of features, one-dimensional features such as edges, and two-dimensional features such as corners and junctions are more popular than others. Detection of these low-level features in images is also a classic problem in computer vision since it is believed that these features can be detected without knowledge of the object in the world that caused them. From these features people can get the geometric property of any objects, which is very useful for image analysis, object recognition, motion tracking, or the other applications. Many different models have been proposed, some to detect one of these features, others to detect all of them. Some detectors are designed for real-time applications, while others are more complex. Because our project is mainly about the screen corner detection, so in the following part, we will mainly focus on the corner detection. But we will still mention other features that also detected by these algorithms if they are related to the corner detection.

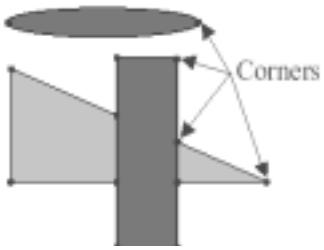



Figure 2.1 Normal corners

By the term *corner*, it means the point at which the direction of the boundary of object changes abruptly. The object is a continuous image area with a constant (or nearly constant) brightness or color. We can then define the corner as an intersection point between two or more edge segments; Fig.2.1 shows some examples of the common corners. And in the real images the corners can be different kind of shapes; Fig 2.2 shows the zoom-in corners in the real images. And Fig 2.3 shows a brightness function $b(x, y)$ in a neighborhood of a simplest L-corner.

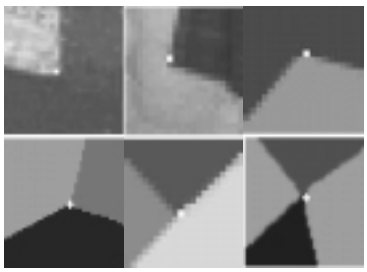


Figure 2.2 Corners in the real images

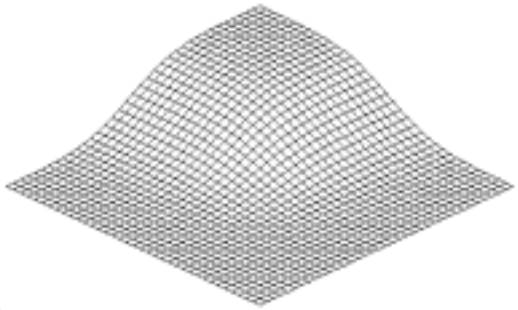


Figure 2.3 The brightness function $b(x, y)$ in a neighborhood of a simplest L-corner.

In the rest of this chapter we will briefly characterize the main approaches that appeared in the history of the corner detection. We will give some comments for every detector that we have introduced. Finally in the conclusion part of this chapter we will use many comparing table to show the different performance of all these detectors.

## 2.1 Gradient approaches

### 2.1.1 Beaudet's detector

**[Bea78][Dre82][Der93]**

This detector is among one of the oldest direct corner detector. Beaudet presented it in his paper [Bea78]. The author used $b(x,y)$ to denote the brightness function of the image and proposed to compute the quantity $Det(x,y)$ for each image point $(x,y)$. The definition of $Det(x,y)$ can be shown as following equation:

$$Det = b_{xx}b_{yy} - b_{xy}^2 . \tag{2.1}$$

where $b_{xx}, b_{yy}$ and $b_{xy}$ are the second partial derivatives of the brightness function and $b$ stands for the brightness function $b(x,y)$. Then whether an image point belongs to the corner points can be decide by the value of its $Dev$. Each image point at which the value of $Dev$ exhibits its positive local maximum and is greater than a predefined threshold $s$ is marked as a corner point. The quantity of $Dev$ is connected with the Gaussian curvature of the surface of the brightness: $z = b(x,y)$. The Gaussian curvature is defined by the equation $K = k_1 k_2$ where $k_1$ and $k_2$ are the principal curvatures of the surface $z = b(x,y)$ at the point to be processed. Evaluating the principal curvatures yields the result:

$$K = \frac{b_{xx}b_{yy} - b_{xy}^2}{\left(1 + b_x^2 + b_y^2\right)^2} . \tag{2.2}$$

It can be seen in above Eq. (2.2), the numerator of the equation is equal to Beaudet's $Dev$. And it is also very clear that the denominator is always be positive, so the sign of the value of $Dev$ is always the same as the sign of the Gaussian curvature. Thus the value of $Dev$ can be used to classify the points lying on the brightness surface (see table 2.1).

Table 2.1 Points classification

| Point category | Value range |
|---|---|
| Elliptic point | $Dev > 0$ |
| Parabolic point | $Dev = 0$ |
| Hyperbolic point | $Dev < 0$ |

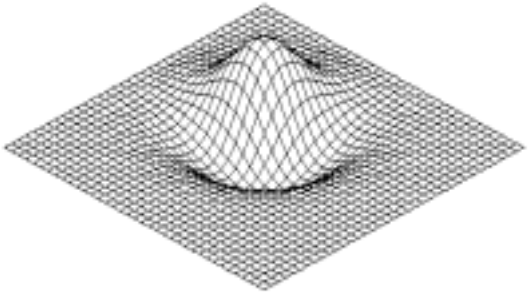


Figure 2.4 The response provided by Beaudet's $Dev$ operator.

This method proposed by Beaudet detects the elliptic points with a locally maximal value of Gaussian curvature. It should be remarked that $Dev$ could be interpreted as the Hessian determinant. If we depict the function $Dev(x,y)$ for the corner defined by Eq. (2.2) (the shape of the corner is shown in Fig.2.3 ), the principle of Beaudet method is illustrated by Fig.2.4. It can be seen that

*Dev* gives two local extremum for a corner, a negative extremum for the hyperbolic point and a positive extremum for the elliptic point. According to the extreme value, the corner point can be found respectively.

Later on there are several other authors who also proposed different algorithms based on the Gaussian curvature principle or directly on based on Beaudet's operator *Dev*. For example, Dreschler and Nagel [Dre82] proposed an operator based on the curvature but not use the Beaudet operator. The different way is that they select the locations of the corners with locally extreme Gaussian curvature and pair each maximum elliptic point $E$ (the point at which the Gaussian curvature exhibits its positive local maximum) with a maximal hyperbolic point $H$ in its neighborhood (the point where Gaussian curvature has its negative local minimum). Then the corner is detected on the line connecting $E$ and $H$, at the point where one of the principal curvatures is equal to zero. In Fig.2.5, this kind of pair is depicted clearly. The authors used several straightforward rules for finding such kind of pairs:

1. One of the principal curvatures should have approximately the same size and direction at both points that from the pair.
2. For the second curvature, the same is required. In addition, the curvatures should have the opposite signs at both points.
3. The points should be closed enough, here the author used value of 4 pixels.

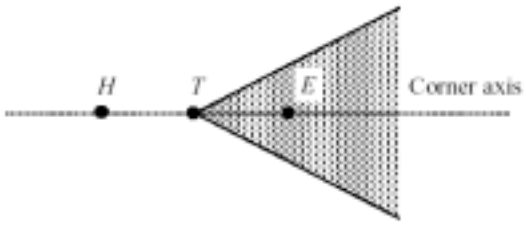



Figure 2.5 Method proposed by Dreschler and Nagel

Deriche and Giraudon presented another different approach [Der93]; they use quite similar idea with the above methods. In their algorithm, the authors convolve the input image with the Gaussian function. They do this kind of convolve twice with two different values of $\sigma$. In this way they get two blurred images, it is essential that the degree of blurring in both images is significantly different. Then they use Beaudet's corner detector method to find corners in both blurred images. When each corner in one image is detected, its corresponding corner is determined in the other image, here they used nearest criterion to decide the correspondent corners. The authors proposed to use the "spiral searching technique" and the space in which the corresponding corner is searched for is limited to a certain small searching window, here they used $7\times7$ pixels window. Then each pair of corners (a corner from one image and its nearest corresponding corner in the other image) defines the axis of a corner in the original input image. For example, in Fig. 2.6, the theoretical corner point *X*. (The corners *X*1 and *X*2 detected in the images with different level of blur determine the corner axis (*dotted line*). The theoretical corner point is determined by passing along

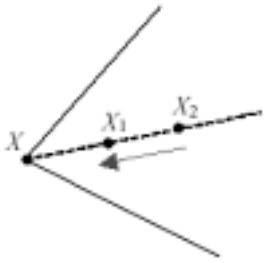



Figure 2.6 Searching for the real corner

the corner axis in the direction of the arrow and by searching for the zero-crossing location of the Laplacian of the original input image.

#### 2.1.1.1 Comments

These papers are all related to Beaudet's corner detection algorithm, so they can be included in the more or less the same kind of approach. But they still have difference in the way to deal with the operator and also data. Each method is based on the former method but gives some kind of improvement comparing to the former one. In paper [Bea78], the author did not present the mathematical analysis of the Beaudet detector in his original work, but we can find some of the explanations in [Der93] and [Roh94]. From the papers we can see that *Dev* gives two local extreme for a corner (a negative extreme for the hyperbolic point and a positive extreme for the elliptic point), and that is exactly the reason to explain why a localization error occurs if the corners are detected simply as the points with the local extreme *Dev* value. It is due to the fact that the elliptic maximum point always lies inside the corner, which can be seen in Fig.2.3 and Fig.2.4. In [Roh94], the localization error of the detector was analyzed very exactly and it was shown that it depends on the corner angle. Deriche and Giraudon studied the known corner detectors and revealed that most of them detect the corners with more or less significant localization errors. They tried to remedy the problem by constructing their own algorithm in [Der93]. They observe that Beaudet's operator always gives the maximal responses on the corner axis inside the corner and the size of the localization error increase with the increasing blurring of the image. In the other paper [Soj02a], the authors mentioned their own experience with Deriche and Giraudon algorithm, this algorithm works well especially on simpler images in which the pairs of corresponding corners may be found reliably. If an image is to be processed containing areas with dense occurrence of corners, the corresponding pairs of corners cannot usually be found reliably, which yields completely false detections.

### 2.1.2 Kitchen and Rosenfeld's detector

**[Kit82][Zun83][Bru92][Wan95]**

Kitchen and Rosenfeld presented their corner detection almost the same period as Beaudet in [Kit82]. Their method is based on edge detection and they evaluated the change of the gradient direction along the edge contour and multiplied this quantity by local gradient magnitude. They used $\varphi(x, y)$ as the function to describe the gradient direction along the edge. Let

$$\varphi(x, y) = \tan^{-1}\left(\frac{b_y}{b_x}\right). \tag{2.3}$$

It is very clear that if passing along the edge, the gradient direction must be almost constant, but in the case that passing edge contour through the corner, the gradient direction must change. The fact whether the gradient direction changes along the edge can be examined by computing the derivative of $\varphi(x, y)$ in the direction of edge. From Fig.2.7, the idea can be illustrate clearly.

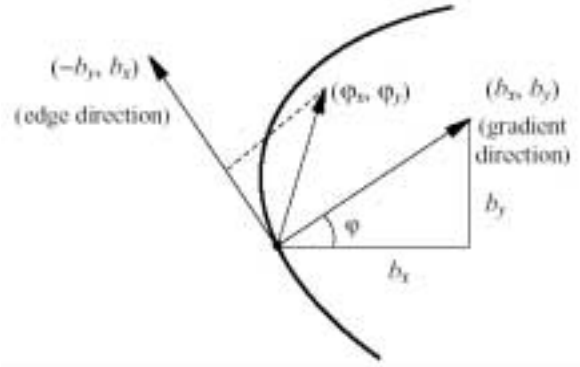



Figure 2.7 Kitchen and Rosenfeld's method

Then the authors proposed to multiply this derivative by the local gradient magnitude. We can denote Kitchen and Rosenfeld's corner response function by $C_{KR}$, since the edge, which is perpendicular to the gradient, its direction can be expressed by $(-b_y, b_x)$, and then we can get

$$C_{KR} = \frac{(\varphi_x, \varphi_y)(-b_y, b_x)}{|(-b_y, b_x)|}|(b_x, b_y)| = (\varphi_x, \varphi_y)(-b_y, b_x) = \frac{b_{xx}b_y^2 + b_{yy}b_x^2 - 2b_{xy}b_xb_y}{(b_x^2 + b_y^2)}. \quad (2.4)$$

Because the gradient direction may be increase or decrease along the edge, the value of $C_{KR}$ may be either positive or negative correspondently. Therefore, both the local positive maxim and the negative minim of $C_{KR}$ can be determined. It can be concluded that the value of $C_{KR}$ is equal to the second derivative of the gradient of the brightness function in the direction perpendicular to the gradient. We can also depict the response provided by Kitchen and Rosenfeld's operator for the corner in Fig.2.8 from Fig. 2.3.

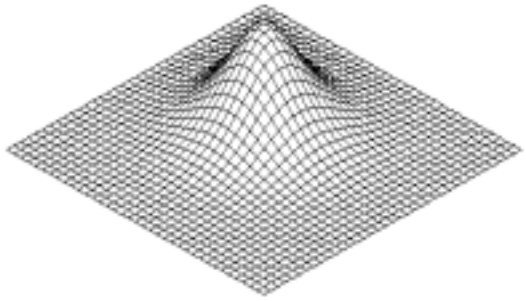


Figure 2.8 The response provided by Kitchen and Rosenfeld's operator.

Later on, Brunnstrőm et al. [Bru92] proposed to use only the numerator of Eq. (2.4) as their corner response function $C_{BR}$, and $C_{BR} = C_{KR}|\nabla b|^2$. In their approach the corner with high gradient magnitudes are thus preferred.

Wang and Brady presented another approach in [Wan95] in the similar idea. They proposed to use the corner response function in the form of

$C_{WB} = \left(\partial^2 b / \partial t^2\right)^2 - S\left|\nabla b\right|^2$, where $t$ is the direction perpendicular to the gradient and $S$ is a constant. They use $S\left|\nabla b\right|^2$ to reduce the false responses.

There are also several other authors presented the modifications of Kitchen and Rosenfeld's approach for corner detection. For example, Zugina and Haralick [Zun83] presented a very similar with Kitchen and Rosenfeld, but they used only the curvature of the edge contour without multiplying it by the gradient magnitude $\left|\nabla b\right|$. Thus their corner response function $C_{ZH}$ is tightly connected with the function $C_{KR}$ by the equation $C_{ZH} = C_{KR} / \left|\nabla b\right|$. , Zugina and Haralick also used the facet model in their approach. In the neighborhood of the point being examined, they modeled the brightness function by the bi-cubic polynomial surface.

#### 2.1.2.1 Comments

Kitchen and Rosenfeld mentioned fitting a function of two spatial variables to the gray-level values in image, and determining the value of $C_{KR}$ by analytic means, which should reduce the influence of noise. The authors typically chose a function that was a polynomial of a low degree (usually 2 or 3) that fits the gray-level data in a small neighborhood (from $3\times3$ to $7\times7$ pixels) with minimal sun of squared errors. In Kitchen and Rosenfeld's version of their algorithm, however, does not seem to be realized too frequently since it is computationally expensive. At this point, many other authors even do not mention this fact. Instead, the noise reduction is often achieved by convolving the image with the Gaussian function, which is carried out before the corner detection. The derivatives in Eq. (2.4) are then replaced by the differences. Later on in [Soj02a], the author figured out that the algorithm in Kitchen and Rosenfeld's version tended to give too many false responses if only the simple detection of maxima/minima of $C_{KR}$ in a window of a certain size was used. The algorithm can be improved in such way that in the $C_{KR}$ image, the continuous areas where the value of $C_{KR}$ is greater or less than a threshold are detected. If the size as the number of pixels of such an area lies in a predefined range, then we choose the extremum in the area that is declared to be a corner. By this kind of improvement the performance of Kitchen and Rosenfeld's algorithm should be better than the original one.

### 2.1.3 Noble's detector

**[Nob88]**

Noble proposed a method in [Nob88] that detects the corners at places where the gradient magnitude and the principal curvature are high. He defined the corner response function as following equation:

$$C_N = \sqrt{g_{11}g_{22} - g_{12}^2}\,\frac{\left|k_1\right| + \left|k_2\right|}{2}. \tag{2.5}$$

where $g_{11}, g_{12}$ and $g_{22}$ are the coefficients of the first fundamental form of the surface that is defined by the brightness function $z = b(x, y)$ and $k$ , $k_2$ are the

principal curvatures of this surface. Here it is worth mentioning that, except the absolute values, the second term in Eq. (2.5) resembles the so called "'mean curvature"', which is the simple average $((k_1 + k_2)/2$. The principal curvature and, therefore, also the sum of their absolute values are high in the corner neighborhood. It can be easily shown that the following equations hold $g_{11} = 1 + b_x^2$, $g_{22} = 1 + b_y^2$, $g_{12} = b_x b_y$, which yields

$$\sqrt{g_{11}g_{22} - g_{12}^2} = \sqrt{1 + b_x^2 + b_y^2}. \tag{2.6}$$

From the differential theory of surface, it is also known that the expression:

$$\sqrt{g_{11}g_{22} - g_{12}^2}\, d_x d_y \tag{2.7}$$

which determines the size of the area on the surface $z = b(x, y)$ over the rectangle $d_x d_y$. This explains the meaning of the first term in the right part of Eq.(2.5), which is the size of the area of the brightness surface over a square that corresponds to a point. Fig, 2.9 a) shows when this size is the lowest possible if the brightness is constant, and Fig. 2.9 b) shows in all other cases that this size must necessarily be higher then the former situation.

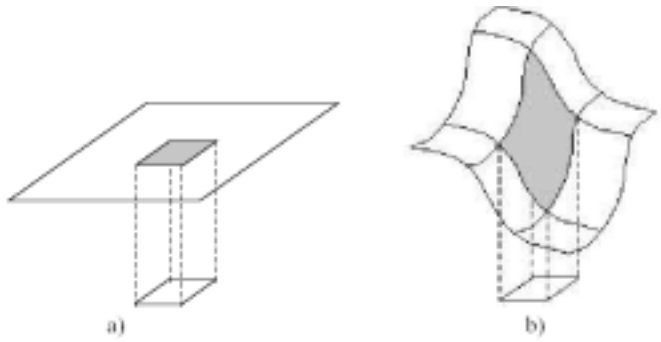



Figure 2.9 The size of the area of brightness over a unit square expressed by $\sqrt{g_{11}g_{22} - g_{12}^2}$ in Eq. (2.5)

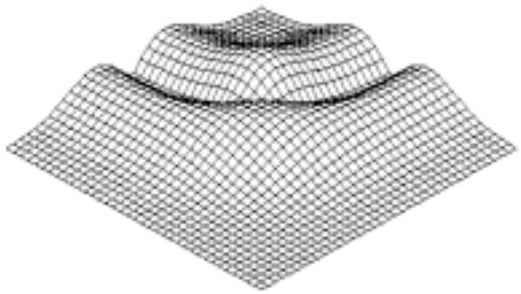


Figure 2.10 The response provided by Noble's operator

The values of $C_N$ for the corner from Fig. 2.3 are shown in Fig. 2.10. As can be seen from the image that the value of $C_N$ are high not only in the neighborhood of the corner but also in the neighborhood of the edges connecting in the corner. And also the same time, two maxima values are obtained in the neighborhood of each corner. In Noble's algorithm, it then combines computing the value of $C_N$ with classification of image points using the coefficients of the second fundamental form of the brightness surface. Only very strong hyperbolic points are accepted as corners.

#### 2.1.3.1 Comments

This paper proposes a solution to the 2D corner and junction feature problem. Although this algorithm will isolate image L-junctions, its performance cannot be predicted for T-junctions and other higher-order image structures. Instead, an image representation is proposed that exploits the local differential geometrical "topography" of the intensity surface. Theoretical and experimental results are presented which demonstrate how idealized instances of 2D surface features such as junctions can be characterized by the differential geometry of a simple facet model. He also presents the needed equations for the Chebychev facet model over a $3\times3$ neighborhood. But later on according to the other authors such as [Soj02a], Noble's method is not an efficient one.

### 2.1.4 Harris and Stephens' detector

**[Har88][Tra98][Zhe99]**

In [Har88] Harris and Stephens described well-known Plessey feature point detector. The computations in his algorithm were entirely based on the first-differential quantities. The detector computes the derivatives $b_x, b_y$ using $n\times n$ first-difference approximations (where $b_x = \frac{\partial I}{\partial x}, b_y = \frac{\partial I}{\partial y}$ and $I(x,y)$ is the gray level intensity of the image point). Using the Gaussian smoothing kernel $G(x,y)$ of a standard deviation $\sigma$, the algorithm then continues with computing the sampled means $\langle b_x^2 \rangle$, $\langle b_y^2 \rangle$, and $\langle b_x b_y \rangle$. Let the value of $\langle b(x,y) \rangle$ defined as following:

$$\langle b(x,y) \rangle = b(x,y) * G(x,y). \tag{2.8}$$

For each image pixel we have such matrix to be considered:

$$M = \begin{bmatrix} \langle b_x^2 \rangle & \langle b_x b_y \rangle \\ \langle b_x b_y \rangle & \langle b_y^2 \rangle \end{bmatrix}. \tag{2.9}$$

Then the two eigenvalues $\mu_1$ and $\mu_2$ of matrix $M$ can be used to decide the ratio of the cornerness. Let the ratio to be $C_P$ then we can get the corner response function (here we use CRF to denote the corner response function) as following:

$$C_P = \frac{Trace\ M}{Det\ M} = \frac{\langle b_x^2 \rangle + \langle b_y^2 \rangle}{\langle b_x^2 \rangle \langle b_y^2 \rangle - \langle b_x b_y \rangle^2} = \frac{\mu_1 + \mu_2}{\mu_1 \mu_2}. \tag{2.10}$$

If at a certain pixel point the two eigenvalues of the matrix $M$ are sufficiently large, then a small motion in any direction will cause an important change of gray level. This indicates the point is a corner. A point is marked as a corner if the value of $C_P$ is small (which is a locally minimal).

Later on Harris and Stephens presents a modified version of the Plessey detector, they change the corner response function into the following expression:

$$C_{HS} = Det\, M - kTrace^2 M \,. \tag{2.11}$$

They set the value of $k$ to 0.04 and they explained the goal of the second term in Eq. (2.11) is that it provides the discrimination against the high-contrast pixel step edges. The value of the corner response function $C_{HS}$ for the corner from Fig.2.3 can be depicted in Fig.2.11. From the figure we can see that the valleys in the edge area are caused by the second term on the right side of Eq. (2.11).

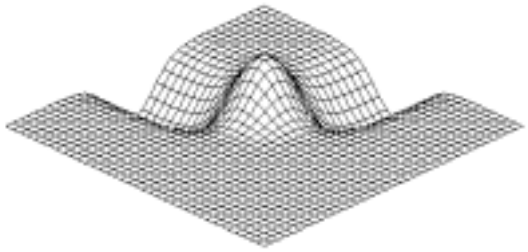


Figure 2.11 The response provided by Harris and Stephens' operator

In 1998, Trajkovič and Hedley presented a way to reduce the computational complexity of Plessey detector in [Tra98]. In their paper they propose a modified version of Harris algorithm that achieves almost the same performance as the original algorithm, but has a much lower computational cost. In addition to a high CRF, it also requires a pixel to have high image gradient in order to be a comer candidate. For each pixel they first compute the image gradient, and if it is lower than some threshold it is not necessary to evaluate the computationally expensive CRF. Since for most real images only 10-20% of image pixels have a high gradient, we do not need to compute the CRF for the majority of pixels. The drawback is that the Gaussian convolution now cannot be fully decomposed. Nonetheless, a speed up factor of two to three is obtained, depending of content of the image.

In 1999, another detector inspired by Harris and Stephens' algorithm was proposed in [Zhe99] by Z.Zheng, H.Wang and E.K.Teoh. In this paper the analysis of gray level corner detection has been carried out. Performances of various cornerness measures are discussed with respect to four performances of robustness: detection, localization, stability and complexity. They have analyzed the interior differential features of the image surface of these cornerness measures. This paper presents a new approach called gradient-direction corner detector for the corner detection, which is developed from the popular Plessey corner detection. The gradient-direction corner detector is based on the measure of the gradient module of the image gradient direction and the constraints of the false corner response suppression.

#### 2.1.4.1 Comments

Several different authors in their papers nicely explained the Plessey algorithm, but unfortunately the analysis is too long to be outlined clearly in the original paper. Similar with Harris and Stephens' approach, different authors use different corner response function know as CRF to decide the cornerness of the point. Before Harris

and Stephens, Förster used $C_P = \frac{Det\,M}{Trace\,M}$ as the CRF, Harris and Stephens then used $C_P = \frac{Trace\,M}{Det\,M}$ as the CRF, later on they improved their algorithm and used $C_{HS} = Det\,M - kTrace^2 M$ instead of the $C_P$ that they presented in earlier time. Recently there are also lots of new corner detectors focuses on Harris and Stephens' algorithm but giving some new modified speeding up the algorithm and improving the performance of Harris and Stephens' Plessey algorithm.

Harris and Stephens' corner detector with the corner response function (CRF) defined by Eq.(2.8) is commonly regarded as very good performance in the sense of reliability and stability of detection [Ale02]. A certain drawback is that it may potentially be slower than other detectors, which is mainly due to the fact that the Gaussian smoothing must be computed three times during the processing phase. But this problem has already been resolved in Trajkovič and Hedley's paper. They reduced the time complexity by excluding the pixels with low values of gradient magnitude from computation. Later on Z.Zheng, H.Wang and E.K.Teoh derived a certain approximation of the Harris corner response function Eq. (2.11) and introduced $k$ as a function $k(x, y)$. Their method requires only one use of Gaussian smoothing. But they reported slightly worse successfulness of detection than in the case of the Harris and Stephens' detector, better localization of corners, and better running times.

### 2.1.5 Sojka's new algorithm for corner detection in digital images

**[Soj02a]**

E.Sojka presented his new algorithm [Soj02a], [Soj02b], [Soj03] for corner detection recently. Like the other known algorithms, the new algorithm also determines the corner response function (CRF) that combines the angle and the contrast of the corner. The algorithm that they propose is based on measuring the variance of the directions of the gradient of brightness. The probability of the event that the image points belongs to the approximation of the straight segments of the isoline of brightness containing the corner candidate being examined is determined using the technique of Bayesian estimations and then used for computing the angle between the segments intersecting at the candidate.

The main idea of the new algorithm is as following. The values of $g(X)$ and $\varphi(Y)$ are the magnitude and the direction of the gradient of brightness. First of all, $g(X)$ and $\varphi(Y)$ are computed for all image points (pixels). The derivatives $\frac{\partial b(X)}{\partial x}$ and $\frac{\partial b(Y)}{\partial y}$ are replaced by the differences that are computed using the following mask

$$\begin{bmatrix} -0.1 & 0 & 0.1 \\ -0.3 & 0 & 0.3 \\ -0.1 & 0 & 0.1 \end{bmatrix} \quad and \quad \begin{bmatrix} 0.1 & 0.3 & 0.1 \\ 0 & 0 & 0 \\ -0.1 & -0.3 & -0.1 \end{bmatrix}.$$

After getting $g(X)$ and $\varphi(Y)$, the image points at which the magnitude of the gradient of brightness is greater than a predefined threshold are considered to be candidates for corners. The candidates are then examined by determining the values $\mu_\varphi(Q)$, $\sigma_\varphi(Q)$, $Corr(Q)$ and $Appar(Q)$. They can be calculated in such way:

$$\mu_\varphi = \frac{\sum_{X_i \in \Omega} w(X_i)\varphi(X_i)}{\sum_{X_i \in \Omega} w(X_i)\varphi}, \qquad \sigma_\varphi^2 = \frac{\sum_{X_i \in \Omega} w(X_i)[\varphi(X_i) - \mu_\varphi]^2}{\sum_{X_i \in \Omega} w(X_i)}. \tag{2.12}$$

where $w(X)$ is a weight expression for determining the angle of the break at a point $Q$ and it comes from a positive weight function $w_r(r(X))$ ( in this function $r(X)$ stands for the distance. The weight can be calculate as following:

$$w(X) = P_{SG}(X)w_r(r(X)) = A_0 \min_{Y \in QX}\{p_d(\beta^{-1}(\Delta b(Y)))p_d(h(Y))P(AngQ|\Delta\varphi(Y))\}w_r(r(X)). \tag{2.13}$$

Here it should be mentioned that the $p_{SG}(X)$ indicated the probability of the event that X belongs to the approximation of a straight isoline segment containing $Q$. And the authors set $p_{SG}(X)$ like this:

$$P_{SG}(X) = \min_{Y \in QX}\{P(BrQ|\Delta b(Y))P(DirQ|h(Y))P(AngQ|\Delta\varphi(Y))\}. \tag{2.14}$$

The term $P(BrQ|\Delta b(Y))$ expresses the probability of the event that $Y$ is a point of the approximation of the isoline whose brightness is $b(Q)$ ($Y$ is a point of the line segment $\overline{QX}$). The term $P(DirQ|h(Y))$ expresses the probability of the event that $Y$ is a point of the approximation of an isoline segment (not the same brightness as $Q$) that aims at $Q$. Both events are independent. Since the neighborhood $\Omega(Q)$ (the neighborhood $\Omega(Q)$ is square-shaped with Q at its center) may contain more than one corner, the segments of the isoline whose brightness is $b(Q)$ need not generally aim at point $Q$ (see Fig. 2.12).

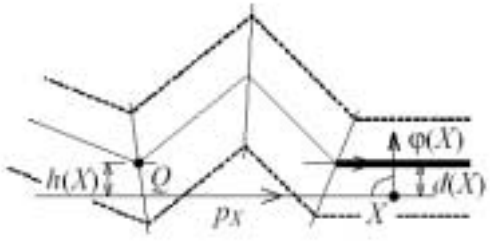



Figure 2.12 The isoline of brightness passing through $Q$ (*continuous line*), an isoline segment aiming at $Q$ (*bold line*), a point $X$ of the approximation of the segment, the deviation $d(X)$ of $X$ from the segment, the deviation $h(X)$ of $Q$ from $p_x$.

Conversely, an isoline segment aiming at $Q$ may generally have an arbitrary brightness. The term $P(AngQ|\Delta\varphi(Y))$ expresses the probability of the event that $Y$ belongs to the area of possible corner at $Q$. Fig. 2.13 shows that there may exist points that do not belong to the corner area in spite of the fact that the $\Delta b(X)=0$, $h(Y)=0$ are both satisfied.

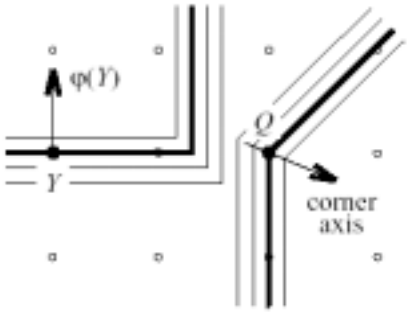



Figure 2.13 Although the conditions $\Delta b(Y)=0$, $h(Y)=0$ are both satisfied at *Y*, *Y* does not belong to the area of the corner at *Q* (both the bold isolines have the same brightness). This fact is detected by the condition $0 \le \Delta\varphi(Y) \le \pi/2$, which is not satisfied for *Y*. Small circles show the position of samples.

Then we can get $Corr(Q)$ and $Appar(Q)$ easily from $\mu_\varphi(Q)$ and $\sigma_\varphi(Q)$ in the following functions:

$$Corr(Q)=g(Q)\sigma_\varphi^2(Q), \tag{2.15}$$

$$Appar(Q)=\sum_{X_i\in\Omega} P_{SG}(X_i)g(X_i)\left|\varphi(X_i)-\mu_\varphi\right|. \tag{2.16}$$

The candidate at which the value $Corr(Q)$ exhibits its local maximum and at which the values of $\sigma_\varphi(Q)$, $Appar(Q)$ are greater than chosen thresholds is a corner. The authors mentioned here the declared size (usually from $7\times7$ up to $11\times11$) of the neighborhood $\Omega(Q)$ is not too critical since the effective size is always determined adaptively by the value of $p_{SG}(X)$.

The author introduce the theoretical foundations of the corner detector in details very clearly in their paper, but here we will not use many words to explain it. Now let's see the computational complexity of this algorithm. If the size of neighborhood $\Omega(Q)$ is $M\times M$ pixels, it is easy to see that the value of $p_{SG}(X)$ for all $X\in\Omega(Q)$ can be computed in $\theta(M^2)$. The same time is needed to compute $\mu_\varphi(Q)$, $\sigma_\varphi(Q)$ and $Appar(Q)$. So one corner candidate is thus completely processed in $\theta(M^2)$. The authors mentioned some methods to speed up the program:
If the value of $p_{SG}(X)$ is very small, then *X* need not be considered in the sums on the right-hand sides of Equations of computing $\mu_\varphi(Q)$, $\sigma_\varphi(Q)$, and $Appar(Q)$; the influence of such a point may be neglected.

If the technique of neglecting is used, the small values of $p_{SG}(X)$ need not even be evaluated. Applying this rule recursively, we see that we can neglect a lot of points within $\Omega(Q)$ then. Therefore, the values of $p_{SG}(X)$ are usually computed only for small parts of $\Omega(Q)$.

#### 2.1.5.1 Comments

In this paper, a new and efficient algorithm is presented. The probability of the event that a point belongs to the approximation of the straight segment of the isoline of brightness containing the corner candidate being tested is determined using the technique of Bayesian estimations, and then used for computing the angle between the segments. The results of the tests show that, in the sense of the successfulness of detection, the new algorithm is better than the other known algorithms that are usually used for solving the problem. In the test they found their detector was better (the successfulness of detection) than other recognized direct corner detectors such as Beaudet, Harris-Stephens [Har88] and Kitchen-Rosenfeld's [Kit82] approachs and also better than the Susan algorithm. Further more the speed of their detector is a little bit slower than the formal direct corner detectors but significantly much faster than the M.A.Ruzon and C.Tomasi's Compass operator [Ruz01] and J.Q.Song, M.R.Lyu and M.Cai's proposed operator [Son03]. According to this strong advantage, it can be used in a real-time application. So their new approach is very fit for our UI-Wand project. But the new algorithm has several parameters need to be changed according to the different conditions of the processing images and the application command. Some of these parameters also have very significant influence on the accuracy and the speed of the detecting procedure. Although the authors gave some suggestions in their program about how to choose these parameters, but we find from our own experience the best combination of the parameters is still a big challenge for us.

## 2.2 Color distribution approaches

### 2.2.1 SUSAN edge and corner detector

**[Smi97]**

S.M Smith and J.M. Brady presented their new approach [Smi97] in 1997. The Susan (Smallest Univalue Segment Assimilating Nucleus) method is an approach to low-level image processing, especially, one-dimensional feature detection as edge detection, two-dimensional feature detection as corner/junction detection and also structure preserving noise reduction. This approach is substantially different from those that were published before..

The concept of each image point is associated with its local area of similar brightness is the basis for the SUSAN principle. Fig.2.14 showing a dark rectangle on white background, and five circular masks having a center pixel called "nucleus" are shown at five image positions. If brightness of each pixel within a mask is compared with the brightness of that mask's nucleus, then an area of the mask can be defined which has the same (or similar) brightness as the nucleus. This area of the mask can be defined as the "USAN"(Univalue Segment Assimilating Nucleus).

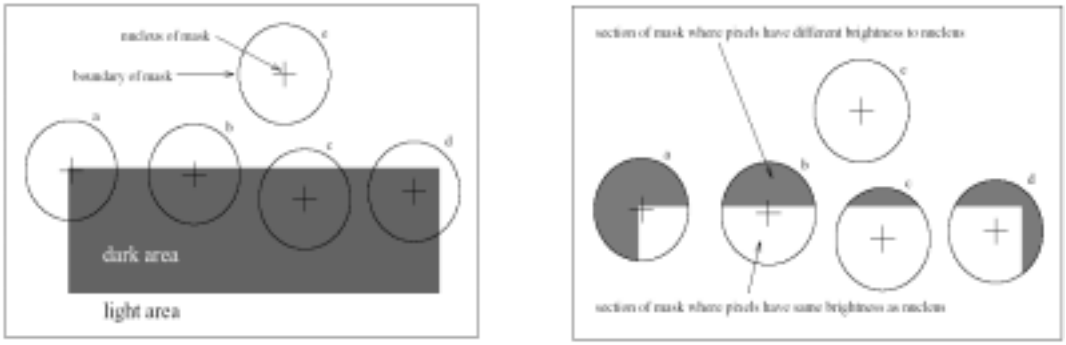



Figure 2.14 Five circular masks at different places on a simple image. The right image draws each mask from the left image is depicted with its USAN (in white).

The area of USAN conveys the important information about the structure of the image in the region around any point in question. As that can be seen from Fig.2.14, the USAN area is at a maximum when the nucleus lies in a flat region of the image surface, it falls to half of this maximum very near a straight edge, and falls even further when inside a corner. It is this property of the USAN 's area, which is used as the main determinant of the presence of edges and two-dimensional features. From the size of the USAN and gravity center of the mask related to nucleus position and other information, features such as edges and corners can be detected.

#### 2.2.1.1 The SUSAN edge detector

The algorithm performs the following steps at each image pixel:

1. **Place a circular mask around the pixel in question (the nucleus).**
   The mask (usually $3\times3$) is placed at each point in the image and for each point; the brightness of each pixel within the mask is compared with that of the nucleus (the center point of the mask):

$$c(\vec{r},\vec{r}_0)=\begin{cases}1 & if\ |I(\vec{r})-I(\vec{r}_0)|\le t\\ 0 & if\ |I(\vec{r})-I(\vec{r}_0)|>t\end{cases}. \tag{2.17}$$

where $\vec{r}_0$ is the position of the nucleus in the two dimensional image, $\vec{r}$ is the position of any other point within the mask, $I(\vec{r})$ is the brightness of any pixel, $t$ is the brightness difference threshold and $c$ is the output of the comparison. The parameter $t$ determines the minimum contrast of features that will be detected and also the maximum amount of noise, which will be ignored. Using Eq. (2.17) can give very good result, but a much more stable and sensible Eq. (2.18) is used in place of Eq.(2.17) is:

$$c(\vec{r},\vec{r}_0)=e^{-\left(\frac{I(\vec{r})-I(\vec{r}_0)}{t}\right)^6}. \tag{2.18}$$

The form of Eq.(2.18) was chosen to give a "smoother" version of Eq.(1.17). This allows a pixel's brightness to vary slightly without having too large an effect on $c$; even it is near the threshold position. The power chosen as 6 is a theoretical optimum.

2. **Using Eq.(2.19) calculate the number of pixels within the circular mask which have similar brightness to the nucleus. (These pixels define the USAN).**
Originally a simple Eq. (2.19) determined this comparison. Then this comparison is done for each pixel within the mask, and a running total $n$ out of the outputs $c$ is made.

$$n(\vec{r}_0)=\sum_{\vec{r}} c(\vec{r},\vec{r}_0). \tag{2.19}$$

If we just use simple Eq. (2.17), then this total $n$ is just the number of pixels in USAN. Next, $n$ is compared with a fixed threshold $g$, which is called "geometric threshold" and it is always set to $3n_{\max}/4$, where $n_{\max}$ is maximum value which $n$ can take.

3. **Using Eq. (2.20) subtract the USAN size from the geometric threshold to produce an edge strength image.**
Using the following rule then creates the initial edge response:

$$R(\vec{r}_0)=\begin{cases}g-n(\vec{r}_0)<g & if\ n(\vec{r}_0)<g\\ 0 & otherwise\end{cases}. \tag{2.20}$$

where $R(\vec{r}_0)$ is the initial edge response. This is a simple formulation of SUSAN principle, and it means the smaller the SUSAN area, the larger the edge response.

4. **Use moment calculations applied to the USAN to find the edge direction.**
Computation of the edge direction is necessary for feature detection especially for the screen edge detection that maybe we will use in our project. Because with the

direction information we can estimate the screen edge by it's geometric profile. The direction of an edge associated with an image point which has a non zero edge strength is found by analyzing the USAN in one of the two ways, depending on the type of edge point which is being examined. If USAN shapes that would be expected for an ideal step edge, (*inter*-pixel edge case), the vector between the center of gravity $r$ of the USAN and the nucleus of the mask is perpendicular to the local edge direction. The central gravity can be calculate in this way:

$$\vec{\bar{r}}(\vec{r}_0) = \frac{\sum_{\vec{r}} \vec{r} c(\vec{r}, \vec{r}_0)}{\sum_{\vec{r}} c(\vec{r}, \vec{r}_0)}. \tag{2.21}$$

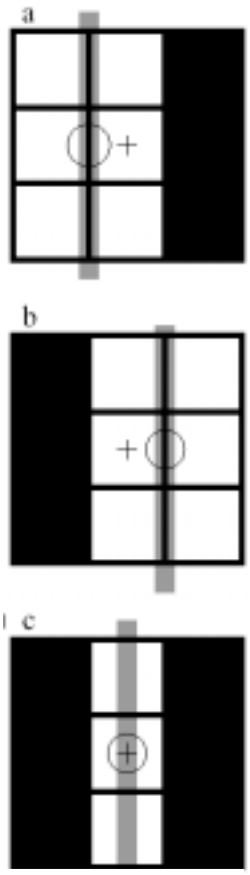



Figure 2.15 Two main edge types

If the point lies on a thin band, which has brightness roughly half way between the brightness of the two regions, generate the edge (*intra*-pixel edge case). The USAN formed is a thin line in the direction if the edge. The edge direction is thus calculated by finding the longest axis of symmetry. This estimated by taking the sums:

$$\overline{(x-x_0)^2}(\vec{r}_0) = \sum_{\vec{r}} (x-x_0)^2 c(\vec{r}, \vec{r}_0), \tag{2.22}$$

$$\overline{(y-y_0)^2}(\vec{r}_0) = \sum_{\vec{r}} (y-y_0)^2 c(\vec{r}, \vec{r}_0), \tag{2.23}$$

$$\overline{(x-x_0)(y-y_0)}(\vec{r}_0) = \sum_{\vec{r}} (x-x_0)(y-y_0) c(\vec{r}, \vec{r}_0), \tag{2.24}$$

The ratio of $\overline{(y-y_0)^2}$ to $\overline{(x-x_0)^2}$ is used to determine the orientation of the edge; the sign of $\overline{(x-x_0)}\overline{(y-y_0)}$ is used to determine whether a diagonal edge has positive or negative gradient. Thus in case (c) the edge direction is again found in a simple manner. The remaining question is how to automatically determine which case fits any image point. . Fig. 2.15 shows clearly the logic behind this. Firstly, if the USAN area (in pixels) is smaller than the mask diameter (in pixels) then the intra pixel edge case is assumed. If the USAN area is larger than this threshold, then the center of gravity USAN is found, and used to calculate the edge direction according to the inter-pixel edge case. If however, the center of gravity is found to lay less than one pixel away from the nucleus then it is likely that the intra-pixel edge case more accurately described this situation.

5. **Apply non-maximum suppression, thin and sub-pixel estimation, if required.** Finally, therefore, the edge response image is suppressed so that non-maxim (in the direction perpendicular to the edge (are prevented from being reported as edge points. Following this, the "strength thinned" image can be "binary thinned". A set of rules for binary thinning has been implemented (still making use of the strengths in the non-suppressed edge response image) which work well to give a good final binary edge image.

#### 2.2.1.2 The SUSAN corner detector

The SUSAN "corner" detector is very similar to the edge detector, particularly in the early stages. First, all pixels within a circular mask are compared with the nucleus, using exactly the same comparison equation. The sum of these comparisons is also calculated in the same way, using Eq. (2.19). Next, $n$ is again compared with the geometric threshold $g$, here the "corner" detector is quite different from the edge detector, where $g$ only necessary in the presence of noise. For a " corner" to be present, $n$ must be less than half of its maximum possible value, $n_{\max}$. For the reason that if the nucleus lies on a corner then the USAN area will be less than half of the mask area, and will be local minimum. It is safe to set $g$ to be exactly half of $n_{\max}$ because of the nature of a quantized straight edge; an USAN formed from a straight edge will always be larger than half of the mask size, as it includes the nucleus.

In summary then, the algorithm performs the following steps at each image pixel:
1. Place a circular mask around the pixel in question (the nucleus).
2. Using Eq.(2.19) calculate the number of pixels within the circular mask which have similar brightness to the nucleus. (These pixels define the USAN).
3. Using Eq.(2.20) subtract the USAN size from the geometric threshold (which is set lower then when finding edges) to produce a corner strength image.
4. Test of false positives by finding the USAN 's centroid and its contiguity.
5. Use non-maximum suppression to find corners.

#### 2.2.1.3 Comments

This approach to feature detection has many differences to the well-known methods; the most obvious is that no image derivatives are used and no noise reduction is needed. So the principle of SUSAN method is: An image processed to give as output inverted USAN area has edges and two-dimensional features strongly enhanced, with the two-dimensional features more strongly enhanced than edges. The fact that SUSAN edge and corner enhancement uses no image derivatives explains why the performance in the presence of noise is good. The integrating effect of the principle, together with its non-linear response, also gives strong noise rejection. This can be understood very simply if an input signal with identically independently distributed Gaussian noise is considered. As long as the noise is small enough for the SUAN function to contain each similar value, the noise is ignored. The integration of individual values in the calculation of areas further reduces the effect of noise. The other strength of the SUSAN edge detector is that the use of controlling parameters is much simpler and less arbitrary and therefore easier to automate than with most other edge detection algorithms.

### 2.2.2 Edge, junction, and corner detection using Compass operator

**[Ruz01]**

This algorithm was presented by M.A. Ruzon, and C. Tomasi in [Ruz99a] [Ruz99b] and [Ruz01]. Their detector is well known as the Compass operator. The Compass Operator detects step edges without assuming that the regions on either side have constant color. Using distributions of pixel colors rather than the mean; the operator finds the orientation of a diameter that maximizes the difference between two halves

of a circular window. Junctions can also be detected by exploiting their lack of bilateral symmetry. This approach is superior to a multidimensional gradient method in situations that often result in false negatives, and it localizes edges better as scale increases.

The algorithm requires computing the perceptual distance between the representations of adjacent image neighborhoods. First of all they consider the problems of representing a neighborhood as a color distribution, computing the similarity between individual colors, and computing the dissimilarity between two color distributions. They create color signatures in such a way: a color signature is a data structure consisting of a set of ordered pairs $\{(x_1, v_1), (x_2, v_2), \ldots, (x_n, v_n)\}$, where the $v_i$ s are vectors in a color space to which the weights $x_i$ are assigned. A signature is equivalent to a probability mass function when the $v_i$ s sum to one. Signatures are superior to histograms because they adapt to the data; they do not force an arbitrary partitioning of color space. Before computing the perceptual distance between two color signatures, they compute the ground distance between each pair of colors in the window. They chose the ground distance as $d_{ij} = 1 - \exp\{-E_{ij} / \gamma\}$. The ground distance between color $i$ and color $j$ is an exponential measure, with steepness governed by $\gamma$ (they theme $\gamma = 14.0$), of the Euclidean distance $E_{ij}$ between them in CIE-Lab. Then measuring distance between two color signatures is a sub-problem of measuring distance between probability density functions. The Earth Mover's Distance (EMD) formulates the distance measurement as an instance of the transportation problem.

#### 2.2.2.1 Compass Edge detection

Here, they present a method of estimating the strength and orientation of an edge hypothesized to split a window. They divide the window in half with a line segment, compute a color signature for each half, and find the EMD between them. Repeating this process using line segments with different orientations reveals the true strength and orientation of the edge. The radius of the circle is $3\sigma$, where $\sigma$ is a scale parameter. They perform vector quantization on all pixels whose intersection with this circle is nonzero. To form two color signatures, they map each pixel in a semicircle to its nearest color cluster and add its weight to the corresponding point mass. The weight given to each pixel in this method is the product of three factors: the area of a pixel (which they model as a unit square) that falls inside a semicircle, its relative importance to the computation, and a normalization factor so that the total mass of each signature is one. They model a pixel's importance as a Rayleigh distribution on the distance from a pixel's center to the circle's center. Rayleigh distributions are expressed in polar coordinates as $f(r, \theta) = r \exp\{-r^2 / 2\sigma^2\}$. They depart significantly from their formulations in $2D$ by revolving the $1D$ curve into an isotropic function. The result of computing the EMD over a set of orientations at each window is a function $h(\theta), 0 \le \theta \le 180$. Fig. 2.16 shows two applications of the algorithm on two windows, the first straddling an ideal step edge and the second taken from an image containing a bush and a tree. The strength of each edge is the maximum value of $h(\theta)$, which never exceeds one, and the orientation of the edge is the argument that produced the strength.

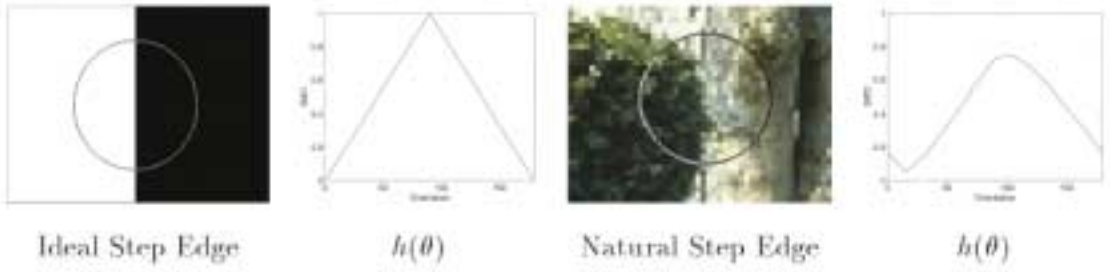



Figure 2.16 Applying the EMD to a circular neighborhood over a range of diameter orientations produces a function from which edge information is extracted.

### 2.2.2.2 Compass Junction detection

A junction is a point where three or more image regions meet, but the EMD, like other distance measures, cannot handle three-color signatures at a time. It is certainly possible to split a circle into three or more neighborhoods and find the EMD between each pair, but this is an order of magnitude more expensive, and the added problem of determining the number of image regions meeting at a point is cumbersome. Instead, they opt for a less direct but simpler approach. The minimum value of $h(\theta)$ is called the abnormality and it serves a much more interesting purpose. When abnormality is high, it indicates a complete lack of symmetry in the image data that usually corresponds to a junction. The synthetic image in Fig. 2.17 illustrates this concept.

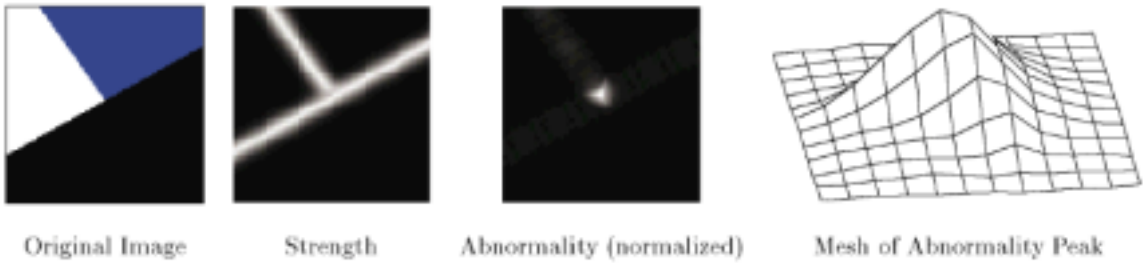



Figure 2.17 Whereas the strength at each point represents the maximum EMD value over all orientations, the abnormality represents the minimum, which is high near junctions. The shape of the mesh resembles a triangular pyramid.

Points near the edges have high strength, but only those near the junction have high abnormality as well. The shape of the mesh provides a clue as to how to interpret the image content in this neighborhood. They propose the following algorithm for reconstructing junctions containing three regions: For each local maximum above a threshold, examine a small neighborhood (with size proportional to $\sigma$) centered at the maximum. Fit a triangular pyramid to the abnormality values and the sides of the base will be almost normal to the edges that form the junction. For T-junctions, the error in this approximation must be more than for Y-junctions.

### 2.2.2.3 Compass Corner detection

They cannot detect corners using abnormality because the EMD of the corner point at the orientation the bisectors the corner is usually zero. Finding corners, therefore, requires a separate approach.

In this paper they presents a method based on their edge detector that works directly on color images. Under the proposed framework of color distributions, the only difference between detecting edges and detecting corners is that they no longer have half the wedges of the circle in each signature. An extra parameter $\beta$ is introduced to measure the angle subtended by the corner. The range of $\theta$ must now be $[0,360)$ since the two radii that split the circle no longer have odd symmetry. They refer to the two sides as clockwise and counterclockwise, and $\theta$ refer to the orientation of the clockwise side. Fig. 2.18 illustrates the parameters. Otherwise, the algorithm for computing strength and orientation at each point is similar to that for edge detection. A circular neighborhood is quantized and two color signatures are formed. The EMD measures the perceptual distance between signatures.

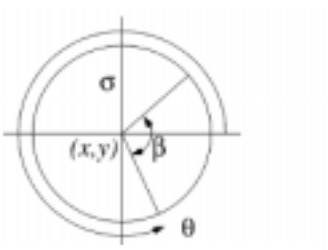


Figure 2.18 Parameters

1. **Computing Distance between Signatures of Unequal Mass**
   When $\beta$ is less than $180^\circ$, the two color signatures have unequal amounts of mass. There are two possible methods for accounting for this difference. The first method, normalized matching, is to normalize the larger signature so that its mass equals that of the smaller. The second method, partial matching, matches the smaller signature to the subset of a larger signature that minimizes the total work. Regardless of the method, the mass of the smaller signature is set to one so that the EMD continues to lie in the range $[0,1]$. Both types of matching can fail under certain circumstances. Partial matching method will miss the corners and normalized matching, however, often performs worse than partial matching on typical corners. The testing result is that the size of a corner is often underestimated using normalized matching. They finally chose partial matching for their experiments. They may miss a few small corners, but experience shows that such corners rarely appear in natural images. For the corners that are found, it is better to describe them accurately.

2. **Finding Corners**
   The result of applying the corner detector to an image is a four-dimensional $(x, y, \theta, \beta)$ array of EMD values, and corners are relative maxima above a minimum strength in this array. However, there are some complicating factors: first, a corner is a response to a phenomenon that takes place over a relatively large portion of the image, so checking only the nearest neighbors in the array will produce too many corners. It is possible to get many maxima all responding to the same corner phenomenon but with noticeably different parameters; second, they cannot directly compare EMD responses from parameter values that differ in $\beta$. Changing the size of the corner also changes the statistics of the EMDs that are generated. Therefore, they include edge information to winnow the set of maxima over $x$, $y$, and $\theta$ for every value of $\beta$ to the actual corners. If multiple responses to a corner still remain, they select one according to a heuristic. The steps are explained in details below.

   - Testing Corner Candidates

For each corner, they compute $\theta_C$ and $\theta_{CC}$, the difference between the orientation of the clockwise and counterclockwise sides of the corner, respectively, and the edge orientation found at the endpoint of each side. The fit to the model is expressed as $P = \cos\theta_C + \cos\theta_{CC}$, where P lies between 0 and 2. Then they set the threshold P at 1.97. Also, where the edge crosses the corner's axis of symmetry, the projection of the edge response vector onto the line normal to this axis must be weaker than the corner response. They check responses on a small interval along this axis centered at a point that is $3\sigma(\sec\frac{\beta}{2} - \tan\frac{\beta}{2})$ pixels away from the corner. This quantity is the distance from the corner point to the circumference of an imaginary circle tangent to the sides of the corner at its endpoints. Fig. 2.19-middle) shows the initial corner candidates found by setting threshold of the relative maxima. Most of these responses are false positives, and this procedure greatly reduces their number.

Figure 2.19 Detecting corners from an initial set of candidates, applying an edge model followed by pruning multiple responses if they exist.

- Pruning Multiple Responses
  Fig. 2.19-middle shows that after the above testing the remained corner candidates are always not just one. So they offer a heuristic to deal with this condition. First, they must decide when two corner candidates are responding to the same actual corner. They define two corners as being close enough if the corner points are within $9\sigma/4$ pixels of each other and one of two conditions is true: 1) either the two clockwise or the two counterclockwise orientations differ by no more than $10^\circ$, or 2) the sum of these differences is no more than $40^\circ$. These conditions group nested corners while preserving multiple corners near junctions. An ambiguity arises when corner $X$ is close to corners $Y$ and $Z$, but $Y$ and $Z$ are not close to each other. However, the application of the edge model removes enough candidates to prevent this. Once they have computed the transitive closure, they select the member of each set that maximizes the expression $2C + P + E$, where $C$ is the corner strength, $P$ is the degree of orientation match described earlier, and $E$ is the sum of the edge strengths at the endpoints of the two sides of the corner. $C$ is doubled so that each term contributes equally.

#### 2.2.2.4 Comments

This research models a neighborhood as a distribution of colors. Their method shows that the increase in accuracy of this representation translates into higher-quality results for low-level vision tasks on difficult, natural images, especially as neighborhood size increases. They emphasize large neighborhoods because small ones often do not contain enough information. They emphasize color because it subsumes gray scale as an image range and because it is the dominant form of human

perception. They discuss distributions in the context of detecting edges, corners, and junctions, and they show results for each. They believe the resulting algorithm to be the first corner detector that can handle either color or most arbitrary textures.

The improvement in the quality of edges detected, the potential for reconstructing junctions, and the ability to find corners in textured, color images at large scales all give testimonial to the added power of distributions in feature detection. Being able to find all three features from one array of values is encouraging. They believe this work to be most applicable to figure-ground separation, a task other operators can carry out only in simple situations. Finding corners with the help of edges and finding occluding edges through the reconstruction of junctions brings them closer to this ultimate goal than other methods. Among the secondary principles established by this work, three stand out. The first is that it is best to treat color as a vector instead of three components; only then can they be reasonably certain that an algorithm's representation of color is similar to that of a person. Second, using a saturating distance measure is more in line with the notion that all pairs of dissimilar colors are equally different. Finally, the Earth Mover's Distance, previously shown to be applicable to the color distributions of entire images, has been found to be useful for local neighborhoods. It is what allows them to combine distributions and the saturating distance measure. The limitations of the algorithm are more practical and deal with the high computation cost. Quantizing each image window and computing the EMD many times is much slower than the corresponding operations in other detectors, though these operations can be done largely in parallel. Experiments designed to reduce the running time had the disadvantage of producing less accurate results.

### 2.2.3 A color-distribution-based detector

**[Son03]**

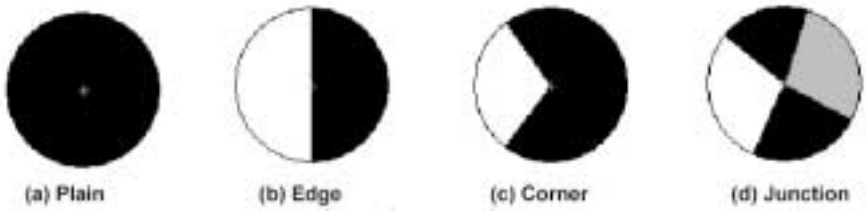
J.Song, M.R.Lyu, and M.Cai presented their approach [Son03] in 2003. This is an algorithm to detect edges, corners and junctions in one pass based on both spatial and statistic color distributions in a neighborhood.

#### 2.2.3.1 Generic neighborhood model

First of all the "'generic neighborhood model"' is defined as a circular neighborhood of any pixel $P$ consists of all pixels whose Euclidean distances of image coordinate to $P$ is less than the circular mask. The neighborhood of $P$ (labeled $NB(P)$) can be represented by a set of sectors. Each sector is of the most homogeneous color distribution inside, and every two adjacent sectors are of the most significant difference in color distribution. After getting the best-divided sectors, the type of $P$ can be classified according to the number and relationship of the sectors. Let $\theta i$ be the angle subtended by *Sectori* ($0 < \theta_i < 2\pi$). Table 2.2 shows the classification criteria.

Table 2.2 Classification criteria of the pixel type based on the best-divided sectors

| Sector number | Pixel type | Example | Relation |
|---|---|---|---|
| n=1 | Plain | Below (a) | Any |
| n=2 | Edge | Below (b) | $\|\theta1-\theta2\|<=\varepsilon$ |
| | Corner | Below (c) | $\|\theta1-\theta2\|>\varepsilon$ |
| n>2 | Junction | Below (d) | Any |




#### 2.2.3.2 A generic edge, corner and junction detection algorithm

The splitting-and-merging sectors scheme is applied in their algorithm. The circular neighborhood is split along radial directions, by which they can obtain the spatial color distributions. The complete detection algorithm consists of eight steps:

1. **Splitting**
   The circular neighborhood is split into $n$ even (usually 8 to 24) unit sectors (hereinafter, called slice) with the same angle. Fig. 2.20 shows a synthetic example of circular neighborhood, which is split into 12 slices.


Figure 2.20 A split neighborhood

2. **Color distribution feature**
   Merging slices into sectors is based on comparing the color distribution features of slices. $CDF_i$ denotes the color distribution feature of $Slice_i$. The function $Dist(CDF_i, CDF_j)$ is to calculate the distance between two $CDF_s$, which returns a real number between [0,1]. The expected response of $Dist(CDF_i, CDF_j)$ approaches 0 when $CDF_i$ and $CDF_j$ are very similar perceptually, and approaches 1 when they are significantly different.

3. **Integrated slice distance**
   Since the merging operation only takes place between two adjacent slices in a circular neighborhood, the distance between every pair of adjacent slices, Adjacent Slice Distance (*ASD*), is first calculated as:

$$ASD_i = Dist\left(DCF_i, \quad CDF_{Next(i)}\right), \quad i = 1..n. \tag{2.25}$$

   Let $T_d$ denote the threshold of *CDF* distance. Two slices whose *CDF* distance is larger than $T_d$ are classified into different regions. Therefore, the peaks (higher than $T_d$) in the *ASD* profile indicate the potential boundaries between different

regions. However, from the Fig.2.21-a) the weakness of *ASD* comes from its "local" nature, the Global Slice Distance (*GSD*) is then calculated. *GSD* is actually a normalized distance whose base is the slice with the smallest *ASD*. *GSD* is defined as:

$$GSD_i = Dist\left(CDF_i, CDF_{\min\_inx}\right), \quad i = 1..n. \tag{2.26}$$

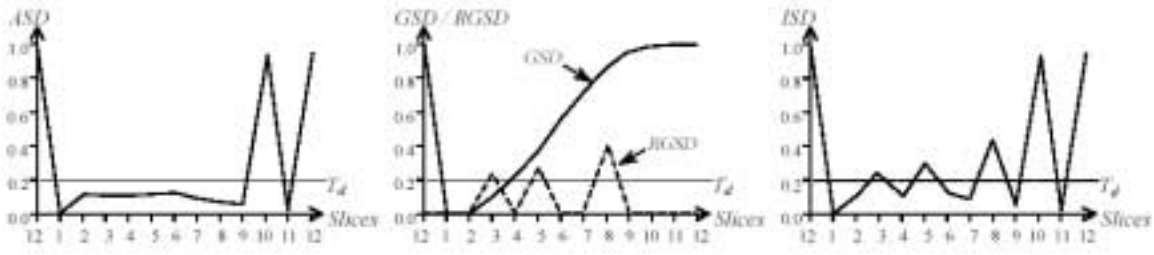



Figure 2.21 Slice distance analysis: a) *ASD* profile: b) *GSD/RGSD* profile; c) *ISD* profile.

But, the *GSD* profile cannot be used directly because they need the relative distance of adjacent regions. So, they translate the *GSD* profile into the *RGSD* (Relative *GSD*) profile. Fig. 2.21-b) shows the *RGSD* profile (dashed line). However, the weakness of *GSD/RGSD* is that it cannot distinguish the difference between two slices whose distances to $Slice_{\min\_inx}$ are similar but whose $CDF_s$ are very different. Fortunately, *ASD* can distinguish them. So *ASD* and *RGSD* are complementary. They are integrated into the Integrated Slice Distance (*ISD*) for merging slices correctly.

$$ISD_i = MAX\left(ASD_i, RGSD_i\right) \quad i = 1..n. \tag{2.27}$$

In the *ISD* profile (Fig. 2.21-c)), the slices that are the boundaries of different regions are enhanced, and the internal slices are suppressed. Therefore, obtaining the best-divided sectors.

**4. Boundary slice detection**

To merge slices into sectors, they detect the boundary slices to separate sectors in the *ISD* profile. They detect the boundary slices by the following four steps:

- Step 1: Detect all peaks in the *ISD* profile, and sort them in a list in the descending order.
- Step 2: If the list is empty, terminate the detection; otherwise, pop the first peak as the current peak and go to Step 3. The current slice is corresponding to the current peak.
- Step 3: If neither the preceding slice nor the succeeding slice of the current slice is a boundary slice, the current slice is identified as a boundary slice. If both of them are already boundary slices, judge whether or not to add a new boundary slice.
- Step 4: Go to Step 2.

After the detection terminates, they obtain a group of boundary slices. Normally, the number of boundary slices should be 0 (for "Plain"), 2 (for "Edge" or "Corner") or more than 2 (for "Junction").

**5. Spurious junction elimination**

When the number of sectors is larger than 2, there is a potential junction, some of them are better considered as edges or corners, especially for weak junctions. Thus, for a weak junction, they compare the strength of being a junction and that of being an edge (or corner) to decide which is the better result. If its junction strength is lower than the edge (or corner) strength, it should be converted into an edge (or corner). They first calculate the junction strength as follows:

$$JunctionSt\ rength = \frac{1}{m} \cdot \sum_{i=1}^{m} Dist\left(SectorCDF_i, SectorCDF_{NextSector(i)}\right) \tag{2.28}$$

Note $SectorCDF_i$ is the color distribution feature of $Sector_i$. It is unnecessary to check strong junctions, the junctions whose strengths are higher than $T_{js}$ are accepted directly without the following check. Then, they compute the strength of being an edge (or a corner) for this junction. To divide the sectors into two groups of consecutive sectors that are most similar within each group and most different between two groups $g_1$ and $g_2$. They define $S(g)$ to calculate the similarity of a group of sectors, and define $D(g_1, g_2)$ to calculate the difference between two groups of sectors. Therefore, the maximum strength of being an edge (or a corner) is:

$$MAXStrength = MAX\{D_k(g_1, g_2), \quad if\ MIN[S_k(g_1), \quad S_k(g_2)] > Th\}, \quad k = 1,2,\ldots, C_m^2. \tag{2.29}$$

Note *Th* is the similarity threshold, which should be larger than (1-$T_d$). If *MaxStrength* is larger than *JunctionStrength*, the junction is converted to an edge (or a corner) by merging sectors according to the division and by updating the boundary slices.

**6. Boundary refinement**

As described above, the boundaries of sectors are the borders of boundary slices, which are not necessarily the real boundaries. It is safer to state that the real boundary is between the bisectors. They defined the Distance Ratio (*DR*) and $x(DR)$ to calculate the exact boundary in the bisectors.

**7. Classification**

With the best-divided sectors and their accurate boundaries, edges, corners and junctions can be classified clearly according to the criteria listed in Table 1. If the number of sectors is two, the smaller angle of two boundaries, denoted by $\alpha$ ($\alpha <= \pi$), is calculated to discriminate between edge and corner. If there are more than two sectors, the pixel type is "Junction". The strength of all junctions has already been calculated using Eq. (2.28), but not that of edges and corners. With the accurate boundary information, they compute the edge strength and the corner strength as follows:

$$EdgeStrength = D'(g_1, g_2) \times \frac{\alpha}{\pi - \varepsilon}. \tag{2.30}$$

$$CornerStrength = D'(g_1, g_2). \tag{2.31}$$

$D'(g_1, g_2)$ has a form similar to $D(g_1, g_2)$ (GSD) with a small difference that the two slices containing the real boundaries are ignored. The rightmost part of Eq. (2.30) is to enhance the edge whose $\alpha$ is approaching $\pi$.

**8. Localization**

This is the last step of the edge, corner and junction detection process. The above seven steps are performed orderly on each pixel in an image. After all feature pixels (i.e., not "Plain" pixels) are classified and assigned the strength, this step takes place to locate the final feature position accurately. It consists of three successive operations: (1) use directional non-maximal suppression to locate edges and to eliminate false positives of corners; (2) use regional non-maximal suppression to locate them rather than use the directional suppression, and (3) use integrity verification to match the branches of remaining corners or junctions with edges.

#### 2.2.3.3 Comments

Unlike the previous detectors which only focus on the statistic color distribution, their approach emphasizes both spatial and statistic color distributions. It splits the circular neighborhood into slices to obtain the spatial color distribution, and calculates the statistic color distribution in each slice. By analyzing the integrated slice distance, the slices are merged into sectors, whose number and relationship determine the pixel type. They implemented the approach with improved techniques and test it using both synthetic and real images. The experimental results demonstrate that their new approach has high accuracy on both feature localization and boundary direction. This model emphasize (1) the circular neighborhood to ensure the isotropy, (2) the statistic color distribution to handle both uniform-colored regions and textured regions, and (3) the spatial color distribution to distinguish edge, corner and junction. So this detector not only gives the location of edges, corners and junctions, but also indicates the accurate direction of their boundaries, which are very important in reconstructing the image structure. Based on this model, they propose a low-level feature detection approach which is the first one with all the following four capabilities: (1) detecting edges, corners and junctions in the same pass, which makes the reconstruction of corner/junction very convenient since all the required information, including corner/junction location, their boundary directions and edge direction, is ready; (2) handling both uniform-colored regions and textured regions; (3) able to detect free-shaped corners and junctions; (4) producing the accurate boundary direction for corners and junctions. They also mentioned some speed-up techniques in their paper to resolve the speed problem, which is occurred very seriously in Compass operator.

## 2.3 Corner Detection Approaches Comparison

After deeply studying the main known corner detectors, we can divide them into three paradigms. The first paradigm is *boundary-based corner detectors*, the second paradigm is *direct corner detectors* and the third paradigm is *color distribution based detectors*. The main idea of *boundary-based corner detectors* is to extract the boundary of the object first and analyze the shape afterwards. However, this kind of method is computationally expensive. The *direct corner detectors* work directly with the values of brightness of images without segmenting the image in advance, they usually model the image as a surface and try to fit some kind of template to the image in a neighborhood, where the template can be either a parametric surface or a definition of "points of interest". So the processing speed is usually fast enough for real-time application. So above two paradigms (*boundary-based corner detectors* and *direct corner detectors)* can be included into *gradient approaches*. The *color distribution based detectors* are very different with the *gradient approaches*, it do not model an image as a surface; instead, they concern the statistic color distribution in the circular neighborhood centering at each pixel rather than compute the directional gradients or derivatives. So the *color distribution based detectors* can be included into *color distribution approaches*. Table 2.3 shows which paradigm the different corner detectors belong to.

Table 2.3 Corner detector paradigms

| Detector Name | Paradigm | Paper Index |
|---|---|---|
| Beaudet | Direct corner detectors | [Bea78] |
| Kitchen-Rosenfeld | Boundary-based Corner detectors | [Kit82] |
| Harris-Stephens | Direct corner detectors | [Har88] |
| Deriche-Giraudon | Direct corner detectors | [Der93] |
| SUSAN | Color distribution based detectors | [Smi97] |
| Compass | Color distribution based detectors | [Ruz01] |
| Song-Lyu-Cai | Color distribution based detectors | [Son03] |
| Sojka | Direct corner detectors | [Soj02a] |

Now some compares between these different detectors can be made. In order to compare the accuracy between different corner detectors, we need common criteria. The corner detectors should satisfied the following criteria:

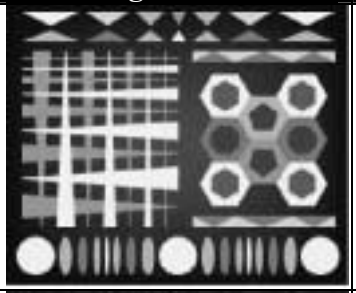
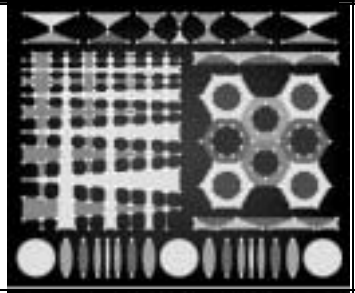
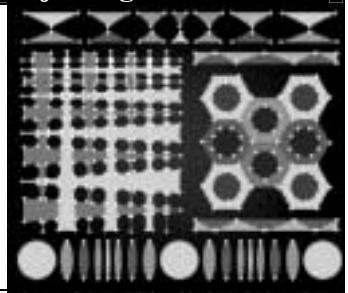
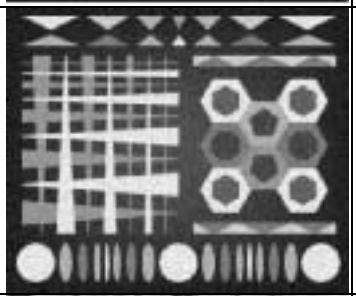
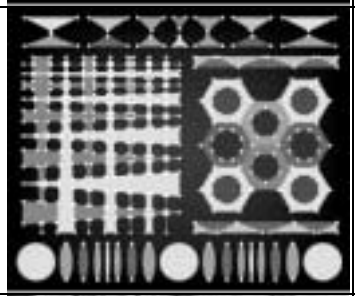
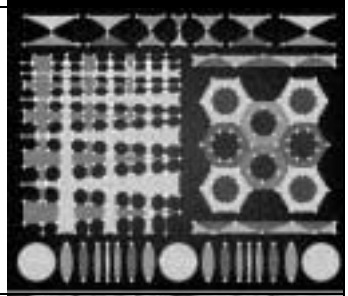
1. All the true corners should be detected.
2. No false corners in the detected corners.
3. The detected corners should be located with a small localization error.
4. The detection should be stable, which means if a corner is detected in one image, the corresponding similar corners should also be detected in another image.
5. The detectors should not be sensitive to the noise in images.
6. A low time complexity of detector is often required by the real time processing.

First of all, the processing speed of these detectors is shown in Table 2.4 (here we choose the same hardware conditions for each detector):

Table 2.4 Processing time comparing

| Detector Name | Feature Detected | Running Time (ms) | Grade |
|---|---|---|---|
| Beaudet | Corner | 5-8 | 5 |
| Harris-Stephens(Plessy) | Corner | 34-39 | 4 |
| Kitchen-Rosenfeld | Corner | 16-20 | 4.5 |
| SUSAN | Edge | 26-48 | 4 |
| ($\sigma = 3 or \sigma = 4$) | Corner | 27-49 | |
| Song-Lyu-Cai ($\sigma = 3 or \sigma = 4$) | Edge, Corner & Junction | 46-78 | 3.5 |
| Compass | Edge & Junction | 682-807 | 1 |
| ($\sigma = 3 or \sigma = 4$) | Corner | 6847-7460 | |
| Sojka | Corner | 80-155 | 3 |

Table 2.5 Test images

| | Test Image | Reference Solution | Sojka algorithm Result |
|---|---|---|---|
| Image #1 (A synthetic image without noise) |  |  |  |
| Image #2 (A synthetic image with a Gaussian noise) |  |  |  |
| Image #3 (An image from a CCD camera) |  |  |  |

Form Table 2.4, it is very clear shown that most of the *direct corner detectors* are much faster than *color distribution based approach*. Especially, the Compass detector, it is so slow to using in any real-time applications. The Song-Lyu-Cai 's approach is much faster than the Compass operator, but it will detect the edge, corner and junction in just one round, which is not so necessary for our further application. In the *direct corner detectors* the processing speed are also not quite the same, for example, Beaudet detector is the most faster one, and the Sojka detector is a little bit slow comparing to others. Because we will focus on the real-time processing

application in our project, we will mainly do the following compares in those fast corner detectors (e.g. Beaudet, Harris-Stephens, Kitchen-Rosenfeld, SUSAN and Sojka).

Except the speed aspect, the accurate level is also among the most critical criteria to evaluate the quality of the detectors. In Table 2.5, the testing images are shown, where test image #1 is a synthetic image without noise, the test image #2 is the same as test image #1 but with a Gaussian noise in it, the test image #3 is a real image obtained from a CCD camera, all the test images are gray level images.

In Table 2.6, Table 2.7 and Table 2.8, using the above three test images, the accurate compares between these detectors are carried out among the Beaudet's, the Harris-Stephens', the Kitchen-Rosenfeld's, the SUSAN's and Sojka's corner detectors.

Table 2.6 The testing result of test image #1

| Detector Name | Total Corners | Correct Detections | Missed Corners | False Detections | Multiple Detections | Total Error | Localization Error | Grade |
|---|---|---|---|---|---|---|---|---|
| Beaudet | 470 | 406 | 64 | 5 | 35 | 104 | 1.49 | 3.0 |
| Deriche-Giraudon | | 337 | 133 | 0 | 7 | 140 | 1.21 | 2.0 |
| Harris-Stephens | | 442 | 28 | 0 | 6 | 34 | 0.72 | 4.0 |
| Kitchen-Rosenfeld | | 420 | 50 | 32 | 67 | 149 | 0.96 | 3.0 |
| SUSAN | | 370 | 100 | 20 | 9 | 129 | 1.23 | 2.5 |
| Sojka | | 470 | 0 | 0 | 0 | 0 | 0.56 | 5.0 |

Table 2.7 The testing result of test image #2

| Detector Name | Total Corners | Correct Detections | Missed Corners | False Detections | Multiple Detections | Total Error | Localization Error | Grade |
|---|---|---|---|---|---|---|---|---|
| Beaudet | 470 | 363 | 107 | 54 | 24 | 185 | 1.62 | 2.5 |
| Deriche-Giraudon | | 308 | 162 | 22 | 5 | 189 | 1.49 | 2.0 |
| Harris-Stephens | | 429 | 41 | 10 | 9 | 60 | 0.77 | 4.0 |
| Kitchen-Rosenfeld | | 360 | 110 | 59 | 30 | 199 | 1.37 | 2.5 |
| SUSAN | | 358 | 112 | 29 | 2 | 143 | 1.58 | 2.0 |
| Sojka | | 466 | 4 | 1 | 1 | 6 | 0.57 | 5.0 |

Table 2.8 The testing result of test image #3

| Detector Name | Total Corners | Correct Detections | Missed Corners | False Detections | Multiple Detections | Total Error | Localization Error | Grade |
|---|---|---|---|---|---|---|---|---|
| Beaudet | 291 | 155 | 136 | 21 | 10 | 167 | 1.85 | 2.0 |
| Deriche-Giraudon | | 142 | 149 | 25 | 10 | 184 | 2.05 | 1.5 |
| Harris-Stephens | | 187 | 104 | 10 | 6 | 120 | 0.98 | 3.5 |
| Kitchen-Rosenfeld | | 163 | 128 | 26 | 15 | 169 | 1.87 | 2.0 |
| SUSAN | | 152 | 139 | 29 | 1 | 169 | 1.63 | 2.5 |
| Sojka | | 229 | 62 | 9 | 8 | 79 | 0.81 | 4.5 |

Comparing criterion:

- Total corners: the number of corners that is given by the reference ahead of time.
- Correct detections: the number of detected corners that have the same (or near) location with the reference corners.
- Missed corners: the number of reference corners subtracts the number of correct detections.
- False detections: the number of corners that are detected at a point at which (near which) no real corner exists.
- Multiple detections: the number of corners that are detected at a correct place but more than once.
- Total error: the sum of missed corners, false detections and multiple detections.
- Localization error: the average location deviation error of the total detections.
- Grade: the level of detection performance (from 0-5, higher score means better performance).

Analyzing from the above tables, the performance of the Beaudet detector, the Deriche-Giraudon detector, the Kitchen-Rosenfeld detector and the SUSAN detector is not so satisfied, there are a lot of errors take place when using these detectors. For the real image, the test result shows that the accurate of all the detector will fall down, but Sojka and Harris detector still can get very high score, especially, the Sojka detector, it can detect all the corners in image #1, detect 98.2% corners in image #2 and almost 80% corners in the real image, which is quite higher than the other detectors.

Finally, it is also necessary for a good detector to be stable, which means the detectors should get the similar results from the similar test image. Here we use the rotated image for the stable testing (see Table 2.9).

Table 2.9 The stability of the different kind of detectors (rotated from $+\pi/29$ to $+\pi/4$) and the total number of corners should be 470.

| Name | Rotate Angle | Correct Detection | Missed Corners | False Detection | Multiple Detection | Total Error | Localization Error | Grade |
|---|---|---|---|---|---|---|---|---|
| Beaudet | $\pi/29$ | 136 | 334 | 4 | 1 | 339 | 1.89 | 3 |
| | $\pi/13$ | 293 | 117 | 40 | 10 | 227 | 1.89 | |
| | $\pi/7$ | 114 | 356 | 4 | 0 | 360 | 1.76 | |
| | $\pi/4$ | 152 | 318 | 13 | 0 | 331 | 1.91 | |
| Harris-Stephens | $\pi/29$ | 376 | 94 | 10 | 7 | 111 | 1.36 | 4 |
| | $\pi/13$ | 399 | 71 | 14 | 7 | 92 | 1.22 | |
| | $\pi/7$ | 390 | 80 | 9 | 7 | 96 | 1.30 | |
| | $\pi/4$ | 370 | 100 | 10 | 5 | 115 | 1.41 | |
| Kitchen-Rosenfeld | $\pi/29$ | 346 | 124 | 77 | 14 | 215 | 1.97 | 3.5 |
| | $\pi/13$ | 342 | 128 | 78 | 24 | 230 | 1.56 | |
| | $\pi/7$ | 340 | 130 | 33 | 19 | 182 | 1.60 | |
| | $\pi/4$ | 326 | 144 | 13 | 6 | 163 | 1.66 | |
| Sojka | $\pi/29$ | 455 | 15 | 4 | 9 | 28 | 1.12 | 4.5 |
| | $\pi/13$ | 458 | 12 | 2 | 8 | 22 | 1.07 | |
| | $\pi/7$ | 463 | 7 | 4 | 7 | 18 | 1.10 | |
| | $\pi/4$ | 453 | 17 | 6 | 6 | 29 | 1.30 | |

From the Table 2.9, you can see the Sojka detector works best and the Beaudet detector is not so stable with the rotated images.

Until now all the necessary compares are shown in the above tables, from the result data, we find that the *boundary-based corner detectors* are a little bit computational complex, because they must based on the edge or contour information. And now it is not a popular method for corner detection. The *direct corner detectors* usually have faster processing speed than other kind of detectors. So they are very fitful for the real-time applications. But most of old *direct corner detectors* are not accurate enough for detecting the corners in real image with a lot of noise. Sojka's detector is an exception of them, it can work very accurately with the real images and the speed is just a little bit slower than the older ones.The *color distribution based approaches* work very good in the complex real image, but some of them (Compass) are quite slow and is impossible for lots of real time application. The Song-Lyu-Cai's detector is the best one in the *color distribution based approaches,* it can detect the edges, corners and junctions during the same round, and the processing speed is almost the same fast as the *direct corner detectors.* In Table 2.10, the final grades of all the corner detectors are given.

Table 2.10 Final Grade of all the corner detectors

| Detector Name | Feature Detected | Accurate | Speed | Real Time Processing | Grade |
|---|---|---|---|---|---|
| Beaudet | Corner | Low | Very fast | Yes | 3 |
| Harris-Stephens(Plessy) | Corner | High | Middle | Yes | 4 |
| Kitchen-Rosenfeld | Corner | Low | Fast | Yes | 3 |
| Deriche-Giraudon | Corner | Very low | | | 2 |
| SUSAN | Edge & Corner | Low | Middle | Yes | 3 |
| Song-Lyu-Cai | Corner, Edge & Junction | Very high | Middle | Yes | 4.5 |
| Compass | Corner, Edge, & Junction | Very high | Very slow | No | 3 |
| Sojka | Corner | Very high | Middle | Yes | 5 |

Because our project will be focus on the screen corner detection, the *direct corner detectors* are most fitful for the application, and among them, the Sojka's approach is the best one, which is satisfied with almost all the most important aspects of an idea corner detector, so we will probably choose it as the screen corner detector that will be used in our future application.

# Chapter 3

# 3 Classification, Regression Models

In the previous sections, we investigated many papers that use image-processing methods to detect corners and edges. But they are not enough for our purpose. In corner detection model, we want to select out not only general corners but also want to indicate the corners of the special screen, and for edge detection we cannot use it directly for screen detection without detailed recognition algorithm, which is also time consuming procedure. So another solution to finish this task is based on classification approaches. Without detecting the corners, edges, shapes or contours in one image, classification approaches are just to train a model with lots of sample images labeled with target and after it learn the parameters for the target, it can justify whether a new target is the target we are expecting by doing a simple computation. In this case, these classification tasks are to classify if a sample image is a screen corner or not. There are many classification approaches. Which is better for our tasks? And how they work? So in this section, we give our research on several papers focusing on classification methods theory, corresponding techniques implementation and some practical use.

## 3.1 K-Nearest Neighbors Classification

The basic classification method is K-Nearest Neighbors algorithm, the theory of which can be easily understood, and its performance is also not bad for some of classification cases. But its disadvantage is also obvious that is the low comparison speed given a huge training samples data set. Therefore, there are many algorithms to accelerate it and some of them success in speeding up it. The sections below show its theory and one promising accelerating algorithm.

### 3.1.1 Theory brief description

**[Web02]**

k-Nearest Neighbors (k-NN) algorithm, is a simple nonparametric technique of pattern recognition. For density estimation, its principle is very easy. The key issue is just to calculate the radius of the hyper sphere, which include a certain number of nearest neighbors inside it [Web02]. Then it can define the density of the point (the center of the hyper sphere) and realize classification by classify the current point to the class that has most points in this sphere.

However, the exhaustive k-NN search, which requires intensive dissimilarity computations, particularly for a large training set, becomes unacceptable. Accelerating the k-NN search has been an active research field in the past three decades. While non-metric dissimilarity measures have been briefly explained in pattern recognition literature, most of the existing fast algorithms for the k-NN search are effective only with metric dissimilarity measures.

### 3.1.2 Fast k-Nearest Neighbor Classification Using Cluster-Based Trees [Zha04]

In this paper, the author proposed a new cluster-based tree method to accelerate k-NN classification. The more important feature is this algorithm need not to any presuppositions about the metric form and properties of a dissimilarity measure, which is necessary for most of k-NN accelerating algorithms.

As the author summarized, the algorithms for speeding-up the k-NN search fall into two categories: template condensation and template reorganization. Template condensation removes redundant patterns in a template set [RIT75] and template reorganization restructures templates for efficient search of k nearest neighbors [BRO86]. Incorporating the template condensation rules into template reorganization leads to an innovative algorithm, the condensation-based tree algorithm by Brown [BRO95]. While a number of template reorganization algorithms rely on the essential properties of metric dissimilarity measures, the others, which are applicable for non-metrics, are only effective in low-dimensional feature spaces. The exception is the condensation-based tree algorithm, which is applicable for any dissimilarity measure, metric, or nonmetric. The condensation-based tree algorithm, however, is not sufficiently efficient. The algorithm requires intensive sorting operations in intermediate nodes when performing classification and the computation cost of sorting becomes substantial although dissimilarity computations can be largely pruned. So in this paper, they proposed a new method to accelerate k-NN classification with out any presuppositions about the metric form and properties of dissimilarity measures.

The cluster-based algorithm consists of two phases: tree generation and classification. Different from the existing tree classification methods, this algorithm introduces class-conditional clustering and establishes two decision levels for early decision making. A decision level in a tree is a level where each node and its sub-nodes have a unique class label. Thus, at a decision level, the class of an unseen template can be decided using k-NN classification. It is obviously, the bottom level of the tree, $B$, consisting of all templates in the set of training, **T**, is a decision level. Another decision level, the hyper-level, is generated through class-conditional clustering over **T**. The detailed tree generation procedure and classification procedure can be found [Zha04].

They tested their approach on NIST database and MNIST database respectively, the latter one is more difficult to recognize than the former one. The cluster tree structure after training was 4-level, 28,252,1564,53994 for NIST test and 29,337,2888,60000 for MNIST test. The results they obtained are quite well. Comparing to condense tree algorithm (Condense: 400), their approach performed much faster and executed much less computation under the same accuracy condition. Below are two tables to indicate their performance.

Table 3.1 Cluster tree and condensation tree (Condense: 400) performance on NIST

| | **Acc=99.03%** | | **Acc=99.17%** | | **Acc=99.24%** | | **Acc=99.28%** | |
|---|---|---|---|---|---|---|---|---|
| | Comp. | Time | Comp. | Time | Comp. | Time | Comp. | Time |
| Cluster | 992 | 0.447 | 1394 | 0.725 | 1737 | 1.080 | 2079 | 1.511 |
| Condense | 1316 | 1.782 | 1555 | 2.104 | 1839 | 2.492 | 2178 | 2.960 |

Table 3.2 Cluster tree and condensation tree (Condense: 400) performance on MNIST

| | **Acc=99.03%** | | **Acc=99.17%** | | **Acc=99.24%** | | **Acc=99.28%** | |
|---|---|---|---|---|---|---|---|---|
| | Comp. | Time | Comp. | Time | Comp. | Time | Comp. | Time |
| Cluster | 1115 | 0.506 | 1919 | 1.066 | 2329 | 1.484 | 3840 | 4.010 |
| Condense | 1580 | 1.998 | 2675 | 3.415 | 3509 | 4.524 | 4052 | 5.230 |

#### 3.1.2.1 Comments

We can draw a conclusion from the extensive experiments that the cluster tree method is a very good algorithm to accelerate k-NN classification. Its mechanism of early decision making and minimal side-operations for choosing searching paths largely contribute to the efficiency of classification through a cluster tree. Moreover, it can be easily tuned to fit any trade off between accuracy and speed by changing the value of $\varsigma$ utilized in the algorithm. The bigger $\varsigma$ is the higher accuracy but low recognition speed.

In addition, although the cluster tree generation is time consuming, the accuracy is comparable with SVM. In the test on MNIST database, cluster algorithm's accuracy is around 99%, the same with SVM 99%. The speed of classification is also very fast, and only uses 1.179-millisecond per sample 16.5 times faster than the exhaustive search, which reached a real-time application requirement.

## 3.2 Support Vector Machine

Though the last algorithm about KNN reaches very high classification accuracy and a high speed, for most of classification tasks people more likely choose the popular model, SVM, because of its high training speed and classification rates. Since the theory of SVM is known for every people, we give a brief overview for its theory part and give more papers about its practical use.

### 3.2.1 SVM Theoretical Overview

**[Vap95][Cor95]**

If use a brief sentence to describe SVM, it is that given a set of feature vectors which belong to either of two classes, a SVM finds the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyperplane. This hyperplane is called Optimal Separating Hyperplane (OSH), which is minimizes the risk of misclassifying not only the examples in the training set but also the examples of the test set. There are several cases in SVM, they are linearly separable case, linearly non-separable case, and non-linearly case.

In linearly separable case, assume, there is a set of training samples they are $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l$, $\mathbf{x}_i \in \Re^d$, and the target values are $y_1, y_2, \ldots, y_l$, $y_i \in \{1,-1\}$, which indicate the class of $\mathbf{x}_i$. Then if hyperplane $\mathbf{w}\mathbf{x}+b$ can separate these sample points, then we can express it as:

$$y_i(\mathbf{w}\mathbf{x}_i + b) \geq 1, \ \forall i \in \{1,\ldots,l\} \tag{3.1}$$

According to statistical theory, OSH should not only be a separable hyperplane, but also maximize the margin. So the problem is to minimize $|\mathbf{w}|$ in $d(\mathbf{w},b) = \frac{2}{|\mathbf{w}|}$ with the subject to the constrain of Eq. (3.1). Based on Lagrange multipliers method, the problem transform to minimize: $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$, with subject to constrain, $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$, $\forall i \in \{1,\ldots,l\}$. Every Lagrange multiplier is associated with one sample in the training set. And those samples whose $\alpha_i \geq 0$, are called support vectors. The final classification function we obtained is:

$$f(\mathbf{x}_*) = \mathrm{sgn}(\sum_i^N \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}_* + b) = \mathrm{sgn}(\mathbf{w}\mathbf{x}_* + b), \tag{3.2}$$

where N is the number of support vectors.

For the linearly non-separable case, we cannot use the optimization method and SVM introduced an error term and the problem turns to minimize: $|\mathbf{w}|^2 + C(\sum_i \xi_i)$. In this case, support vectors are the points on the hyperplane, and the samples were classified incorrectly.

Another case in SVM is that the problem is non-linearly case, to deal with this, SVM utilize the kernel function to solve the transform from non-linear to linear. By introducing a kernel function, the new classification function change to:

$$f(\mathbf{x}_*) = \mathrm{sgn}(\sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_*) + b) = \mathrm{sgn}(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_*) + b) \quad (3.3)$$

In the SVM, kernel function cannot be used arbitrarily. The kernel function must be a function satisfying Mercer's condition, such functions as, polynomial function $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}\mathbf{y} + 1)^p$, Gaussian function $K(\mathbf{x}, \mathbf{y}) = e^{-|\mathbf{x}-\mathbf{y}|^2/2\sigma^2}$, and etc.

### 3.2.2 Support Vector Machines for 3D object Recognition

**[Pon98]**

The aim of this paper is to introduce the theory of SVM and illustrate the potential of SVMs on a computer vision problem, the recognition of 3D objects from single images.

In this paper, the authors implemented SVM as a classifier and test it on the COIL (Columbia Object Image Library database) consisting of 7,200 images of 100 objects. Half of the images were used as training examples, the remaining half as test images. They discarded color information and tested the method on the remaining images corrupted by synthetically generated noise, bias, and occlusions. The remarkable recognition rates achieved in the performed experiments indicate that SVMs are well suited for aspect-based recognition. Comparisons with other pattern recognition methods, like perceptions, show that the proposed method is far more robust in the presence of noise.

In their recognition system, they did some pre-processing to the images like reducing the resolution of the images from 128x128 to 32x32 and change color image to gray level by function, E= 0.31R + 0.59G + 0.10B. After that by using 36 images for each of the 32 objects and the test sets of the remaining 36 images for each object, they got the results below:

For plain images they got average error rates (A.E.R) as: 0.03%
For noise-corrupted images they got A.E.R as: 0.3%(±25),1.6%(±100) [2]
For shifted images they got A.E.R as: 0.6% (3 pixels), 2.0% (5 pixels)
For shifted and noise-corrupted images, A.E.R are: 0.6%(±25),1.8%(±100)
For occlusions test, they did two series of experiments. In the first series they randomly selected a subwindow (k by k) in the test images and assigned a random value between zero and 255 to the pixels inside the subwindow. In the second series, they randomly selected n columns and m rows in the rescaled images and assigned a random value to the corresponding pixels. (See Fig. 3.1)

[2] The value in the brackets is a criterion to indicate noise level.

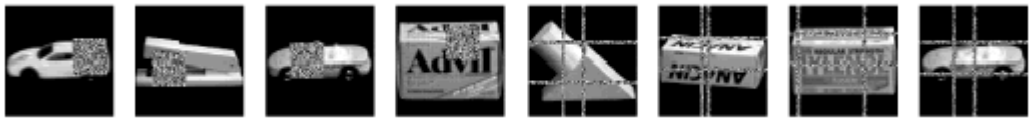



Figure 3.1 Occluded objects correctly classified by the system

The A.E.R of the first test of occlusions is: 0.7% (k=4), 2,0% (k=6)
The A.E.R of the second test of occlusions is: 2.1% (n=1,m=1), 6.1%(n=2,m=2)

In these experiments, by inspection of the obtained results, they mentioned that most of the errors were due the three chewing gum packets that become practically indistinguishable as the noise increases. After leaving out two of them, the result is much better. About detailed experiments data, you can refer to the original paper.
In addition, the authors also compare SVM with Single Perceptrons method (SP) and Average Perceptrons method (AP), and got the result that for the case if the noise is very big, then the SVM accuracy is much higher than them.
I.e. noise=$\pm$50 , A.E.R: SP=2.8%, AP=1.7%, SVM=**0.0**%
noise=$\pm$250 , A.E.R: SP=29.3%, AP=20.2%, SVM=**0.0**%

#### 3.2.2.1 Comments

From this paper, we got known that SVM perform very well on 3D objects recognition and by the experiments result it is very flexible and tolerant for some image transformation, such as, noise, shift and occlusion. The most important result is the occlusion case, and it shows that SVM even works with lack of content, which tell us we can use SVM to classify content unstable objects, such as screen corners. In addition, the author compared SVM to perceptron methods, and proofed the former is more robust.

### 3.2.3 Visual Object Recognition With Supervised Learning

**[Hei03]**

This paper gives a component-based approach to visual object recognition rooted in supervised learning, which allows for a vision system that is more robust against changes in an object's pose or illumination. The author mentioned that generally people approach object categorization by representing all the search window's contents by one feature vector that is fed to a single classifier. This global approach worked well for detecting objects under fixed viewing conditions. However, problems occur when the objects' viewpoint and pose vary especially when the training set doesn't cover all viewing variations in the test set. For example, a face detection system that is trained on frontal, upright faces was tested on small-rotated faces and got a worse result. To overcome this problem, they developed a component-based approach that breaks the object into a set of components that are interconnected by a flexible geometrical model.

They developed a two-level classification system (See Fig.3.2) for face detection that implies geometrical relations between components. On the first level, face components are detected. On the second level, the system combine all components' location into one feature vector and send it to a classifier trained by geometrical model of a face and check if it is correct one.

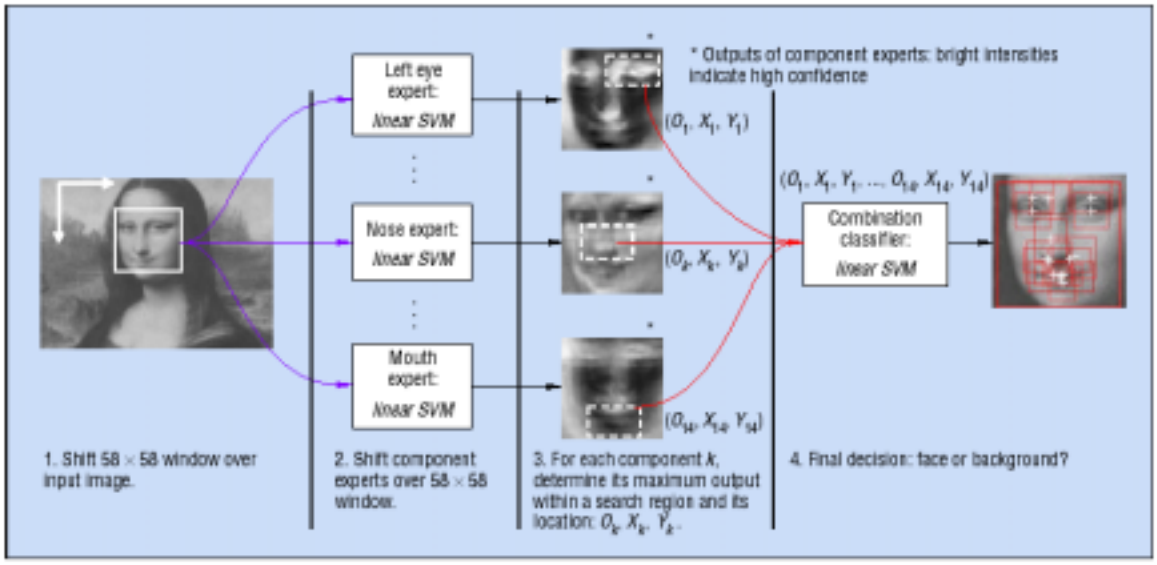



Figure 3.2 System overview of the component-based detection system.

For the second level, it is not very difficult, just realized by one linear SVM. But in the first level, there is one issue is not easy that is instead of manually choosing the components, it would make more sense to choose automatically based on their discriminative power and robustness against pose and illumination changes. In this paper, the author raised an algorithm to do that. The algorithm started with a small, rectangular component located around a pre-selected point on the face (center of the left eye, for example). The algorithm extracted the component from all face images to build a training set of positive examples. They also generated a training set of non-face patterns that had the same rectangular shape as the face component. After training an SVM on the component data, they determined the SVM's performance based on a rough estimate $\tilde{L}$ of given by $\tilde{L} = \frac{1}{l}\frac{R^2}{M^2}$, which is a quantity that indicates the expected error of the SVM. After determining this $\tilde{L}$, they enlarged the component by expanding the rectangle into one of the four directions. Again, they generated training data, trained an SVM, and determined $\tilde{L}$. They did this for expansions in all four directions and kept the expansion that decreased the most. Continued this process until the expansions in all four directions led to an increase of $\tilde{L}$. Finally, they got 14 components for human face.

### 3.2.3.1 Comments

Comparing to a global classifier trained on the whole face pattern, the component-based system got a much better result. (See Fig.3.3)

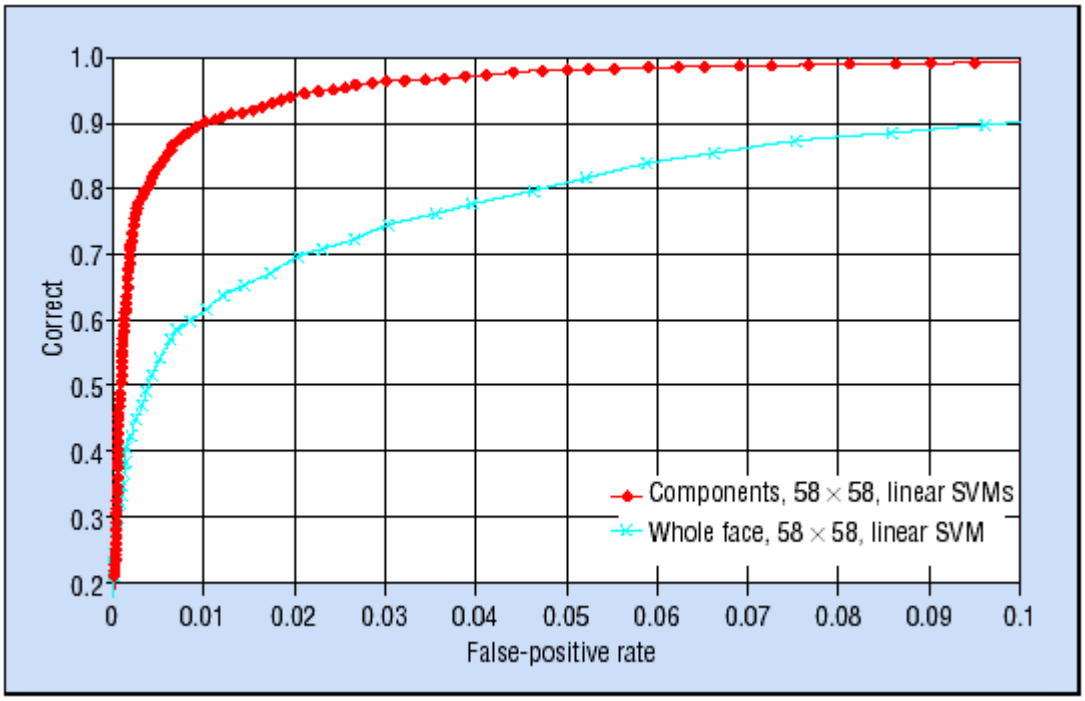



Figure 3.3 Characteristic curves for a linear whole face classifier and a component classifier consisting of 14 linear component classifiers and a linear combination classifier.

The important hint we got from this paper is that perhaps we should not fix an object into one feature vector. By decomposing it into several feature components, may be we can get much better classification rates.

### 3.2.4 Feature-Based Shape Recognition by Support Vector Machine

**[Ard00]**

A model identification technique for the objects in a gray level image is proposed, based on the extraction of a compact shape feature in terms of the statistical variance pattern of the objects' surface. A shape recognition system has been developed, that detects automatically image ROIs (region-of-interesting) containing single objects, and classifies them as belonging to a particular class of shapes, which are cube, cylinder, pyramid, cone, ellipsoid, and box. Fig.3.4 is the result of this system

Figure 3.4 Detection result of this system

The two important information we got from this paper are, the first, they use the eigen values of the covariance matrix computed from the pixel rows of a single ROI, and arranged these quantities in a vector, then classified using Support Vector Machines (SVMs). The selected feature allows them to recognize shapes in a robust fashion, despite rotations or scaling, and, to some extent, independently from the light conditions. The second thing is, that they used two-level classification strategy in their system. Two groups of SVMs were trained and separately for one-versus-others classification phase and pair-wise classification phase. They tested their classification strategy and got some interesting results.

Below is a brief description to two important phases in their system:

- **Eignenvlaues as Feature Vectors:** In their approach, the image is automatically scanned to locate ROIs containing single objects. The objects' shape is described in terms of the eigenvalues of the covariance matrix computed from the pixel rows of the ROI by KL transform [Gon87][Pen91][Tai98]: the eigenvalues are arranged as a vector, $\lambda$. They analyzed the histogram of the components of $\lambda$ computed from several images both synthetic and real, depicting single shapes under varying attitudes and lighting. This histogram performs as an "almost invariant" under varying illuminant conditions and attitude of the object. The histogram exhibits some dominant modes, whose relative position and amplitude depend on the shape observed. The amplitude and position of these histogram modes remain almost unchanged under rotation, translation, and scaling of the object. In addition, after experiments, the light direction affects in a uniform manner all the components of $\lambda$. So they have experimental evidence that in their setup varying l doesn't affect the histogram too much. The almost invariant behavior of the $\lambda$ vector implies that similar shapes tend to form clusters in the feature space.

- **2-level classification procedure:** Like all other object recognition algorithm, before the classification, their system uses a search algorithm to realize localization. The search algorithm they implemented is based on a two-pass strategy. The first step performs a rough location of the ROIs for the horizontal and vertical displacement. The second step defines the windows' dimensions for all the selected positions. The principle of this search is based on the maximization of correlation between the actual $\lambda$ vector and some sample vectors from the different shape classes, we don't describe more here.

  After searching out the ROIs, the classification phase comes. The SVM in its original formulation is designed for two-class discrimination, so they used a particular training strategy, in order to cope with the multi-class task. Two different kinds of SVMs have been trained on six shape classes: cube, cylinder, pyramid, cone, ellipsoid, and box. First, six SVMs have been trained in a *one-versus-others* fashion, each of them being able to discriminate between a particular class and all other objects. Besides, a second pool of 15 SVMs have been trained using a *pair-wise* strategy: each SVM is trained to discriminate between a single pair of the desired classes, so for $K$ classes it need $K(K\text{-}1)/2$ different machines.

### 3.2.4.1 Comments

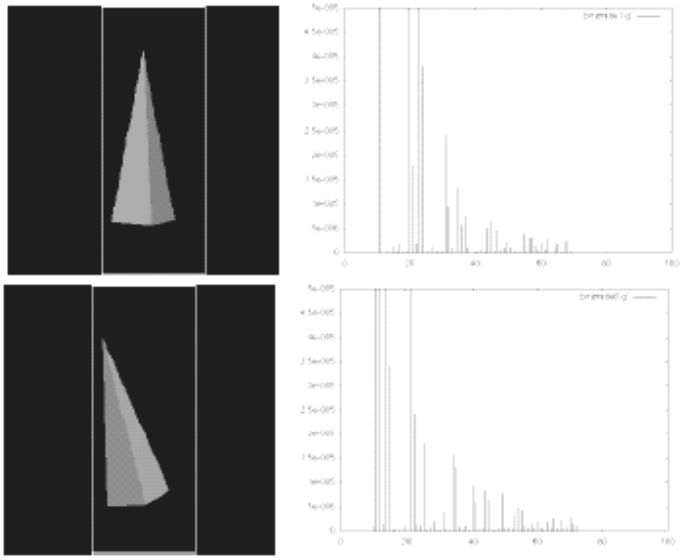


Figure 3.5 Shape samples (left), relative vector $\lambda$ histogram (right)

By their feature vector extraction method and 2-level classification strategy, they got a not bad result. But the interesting thing is that the 2-level classification strategy did not help a lot to their system. As they imagined, if a *one-versus-others* training leaves some uncertainty regions in the feature spaces where they are not able to decide correctly to which class belongs the actual sample, then to provide a refinement of the boundary locations between multiple classes is the use of a *pair-wise* learning strategy. But their result shows *pair-wise* classification only refined one of test cases. In addition, it was wrong due to the closeness between the two shape classes. This conclusion told us that *pair-wise* strategy for multi-classification does not prior to the *one-versus-others* strategy in some case. So we should choose carefully from them when we deal with multi-classification problem.

Another very useful conclusion of their paper is the KLT eigenvalues feature vector is insensible to the shape displacement and transforms a feature in a compact way, which indicates it is a good feature extraction method and give us another good choice when we are facing feature problems. (See Fig.3.5)

## 3.3 Relevance Vector Machine

SVM is very popular for classification tasks, there are huge existing projects using this model. But a disadvantage of SVM is that it is a hard classification method, which cannot supply a probability output for this classification result. For some cases, maybe it is not important, but for some projects the probability value for a classified target will be very helpful information. So there are some papers add some calculation steps into SVM to get a score for the classification, but they are not based real Bayesian theory. To solve this problem, Tipping give us an alternative model to SVM, which is called relevance vector machine. Because RVM is a new model, there are few papers to talk about its practical use, so we investigate more papers on its theory.

### 3.3.1 Sparse Bayesian Learning and the Relevance Vector Machine

**[Tip01]**

#### 3.3.1.1 What is RVM?

Relevance Vector Machine, RVM for abbreviation, is a new classification method based on Bayesian framework. The model it utilizing is identical functional form to the popular and state-of-the-art "SVM" discussed in above sections. The differences are that by exploiting a probabilistic Bayesian learning framework, the author derive accurate prediction models which typically utilize dramatically fewer basis functions than a comparable SVM while offering a number of additional advantages, which include the benefits of probabilistic predictions, automatic estimation of parameters, and the facility to utilize arbitrary basis functions, which are not necessary to be 'Mercer' kernels.

In supervised learning we are given a set of examples of input vectors $\{\mathbf{x}_n\}_{n=1}^N$ along with corresponding targets $\{t_n\}_{n=1}^N$, the latter of which might be real values (in regression) or class labels (classification). From this 'training' set we wish to learn a model of the dependency of the targets on the inputs with the objective of making accurate predictions of *t* for previously unseen values of x.

Typically, we base our predictions upon some function y(x) defined over the input space, and 'learning' is the process of inferring this function. The popular y(x) is that of the form:

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(\mathbf{x}), \tag{3.4}$$

Where the output is a linearly weighted sum of M, generally non-linear and fixed, basis functions $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_M(\mathbf{x}))^{\mathrm{T}}$. The objective is to estimate good values for parameters $\mathbf{w} = (w_1, w_2, ..., w_M)^{\mathrm{T}}$.

Above theory is the same with SVM, but the key feature of RVM is that as well as offering good generalization performance, the inferred predictors are exceedingly sparse in that they contain relatively few non-zero $w_i$ parameters. The majority of parameters are automatically set to zero during the learning process, by a Bayesian probabilistic framework learning in general models of the form Eq. (3.4). In addition, comparing to RVM, SVM has some defects, such as:

- Although relatively sparse, SVM make unnecessarily liberal use of basis functions. Some form of post-processing is often required to reduce computational complexity.
- SVM predictions are not probabilistic.
- Have to estimate the error/margin trade-off parameter 'C', which is wasteful both of data and computation.
- The kernel function $K(\mathbf{x}, \mathbf{x}_i)$ must satisfy Mercer's condition.

Instead, RVM does not suffer from any of these limitations. The author adopt a fully probabilistic framework and introduce a prior over the model weights governed by a set of hyperparameters, one associated with each weight, whose most probable values are iteratively estimated from the data. Sparsity then is achieved because in practice the result shows that the posterior distributions of many of the weights are sharply (indeed infinitely) peaked around zero. The author term those training vectors associated with the remaining non-zero weights "relevance' vectors.

#### 3.3.1.2 RVM regression model specification

The author define the standard probabilistic formulation as:

$$t_n = y(\mathbf{x}_n; \mathbf{w}) + \varepsilon_n, \tag{3.5}$$

Where $\varepsilon_n$ are independent samples from some noise process, mean-zero Gaussian with variance $\sigma^2$ . Thus $p(t_n \mid \mathbf{x}) = N(t_n \mid y(\mathbf{x}_n), \sigma^2)$ and the function *y* is Eq. (3.4). He also defined his basis function as: $\phi_i(\mathbf{x}) \equiv K(\mathbf{x}, \mathbf{x}_i)$, so he got the likelihood of the complete data set as:

$$p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{t} - \mathbf{\Phi}\mathbf{w}\|^2\right\}, \tag{3.6}$$

where $\mathbf{t} = (t_1 \cdots t_N)^T, \mathbf{w} = (w_0 \cdots w_N)^T$ and $\mathbf{\Phi}$ is the $N \times (N+1)$ matrix with $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)]^T$, wherein $\phi(\mathbf{x}_n) = [1, K(\mathbf{x}_n, \mathbf{x}_1), K(\mathbf{x}_n, \mathbf{x}_2), \ldots, K(\mathbf{x}_n, \mathbf{x}_N)]^T$.
To avoid over-fitting by maximum-likelihood estimation of w and $\sigma^2$ from Eq. (3.6), the author introduced a set of hyperparameters $\boldsymbol{\alpha}$, each of elements control a weight parameter, and chose a zero-mean Gaussian prior distribution over w:

$$p(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{i=0}^{N} N(w_i \mid 0, \alpha^{-1}), \tag{3.7}$$

He also defined the prior distribution for $\sigma^2$ and $\boldsymbol{\alpha}$, which are two Gamma distribution: $p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} \text{Gamma}(\alpha_i \mid a, b)$, and $p(\sigma^{-2}) = \text{Gamma}(\beta \mid c, d)$.
Actually, we want to know the predictive target $t_*$ distribution, given a new test point $\mathbf{x}_*$, by Bayes' rules we can write the distribution as following expression:

$$p(t_* \mid \mathbf{t}) = \int p(t_* \mid \mathbf{w}, \boldsymbol{\alpha}, \sigma^2) p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2 \mid \mathbf{t}) d\mathbf{w} d\boldsymbol{\alpha} d\sigma^2 , \tag{3.8}$$

But $p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2 \mid \mathbf{t})$ cannot be computed so the author depose the posterior as:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2 \mid \mathbf{t}) = p(\mathbf{w} \mid \mathbf{t}, \boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}, \sigma^2 \mid \mathbf{t}), \tag{3.9}$$

Where, the first term is directly computable by Bayes rules:

$$p(\mathbf{w} \mid \mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = (2\pi)^{-(N+1)/2} |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{w} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{w} - \boldsymbol{\mu}) \right\}, \tag{3.10}$$

Where the posterior covariance and mean are respectively:

$$\boldsymbol{\Sigma} = (\sigma^{-2} \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \mathbf{A})^{-1}, \quad \boldsymbol{\mu} = \sigma^{-2} \boldsymbol{\Sigma} \boldsymbol{\Phi}^{\mathrm{t}} \mathbf{t} \text{, with } \mathbf{A} = \mathrm{diag}(\alpha_0, \alpha_1, \ldots, \alpha_N) \tag{3.11}$$

For the second term of Eq.(3.9), the value have to be learned. Because of $p(\boldsymbol{\alpha}, \sigma^2 \mid \mathbf{t}) \propto p(\mathbf{t} \mid \boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}) p(\sigma^2)$, the author just maximized the term $p(\mathbf{t} \mid \boldsymbol{\alpha}, \sigma^2)$ with respect to $\sigma^2$ and $\boldsymbol{\alpha}$ by using type-II maximum likelihood method. The detailed hyperparameters optimizing procedure can be found in the original paper.

After the maximization, we can get the final $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and by passing them into below function, then given a new datum $\mathbf{x}_*$, we can get predicative target value and noise variance: $y_* = \boldsymbol{\mu}^{\mathrm{T}} \phi(\mathbf{x}_*)$, $\sigma_*^2 = \sigma_{MP}^2 + \phi(\mathbf{x}_*)^T \boldsymbol{\Sigma} \phi(\mathbf{x}_*)$, and predictive distribution is: $p(t_* \mid \mathbf{t}) = N(t_* \mid y_*, \sigma_*^2)$.

#### 3.3.1.3 RVM classification

The sparse Bayesian learning let RVM classification works like regression very much. RVM classification follows an essentially identical framework as detailed for regression, but the different is there is no 'noise' variance here, and likelihood is changed to:

$$P(t \mid w) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x}_n; \mathbf{w})\}^{t_n} [1 - \sigma\{y(\mathbf{x}_n; \mathbf{w})\}]^{1-t_n} \text{ , Where } t_n \in \{0,1\} \tag{3.12}$$

and $\sigma(y) = 1/(1 + e^{-y})$ is applied by following statistical convention.

But in classification case, weights cannot be computed analytically, so the author used an approximation procedure based on Laplace's method. The detailed procedure can be found in the original paper. After the approximation, we can get:

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^{\mathrm{T}} \mathbf{B} \boldsymbol{\Phi} + \mathbf{A})^{-1} \tag{3.13}$$

and

$$\mathbf{w}_{\mathrm{MP}} = \boldsymbol{\Sigma} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{B} \mathbf{t} \tag{3.14}$$

Using these two statistics $\boldsymbol{\Sigma}$ and $\mathbf{w}_{MP}$, the hyperparameters $\boldsymbol{\alpha}$ are updated using the same expression with regression case. The final $\mathbf{w}_{MP}$ are the parameters we are expecting, and in the practice both in regression case and the classification case, most of them are zero, so that the sparsity is accomplished.

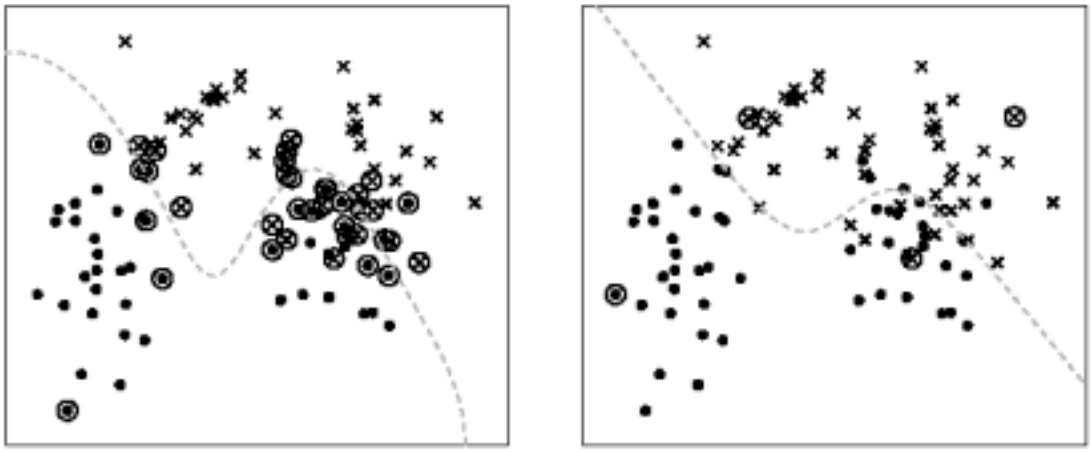


Figure 3.6 SVM (left) and RVM (right) classifiers on 100 examples from Ripley's Gaussian-mixture data set. The decision boundary is shown dashed, and support/relevance vectors are shown.

#### 3.3.1.4 Comments

In this paper, the author compared his RVM with the popular SVM, and we show two of the tables here. They show respectively the accuracy and "important" vectors number in regression test and classification test.

Table 3.3 Comparison table for regression

| | | | **SVM** | **RVM** | **SVM** | **RVM** |
|---|---|---|---|---|---|---|
| *Data Set* | *Num* | *Dim.* | *Errors* | | *Number of vectors* | |
| Sinc (Gaussian noise) | 100 | 1 | 0.378 | 0.326 | 45.2 | 6.7 |
| Sinc (Uniform noise) | 100 | 1 | 0.215 | 0.187 | 44.3 | 7.0 |
| Friedman #2 | 240 | 4 | 4140 | 3505 | 110.3 | 6.9 |
| Friedman #3 | 240 | 4 | 0.0202 | 0.0164 | 106.5 | 11.5 |
| Boston Housing | 481 | 13 | 8.04 | 7.46 | 142.8 | 39.0 |
| Normalized Mean | | | 1.00 | **0.86** | 1.00 | **0.15** |

Table 3.4 Comparison table for classification

| | | | SVM | RVM | SVM | RVM |
|---|---|---|---|---|---|---|
| *Data Set* | *Num* | *Dim* | *Errors (%)* | | *Number of vectors* | |
| Pima Diabetes | 200 | 8 | 20.1 | 19.6 | 109 | 4 |
| U.S.P.S. | 7291 | 256 | 4.4 | 5.1 | 2540 | 316 |
| Banana | 400 | 2 | 10.9 | 10.8 | 135.2 | 11.4 |
| Breast Cancer | 200 | 9 | 26.9 | 29.9 | 116.7 | 6.3 |
| Titanic | 150 | 3 | 22.1 | 23.0 | 93.7 | 65.3 |
| Waveform | 400 | 21 | 10.3 | 10.9 | 146.4 | 14.6 |
| German | 700 | 20 | 22.6 | 22.2 | 411.2 | 12.5 |
| Image | 1300 | 18 | 3.0 | 3.9 | 166.6 | 34.6 |
| Normalized Mean | | | **1.00** | 1.08 | 1.00 | **0.17** |

From the above tables, we can see that RVM's accuracy keep the same level with SVM and give more advantages, such as fewer basis functions, probabilistic output, and etc. But the disadvantage of RVM is obvious as well, that is it need much longer time on learning then SVM, in this time, SVM is extremely high speed. But the author gives us a faster algorithm in the next paper.

### 3.3.2 Fast Marginal Likelihood Maximization for Sparse Bayesian Models

**[TipNEW]**

In the previous papers, we introduce the sparse Bayesian modeling approach, which benefits a number of advantages comparing to SVM. But the defect of RVM is that its training time is very long. In this paper, Tipping gives us a new algorithm, which exploits recently elucidated properties of marginal likelihood function to enable maximization via a principled and efficient sequential addition and deletion of candidate basis functions.

From the previous RVM introduction paper [Tip01], we have known the key step for RVM is to use type-II maximum likelihood procedure to estimate $\alpha_{MP}$. That is, sparse Bayesian learning is formulated as the (local) maximization with respect to $\alpha$ of the marginal likelihood, or equivalently, its logarithm $\ell(\boldsymbol{\alpha})$:

$$\ell(\boldsymbol{\alpha}) = \log p(\mathbf{t} \mid \boldsymbol{\alpha}, \sigma) = \log \int_{-\infty}^{\infty} p(\mathbf{t} \mid \mathbf{w}, \sigma^2) p(\mathbf{w} \mid \boldsymbol{\alpha}) d\mathbf{w} = -\frac{1}{2}[N \log 2\pi + \log |\mathbf{C}| + \mathbf{t}^{\mathrm{T}} \mathbf{C}^{-1} \mathbf{t}]$$

With

$$\mathbf{C} = \sigma \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \tag{3.15}$$

In this paper, the author raises a new algorithm for maximization of this marginal likelihood, which is dependent on its properties. The author figures out some key properties in this paper, which can increase the effectiveness.

These two properties are denoted as $s_i$ and $q_i$ which definition can be found in this paper. The important thing is through analysis of reformulated formula $\ell(\boldsymbol{\alpha})$, we will get following conclusion that $\ell(\boldsymbol{\alpha})$ has a unique maximum with respect to $\alpha_i$:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}, \qquad \text{If } q_i^2 > s_i,$$
$$\alpha_i = \infty, \qquad \text{If } q_i^2 \le s_i. \tag{3.16}$$

Another two important rules with respect to these two quantities are:

- If $\phi_i$ is 'in the model' (i.e. $\alpha_i < \infty$) yet $q_i^2 \le s_i$, then $\phi_i$ may be deleted (i.e. $\alpha_i$ set to $\infty$),
- If $\phi_i$ is excluded from the model ($\alpha_i = \infty$) and $q_i^2 > s_i$, $\phi_i$, may be added (i.e. $\alpha_i$ is set to some optimal finite value).

In both these cases, the algorithm can make discrete changes to the model structure while at the same time increasing the marginal likelihood objective function.

The proposed marginal likelihood maximization algorithm is as follows:

1. If regression initialize $\sigma^2$ to some sensible value (e.g. var[t] x 0.1).
2. Initialize with a single basis vector $\phi_i$, setting, $\alpha_i = \dfrac{\|\phi_i\|^2}{\|\phi_i^{\mathrm{T}}\mathbf{t}\|^2 / \|\phi_i\|^2 - \sigma^2}$. All other $\alpha_m$ are notionally set to infinity.
3. Explicitly compute $\sum$ and $\boldsymbol{\mu}$ (which are scalars initially), along with initial values of $s_m$ and $q_m$ for all M bases $\phi_m$.
4. Select a candidate basis vector $\phi_i$ from the set of all M.
5. Compute $\theta_i \cong q_i^2 - s_i$.
   If $\theta_i > 0$ and $\alpha_i < \infty$ (i.e. $\phi_i$ is in the model), re-estimate $\alpha_i$.
   If $\theta_i > 0$ and $\alpha_i = \infty$, add $\phi_i$ to the model with updated $\alpha_i$.
   If $\theta_i \le 0$ and $\alpha_i < \infty$, then delete $\phi_i$ from the model and set $\alpha_i = \infty$.
6. If regression and estimating the noise level, update $\sigma^2$.
7. Recompute/update $\sum$, $\boldsymbol{\mu}$, and all $s_m$ and $q_m$.
8. If converged terminate, otherwise go to 4.

#### 3.3.2.1 Comments

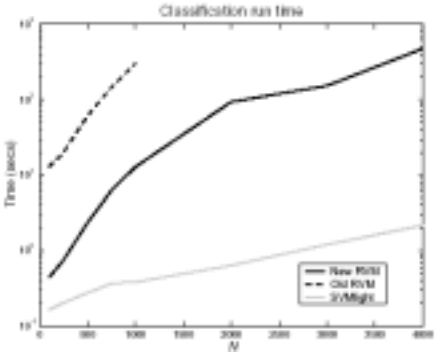



Figure 3.7: Classification Comparison Between SVM, old RVM and new RVM

The author compares this algorithm in this paper with the old RVM and one of implementation of RVM called SVM$^{light}$. (See Table 3.5 and Fig.3.7):

From these figures, we can get that the new RVM, is highly effectively. Though it still much slower than SVM but it offers a very clear speed advantage over the originally proposed approach. Another advantage it offers is that it use less basic functions than old RVM, so together with a number of

advantageous features when compared with SVM in previous papers, RVM now shows more powerful.

Table 3.5 Comparison between old RVM, new RVM and SVM (samples num N=1000)

| | **Regression** | **Classification** |
|---|---|---|
| Old RVM | 4 mins 17 secs | 4 mins 58 secs |
| New RVM | 14.42 secs | 12.84 secs |
| SVM$^{light}$ | 1.03 secs | 0.38 secs |

## 3.4 Markov Models and Hidden Markov Models

If we talk about objects recognition, the model we choose maybe one of them mentioned above. But if we talk about speech recognition, gesture recognition, especially in face recognition, the topic we cannot avoid is HMM, which is a very powerful mathematical model and play a very important role in many areas. So we investigated some papers talking about this technology, which might be useful for our later project. In this section, we chose some papers to describe briefly to its theory and give its practical use in face recognition. And for its use in gesture recognition, we give more papers in the next chapter.

### 3.4.1 MM and HMM Theory Overview

**[Lus95]**

In this paper, the author used one very simple case to explain Markov model and Hidden Markov Models. The example he gave is the weather, which also is taken as an example by other people.

For Markov model, it can be easy to understand that the key is to use Markov assumption that assume $\mathrm{P}(w_n \mid w_{n-1}, w_{n-2}, \ldots, w_1) \approx \mathrm{P}(w_n \mid w_{n-1})$, which simplified the calculation of the probability based on a transition probability matrix. In weather case, it is to say that we can expect tomorrow weather just based on of today instead of other history days.

As to hidden Markov model, the case is more complicated. In this model, the idea is that we cannot know the history states when we want to make a predication, the only information we got are some observations that are related somehow to the states. So we call the states are hidden behind some observation. In weather case, the author supposed that someone was locked in one room and don't know the weather outside. If that person wants to predict the weather, the only piece of evidence he have is whether the person who comes into the room carrying his daily meal is carrying an umbrella or not. So in this case, the observation is the appearance of the umbrella and the states still are the weather condition hidden behind.

The more real instance he gave is about speech recognition.
In speech recognition, the basic idea is to find the most likely string of words given some acoustic input, or:

$$\arg\max_{w \in L} \mathrm{P}(w \mid y) \tag{3.17}$$

Where w is a string of words, L is the language we are interested in, and y is the set of acoustic vectors that you've gotten from your end processor. To compare this to the weather example, the acoustics are observations, similar to the umbrella observations, and the words are similar to the weather on successive days. Remember that the basic equation of speech recognition is Bayes' Rule:

$$\arg\max_{w \in L} \mathrm{P}(w \mid y) = \arg\max_{w \in L} \frac{\mathrm{P}(y \mid w)\mathrm{P}(w)}{\mathrm{P}(y)} \tag{3.18}$$

For a single speech input (e.g. one sentence), the acoustics (y) will be constant, so will P(y), therefore we only need to find:

$$\arg\max_{w \in L} \mathrm{P}(y \mid w)\mathrm{P}(w) \tag{3.19}$$

The first part of this expression is called the model likelihood, and the second part is a prior probability of the word string. Then the author gave a simple word "of" pronunciation example:

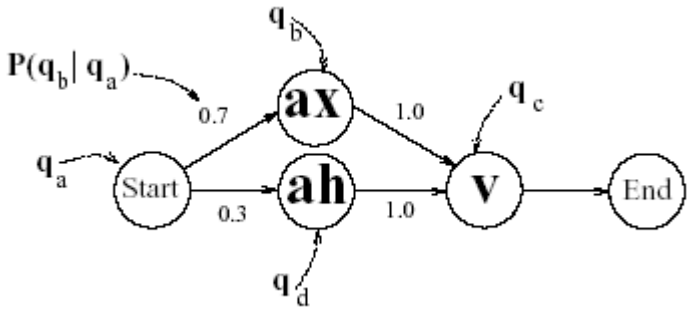



Figure 3.8 Word "of" model

$$\mathrm{P}(y \mid w_i) = \mathrm{P}(y \mid Start, ax, v, End) + \mathrm{P}(y \mid Start, ah, v, End)$$
$$\mathrm{P}(y \mid w_{of}) = \mathrm{P}(q_b \mid q_a)\mathrm{P}(y_0 \mid q_b)\mathrm{P}(q_c \mid q_d)\mathrm{P}(y_1 \mid q_c) + \mathrm{P}(q_d \mid q_a)\mathrm{P}(y_0 \mid q_d)\mathrm{P}(q_c \mid q_d)\mathrm{P}(y_1 \mid q_c)$$

The individual $\mathrm{P}(y_i \mid q_i)$ are given by a Gaussian or multi-layer-perceptron likelihood estimator. The transitions $\mathrm{P}(q_i \mid q_j)$ depend on the pronunciations of words. The author made a simplifying assumption here that for the word "of" he only have two acoustic vectors, y0 and y1.

### 3.4.2 Hidden Markov Models Structure

**[Nef98]**

HMM consist of two interrelated processes: (1) an underlying, unobservable Markov chain with a finite number of states, a state transition probability matrix and an initial state probability distribution and (2) a set of probability density functions associated with each state. The elements of a HMM are:

1. N, the number of states in the model. If $S$ is the set of states, then $S = \{S_1, S_2, \ldots, S_N\}$. The state of the model at time $t$ is given by $q_t \in S$, $1 \le t \le T$, where $T$ is the length of the observation sequence (number of frames).
2. M, the number of different observation symbols. If V is the set of all possible observation symbols (also called the codebook of the model), then $V = \{v_1, v_2, \ldots, v_M\}$.
3. A, the state transition probability matrix, i.e. A=$\{a_{ij}\}$where

$a_{ij} = \mathrm{P}[q_t = S_j \mid q_{t-1} = S_i] 1 \le i, j \le N$

with the constraint,

$0 \le a_{i,j} \le 1$, and $\sum_{j=1}^{N} a_{i,j} = 1,\ 1 \le i \le N$

4. B, the observation symbol probability matrix, i.e. $\mathbf{B}=\{b_j(k)\}$, where

   $b_j(k) = \mathrm{P}[\mathbf{O_t} = v_k \mid q_t = S_j],\ 1 \le i \le N, 1 \le j \le N$

   and $\mathbf{O_t}$ is the observation symbol at time t.

5. $\mathbf{\Pi}$, the initial state distribution, i.e. $\mathbf{\Pi} = \{\pi_i\}$ where: $\pi_i = \mathrm{P}[q_1 = S_i]$, $1 \le i \le N$

Using a shorthand notation, a HMM is defined as the triplet $\lambda = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$.

### 3.4.3 Hidden Markov Models for Face Recognition

**[Nef98]**

The author made use of HMM to recognize face, which is ready proved the best method to go in face recognition. In this paper and the paper following, we can get a brief concept about how to use HMM to detect faces and recognize them.

The author constructed a 5-states HMM in his paper, which is based on the significant facial regions hair, forehead, eyes, nose and mouth. Each of these facial regions is assigned to a state in a left to right 1D continuous HMM. The state structure of the face model and the non-zero transition probabilities $a_{ij}$ are shown below.

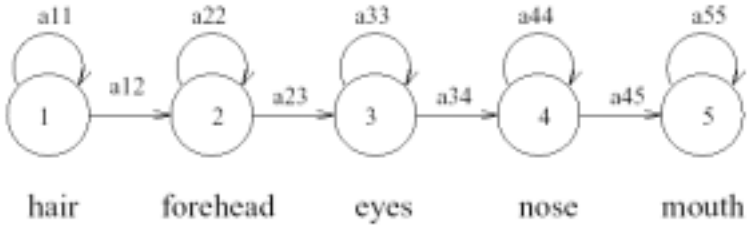



Figure 3.9: Left to right HMM for face recognition

During the process of training the face models, a face database was used. Each individual in the database is represented by a HMM face model. A set of five images representing different instances of the same face is selected to train each HMM.

First, the HMM $\lambda = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$ is initialized. The training data is uniformly segmented from top to bottom in N=5 states and the observation vectors associated with each state are used to obtain initial estimates of the observation probability matrix **B**. The initial values for **A** and $\mathbf{\Pi}$ are set given the left to right structure of the face model.

In the next steps the model parameters are re-estimated using the EM procedure to maximize $\mathrm{P}(\mathbf{O} \mid \lambda)$. The iterations stop, after model convergence is achieved, i.e. the difference between model probabilities at consecutive iterations (k and k+1) is smaller than a threshold C.

One thing should mention is that the author using 2D-DCT coefficients as observation vector instead of the pixel values that were used as observation vector by pervious algorithms. The advantages of DCT coefficient are (1) it will be not sensitive to the noise in the image and (2) it will reduce greatly the dimension of observation vectors. In this case, the author used 13x3 frequencies elements as observation vector elements, so that the dimension of the vector is 23 times smaller than before, which reduce much on computation time.

#### 3.4.3.1 Comments

After experiment, this HMM face recognition method got much higher performance than eigenface method. By testing on the same database, HMM achieved a recognition rate of 84% that is 10 percent higher than eigenface method. Moreover, the processing time required to compute the likelihood of one test image given a face model is decreased from 25 seconds (a previous HMM method), to 2.5 seconds in the present work.

This HMM method is very simple to understand, because it only used 1 dimensional HMM structure. In order to get much higher accuracy rate in recognition, the author invented another 2 dimensional HMM structure that resulted in the recognition rate to 98%.

### 3.4.4 Face Recognition Using An Embedded HMM

**[Nef99]**

Following the last paper, the author raised another topology for HMM, an embedded HMM, which change 1 dimensional HMM to 2 dimensional HMM.

The author think like this way. A one-dimensional HMM maybe generalized, to give it the appearance of a two-dimensional structure, by allowing each state in a one-dimensional HMM to be an HMM. In this way, the HMM consists of a set of super states, along with a set of embedded states. The super states may then be used to model two-dimensional data along one direction, with the embedded HMM modeling the data along the other direction. This model differs from a true two-dimensional HMM since transitions between the states in different super states are not allowed. Therefore, this is referred to as an embedded HMM. The elements of an embedded HMM were given in the original paper and I don't show it again. The structure of embedded HMM can be seen from the figure.

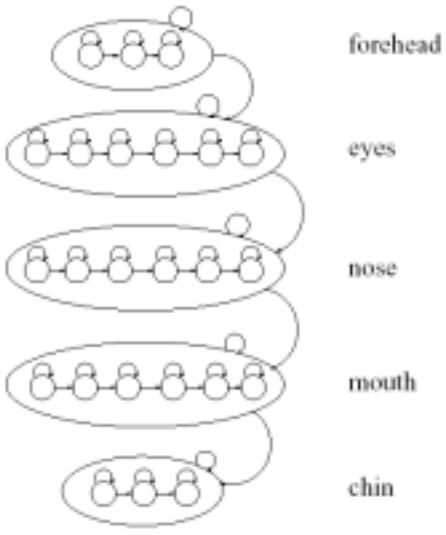



Figure 3.10: Embedded HMM

#### 3.4.4.1 Comments

After testing, the use of an embedded HMM model for the human face is justified by the structure of the face, and is invariant for a large range of orientations, gestures, and face appearances. The use of an embedded HMM increases by over 10% the recognition rate of the one-dimensional HMM and the classical eigenfaces method. The accuracy of the system presented in this paper is increased to 98%. That is why HMM is accepted as the most effective model for face recognition.

## 3.5 Classification Models Comparison

There are many models for classification, but the popular models are what we mentioned in this chapter, KNN, SVM, HMM and the new method, RVM. Each of them has their advantages and corresponding problems to solve, and also has their disadvantages. For example, KNN is a method on basis of simple theory and can be implemented easily. For simple classification and small size of data set, it can reach a high classification rates with a high speed. But to deal with more complicated problems, most of people seek help for SVM and RVM, especially for some object or shape recognition problems. Between them RVM gives more advantages like probability output, less and unconditional kernels and etc, and SVM gives much faster training speed. HMM is a little bit far from other models, it describe the states transition status, but it has been successfully used in speech recognition, face recognition, and gesture recognition, and gives a very high recognition rates when dealing with these tasks.

To give a brief overview for all the theories in this chapter, we summarize a table below that shows their features, so that you can get a rough comparison between them. For detailed comparison of some models like SVM and RVM, you can refer the "relevance vector machine" section, where we gave more accurate tests result.

Lets do some definition to the features of different models in the following table

- KN: Kernel Model. Which model it is.
- MN: Mature or New. If the model is mature, well wide used technique or it is a new one.
- SP: Suitable Problems. To describe which kinds of problems are suitable for this model to solve.
- FL: Flexibility. If this model could be used in other classification problems.
- CO: Complexity. If this model is easy to implement.
- LS: Learning Speed. Need training procedure or not and how fast?
- LSI: Learning Speed Improved. The training speed of improved algorithm
- CS: Classification Speed. How fast to classify one sample?
- CSI: Classification Speed Improved. How fast to classify one sample after Improvement?
- CR: Classification Rates in its problem. What's the performance of this model used to solve its suitable problems.
- PO: Probability Output. If the original model performs based on a probability framework and will give a probability output.
- PI: Paper's Index. Which papers in this chapter are talking about this model?

Table 3.6 Classification models comparison (Assume: 2 classes cases)

| KN | KNN | SVM | RVM | HMM |
|---|---|---|---|---|
| MA | Mature | Mature | New | Mature |
| SP | General Classification | Objects Recognition | Objects Recognition | Face/Gesture/Speech Recognition |
| FL | Flexible | Flexible | Flexible | Flexible |
| CO | Easy | Not Easy | Not Easy | Not Easy |
| LS | No | Yes/Fast | Yes/Slow | Yes/Slow |
| CS | Slow | Very Fast | Very Fast (1ms)[3] | Fast (800 ms) |
| LSI | Slow | Very Fast | Fast | Fast |
| CSI | 1 ms | Very Fast | Very Fast (1ms) | Fast (800 ms) |
| CR | High | Very High | Very High | Very High |
| PO | No | No | Yes | Yes |
| PI | [WEB02] [ZHA04] | [VAP95] [COR95] [PON98] [HEI03] [ZHA04] | [Tip01] [TipNEW] | [Lus95] [Nef98] [Nef99] |

[3] The speed indicated in the brackets is a rough value and comes from the other papers or our test.

# Chapter 4

# 4 Objects Tracking and Gesture Recognition

If UI-Wand can indicate the accurate position on the screen, then the user should expect it could control something instead of just pointing somewhere. For this purpose, we can easily put many buttons on UI-Wand, but that will be not convenient for users. The natural and the simple way to let user control while he or she is pointing somewhere, is to use some gestures. Pointing and hold is the simplest gesture, like a computer mouse, we can map a "click" function to this gesture. That is say to invoke some function, if we realize the user is pointing somewhere and that point does not move in last several frames.

This is a gesture which can be used communicate with the target application. But for more function invocation of that application, one simple gesture will be not enough. We expect to use more complicated gestures to control it. For example, we can make a cross to shut down the application, move it right and back to realize "page up", or rotate it to turn up or down the volume. Then the tasks are not easy any more. So in this chapter, we investigate many practical papers to explain how did they solve gesture recognition in their problems, then we can find the interesting points for our project.

## 4.1 Object tracking methods

Before the gesture recognition, the basic information we need to have is the track information. Given a specific point or pattern, we need to know the places it passed in a sequence of frames. So how to detect that point or pattern and how to track it in the following frames are the important techniques in this section. Below are two papers, which used SVM or RVM to detect the objects need to track, and even track by using these classification methods, which are very interesting and give us many hints.

### 4.1.1 Support Vector Tracking

**[Avi01]**

In this paper, the author introduces a method named Support Vector Tracking, which integrates the Support Vector Machine classifier into an optic-flow based tracker. Instead of minimizing an intensity difference function between successive frames, SVT maximizes the SVM classification score. To account for large motions between successive frames he build pyramids from the support vectors and use a coarse-to fine approach in the classification stage. The paper introduces the SVM as his classifier that we already investigated in the previous section. Here, we just focus on his tracking approach.

Tracking algorithms find how does an image region move from one frame to the next. This implies the existence of an error function to be minimized, such as the sum of squared differences (SSD), between the two image regions. This paradigm makes no assumption about the nature of the tracked object. For the interest that people want to tracking a particular class of objects, the general method is to use the tracker and the

classifier sequentially. The tracker will find where did the object move to and the classifier will give it a score. But the problem of it is that the tracker is not guaranteed to move to the best location. Shuai's solution is to try to maximize the classification score given by classifier, SVM in this case, take the replace of minimizing the SSD error function.

The SVT principles are following alike:

Kernel-SVM is: $\sum_{j=1}^{l} y_j \alpha_j k(\mathbf{I}, \mathbf{x}_j) + b$ where $x_j$ are the $l$ support vectors, $y_j$ are their sign and $\alpha_j$ are their distances from the hyperplane. $k(\mathbf{I}, \mathbf{x}_j)$, I is the kernel they choose, in his case, he chose homogeneous quadratic polynomial given by $k(\mathbf{x}, \mathbf{x}_j) = (\mathbf{x}^T \mathbf{x}_j)^2$, and I is the image region to test.

If let $\mathbf{I}_{\text{init}}$ represent some initial guess of the position of the object in a given image, $\mathbf{I}_{\text{final}}$ can be expressed by using first-order Taylor expansion:

$$\mathbf{I}_{\text{final}} = \mathbf{I}_{\mathbf{init}} + u\mathbf{I}_{\mathbf{x}} + v\mathbf{I}_{\mathbf{y}}, \tag{4.1}$$

where $\mathbf{I}_{\mathbf{x}}, \mathbf{I}_{\mathbf{y}}$ are the x and y derivatives of image $\mathbf{I}_{\text{init}}$ and u, v are the motion parameters. Plugging Eq. (4.1) in kernel-SVM and taking the u and v derivatives and rearranging terms then we will get a equation, which resembles the standard optic-flow equations with the support vectors replacing the role of the second image in the equation. This means that all computations are done on a single frame each time and not on a pair of successive frames. After a number of iterations, using the derived equation, the maximum SVM score can be reached. But the problem is, this approach can handle small motions in the image. Larger motions must be handled in a coarse-to-fine manner, which the author call it Pyramid SVT.

Shai created two pyramids in his algorithm, one is created on support vectors, which actually are sub-sampled images of learning set, and another one is created for the test image with the same size as support vectors. In the tracking algorithm, he first runs SVT on the top level of the support vector pyramid and the top level of the test image pyramid. The recovered motion parameters are at the position with the best SVM score. This position serves as the initial guess for the SVT on the next level of the pyramid and so on until the motion parameters are recovered. The SVM score is the score of the bottom level of the pyramid.

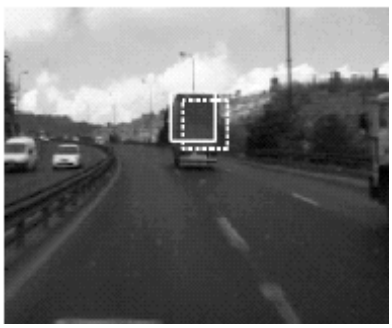


Figure 4.1 The solid rectangle denotes the SVT tracking result, the dashed rectangle denotes a simple SSD tracker.

In the experiments phase, he took 10000 images of vehicles and non-vehicles. Vehicles include cars, SUVs and trucks in different colours and sizes. The pyramids he took are 2 level Gaussian pyramids. By his experiments, SVT give a better result (See Fig.4.1) comparing to SSD tracker. The detailed experiments data, you can refer his paper.

#### 4.1.1.1 Comments

Avidan's paper is very interesting and gives us many hints. For example, instead of finding the motion vector from intensities values of image, he calculated the SVM score in an area to indicate the place with the highest score as the target place. And the pyramids method he used also gives us a hint for the acceleration of track speed.

As the test result show, Avidan's method gives a very good method both from speed and from the positioning accuracy, which can help us to realize screen tracking in UI-Wand, but for screen tracking, there are some unavoidable problems, such as the changeable content of the screen, the rotation of the image, and etc. If we want to use similar algorithm, we have to find good solutions for these issues.

### 4.1.2 A Sparse Probabilistic Learning Algorithm for Real-Time Tracking [Wil03]

This paper focus on the tracking issues, the authors raise a new tracker algorithm using sparse probabilistic regression by RVMs. Comparing to SVT [Avi01], instead of constructing a recogniser based on SVM, this paper's algorithm invents a localizer based on RVM regression form that give a fusion between observation (obtained from RVM) and Gaussian distributions. This algorithm shows probabilistic inference has greatly increased both the stability and the robustness of tracking and the speed is rapid for real-time requirement. (During continuous tracking only around 15ms/frame of CPU time is required.) Below is brief indication to their algorithm.

As their understanding, the SVT is a procedure that is: Given an object q (sub-image), perturbation analysis allows the translation $T_{\mathbf{u}}$, by a vector u, to be found such that the value $f(T_{\mathbf{u}}\mathbf{q})$ of the classification function is maximized. The perturbed classification function is expressed in terms of image gradient as:

$$f(T_{\mathbf{u}}\mathbf{q}) = f(\mathbf{q} + \mathbf{u} \cdot \nabla\mathbf{q}) \tag{4.2}$$

Using this expression to compute approximately the displacement u that maximized the classification function $f$ can save computation by reducing the density of tessellation required to achieve a given degree of tracking precision. So in SVT algorithm, SVM is looked as a recogniser function $f$ that classify whether this is an object or not and give a score as confidence.

But in their approach, instead of training a machine to recognize known objects verses non-objects, they trained a machine to estimate the displacement of known objects alone, which is a kind of regression estimation. They think this is of primary interest for tracking motion. So they construct a model for motion classification, which they use several SVMs to implement, but the result is not as good as they expected. But after they change them to RVMs, together with Kalman filter, the experiments results are got greatly improvement. The motion classification procedure is summarized as following:

The tracker follows a 2D image region containing an object. Intensity changes within this region are assumed to be due to motion only. Therefore, a four-dimensional state vector is used describing a Euclidean similarity transformation:

$$\mathbf{X} = [u, v, s, \theta] \in GE(2) \tag{4.3}$$

$\mathbf{I}_t$, is a vectorized pixel array at time t, which is some unknown function of the present state: $\mathbf{I}_t = H(\mathbf{X}_t)$. Inverse the function, we got $\mathbf{X}_t = H^{-1}(\mathbf{I}_t)$. It should be possible to find a regression inferring a state estimate, $\tilde{\mathbf{X}}$.

They constructed a vector of RVMs required in order to obtain a vector estimate of state dimension value as output. Each element has one of the four state space dimensions as its dependent variable. They all use the same training set. This means that the machine inferring x-translation, say, has been trained on images perturbed in the other three dimensions too and is insensitive to these. The regression function values of the 4 RVMs will be the displacement value in corresponding state dimension. These values are passed to the expression below to get estimated change in state, $\delta\mathbf{X}$:

$$\delta\mathbf{X} = \mathbf{Af(I)} + \mathbf{b} \tag{4.4}$$

$\mathbf{b} \in \Re^4$ $\mathbf{A} \in \Re^{4x4}$, A and B can be learned by conventional regression.

To secure benefits of temporal fusion of data, observations must be obtained in a probabilistic setting. So the authors use the probabilistic result and together Kalman filter construct a probabilistic state inference. And the experiments result (See Fig.4.2) shows this combination increase the performance greatly.

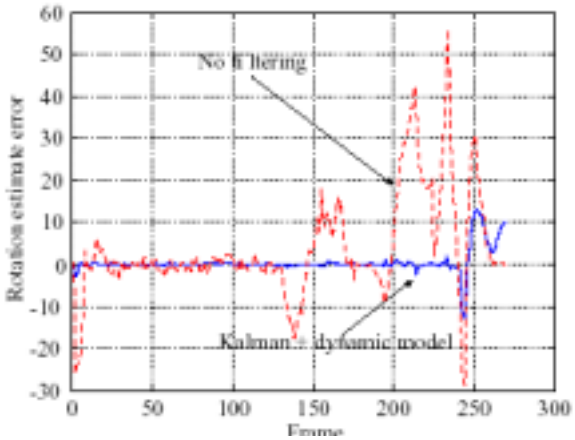



Figure 4.2 the performance after using Kalman filter

For the initialisation and recovery, this paper did not give some new method. They just used SVM as object as recogniser.

Figure 4.3 The result of this Tracking approach to track a car and its license plate

#### 4.1.2.1 Comments

It is a good mechanism to realize tracking (See Fig.4.3), instead of SVM like [Avi01], they use RVM regression form to estimate current state change, which give a probabilistic value to displacement and fused temporally with motion prediction by Kalman filtering. This probabilistic inference has greatly increased both the stability and the robustness of tracking. From their experiment, this algorithm speed reached the requirement for real-time application. But it seems they did not consider too much on initialization and recovery method, which will spend 1 second, which might be a lack for their tracking approach.

### 4.1.3 Kalman Algorithm

**[Kal60][Wel01]**

Within the significant toolbox of mathematical tools that can be used for stochastic estimation from noisy sensor measurements, one of the most well-known and often-used tools is what is known as the Kalman filter. The Kalman filter is named after Rudolph E. Kalman, who in 1960 published his famous paper [Kal60] describing a recursive solution to the discrete-data linear filtering problem. Later on a lot of different application using Kalman filter appeared. And there is a very detailed explanation about the Kalman algorithm in [Wel01]. Kalman filter is a very popular model that can be used to track target and do some parameter estimation. Kalman filter also has very convenient form for online real time processing, so it can be used as a tracking model in UI-Wand.

#### 4.1.3.1 The Discrete Kalman Filter

This section describes the filter in its original formulation (Kalman 1960) where the measurements occur and the state is estimated at discrete points in time.

#### 4.1.3.2 The Process to be estimated

The Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}. \tag{4.5}$$

with a measurement $z \in \Re^m$ that is

$$z_k = Hx_k + v_k \ . \tag{4.6}$$

The random variables $w_k$ and $v_k$ represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(w) \sim N(0,Q), \tag{4.7}$$
$$p(v) \sim N(0,R). \tag{4.8}$$

In practice, the *process noise covariance* and *measurement noise covariance* matrices might change with each time step or measurement, however here we assume they are constant.
The $n \times n$ matrix *A* in the difference equation Eq. (4.5) relates the state at the previous time step $k-1$ to the state at the current step, in the absence of either a driving function or process noise. The $n \times l$ matrix *B* relates the optional control input $u \in \Re^l$ to the state *x*. The $m \times n$ matrix *H* in the measurement equation Eq. (4.6) relates the state to the measurement $z_k$. In practice *H* might change with each time step or measurement, but here we assume it is constant.

#### 4.1.3.3 The Computational Origins of the Filter

We define $\hat{x}_k^- \in \Re^n$ to be our *a priori* state estimate at step *k* given knowledge of the process prior to step *k*, and $\hat{x}_k \in \Re^n$ to be our *a posteriori* state estimate at step *k* given measurement $z_k$. We can define *a priori* and *a posteriori* estimate errors as $e_k^- \equiv x_k - \hat{x}_k^-$ and $e_k \equiv x_k - \hat{x}_k$.
The *a priori* estimate error covariance is then

$$P_k^- = E\left[e_k^- e_k^{-T}\right], \tag{4.9}$$

and the *a posteriori* estimate error covariance is

$$P_k = E\left[e_k e_k^T\right]. \tag{4.10}$$

In deriving the equations for the Kalman filter, we begin with the goal of finding an equation that computes an *a posteriori* state estimate $\hat{x}_k$ as a linear combination of an *a priori* estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement $z_k$ and a measurement prediction as shown below in Eq. (4.11). Some justification for Eq. (4.11) is given in "The Probabilistic Origins of the Filter" found below.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{4.11}$$

The difference $(z_k - H\hat{x}_k^-)$ in Eq. (4.11) is called the measurement *innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement $H\hat{x}_k^-$ and the actual measurement $z_k$. A residual of zero means that the two are in complete agreement. The matrix *K* in Eq. (4.11) is chosen to be the *gain* or *blending factor* that minimizes the *a posteriori* error covariance Eq.(4.11). This minimization can be accomplished by first substituting Eq. (4.11) into the above definition for $e_k$, substituting that into Eq. (4.10), performing the indicated expectations, taking the derivative of the trace of the result with respect to *K*, setting that result equal to zero, and then solving for *K*. One form of the resulting *K* that minimizes Eq. (4.10) is given by

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{HP_k^- H^T + R}. \tag{4.12}$$

Looking at Eq.(4.12) we see that as the measurement error covariance approaches zero, the gain *K* weights the residual more heavily, specifically, $\lim_{R_k \to 0} K_k = H^{-1}$. On the other hand, as the *a priori* estimate error covariance $P_k^-$ approaches zero, the gain

$K$ weights the residual less heavily, specifically, $\lim_{P_k^- \to 0} K_k = 0$.

Another way of thinking about the weighting by $K$ is that as the measurement error covariance $R$ approaches zero, the actual measurement $z_k$ is "trusted" more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. On the other hand, as the *a priori* estimate error covariance $P_k^-$ approaches zero the actual measurement $z_k$ is trusted less and less, while the predicted measurement is trusted more and more.

#### 4.1.3.4 The Probabilistic Origins of the Filter

The justification for Eq. (4.11) is rooted in the probability of the *a priori* estimate $\hat{x}_k^-$ conditioned on all prior measurements $z_k$ (Bayes'rule). For now let it suffice to point out that the Kalman filter maintains the first two moments of the state distribution $E[x_k] = \hat{x}_k$ and $E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k$.

The *a posteriori* state estimate Eq. (4.11) reflects the mean (the first moment) of the state distribution— it is normally distributed if the conditions of Eq.(4.7) and Eq.(4.8) are met. The *a posteriori* estimate error covariance Eq.(4.10) reflects the variance of the state distribution (the second non-central moment). In other words

$$p(x_k | z_k) \sim N(E[x_k], E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]) = N(\hat{x}_k, P_k).$$

#### 4.1.3.5 The Discrete Kalman Filter Algorithm

We will begin this section with a broad overview, covering the "high-level" operation of one form of the discrete Kalman filter (see the previous footnote). After presenting this high-level view, we will narrow the focus to the specific equations and their use in this version of the filter.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate. The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems as shown below in Fig 4.4.

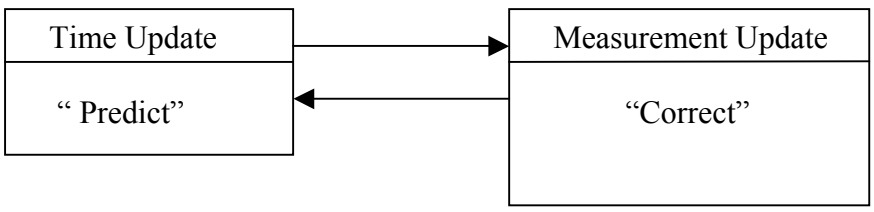



Figure 4.4 The ongoing discrete Kalman filter cycle. The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time.

The specific equations for the time updates are presented below in Eq. (4.13) and Eq. (4.14).

$$x_k^- = A\hat{x}_{k-1} + Bu_k \tag{4.13}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4.14}$$

The specific equations for the measurement updates are presented below in Eq. (4.15) Eq. (4.16) and Eq. (4.17).

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{4.15}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4.16}$$

$$P_k = (I - K_k H)P_k^- \tag{4.17}$$

Again notice how the time update equations project the state and covariance estimates forward from time step to step, and *B* are from Eq. (4.5), while is from Eq. (4.7). Initial conditions for the filter are discussed in the earlier references. The first task during the measurement update is to compute the Kalman gain, $K_k$. Notice that the equation given here as Eq. (4.15) is the same as Eq. (4.12). The next step is to actually measure the process to obtain $z_k$, and then to generate an *a posteriori* state estimate by incorporating the measurement as in Eq.(4.16). Again Eq.(4.16) is simply Eq.(4.11) repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via Eq.(4.17). After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very appealing features of the Kalman filter. The Kalman filter instead recursively conditions the current estimate on all of the past measurements. Fig. 4.5 below offers a complete picture of the operation of the filter, combining the high-level diagram of Fig. 4.4 with the above equations.

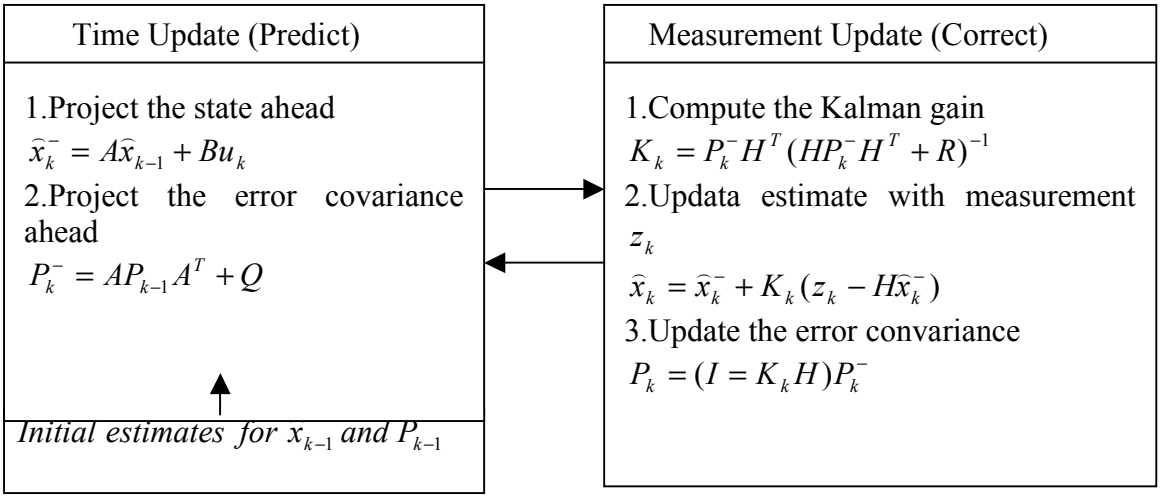



Figure 4.5 A complete picture of the operation of the Kalman .lter, combining the high-level diagram of Figure 4.4 with the above equations

#### 4.1.3.6 Comments

Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by a *linear* stochastic difference equation. It also can be used to solve the nonlinear problem using the "Extended Kalman Filter"(EKF). The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is *optimal* in the sense that it minimizes the estimated *error* covariance—when some presumed conditions are met. Since the time of its introduction, the *Kalman filter* has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. This is likely due in large part to advances in digital computing that made the use of the filter practical, but also to the relative simplicity and robust nature of the filter itself. Rarely do the conditions necessary for optimality actually exist, and yet the filter apparently works well for many applications in spite of this situation. It has been used extensively for tracking in interactive computer graphics.

## 4.2 Gesture Recognition

### 4.2.1 Recognition of human gestures and behavior

**[Psa02]**

In this paper, the authors purpose an improved recognition method for gestures and human behavior. They did two kinds of experiments to test their approach. One is for human movements in an office and the other is for human gestures made by a people sitting in front of single color background. The result of these tests shows that the accuracy of the algorithm is improved better than some method before.

The authors firstly summarize several gestures recognition method and mentioned the disadvantages in these methods:

- DTW (Dynamic Time Warping): It is successful in small tasks, but its main limitations are that it needs a large number of templates in order to model a range of variations and it cannot handle undefined patterns.
- HMM (Hidden Markov Models): The main disadvantages are that they can be used to estimate the probabilities for only one model at a time and they can only give an estimate of the final probability for each model.
- ANN (Artificial Neural Networks): Its problem is the network inability to converge when trained with high-dimensional structures.
- Condensation (Conditional Density Propagation): Its defect is that the model does not use any prior knowledge on both state transitions and measurement covariance.

Then they introduce a framework to recognize gestures and behaviors based on both learning prior and continuous propagation of density models of behavior patterns. Prior models are learned from training sequences using HMM and density models are augmented by current visual observation.

They described the spatial-temporal trajectory exhibited by a gesture or behavior as a first-order Markov process under which salient phases or states of the movement are explicitly modeled over time. By their first-order Markov processes, the final issues is to calculate the state probability $p(\mathbf{q}_t \mid \mathrm{O}_t)$ based on Bayes' rule as follows:

$$p(\mathbf{q}_t \mid O_t) = k_t p(\mathbf{o}_t \mid \mathbf{q}_t) p(\mathbf{q}_t \mid \mathrm{O}_{t-1}) \tag{4.18}$$

where $p(\mathbf{q}_t \mid \mathrm{O}_{t-1})$ is the prior from the accumulated observation history up to time t-1, $p(\mathbf{o}_t \mid \mathbf{q}_t)$ is the conditional observation density and $k_t$ is a normalization factor. In order to get priors on temporal structures, they use HMM to learn it from training examples. By training HMMs on a set of observed trajectories of activities, a priori knowledge on both state propagation and conditional observation density can be learned by assigning the hidden Markov state transition probabilities $p(\mathbf{q}_t = j \mid \mathbf{q}_t = i)$ of a trained model $\lambda = (\mathbf{A}, \mathbf{b}, \pi)$ to the condensation state propagation densities of:

$$p(\mathbf{q}_t \mid \mathbf{q}_{t-1}) = p(\mathbf{q}_t = j \mid \mathbf{q}_t = i, \lambda) = a_{ij} \tag{4.19}$$

And the prior on the observation conditional density $p(\mathbf{o}_t \mid \mathbf{q}_t)$ is given by the Markov observation densities at each hidden state as:

$$p(\mathbf{o}_t \mid \mathbf{q}_t) = p(\mathbf{o}_t \mid \mathbf{q}_t = j, \lambda) = b_j(\mathbf{o}_t) \tag{4.20}$$

The Markov observation density at each Markov state $b_j(\mathbf{o}_t)$ is used to provide the prior knowledge about the observation covariance. As a result, the process of both sampling and propagating condensation states is made not only more focused but also robust against observation noise.

Another technique they used in their approach is to take current observation into account before predication, which made recognition based on prior more robust. In this step, the kernel issue is to calculate the state transition density, $p(\mathbf{q}_t \mid \mathbf{q}_{t-1}, \mathbf{o}_t)$ instead of $p(\mathbf{q}_t \mid \mathbf{q}_{t-1})$, which both improves the recognition rate and reduces the number of samples used for propagation.

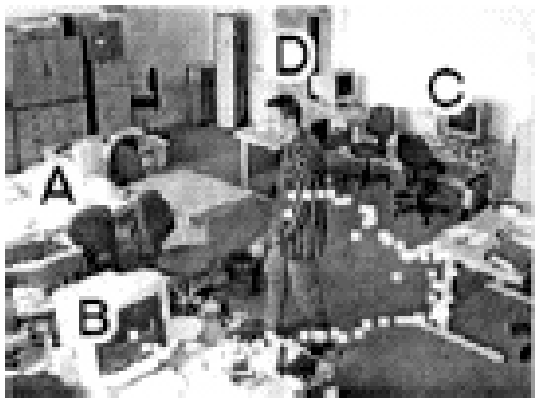



Figure 4.6 The office environment, and movement track

#### 4.2.1.1 Comments

In this paper, the authors did not mention the speed of their system. But comparing the approach with condensation algorithm, the accuracy of recognition is improved much. The recognition rate of the walking behavior is increased by 70% and the recognition rate of the communicative and symbol gestures is increased by 40% and 25%, respectively. Moreover, they also showed that their algorithm only needs a small number of samples.

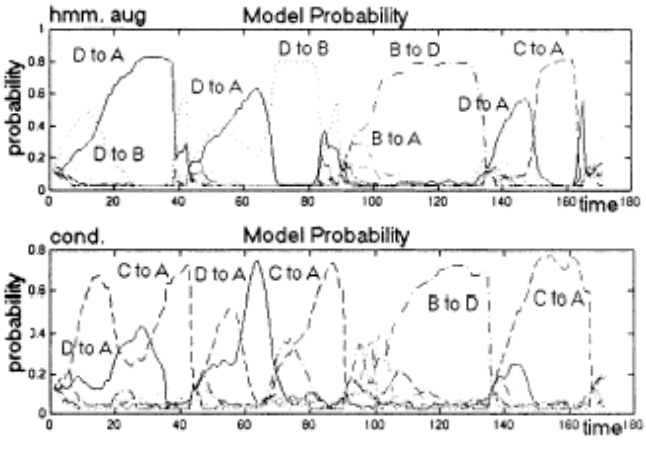



Figure 4.7 Behavior likelihood estimated over time. Their approach (top), condensation (bottom)

### 4.2.2 Recognizing temporal trajectories using the condensation algorithm

**[Bla98]**

This paper describes an incremental recognition strategy that is an extension of the "Condensation" algorithm proposed by Isard and Blake (ECCV'96). Gestures are modeled as temporal trajectories of some estimated parameter over time (in this case velocity). The condensation algorithm is used to incrementally match the gesture models to the input data. Their method was demonstrated with an example of an

augmented office whiteboard in which a user made simple hand gestures to grab regions of the board, print them, save them, etc.

#### 4.2.2.1 The model of their approach

Their goal is to take a set of M model trajectories $\{\mathbf{m}^{(\mu)}, \mu = 1,\ldots,M\}$ and match them against an input trajectory (see Fig.4.8). The models are taken to be discretely sampled curves with a phase parameter $\phi \in [0, \phi_{\max}]$ representing the current position in the model. The model values at position $\phi$ are a vector of N values $\mathbf{m}_{\phi}^{(\mu)} = (\mathrm{m}_{\phi,1}^{(\mu)},\ldots,\mathrm{m}_{\phi,N}^{(\mu)})$ where the stored discrete curve is linearly interpolated at phase $\phi$. At time $t$ the input trajectory is an observation vector $\mathbf{z}_t = (z_t,\ldots,z_{t,N})$.

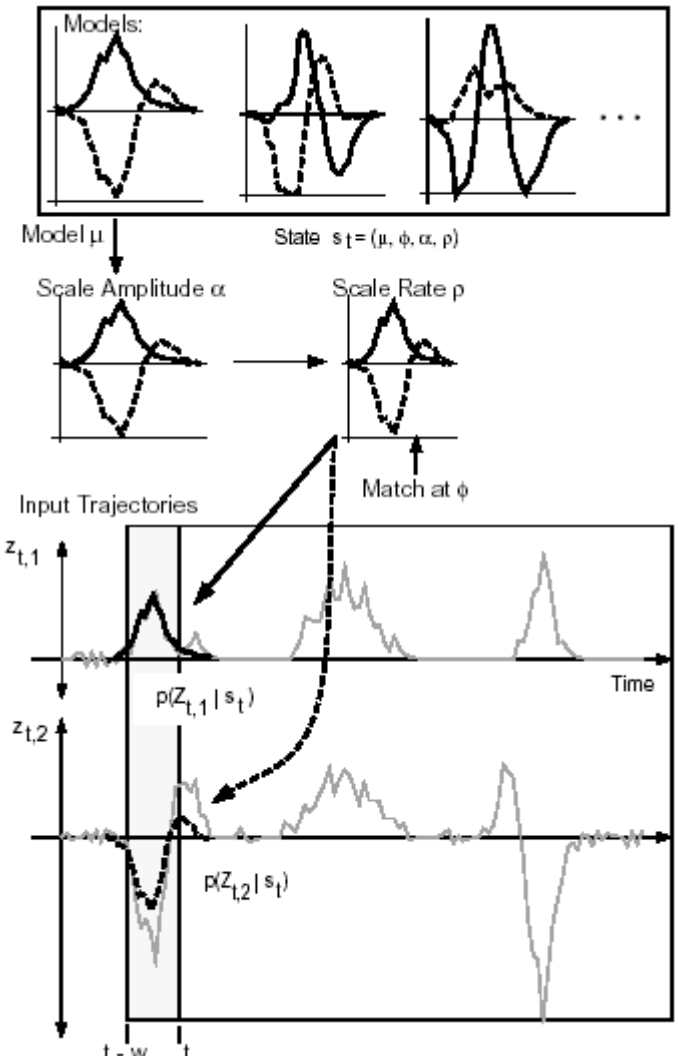



Figure 4.8 Their approach idea is to match input trajectory with the models

The parameters they need to estimate to match a model to the data are:

- $\mu$, the model number.
- $\phi$, the position within model that aligns the model at the time t.
- $\alpha$, the parameter to scale the model vertically.
- $\rho$, the parameter to scale the model horizontally.

Then they defined a state at time t to be a vector of parameters $\mathbf{s}_t = (\mu, \phi, \alpha, \rho)$. So their goal is to find the state $\mathbf{s}_t$ that is most likely to have given rise to the observed data $Z_t = (\mathbf{z}_t, \mathbf{z}_{t-1}, \ldots)$, and then they defined, $p(\mathbf{z}_t \mid \mathbf{s}_t) = \prod_{i=1}^{N} p(Z_{t,i} \mid \mathbf{s}_t)$, where $Z_{t,i} = (z_{t,i}, z_{t-1,i}, z_{t-2,i}, \ldots)$ that is the vector of observations of a particular trajectory, *i*, over time. By this definition, they can construct a discrete representation of the entire probability distribution over the possible states. And they computed the probability of the state $p(\mathbf{z}_t \mid \mathbf{s}_t)$. They did this for *S* samples and gives them a set $\{\mathbf{s}_t^{(n)}, n = 1,\ldots,S\}$ of samples, then they normalized these probabilities so that they sum to one, producing weights $\pi_t^{(n)}$:

$$\pi_t^{(n)} = \frac{p(\mathbf{z}_t \mid \mathbf{s}_t^{(n)})}{\sum_{i=1}^{S} p(\mathbf{z}_t \mid \mathbf{s}_t^{(i)})} \tag{4.21}$$

The condensation algorithm uses the sample states and their weights to predict the entire probability distribution at the next time instant. Detailed procedure is in [Jep98].

#### 4.2.2.2 Experiment

In their experiment, they construct several gesture models attached with nine primitive events). To construct models for the gestures, each gesture was performed approximately half a dozen times and the trajectories were saved. For a given gesture the training trajectories were manually aligned and the mean trajectories were computed. A standard deviation from the mean trajectory was also computed for each curve. The trajectory models for each gesture are shown in Figure.

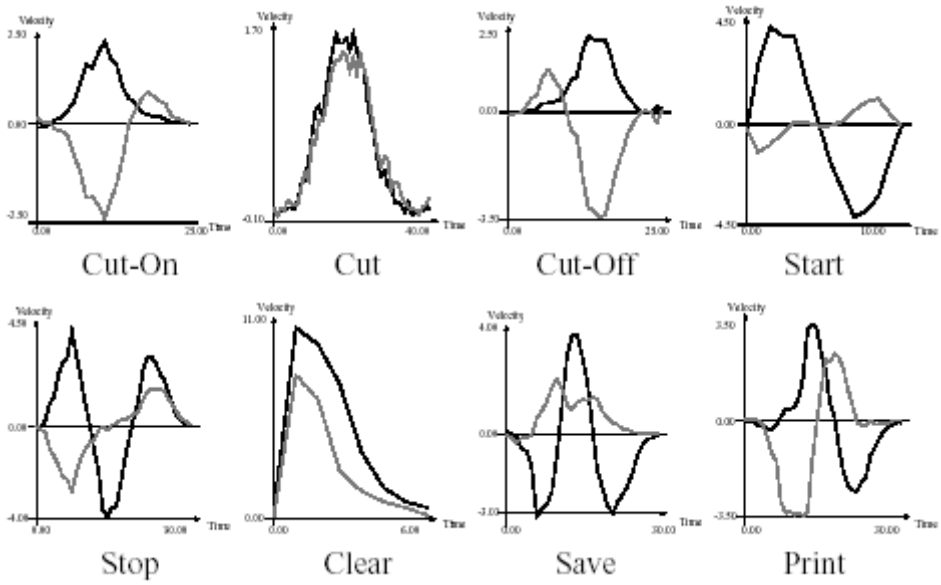



Figure 4.9 Gesture models. Temporal trajectories of x-velocity and y-velocity

#### 4.2.2.3 Comments

The approach purposed in this paper, is an extension to the Condensation algorithm that performs probabilistic matching of model curves to input curves. This method allows the recognition of complex gestures than is possible with the standard Condensation algorithm. But the disadvantages are obvious in the approach. The speed is significantly slower than real time, which need to do some change in the algorithm and the transition probabilities were set by hand, which should be learned automatically.

### 4.2.3 An HMM-Based Threshold Model for Gesture Recognition

**[Lee99]**

M.Isard and A.Blake in [Isa96] and [Isa98] used the extension of "Condensation algorithm" to deal with gesture recognition. Later on M.H.Yang and N.Ahuja [Yan99] used a method that uses multi-scale motion segmentation to monitor motion over time and a Time Delayed Neural Network to match this motion to recognize gestures. But the most popular way to realize the gesture recognition is using Hidden Makov Model(HMM), a lot of HMM-based model have already made by different people. In1999, Hyeon-Kyu Lee and Jin H.Kim presented their HMM-based model [Lee99] for gesture recognition. They chose the HMM-based approach because it can be applied to analyzing time-series with spatio-temporal visibilities and can handle undefined patterns. It also has elegant and efficient algorithms for learning and recognition, such as the Baum-Welch algorithm and Viterbi search algorithm.

For correct gesture spotting, the likelihood of a gesture model for a given pattern should be distinct enough. But they found a simple threshold for the likelihood often does not work. Therefore, they propose a new concept, called threshold model, which yields the likelihood value to be used as a threshold. A gesture is recognized only if the likelihood of the best gesture model is higher than that of the threshold model. The HMM's internal segmentation property implies that each state with its self-transition represents a segmental pattern of a target gesture and that outgoing transitions represent a sequential progression of the segments in a gesture. With this property, they construct an ergodic model with the states copied from all gesture models in the system and then fully connect the states (Fig. 4.10).

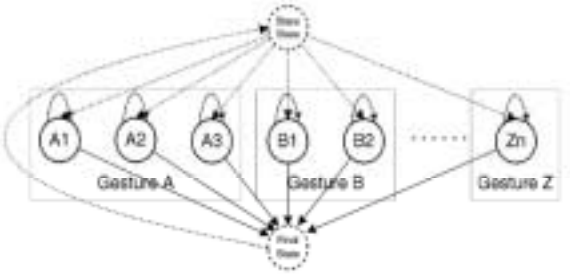



Figure 4.10 Ergodic HMM model

In this model, each state can be reached by all other states in a single transition. Output observation probabilities and self-transition probabilities in this model are kept as in the gesture models, but all outgoing transition probabilities are equally assigned as $a_{ij} = \frac{1-a_{ij}}{N-1}(for\ all\ j, i \neq j)$, where $a_{ij}$ is the transition probability from state $s_i$ to $s_j$ and N is the number of states (the sum of all states excluding the start and final states). The start and final states produce no observation. The likelihood of the model, given a gesture pattern, would be smaller than that of the dedicated gesture model because of the reduced forward transition probabilities. The likelihood can be used as an adaptive threshold for selecting the proper gesture model. For this reason, they call it the threshold model. The threshold model acts as a base-line. A candidate gesture is found when a

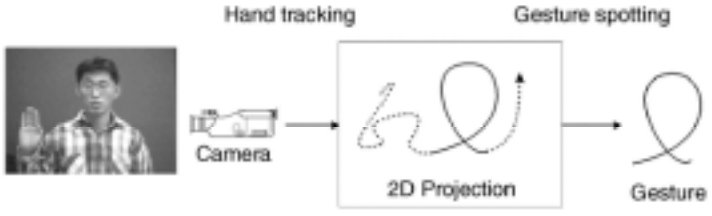



Figure 4.11 Gesture spotting precedure

specific gesture model rises above the threshold model.

A gesture spotter receives a stream of continuous hand motion with an intermittent sequence of several gestures (see Fig. 4.11). To spot them in the input stream, they constructed a circular gesture-spotting network (GSN). The gesture-spotting network finds the start and end points of gestures embedded in the input stream. For this, it is desirable to know how and in what state sequence the model produces the most likely observation sequence. They can uncover a state sequence using the Viterbi algorithm, where they adopt the optimality criterion of maximizing the probability of a state sequence that produces the observation sequence. They use the Viterbi algorithm for calculating likelihood and finding the start point from the end point.

For reliable spotting, the model transition probability into the threshold model $p(TM)$ is tuned to satisfy:

$$\begin{aligned} P(X_G|\lambda_G)p(G) &> P(X_G|\lambda_{TM})p(TM) \\ P(X_{TM}|\lambda_G)p(G) &< P(X_{TM}|\lambda_{TM})p(TM) \end{aligned} \tag{4.22}$$

where $X_G$ denotes a gesture pattern, $X_{TM}$ a non gesture pattern, $\lambda_G$ the target gesture model, $\lambda_{TM}$ the threshold model, and $N_G$ the number of gesture models in the system.
Inequalities (4.22) imply that a gesture should best match with the corresponding gesture model and a non-gesture with the threshold model, respectively. The model transition probabilities into gesture model $p(G)$ are set equal using $p(TM)$ as $p(G) = \frac{1 - p(TM)}{N_G}$. With a sequence of spotting experiments, they have decided $p(TM)$ as the value generating the best result. First inequality of Eq. (4.22) says that the likelihood of a gesture model should be greater than that of the threshold model. The time satisfying such a condition can be called a Candidate End Point (CEP). Once they obtain CEP, its corresponding start point can easily be found by backtracking the Viterbi path because the final state can only be reached through the start state in the left-right HMM. There are, in general, several such CEPs satisfying First inequality of Eq. (4.22). Thus, the remaining problem is the determination of the right end point. This will be described in the next section.

The end-point detection is the process of choosing the best among these CEPs. It removes nested gestures and makes the spotter fire only once when a gesture is encountered. The process is initiated when the last CEP of the current gesture is found after the preceding gesture to fire or when the elapsed time steps since the last gesture are greater than a given length. The last condition implements a duration constraint. The detection criterions defined as follows:

1. When the immediately following pattern B is not a gesture, as in Fig. 4.12(a), the last CEP of the preceding gesture A is determined as the end point. A is reported.

2. When the immediately following pattern B is by itself a gesture, there are two alternatives:

a. When the start point of B precedes the first CEP, as in Fig. 4.12(b) of A, A is regarded as a part of a larger current gesture (B), which includes A and extends beyond the CEP of A. All the CEPs of A are ignored.
b. When B starts between the first and the last CEPs of A, as in Fig. 4.12(c), the immediately preceding CEP is chosen as the end point of A.

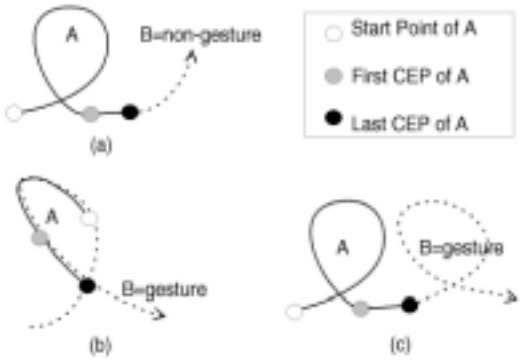



Figure 4.12 Detection of an end point. (a) Non-gesture is following. (b) Nested gesture. (c) Gesture is following. Each dark curve A presents the preceding gesture that may be reported and the dotted curve B represents the immediately pattern or gesture.

Since they constructed the threshold model by combining all the gesture models in the system, the number of states in the threshold model is equal to the sum of the states of all gesture models excluding the start and final states. This means that the number of states in the threshold model increases as the number of gesture models increases. Consequently, the increased number of states increases time and space requirements. To overcome this problem, they propose using relative entropy to reduce the number of states of the threshold model. In their gesture spotter, they quantize the direction of hand movement in 16 values and the number of states in the threshold model is reduced from 44 to 24. Consequently, the expected saving of the matching time with such a reduction is 63.91 percent.

#### 4.2.3.1 Comments

This paper is about the HMM-based threshold model approach for gesture recognition. The task of automatic gesture recognition is highly challenging due to the presence of unpredictable and ambiguous non-gesture hand motions. To handle non-gesture patterns, they introduce the concept of a threshold model that calculates the likelihood threshold of an input pattern and provides a confirmation mechanism for the provisionally matched gesture patterns. Because some weakness of the threshold model, the likelihood can be used as an adaptive threshold for selecting proper gesture model. The other advantage of their approach is that they merge the states with similar probability distributions, utilizing the relative entropy. By using this method a lot of states can be reduced. Because without reduction the threshold model is constructed by collecting the states of all gesture models in the system, the time and space requirement will be increased. In their paper they also give a lot of math explanation for their algorithm. Their method is highly accurate and also has very fast speed for real-time applications. Experimental results show that the proposed method can successfully extract trained gestures from continuous hand motion with 93.14 percent reliability.

### 4.2.4 Gesture Modeling and Recognition Using Finite State Machines

**[Hon02]**

Pengyu Hong, Matthew Turk and Thomas S. Huang's method [Hon02] proposed a technique for gesture modeling and recognition in real-time, interactive environments. The training data consists of tracked 2D head and hand locations, captured while performing each repeatedly. The spatial and temporal information of the data are first decoupled. In the first phase, the algorithm learns the distribution of the data without temporal information via dynamic *k*-means. The result of the first phase provides support for data segmentation and alignment. The temporal information is then learned from the aligned data segments. The spatial information is then updated. This produces the final state sequence, which represents the gesture. Each state sequence is a FSM recognizer for a gesture.

#### 4.2.4.1 Gesture Modeling

The features used for input to gesture modeling and recognition, are the 2D positions of the centers of the user's face and hands (see Fig. 4.13). Basically, a gesture is defined as an ordered sequence of states in the spatial-temporal space. Each state $S$ has parameters $<\vec{\mu}_s, \Sigma_s, d_s, T_s^{\min}, T_s^{\max}>$ to specify the spatial-temporal information captured by it, where $\vec{\mu}_s$ is the 2D centroid of a state, $\Sigma_s$ is the $2\times 2$ spatial covariance matrix, $d_s$ is the distance threshold, and $\left[T_s^{\min}, T_s^{\max}\right]$ is a duration interval. The spatial-temporal information of a state and its neighbor states specifies the motion and the speed of the trajectory within a certain range of variance.

Figure 4.13 Gesture features

#### 4.2.4.2 Computing the gesture model

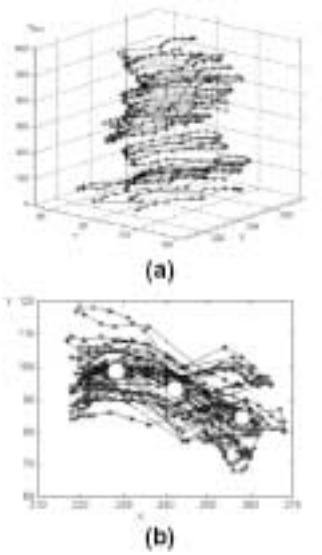



Figure 4.14 "Wave left hand" gesture. (a) With temporal information. (b) Without temporal information.

They first decouple the temporal information from the spatial information, roughly learn the spatial information, incorporate the temporal information, and then refine the spatial information. The training data is captured by observing a gesture repeated several times continuously. The learning is divided into two phases as follows.

In the spatial clustering phase, they learn the distribution of the data distributions without temporal information. A threshold is defined to specify the coarse spatial variance allowed for a gesture. Currently this threshold is fixed. Each state $S$ indicates a region in the 2D space that is represented by the centroid $\vec{\mu}_s = E\left(\vec{x}\right)$ and the covariance matrix $\Sigma_s = E\left(\left(\vec{x}-\vec{\mu}\right)^T\left(\vec{x}-\vec{\mu}\right)\right)$, where $\vec{x} = (x, y)$. Given input data $\vec{x}$, the distance from the data to the state $S$ is

defined as the Mahalanobis distance: $D\left(\overrightarrow{x},S\right)=\sqrt{\left(\overrightarrow{x}-\overrightarrow{\mu_s}\right)\Sigma_s^{-1}\left(\overrightarrow{x}-\overrightarrow{\mu_s}\right)^T}$. At beginning, they assume that the variance of each state is isotropic. Beginning with a model of two states, they train the model to represent the data using dynamic *k*-means. When the error improvement is very small, they split the state with the largest variance that is higher than the chosen threshold. The training stops after all the variances of the states drop below the threshold. For example, after training, the data in Fig. 4.14(a) has three states whose centroids are represented by the white circles in Fig. 4.14(b).

In the temporal alignment phase, each data point is assigned a label corresponding to the state to which it belongs. Thus they get a state sequence corresponding to the data sequence, as illustrated in Fig.4.14. By manually specifying the temporal sequence of states from the gesture examples, they obtain the structure of the Finite State Machine (FSM) for the gesture. For example, the state sequence for one cycle of the "wave left hand" gesture, shown in Fig.4.15, is [1 2 0 2 1]. Once this is determined, the training data is segmented into gesture samples. A sample of [1 1 1 2 2 2 2 0 0 0 0 2 2 2 1 1], for example, consists of the five states with (3, 4, 4, 3, and 2) samples per state, respectively. The number of samples in a state is proportional to the duration of the state. The example gestures are all aligned in this manner, resulting in N examples with the same state sequence, differing only in the number of samples per state. Calculating the minimum and maximum number of samples per state over the training data set the duration interval $\left[T_s^{\min},T_s^{\max}\right]$. Since the user may stay at the first state of the FSM for an indefinite period of time, they set $T_0^{\max}$ to be infinite. They use the simple FSM to model the spatial-temporal information of the gestures. The temporal information of each state is represented by a duration variable whose probability is modeled as uniform over a finite interval $\left[T_s^{\min},T_s^{\max}\right]$. The duration variable tells approximately how long the trajectory should stay at a certain state.

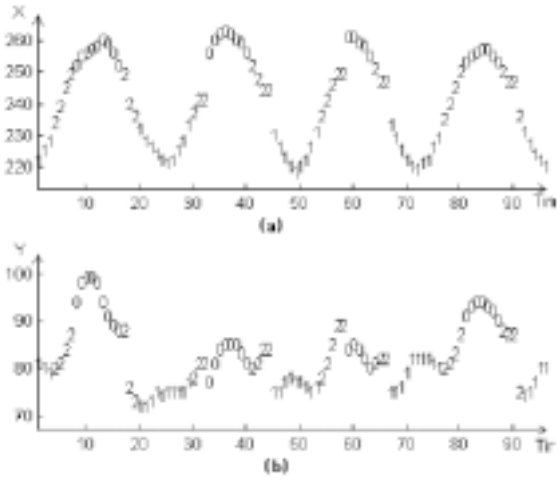



Figure 4.15 The state sequence for one cycle of the "wave left hand" gesture

#### 4.2.4.3 Recognition

Real-time, online recognition is done by considering only the data acquired at the current time point. A gesture is recognized when all the states of a gesture recognizer are passed. Although they only examine the data sample at current time point, they do use the context information stored in the FSM for recognition. The context information of a gesture recognizer *g* can be represented as: $c=<s_k,t>$ where $s_k$ is the current state of the recognizer *g*, *t* is the how long the recognizer has stayed at $s_k$.

Since a FSM is an ordered state sequence, $s_k$ stores the history of the trajectory. When a new data sample *x* comes, if one of the following conditions is met, the state transition happens.

(1) $(D(x, s_{k+1}) \le d_{k+1})$ & $(t > t_k^{\max})$

(2) $(D(x, s_{k+1}) \le d_{k+1})$ & $(D(x, s_{k+1}) \le D(x, s_k))$ & $(t \ge t_k^{\min})$

(3) $(D(x, s_{k+1}) \le d_{k+1})$ & $(t > t_k^{\max})$

The recognizer only takes into account the current data sample, since the past is modeled by the current state. Each state *S* has its own threshold $d_s$. If the new data *x* does not belong to the current state and the state transition cannot happen, the recognizer is reset, Thus the computation complexity at each time point is approximately *O(n)* where n is the total number of the FSM models. If a data sample happens to make more than one gesture recognizer fire, there is an ambiguity. To resolve the ambiguity, they choose the gesture with the minimum average accumulate distance: $Gesture = \arg\min_g \left( \sum_{i=1}^{n_g} D\left( \overset{\rho}{x}_s, s_{gi} \right) / n_g \right)$, where $g_{si}$ is the state of the gesture *g* that the data sample $\overset{\rho}{x}_i$ belongs to, $n_g$ is the number of the data accepted by the recognizer of the gesture *g* up to the current time point.

#### 4.2.4.4 Comments

A major advantage of the method is that it handles gestures which with different number of states. Their method quickly produces a recognizer for different gestures by specify only the variance, even when only a few training examples are available. Potentially, their approach is able to handle gestures with trajectories that contain loops with more than one intersection. Real-time, online recognition is done by considering only the data acquired at the current time point. A gesture is recognized when all the states of a gesture recognizer are passed. This is different from the traditional approaches, which require that the data segment provided to the recognizer contain the complete gesture data. There are similarities between HMMs and their approach. One difference is that with HMMs the number of states and the structure of the HMM must be predefined. To train a HMM, well-aligned data segments are required. The FSM method they proposed segments and aligns the training data and simultaneously produces the gesture model. They are currently pursuing an unsupervised model construction method to make the training simpler and better. During the recognition phase of an HMM, the system takes a segment of data as input, calculates the combined probability of the membership, compares the probability with a threshold, and decides whether the data is accepted or rejected. In their approach, since each state is associated with a threshold that is learned from the data, recognition is done based on the data at current point in time and the context information that is stored in the FSM. This dramatically reduces the computation complexity.

## 4.3 Gesture Recognition Approaches Conclusion

We found four different kinds of models for gesture recognition; they are improved HMM model approach, Condensation extension algorithm approach, HMM-based threshold model approach, Finite State Machines model approach. These approaches are based on different kinds of models and also apply for different kinds of gesture.

- The improved HMM model mentioned in [Psa02] is constructed for a behavior recognition in a certain environment, the theory is based on HMM, but introduced observation density into the model so that the whole system got improvement comparing with pure HMM methods, but the paper did not mention the speed of system, which is a little bit pity for us.
- The condensation extension algorithm gave an approach to raise the gesture recognition rates. It gave a curve model set that can help us gesture models easily. After the tests, it indeed makes the rates higher, but the disadvantage is the speed is not fast, cannot used in real time application.
- The HMM-based threshold model approach is to recognize the user's hand gesture from an image sequence in real time using HMM model, and for handling the non-gesture patterns they introduce the concept of a threshold model that calculates the likelihood threshold of an input pattern and provides a confirmation mechanism for the provisionally matched gesture patterns, they also use the merging states way to speed up the application.
- The Finite State Machines model approach is used in a real-time interactive system. The feature of the gesture is the center of the user's head and both hands. The recognition is done based on the data at current point in time and the context information that is stored in the FSM. This model has low computation complexity.

Now from the usual criteria of a gesture recognition model, we can conclude these four models in

Table 4.9 Comparison of gestures models

| Model Name | Improved HMM | HMM-based Threshold | FSM | Improved Condensation |
|---|---|---|---|---|
| Features | Walking Behavior | Hand positions | Center of the user's head and both hands | Hand positions |
| Model | HMM | HMM | FSM | Condensition |
| Training | Yes | Yes | Yes | Yes |
| Complexity | High | High | Low | High |
| Real Time | Yes | Yes | Yes | No |
| Accurate | High | Very High | High | High |
| Paper Index | [Psa02] | [Lee99] | [Han02] | [Bla98] |

The gesture recognition is still a very hot topic being studied. Until now the HMM model is the most popular approach with high accurate and it can be used in the real-time processing. But HMM model will need a lot of effort with the training step, so there are still lots of other simple models can be used. How to choose a good and fitful model for the application depends on what kind of features you will take, what kind of gesture that you want to recognize, how fast the recognition speed that you need, and so on.

# Chapter 5

# 5 Literature Summary

Based on the main requirements of our application UI-Wand, we divide our literature research into the following three main sub topics as Corner detection, Corner classification and recognition, Gesture recognition. We found a lot of papers related to each of these three topics. Our literature survey is organized in the same order and each part relates to one requirement of our UI-Wand application.

In the first part of our literature survey, we define the UI-Wand application and introduce the following part of the survey.

In the second part of our literature survey, we introduced a lot of corner detectors. There are lots of different approaches already exist, among them boundary based corner detectors, direct corner detectors and color distribution based detectors are the main three paradigms. Each kind of approaches has its own properties, for example, direct corner detectors have very fast processing speed but the accurate is always not very high; color distribution based detectors always have very high accurate and some of them can detect edges, corners and junctions within one round, but sometimes their processing speed is not fast enough for real time application. Because UI-Wand needs to detect accurate four screen corners in real time, so we must focus our application on a corner detector with very high accuracy and also very fast processing speed. It is a big challenge for us, it seems that these two criterions are often conflict to each other. In quite recent years, this conflict seems to be solved. There are a few new detectors appeared, which are also based on the traditional approaches but have excellent performance. They have fast speed for real time processing and can give satisfied accuracy result in corner detection, which is exactly what we need in our UI-Wand application.

After getting accurate corner positions in the image sequence, we need to know which corner is exact the screen corner. So in the third part of our literature survey, we carried out a lot of approaches can be used. One traditional approach is to use the geometric relations to get the screen corner. But this approach need more information about the shape of the object, here in UI-Wand, we need to know the edge information, distance information or more other information. And the biggest drawback of the geometric approach is that it cannot be extended if we change the application into the other fields. For example, if we do not use UI-Wand to detect the screen point anymore, we must change all the geometric information, because for a new object, the former geometric information will be totally useless. So we prefer to use the classification way for corner recognition. There are a lot of models can be used as classification model, such as HMM (Hidden Markov Model), KNN (K Nearest Neighborhood), SVM (Support Vector Machine) and RVM (Relevance Vector Machine). Among these four models, HMM is a little bit different from other three models, it describe the states transition status, and it has already been successfully used in speech recognition, face recognition and gesture recognition. It can give a very high recognition rates when dealing with these problems. KNN is a very old and quite simple model, for simple classification and small data set, it can

reach a high classification rates with a high speed. But to deal with more complicated problems, SVM and RVM will be more efficient, especially for the object recognition problems. Comparing between RVM and SVM, RVM is a quite new model, and it can give probability output, which is the classification confidence that SVM can not gives. But SVM has much faster training speed than RVM. Because SVM is already a well-known model, in most object classification papers, authors still choose SVM as their classification model. As to RVM, the research is still going on, and there are still some open questions existing. But according to the properties of RVM, it is a very attractive model for us.

From classification step, we can get four 2D corner positions in the digital image. Using these 2D positions as the input we can get 3D positions of these screen corners within UI-Wand by searching in a 3 dimensional space. After the positioning work, we can track the gestures of UI-Wand and finally recognize these gestures. We focused on the gesture recognition in the fourth part of our literature survey. In the gesture recognition part, a lot of similar applications can be found, but they are based on different features and also different kind of gestures. We found out four different kinds of models in gesture recognition; they are improved HMM model approach, Condensation extension algorithm approach, HMM-based threshold model approach, Finite State Machines (FSM) model approach. Improved HMM model approach is mainly based on HMM model but introduce observation density into the model, which can improve the performance of HMM. The condensation extension algorithm can raise the gesture recognition rates. It gave a curve model set that can help us to recognize gesture models more easily. Using this model can give high accuracy but the speed cannot be real time. HMM-based threshold model is a quite good approach, it introduces the concept of a threshold model that calculates the likelihood threshold of an input pattern and provides a confirmation mechanism for the provisionally matched gesture patterns, they also use the merging states way to speed up the application. This model can correctly recognize more complicated gesture in real time. Quite different from the above HMM based models, FSM model is quite simpler and this approach is able to handle gestures with trajectories that contain loops with more than one intersection. Real-time, online recognition is done by considering only the data acquired at the current time point. For our UI-Wand application, we do not expect too complicated modeling and training work to be done such as using HMM model. Inspired by the pattern recognition idea, we think we can try a new way in gesture recognition part, RVM or other classification model can be used to recognize different gestures by their different tracking positions, which can be easily got from former steps in UI-Wand. But with no existing experience, the efficiency of this approach cannot be known clearly.

In the conclusion part of every chapter, we gave many detailed technical tables to compare different models and algorithms that dealing with the same problem. In this chapter, we summarized the most important 25 papers that we read during the literature research period in the following Table 5.1, in order to give readers a brief overview for papers about their topics and reading value. The columns of this table we indicate as following:

- In paper type, we use [APP] to indicate that this is a paper about an application or approach to its corresponding topic. The same thing, we use [ALG] to say it is an algorithm, [THE] is theory, [TES] is test, and [SUR] means it is a survey.

- In paper topic column, we filled in the topic or model that this paper mainly focuses on. If this paper is about an application, then we give the application field after a comma.
- Paper Quality of the table marks our evaluation for the quality of this paper, and "Reading Value" item means how valuable it is for UI-Wand. We used score from 1 to 5 to mark. For those scores are highest, 5, we used gray color to highlight.

Table 5.1 Features of important papers investigated in this literature survey

| Paper Index | Paper Type | Paper Topic | Paper Quality | Reading Value |
|---|---|---|---|---|
| [Ard00] | [APP] | SVM, Shape | 3 | 4 |
| [Avi01] | [APP] | Track, Objects | 4 | 4 |
| [Bla98] | [ALG][APP] | Gesture, Hands | 2 | 2 |
| [Cor95] | [THE] | SVM | 4 | 4 |
| [Der93] | [ALG] | Corner Detector | 3 | 4 |
| [Har88] | [ALG] | Corner Detector | 3 | 3 |
| [Hei03] | [APP] | SVM, Face | 3 | 3 |
| [Hon02] | [APP] | Gesture, Head and Hands | 4 | 4 |
| [Lee99] | [ALG][APP] | Gesture, Hands | 4 | **5** |
| [Lus95] | [THE] | HMM | 4 | **5** |
| [Nef98] | [THE][APP] | HMM, Face | **5** | 4 |
| [Pon98] | [TES] | SVM | 3 | 3 |
| [Psa02] | [APP] | Gesture, Behavior | 3 | 3 |
| [Ruz01] | [THE] | Corner and Edge Detector | 4 | 4.5 |
| [Smi97] | [THE] | Corner and Edge Detector | 4 | 4 |
| [Soj02a] | [SUR][ALG] | Corner Detector | **5** | **5** |
| [Son03] | [ALG] | Corner and Edge Detector | 4 | **5** |
| [Tip01] | [THE] | RVM | **5** | **5** |
| [TipNEW] | [THE][ALG] | RVM | **5** | 4 |
| [Tra98] | [ALG] | Corner Detector | 3.5 | 3.5 |
| [Web02] | [THE] | Pattern Recognition | **5** | **5** |
| [Wel01] | [THE] | Kalman Filter | 4 | 4 |
| [Wil03] | [APP] | Track, Objects | 4 | 4 |
| [Zha04] | [ALG] | KNN Classification | 4 | 3 |
| [Zhe99] | [ALG] | Corner Detector | 3 | 3 |

After widely researching on each related technique that will probably be applied in the UI-Wand application, we get an overview of almost all the existing approaches in each part of the whole application, and now we have a more clear idea about how to realize the whole system. But UI-Wand is still a quite new application in computer vision, no more experience can be found from others, so there are still a lot of exploring work waiting for us.

# 6 Reference

[Ale98] A.Alexandrov. Corner detection overview and comparison, *Computer Vision*, vol. 588, 2002

[Ard00] E.Ardizzone, A.Chella, R.Pirrone. Feature-based shape recognition by Support Vector Machines, *Proc. of ECAI 2000 Workshop on Machine Learning in Computer Vision*, Berlin, Germany, Aug 2000.

[Avi01] S.Avidan. Support Vector Tracking. Proc. Conf. Computer Vision and Pattern Recognition, Hawaii, 2001.

[Bae01] S.C.Bae, I.S.Kweon and C.D.Yoo. A new corner detector, *Pattern Recognition Letters*, vol. 23, pp. 1349-1360, 2001.

[Bea78] P.R.Beaudet. Rotationally invariant image operators, *Proc. Fourth Int. Joint Conf. on Pattern Recognition*, Tokyo, pp. 579-583, 1978.

[Bla98] M.J.Black and A.D.Jepson. Recognizing temporal trajectories using the condensation algorithm Automatic Face and Gesture Recognition, *Proc. Third IEEE International Conf.*, pp. 16–21, Apr 1998.

[Bro86] A.J.Broder. Strategies for Efficient Incremental Nearest Neighbor Search, *Pattern Recognition*, vol. 23, pp. 171-178, Nov 1986.

[Bro95] R.L.Brown. Accelerated Template Matching Using Template Trees Grown by Condensation, *IEEE Trans. Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 523-528, Mar 1995.

[Bru92] K.Brunnstrőm, T.Lindeberg, and J.O.Eklundth. Active detection and classification of junctions by foveation with a head-eye system guided by the scale-scape primal sketch, *Proc.* $2^{nd}$ *European Conf. on Computer Vision, Springer-Verlag LNCS*, vol. 558, pp. 701-709, 1992.

[Cor95] C.Cortes and V.N.Vapnik. Support Vector Network, *Machine Learning*, vol. 20, pp. 273-297, September 1995

[Der93] R.Deriche and G.Giraudon. A computational approach for corner and vertex detection, *International Journal of Computer Vision*, vol. 10, issue. 2, pp. 101-124, 1993.

[Dre82] L.Dreschler and H.H.Nagel. On the selection of critical points and local curvature extrema of region boundaries for interframe matching, *Int. Conf. on Pattern Recognition*, pp. 542-544, 1982.

[Gon87] R.C.Gonzalez and P.Wintz. *Digital Image Processing*, Addison-Wesley, ii edition, 1987.

[Har88] C.G.Harris and M.Stephens. A combined corner and edge detector, *Proc* $4^{th}$ *Alvey Vision Conf.*, pp.189-192, Manchester, 1988.

[Hei03] B.Heisele. Visual Object Recognition With Supervised Learning. *IEEE Tran. Intelligent Systems*, vol.18, no.3, May-June 2003.

[Hon02] P.Hong, M.Turk and T.S.Huang. Gesture Modeling and Recognition

Using Finite State Machine, *IEEE Conf. on Face and Gesture Recognition*, 2002.

[Isa96] M.Isard and A.Blake. Contour tracking by stochastic propagation of conditional density. *Proce. European Conf. on Computer Vision*, pp. 343-356, 1996.

[Isa98] M.Isard and A.Blake. A mixed-state condensation tracker with automatic model-switching, *Proc. 6th Internal Conf. Computer Vision*, pp. 107-112, 1998.

[Kal60] R.E.Kalman. A new approach to linear filtering and prediction problems, *Tran. the ASME—Journal of Basic Engineering,* vol. 82, series. D, pp.35-45, 1960.

[Kit82] L.Kitchen and A.Rosenfeld. Gray-level corner detection, *Pattern Recognition Letters*, vol. 1, pp. 95-102, 1982.

[Lee99] H.K.Lee and J.H.Kim. An HMM_based threshold model approach for gesture recognition, *IEEE Tran. Pattern Analysis and Machine intelligience,* vol. 21, no.10, 1999.

[Lus95] E.F.Lussier. Markov Models and Hidden Markov Models: A brief tutorial, *Introduction to Artificial Intelligence at the University of California, Berkeley,* 1995 semester of CS188.

[Nef98] A.V.Nefian and M.H.Hayes. Hidden Markov Models for Face Recognition, *Proc. International Conf. on Acoustics, Speech and Signal Processing,* vol. 5, pp. 2721-2724, Seattle, 1998

[Nef99] A.V.Nefian and M.H.Hayes III. Face recognition using an embedded HMM. *Proc. Audio- and Video-Based Biometric Person Authentication (AVBPA'99),* Washington DC, USA, 1999.

[Nob88] J.A.Noble, Finding corners, *Image and Vision Computing*, vol. 6, no. 2, pp.121-128, 1988.

[Pen91] A.Pentland and M.Turk. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.

[Pon98] M.Pontil. Support Vector Machines for 3D object Recognition, *IEEE Tran. Pattern analysis and Machine intelligence*, vol. 20, no. 6, pp. 637-646, Jun 1998.

[Psa02] A.Psarrou, S.G.Gong, M.Walter. Recognition of human gestures and behavior based on motion trajectories, *Image and Vision Computing,* vol.20, Issues.5-6, pp.349-358, 2002

[Rit75] G.L.Ritter, H.B.Woodruff, S.R.Lowry, and T.L.Isenhour. An Algorithm for a Selective Nearest Neighbor Decision Rule. *IEEE Trans. Information Theory*, vol. 21, pp. 665-669, Nov 1975.

[Roh94] K.Rohr. Localization properties of direct corner detectors, *Journal of Mathematical Imaging and Vision*, vol. 4, pp. 139-150, 1994.

[Ruz01] M.A.Ruzon, and C.Tomasi. Edge, junction, and corner detection using

color distributions, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1281-1295, 2001.

[Ruz99a] M.A.Ruzon and C.Tomasi. Color Edge Detection with the Compass Operator, *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp.160-166, 1999.

[Ruz99b] M.A.Ruzon and C.Tomasi. Corner Detection in Textured Color Images, *IEEE Seventh International Conf. on Computer Vision*, vol. 2, pp.1039-1045, 1999.

[Smi97] S.M.Smith, J.M.Brady. SUSAN-A new approach to low level image processing, *International Journal of Computer Vision*, vol. 23, pp. 45-78, 1997.

[Soj02a] E.Sojka. A new algorithm for detecting corners in digital images, *Proc. SCCG 2002*, ACM SIGGRAPH, NY, pp. 55-62, 2002.

[Soj02b] E.Sojka. A new and efficient algorithm for detecting the corners in digital images, *Proc. 24th DAGM Symposium, Springer, LNCS 2449, Berlin, NY*, pp. 125-132, 2002.

[Soj03] E.Sojka. A new approach to detecting the corners in digital images, *Accepted for publication in IEEE ICIP*, 2003.

[Son03] J.Song, M.R.Lyu, and M.Cai. A generic color-distribution-based approach to detect edges, corners and junctions simultaneously in color images, submitted to *IEEE Trans. Image Processing*, 2003.

[Tal98] A.Talukder and D.Casasent. A general methodology for simultaneous representation and discrimination of multiple object classes. *Optical Engineering, Special Issue on Advanced Recognition Techniques*, Mar 1998.

[Tip01] M.E.Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, vol. 1, pp. 211-244, 2001.

[TipNEW] M.E.Tipping. Fast Marginal Likelihood Maximization for Sparse Bayesian Models.

[Tra98] M.Trajkovč and M.Hedley. Fast corner detection, *Image and Vision Computing*, vol. 16, pp. 75-87, 1998.

[Vap95] V.N. Vapnik. *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag, 1995.

[Wan95] H.Wang and M.Michaelis. Real-time corner detection algorithm for motion estimation, *Image and Vision Computing*, vol. 13, no. 9, pp. 695-703, 1995.

[Web02] A.R.Webb. Statistical Pattern Recognition, second edition, John WILEY & SONS, Ltd, 2002.

[Wel01] G.Welch and G.Bishop. An introduction to the Kalman filter. *University of North Carolina at Chapel Hill, Department of Computer Science*, Course 8 SIGGRAPH, 2001.

[Wil03] O.Williams, A.Blake and R.Cipolla. A Sparse Probabilistic Learning Algorithm for Real-Time Tracking. *Int. Conf. on Computer Vision, ICCV,* Nice, France, 2003.

[Yan99] M.H.Yang and N.Ahuja. Recognizing hand gesture using motion trajectories, *IEEE CS Conf. on Computer Vision and Pattern Recognition*, vol.1, pp. 466-472, Jun 1999.

[Zha04] B.Zhang, S.N.Srihari. Fast *k*-Nearest Neighbor Classification Using Cluster-Based Trees, *IEEE Tran. Pattern Analysis and Machine Intelligence*, vol. 26, no. 4, Apr 2004.

[Zhe99] Z.Zheng, H.Wang and E.K.Teoh. Analysis of gray level corner detection, *Pattern Recognition Letters*, vol. 20, pp.149-162, 1999.

[Zun83] O.A.Zuniga and R.M.Haralick. Corner detection using the facet model, *Proc. IEEE, Conf. on Pattern Recognition and Image Processing*, pp. 30-37, 1983.