國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

以隱含形狀模型為基礎之靜態手勢辨識

Static Hand Posture Recognition Based on an Implicit Shape Model

劉哲維

Che-Wei Liu

指導教授：顏嗣鈞 博士

Advisor: Hsu-Chun Yen, Ph.D.

中華民國 99 年 7 月

July, 2010

國立臺灣大學碩士學位論文
口試委員會審定書

以隱含形狀模型為基礎之靜態手勢辨識
Static Hand Posture Recognition Based on
an Implicit Shape Model

本論文係劉哲維君（R97921072）在國立臺灣大學電機工程學系完成之碩士學位論文，於民國 99 年 7 月 27 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

______________________________（簽名）
（指導教授）

__________________　__________________

__________________　__________________

系主任、所長 ______________________（簽名）

誌謝

能完成這篇論文，首先要特別感激我的指導教授顏嗣鈞老師，讓我在探索知識之路上能自由地往感興趣的學術領域前進,並感謝老師適時地給予資源與指導。感謝口試委員雷欽隆教授、莊仁輝教授、郭斯彥教授給予許多寶貴意見，而使得本論文能更加完整。

感謝台大資工系徐宏民教授開授的「高等多媒體資訊分析與檢索」課程，在此課程所學習到的內容，成為本論文最主要的靈感來源。另外，也感謝台大資工系王傑智老師的「機器人知覺與學習實驗室」提供該實驗室所使用的手勢資料庫，使得本論文的評估得以順利完成。感謝趨勢科技曾志新經理和 Findbook.tw 團隊給我工讀實習的機會，在工讀期間見習到許多業界實務內容，從中學習到的程式技能讓我得以順利將學術上的靈感付諸實現。

感謝在計算理論實驗室的每一位成員。感謝澤乙，家銘，俊利同窗這二年研究生活一起努力與切磋。感謝學弟郁元，幫忙把本學位論文改寫成會議論文版本，使之能順利發表於第 23 屆電腦視覺、圖學暨影像處理研討會之上。感謝實驗室學長奕廷、秉賢、克仁、春成、建良、紹祁、柏源、紀憲，以及學弟威毅、品翰，謝謝各位平時的對我的指教與照顧。

感謝父母、哥哥給我的支持與鼓勵，有家人在背後支持與關懷是我最大的精神支柱，讓我能無後顧之憂大膽前進。感謝女友雨蓁的相陪，在我遇到困難時能給予我突破難關的勇氣與力量。謝謝你們。

劉哲維 謹誌於

國立臺灣大學電機工程學研究所

中華民國九十八年七月

中文摘要

手勢辨識在人機互動領域是相當熱門的研究主題，因為手勢是人類自然的溝通方式。先前的研究主要是專注在固定尺寸的影像上，評估是否符合某手勢的特徵，而常見方法是使用機器學習的 AdaBoost 演算法尋找手勢的重要特徵。近年來，局部特徵演算法逐漸受到重視，因為具備許多重要性質的強健性，例如亮度、尺度、方向等等。因此本論文改進了以局部特徵為基礎的隱含形狀模型方法，並使用此方法來解決靜態手勢辨識問題。我們發現精確度相較於先前文獻方法增進，並且我們的方法具有偵測手勢的方向，以及可辨識不同角度的手勢等特點。最後，本論文採用的演算法執行時間近乎即時，可用於一般的靜態手勢辨識應用，或是作為動態手勢的基礎。

關鍵詞：隱含形狀模型、靜態手勢辨識。

ABSTRACT

Hand gesture recognition has become increasingly popular in Human-Computer Interaction (HCI) research as gestures provide a natural way of communication. Previous research has focused on searching a fixed size sub-window by evaluating a subspace of feature space that is found from machine learning algorithms such as AdaBoost. In recent years, however, local features have become increasingly popular as they offer robustness in illumination of the environment, scale, and rotational invariance of the hand itself. In this thesis, we describe a novel method of static hand posture recognition that is based on an *Implicit Shape Model* (ISM) of local features. We find improvement in recognition accuracy over former methods. In addition, our algorithm enhances the sliding-window paradigm by providing useful information such as hand orientation and rotational invariance. The execution time of the algorithm is also provided in order to assess its potential to be incorporated into a near real-time posture recognition application or a hand gesture system module.

Keywords: Implicit shape model; static hand posture recognition.

CONTENTS

口試委員會審定書 .... #

誌謝 i

中文摘要 ii

ABSTRACT iii

CONTENTS iv

LIST OF FIGURES vi

LIST OF TABLES viii

Chapter 1 Introduction 1

Chapter 2 Related Work 4

2.1 Feature Algorithms 4

2.1.1 Harr-like Feature 4

2.1.2 Modified Census Transform (MCT) 5

2.1.3 Scale Invariant Feature Transform (SIFT) 6

2.2 Classification Methods 8

2.2.1 AdaBoost Approach 9

2.2.2 Support Vector Machine Approach 9

2.2.3 Explicit Shape Model Approach 10

2.3 Visual Word Representation 10

2.3.1 Visual Codebook Generation 11

2.3.2 Vector Projection 12

2.3.3 Visual Word with Spatial Information 12

Chapter 3 Hand Posture Recognition with an Implicit Shape Model 14

3.1 Overview of Our Approach 14

3.2 Learning the Implicit Shape Model 16

3.2.1 Visual Vocabulary Construction 16

3.2.2 Hand Center Estimation 18

3.2.3 Record the Occurrence Vectors 19

3.3 Posture Recognition 24

3.3.1 Hand Center Prediction 24

3.3.2 Voting Procedure 25

3.3.3 Preprocessing the Images 29

3.3.4 Summary of Our Recognition Algorithm 30

Chapter 4 Experimental Results 33

4.1 Hand Posture Recognition System 33

4.2 Evaluation 34

4.2.1 Hand Posture Database 34

4.2.2 Results of Triesch Hand Posture Database 36

4.2.3 Evaluation of T. C. Liu Posture Database 40

4.2.4 Running Time Analysis 42

Chapter 5 Conclusion and Future Work 45

REFERENCES 47

LIST OF FIGURES

Figure 2.1: Harr-like feature (a) four types of rectangle features (b) two most significant features selected from AdaBoost algorithm [14] 5
Figure 2.2 Example of the Modified Census Transform [16] 6
Figure 2.3: SIFT example (a) SIFT features on a posture (b) Matching result from Lowe's method. 8
Figure 3.1: An example of training phase and predicting phase. Rectangle denotes visual word; Rectangles of the same color indicate the same visual word. Red arrow represents the posture direction. Dotted vector is a vector from visual word to hand center 15
Figure 3.2: An example of visual words, each row is a visual cluster we called a visual word and features belong to a visual word exhibiting similar feature appearances 17
Figure 3.3: Some examples of hand center estimations from the T. C. Liu posture database 18
Figure 3.4: Some examples of hand center estimations from the Triesch posture database 19
Figure 3.5: Hand direction of training images. 21
Figure 3.6: The training procedure. For each training image, local features are extracted and clustered to visual words. The relationship between visual words and their center are record as occurrence vectors. Finally, we group the occurrence vectors by each posture classes. 22
Figure 3.7: Diagram of visual word and hand center 24
Figure 3.8: Choosing the most possible hand location by summing up all hand

predicting vectors in each sub-region and find the largest vectors to be our final prediction 27
Figure 3.9: Hand posture predicting result (a) ~ (c) are three posture prediction results. Because the magnitude of fist vector is largest, this posture image is more likely to be a fist. 28
Figure 3.10: Comparative of large and small visual words 29
Figure 3.11: Posture recognition overview 30
Figure 4.1: (a) Webcam (b) snapshot, the rectangle is center and the line denotes the posture direction 33
Figure 4.2: Samples of Triesch posture database 35
Figure 4.3: Samples of T. C. Liu database 35
Figure 4.4: Validation Result of the Triesch Database 37
Figure 4.5: 5-fold cross validation result of Liu Database 40

LIST OF TABLES

Table 4.1: Statistics of two posture databases 35

Table 4.2: Statistics of the Triesch hand posture database protocol 1 36

Table 4.3: A comparison of the recognition results in Just and ours 37

Table 4.4: Recognition results with more training data.. 38

Table 4.5: Comparison of metrics between of our method and Just [16]....................... 39

Table 4.6: The confusion matrix of our result at B=32, $\Delta\theta t = 0.125\pi$, $st = 3$ 41

Table 4.7: The confusion matrix of our result at B=32, $\Delta\theta t = 2\pi$, $st = 0$................... 41

Table 4.8: Recognition result comparison with ours and T. C. Liu [10] 42

Table 4.9: Summary of average time in training an ISM model 43

Table 4.10: Summary of average recognition time... 43

Chapter 1 Introduction

Hand gesture is one of the most intuitive communication methods among people. Recently, there has been a large amount of literature in human–computer interaction (HCI) focused on to developing a user-friendly interface by adopting hand gesture recognition system. For example, “wear Ur world” or Sixth Sense [1] and Ambient Assisted living [2] are two recently successful applications to use gesture as a module in their systems.

Static hand posture recognition is an essential component in hand gesture systems. In some works [2], [3], recognition is divided into two hierarchical levels. The lower-level part is the static posture recognition. The higher-level gesture part is based on the static posture output. It should be noted that hand *gesture* and *posture* are two terms with different meanings in literature [3]. A hand posture is a static hand pose, and a hand gesture is a sequence of static hand postures over continuous motion.

Gesture recognition is still a challenging problem in computer vision because hand shapes vary among people and the uniform appearance of hand so that make it is difficult to recognize hand in cluttered background or in the various posture classes.

Present approaches are mainly divided into vision-based and data-glove based methods. The data-glove based methods use sensors to detect movement of the hand and fingers. Using the data gloves can enhance the reliability of gesture system but such devices are expensive, a little cumbersome and not friendly to use. In other hand, the vision based methods usually use one or more [4] cameras as the input device. In this paper, we focus on vision-based method because it is more natural and convenient to use than data-glove based methods.

In recent years, there have been some works [5] using local features to deal with hand posture recognition. However, the local feature is often a high-dimensional vector so operating on raw feature vector is time consuming. Hence, Sivic [6] purposed a clustering method on features, and then the quantized features are called visual word. Using visual word not only increase the efficiency in matching but also the flexibility to adopt some techniques from Information Retrieval field.

Zhou et al. [7] are probably the first ones to work on the visual word concept for posture recognition. In addition, Wang [8] and Liu [5] used local feature algorithm (ex. SIFT) in posture recognition problem. However, few studies have provided an approach combine them, that is, how to use visual word from robust local feature algorithms to deal with the hand posture recognition.

Some works [9] based on visual words simply use the bag of visual words model, that is, discarding the spatial information of visual words. However, this approach may not be effective because different hand postures' visual words may be very similar, so the key information is the spatial information between them. Here, we propose an approach based on implicit shape model (ISM) [10], which has performed well on pedestrian detection [11]. The main idea of implicit shape model is that all local features vote for all possible object centers. Next, the densest region has a high probability of containing the object. We extend the implicit shaped model voting scheme so that is feasible in posture recognition with rotation cases and the results are assuring.

The main contributions of this thesis are the following:

1. Using implicit shape model to realize hand posture recognition and the improved accuracy considerably
2. For hand posture rotation issues, we extended the implicit shape model by considering the orientation of features
3. A novel method for concurrent posture localization and recognition

This thesis is organized as follows: In Chapter 2, we review some related works in posture recognition. In Chapter 3, the recognition algorithm based on an implicit shape model is addressed in details. In Chapter 4, we conclude our findings and indicate some direction for future works.

Chapter 2 Related Work

There have been many works in hand posture recognition field. In this chapter, we review some techniques related to our work and the approaches based on vision-based are introduced here.

2.1 Feature Algorithms

Although a digital image is composed of pixels, we usually do not use pixel color or intensity as image feature. We could like to represent an image as a set of feature points from some feature algorithms with some invariant properties.

2.1.1 Harr-like Feature

Using Harr-like features to detect objects has been adopt in many works [12], [13], [14]. Viola et al. [12] used Harr-like features in face detection and this framework is also feasible in hand detection [13].

To search an object in images, we specify sub-windows to evaluate whether the object is within them. For each sub-window, rectangle features are used to describe the image appearance. A rectangle feature is composed by grey parts and white parts. The feature value is the sum of pixel intensities in white rectangles subtracted from the sum of pixel intensities of grey rectangles. The exhaustive set of rectangle is large. Even each feature can be computed efficiently, but the computing completely set is prohibitively expensive. Hence, we typically select a small subset from feature space to represent an object class. The most common way to archive this is using AdaBoost, which will be address later in Section 2.2.1. Finally, some common rectangle features and the most significant feature in face detection are listed in Figure 2.1.

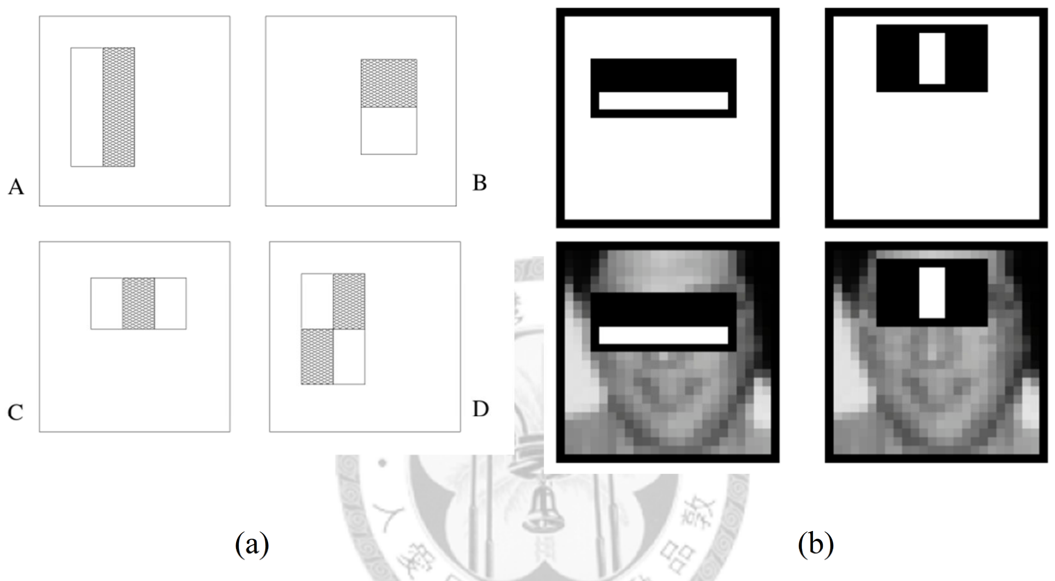


Figure 2.1: Harr-like feature (a) four types of rectangle features (b) two most significant features selected from AdaBoost algorithm [14]

2.1.2 Modified Census Transform (MCT)

Here, we briefly introduce the Modified Census Transform (MCT), which are proposed by Froba et al. [15]. The features from MCT are illumination invariant and have been successfully used for face detection [15] and posture recognition [16].The MCT feature space is defined as a set of 3x3 kernels. For a position at an image, we first compute the average pixel intensities between all neighborhoods, for example 3x3, is called mean intensity. Next, the neighbor pixels that their intensities are lower than mean are marked as "0" bit and the others are marked as "1" bit. An example is in Figure 2.2.This feature algorithm is used in Just et al. that is one of our baseline methods.

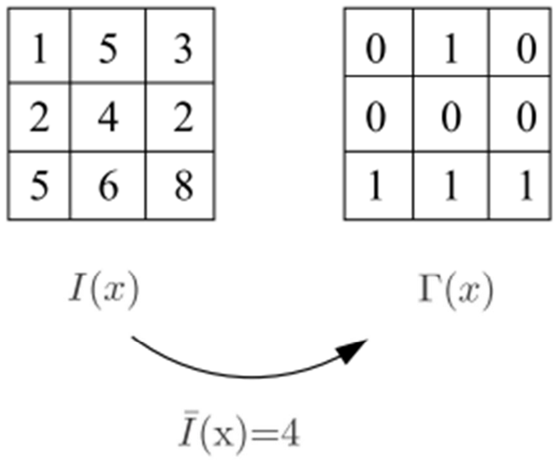


Figure 2.2 Example of the Modified Census Transform [16]

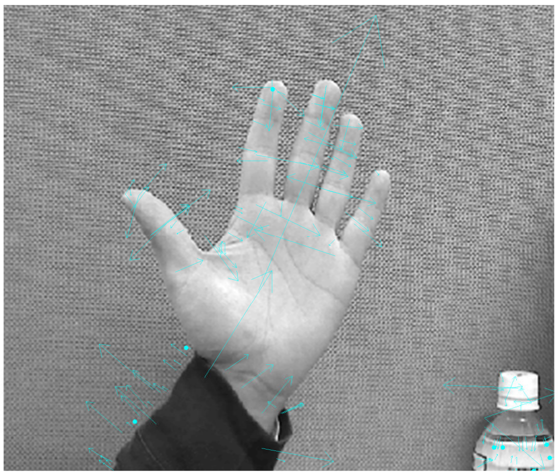
2.1.3 Scale Invariant Feature Transform (SIFT)

Scale invariant feature transform is an algorithm to detect and describe local features in images. As the name suggests, SIFT is invariant to scale. Moreover, it is also invariant to orientation, affine distortion and partially invariant to illumination [17]. With these properties, we often call SIFT feature as robust local feature. There are two stages to generate a local feature from an image. The former is to detect keypoints, that is, to find where the interesting point is. The latter is to describe the local feature as a vector so that the similarity of patches can be measured by their vector distance.

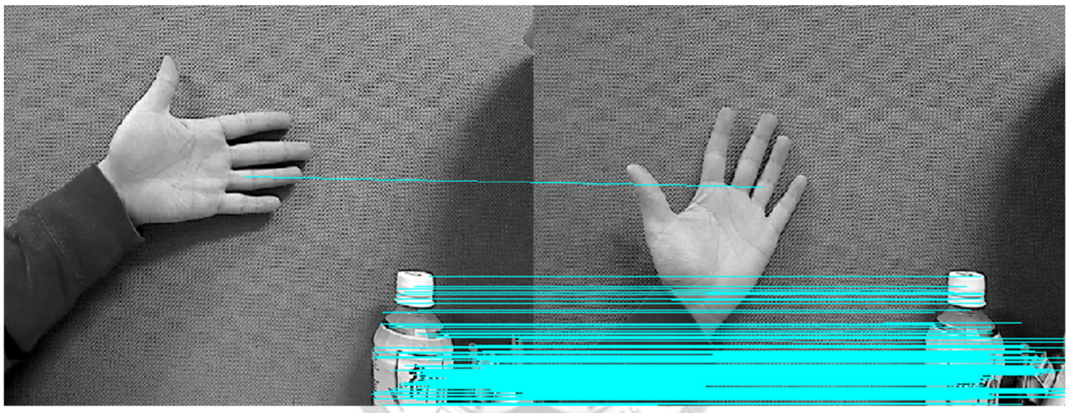
In Lowe's method, the keypoint detector algorithm is difference of Gaussian. Although two stages are generally called as SIFT algorithm, we can replace detector algorithm with others. The choice depends on application requirements. In addition, Fei-Fei et al. [18] indicated that using dense regular grid features outperform than interest points (sparse feature). However, the computation time required for dense feature is much more than sparse feature representation. Hence we use sparse feature for our posture recognition.

To the best of our knowledge, Wang et al. [8] are probably the first use SIFT to deal with the posture recognition problem. Because of these robust properties, we believe that SIFT is a good choice to be the feature method for posture recognition problem.

There is a matching algorithm proposed by Lowe et al. [17]. This scheme provided a naïve approach to measure the similarity of posture images. However, our experience showed that this works not well. The number of matching features are often too less to support the recognition decision. Some SIFT features and matching examples are in Figure 2.3. We can find that the matching algorithm provided by Lowe is good at rigid object, but it works not well on non-rigid object like human hands. Each SIFT feature is a 128 dimensional vector, that is too large to handle, so we used visual word mentioned latter to resolve that.



(a)



(b)

Figure 2.3: SIFT example (a) SIFT features on a posture (b) Matching result from Lowe's method.

2.2 Classification Methods

In previous section, we reviewed some feature methods to represent an image. Based on the features we extract from images, we expect the same posture categories will have similar features. In this section, we go through some common machine learning algorithms that are adopted in posture recognition or related field.

2.2.1 AdaBoost Approach

Many posture recognition approaches [8], [12], [19] apply AdaBoost to weight the importance of features. Feature space is usually too huge to handle all possible cases, so we need to know which subspaces of feature space are useful to our recognition problem. AdaBoost is short for Adaptive Boosting. The main idea of AdaBoost is to integrate a set of weak classifiers to be a strong one. A weak classifier is just slightly better than coin tossing. However, since varied viewpoints of weak classifiers, we can care about the problem separately. AdaBoost are repeatedly in T iterations. For each round, one weak classifier is selected for optimal classifier to solving problem. Then we emphasize the wrong samples and re-selected optimal classifier based on the emphasized samples. With this procedure, we can focus on different viewpoint of the problem in each round. Some of the advantages of AdaBoost are that it is less influenced by the overfitting problem, and it is efficient in running time because only a small subset of features we need to evaluate.

2.2.2 Support Vector Machine Approach

Support Vector Machine is a popular classification method in machine learning. It is a supervised learning method that data point is with a class label. Usually, a data point is in fixed dimensions.

Based on visual word framework, we use K dimensional vectors where K is the vocabulary size. The attribute is the times of visual word in the image, and this approach works well in many object classification problems [20], [21]. However, the spatial information is essential some problem like posture recognition, so discarding the spatial information of local features is infeasible. Some researchers tried to model co-occurrence relationships among visual words from the training images.

2.2.3 Explicit Shape Model Approach

In order to search objects in an image, we need to find object location. We can achieve this from explicit or implicit way to model object. Implicit method is based on voting scheme, which is our main core component. In contrast to implicit, explicit method is to model the probability density functions, which are often Gaussian distribution function. Then, the object shape is model explicitly. The objects are modeled as flexible constellations of parts [22], [23]. Each part represents a significant partition to an object. Take face model as an example, there possible result may to be the forehead, eye, and mouth part to model face object.

2.3 Visual Word Representation

In short, visual word is a clustering method on local features and then each feature is assigned to a cluster. The features among a cluster are considered as a same meaning, that is, a visual word. This procedure is the same as vector quantization from signal processing. In order to use visual word representation, we need to decide on the methods for visual codebook generation and vector projection. Some approaches are addressed as follows.

Every vector quantization method could always generate errors in some respects. Because we usually have no exact idea what the number of cluster is, and ambiguous features on near the cluster boundary may be assigned to wrong clusters. For this reason, the quality of visual codebook is a essential key for any recognition problem.

2.3.1 Visual Codebook Generation

A visual codebook contains the information of clusters. Hierarchical clustering and partition clustering are two common methods to build the visual codebook. Hierarchical clustering is conceptual more compact than partition clustering, but it is required more computation time and memory space. Hence, the hierarchical clustering only fit in small-scale problem.

The most well-known algorithm in partition clustering is k-means algorithm. Initially random points are chosen to be the initial centroids. Next, all points are assigned to its nearest centroid and the new centroids for next iteration updated from the average coordinates of same clusters. In moderate number of features, k-means works well in many visual word based recognition problems. However, one problem of k-means is over-sampling of dense regions and Radius-based clustering overcomes this problem [24].

2.3.2 Vector Projection

A main drawback of k-means is slow. To assign a feature to a cluster, we need to compute all distance between feature and centroids. In practice, the number of cluster could be in the millions scale, so the project time in transitional method is computationally expensive. Some indexing methods could help, for example kd-tree.

For features in large scale, some methods are better than k-means in terms of efficiency. For instance, random forest [25] and vocabulary tree [9] are usually good choices. These methods in some sense are not precise as traditional k-means, but the trade-off is worthy for large-scale datasets.

2.3.3 Visual Word with Spatial Information

In computer vision, the representation of visual word discarding spatial information is called “Bag of visual words model”. However, the loss of spatial information is imprecise in some applications. For example, two different of posture classes may share similar visual words. Therefore, bag of visual words model is not satisfied so that some researchers have been proposed visual word methods encoding spatial information to improve recognition accuracy.

Pyramid match kernel [26], [27] is a method to extend vector dimension so that the extra attributes contain spatial content. The image is partitioned into sub-regions and computing histograms of local feature inside each sub-region. Pyramid match kernel is effective for the scene category problem but for recognition problem with heavily rotation issue. Another approach is based on language model. Visual word is a term borrowed from text retrieval field. Tirilly et al. [28] proposed a language modeling approach to image classification. The main idea is to build a language model for an object class. In predicting phase, we find the most probability of language model.

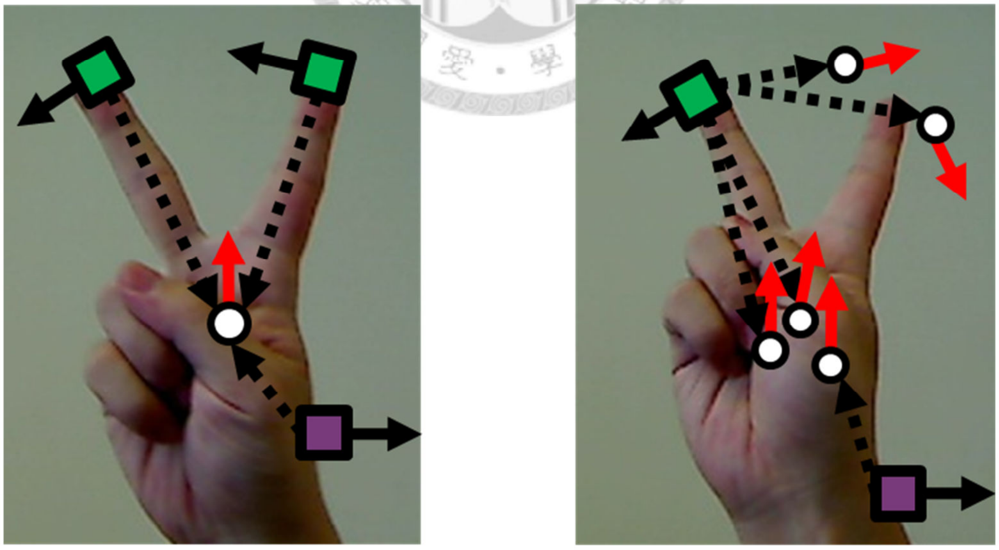
Chapter 3 Hand Posture Recognition with an Implicit Shape Model

In this chapter, we address how to adopt implicit shape model framework to deal with our hand posture recognition in detail. As mentioned in Chapter 1, our recognition approach is inspired by implicit shape model method. For every training visual word, we record the information of all occurrences related to its hand center. The organization of this chapter is given as follows. In Section 3.1, we give an overview of our approach. We explain how to learn the model in Section 3.2.1. Finally, the recognition method is included in Section 3.3.

3.1 Overview of Our Approach

In the beginning, we give a general description of our recognition approach. There are two phase to achieve our recognition: training phase and predicting phase. In training phase, we learn a model for each posture class. For each training image, it is represented by a set of visual words we mentioned in Section 2.3. We collect all visual words' information including cluster id, position, orientation and scale in the image. These visual words should be consistent and meaningful to their object centers. Instead of defining the shape explicitly, every visual word around object center plays a role to define object shape. With this assumption, we let each visual word to cast all possible object center locations from the occurrences situations. That is, we learn the shape model implicitly.

In Figure 3.1, we give an example how this scheme works. In training phase, we assume that our hand images are upright. We record all occurrence vectors from visual words to the hand center. In predicting phase, we predict all posture class separately .Each visual word in the test image cast votes to all possible hand centers via predicting the hand direction and location based on the training experiences in the specific class. Finally, we find the densest voting region from all possible classes to be our prediction that posture class, location and orientation are included.



(a) Training Phase (b) Predicting Phase

Figure 3.1: An example of training phase and predicting phase. Rectangle denotes visual word; Rectangles of the same color indicate the same visual word. Red arrow represents

the posture direction. Dotted vector is a vector from visual word to hand center

3.2 Learning the Implicit Shape Model

In this section, we focus on the learning procedure to build implicit shape model. We based on the voting scheme from [10]. However, the original work does not consider the orientation of features. We slightly modified learning procedure so that is feasible in posture recognition.

3.2.1 Visual Vocabulary Construction

Visual vocabulary is the number of visual words. Before getting visual words, we need detect the local features from images. In our task, the number of total features is usually in the thousands scale, so we use the traditional visual codebook algorithm k-means for generating a compact codebook. Figure 3.2 shows an example of feature patches after clustering. Each row is a codebook entry representing some similar appearances

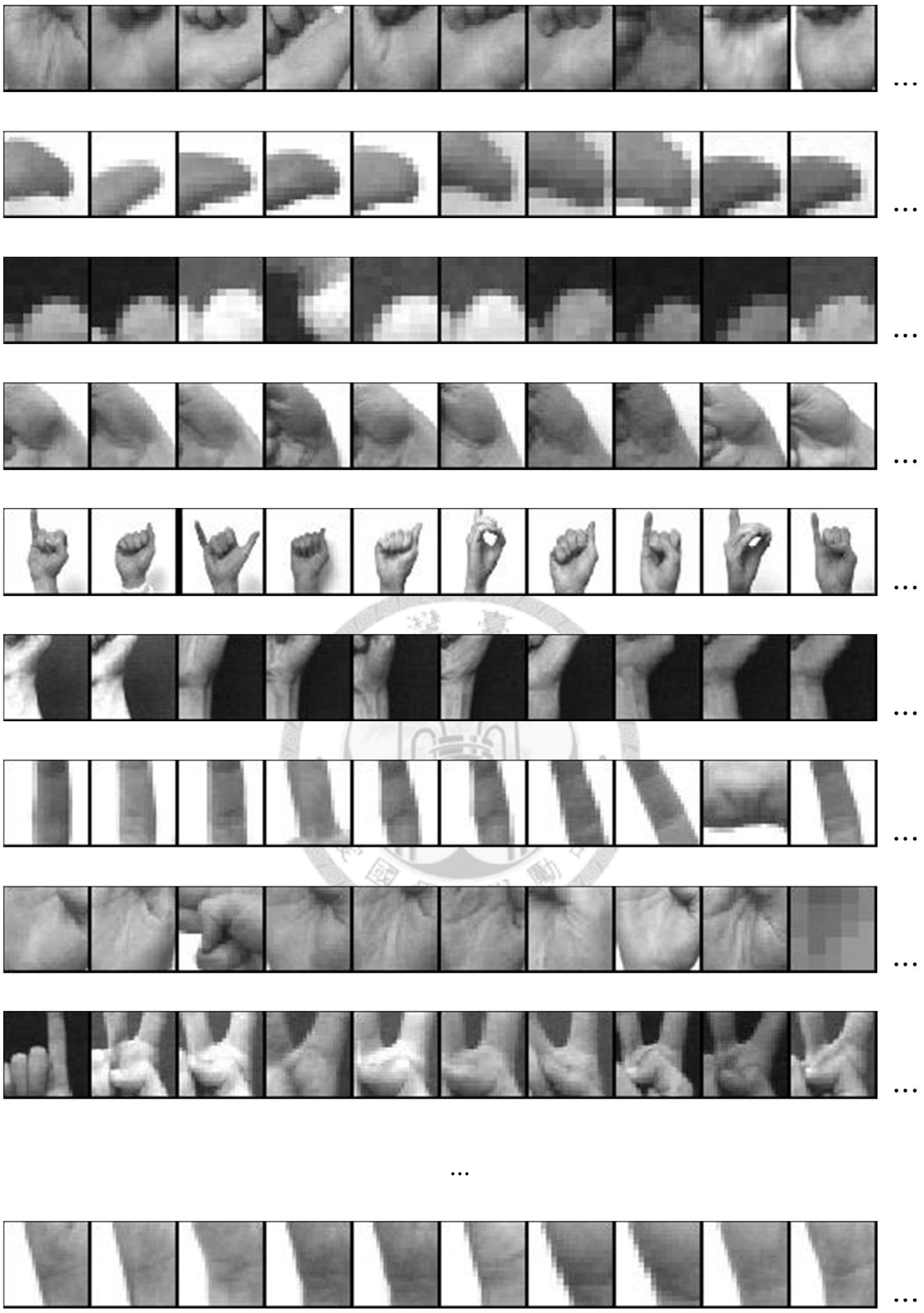


Figure 3.2: An example of visual words, each row is a visual cluster we called a visual word and features belong to a visual word exhibiting similar feature appearances

3.2.2 Hand Center Estimation

In previous work, object center locations are often defined as a manually way [10], [11], [29] .Unfortunately, our posture datasets have no hand center information. Hence, we propose a simple estimation for object center.

We estimate the center of a hand image from the average coordinates of all feature coordinates. In our observations, the estimated locations are consistent for all posture classes if hand images are cropped or in uniform background. The Figure 3.3 and Figure 3.4 show some examples of our estimations.

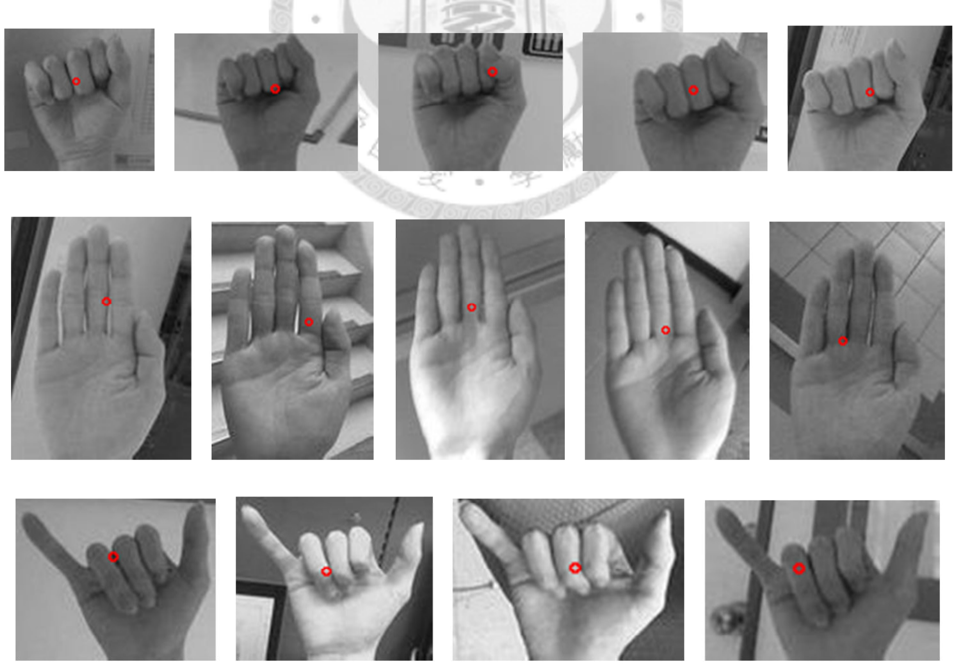


Figure 3.3: Some examples of hand center estimations from the T. C. Liu posture database

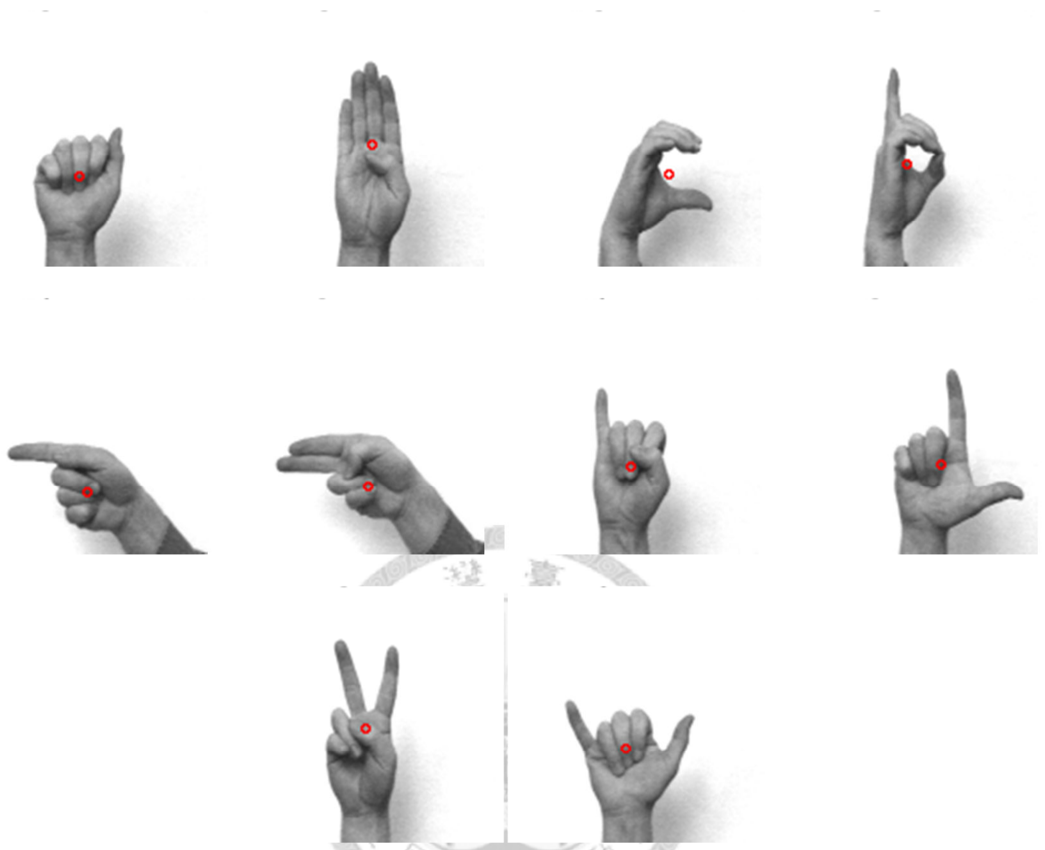


Figure 3.4: Some examples of hand center estimations from the Triesch posture database

3.2.3 Record the Occurrence Vectors

After our training image center coordinates are estimated, we next establish the relationship between hand center and visual words. Given a training dataset as

$$H = \{h = (h_I, h_x, h_y) \mid h \in H\}$$

where h_I is the hand image, (h_x, h_y) is the hand center coordinate estimated from section 3.2.2. A hand image contains a set of local features and the extracted features are matched to the visual cookbook entries. After that, each local feature's appearance is represented by a visual cluster id. For each visual word is characterized as

$$f = (f_p, f_c, f_x, f_y, f_s, f_\theta)$$

where each local feature belongs to posture class f_p and is assigned to a codebook entry f_c. The feature is observed at location (f_x, f_y) and with the scale f_s and orientation f_θ. With a set of features from a hand image, we record the difference between each visual word and its hand center by a three dimensional occurrence vector as

$$\mathbf{o} = \begin{bmatrix} o_x \\ o_y \\ o_\theta \end{bmatrix} = \begin{bmatrix} (h_x - f_x)/f_s \\ (h_y - f_y)/f_s \\ f_\theta \end{bmatrix}$$

The feature scale normalizes occurrence vector in order to predict with different scales in testing images. Leibe et al. [30] proposed this normalization to achieve the categorization in multiple scales. These occurrence vectors are the fundamental elements of our model and they are grouped together by posture class.

It should be note that we assume our training image are upright. Hence we assign the training hand direction to be the upward direction because the training hand directions are all in 90 degrees and reference axis is the x-axis. Figure 3.5 shows an example of hand direction of a training image. This setting lets us to train model with the consistent shape direction.

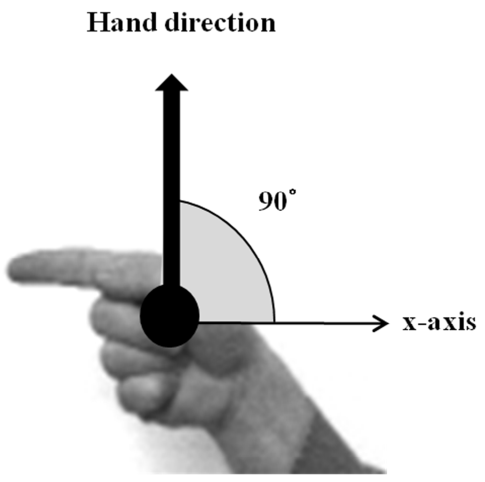


Figure 3.5: Hand direction of training images.

We use inverse document frequency (IDF) to measure how important a visual is in the training data. For each visual word importance, we use IDF weighting inspired from text retrieval problem. It is a good criterion for measuring the distinctiveness. The IDF value of a visual word f_c is computed as

$$\mathrm{IDF}(f_c) = \log \frac{|\{p_i\}|}{|\{p_i | f_c \in p_i\}|}$$

where p_i is the i^{th} posture class.

Ideally, each posture should have at least one distinctive feature or visual word, but these distinctive features are less and not stable to find. Hence, soft weighting assignments are usually adopted like IDF weighting. In general, a visual word is highly distinctive if it appears in less posture class, hence we give higher weighting to this visual word.

In conclusion, we record two parts from training data in learning model step. The former is the occurrence vectors. For difference posture classes, we record them separately as Figure 3.6. The latter is IDF weighting for measuring the distinctiveness. The overall training algorithm is as Algorithm 1.

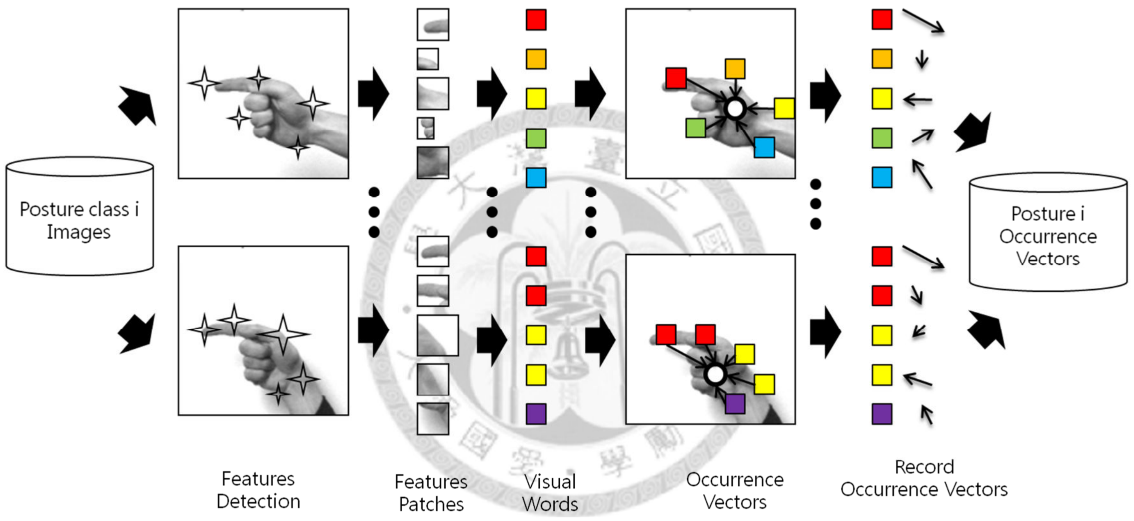


Figure 3.6: The training procedure. For each training image, local features are extracted and clustered to visual words. The relationship between visual words and their center are record as occurrence vectors. Finally, we group the occurrence vectors by each posture classes.

Algorithm 1: Training procedure

```
Input    : training images H
Output : occurrences vectors Occ
//Initialization
For all posture class i do
    For all codebook entries j do
        $Occ(f_p,f_c) = \emptyset$
//Record occurrences vectors
For all $h \in H$ do
    For all feature $f \in$ image $h_I$ do
        $occ_f = \begin{bmatrix} o_x \\ o_y \\ o_\theta \end{bmatrix} = \begin{bmatrix} (h_x - f_x)/f_s \\ (h_y - f_y)/f_s \\ f_\theta \end{bmatrix}$
        $Occ(f_p,f_c) = Occ(f_p,f_c) \cup occ_f$
```

3.3 Posture Recognition

After the ISM model is built, we can recognize a test hand image to the most possible posture class. In this section, we address this recognition in detail.

3.3.1 Hand Center Prediction

Given a feature from a training image and a feature from a testing image, for example, we assume these two features are assigned to the same codebook entry, that is, they have similar appearances. We can represent them by their scale, position and orientation on a standard two-dimensional coordinate plane. In training phase, we can compute the angle between the feature vector and the x-axis. In prediction phase, we compute the transferred hand center by the differences in orientations and scale. In conclusion, we predict the hand centers by all occurrence vectors in training phase.

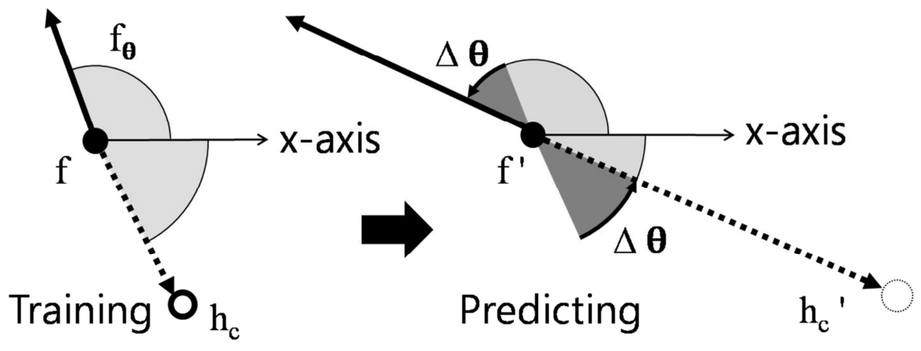


Figure 3.7: Diagram of visual word and hand center

Given a testing visual word $f = (f_p, f_c, f_x, f_y, f_s, f_\theta)$, and the corresponding occurrence vector $o = (o_x, o_y, o_\theta)$ which is looked up by (f_c, f_p), we predict the hand center location by

$$h_{pred,f_p} = \begin{bmatrix} h_x \\ h_y \end{bmatrix} = \begin{bmatrix} f_x + f_s \sqrt{{o_x}^2 + {o_y}^2} \cos\left(\operatorname{atan2}(o_y, o_x)^{1} + (f_\theta - o_\theta)\right) \\ f_y + f_s \sqrt{{o_x}^2 + {o_y}^2} \sin\left(\operatorname{atan2}(o_y, o_x) + (f_\theta - o_\theta)\right) \end{bmatrix}$$

We illustrate how the prediction works in Figure 3.7. It should be noted that the predicting hand centers are usually at least one because there may have many training samples in same visual word and posture class.

3.3.2 Voting Procedure

Unlike the original work of Leibe, we do not use scalar weight voting to predict possible object centers. Here, we propose a new voting scheme based on a vector weight vote to predict the two dimensional hand center vectors. We concurrent consider the feature scale and orientation, so this posture prediction is scale and rotation invariance. These two properties are robust for our posture recognition system.

For each posture class, the visual word cast votes to all possible hand centers. A

---

[1] The two-argument function atan2 is a variation of arctangent function.atan2(x,y) is the angle in radians between positive x-axis and the point (x,y). the angle is positive in upper half-plane, and negative for lower half-plane

vote is represented as a unit two dimensional vector to predict a likely hand location and direction. With this vote scheme, we hope the votes from noisy feature have arbitrarily orientation. Hence, the summing up vectors from the region will have less magnitude and negate each other. In contrast, a high potential hand region should be received consistent hand direction votes. The voting scheme is illustrated as Figure 3.8.

To decide the most likely hand center region, we discretized the two dimensional space into bins. We suppose the bin width is B, image height is H and width is W. Therefore, there are $(\frac{H}{B}) \times (\frac{W}{B})$ bins. The bin width is a parameter of our algorithm to control the precision of hand locations. To small bin width will be sensitive by noisy occurrence vectors. In our experiments, we set B to be 16 pixels. Finally, we accumulate the votes group by bins, and find the largest vector among them to be final our recognition result and an example is illustrated in Figure 3.8.

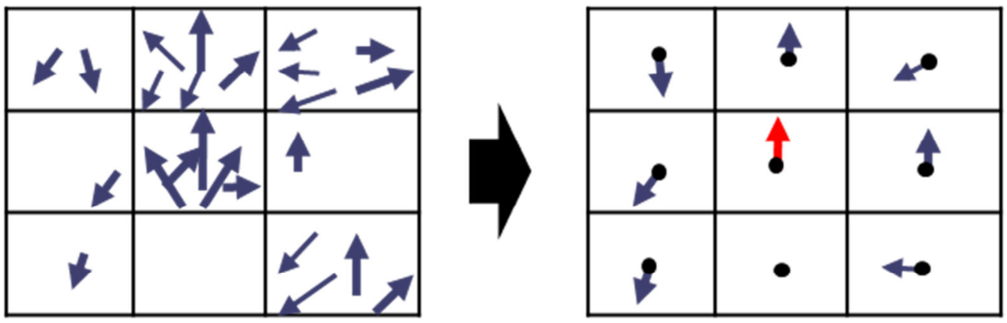
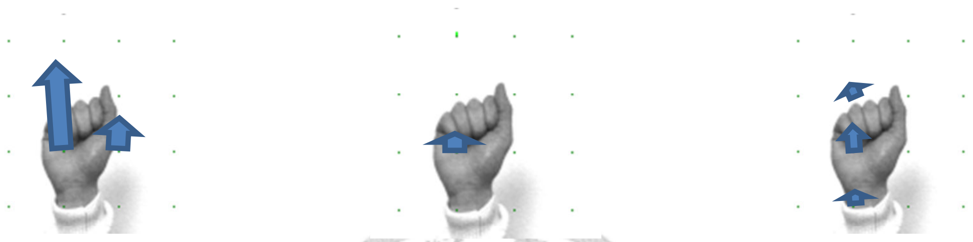


Figure 3.8: Choosing the most possible hand location by summing up all hand predicting vectors in each sub-region and find the largest vectors to be our final prediction

For each posture class, we computed the voting vectors separately as Figure 3.9, and the largest vectors among all posture classes to be our final prediction. The direction of largest vector provides the hand posture direction. This orientation information could be a nice feature to some postures like point posture.



(a) posture fist prediction (b) posture palm prediction (c) posture six prediction

Figure 3.9: Hand posture predicting result (a) ~ (c) are three posture prediction results. Because the magnitude of fist vector is largest, this posture image is more likely to be a fist.

3.3.3 Preprocessing the Images

In practice, we could assume some constrains about hand image inputs. For example, it is nearly impossible for a posture prediction that is downward in general application. In addition, hand image input should have moderate size because we usually do not care for a hand image input if it is too small. Based on these assumptions, we introduce two thresholds to eliminate some unstable features: scale threshold s_t and the difference of orientation threshold $\Delta\theta_t$.

Hand appearance is more uniform than other objects in object recognition problem, for example car, so feature in small scale could arise in any hand part. These features are less informative than feature in large scales. A comparative of small visual word and large visual word is in Figure 3.10.

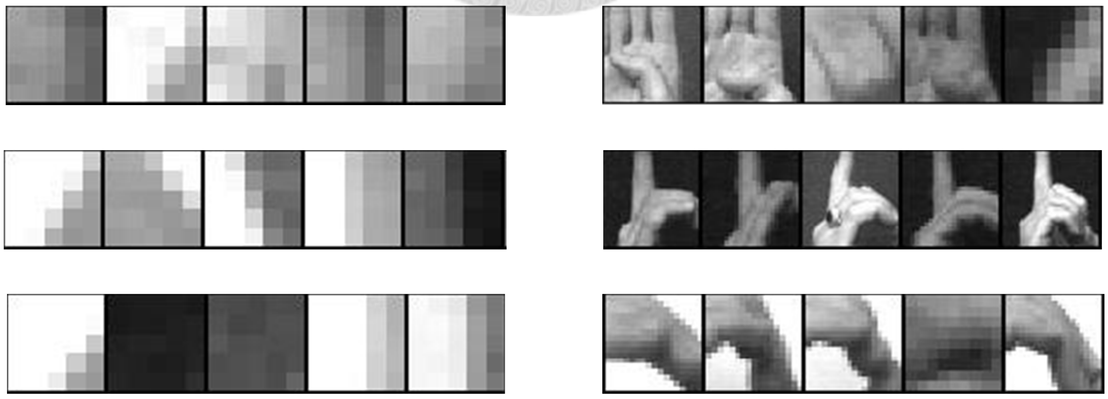

(a) Small visual words, radius < 3 pixels (b) Large visual words, radius > 3 pixels

Figure 3.10: Comparative of large and small visual words

Orientation constrains could take place in two situations. Given a testing visual word, the orientation of this visual word should be consistent to the training visual word‘s orientation. If the difference between training visual word and testing visual word is large, it is likely to be a nosy feature. Here, we simply drop these features.

3.3.4 Summary of Our Recognition Algorithm

In this section, we conclude all concepts into a recognition algorithm that predict a unknown hand image to posture class based on implicit shape model. In Figure 3.11, we give an outline of our recognition algorithm. Given a test image with unknown posture label, we predict each posture class separately. Finally, the strongest response grid from them will be our prediction.

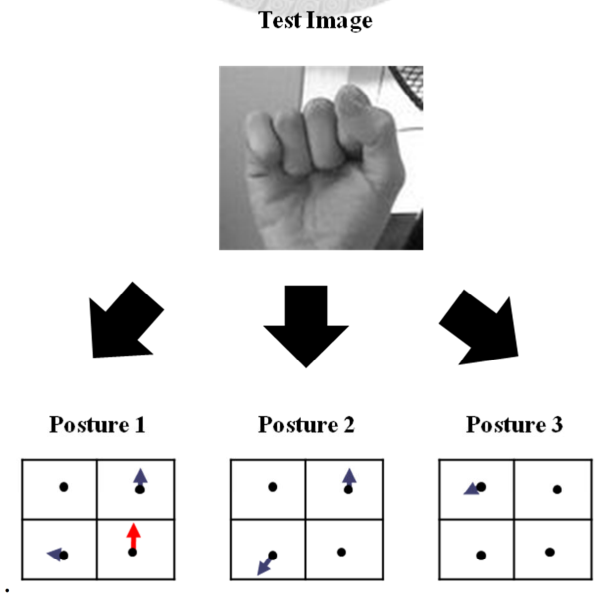


Figure 3.11: Posture recognition overview

There are three parameters in our recognition algorithm: width B, difference of orientation threshold $\Delta\theta_t$ and scale threshold s_t. B is less to improve the localization accuracy but it's more sensitive since there are fewer features to support in the grid. $\Delta\theta_t$ and s_t settings depend on the posture image scope. In general, if we set the threshold more fit the hand posture data. We could boost the accuracy up. The recognition algorithm is as Algorithm 2.

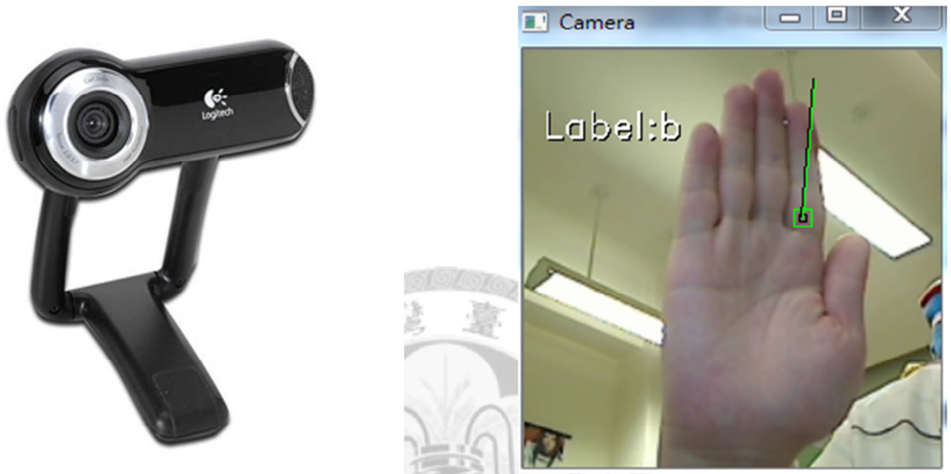
Algorithm 2 ISM recognition algorithm

```
Input: bin width B, difference of orientation threshold $\Delta\theta_t$, sclae threshold $s_t$,
test features $F_{test}$
Output： posture label l and bin location $(x, y)$
For all $p_i$ in all posture classes P do  //Initialization
    For all $(x, y)$in spatial bins do
        $\mathrm{score}(p_i, x, y) = \begin{bmatrix}0\\0\end{bmatrix}$
For all $f = (f_p, f_c, f_x, f_y, f_s, f_\theta)$ in $F_{test}$
    For all $o = (o_x, o_y, o_\theta)$ in database look up by $(f_p, f_c)$
        If $f_s > s_t$ and $|f_\theta - o_\theta| < \Delta\theta_t$ then
            $\mathrm{length} = f_s\sqrt{{o_x}^2 + {o_y}^2}$
            $\theta = \mathrm{arctan2}(f_y, f_x)$
            $c_x = f_x + \cos(\theta + \Delta\theta)\mathrm{length}$
            $c_y = f_y + \sin(\theta + \Delta\theta)\mathrm{length}$
            $\mathrm{predictVector} = \begin{bmatrix}\cos\left(\frac{\pi}{2} + \Delta\theta\right)\mathrm{IDF}(f_c)\\ \sin\left(\frac{\pi}{2} + \Delta\theta\right)\mathrm{IDF}(f_c)\end{bmatrix}$
            $\mathrm{score}\left(f_p, \left\lfloor\frac{c_x}{B}\right\rfloor, \left\lfloor\frac{c_y}{B}\right\rfloor\right) \leftarrow \mathrm{score}\left(f_p, \left\lfloor\frac{c_x}{B}\right\rfloor, \left\lfloor\frac{c_y}{B}\right\rfloor\right) + \mathrm{predictVector}$
    return $\mathrm{argmax}_{p,x,y}\ ||\mathrm{score}(p, x, y)||$
```

Chapter 4 Experimental Results

In this chapter, our prototype system is described in Section 4.1. The evaluation results compared to others are in Section 4.2.

4.1 Hand Posture Recognition System



(a) Logitech Pro 9000 (b) System snapshot

Figure 4.1: (a) Webcam (b) snapshot, the rectangle is center and the line denotes the posture direction

In order to demonstrate our posture recognition approach, we have built a demo program. The program is realized on OpenCV with python wrapper and use the webcam as the input device. The webcam specification is the Logitech Pro 9000. The system snapshot is as Figure 4.1 (b) Using digital still camera to get training image is also suitable, but we prefer the webcam device because the ability of near real-time in demonstration. The system is run on a dual core at 1.66 GHz with 2.5GB of ram.

4.2 Evaluation

Our method is evaluated on two posture databases. First, we introduced the features of posture database in Section 4.2.1. Triesch database's result is in Section 4.2.2.The T. C. Liu database evaluation is in Section 4.2.3. Finally, the running time analysis is included in Section 4.2.4.

4.2.1 Hand Posture Database

In this section, we introduce two posture databases we used in this thesis. The Triesch hand posture database included ten posture classes from American Sign Language. This posture database has been used in many works [16], [31], [32]. The main feature of this dataset is that the varied posture classes in both uniform and complex background. An example of this dataset is shown in Figure 4.2.

Next, T. C. Liu database is provided from Robot Perception and Learning Lab at National Taiwan University. The database is tailored toward robot recognition of three basic postures – fist, palm, and six as commands for directing robots. Each image is cropped but with variance of illumination and background as Figure 4.3. The works [5], [8] based on T. C. Liu database are also taking advantage of the robustness of SIFT features. Finally, Table 4.1 summarizes the features of the two database.

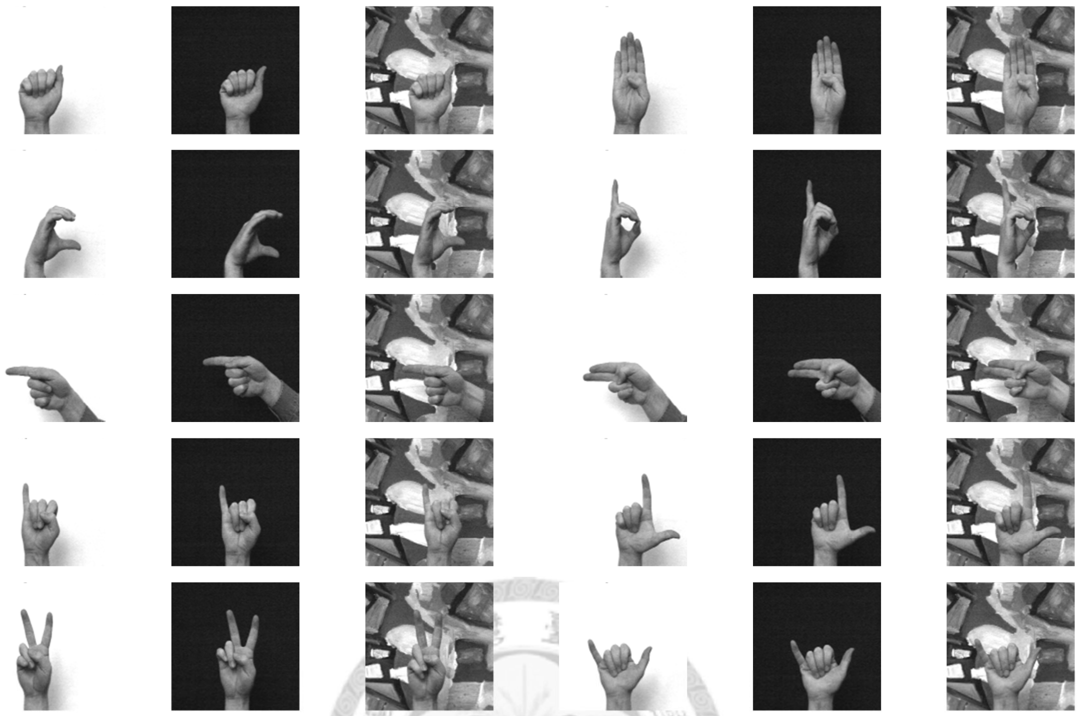
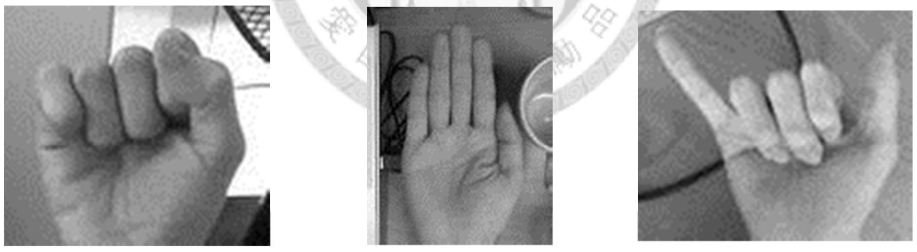


Figure 4.2: Samples of Triesch posture database



(a) Fist (b) Palm (c) Six

Figure 4.3: Samples of T. C. Liu database

Table 4.1: Statistics of two posture databases

	Classes	Background	Images/Class
Triesch Database	10	Uniform/Cluttered	72
T. C. Liu Database	3	Cluttered	300

4.2.2 Results of Triesch Hand Posture Database

We followed the protocol setting from Just el al. [16].There are two protocols used in their work. The protocol 1 we used in this thesis contains only training data in uniform background. The other protocol contains training data in both uniform and complex background. Because the training data in complex could be noisy data in our learning, we adopted the former protocol as our dataset setting. There are two testing images were lost by Triesch [16]. The protocol 1 statistics of the Triesch database is in Table 4.2.

Table 4.2: Statistics of the Triesch hand posture database protocol 1

	Training	Validation	Testing
Number of people	4	4	16
Number of images	80	80	558
Background type	Uniform	Uniform	Uniform/Complex

The principal parameter of our training algorithm is the number of visual vocabulary. In order to determine the visual vocabulary size, we used the validation set to evaluate the trained model from different visual vocabulary sizes. The validation result is in Figure 4.4. The accuracy is best on visual vocabulary equals 2750.

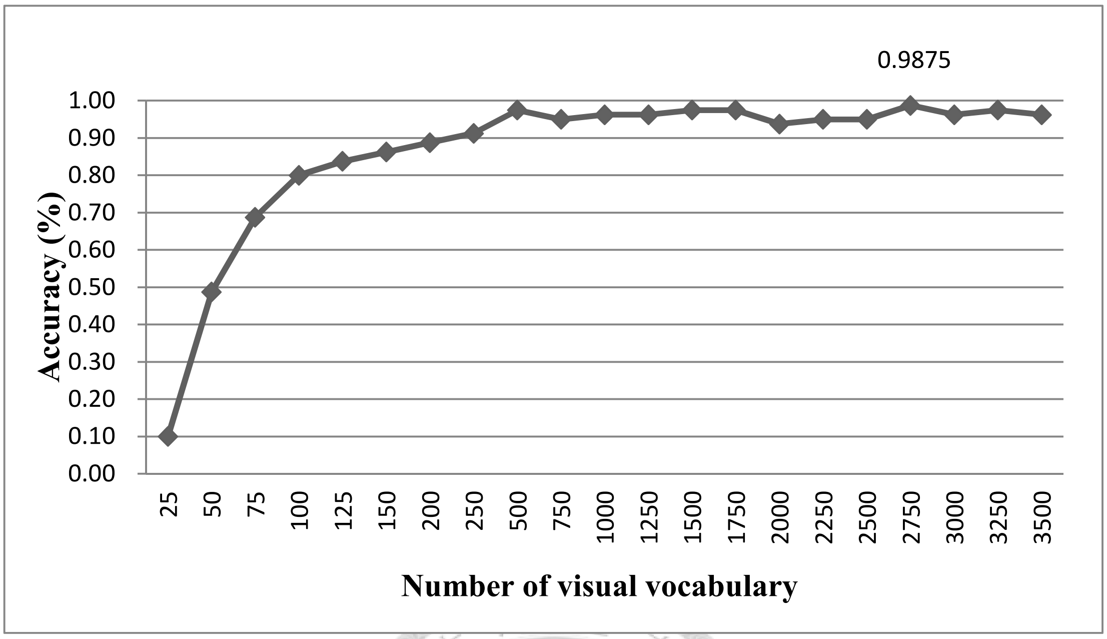


Figure 4.4: Validation Result of the Triesch Database

Table 4.3 shows our recognition result in both uniform background and complex background. We can see that in both backgrounds our approach achieves a better accuracy when compared to a posture classification method based on modified census transform in Just et al. [16]. As expected, recognition accuracy is greatly affected when the background is cluttered.

Table 4.3: A comparison of the recognition results in Just and ours

Accuracy	Uniform Background	Complex Background
Just [16]	89.97 %	64.38 %
Ours	93.10 %	65.27 %

Table 4.4: Recognition results with more training data

Accuracy	**Uniform Background**	**Complex Background**
Training with validation set	95.61%	69.03 %

In addition, we found that the experiment result can be improved further if the images of validation set are also included for training model. The result is in Table 4.4. Ideally, the recognition accuracy can be improved further if we have more training samples. The reason we did this setting is that the validation set is usually also useful information and they are available data. Training model included validation set is better than without it.

Aside from the 1% and 3% improvement in recognition accuracy in uniform and complex background, respectively, our approach improves upon the method proposed by Just et al. in additional number of ways. Table 4.5 summarizes this. Despite being able to recognize postures with good accuracy, their method was not able to provide a localization and orientation of the hand while the inherent property within our approach allows us to easily gather such information.

In addition, we observe from the dataset that assumptions are made in terms of hand position and orientation that requires the hand to be upright and centered in order for recognition to occur. Like the method proposed by Just et al., they need to scan all sub-images with varied scales to detect a possible hand posture. It is called sliding window approach. In our approach however, the algorithm exhibits scale and rotational invariance through the use of SIFT features, which are inherently scale invariant, and the occurrence vectors, shown to be rotational invariant in previous sections. By using SIFT features also allows us to detect the hand with robustness since we do not place such a constraint on the placement of the hand in the center of image.

Table 4.5: Comparison of metrics between of our method and Just [16]

Comparison	**Orientation**	**Rotation/Scale Invariance**	**Centering**
Just et al. [16]	No	No	No
Ours	Yes	Yes	Yes

4.2.3 Evaluation of T. C. Liu Posture Database

For each posture class, we have 300 images samples. We let 200 images for training and 100 images for testing. The size of each image is around 100 x 100 pixels. In order to determine the number of visual vocabulary, we ran 5-fold cross validation on training set. The 5-fold cross validation result is in Figure 4.5. Hence, we set K to 1250 in recognition.

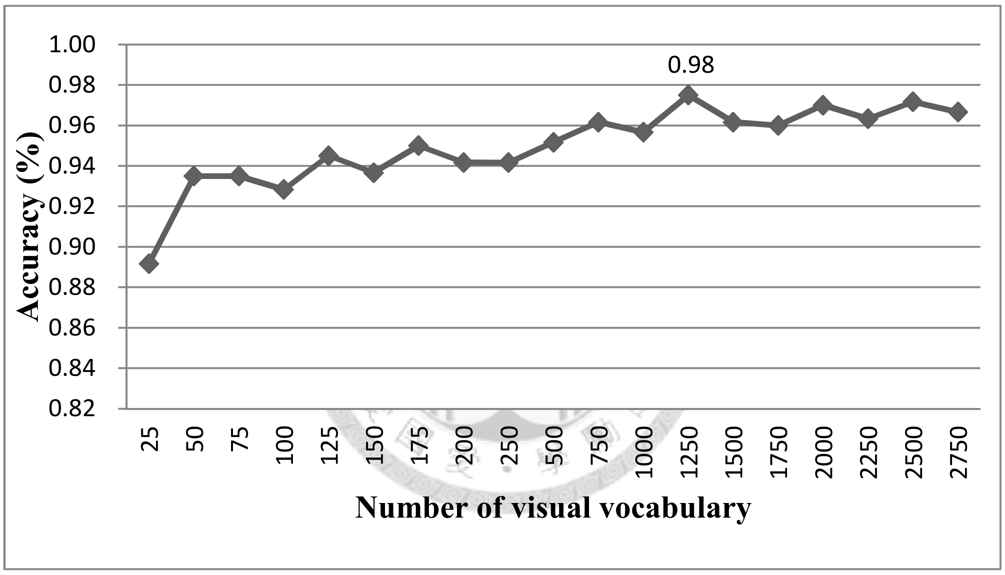


Figure 4.5: 5-fold cross validation result of Liu Database

We tested 300 test images, with each posture class containing 100 images each. The results show a 98.67% recognition rate for the posture classes. It should be noted that the algorithm allows for π/8 rotation of the hand. If the system detects that the feature in the test image has a difference of orientation greater than the threshold, so it could possibly be a noisy feature. Here, we simply drop it. Table 4.7 shows the dataset being run using different inherent parameters, namely, allowing for a hand rotation of 360° in order to show rotational invariance.

Table 4.6: The confusion matrix of our result at B=32, $\Delta\theta_t = 0.125\pi$, $s_t = 3$

Ours	Fist	Palm	Six	Total	Accuracy
Fist	99	0	1	100	99%
Palm	0	100	0	100	100%
Six	1	2	97	100	97%
Total	100	102	98	300	98.67%

Table 4.7: The confusion matrix of our result at B=32, $\Delta\theta_t = 2\pi$, $s_t = 0$

Ours	Fist	Palm	Six	Total	Accuracy
Fist	99	0	1	100	99%
Palm	0	98	2	100	98%
Six	4	9	87	100	87%
Total	103	107	90	300	94.67%

Finally, we want to compare our results with T. C. Liu's method, which is similar to ours. Table VII provide a summary of this result. We have a 98.67% accuracy rate when compared to his method, with only an accuracy value of 87.6%.

Table 4.8: Recognition result comparison with ours and T. C. Liu [10]

Method	**Accuracy**
Ours	98.67%
T. C. Liu's one-against-other [10]	87.6%
T. C. Liu's multi-class approach [10]	86.7%

4.2.4 Running Time Analysis

In this section, we analyze the running time and there are training phase and test phase in our method. In training phase, it takes time to extract features, generate visual codebook and build the ISM model. On the other hand, it takes time to extract feature, project feature to visual word and run the ISM model when a test image is given.

The summary of time required for training an ISM model is as model. The most of time is on feature extraction and visual codebook generation, so building an ism model is very fast. The feature extraction is based on SIFT algorithm and an alternative is the well-known SURF algorithm [33]. In addition, some works [9] prefer vocabulary tree to generate visual codebook for speed purpose. The time list as table was computed from the average of ten executions.

Table 4.9: Summary of average time in training an ISM model

Training Time	Triesch database protocol 1	T. C. Liu database
Feature extraction	12.01 secs	103.9 secs
Codebook generation	46.2 secs	173.8 secs
ISM building	0.28 secs	1.296 secs
Total	58.5 secs	279.0 secs

There is often an interest in the recognition speed of any posture recognition algorithms in order to assess its potential as a practical application. Here, we will evaluate the execution time of the algorithm on the two datasets. Table 4.10 shows a summary of the result.

Table 4.10: Summary of average recognition time

Recognition time / per image	Triesch database	T. C. Liu database
Feature extraction	276 ms	179.3 ms
Projection	848 ms	373 ms
Classification	23 ms	22 ms
Total	1,147 ms	574 ms

The average execution time of our algorithm in recognition is 1.147 s for an image from Triesch dataset and 0.574 s for an image from T. C. Liu dataset. It would be an acceptable value for any static hand posture application that requires a performance similar to near real-time due to the fact that it is within a tolerable threshold. It turns out that the bulk of the operation is in extracting and projecting SIFT features, which takes up 97% of the total execution time. This observation leads to an idea for further improvement by using a more efficient way of extracting local features from an image. There are a number of improvements over SIFT, e.g. SURF motioned above, such that we can reduce execution time. In addition, we can use a fast visual codebook algorithm to speed up the projection time as we described in Section 2.3.2.The project time depends on the number of visual vocabulary since we need to project to the nearest visual word.

Chapter 5 Conclusion and Future Work

How to use implicit shape model to recognize hand posture images was addressed in this thesis, and the principal findings suggested that using implicit shape model to realize hand posture recognition and the improved accuracy considerably. We extended the implicit shape model by considering the orientation of features and a novel method for concurrent posture localization and recognition.

One of the advantages of our system is that our training time is fast. In our learning method, each posture model is learnt from a set of hand images in the same posture class. In some applications like HCI, users may prefer to use the intuitive hand postures they like. Hence, we could not learn the model previously. Because our training procedure is fast, the posture models can be learnt when the posture system is initializing. We should examine not only the recognition but also the localization. In this thesis, we proposed an implicit shape model approach to recognize hand postures. However, the location information of hand movements is an important issue of dynamic hand gesture application. We discretized the two dimensional space into bins and each bin is a hand posture candidate. This approach is simple but not accurate. Some authors [10], [29] applied mean shift algorithm to seek the hypothesis. Because of time constraints, we have not investigated that yet. It could be a better approach to detect multiple hand instances in an image.

In practice, SIFT and SURF are the two most common local feature algorithms. Because hands are articulated, the scale and rotation invariance properties from algorithms are very important for our recognition task. We adopt SIFT for our local feature algorithm since it is more distinctive than SURF in most of our tasks, although SIFT is much slower than SURF. However, users usually don't give multiple hand gesture command over short periods of time in pure static posture application. For real-time systems such as dynamic hand gesture system, our framework can readily use SURF indeed.

Future research directions could be conducted on dynamic hand gesture recognition based on static posture recognition result for more varied hand commands including both hand poses and movements like waving goodbye. In this paper, we only consider the recognition problem. That is, we force to decide a posture class even if there is no hand in the input image. In order to solve this hand detection problem, the simplest solution is to define a threshold to reject hypotheses with low score. However, how to decide thresholds in different circumstances is non-intuitive. In addition, some other cues might be beneficial to improve accuracy. For example, using a skin color model predicts the possibility of pixels to be a part of hand.

REFERENCES

[1] P. Mistry, P. Maes, and L. Chang, “WUW - wear Ur world: a wearable gestural interface,” *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, Boston, MA, USA: ACM, 2009, pp. 4111-4116.

[2] R. Goshorn, D. Goshorn, and M. Kölsch, “The Enhancement of Low-Level Classifications for Ambient Assisted Living,” *Behavioral Monitoring and Interpretation Workshop at the German Conference on Aritificial Intelligence, KI*, 2008.

[3] Q. Chen, N. Georganas, and E. Petriu, “Hand gesture recognition using Haar-like features and a stochastic context-free grammar,” *IEEE Transactions on Instrumentation and Measurement*, vol. 57, 2008, pp. 1562–1571.

[4] Y. Chen and K. Tseng, “Multiple-angle hand gesture recognition by fusing svm classifiers,” *IEEE International Conference on Automation Science and Engineering, 2007. CASE 2007*, 2007, pp. 527–530.

[5] T. Liu, K. Wang, A. Tsai, and C. Wang, “Hand posture recognition using hidden conditional random fields,” *The IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore*, 2009.

[6] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos,” *Ninth IEEE international conference on computer vision, 2003. Proceedings*, 2003, pp. 1470–1477.

[7] H. Zhou, D. Lin, and T. Huang, “Static hand posture recognition based on okapi-chamfer matching,” *Real-Time Vision for Human-Computer Interaction*, 2005, pp. 85–101.

[8] C. Wang and K. Wang, “Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction,” *Recent Progress in Robotics: Viable Robotic Service to Human*, 2009, pp. 317-329.

[9] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, 2161–2168.

[10] B. Leibe, A. Leonardis, and B. Schiele, “Robust object detection with interleaved categorization and segmentation,” *International Journal of Computer Vision*, vol. 77, 2008, pp. 259–289.

[11] B. Leibe, E. Seemann, and B. Schiele, “Pedestrian detection in crowded scenes,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2005.

[12] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, May. 2004, pp. 137-154.

[13] M. Kolsch and M. Turk, "Robust hand detection," *Proc. of the Sixth IEEE Int. Conf. on Automatic Face and Gesture Recognition (FG)*, 2004.

[14] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[15] B. Froba and A. Ernst, "Face detection with the modified census transform," *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, 2004, pp. 91–96.

[16] A. Just, Y. Rodriguez, and S. Marcel, "Hand posture classification and recognition using the modified census transform," *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, 2006, pp. 351–356.

[17] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, Nov. 2004, pp. 91-110.

[18] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, 524–531.

[19] E. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," *Face and Gesture Recognition*, 2004, pp. 889–894.

[20] J. Willamowski, D. Arregui, G. Csurka, C.R. Dance, and L. Fan, "Categorizing Nine Visual Classes Using Local Appearance Descriptors," *ICPR Workshop on Learning for Adaptable Visual Systems*, 2004.

[21] C. Wallraven, B. Caputo, and A. Graf, "Recognition with local features: the kernel recipe," *Ninth IEEE International Conference on Computer Vision*, 2003, pp. 257–264.

[22] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*, 2003.

[23] M. Weber, M. Welling, and P. Perona, "Unsupervised learning of models for recognition," *Computer Vision-ECCV 2000*, 2000, pp. 18–32.

[24] J. van Gemert, C. Snoek, C. Veenman, A. Smeulders, and J. Geusebroek, "Comparing compact codebooks for visual categorization," *Computer Vision and Image Understanding*, vol. 114, Apr. 2010, pp. 450-462.

[25] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, 2006, pp. 3–42.

[26] K. Grauman and T. Darrell, “The pyramid match kernel: Discriminative classification with sets of image features,” *Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005*, 2005.

[27] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[28] P. Tirilly, V. Claveau, and P. Gros, “Language modeling for bag-of-visual words image categorization,” *Proceedings of the 2008 international conference on Content-based image and video retrieval*, 2008, pp. 249–258.

[29] A. Oikonomopoulos, I. Patras, and M. Pantic, “An implicit spatiotemporal shape model for human activity localization and recognition,” *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2009.*

[30] B. Leibe and B. Schiele, “Scale-invariant object categorization using a scale-adaptive mean-shift search,” *Pattern Recognition*, 2004, pp. 145–153.

[31] J. Triesch and C. von der Malsburg, “Classification of hand postures against complex backgrounds using elastic graph matching,” *Image and Vision Computing*, vol. 20, 2002, pp. 937–943.

[32] J. Triesch and C.V. Malsburg, “Robust classification of hand postures against

complex backgrounds," *International Conference On Automatic Face and Gesture Recognition, Killington*, 1996.

[33] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, 2006, pp. 404–417.