Visual interpretation of hand postures for human-machine interaction

van Toi Nguyen

UNIVERSITÉ DE LA ROCHELLE
ÉCOLE DOCTORALE S2IM
LABORATOIRE L3i

INSTITUT POLYTECHNIQUE DE HANOI
INSTITUT DE RECHERCHE INTERNATIONAL MICA

THÈSE présentée par :

Van-Toi NGUYEN

soutenue le : 15 Décembre 2015
pour obtenir le grade de : **Docteur de l'université de La Rochelle**
Discipline : **Informatique et Applications**

Interprétation visuelle de gestes pour l'interaction homme-machine

---

JURY :

Jean-Philippe DOMENGER	Professeur, Université de La Rochelle, Président du jury
Patrick LAMBERT	Professeur, Université Savoie Mont Blanc, Rapporteur
Duc-Dung NGUYEN	Docteur, Directeur adjoint IOIT - Vietnam Academy of Science and Technology (VAST), Rapporteur
Rémy MULLOT	Professeur, Université de La Rochelle, Directeur de thèse
Vincent COURBOULAY	Maître de Conférence HDR, Université de La Rochelle, Encadrant scientifique
Thi-Lan LE	Maître de Conférence, Institut de recherche international MICA - Institut Polytechnique de Hanoi, Co-Directrice de thèse
Thi-Thanh-Hai TRAN	Maître de Conférence, Institut de recherche international MICA - Institut Polytechnique de Hanoi, Encadrant scientifique
Eric CASTELLI	Professeur, Institut de recherche international MICA - Institut Polytechnique de Hanoi, Encadrant scientifique

Acknowledgements

Van-Toi NGUYEN

Interprétation visuelle de gestes pour l'interaction homme-machine

Aujourd'hui, les utilisateurs souhaitent interagir plus naturellement avec les systèmes numériques. L'une des modalités de communication la plus naturelle pour l'homme est le geste de la main. Parmi les différentes approches que nous pouvons trouver dans la littérature, celle basée sur la vision est étudiée par de nombreux chercheurs car elle ne demande pas de porter de dispositif complémentaire. Pour que la machine puisse comprendre les gestes à partir des images RGB, la reconnaissance automatique de ces gestes est l'un des problèmes clés. Cependant, cette approche présente encore de multiples défis tels que le changement de point de vue, les différences d'éclairage, les problèmes de complexité ou de changement d'environnement. Cette thèse propose un système de reconnaissance de gestes statiques qui se compose de deux phases : la détection et la reconnaissance du geste lui-même. Dans l'étape de détection, nous utilisons un processus de détection d'objets de Viola Jones avec une caractérisation basée sur des caractéristiques internes d'Haar-like et un classifieur en cascade AdaBoost. Pour éviter l'influence du fond, nous avons introduit de nouvelles caractéristiques internes d'Haar-like. Ceci augmente de façon significative le taux de détection de la main par rapport à l'algorithme original. Pour la reconnaissance du geste, nous avons proposé une représentation de la main basée sur un noyau descripteur KDES (Kernel Descriptor) très efficace pour la classification d'objets. Cependant, ce descripteur n'est pas robuste au changement d'échelle et n'est pas invariant à l'orientation. Nous avons alors proposé trois améliorations pour surmonter ces problèmes: i) une normalisation de caractéristiques au niveau pixel pour qu'elles soient invariantes à la rotation ; ii) une génération adaptive de caractéristiques afin qu'elles soient robustes au changement d'échelle ; iii) une construction spatiale spécifique à la structure de la main au niveau image. Sur la base de ces améliorations, la méthode proposée obtient de meilleurs résultats par rapport au KDES initial et aux descripteurs existants. L'intégration de ces deux méthodes dans une application montre en situation réelle l'efficacité, l'utilité et la faisabilité de déployer un tel système pour l'interaction homme-robot utilisant les gestes de la main.

Mots clés : Vision par ordinateur, apprentissage automatique, reconnaissance de posture de la main, visualisation basée sur l'interaction homme-machine, détection de la main, caractéristiques Haar-like internes, AdaBoost, Cascade de classifieurs, noyaux descripteurs, machine à vecteurs de support (SVM).

Van-Toi NGUYEN

Visual interpretation of hand postures for human-machine interaction

Nowadays, people want to interact with machines more naturally. One of the powerful communication channels is hand gesture. Vision-based approach has involved many researchers because this approach does not require any extra device. One of the key problems we need to resolve is hand posture recognition on RGB images because it can be used directly or integrated into a multi-cues hand gesture recognition. The main challenges of this problem are illumination differences, cluttered background, background changes, high intra-class variation, and high inter-class similarity.

This thesis proposes a hand posture recognition system consists two phases that are hand detection and hand posture recognition.

In hand detection step, we employed Viola-Jones detector with proposed concept Internal Haar-like feature. The proposed hand detection works in real-time within frames captured from real complex environments and avoids unexpected effects of background. The proposed detector outperforms original Viola-Jones detector using traditional Haar-like feature.

In hand posture recognition step, we proposed a new hand representation based on a good generic descriptor that is kernel descriptor (KDES). When applying KDES into hand posture recognition, we proposed three improvements to make it more robust that are adaptive patch, normalization of gradient orientation in patches, and hand pyramid structure. The improvements make KDES invariant to scale change, patch-level feature invariant to rotation, and final hand representation suitable to hand structure. Based on these improvements, the proposed method obtains better results than original KDES and a state of the art method.

Keywords : Computer vision, Machine learning, Hand posture recognition, Visual based Human-machine interaction, Hand detection, Internal Haar-like feature, AdaBoost, Cascade of classifiers, Kernel descriptor, Support vector machine.

Laboratoire L3i

Pôle Sciences & Technologie,
Avenue Michel Crépeau,
17042 La Rochelle Cedex 1 - France

Institut de recherche international MICA

Bâtiment B1,
Institut Polytechnique de Hanoi (IPH),
n° 1, rue Dai Co Viet, Ha Noi, Viet Nam

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Objective

The aim of this work is to develop a good hand posture recognition method within video frames for human-machine interaction using consumable 2D camera. Despite the huge effort of researchers over decades, this objective has remained, especially in the real environment, unreached. Although reasonably successful attempts have been made for certain constraints, such as uniform background, no satisfactory methods exist that work with actual conditions and a large number of hand posture classes. There are two important tasks that are to detect the hand in the frame and classify it into a predefined class of hand posture.

This thesis will develop a method including two phases that are hand detection and hand posture recognition. The hand detection method will be able to work in real time within frames captured from real complex environments. The hand posture recognition method is then required to work with hand images containing complex background that are the results of the detection step. The two-phase integrated system will be used to build a human-machine interaction system.

1.2 Motivation

Hand gestures are a powerful communication channel among human. In fact, hand gestures play a significant role in information transfer in our everyday life. Especially, sign languages are used informally by dumb people. Gesturing is a natural way of interaction.

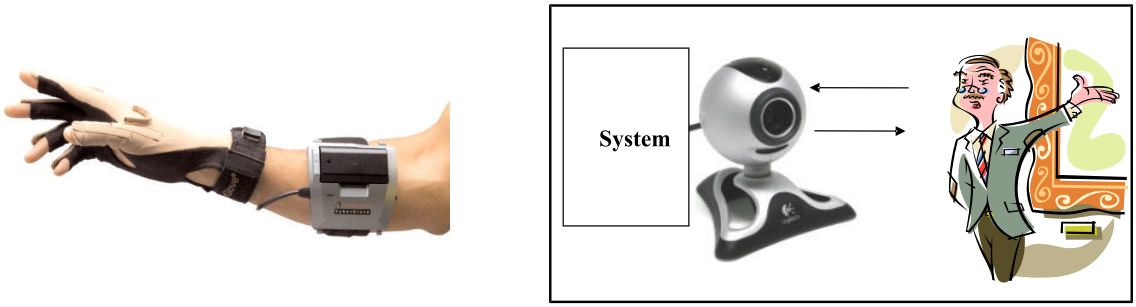
Once sensor-based machines can understand the meaning of human hand gestures, the future communication between people and machines will be more natural and intuitive. In other word, these sensor-based human-machine interaction systems will allow people to interact with machines more closely resembling human-human communication.

Hand gestures can be used in a broad range of applications. Such applications are conventional human-computer interactions, objects manipulation in Virtual Environments, exchanging information with other people in a virtual space, guiding some robots to perform certain tasks in a hostile environment.

Over the recent three decades, many researchers have studied and developed hand-gesture based human-machine interfaces. We would like to contribute a study to the progress in this field.

There are two main approaches for hand gesture interaction that are "Data-Glove based" and "Vision Based" approaches. The Data-Glove based approach uses sensor devices to capture hand and finger motions (Fig. 1.1(a)). The extra sensors facilitate to collect easily hand configuration and movement. Nevertheless, the sensor devices are quite expensive and make users feel cumbersome and inconvenient.

In contrast, the Vision Based methods require only a camera [29] (Fig. 1.1(b)). With this approach, the interaction between humans and computers is natural without the use of any extra devices; as a result, it makes the interface more convenient. The system will capture the video stream from the camera as input then use vision-based hand gesture recognition techniques to recognize hand gestures. Besides, vision-based hand gesture recognition systems can provide an intuitive communication channel for human-machine interaction. For this reason, we decide to follow this approach.



(a) (b)

Figure 1.1 – Two main approaches for hand gesture interaction. (a) The Data-Glove based approach (An example of the Data Glove: The CyberGlove from the Immersion Corporation [1]). (b) Vision Based approach.

We normally identify two types of hand gesture: static gesture (hand posture) and dynamic gesture [16,48]. A hand posture is a specific configuration of hand with a static pose and its current location without any involved movements. A hand gesture is a sequence of hand postures connected by continuous hand or finger movements over a short time span. We focus on hand posture recognition because hand postures can directly replace some remote control devices, using a one-to-one correspondence between hand postures

and commands. Moreover, the identification of key hand postures is useful for dynamic hand gesture recognition.

Nowadays, depth sensors such as Microsoft Kinect has become a common device in many areas including computer vision, robotics human interaction, augmented reality. These devices provide depth information of the scene. Depth information is very useful for hand detection and recognition. However, we are still interested in improving methods working on color information because of three reasons: (1) In many application systems, depth information is not satisfied because of the particular designs of the systems and the limitation of the measurable ranges of the depth sensors; (2) The depth information is quite noisy; (3) Most of the systems use depth information combining with color cues even speech. Therefore, good methods based on color information are useful for multiple cue interaction systems.

Most of hand posture recognition systems consist of two phases, hand detection and hand posture recognition. This framework is reasonable because hand covers a small region of the image and it is inefficient to use the whole image as the input of hand posture recognition method. A few hand posture recognition methods work on entire input image without hand detection step. However, the assumption of these methods is that hand covers a significant region of the input image.

In summary, the above analysis motivates us to do research to develop a hand posture recognition system for human-machine interaction that works well on color frames sequences. This system will have two main steps that are hand detection and hand posture recognition.

1.3 Context, constraints, and challenges

We can easily imagine about many interesting applications of hand posture recognition in controlling instruments in house such as televisions and communicating with some kinds of service robots such as information consultation robot in libraries. In these systems, instruments are often immobile and work in indoor environments. In case of moving robots, they also often stand still while interacting with the user. The vision-based systems can exploit hand postures for the command in smart environments without any remote control unit. The camera is installed in a strategic position to obtain good performance as well as to make users feel comfortable. To develop these kinds of application system, we define a convenient installation as well as the constraints for study on hand posture recognition as the following:

- Indoor Environment: The users interact with machines through a camera in the indoor environment.

- Fixed Camera: The camera is immobile while interacting.
- Face-to-face stand: The user stands in front of the camera that is installed on the machine. The user naturally raises one hand to control the machine using hand postures. When the user raises their hand, the camera and the hand are approximately at the same height so as the camera can perceive the hand clearly.

In this context, we have to cope with the following issues:

- Illumination difference. The value of a pixel in the image will change when the lighting of the environment changes. The pixel values could be shifted or scaled. When the positions of the light sources change, the pixel values will change according to a non-linear transformation and/or be complicated by shadows falling on the hand.
- Background clutter and change. In the real environment, in most cases, the background is clutter. Moreover, the background contains many other objects with similar skin color. Thus, it is very difficult to clearly detect and segment hand from the background. In addition, background often changes because of changing lighting condition and the other objects' movement.
- Scale change. The change in distance between the user and the camera depends on each user and working section. The distance changes make the scale change of hand images. Moreover, sizes of hand are also different from user to user.
- High intra-class variation and high inter-class similarity. The hand is a highly deformable object; there is a considerable number of mutually similar hand postures as well as there are high varieties of instances of each hand posture. Two instances of a hand posture class could be different because of different angles of the hand and the variance of the fingers' distortion. For the same reasons, a hand posture instance could be different from its class and similar to other class concurrently. In Fig.1.2(a-d), (a) and (b) are images of Posture #12 in our dataset; (c) and (d) are images of Posture #6. However, we can see that the similarity between (b) and (c) is higher than between (a) and (b) as well as between (c) and (d) concerning the positions of the fingers in the hand images. Moreover, some hand posture class are very similar. For example Fig.1.2(i,j,k) are three different hand postures however they are quite similar.
- Computational time and user-independence. Applications using hand posture generally require real-time and user-independent recognition.

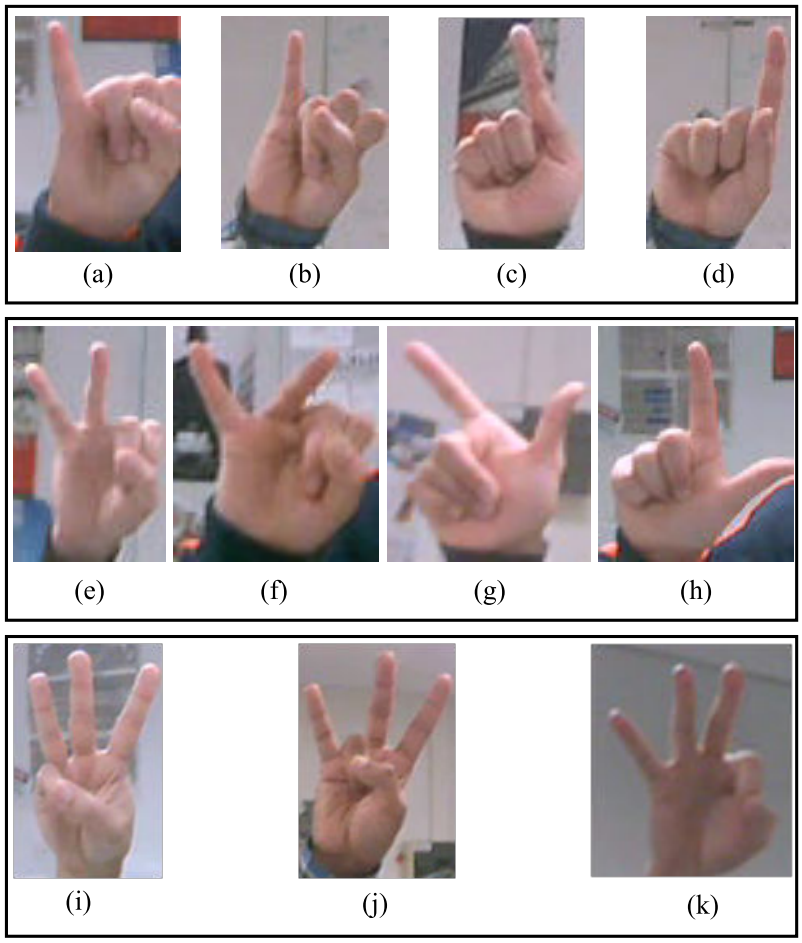


Figure 1.2 – Some examples of Intra-class variation and Inter-class similarity

Our work in this thesis aims at addressing the aforementioned issues for hand posture recognition.

1.4 Contributions

In this thesis, we propose a framework of hand posture recognition system consisting of two main phases: hand detection and hand posture recognition. We then propose a strategy to integrate the proposed methods into a real application. In the following, we present more detail of our contributions in terms of hand detection, hand posture recognition, and application:

- ***Hand detection***: Our hand detection method is inspired by Viola-Jones method that uses Haar-like features and Cascade of AdaBoost classifiers. As a consequence, the proposed hand detector has the properties of Viola-Jones detector such as fast computation time. However, when applying Viola-Jones detector into hand detection, we meet a problem, that is the unexpected effect of the background. To tackle this

issue, we introduce a new concept of *internal Haar-like feature*. It is shown that internal Haar-like features outperform Haar-like features.

- ***Hand posture recognition***: Concerning hand posture recognition, we propose a new hand posture representation based on kernel descriptor (KDES) [5] with the following properties:
  - *Invariant to rotations at patch level*: At patch level, the original KDES computes gradient based features without considering the orientation. Therefore, the generated features will not be invariant to rotation. We propose to compute the dominant orientation of patch and normalize all gradient vectors in the patch to this orientation. By this way, patch-level features will be invariant to rotation.
  - *Robust to scale change*: The original KDES computes features on patches of fixed size. At two different scales, the number of patches to be considered and the corresponding patch descriptions will be different. We propose a strategy to generate patches with adaptive size. This strategy makes the number of patches remain unchanged and patch description robust. As a result, the image-level feature is invariant to scale change.
  - *Suitable specific structure of the hand*: At the image level, the original KDES organizes a spatial pyramid structure of patches to build the final description of the image. However, we observe that hand is an object with a particular structure. We hence design a new pyramid structure that reflects better the structure of the hand.
- ***Application***: To illustrate the applicability of the proposed methods, we propose a deployment of the hand gesture recognition on a service robot in a library environment. A strategy for efficient recognition over multiple temporal frames is proposed, despite the recognition system taking still images as input. The experimental results in a simulated library context show good recognition performance for most of the tasks.

1.5 Outline of the thesis

The structure of the thesis is as follows:

- In **Chapter 2**, we review existing works in the field of hand detection and hand posture recognition. We classify existing works according to the feature extraction and hand representation aspects.

- **Chapter 3** presents proposed hand detection method based on Viola-Jones detector with a new concept that is internal Haar-like feature. In this chapter, we also describe our own dataset (MICA-L3i dataset) which is used to evaluate the performance of hand detection method as well as hand posture recognition method. The experimental results on hand detection are presented and discussed at the end of this chapter.

- In **Chapter 4**, our new hand representation based on kernel descriptor is presented. The hand posture recognition method is described step by step. In each step, we point out our improvements compared with original KDES.

- In **Chapter 5**, based on our works for hand detection and hand posture recognition, we have built a fully automatic hand posture recognition system and applied it in a human-robot interaction application: service robot in library. The goal of this application is to demonstrate that we can apply proposed methods of hand detection and hand posture recognition to build a human-machine interaction system.

- In **Chapter 6**, we draw the conclusions and discuss the future works.

Chapter 2

Literature review

2.1 Introduction

In this chapter, we will present a survey of hand posture recognition system. As presented in Chapter 1, our work aims to develop a hand posture recognition system for human-machine interaction in the real indoor environment. The framework of our expected system consists of two phases that are hand detection and hand posture recognition. Therefore, we will analyze the state of the art works with regard to two problems: hand detection and hand posture recognition.

In general, the hand gesture recognition system consists of hand detection, tracking and recognition steps. However, some papers present only hand detection step while other works focus on hand posture recognition. We also review some papers working on dynamic hand gesture recognition because they also perform the hand detection and recognition.

In this thesis, our work towards to feature extraction and representation because they are important components in object recognition systems. Therefore, in this chapter we focus on analyzing the methods for feature extraction and hand representation.

2.2 Hand detection

Hand detection is a process that aims at determining the hand region in frames/images. This is the first and important step in the hand postures recognition since the quality of this step will affect the performance of the whole system. However, accurate detection of hands in still images or video remains a challenging problem, due to the variability of hand appearances and environments. In this section, we focus on reviewing the works that are closely related to our work. Some comprehensive surveys on hand detection are available [29, 65, 76].

A number of features have been proposed for hand detection. We divide these features into five categories: pixel value, shape, topography, context, and motion. In most of the methods, a combination of more than one types of the feature is used. Table 2.1 shows the features used in different works in the literature. In the following subsections, we will present a brief description of these features.

2.2.1 Pixel value (intensity/color)

Most hand detection methods utilize pixel values. The pixel value can be the intensity and/or the color. Many methods use color cues to detect skin pixel while some others use intensities to decide whether a pixel belongs to hand region. We can divide these works into two main categories: individual and relationship pixel value. Approaches in the first category rely on the value of individual pixels that often detects hand pixels based on skin color while approaches in the second category utilize the relationship between pixels or regions.

Individual pixel

In the first category, the value of each pixel in the image is matched with a skin color model or a criteria to define whether it is skin pixel or not. To the best of our knowledge, skin color is a popular cue used in hand detection (see Tab. 2.1). However, using only skin color normally is not enough because of unexpected effects of background and illumination (see Fig.2.1 for example). For this reason, the hand detection methods using skin color segmentation employ more features such as context information like faces and other human components. In the following, we will review the detail of some representative papers.

Figure 2.1 – Variation of skin color under different lighting conditions [90].

Table 2.1 – Visual features for hand detection in the literature

ID	Reference	**Pixel value**		**Shape**	**Topography**	**Context**	**Motion**
		Individual	Relationship				
1	Triesch and Malsburg, 1998 [93]	x					x
2	Huang and Huang, 1998 [35]			x			
3	Marcel and Bernier, 1999 [57]	x				x	
4	Zhu *et al.*, 2000 [103]	x					
5	Wu *et al.*, 2000 [100]	x					x
6	Kurata *et al.*, 2001 [44]	x					
7	Lockton and Fitzgibbon, 2002 [53]	x			x		
8	Kolsch and Turk, 2004a [42]		x				
9	Kolsch and Turk, 2004b [59]		x				
10	Ong and Bowden, 2004 [70]			x			
11	Licsar and Sziranyi, 2005 [49]				x		x
12	Wang and Wang, 2007 [98]		x				
13	Francke *et al.*, 2007 [24]	x	x				
14	Choi et al., 2009 [13]	x		x	x		
15	Ravikiran J *et al.*, 2009 [77]			x			
16	Stergiopoulou *et al.*, 2009 [83]	x					
17	Yoder and Yin, 2009 [102]	x					
18	Ding and M. Martinez, 2009 [20]						
19	Le and Mizukawa, 2010 [22]	x			x		
20	Roomi *et al.*, 2010 [78]	x					
21	Tran and Nguyen, 2010 [90]	x					
22	M.Hasan and K.Mishra, 2010 [63]	x					
23	Mittal *et al.*, 2011 [66]	x	x			x	
24	Lee and Lee, 2011 [45]	x					x
25	Dardas and Georganas, 2011 [19]	x		x		x	
26	Pisharady and Vadakkepat, 2012 [75]				x		
27	Liu *et al.*, 2012 [52]	x	x				
28	Boughnim *et al.*, 2013 [7]						x
29	Sgouropoulos *et al.*, 2013 [82]	x			x		
30	Priyal and Bora, 2013 [71]	x					
31	Stergiopoulou *et al.*, 2014 [84]	x					x
32	Chuang *et al.*, 2014 [15]	x			x		
33	Mei *et al.*, 2015 [61]	x	x				
34	Bretzner *et al.*, 2002 [8]	x			x		
35	Wachs *et al.*, 2005 [97]	x					x
36	Chang *et al.*, 2006 [10]	x					
37	Yin and Xie, 2007 [101]	x					
	#Papers used the feature	27	7	5	8	3	7

Some methods used only skin segmentation in hand detection step [63, 78, 83, 90, 102, 103]:

Zhu *et al*. [103] proposed a way to determine the hand in a wearable environment. For a given image, a hand color model and a background color model are generated using Gaussian Mixture Models with the restricted EM algorithm. Then, each pixel in the image is classified into hand pixel and background one based on the generated models. The success of this method relies on the assumption that hand color in a given image is consistent, and hence can be modeled by a Gaussian distribution. Another important prerequisite is that several positions where hand tends to occur with high probability are predefined so that the average hand color in a given image can be estimated reliably. However, in fact, in many applications (e.g. interaction with robots in the real environment), the user stands far from the camera; therefore the above constraints are not satisfied.

In [83], Stergiopoulou *et al.* applied a color segmentation technique based on a skin color filtering procedure in the YCbCr color space. However, the input image using in this work is simple because it contains only the hand taken in a uniform background.

Yoder and Yin [102] proposed a hand detection approach using a Bayesian classifier based on Gaussian Mixture Models (GMM) for identifying pixels of skin color. A connected component based region-growing algorithm is included for forming areas of skin pixels into areas of likely hand candidates. The skin tone pixel segmentation uses a single Bayesian classifier based on the GMMs that are determined from a static set of training data. The classifier is fully computed ahead of time by a training program which processes a set of example images to extract the user-specified color components and train the GMMs accordingly. Once the training program has completed, the classifier remains fixed, and may be used by a separate hand detection application. The hand detection application examines each pixel independently and assigns the pixel to a given class based solely on the output of the classifier for that pixel. Once the Bayesian classifier has identified each block as either skin or background (essentially producing a down-scaled classification image), the results are scanned for connected regions consisting of blocks with a skin confidence of at least 50%. They applied a skin-pixel based region-growing approach to detect the connected components of skin-regions. Connected regions whose width or height is below an empirically derived threshold are assumed to be false positives and are discarded. Any remaining connected regions are presumed to be hands. There are still some limitations in terms of the color information versus various imaging conditions. Hand occlusion is also a challenging issue for model based gesture tracking. The authors indicated a perspective to improve hand detection method that is developing a second post-detection algorithm for hand patch estimation based on the detected skin-pixels. This expected algorithm is intended to separate hand regions from other skins

regions. They pointed out an approach like local binary pattern with AdaBoosting (similar to face detection approach) will be investigated to improve the performance of hand detection and classification.

Tran *et al.* [90] proposed a method to detect skin regions using an algorithm of color segmentation based on thresholding technique. This segmentation is robust to lighting condition thank to a step of color normalization using a neural network. However, the normalization takes so much time because each pixel will be passed to the neural network.

Hand detection methods using only skin color segmentation often work with a constraint that is the input image contains only hand object taken in a simple background. In case of the complex background, the skin segmentation is used in order to reduce the search space for the next step that is a sliding window technique for hand posture recognition [90].

To improve the accuracy of hand detection, many additional features are combined with skin color cue. In many works, the pixel value based visual features reflect relationship between pixels or regions are combined with skin color [24, 52, 61]. Some methods utilized topographical features to decide if a skin region is hand region or not [8, 15, 22, 53, 82]. Context information is used in [19, 57, 66]. One of the popular additional features combined with skin segmentation is motion [45, 84, 93, 97, 100].

In some cases [19,66], hand detection method integrates three or more kinds of feature to obtain good accuracies. In [19], after detecting face region using Viola-Jone detector, the face area is removed by replacing by a black circle. Then, hand region is searched using skin detection and hand contour comparison algorithm based on given hand posture templates. Mittal *et al.* [66] proposed a hand detector using a two-stage hypothesize and classify framework. In the first stage, hand hypotheses were proposed from three independent methods including a sliding window hand-shape detector, a context-based detector, and a skin-based detector. In the second stage, the proposals are scored by all three methods and a discriminatively trained model is used to verify them. This method obtains improvements in precision and recall (average precision: 48.20%; average recall: 85.30% on PASCAL VOC 2010 dataset). However, the computation time is too expensive. The time taken for the whole detection process is about 2 minutes for an image of size 360×640 pixels on a standard quad-core 2.50 GHz machine.

Relationship between pixels or regions

In contrary to the approaches in the first category, the methods in the second category use features that reflect the relationship between pixels/regions or statistic information. Such features are Local Binary Pattern feature (LBP) [68], Histogram of Gradient (HOG) [18], Scale Invariant Feature Transform (SIFT) [54], and Haar-like [96].

In [24], Francke *et al.* combined Haar-like and mLBP (modified Local Binary Pattern) with adaptive skin model generated from face region to detect hand before tracking. Wang *et al.* [98] use sharing SIFT features of different hand posture classes to detect hand. Sharing features are common and can be shared across the classes [87]. HOG feature is often used in hand detection [52, 61, 66]. Mittal *et al.* proposed a hand detection using multiple proposals. In this method, deformable models based on HOG feature is used to detect the hand and the end of the arm. In [52], the hand is located in segmented skin regions using a Cascades AdaBoosted detector based on HOG feature. The hand detection method proposed by Mei *et al.* [61] also segments the skin-color regions first to reduce the detection area then use Gentle Adaboost and Cascade classifier with 3 features: HOG feature, VAR Feature, and Haar feature. VAR feature value of an image is the variance of the gray-scale values of pixels in the image.

Recently, the methods based on Haar-like feature with AdaBoost and Cascade of classifier [96] have obtained good results in face and hand detection. It was used alone [2,42,59,70] or combine with other features [24,61]. To the best of our knowledge, Haar-like feature is one of the most popular features for hand detection [2, 24, 42, 59, 61, 70]. In [76], the authors also gave a similar comment and spent significant space in reviewing these methods. We will survey in more detail below.

Ong *et al*. [70] presented an unsupervised approach to train an efficient and robust detector which detects the hand in an image and classifies the hand shape. In their paper, a tree structure of boosted cascades is constructed. The root of the tree provides a general hand detector while the individual branches of the tree classify a valid shape as belonging to one of the predetermined clusters. For the general hand detector, they trained a cascade of 11 layers with a total of 634 weak classifiers. To build weak classifiers, they used Haar wavelet like features and FloatBoost algorithm. With their database, they reported that the general hand detector obtained an unexpectedly high success rate of 99.8%. However, the hand images for both training and test databases have fairly simple and similar backgrounds.

Kolsch and Turk [42] presented a view-specific hand posture detection. They employed Viola-Jones detector [96]. Since training a detector for every possible hand posture (in order to find the best-performing one) is prohibitively expensive, they proposed a method to quickly estimate the classification potential, based on only a few training images for each posture. They found vast differences in detectability with Viola-Jones' method to find a good Vision-based interfaces initialization gesture. The best detector combined with skin color verification achieves outstanding performance in the practical application, indoors and outdoors: about one false positive in 100,000 frames. The final hand detector that they chose for their application detects the *closed* posture. For sce-

narios where they desired fast detection, they picked the parameterization that achieved a detection rate of 92.23% with a false positive rate of $1.01*10^{-8}$ in the test set, or one false hit in 279 VGA sized frames. According to this paper, mostly convex appearances with internal gray-level variation are better suited to the purpose of detection with the rectangle feature-classification method. Background noise hinders extraction of consistent patterns. The detector's accuracy confirms the difficulty to distinguish hands from other appearances.

Kolsch and Turk [59] analyzed the in-plane rotational robustness of the Viola-Jones object detection method [96] when used for hand appearance detection. They determined the rotational bounds for training and detection for achieving undiminished performance without an increase in classifier complexity. The result - up to 15°total - differs from the method's performance on faces (30°total). They found that randomly rotating the training data within these bounds allows for detection rates about one order of magnitude better than those trained on strictly aligned data. Fig.2.2 shows the feature types used in [42,59].

Figure 2.2 – The feature types used in [42, 59]

Barczak and Dadgostar [2] performed a detailed analysis of Viola-Jones detectors [96] for in-plane rotations of hand appearances. To experiment with hand detection, they implemented a version of Viola-Jones' method using parallel cascades. Each cascade is able to detect hands (one particular gesture) within a certain angle of rotation (on an axis normal to the image's plan). The original set of images is automatically twisted to angles from -90 to 90, spaced by 3 degrees. According to this way, a total of 61 orientations were trained. They indicated that only about 15° of rotations could be efficiently detected with one detector. The training data must contain rotated example images within these rotation limits.

In [24], a hand detector was implemented using a cascade of boosted classifiers to detect hands within the skin blobs. The authors commented that although detectors using cascade of boosted classifiers allow obtaining robust object detectors in the case of face [96] or car objects [31], we could not build a reliable generic hand detector. The reasons are those: (i) hands are complex, highly deformable objects, (ii) hand possible poses have a large variability, and (iii) their target is a fully dynamic environment with cluttered background. Therefore, they decided to switch the problem to be solved. The first hand should be detected then the hand is tracked in the consecutive frames. To detect the first hand, they assume that a specific gesture (fist posture) is made firstly. To determine which posture is being expressed, they apply in parallel a set of single posture detectors over

the ROIs delivered as the output of the tracking module. They indicated that detectors is greatly difficult to process in cluttered backgrounds.

2.2.2 Shape

The shape feature has been utilized to detect the hand in images. The shape feature is often obtained by extracting the contours and edges [13, 19, 35, 77].

The system in [35] applies the corona effect smoothing and border extraction algorithm to find the contour of the hand then uses the Fourier descriptor (FD) to describe the hand shapes. The Fourier Descriptor is defined as the power spectrum of discrete Fourier Transform of the boundary points that are represented by complex numbers. The Hausdorff distance measure is used to track shape-variant hand motion. A combination of the shape and motion information is used to select the key frames. While, Dardas and Georganas [19] used contour comparison algorithm to search for the human hands and discard other skin-colored objects for every frame captured from a webcam or video file. In [77], Canny edge detector and a clipping technique are used to detect edges then the boundary is tracked for fingertip detection.

Choi *et al*. [13] propose a method based on the assumption that a hand-forearm region (including a hand and part of a forearm) has different brightness from other skin colored regions. They firstly segment the hand-forearm region from other skin colored regions based on the brightness difference. The brightness difference is represented by edges. They distinguish the hand-forearm region from others by using the shape feature. They regard the long and big blob as the hand-forearm region. The method can not detect hand region without forearm. While, the constraint is often not satisfied in real applications. After detecting the hand-forearm region, they detect the hand region from the hand-forearm region by detecting a feature point that indicates the wrist. Finally, they extract the hand by using the brightness based segmentation that is slightly different from the hand-forearm region detection.

We can remark that if we can detect the hand contour correctly, the contour will represent well the shape of the hand. However, in the real environment, hand contour extraction is still a challenge.

2.2.3 Topography

The topographical information of the hand such as blobs and ridges, fingertips, wrist, hand center.

Some methods utilized additional topographical features to decide if a skin region is hand region or not [8, 15, 22, 53, 82]. Lockton *et al.* asked user wears a wristband to

compute hand orientation and scale. After detect skin pixels, Le *et al*. [22] determine the center of the hand and the fingertip positions based on the distance transformation image, the connected component labeling image, and a type of feature pixels, which is called distance-based feature pixel. However, this method is required to perform on a good skin color detection image. Sgouropoulos *et al.* [82] detect hand blobs from segmented skin regions based on the size of blobs compared to the face size. In [15], Chuang *et al.* use an integration of a general image saliency detection method and skin information to improve the performance of hand posture detection. In [8], blobs and ridges are extracted from segmented skin regions. Blob and ridge features are then used in hand posture detection, tracking, and recognition.

Some methods [13,49] extract hand region from the hand-forearm region by detecting feature points indicating the wrist. Licsar and Sziranyi [49] used a width-based method to detect wrist points after segmenting hand and arm region based on background segmentation with specific constraints of a camera-projector system. The wrist points then allow of hand region segmentation. The background subtraction method based on the difference between the hand and background reflection in the camera-projector system obtains good results. This good hand and arm segmentation make wrist points detection reliable.

In [75], the hand postures are detected by thresholding the saliency map. Saliency map is created using the posterior probabilities of locations based on shape, texture, and color attention, for the set of hand posture and the background images. A Bayesian model of visual attention is utilized to generate a saliency map, and to detect and identify the hand region. If the posterior probability is above a threshold value, the presence of hand is detected.

2.2.4 Context

Context information is used in some works [19, 57, 66]. They are often combined with other information such as color. In [57], Marcel *et al.* determine if a skin color blob is a hand candidate if it enters an "active window". "Active windows" are defined in the body-face space. Mittal *et al.* [66] use a context-based detector combining with two other detectors (a sliding window hand-shape detector and a skin-based detector) to built a multiple proposals hand detector. The context-based detector proposes hand bounding boxes depending on the end of arms. In [19], Dardas and Georganas used face subtraction combining with skin detection and hand posture contour to detect and track bare hand in the cluttered background.

2.2.5 Motion

Motion is one of the popular feature for hand detection [7, 45, 49, 84, 93, 97, 100]. Motion feature is often combined with skin color [45, 84, 93, 97, 100].

Triesch *et al.* [93] used a thresholded version of the absolute difference images of the intensity combining with skin information to track the user's hand. In [100], Ying Wu *et al*. used motion segmentation to make the localization system based on color segmentation more robust and accurate. In [45], hand regions are found by selecting skin regions having a large number of pixels with sufficiently small values of consecutive count of non-movements (CCNM).

Stergiopoulou *et al.* used a combination of existing techniques, based on motion detection and a skin color classifier to detect hand. Motion detection is based on image differencing and background subtraction. Specifically, image differencing of three consecutive frames, which detects sudden movements, are considered to define the motion Region of Interest (mROI). Consequently, a background subtraction step is applied on the mROI, to track the hand even if it stops moving temporarily.

In [97], the motion of the hand is interpreted by a tracking module. Boughnim *et al.* [7] used a pyramidal optical flow for the detection of large movements and hence determine the region of interest containing the expected hand. They employed an elliptic least-squares fitting to remove non-hand moving points. They hence segment the hand surface.

2.2.6 Discussions

We can see that motion is one of the good features for hand detection in case of dynamic hand posture. However, the goal of our work is static hand gesture recognition. Besides skin color, topographical features and shape features are used more often than context information because they are good for hand detection and hand posture representation. Nevertheless, the extraction of these features of hand posture is still a challenge in real environments.

Skin color also is a popular feature for hand detection. However, in most of real systems, good skin color detection is not feasible because of very complex conditions such as cluttered background and variety of illumination. The precision depends strongly on lighting condition, camera characteristics, and human ethnics [37]. In [76], the authors review many methods in skin color segmentation then also gave a similar conclusion.

To obtain good results of skin color segmentation, many approaches are used. Some of them is based on specific constraints in that the skin detector can work well. Tran *et al.* [90] proposed a skin color normalization method based on neural network however the

computation time is too expensive.

Another way to improve accuracy of hand detection is combining skin color with some additional features to build complex detectors. For example, recently, a hand detection using multiple proposals [66] was proposed. The three proposal mechanisms ensure good recall, and the discriminative classification ensures good precision. However, the time taken for the whole detection process is about 2 minutes for an image of size 360×640 pixels on a standard quad-core 2.50 GHz machine. The complex hand detectors with high computation time could not be used for human-machine interaction systems. Moreover, an integrated method that use skin color combining with other features still obtains false negatives when real skin pixels are not matched skin model because of complex conditions concerned above.

The way to cope with hand detection problem based on background subtraction and motion is good for dynamic hand gestures. However, background subtraction is typically based on the assumption that the camera system does not move with respect to a static background while the foreground moves. This constraint is not satisfied in our work where the hand is often static during user trigger a command by a hand posture.

To the best of our knowledge as well as information in a recent comprehensive survey [76], one of the most popular features reflects relationship between pixels or regions is Haar-like because of its benefits. The methods utilized Viola-Jones detector that use Haar-like feature and Cascade of AdaBoost classifiers are outstanding because of their advantages that are to be invariant to scale change and illumination change, and have real-time performance. Using Viola-jones detector for hand detection in our context is a realizable approach. However, the main drawback of Viola-Jones method for hand detection is that it is affected by background. In our work, we will try to adapt Viola-Jones method into hand detection that keeps advantages of Viola-Jones method as well as avoids unexpected effects of background.

2.3 Hand postures recognition

Hand postures recognition takes a hand region image as a result of hand detection step and returns a label of hand posture. Challenges to vision-based hand posture recognition are the following: (i) similar to other problems in computer vision, vision-based hand posture recognition is affected by changes in lighting condition, cluttered backgrounds, and changes in scale; (ii) hand is a deformable object; there exist a considerable number of mutually similar hand postures; (iii) applications using hand posture generally require real-time, user-independent recognition.

A number of hand recognition methods have been proposed to address these chal-

lenges [11, 33, 72, 76]. These methods can be divided into two main categories depending how the hand is represented: explicit or implicit. Table 2.2 shows different types of visual features used for hand posture recognition in the literature.

2.3.1 Implicit representation

Implicit representation means that the representation replies on visual features are computed directly from pixel values or reflect the relationship between pixels or regions.

Pixel value based features

One of the simple kinds of features is raw values of pixels. In this case, the image is often shaped into a 1D vector as the feature vector.

Marcel and Bernier [57] used resized hand image as the input of a neural network model that is already applied to face detection: the constrained generative model (CGM) to recognize hand posture. The number of inputs for each neural network model hence corresponds to sizes of hand images for each posture. In [53], a hand image is represented as a 1D column vector that is the concatenation of the image columns. On the raw 1D vectors, they use a combination of exemplar-based classification [30, 88] and their proposed "deterministic boosting" algorithm to recognize hand postures.

The using raw images often makes dimensionality of feature vector space large. To reduce the dimensionality, the Principal Component Analysis (PCA) is used [13, 99]. In [99], Who and Huang test their proposed learning approach, the Discriminant-EM (D-EM) algorithm, on physical and mathematical features. To extract mathematical features, a hand image is resized to 20×20, which gives a 400-dimension raw image space. PCA is then employed to find a lower-dimensional feature space. In [13], hand gestures are recognized by using PCA and Neural Network. The input of PCA is a column vector of the elements including the pixel values of a hand image. The weight vector of PCA is used as the input of Neural Network.

Besides the using PCA, Hasan and Mishra [63] divide the input hand posture image into 25×25 blocks then calculate the local brightness of each divided block to compute the feature vector. Each hand posture image produces 25×25 feature values. The recognition algorithm is based on their proposed matching algorithm.

The pixel value based features are simple and easy to calculate. However, they do not sound robust to rotation as well as do not capture the relationship between pixels/regions. These hand representations work well with aligned hands, nevertheless it will not work in case of the variability in alignment.

Table 2.2 – Visual features for hand posture recognition in the literature

ID	Reference	Implicit		Explicit	
		Pixel value based	Relationship between pix-els/regions	Shape	Topography
1	Freeman and Roth, 1994 [25]		x		
2	Triesch *et al.*, 1996 [91]				x
3	Triesch and Malsburg, 1998 [93]				x
4	Huang and Huang, 1998 [35]			x	
5	Marcel and Bernier, 1999 [57]	x			
6	Wu and Huang, 2000 [99]	x	x	x	
7	Lockton and W. Fitzgibbon, 2002 [53]	x			
8	Ong and Bowden, 2004 [70]		x	x	
9	Licsar and Sziranyi, 2005 [49]			x	
10	Just *et al.*, 2006 [36]		x		
11	Chen *et al.*, 2007 [12]		x		
12	Wang and Wang, 2007 [98]		x		
13	Francke *et al.*, 2007 [24]		x		
14	Choi et al., 2009 [13]	x			
15	Ravikiran J *et al.*, 2009 [77]				x
16	Stergiopoulou *et al.*, 2009 [83]				x
17	Ding and M. Martinez, 2009 [20]				x
18	Kaufmann *et al.*, 2010 [38]			x	
19	Kelly *et al.*, 2010 [39]			x	
20	Roomi *et al.*, 2010 [78]				x
21	Tran and Nguyen, 2010 [90]		x		
22	M.Hasan and K.Mishra, 2010 [63]	x			
23	Kumar *et al.*, 2010 [43]		x		
24	Chuang *et al.*, 2011 [14]		x		
25	Lee and Lee, 2011 [45]				x
26	H. Dardas and D. Georganas, 2011 [19]		x		
27	Pisharady and Vadakkepat, 2012 [75]		x		
28	Liu *et al.*, 2012 [52]				x
29	Gupta *et al.*, 2012 [32]		x		
30	Boughnim *et al.*, 2013 [7]			x	
31	Sgouropoulos *et al.*, 2013 [82]				x
32	Li and Wachs, 2013 [46]				x
33	Li and Wachs, 2014 [47]				x
34	Priyal and Bora, 2013 [71]			x	
35	Chuang *et al.*, 2014 [15]		x		
36	Bretzner *et al.*, 2002 [8]		x		x
37	Wachs *et al.*, 2005 [97]		x		
38	Chang *et al.*, 2006 [10]				x
39	Yin and Xie, 2007 [101]				x
	#Papers use feature	5	15	8	14

Features reflect relationship between pixels/regions

Instead of the using directly pixel values, some other visual features that reflect relationship between pixels or regions are proposed for hand posture recognition.

One of the popular features using for hand posture recognition is Gabor filter. Gabor filter has used for feature extraction since a long time in image processing because of its remarkable mathematical and biological properties. Moreover, frequency and orientation representations of Gabor filters are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination. For these reasons, they are applied to hand posture recognition.

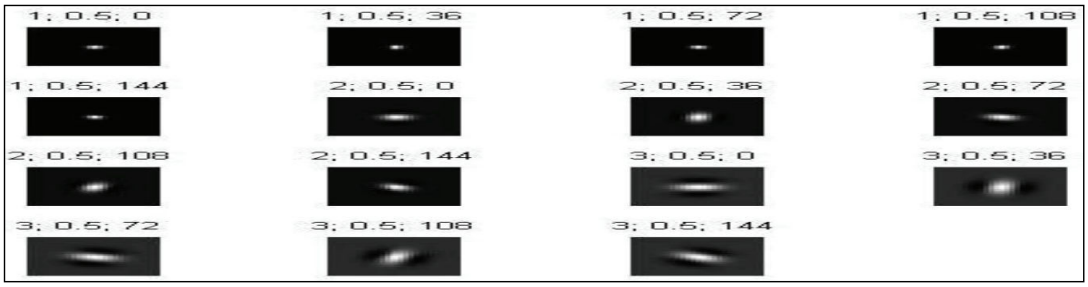
In [99], the authors test their proposed learning algorithm, Discriminant-EM (D-EM), on two kinds of feature physical and mathematical features. Where the physical features is a concatenation of texture features (Gabor wavelet filters) and some other features including edge, Fourier descriptor (10 coefficients), statistic features (the hand area, contour length, total edge length, density, and 2-order moments of edge distribution). To extract texture features, they use Gabor wavelet filters with 3 levels and 4 orientations. Each of the 12 texture features is the standard deviation of the wavelet coefficients from one filter.

Kumar *et al.* [43, 75] used Gabor filters at two layers of a 4-layer hierarchical system based on the primate visual system. In this system, the simple cells in the primary visual cortex (V1) are imitated at Layer 1 by a battery of Gabor filters with 4 orientations (0°,45°,90°,135°) and 16 sizes (divided into 8 bands). At Layer 2, the complex cells in V1 are modeled by applying a MAX operator locally to the outputs of Layer 1 (over different scales and positions). The standard model features (SMFs) [81] are computed from Layer 2 then passed into an SVM classifier with linear kernel to classify hand postures.

Gupta *et al.* [32] used 15 Gabor filters computing on 3 different scales and 5 different orientations. A combination of PCA and LDA is used to reduce the dimensionality of the Gabor filtered image. Classification of hand postures is done using the extracted features, with a multiclass SVM classifier. Fig.2.3 illustrates the Gabor filters using in [32].

Besides Gabor filter, Haar-like feature is another wavelet feature being more simple and cheaper computation time. Haar-like feature is well known in Viola-Jones detector for face detection [96]. Many researchers also have employed it to design hand posture recognition [12, 24, 70, 90, 97]. Most of them are obtained by using several single gesture detectors working in parallel.

Ong *et al*. [70] trained each hand shape detector corresponding to a group of hand shapes using a Cascade of AdaBoost Classifiers and Haar-like feature. In [12], a two-level approach is used. The lower level of the approach implements a parallel cascades structure using Haar-like features and the AdaBoost learning algorithm to classify different hand postures. The higher level implements the linguistic hand gesture recognition using a



(a)

(b)

Figure 2.3 – (a) Kernel response with different scales and orientations. (b) Gabor filter response of a typical hand gesture [32]

.

context-free grammar-based syntactic analysis.

Francke *et al.* [24] uses a set of single gesture detectors in parallel over the ROIs (output of the tracking module) to classify hand postures. Each single gesture detector is implemented using a cascade of boosted classifiers (active learning and bootstrap technique with Haar-like and mLBP-modified Local Binary Pattern feature). Wachs *et al.* [97] used Haar-like features to represent the shape of the hand. These features are then input to a Fuzzy C-Means Clustering algorithm [4] for pose classification.

Tran *et al*. [90] proposed a hand posture classification method consisting of 2 steps. The first step aims at detecting skin regions using a very fast algorithm of color segmentation based on thresholding technique. In the second step, each skin region is classified into one of hand posture classes using Cascaded Adaboost technique. The methods in this category avoid the bad effect of variant lighting condition and some case of skin-colored background region. However, they are affected by cluttered background because their positive samples still contain some background surround the object.

Besides Gabor filter and Haar-like feature, some other features have been used for hand posture recognition. Such features are orientation histogram (used in [25]), Local Binary Pattern feature (LBP) (used in [24]), the features based on the Modified Census Transform (MCT) (used in [36]), visual cortex-based feature (use in [15, 43]), and symbolic features (used in [97]).

Among many visual features for hand posture recognition, recently some researchers have obtained good results with SIFT features. SIFT features, proposed by Lowe [54], are local and based on the appearance of the object in the image from particular interest points. They are invariant to image scale and rotation, robust to illumination changes, noise, and minor changes in viewpoint. SIFT features are highly distinctive of the image. For this reason, SIFT feature has used in hand posture recognition [14, 19, 98]. Wang *et al.* [98] use the discrete AdaBoost learning algorithm with SIFT. Chuang *et al.* [14] use a Hierarchical Bag-of-Features with Affine-SIFT (ASIFT) to extract features.

Among the methods using SIFT features, a remarkable method is proposed by Dardas and Georganas in [19]. In [19], good results were obtained when applying SIFT features with BoW (Bag of Words) and SVM (Support Vector Machine) into hand posture recognition. This method, however, does not work well with low resolution due to the limited number of detected keypoints. Fig.2.4 shows some images with their keypoints in [19].

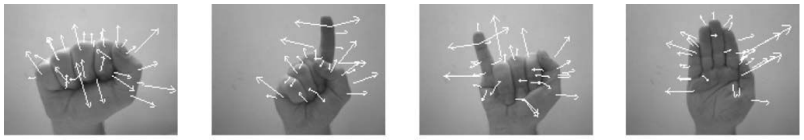


Figure 2.4 – The keypoints extracted from 640×480 training images in [19]. (a) First with 35 features. (b) Index with 41 features. (c) Little finger with 38 features. (d) Palm with 75 features.

2.3.2 Explicit representation

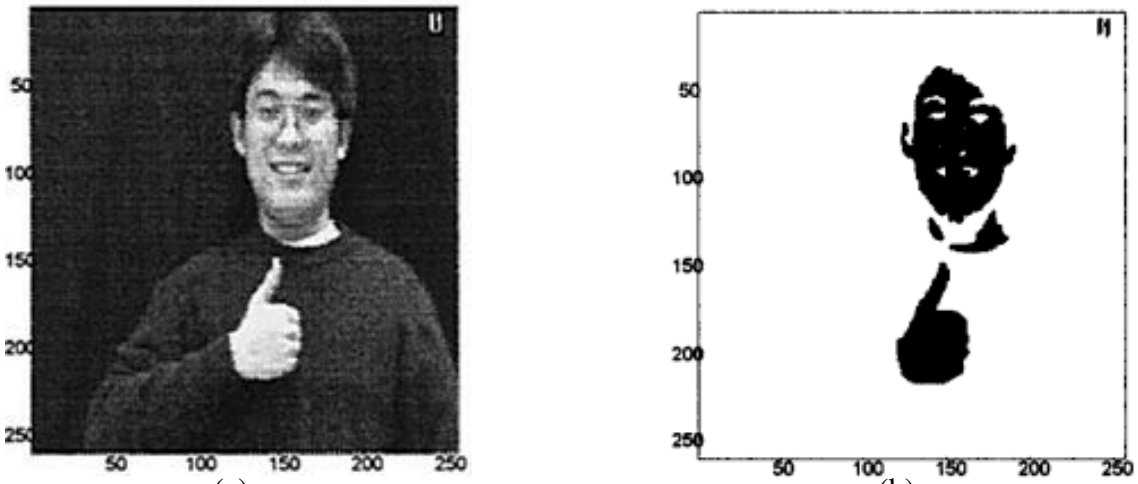
We class hand representation methods which present intuitive features of hand to "explicit representation" category. The methods belonging to this category often require good hand segmentation results to extract hand shape features (for example edges, contours) or topographical features such as fingers. Shape and topographical features are good for hand posture representation if we can segment well the hand region from the image.

Shape

Shape features described here include edges, contours, and the features extracted based on edges and contours of the hand. Some of these features are Fourier descriptors, Shape Context. The following is some methods for hand posture recognition using shape features.

Huang and Huang [35] applied the scale and rotation-invariant Fourier descriptor to characterize hand figure. They firstly apply Otsu thresholding method, the corona effect

smoothing and border extraction algorithm to find the border of the hand shape. The hand region is segmented clearly because of simple background, Fig.2.5. The hand boundary



(a) (b)

Figure 2.5 – Good segmentation result from simple background in [35]. (a) Original input image. (b) Binary image produced by Ostu thresholding.

points is described by Fourier series representation with coefficients. The closed contour then be depicted by a Fourier descriptor (FD) vector [17]. They assume that the local variations of hand shape is smooth. The hand gesture model is built based on motion and shape information of the key frames. The gesture recognition is done by a graph matching between the input gesture model and the stored models using a 3D Hopfield neural network (HNN).

In [99], 10 coefficients from the Fourier descriptor are used to represent hand shapes combining with Gabor wavelet filters to build mathematical features. The mathematical features are used to test the authors' proposed algorithm: Discriminant-EM (D-EM). Licsar and Tamas [49] applied a boundary-based method for the classification. The hand contours are classified by the nearest-neighbor rule and the distance metric based on the modified Fourier descriptors (MFD) [79] being invariant to transition, rotation and scaling of shapes.

Besides Fourier Descriptor, some other hand representations based on shape have been proposed. Ong *et al.* [70] used Shape Context [3] (Fig.2.6) with K-mediod algorithm to classify hand shapes into a number of clusters. A cascade of classifiers was then trained on the images of each cluster to build a tree of hand detectors. The head of the tree is a general hand detector, and the individual branches of the tree classify a valid shape as belong to one of the predetermined clusters.

In [38], Kaufmann *et al.* proposed a hand posture recognition based on searching hand models (Fig.2.7). The searched models are hand contours shapes, encoded as lists of

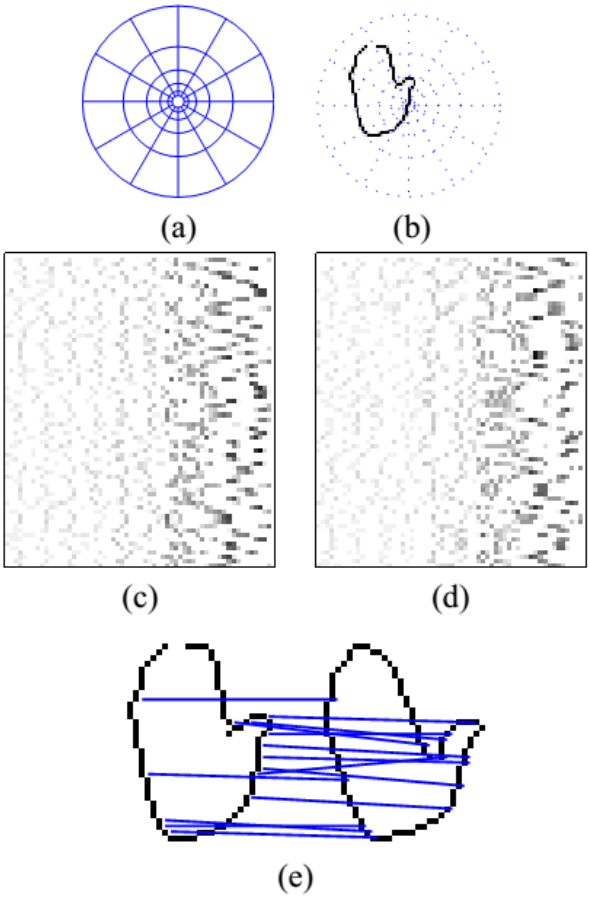


Figure 2.6 – Shape context components [70]: (a) the log polar histogram used, (b) shows it centered on a point on a hand contour. (c) and (d) visualisations of the set of log polar histograms for two hand contours. (e) some correspondances between points on two hand contours using the shape context metric.

points. The search space, i.e. the space of all possible solutions, is a 5 dimension space, which corresponds to the 5 following parameters: hand model; horizontal translation of the model; vertical translation; apparent scale of the model, which varies with the size of the user's hand and its distance to the camera; and rotation of the hand with respect to the optical axis of the camera.

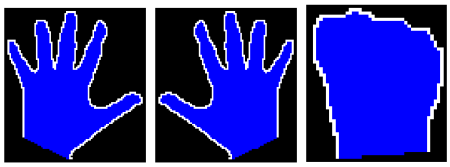


Figure 2.7 – Hand models used in [38]

Kelly *et al.* [39] used a combination of eigenspace Size Functions and Hu moments features to classify different hand postures. Eigenspace Size Functions is a variation of

Size Functions. Size Functions are integer valued functions that represent both qualitative and quantitative properties of a visual shape [94]. While, Hu moments [34] are a set of transition, scale and rotation invariant moments. The set of Hu moments are calculated from the hand contour. For each posture class, they use a set of two support vector machines corresponding to eigenspace Size Functions and Hu moments respectively to build a classifier.

Boughnim *et al.* [7] used a preprocessing step to compute the hand contour then use an image scanning method providing a signature that characterizes non-star-shaped contours with a one-pixel precision for hand posture recognition. Star-shaped contours is described by a relation from the angular coordinates of its pixels into their radial coordinates. The signature is a set of data that characterize the corresponding contour. The components of the signature are equal to the distance from the pixel located at different intervals and along different directions in polar coordinates. They use PCA to reduce the dimensionality of the signature data. The hand posture classification is done by a Bayesian classifier with Mahalanobis distance.

In [71], after segment and normalize hand region, Priyal and Bora use Krawtchouk moments to represent hand shape (Fig.2.8). The Krawtchouk moments are discrete orthogonal moments derived from the Krawtchouk polynomials [41]. The moment features are then computed with a Nearest Neighbourhood classifier to classify static hand gestures.

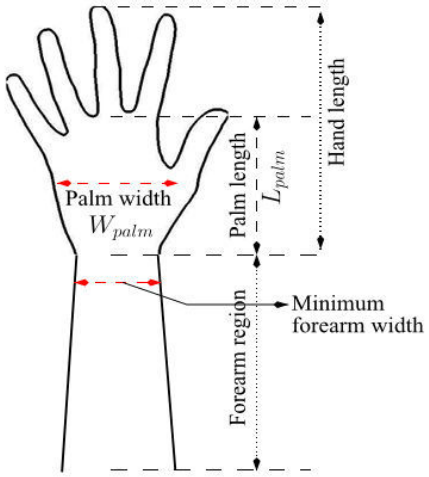


Figure 2.8 – Hand geometry [71]

Shape features are important information allowing to improve the performance of hand posture recognition. Most of the concerned methods above are based on an assumption that the hand segmentation result is good. However, hand segmentation is still a challenge in the real environment.

Topography

Topographical features such as fingers have used in many hand posture recognition methods because of they are intuitive representations of hand posture. Palm and fingers are popular topographical features used for hand posture recognition. Many hand posture recognition base on extracted fingers, fingertips and/or palm [8, 10, 20, 45, 52, 77, 82, 83, 101].

Ravikiran *et al* [77] proposed a boundary-trace based finger detection technique to locate the fingertip. They used the locations of fingertips to identify 9 classes of hand gestures belonging to the American Sign Language which have open fingers. This method requires good segmentation results. The images in this paper have simple backgrounds, Fig.2.9.

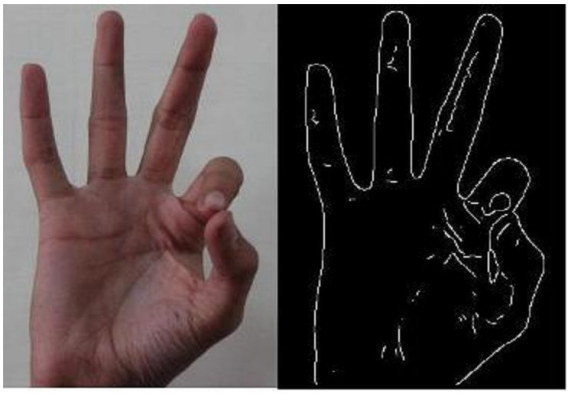


Figure 2.9 – Image of a Hand Gesture Before and After Edge Detection in [77]

In [83], the region of hand is detected by applying a color segmentation technique on a simple uniform background. Then, the palm morphological characteristics and finger features that allow identifying the raised fingers are extracted based on the Self-Growing and Self-Organized Neural Gas network, Fig.2.10. The likelihood-based classification technique is used to recognize hand gestures. This method could not be applied for applications with complex backgrounds because the segmentation of the hand in a real environment with cluttered backgrounds is not always good.

In [20], hand shapes are defined by the 3D-world positions of a set of hand points (fiducials), corresponding to the hand knuckles, finger tips and the wrist (Fig.2.11). A new test hand shape is classified to have the same meaning (class) as that shape with the smallest value for a simple combination of the three errors (error of the fitting process, the comparison of the angle, and the visibility difference between two handshapes). However, all the three components can be reliably recovered from two-dimensional video sequences because of using semi-automatic detection.

In [82], Sgouropoulos *et al.* used the hand morphology described by the palm and

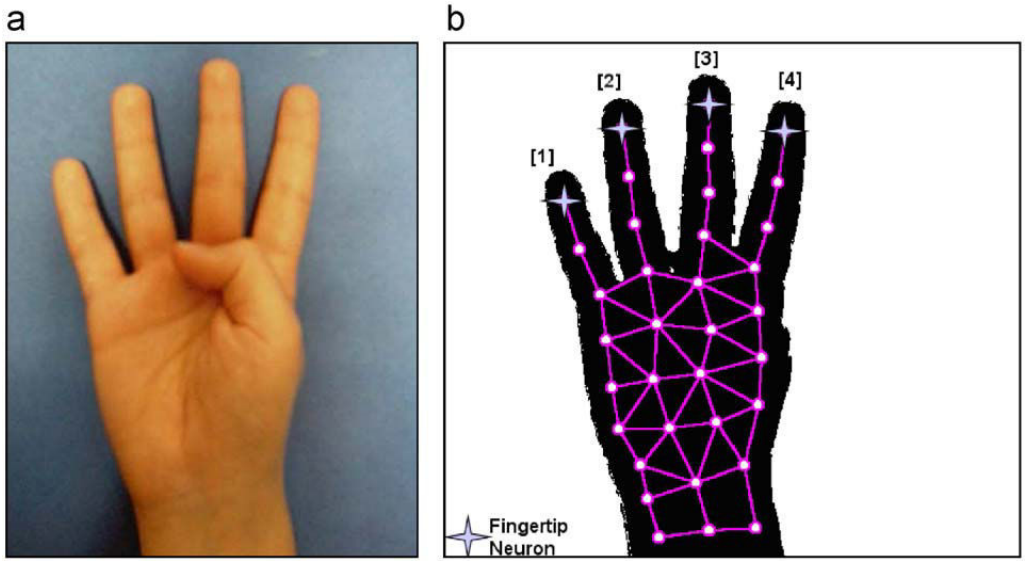


Figure 2.10 – Identifying the raised fingers in [83]: (a) Input image and (b) numbering of fingers.

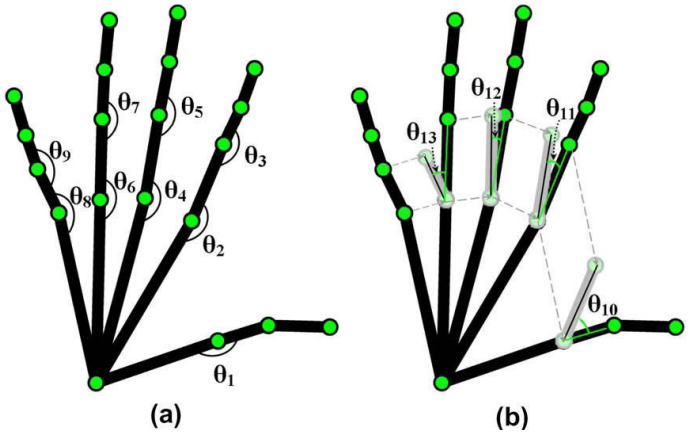


Figure 2.11 – An illustration of (a) the joint angles, and (b) the angle differences between fingers used in [20].

its centre, the number of the raised fingers and their tips and roots to represent hand posture. To classify the raised fingers into five classes (thumb, index, middle, ring, little), they extracted three features based on the hand morphology: RC angle, TC angle and distance from the palm centre for the index finger then used a HMM classifier. RC Angle corresponds to the angle between the vertical axis and the line that joins the root point and the palm centre. While, TC Angle is the angle of the line that joins the fingertip point and the palm centre, Fig.2.12.

Liu *et al.* [52] presented a hand posture recognition using finger geometric feature. In this system, the geometric features among fingers, palm and forearm are extracted based on arranged hand components that are found out with the help of skeleton. The finger

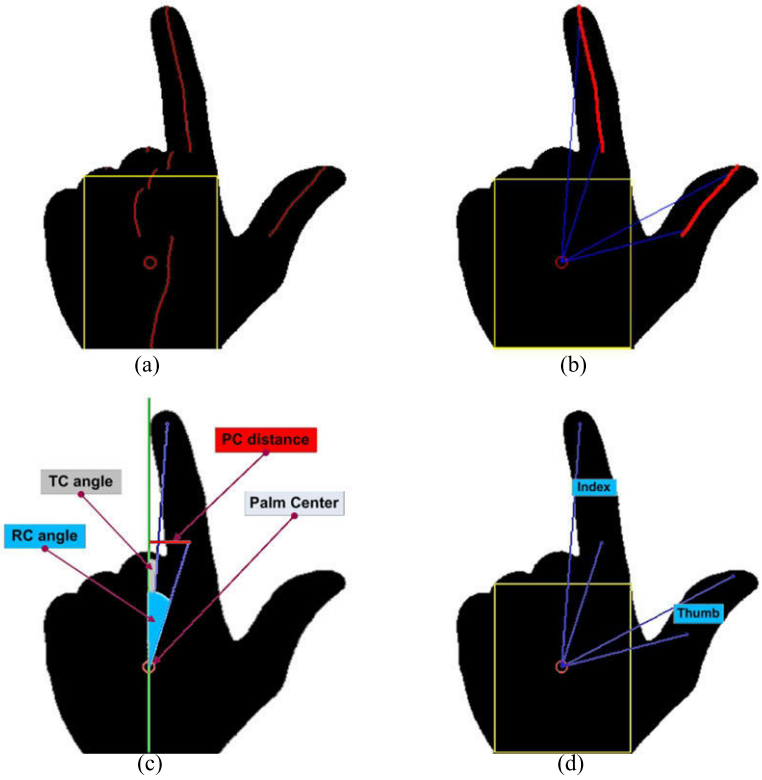


Figure 2.12 – Hand representation used in [82]. (a) Local maxima of horizontal distance transform; (b) Root and fingertips points; (c) RC, TC angles and distance from the palm center; (d) Raised fingers classification.

geometric features are translation, rotation and scale invariant. Those features are used in SVM classification for hand posture recognition. This method can recognize twelve different types of hand postures for both hands respectively. However, the extraction of this hand posture descriptor needs a good hand boundary that is the result of hand segmentation. For this reason, the poor skin segmentation caused by complex backgrounds will lead to wrong recognition results.

Beside the representation of components of the hand such as fingers, blobs and ridges are used in some methods [8, 52].

In [52], blob and ridge features are extracted. After that, hand components are searched with the help of skeleton. Then they are ordered into a serial arrangement. Fingers, palm, forearm and relative geometric features are also extracted to explore the kinematical constraints of the hand with forearm in order to extract finger geometric features. Those features are used in SVM classification for hand posture recognition.

Bretzner *et al.* [8] used a hand model that consists of the palm as a coarse scale blob, the five fingers as ridges at finer scales, and fingertips as even finer scale blobs. Fig.2.13 shows an example of blobs and ridges feature extracted from hand image. Blobs and ridges are intuitive for palm and fingers representation.

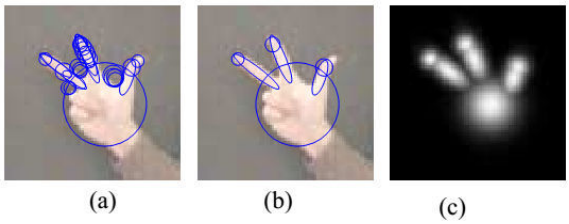


Figure 2.13 – The result of computing blob features and ridge features from an image of a hand [8].

In [10], to extract hand feature for hand posture recognition, Chang *et al.* decomposed the binary hand silhouette into the finger part and the palm part by morphological operations according to the radius of the minimum bounding circle (MBC). The Zernike moments (ZMs) [40] and pseudo-Zernike moments (PZMs) of the finger part and the palm part, respectively, are then computed with different importance based on the center of the MBC. They used 1-nearest neighbor techniques to perform feature matching between the input feature vector and stored feature vectors to identify static gestures.

In [101], the topological features, such as the number and positions of fingers, are found based on edge points of fingers. They extract the branch number (BN), the width of the branches (BW), the distance between the finger, and the arm (BD), then recognize hand postures based on a classification criterion of the postures using BN, BW, BD parameters, Fig.2.14. A branch is a segment between two conjoint feature points (edge point) detected on a search circle. The widest branch should be the arm. The parameters are simple and easy to estimate in real time as well as distinctive enough to differentiate hand postures defined explicitly in the paper. The recognition algorithm in this paper is invariant to rotations and user independent because the topological features of human hands are quite similar and stable. However, some extracted parameters will be false because of the false edge points detecting from imperfect segmentation. The authors have proposed a simple threshold-based technique to remove these false detections. However, it is still a limitation.

Another beautiful method for hand posture representation is a method based on Elastic graph matching [46, 47, 91, 93]. This approach does not require a good segmentation, but the computation time is too great for many applications. In [47], hand postures are matched with predesigned bunch graph models (Fig.2.15). Each node in this model contains local features. The classification task is done by finding the best matching region between bunch graph and the target image. The bunch graph is slid on the image. At each position of the bunch graph, the position of each node is refined to find the best one considering the distortions of the nodes.

Figure 2.14 – The extraction of topological features in [101]. The green circles represent the search circles and the red points represent the extracted feature points.

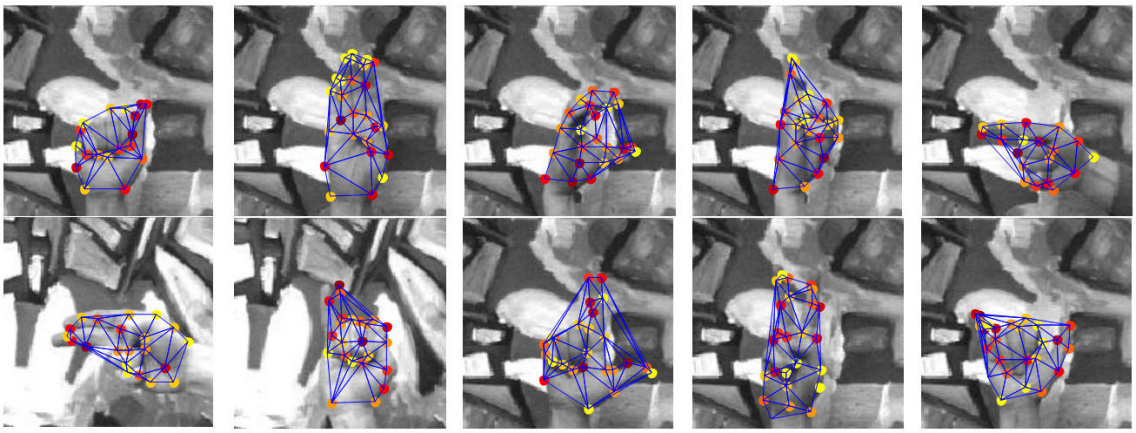


Figure 2.15 – 10 classes of sample hand gesture images after the matching process in [47]

2.3.3 Discussions

The hand representation methods belonging to explicit presentation approach are intuitive and easy to understand for hand posture representation. However, these method requires a good hand segmentation result such as a clear hand contour.

The implicit hand representation methods do not reflect the structure of the hand. These methods hence do not take advantages of the specific structure of the hand. Some method using SIFT feature obtains good results on specific datasets that can provide a rich set of key points. However, when the resolution of hand image is low, the extracted set of key points is poor. In this situation, the method using SIFT feature will obtain a bad performance.

There is not a good combination between implicit and explicit representation. Some method used both kinds of feature implicit and explicit. Nevertheless, to use explicit features, we still need a good segmentation. The method based on Elastic graph matching is good representation for hand posture recognition. However, the computation time is too expensive because of shifting and transforming positions of both graph and nodes to find the best candidate.

In this thesis, we try to find a method for hand presentation that is a flexible combination of implicit and explicit representation approaches. The expected hand representation method does not require a very good segmentation result as well as has an ability to reflect the structure of hand.

2.4 Conclusions

Based on our analysis and discussions above, we summary here our research directions in each task, hand detection and hand posture recognition:

- **Hand detection**: Develop a method based on Viola-Jones detector that takes advantages of Viola-Jones detector and avoid unexpected effects of background at the same time.

- **Hand posture recognition**: Develop a good hand representation that exploits the flexibilities of implicit representation and reflects the structure of hand at the same time. The proposed method will be based on a good implicit visual object representation combining with a structure suitable to hand.

Chapter 3

Hand detection

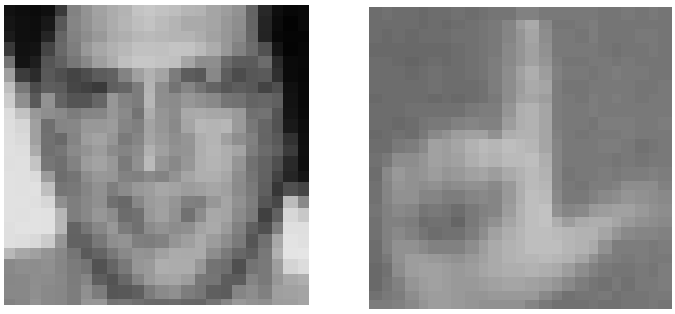
Hand detection is the first step in the overall framework for posture recognition. In this chapter, we present a method for hand detection. Inspired by the ideas of the Viola-Jones detector [96], we introduce a concept that is internal features and then propose to use internal Haar-like features instead of Haar-like features like in the original work of Viola-Jones. The internal Haar-like features are the ones extracted from regions inside object of interest without background. By this way, our method is independent of background changing. We show that internal Haar-like features significantly outperforms Haar-like features on a challenging dataset.

3.1 Introduction

Hand detection takes an image as input, does processing and results in candidate hand regions. This is the first step in most of the hand posture recognition systems. However, this is not a simple task because hand is a deformable object with a high degree of freedom, color and shape changing from one to another. A hand detector is evaluated as good if two main requirements are satisfied: the high detection rate and the small computational time. In a real application of human-robot interaction as our case, the main objective is to design a robust and fast method for hand detection that could be integrated into the posture recognition system. As presented in the section 2.2, many methods have been proposed for hand detection. Among these methods, Viola-Jones detector is one of the most impact algorithms for object detection in still images in the 2000's [96]. It employs the integral image technique to compute Haar-like features in a very fast manner. Then the detector is built from a cascade of classifiers based on Adaboost learning method. Viola-Jones detector achieved competitive detection rates in real-time. A Haar-like feature is defined by a set of black and white rectangles in a particular direction. The feature value is computed as the difference between the sum of pixel values in rectangular regions, so it is invariant

to varieties of skin color and light intensity changing. Meanwhile, the methods for hand detection based on skin color depend on lighting condition and skin color. To be invariant to scale change, the researchers propose a set of features at different scales.

The high detection rates obtained in real-time (e.g. face detection) given by Viola-Jones detector motivate us to use for hand detection. However, we observe some problems of the original Viola-Jones detector. For training, the Viola-Jones detector computes Haar-like features on positive samples that contain the whole object of interest. Unlike the face which is a *pseudo* convex and rigid object, hand posture is deformable and less compact on the image. Therefore, for many hand postures, background pixels possesses a large region in images that make the algorithm strongly affected by background changing [42, 59, 96] (see Fig.3.1). To deal with this issue, we introduce a novel concept that is Internal Haar-like feature. The detection procedure will remain exactly the same as the original Viola-Jones detector.



(a) (b)

Figure 3.1 – (a) A positive face sample used in [96]; (b) A positive hand sample use in [42]

3.2 The framework of hand detection

The framework of hand detection is shown in Fig.3.2. This framework is general with three main steps similar to the Viola-Jones detector.

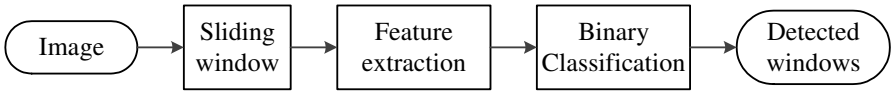


Figure 3.2 – The framework of hand detection

1. Sliding window: This step uses sliding window technique to generate detection windows at different scales.

2. Feature extraction: For each detection window, we extract features to represent hand region. In the original work of Viola-Jones, they extract Haar-like features. In our work, we extract Internal Haar-like features. The main difference between Haar-like and Internal Haar-like feature is its supporting region on which the feature is computed. The Internal Haar-like feature is computed only on internal regions of the interested object without background while Haar-like feature is computed on the region containing the whole object of interest that often includes background. By this way, Internal Haar-like features do not depend on the background changing as Haar-like features.

3. Binary classification: Features extracted from the previous step will be inputted into a binary classifier that is a cascade of Adaboost classifiers. This step classifies detection windows into two categories: hand and non-hand.

In the following, we will present feature extraction and classification steps of the framework.

3.3 Feature extraction

As presented previously, Haar-like features have been used in the original work of Viola-Jones detector and shown to be very powerful in many practical applications, specifically in face detection [96]. The Viola-Jones detector also has been applied successfully in hand detection (see Section 2.2). In this section, we remind what a Haar-like feature is and how to compute it. Then we introduce Internal Haar-like features.

3.3.1 Haar-like features

A Haar-like feature f in a detection window is defined by $\langle L, S, T \rangle$, where:

- L is the location of the pixel at which we compute the Haar-like feature. The location L is defined by a two-dimension coordinate (x, y).

- S is the scale factor showing the size of the Haar-like feature. The scale S is composed of $\langle s_x, s_y \rangle$, where s_x is scale factor of x axis, s_y is scale factor of y axis. s_x and s_y are independent.

- T is the type of Haar-like feature, see Fig. 3.3. A type of Haar-like feature is defined by a set of black and white rectangles $RECT$.

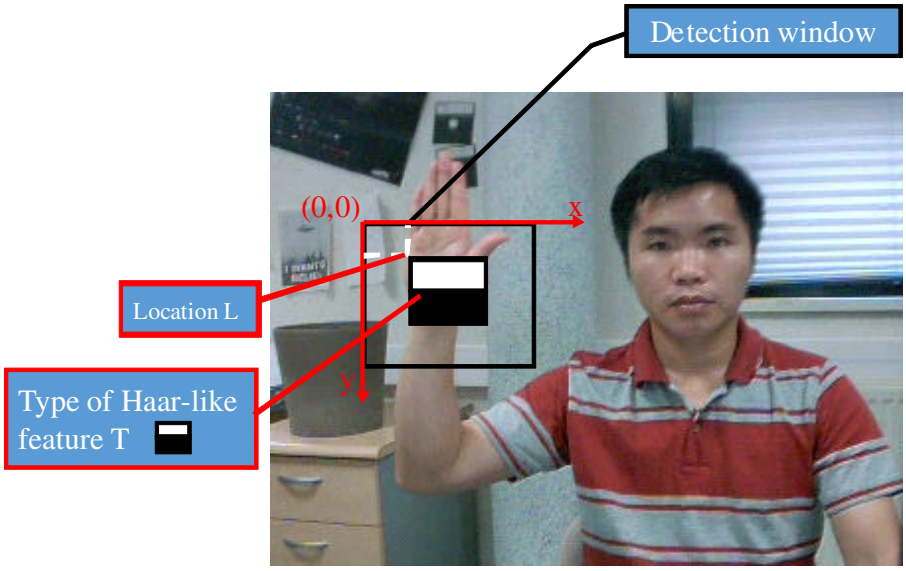


Figure 3.3 – A Haar-like feature in a detection window in a frame

The value of a Haar-like feature is defined as the difference between the sum of pixel values inside black rectangles and the one inside white rectangles. This value on image I is calculated as follows:

$$f(I) = \sum_{r \in RECT} w_r . RectangleSum(r, I) \tag{3.1}$$

where $w_r \in \mathbb{R}$ is the weight of rectangle r. When r is a black rectangle, $w_r < 0$, otherwise $w_r \geq 0$ when r is a white rectangle. The weight compensates for the difference in area size between black rectangles and white rectangles. This means $\sum_{r \text{ is black}} -w_r . Area(r) = \sum_{r \text{ is white}} w_r . Area(r)$. $RectangleSum(r, I)$ is sum of the pixels of image I inside rectangle r that is calculated as:

$$RectangleSum(r, I) = \sum_{(x,y) \in r} I(x, y) \tag{3.2}$$

Haar-like features reflect relationship of intensity between regions in a detection window. Depending on the configuration of rectangles, a Haar-like feature could represent characteristics of edge (see Fig. 3.4(a-d)), line (see Fig. 3.4(e-l), center surround (see Fig. 3.4(m-n)), or diagonal regions (see Fig. 3.4(o)). In pratice, many types of Haar-like features have been proposed [21, 50, 51, 62, 64, 73, 96]. The most popular ones are the set of Haar-like feature types in [96] (Fig. 3.5) and the extended set in [51] (Fig. 3.4). In our work, we use the extended set of Haar-like feature types as in [51].

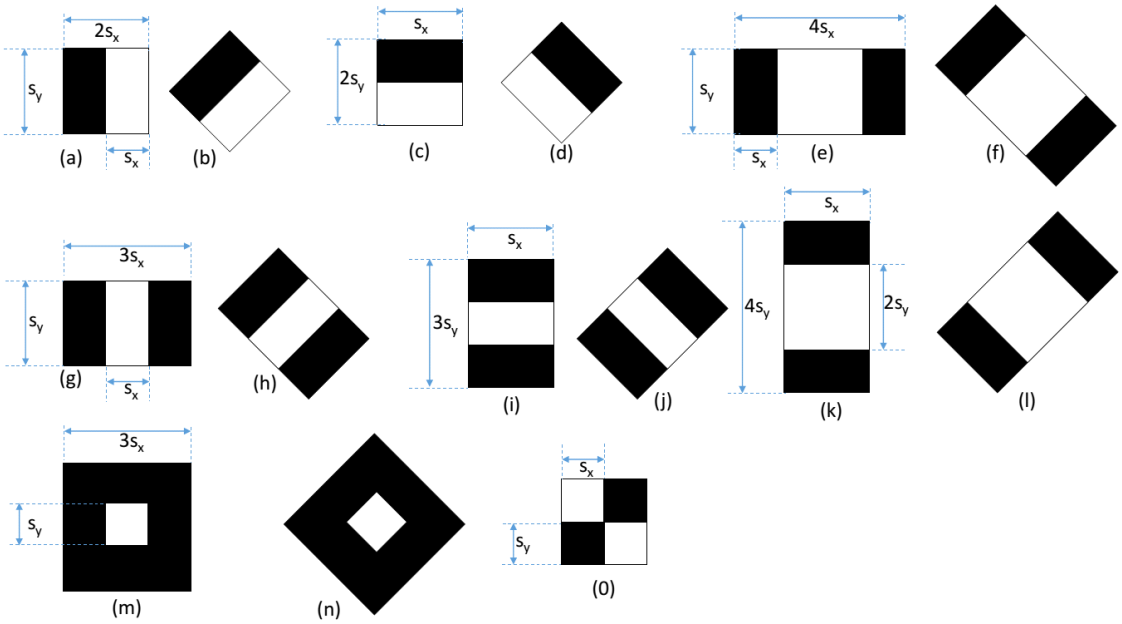


Figure 3.4 – The extended set of Haar-like features [51] used in our system.

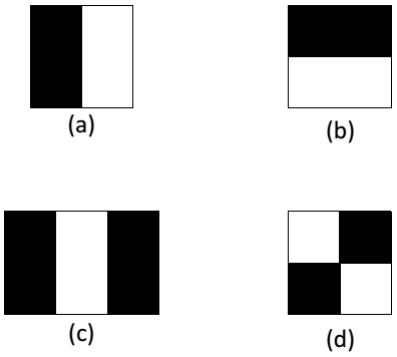


Figure 3.5 – Set of Haar-like feature types in [96].

3.3.2 Fast computation of Haar-like features using integral image

To compute Haar-like features, Viola-Jones [96] proposed a very fast computation method based on integral image. After that, Lienhart *et al.* [51] extended the set of Haar-like feature types by adding some new types and 45° rotated Haar-like features. Lienhart *et al.* also extended the computation method based on integral image for 45° rotated Haar-like features.

Computation method for axis-aligned Haar-like features

Axis-aligned integral image Given a gray-scale image I, the value of integral image $II\,(x, y)$ at position (x, y) is the sum of pixel values inside the axis-aligned rectangle that ranges from the top-left corner of image I at the position $(0, 0)$ to the bottom-right corner at (x, y), see Fig. 3.6. The integral image II can be defined as:

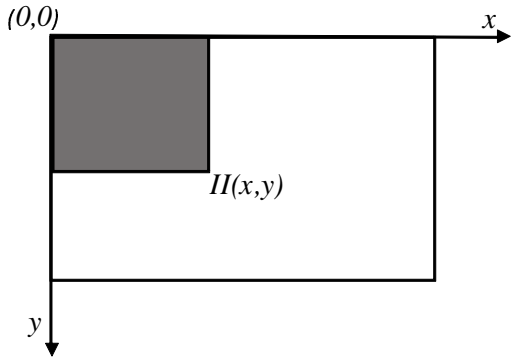


Figure 3.6 – Axis-aligned integral image.

$$II\,(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I\,(i, j) \tag{3.3}$$

$$II\,(x, y) = \begin{cases} I\,(x, y)\,, & \text{if } x = 0 \text{ and } y = 0 \\ \sum_{j=0}^{y-1} I\,(x, j) + I\,(x, y)\,, & \text{if } x = 0 \text{ and } y > 0 \\ \sum_{i=0}^{x-1} I\,(i, y) + I\,(x, y)\,, & \text{if } y = 0 \text{ and } x > 0 \\ \sum_{i=0}^{x-1} \sum_{j=0}^{y-1} I\,(i, j) + \sum_{j=0}^{y-1} I\,(x, j) & \\ \quad + \sum_{i=0}^{x-1} I\,(i, y) + I\,(x, y)\,, & \text{if } y > 0 \text{ and } x > 0 \end{cases} \tag{3.4}$$

$$II(x,y) = \begin{cases} I(x,y), & \text{if } x = 0 \text{ and } y = 0 \\ \sum_{j=0}^{y-1} I(x,j) + I(x,y), & \text{if } x = 0 \text{ and } y > 0 \\ \sum_{i=0}^{x-1} I(i,y) + I(x,y), & \text{if } y = 0 \text{ and } x > 0 \\ \left(\sum_{i=0}^{x-1}\sum_{j=0}^{y-1} I(i,j) + \sum_{j=0}^{y-1} I(x,j)\right) & \\ \quad + \left(\sum_{i=0}^{x-1}\sum_{j=0}^{y-1} I(i,j) + \sum_{i=0}^{x-1} I(i,y)\right) & \\ \quad - \sum_{i=0}^{x-1}\sum_{j=0}^{y-1} I(i,j) + I(x,y), & \text{if } y > 0 \text{ and } x > 0 \end{cases} \tag{3.5}$$

$$II(x,y) = \begin{cases} I(x,y), & \text{if } x = 0 \text{ and } y = 0 \\ \sum_{j=0}^{y-1} I(x,j) + I(x,y), & \text{if } x = 0 \text{ and } y > 0 \\ \sum_{i=0}^{x-1} I(i,y) + I(x,y), & \text{if } y = 0 \text{ and } x > 0 \\ \sum_{i=0}^{x}\sum_{j=0}^{y-1} I(i,j) + \sum_{i=0}^{x-1}\sum_{j=0}^{y} I(i,j) & \\ \quad - \sum_{i=0}^{x-1}\sum_{j=0}^{y-1} I(i,j) + I(x,y), & \text{if } y > 0 \text{ and } x > 0 \end{cases} \tag{3.6}$$

$$II(x,y) = \begin{cases} I(x,y), & \text{if } x = 0 \text{ and } y = 0 \\ II(x,y-1) + I(x,y), & \text{if } x = 0 \text{ and } y > 0 \\ II(x-1,y) + I(x,y), & \text{if } y = 0 \text{ and } x > 0 \\ II(x,y-1) + II(x-1,y) & \\ \quad - II(x-1,y-1) + I(x,y), & \text{if } y > 0 \text{ and } x > 0 \end{cases} \tag{3.7}$$

We can make the computation easier by considering the value at row -1 and col -1 equals to zero:

$$\begin{aligned} II(-1, y) &= 0, y \geq -1 \\ II(x, -1) &= 0, x \geq -1 \end{aligned} \tag{3.8}$$

Then, the equation 3.7 becomes:

$$\begin{aligned} II\left(x, y\right) = \quad & II\left(x, y-1\right) + II\left(x-1, y\right) - II\left(x-1, y-1\right) + I\left(x, y\right), \\ & y \geq 0, x \geq 0 \end{aligned} \tag{3.9}$$

Equation 3.9 is illustrated in Fig. 3.7.

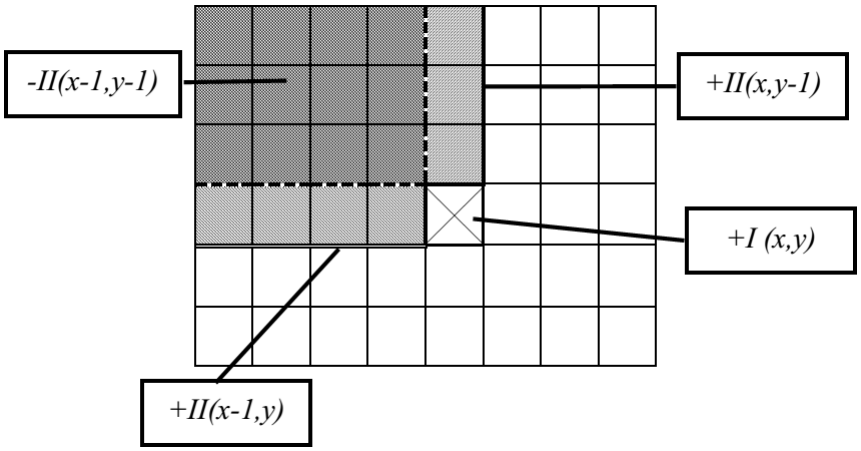


Figure 3.7 – Computation of integral image.

The Integral image II can be calculated with one pass over all pixels of image I, see Fig.3.8

Computation of axis-aligned Haar-like features based-on axis-aligned integral image Using axis-aligned integral image, we can calculate the sum of pixel values inside an axis-aligned rectangle $r = \langle x, y, w, h \rangle$ as follows:

$$\begin{aligned} RectangleSum(r, I) = \quad & II(x+w-1, y+h-1) - II(x+w-1, y-1) \\ & -II(x-1, y+h-1) + II(x-1, y-1) \end{aligned} \tag{3.10}$$

Equation 3.10 is illustrated in Fig. 3.9.

An axis-aligned Haar-like feature is calculated as Eq. 3.1 with sum of rectangle calculated as Eq. 3.10.

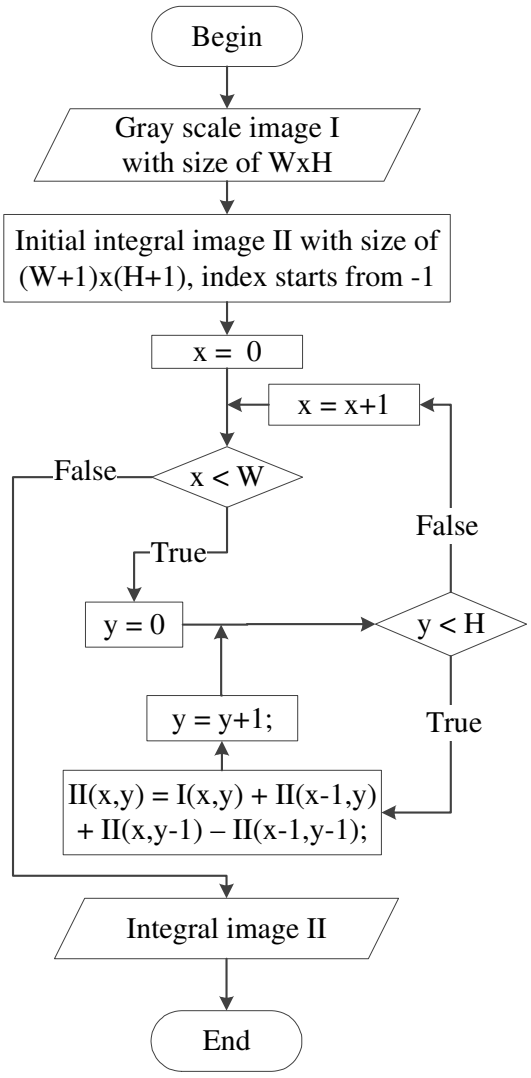


Figure 3.8 – Diagram of computation integral image.

Computation method for 45° rotated Haar-like features

45° rotated integral image Given a gray-scale image I, the value of 45° rotated integral image at (x, y), $RII\,(x, y)$, is the sum of the pixel values of image I in range of rotated rectangle with the bottom most corner at (x, y) (see Fig 3.10). Similar to the computation of axis-aligned integral image, we add two rows -1, -2 and two cols -1, -2 with zero values. The 45° rotated integral image RII can be defined as:

$$\begin{aligned} RII\,(x,y) = \quad & RII(x-1,y-1) - RII(x+1,y-1) - RII(x,y-2) \\ & + I(x,y) + I(x,y-1) \end{aligned} \tag{3.11}$$

The scheme of computation for 45° rotated integral image is shown in Fig. 3.11. In this scheme, integral image RII also can be calculated with one pass over all pixels of I.

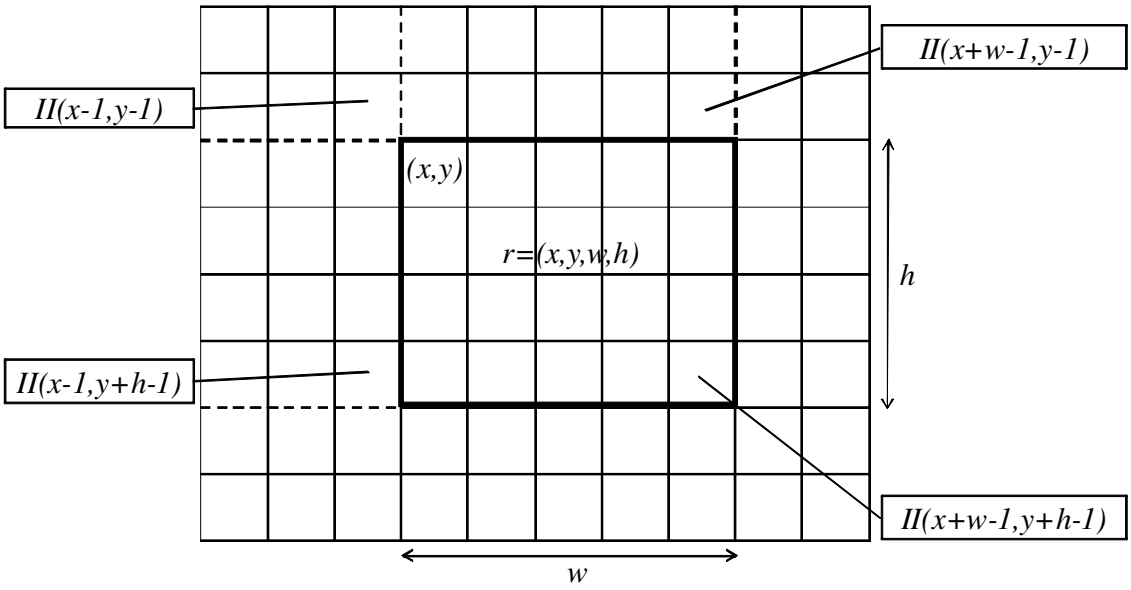


Figure 3.9 – Computation of sum of pixel values inside an axis-aligned rectangle $r = \langle x, y, w, h \rangle$

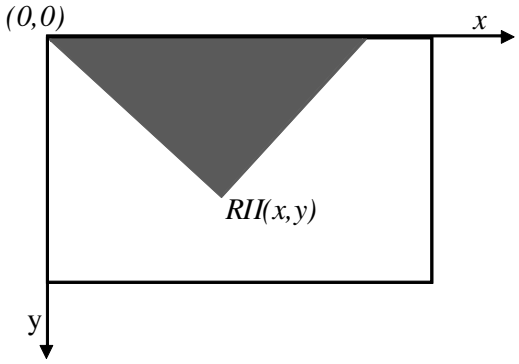


Figure 3.10 – 45° rotated integral image.

Computation 45° rotated Haar-like features based-on 45° rotated integral image 45° rotated integral image allows we calculate sum of pixels inside a 45° rotated rectangle $r = \langle x, y, w, h \rangle$ as:

$$\begin{aligned} RectangleSum(r, I) =& RII(x + w - h, y + h + w - 1) \\ & - RII(x + h, y + h - 1) \\ & - RII(x + w, y + w - 1) + RII(x, y - 1) \end{aligned} \tag{3.12}$$

Equation 3.12 is illuminated in Fig. 3.12. A 45° rotated Haar-like feature is calculated as Eq. 3.1 with sum of rectangle calculated as Eq. 3.12.

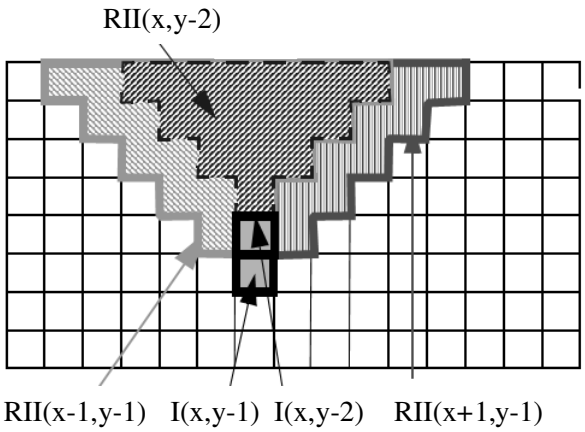


Figure 3.11 – Calculation scheme for $45°$ rotated integral image.

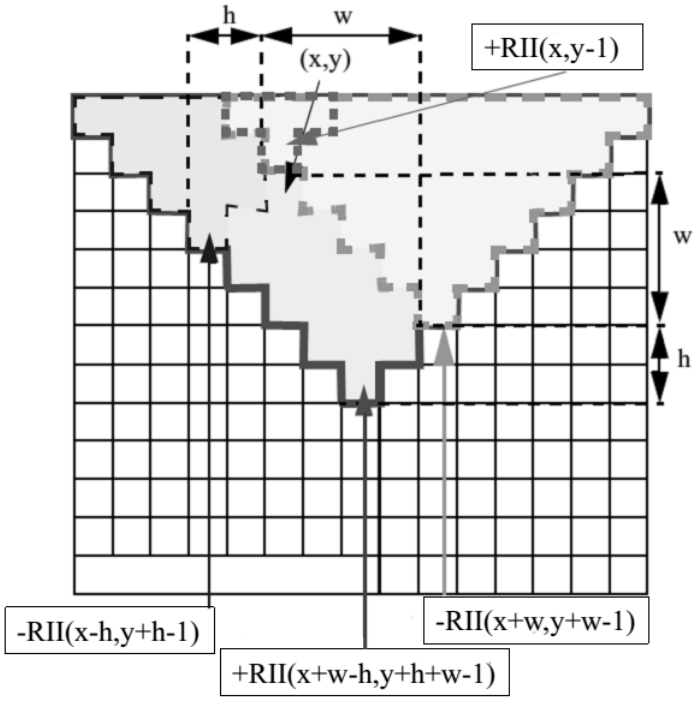


Figure 3.12 – Calculation scheme for sum of $45°$ rotated rectangle.

3.3.3 Internal features

As presented previously, in the original work of Viola-Jones [96], Haar-like features are extracted from the whole region of object of interest including background. In the case where the number of background pixels is important, Haar-like features characterizing background will contribute to learning the model of hand postures. As consequence, the detection could be missed or false when the background changes. Our objective is to

design a method for hand detection that is independent of background changing. We then introduce novel concepts of *Internal features* and *Internal Haar-like features* as follows.

Definition 1 (Internal features)**.** *An Internal feature is a feature extracted on a region inside the object of interest (without background).*

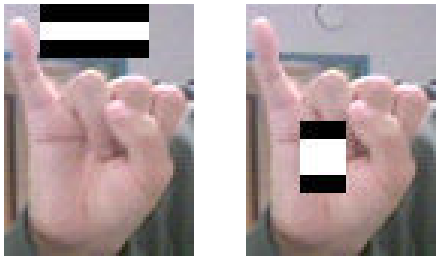
Remark 1. *As computed on a region inside the object, Internal features are invariant to background changes.*

Internal features are features that reflect in general only the own characters of the object but do not include characters of any another component like background.

3.3.4 Internal Haar-like features

Definition 2 (Internal Haar-like features)**.** *An Internal Haar-like feature is a Haar-like feature extracted on a region inside the object of interest (without background).*

Internal Haar-like features are Internal features. As a result, they have the same property of Internal features we present in Remark 1. Figure 3.13b shows an example of Internal Haar-like feature. This Haar-like feature is calculated based-on the difference

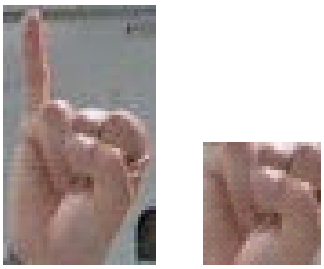


(a) (b)

Figure 3.13 – (a) An example of Haar-like features that is not an Internal Haar-like feature. (b) An example of Internal Haar-like features.

between the sum of the pixel values inside the black rectangles and the ones inside the white rectangles. Notice that all these rectangles are inside the hand area. Therefore, they do not suffer from background changing. Meanwhile, Fig. 3.13a is an ordinary Haar-like feature that is extracted from background region. The value of this Haar-like feature, in this case, will be changed when the background change.

We propose a technique to lead the trainer extract only features inside the hand region. In this technique, the positive sample is the approximate inscribed rectangle area of hand (Fig. 3.14b) instead of the approximate circumscribed rectangle's area of hand (Fig. 3.14a).



(a) (b)

Figure 3.14 – Two kinds of positive sample of a hand posture: (a) Traditional positive sample which is an *ACRH*; (b) Positive sample used in our system which is an *AIRH*.

Definition 3 (ACRH)**.** *An Approximate Circumscribed Rectangle's area of a Hand (ACRH) is an axis-aligned rectangle which contains the whole hand with background as little as possible.*

Definition 4 (AIRH)**.** *An Approximate Inscribed Rectangle's area of a Hand (AIRH) is an axis-aligned rectangle which contains the maximum hand region without background.*

Remark 2. *The Haar-like features extracted in the AIRH are Internal Haar-like features.*

Figure 3.15 illustrates the definitions of *ACRH*, *AIRH*, Haar-like and Internal Haar-like features. Figure 3.16 shows some examples of extracted Haar-like features in *AIRH*

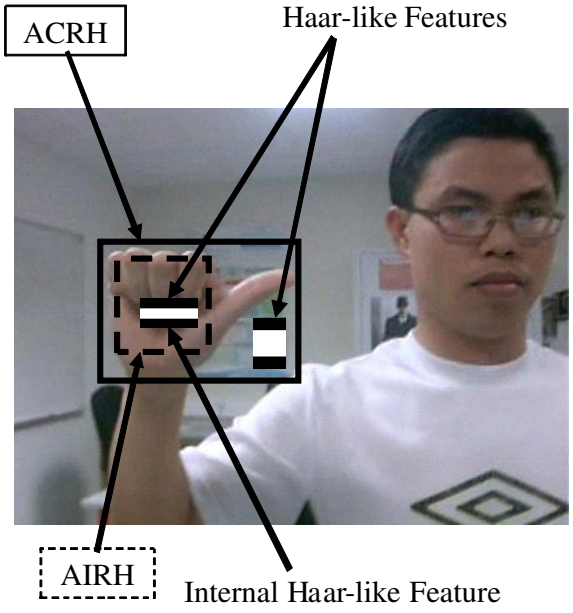


Figure 3.15 – Examples of Haar-like features extracted from *ACRH* and Internal Haar-like features extracted from *AIRH*

and *ACRH* of a positive sample.

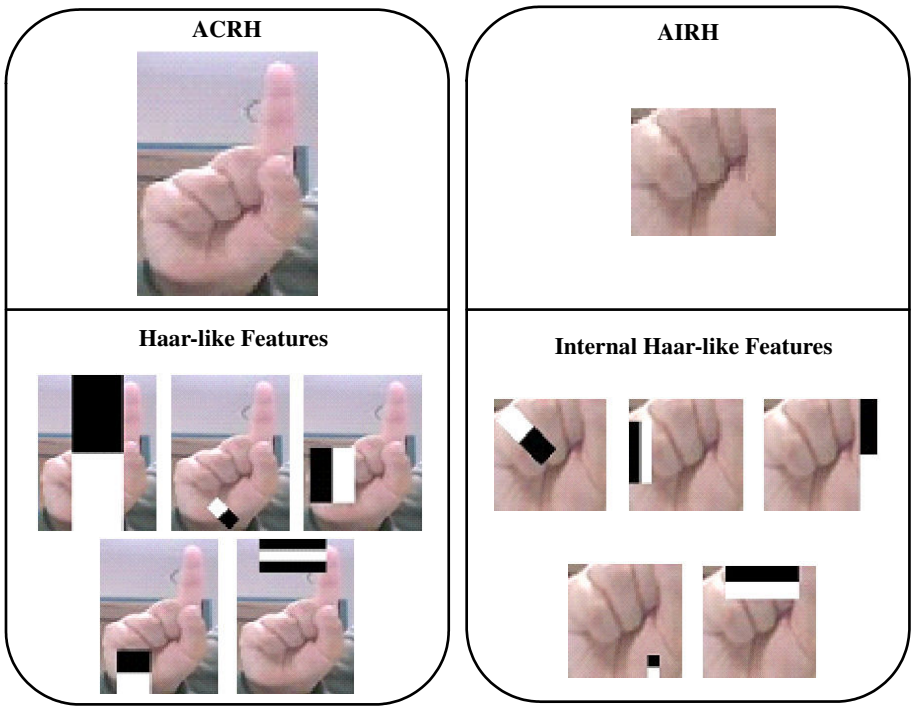


Figure 3.16 – Examples of Haar-like and Internal Haar-like features extracted from *ACRH* (left) and *AIRH* (right) respectively.

As we can see, a lot of Haar-like features have been extracted from the background of the *ACRH*. In contrast, all the features extracted from the *AIRH* is Internal Haar-like features. When we train a hand detector using a positive training dataset that is a set of *AIRH*s, then the detector will avoid the unexpected effect of background. However, the detector will detect *AIRH* that is an internal region of hand but not a region that contains whole hand region. All of the previous works, for example [42, 90], that applied Viola-Jones detector into hand detection or hand posture recognition use positive samples that are *ACRH*s, so the positive sample includes a lot of background. Figure 3.17 shows Haar-like features for 3 types of postures used in [90]. As we can see, there are many features found on background area. During the training process, some of them are chosen to build weak classifiers. This means the selected rectangle features remember characteristics of both background and hand. So, when hand has been detected, the detector may accept a background sub-window that is similar to the remembered characteristics. The detector also may reject foreground sub-window that is not similar to remembered character because of a dissimilar background. For this reason, to avoid the unexpected effect of background, we use Internal Haar-like features instead of traditional Haar-like features.

When we use *AIRH* positive samples to train Viola-Jones detector, the learner only

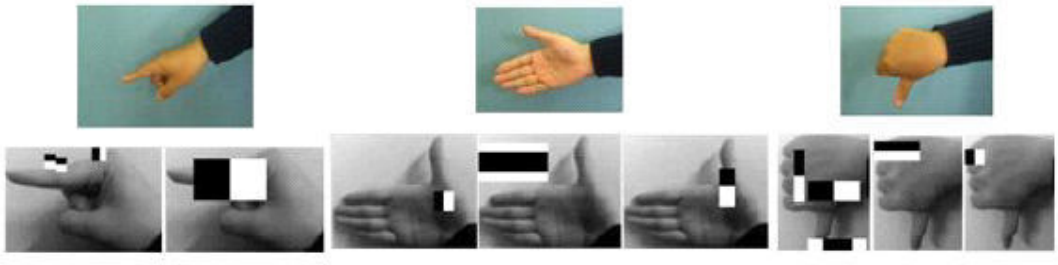


Figure 3.17 – Examples of Haar-like features for 3 types of postures used in [90]

remembers Internal Haar-like features. Therefore, the learner only remembers own characters of the hand postures which helps detector separating non-hand posture windows from hand posture windows. Therefore, the detector will not make confusion of the hand posture with the background. Of course, if there is a background sub-window that is very similar to the positive hand posture then the detector will make confusion of this background sub-window with a hand posture.

There is a problem that may occur in case of a particular hand posture has an internal region with poor characteristics, there are many background regions similar to the internal region of this hand posture. In this case, the detector will make more confusion between the hand and the background. This problem can be resolved in the further processing steps of the hand posture recognition system. According to the experimental results, in overall, the detector using Internal Haar-like features is better than the detector using traditional Haar-like features.

3.4 Classification

Hand detection in image can be formulated as a binary classification problem. All sub-windows of a given input image are considered. Each one is classified to hand or non-hand class. In general binary class problem, the input data for learning is N training examples $(x_1, y_1), ..., (x_N, y_N)$ where $x \in \Re^k$ and $y_i \in \{-1, 1\}$. x_i is a $k - dimension$ vector. Each element of x_i vector encodes a feature relevant for the learning task at hand. The desired two-class output is encoded as -1 and $+1$. In the case of hand detection, the input component x_i is one Haar-like feature. An output of $+1$ and -1 indicates whether the input pattern contains a complete instance of the object class of interest. We employ an object detection framework that was proposed in [96] by Viola-Jones to detect hand. However, instead of using Haar-like features, we use Internal Haar-like features.

Viola-Jones object detection framework uses a cascaded architecture of strong classi-

fiers. Each strong classifier is trained using learning algorithm AdaBoost that was introduced in [26]. AdaBoost algorithm is the improved Boosting algorithm that is proposed by the same authors. The next sections will present these algorithms.

3.4.1 Boosting

Boosting [23, 80] is one of the most important and popular approaches in machine learning. The main idea of boosting is combining a set of weak classifiers for constructing a more powerful classifier. The detail of Boosting algorithm is presented in Algorithm 1.

Algorithm 1: Boosting algorithm [80]

input : N examples $(x_1, y_1), ..., (x_N, y_N)$ with $x \in \Re^k, y_i \in \{-1, 1\}$
output: $H(x)$, a classifier suited for the training set

1 Randomly select, without replacement, $L_1 < N$ samples from Z to obtain Z_1; train weak learner H_1 on it.
2 Select $L_2 < N$ samples from Z with half of the samples misclassified by H_1 to obtain Z_2; train weak learner H_2 on it.
3 Select all samples from Z that H_1 and H_2 disagree on; train weak learner H_3.
4 Produce final classifier as a vote of weak learners $H(x) = sign\left(\sum_{n=1}^{3} H_n(x)\right)$.
5 **return** $H(x)$

Figure 3.18 shows graphically algorithm boosting. The input of Boosting algorithm

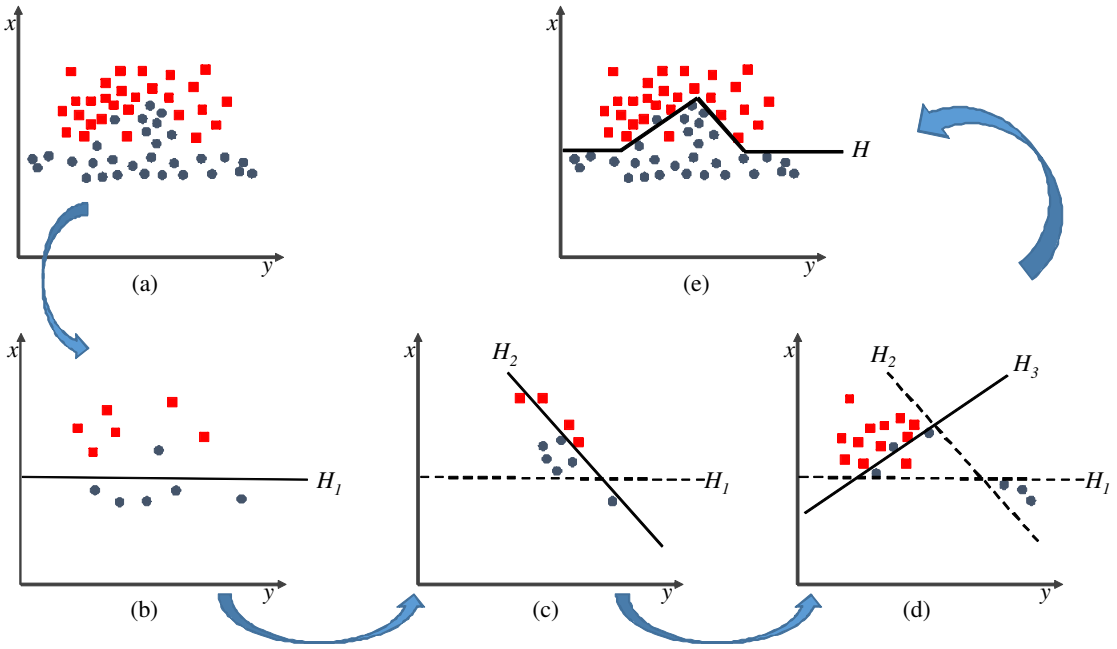


Figure 3.18 – Illustration of boosting algorithm

is illustrated in Fig. 3.18(a). The rectangular red points indicate examples which have a

label of 1, and the round blue points indicates examples which have a label of -1. The L_1 examples that is selected in Step 1 is illustrated in Fig. 3.18(b). The weak classifier H_1 that is trained on selected L_1 may makes many wrong classifications when it works on all the N input examples. The L_2 examples that is selected in Step 2 is illustrated in Fig. 3.18(c). The weak classifier H_2 that is trained on selected L_2 examples helps to classify a part of wrong classifications of H_1. The classifier H_3, in the last, is trained to give the decision in case of that H_1 and H_2 get different results, see Fig. 3.18(d). With this way of construction for classifiers, the final classifier H that produced in Step 4 is a strong classifier, see Fig. 3.18(e). The final strong classifier is constructed by combining three weak classifiers.

3.4.2 AdaBoost (Adaptive boosting)

Adaptive Boosting (AdaBoost) is also proposed by Freund and Schapire [26, 27] in 1996. The main idea of AdaBoost algorithm is graphically presented in Fig. 3.19. Figure 3.20

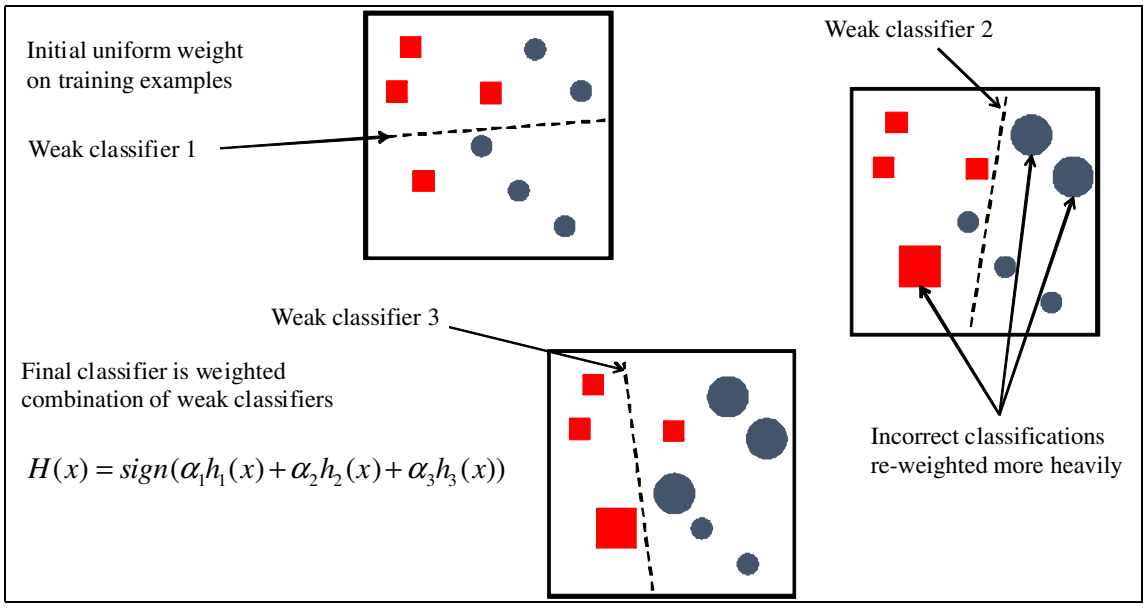


Figure 3.19 – Illustration of AdaBoost algorithm with number of weak classifiers $M = 3$.

shows the detail of this method. The idea of AdaBoost algorithm is also to combine weak classifiers to produce a strong classifier with better performance. Different from Boosting algorithm, in AdaBoost algorithm the adaptive weights of samples are updated during training such as further training process will focus on misclassified samples of previous trained weak classifier.

Figure 3.19 shows AdaBoost algorithm with the number of weak classifiers is 3. As shown in Fig. 3.19, the idea of AdaBoost is to focus on the difficult examples that are

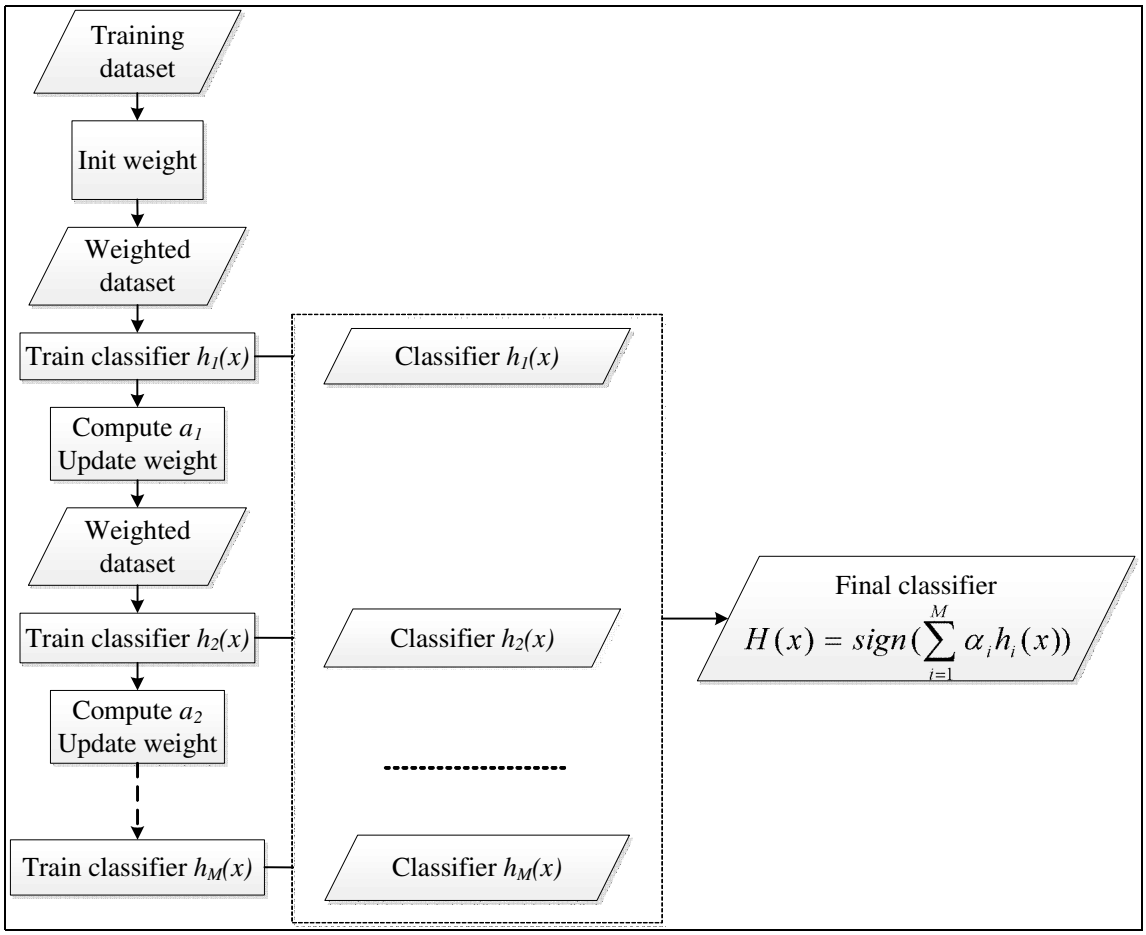


Figure 3.20 – The idea of the AdaBoost algorithm, adapted from [23]

misclassified by previous weak classifier during training process. To do this, the weights of the examples are updated sensibly. The weight of an example will be increased if this example is misclassified and will be decreased if it is correctly classify. In Fig. 3.19, the examples with greater value of weight is indicated by bigger circles. The weights of the examples are normalized so as the sum of them equals 1. The weights can be initialized with equal values. Many different variants of AdaBoost algorithm have developed that are Real AdaBoost, Gentle AdaBoost, Discrete AdaBoost, etc [23]. All of them are identical with respect to computational complexity from a classification perspective but differ in their learning algorithm. The Real AdaBoost algorithm performs exact optimization with respect to weak classifier $f_m(x)$. The Gentle AdaBoost algorithm improves it by using Newton stepping to provide a more reliable and stable ensemble. The Gentle AdaBoost algorithm uses weighted least-squares regression to minimize the function instead of fitting a class probability estimate. In [96], strong classifiers are trained by using Discrete AdaBoost algorithm. We chose Gentle AdaBoost instead for our learning algorithm because it has been shown to outperform other AdaBoost variants in [28, 51]. Gentle Adaboost

algorithm [26] is presented in Algorithm 2.

Algorithm 2: Gentle AdaBoost algorithm

input : N examples $(x_1, y_1), ..., (x_N, y_N)$ with $x \in \Re^k, y_i \in \{-1, 1\}$
output: $H(x)$, a classifier suited for the training set

1 Initial uniform weight on training examples: $w_i = \frac{1}{N}, i = 1, ..., N$
2 **for** $m = 1$ **to** M **do**
3 (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weight w_i
4 (b) Set $w_i \longleftarrow w_i.exp(-y_i.f_m(x_i)), i = 1, ...N$, and re-normalize weights so that $\sum_i w_i = 1$
5 **end**
6 **return** $H(x) = sign[\sum_{m=1}^{M} f_m(x)]$

3.4.3 Cascade of classifiers

The cascade of classifiers structure (Fig.3.21) is employed to speed up the performance. Each stage of the cascade is a strong classifier that consists of a bunch of weak classifiers.

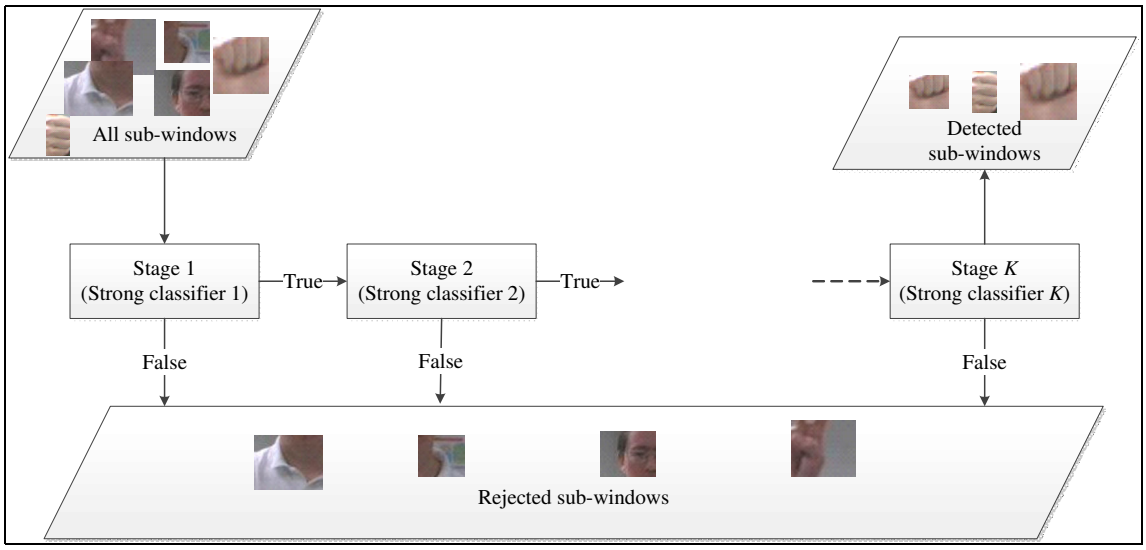


Figure 3.21 – The cascade of classifiers structure

A strong classifier is trained to detect most of the positive samples whereas rejecting a certain fraction of negative samples.

A cascade of classifiers is a degenerated decision. In the detection phase, a sub-window will be rejected if there is any stage (strong classifier) in the cascade rejects this sub-window. A sub-window will be accepted as hand if it is passed all stages. Each

sub-window is processed in the first stage. If the first stage accepts the sub-window, then it is processed in next stage. Otherwise, no any process is performed, and the first stage processes next sub-window.

Suppose that K is the number of stages of trained cascade of classifiers; f_i and d_i are the false positive rate and the detection rate of the i_{th} stage respectively. The false positive rate F of the cascade then is

$$F = \prod_{i=1}^{K} f_i \tag{3.13}$$

and the detection rate is

$$D = \prod_{i=1}^{K} d_i \tag{3.14}$$

For example, each stage is trained to eliminate 50% of the non-hand patterns whereas incorrectly eliminating only 0.1% of the hand patterns; 20 stages are trained. We then will expect an overall false alarm rate about $0.5^{20} \approx 9.5e-07$ and an overall hit rate about $0.999^{20} \approx 0.98$.

3.5 Experiments

The aim of the experiments is to evaluate the advantage of our detector using Internal Haar-like features. To do this, we collect a dataset that is suitable for our application goal. We then train two detectors that are our detector and the traditional Viola-Jones detector. Performances of these two detectors are compared to evaluate the advantages of proposed Internal features.

3.5.1 Dataset and evaluation measure

L3i-MICA hand posture dataset

As described in Chapter 1, we would like to develop a system that recognize hand postures for human-machine interaction application in the indoor environment. Therefore, the dataset for hand detection evaluation will be prepared with regard to this context. We collect our own dataset (L3i-MICA hand posture dataset) from a representative application that is human-robot interaction system.

The robot in this situation is an assistant robot in library or museum. They stay in the reception area. The robot may move around their area. However, they do not move during interacting with the user. The user will stand face to face in front of the robot during interaction using hand postures. The distance from the robot to the user is around 1m to 3m, see Fig. 3.22. All videos are collected in L3i laboratory with a natural fluorescent

Figure 3.22 – The illustration of the setup of capturing dataset.

lighting condition. The background is naturally cluttered. We use a Hercules Deluxe Optical Glass webcam with the default resolution of 320×240 pixels and default frame rate of 30 frames per second.

The easiest and most comfortable way to give a command by hand postures is normally raising right hand and play a hand posture. According to our investigation result with 10 subjects, people feel comfortable to play 21 postures as shown in Fig. 3.23. The easiness for playing each of 21 postures is different. In all postures, the hand point up except posture 4.

The dataset was obtained from 10 subjects in L3i laboratory who come from France, Canada, Japan, and Vietnam. The subjects include 6 males and 4 females. Each person was asked to play the 21 hand postures 4 times at different positions in a room. The length of each video is about 4 seconds. Each video contains one hand posture of one person. The total number of videos is $21 \times 10 \times 4 = 840$. The dataset was divided into two parts. Each part includes 2 videos of every subject. One part was used for training, and another one was used for testing. The information of MICA-L3i dataset is shown in Table 3.1.

Our dataset is challenging at some points:

- The number of hand postures in MICA-L3i dataset is largest (21) in comparison with some other public datasets [43,58,75,91,92]. Table 3.2 shows the comparison between MICA-L3i dataset and previous published datasets.

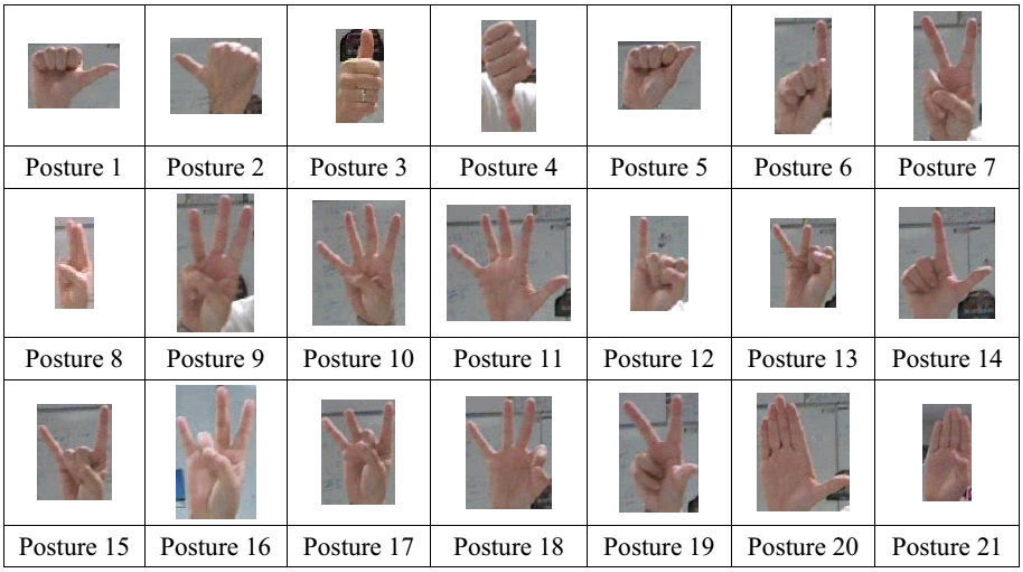


Figure 3.23 – List of 21 upright right-hand postures

Table 3.1 – Information of collected dataset.

Attribute	Value
Number of subjects	10
Number of postures	21
Image resolution	320 x 240
Frame rate	30
Length of each video	about 4 seconds
Number of videos for training: Set 1	420
Number of videos for testing: Set 2	420

Table 3.2 – Comparison of datasets

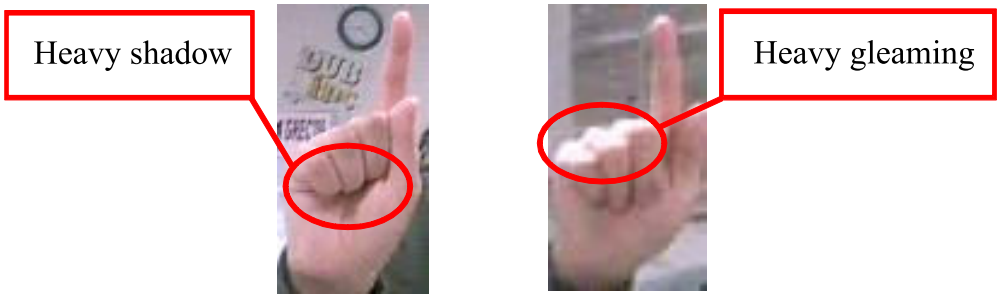
Dataset	#Postures	#Subjects	Background
Jochen Triesch I [91]	10	24	3 backgrounds (light, dark, complex)
Jochen Triesch II [92]	12	N/A	3 backgrounds (light, dark, complex)
Sebastien Marcel [58]	6	10	uniform and complex
NUS I [43]	10	N/A	uniform
NUS II [75]	10	40	complex
Our dataset	21	10	complex

- In MICA-L3i dataset, lighting changes from position to position of robot and user in the room. These illumination changes leads to changes in hand color. Fig.3.24 shows an example in which hand colors changes significantly due to the lighting change.

(a) (b) (c)

Figure 3.24 – An example of the variety of hand colors in MICA-L3i dataset.

- The different positions in a room and the changes in hand pose make different gleaming parts and shadow in the hand. Figure 3.25(a) shows a heavy shadow at the hand center while Fig. 3.25(b) illustrates heavy gleaming part on the top of the hand.



(a) (b)

Figure 3.25 – An example of the variety of gleaming parts and shadows in the hand.

- The rotation angle of the hand also has some difference (see Fig. 3.26).

(a) (b)

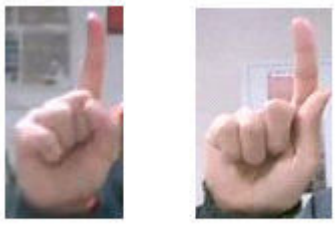
Figure 3.26 – An example of the variety of rotated hand poses in MICA-L3i dataset.

- The distances from human to robot is not fixed. We just ask people to stay in front of robot so as feeling comfortable for interacting face to face with robot. The differences of distances between user and robot cause the differences of the size of hand in the images. We can see an example in Fig. 3.27.

(a) (b)

Figure 3.27 – An example of the variety of scale.

- Each person performs a hand posture differently. For example with posture number 6, some people often place the thumb over the middle finger (see Fig. 3.28(a)) while some other place next to the point finger (see Fig. 3.28(b)).



(a) (b)

Figure 3.28 – An example of the variety of the ways for playing the same hand posture.

- The similar between postures also cause difficulty for hand postures classification methods. For example, figure 3.29(a) is posture 14 that is quite similar to posture number 6 3.29(b) in opening the index finger. The difference of posture 14 and posture 16 is just in the thumb. The thumb in posture 14 is open while the thumb in posture 6 is close. Posture number 12 (see Fig. 3.29(c)) is also similar to posture 6. Both these posture has only one open finger.

- In natural indoor environment, the background is often cluttered that makes difficulties in hand detection and posture recognition.

We prepared positive samples for training by uniform sampling on the set of training videos to make a set of 10.000 frames. We then cropped manually to create a set of

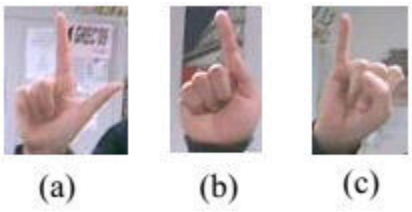


Figure 3.29 – An example of similar postures.

10.000 *AIRH* samples for training our proposed hand detector and a set of 10.000 *ACRH* samples for training traditional Viola-Jones detector for hand detection.

The set of 10.000 negative samples was collected by adding 5020 non-hand frames that captured in an L3i room into the set of 4980 images downloaded from. *http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/*

Evaluation measure

To evaluate the performance of the detector, we used Precision, Recall, and F-measure factors. These factors were defined in [69].:

$$\begin{aligned} Precision &= \frac{tp}{tp+fp} \\ Recall &= \frac{tp}{tp+fn} \\ F &= 2\frac{Precision * Recall}{Precision + Recall} \end{aligned} \tag{3.15}$$

where tp is true positive that means correct detection, fp is false positive that means unexpected result, fn is false negative that means missing result. Jaccard index is employed to determine whether a detection is correct or not. It is computed as follows [60]:

$$JI = \frac{RECT_r \cap RECT_{gt}}{RECT_r \cup RECT_{gt}} \tag{3.16}$$

where $RECT_r$ is result detection rectangle, $RECT_{gt}$ is ground truth rectangle. Our detector is trained to detect $AIRH$, so ground truth for evaluation of our detector is $AIRH$. Meanwhile, ground truth for evaluation of traditional Viola-Jones detector is $ACRH$.

A detection is correct if the value of JI is greater than or equal 50%.

3.5.2 Training detectors

We train two detectors in the same configuration but using two different sets of positive samples presented in section 3.5.1. In each stage of the cascade of classifiers, we choose minimum desired hit rate at 99.5%, and maximum desired false alarm at 50%. We then choose maximum number of stages $\#stages_{max} = 25$. We would like to show investigation results of 25 stages because our experiments show that the optimal number of stages is often met at around 20. All the training samples are resized to a standard size. Rainer Lienhart *et al.* [51] indicated that the sample size of 20×20 is optimal in case of square sample. To determine the standard size of training sample, we computed the average ratio between width and height of all the 10.000 positive samples. The standard size of samples was then selected in such a way that $width$ or $height$ is closest to 20 pixels as well as the size is best fit with the computed average ratio. We determined the standard size for internal hand center sample is 20×20, and whole hand sample is 21×28. The configuration of training process is shown in Table 3.3. Each attribute in both detectors has similar value.

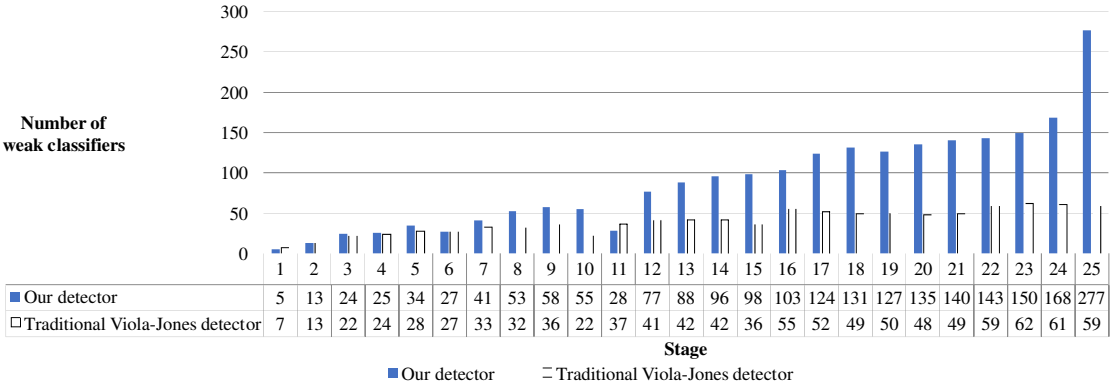
Table 3.3 – Configuration of training process

	Our detector	Traditional Viola-Jones detector
Number of positive samples	10000	10000
Number of negative samples	10000	10000
Size of sample (Width x Height)	20×20	21×28
Minimum desired hit rate of each stage	99.5%	99.5%
Maximum desired false alarm of each stage	50%	50%
Maximum number of stages	25	25

This helps us to compare fairly Internal Haar-like features with traditional Haar-like features.

Both training processes stop when the number of stages is 25. Figure 3.30 shows numbers of weak classifiers in each stage of trained classifiers. Our detector needs more weak classifiers than the traditional Viola-Jones detector. This is because the information in $ACRH$ is richer than $AIRH$. However, the rich information of $ACRH$ contains a lot of background information. Consequently, the traditional Viola-Jones detector will give more mistakes than our detector when the background changes. If we collect all the variants of background for training process, the traditional Viola-Jones detector will work well. However, to collect all variants of background is very expensive and unfeasible.

The frequency of occurrence of Haar-like feature prototypes in the trained cascades is shown in Fig. 3.31. These frequencies of occurrence in two cascades are similar. These figures give recommendations if we want to re-train a hand detector in the future. When



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Our detector	5	13	24	25	34	27	41	53	58	55	28	77	88	96	98	103	124	131	127	135	140	143	150	168	277
Traditional Viola-Jones detector	7	13	22	24	28	27	33	32	36	22	37	41	42	42	36	55	52	49	50	48	49	59	62	61	59

Figure 3.30 – Numbers of weak classifiers in each stages.

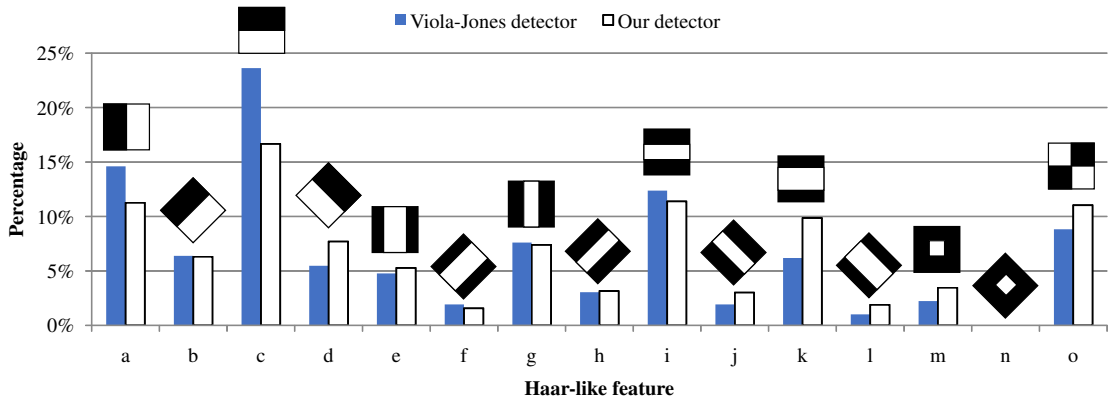


Figure 3.31 – The frequency of occurrence of Haar-like feature prototypes.

we want to remove some unimportant Haar-like feature prototypes in order to get a smaller set of features, we can remove some ones that have the lowest frequency of occurrence such as prototype *f,l,n*, see Fig. 3.4(f,l,n). We should remove prototype *n* because it is not chosen completely. Meanwhile, we should keep prototype *o* (see Fig. 3.4(o)) that is not used in [50] due to their comment that is the prototype *o* can be well approximated by prototype *h* and *j*. We observe that prototype *o* can represent the characteristic of diagonal lines like *h*, *j*, and it simultaneously presents the characteristic of large blocks arranged in diagonal patterns that can not be represented by prototypes *h* and *j*. We also see that the most used prototypes are simple such as prototype *a* and *c*.

3.5.3 Results

The results of our detector and Viola-Jones detector are shown in Table 3.4. The data in this table is obtained by changing the number of stages from 5 to 25. The performance of

Table 3.4 – Detection results with different numbers of stages

Number of stages	Traditional Viola-Jones detector			Our detector		
	Precision	**Recall**	**F**	**Precision**	**Recall**	**F**
5	0.00	0.14	0.01	0.02	0.72	0.04
6	0.01	0.19	0.01	0.03	0.90	0.06
7	0.01	0.31	0.03	0.04	0.97	0.08
8	0.02	0.37	0.04	0.05	0.99	0.10
9	0.03	0.45	0.06	0.07	0.99	0.12
10	0.05	0.52	0.10	0.09	0.99	0.16
11	0.08	0.57	0.14	0.10	0.99	0.17
12	0.11	0.62	0.19	0.14	0.98	0.25
13	0.16	0.66	0.26	0.20	0.97	0.33
14	0.22	0.68	0.33	0.25	0.96	0.40
15	0.28	0.69	0.39	0.38	0.93	0.54
16	0.35	0.70	0.47	0.49	0.90	0.63
17	0.44	0.70	0.54	0.68	0.86	0.76
18	0.50	0.70	0.59	0.79	0.84	0.81
19	0.61	0.67	0.64	0.87	0.81	0.84
20	0.68	0.66	0.67	0.93	0.77	**0.84**
21	0.74	0.65	**0.69**	0.97	0.73	0.83
22	0.76	0.63	0.69	0.99	0.69	0.81
23	0.82	0.59	0.69	0.99	0.66	0.79
24	0.85	0.54	0.66	1.00	0.61	0.76
25	0.87	0.50	0.63	1.00	0.56	0.72

a detector having less than 5 stages makes no sense because the detector in this case has a lot of wrong detection as well as loses a lot of true positive windows. Fig3.32 and Fig.3.33 give an intuitive view of Table 3.4. We could see that our detector gets the best result (in term of F-score) at the number of stages being 20 while traditional Viola-Jones detector gets the best result at the number of stages being 21. To compare the performances of two detectors, we draw the Precision-Recall curves that reflect the relationship between Precision and Recall values of two detectors (see Fig. 3.34). The Precision-Recall curves show that our detector is better than traditional Viola-Jones detector in overall. In addition, Fig.3.35, Fig.3.36, and Fig.3.37 give an intuitive view of comparison between our detector and traditional Viola-Jones detector on individual factors (Precision, Recall, and F-score).

We can see that at each number of stages, our detector outperforms traditional Viola-Jones detector in term of all three factors Precision, Recall, and F-Score.

The reason is that our detector avoids unexpected effects of background thanks to the use of Internal Haar-like feature while traditional Viola-Jones detector meets this unexpected effect. Examples 1 and Examples 2 in Fig.3.38 illustrate the unexpected effect of background on the traditional Viola-Jones detector. These examples also illustrate our

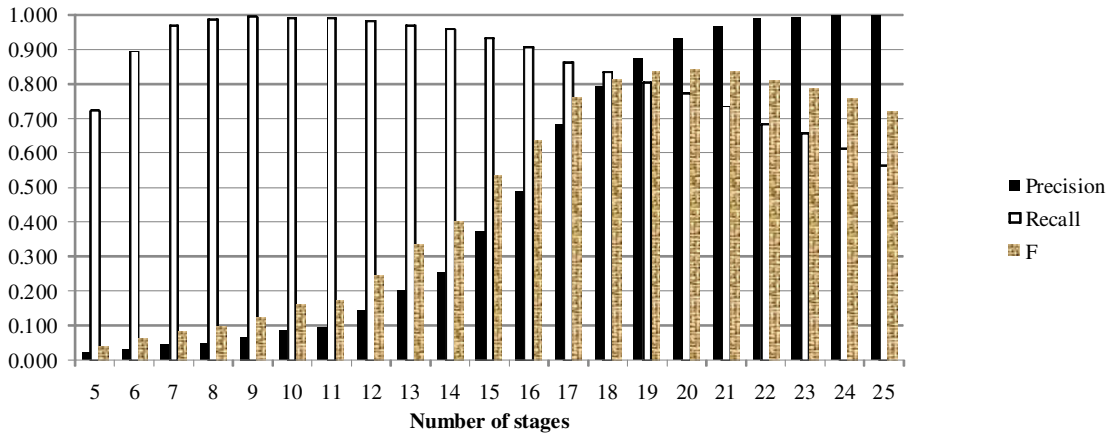


Figure 3.32 – The chart of the results of our detector.

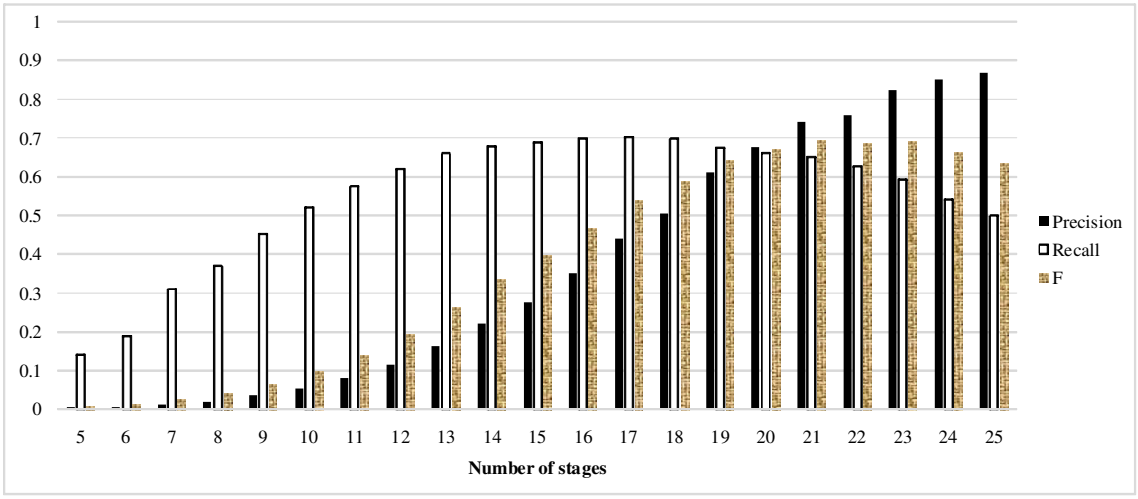


Figure 3.33 – The chart of the results of traditional Viola-Jones detector.

improvement in avoiding this unexpected effect.

However, there are some specific cases in which the Viola-Jones detector works better than ours. Look at Example 3 in Fig.3.38, our detector gives a false positive that is a sub-window in the arm area because this sub-window is similar to *AIRH* of the wrong recognized posture. While, traditional Viola-Jones detector does not make this confusion in this example. The reason is that with traditional Viola-Jones, the sub-window on the arm is not similar to *ACRH*s because *ACRH*s contain a lot of information of background as well as hand within fingers. In case of Example 4 in Fig.3.38, our detector does not detect the hand while the traditional Viola-Jones works well. The cause of this result is that the characteristic of the *AIRH* of this posture is not enough to distinguish the hand from the background. Meanwhile, the *ACRH* contains more information of both the background and the whole hand that helps the detector distinguish the *ACRH* from other regions. In

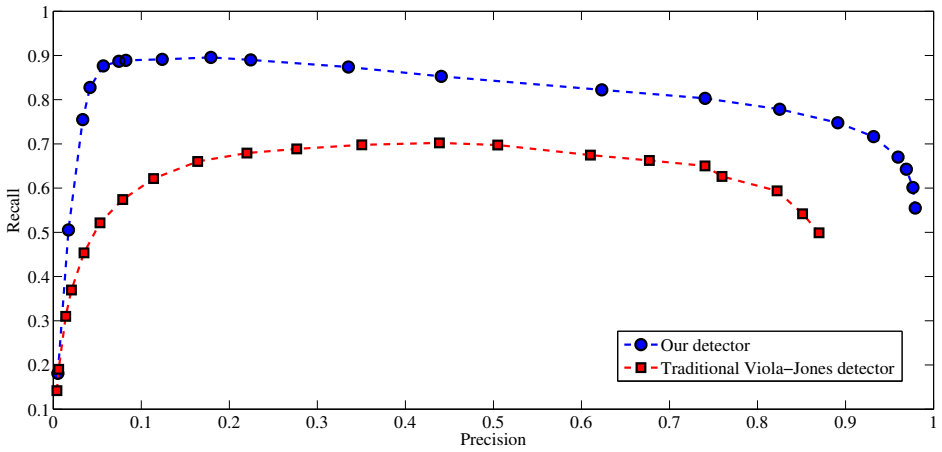


Figure 3.34 – Precision-Recall curves for comparison between our detector and traditional Viola-Jones detector.

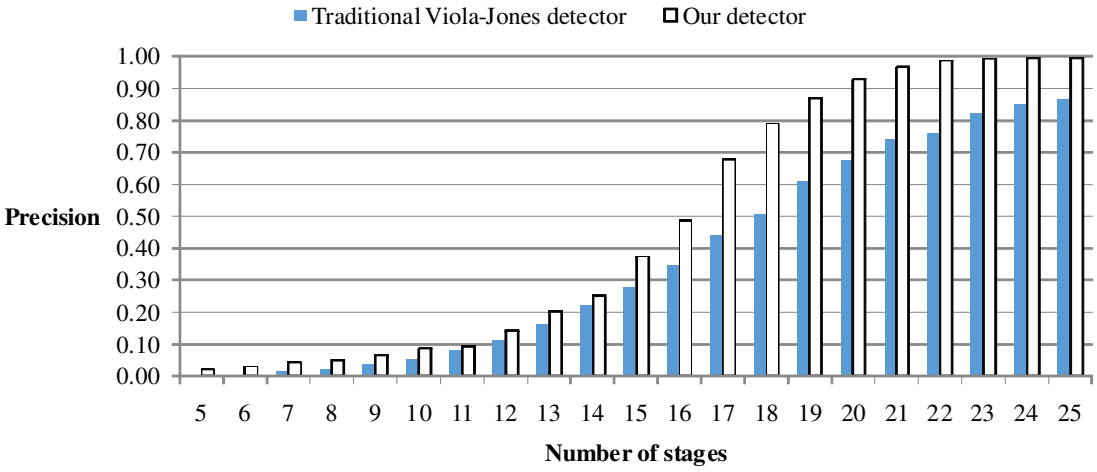


Figure 3.35 – Comparison on Precision

addition, the background in this frame similar to the background in some *ACRH* training positive samples.

In the case where the working environment of hand detection system is defined before training the detector, we can use traditional Viola-Jones detector to get a better result. In this situation, features extracting on background should be selected during training. Conversely, if we do not know the environment of the system, we can improve the performance of our detector by using higher resolution camera in order to get more detail information of the hand.

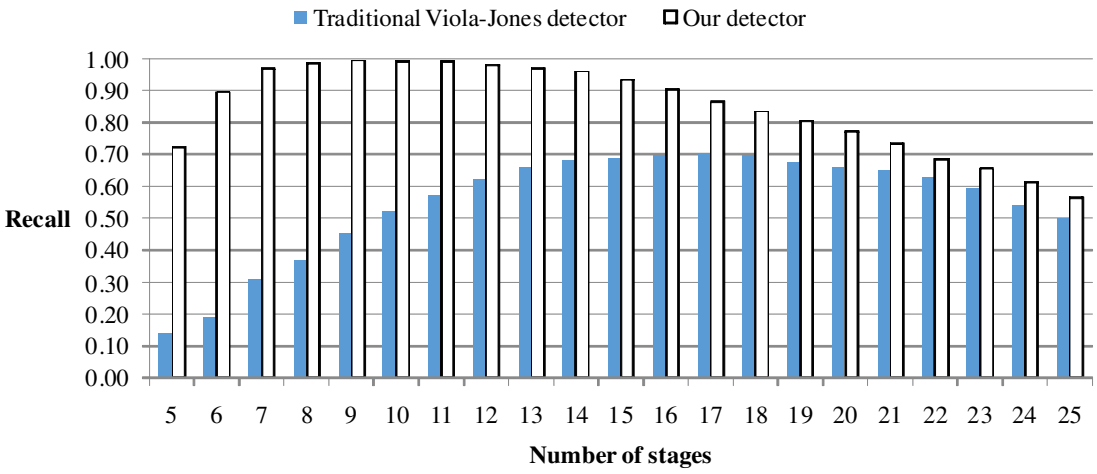


Figure 3.36 – Comparison on Recall

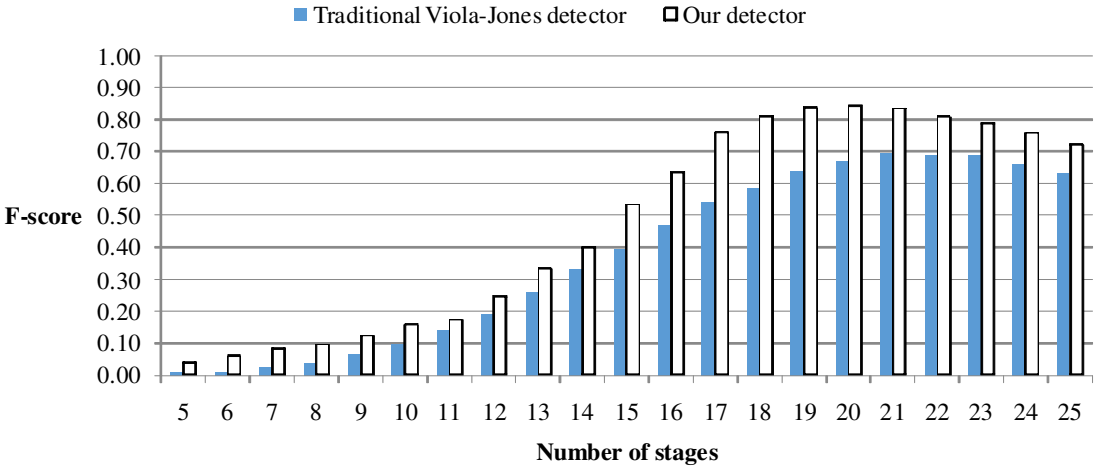


Figure 3.37 – Comparison on F score

3.6 Conclusions

This chapter presents our work on hand detection that employs Viola-Jones object detection framework. Our main contributions are summarized as follows:

- We have introduced a novel concept of Internal features in general and specifically Internal Haar-like features. The detector using internal features will avoid the effect of background changing. We have shown that Internal Haar-like features outperforms Haar-like features in the framework of hand detection of Viola-Jones in term of detection rate while has similar computation time (real-time). The internal features could be applied in any context of object detection.

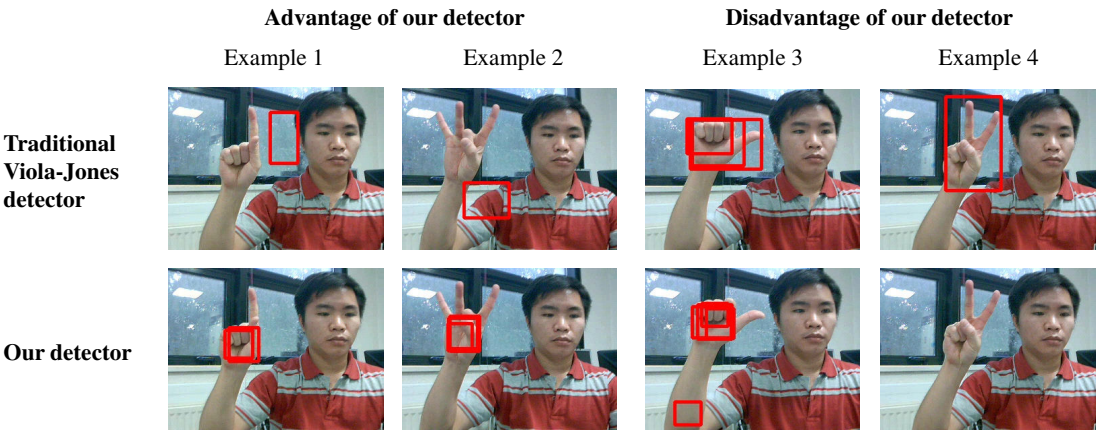


Figure 3.38 – Examples of advantages and disadvantages of our detector.

- We have built a very challenging dataset of hand postures in term of number of hand postures, cluttered background, pose and lighting changes. Our dataset is in the context of human-robot interaction. This dataset could be published for research purpose.
- We have performed extensive experiments on two detectors: Viola-Jones detector and our detector. We analyzed in detail the size of positive samples, the number of stages as well the Haar-like features frequencies. This makes a recommendation when re-training a detector.

Chapter 4

Hand postures recognition

4.1 Introduction

As we analyze in Chapter 2, the explicit hand representation approach requires a good hand segmentation. However, in the context of our work, the background is complicated, so it is very difficult to segment perfectly hand region. In this case, the implicit hand representation is a more suitable approach. The analysis in Chapter 2 also indicate some limitations of the previous works dedicated to implicit hand representation approaches.

Recently, Liefeng Bo *et al*. [5] proposed a descriptor for generic object representation named kernel descriptor (KDES). The authors have proved that KDES outperforms the state-of-the-art methods on different benchmark datasets such as CIFAR-10, Caltech-101, ImageNet. Therefore, we propose to apply KDES for hand posture recognition. The experimental results have shown that KDES is a robust descriptor for hand posture recognition problem [67]. Moreover, among different kernels, the gradient is the best for our problem. However, while working with KDES for hand posture recognition, we have observed several limitations. We will point out the limitations of the KDES for hand posture recognition and our improvements.

- *Patch-level features are not invariant to rotation*: At patch level, the original KDES computes the gradient based features without considering the orientation. For this reason, the generated features are sensitive to rotation. So, we propose to compute the dominant orientation of the patch and normalize all gradient vectors in the patch to this orientation. Patch-level features will thus be invariant to rotation.

- *KDES is not invariant to scale*: The original KDES computes features over patches of fixed size. At two scales, the number of patches to be considered and the corresponding patch descriptions will be different. We propose a strategy to generate patches with adaptive size. This produces the same number of the extracted patches.

As a consequence, image-level feature is invariant to scale change.

- *KDES is not suitable to the specific structure of the hand*: At the image level, the original KDES organizes a spatial pyramid structure of patches to build the final description of the image. However, the hand has its own specific structure. We then design a new pyramid structure that better represents the structure of the hand.

To evaluate the proposed method, we use four datasets (Triesch dataset [91], NUS II dataset [75], our dataset, and a dataset we collected from [9]). We perform different experiments in order to demonstrate the recognition performance according to each proposed improvement, and compare with the state-of-the-art methods.

The remaining of this chapter is organized as follows. The section 4.2 presents the framework of proposed hand postures recognition based on KDES. The section 4.3 describes in detail the proposed method. The experimental results are presented in the section 4.4. The conclusions and future works are given in the section 4.5.

4.2 The framework of hand posture recognition

The proposed framework of hand posture recognition using kernel is presented in Fig. 4.1. It comprises two main steps: Hand posture representation and Hand posture recognition.

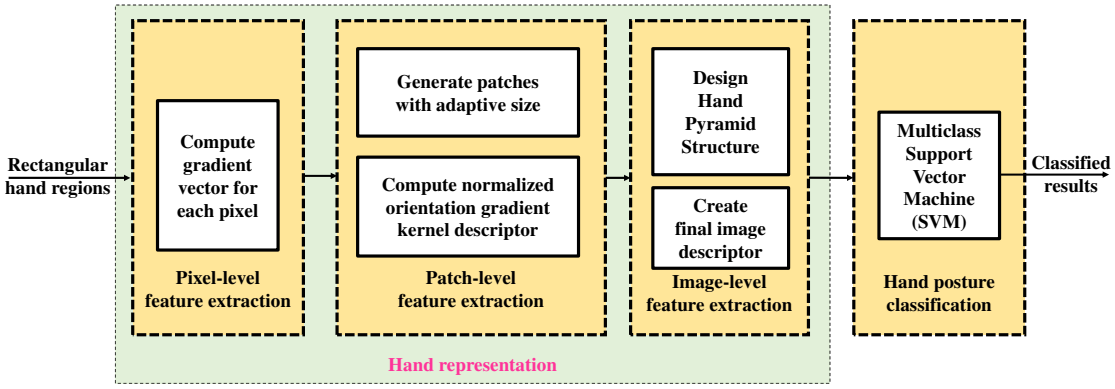


Figure 4.1 – The framework of proposed hand posture recognition method.

- *Hand posture representation*: This step takes a hand region image (from now on called *image*, for short) as input and returns a descriptor of the hand candidate. It is composed of multiple sub-steps:
  - Pixel-level feature extraction: At this level, a gradient vector is computed for each pixel of the image.

  - Patch-level feature extraction: At this level, we firstly have to generate a set of patches then compute patch-level features. Different from [5], depending on image resolution, we create patches with adaptive size instead of fixed size. This adaptive size ensures the number of patches to be considered unchanged. In addition, it makes the patch descriptor more robust to scale change. For each patch, we compute patch features as follows. Given an image patch, we compute a gradient descriptor based on the original idea proposed in [5]. However, unlike [5], we first compute the dominant orientation of the patch, then normalize all gradient vectors to this orientation. This normalization is done inside the gradient kernel allowing the descriptor to be invariant to rotation.
  - Image-level feature extraction: At this step, we propose a modification with respect to [5]: To combine patch features, we propose a pyramid structure *specific to hand postures* instead of a general pyramid structure. This specific pyramid structure makes the descriptor more suitable for hand representation. Given an image, the final representation is built based on features extracted from lower levels using efficient match kernels (EMK) proposed in [5]. First, we have to compute the feature vector for each cell of the hand pyramid structure, and then concatenate them into a final descriptor.

- *Hand posture classification*: Once the hand is represented by a descriptor vector, any classifier could be applied for the classification task. In this paper following the strategy originally proposed [5], we will use Multi-class SVM. In the following sections, we focus to present in detail the successive steps in hand representation.

In the following sections, we present the detail of hand presentation.

4.3 Hand representation

4.3.1 Extraction of pixel-level features

According to [5] and [67], a number of features can be computed at the pixel level, such as pixel values, texture, and gradient. In [67], we argued that gradient is the best feature for hand posture recognition; therefore, we use the gradient at pixel level. We will present pixel values, texture, and investigation results in Section 4.4.

With an input image, we firstly compute the gradient vector at each pixel of the image. The gradient is computed by the same gradient computation as used for SIFT [54]. The gradient vector at a pixel z is defined by its magnitude $m(z)$ and orientation $\theta(z)$. In [5],

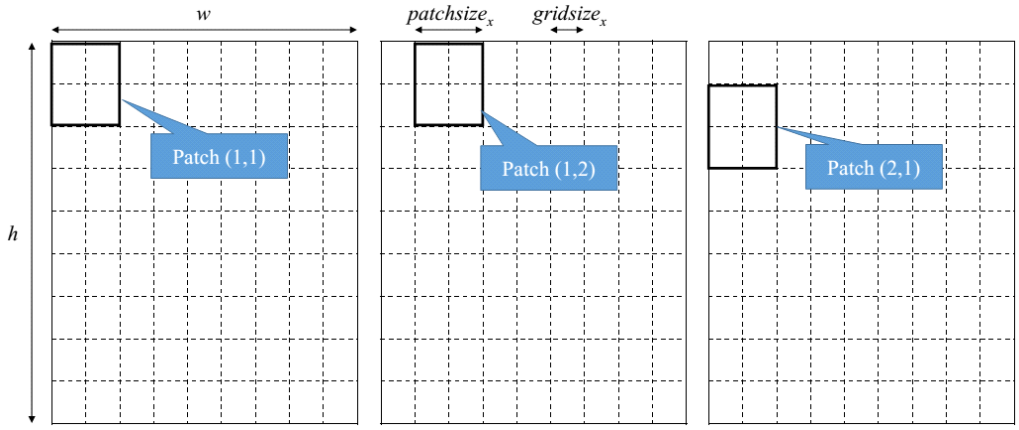


Figure 4.2 – Adaptive patch with $np_x = np_y = 8$, therefore $ngrid_x = grid_y = 9$.

the orientation $\widetilde{\theta}(z)$ is defined as follows:

$$\widetilde{\theta}(z) = [sin(\theta(z))\ cos(\theta(z))] \tag{4.1}$$

4.3.2 Extraction of patch-level features

Generate a set of patches with adaptive size from an image

In the original KDES [5], the author generated patches with a fixed size for all images in the dataset. These images can have different resolutions. For low-resolution images, the number of generated patches can be limited, producing a poor representation of the image. Besides, the feature vectors of two images of the same hand posture at two scales can be highly different. Consequently, the original KDES is not invariant to scale change.

Fig. 4.3(a,b) illustrates this problem. Fig. 4.3(a) and (b) are two images of the same hand posture at two scales. Fig. 4.3(a) has a size of 40×56 while Fig. 4.3(b) is two times bigger (64×96). When we use a uniform patch of size 16×16 and uniform grid 8×8, Fig. 4.3(b) has 77 patches while Fig. 4.3(a) has only 24 patches. A patch of Fig. 4.3(a) contains more real area of hand than a patch of Fig. 4.3 (b). Obviously, the feature vectors of patches are very different.

The above analysis motivates us to make an adaptive patch size in order to get a similar number of patches along both horizontal and vertical axes.

Figure 4.2 shows the description of the proposed adaptive patch technique. Suppose that the given number of patches is $np_x \times np_y$ (np_x patches along the horizontal axis and np_y patches along the vertical axis). The number of grid cells $ngrid_x \times ngrid_y$ is defined as: $ngrid_x = np_x + 1, ngrid_y = np_y + 1$. With an image has size of $w \times h$, the adaptive grid cell size along horizontal axis $gridsize_x = \frac{w}{ngrid_x}$ and the adaptive grid cell size along vertical axis $gridsize_y = \frac{h}{ngrid_y}$. The adaptive patch has the size of

$patchsize_x \times patchsize_y$ where $patchsize_x = 2gridsize_x$ and $patchsize_y = 2gridsize_y$. A patch is constructed from 4 cells of the grid. The overlap of two adjacent patches along the horizontal or vertical axes is a region of two cells of the grid. By this way, the size of the patches is directly proportional to the size of the image.

Fig.4.3(b,c) illustrates the advantage of the proposed adaptive patch. Different from

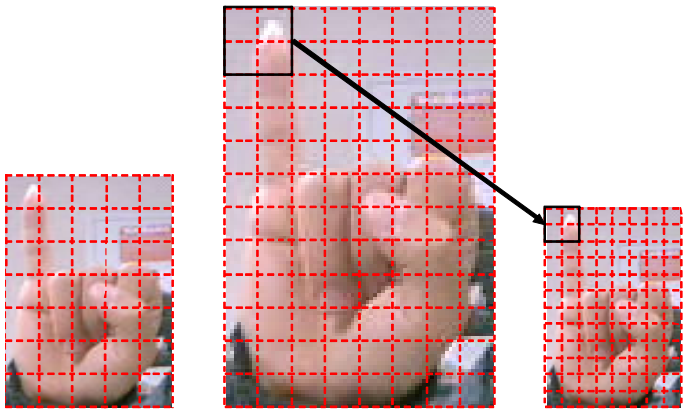


Figure 4.3 – An example of the uniform patch in the original KDES and the adaptive patch in our method. (a,b) two images of the same hand posture with different sizes are divided using a uniform patch; (b, c): two images of the same hand posture with different sizes are divided using the adaptive patch.

Fig.4.3(a,b), we apply the adaptive patch into two image in Fig.4.3(a,b). Fig.4.3(a,b) are two different scales of the same hand posture. With this, we obtain the same number of patches along x axis and y axis. The information in a patch of Fig.4.3(b) is similar to that of the corresponding patch Fig.4.3(c). Therefore, the sets of patch level feature vectors of image in Fig.4.3(b) and (c) are similar. That makes the image level features similar. This means our representation is invariant to scale changes.

Compute patch-level feature

Patch-level features are computed based on the idea of the kernel method. Derived from a match kernel representing the similarity of two patches, we can extract the feature vector for the patch using an approximate patch-level feature map, given a designed patch level match kernel function.

The gradient match kernel is constructed from three kernels that are gradient magnitude kernel $k_{\tilde{m}}$, orientation kernel k_o and position kernel k_p. In [5], gradient match kernel is defined as follows:

$$K_{gradient}(P, Q) = \sum_{z \in P} \sum_{z' \in Q} k_{\tilde{m}}(z, z') k_o(\tilde{\theta}(z), \tilde{\theta}(z')) k_p(z, z') \tag{4.2}$$

where P and Q are patches of two different images that we need to measure the similarity. z and z' denote the 2D position of a pixel in the image patch P and Q. $\theta(z)$ and $\theta(z')$ are gradient orientations at pixel z and z' in the patch P and Q respectively.

Directly using the gradient orientation $\widetilde{\theta}(z)$ in orientation kernel, the patch level features extracted from the match kernel will not be invariant to rotation. We hence propose to normalize gradient orientation before applying in match kernel. Specifically, inspired by the idea of SIFT descriptor [54], we compute a dominant orientation of the patch and normalize all gradient vectors to this orientation. We propose two ways to determine the dominant orientation $\overline{\theta}(P)$ of the patch P. First, we use the dominant orientation of the patch as proposed in [54]. Second, we compute a vector sum of all the gradient vectors in the patch. The normalized gradient angle of a pixel z in P thus becomes:

$$\omega(z) = \theta(z) - \overline{\theta}(P) \tag{4.3}$$

Then, according (4.1), the normalized orientation of a gradient vector will be:

$$\begin{aligned} \widetilde{\omega}(z) &= [sin(\omega(z)) \qquad cos(\omega(z))] \\ &= [sin(\theta(z) - \overline{\theta}(P)) \qquad cos(\theta(z) - \overline{\theta}(P))] \\ &= [sin(\theta(z))cos(\overline{\theta}(P)) + cos(\theta(z))sin(\overline{\theta}(P)) \\ &\qquad cos(\theta(z))cos(\overline{\theta}(P)) - sin(\theta(z))sin(\overline{\theta}(P))] \end{aligned} \tag{4.4}$$

Finally, we define the gradient match kernel with the normalized orientation as follows:

$$K_{gradient}(P, Q) = \sum_{z \in P} \sum_{z' \in Q} k_{\widetilde{m}}(z, z') k_o(\widetilde{\omega}(z), \widetilde{\omega}(z')) k_p(z, z') \tag{4.5}$$

$$\begin{aligned} K_h(P, Q) = & \qquad K_h(p)^\top K_h(Q) \\ = & \sum_{z \in P} \sum_{z' \in Q} \widetilde{m}(z) \widetilde{m}(z') \delta(z)^\top \delta(z') \quad (1) \end{aligned}$$

The gradient magnitude kernel $k_{\widetilde{m}}$ is defined as:

$$k_{\widetilde{m}}(z, z') = \widetilde{m}(z) \widetilde{m}(z') \tag{4.6}$$

Where the normalized gradient magnitude $\widetilde{m}(z)$ is defined as:

$$\widetilde{m}(z) = \frac{m(z)}{\sqrt{\sum_{z \in P} m(z)^2 + \epsilon_g}} \tag{4.7}$$

where ϵ_g is a small constant. $m(z)$ is magnitude of the image gradient at a pixel z. The gradient magnitude kernel $k_{\widetilde{m}}$ is conspicuously a positive definite kernel. Both the orien-

tation kernel k_o and the position kernel k_p are Gaussian kernels which is of the form:

$$k(x, x') = \exp(-\gamma \|x - x'\|^2) \tag{4.8}$$

The factor γ will be defined individually for k_o and k_p that are denoted by γ_o and γ_p respectively.

Now, given the definition of match kernel, how to extract feature vector for a patch. Let $\varphi_o(.)$ and $\varphi_p(.)$ the feature maps for the gradient orientation kernel k_o and position kernel k_p respectively. The Eq. 4.5 is then rewritten as:

$$\begin{aligned} K_{gradient}(P, Q) &= \sum_{z \in P} \sum_{z' \in Q} \widetilde{m}(z) \left(\widetilde{m}(z') \varphi_o(\widetilde{\omega}(z))^\top \varphi_o(\widetilde{\omega}(z')) \right) \left(\varphi_p(z)^\top \varphi_p(z') \right) \\ &= \left(\sum_{z \in P} \widetilde{m}(z) \varphi_o(\widetilde{\omega}(z)) \otimes \varphi_p(z) \right)^\top \left(\sum_{z' \in Q} \widetilde{m}(z') \varphi_o(\widetilde{\omega}(z')) \otimes \varphi_p(z') \right) \\ &= F_{gradient}(P)^\top F_{gradient}(Q) \end{aligned} \tag{4.9}$$

where $F_{gradient}(P)$ feature over image patch P that is defined as:

$$F_{gradient}(P) = \sum_{z \in P} \widetilde{m}(z) \varphi_o(\widetilde{\omega}(z)) \otimes \varphi_p(z) \tag{4.10}$$

Then, the approximate feature over image patch P is constructed as:

$$\overline{F}_{gradient}(P) = \sum_{z \in P} \widetilde{m}(z) \phi_o(\widetilde{\omega}(z)) \otimes \phi_p(z) \tag{4.11}$$

where $\otimes$ is the Kronecker product, $\phi_o(\widetilde{\omega}(z))$ and $\phi_p(z)$ are approximate feature maps for the kernel k_o and k_p, respectively.

The approximate feature maps are computed based on a basic method of kernel descriptor. The basic idea of representation based on kernel methods is to compute the approximate explicit feature map for kernel match function, Fig.4.4.

In other word, the kernel match functions are approximated based on explicit feature maps. This enables efficient learning methods for linear kernels to be applied to the non-linear kernel. This approach was introduced in [5, 6, 55, 56, 95].

One of the methods for approximating explicit features has been presented in [6]. In the following, we review this method briefly. Given a match kernel function $k(x, y)$, the feature map $\varphi(x)$ for the kernel $k(x, y)$ is a function mapping x into a vector space so as:

$$k(x, y) = \varphi(x)^\top \varphi(y) \tag{4.12}$$

Suppose that we have a set of basis vectors $B = \{\varphi(v_i)\}_{i=1}^{D}$, the approximation of

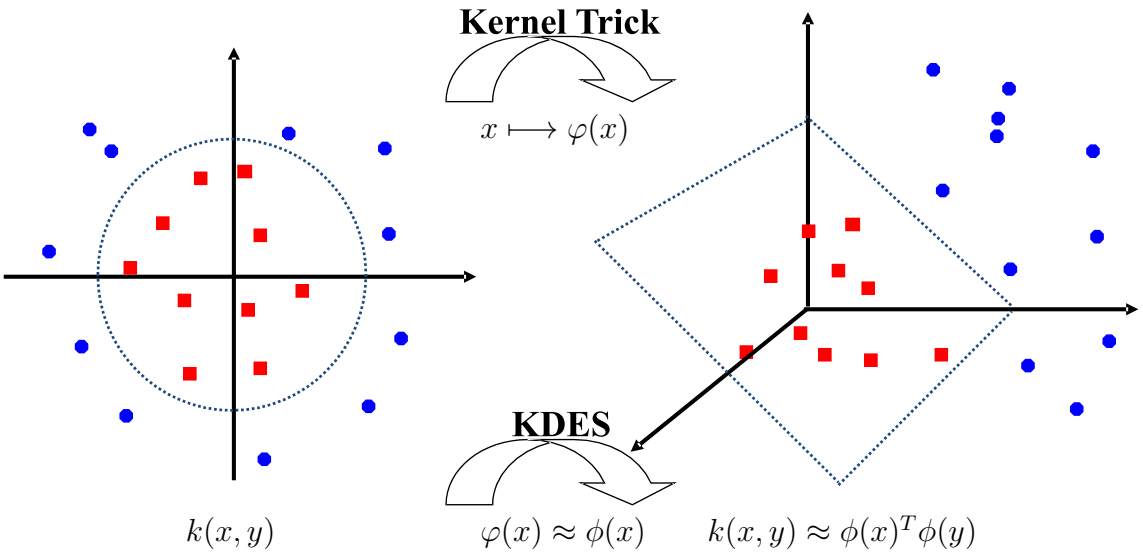


Figure 4.4 – The basic idea of representation based on kernel methods.

feature vector $\varphi(x)$ can be $H\overline{z}_x$ where $\overline{z}_x$ is defined as:

$$\overline{z}_x = \arg\min_{z_x} \|\varphi(x) - Hz_x\|^2 \tag{4.13}$$

where $H = [\varphi(v_1), ..., \varphi(v_D)]$ is the transformation matrix, $\overline{z}_x$ are the projection coefficients of $\varphi(x)$ on B. Eq. (4.13) is a convex quadratic program. The analytic solution of Eq.(4.13) is:

$$\overline{z}_x = (H^\top H)^{-1}(H^\top \varphi(x)) \tag{4.14}$$

The approximated feature map is [6]:

$$\phi(x) = Gk_B(x) \tag{4.15}$$

where G is defined by:

$$G^\top G = K_{BB}^{-1} \tag{4.16}$$

where K_{BB} is $D \times D$ matrix with $\{K_{BB}\}_{ij} = k(v_i, v_j)$. k_B is a $D \times 1$ vector with $\{k_B\}_i = k(x, v_i)$.

With a set of features $X = \{x_1, ..., x_p\}$ the approximated feature map on X is defined as:

$$\overline{\phi}(X) = \frac{1}{p} G \left[\sum_{x \in X} k_B(x) \right] \tag{4.17}$$

The approximated explicit feature map computation requires set of basis vectors $B = \{\varphi(v_i)\}_{i=1}^{D}$. The set of basis vector B is learned from a training pool of F features $\{x_1, ..., x_F\}$ by using the constrained kernel singular value decomposition (CKSVD), [6].

To extract approximate features $\phi_o(\widetilde{\omega}(z))$, $\phi_p(z)$ in Eq.4.11 from match kernels, compact basis vectors need to be generated by learning. The compact basis vectors are learned from sufficient basis vectors using kernel principal component analysis. Where, the sufficient basis vectors are sampled uniformly and densely from support region using a fine grid so as these basis vectors make an accurate approximation to match kernels. We use the shared set of basis vectors and match kernel parameters from [5] that were learned using a subset of ImageNet.

Let the learned set of d_o basis vectors is $B_o = \{\varphi_o(x_1), \varphi_o(x_2), ..., \varphi_o(x_{d_o})\}$ and the set of d_p basis vectors is $B_p = \{\varphi_p(y_1), \varphi_p(y_2), ..., \varphi_p(y_{d_p})\}$ considering k_o and k_p kernels respectively. Where x_i are sampled normalized gradient vectors and y_i are normalized $2D$ position of pixels in an image patch.

The Kronecker product causes high dimension of the feature vector $\overline{F}_{gradient}(P)$. To reduce the dimension of $\overline{F}_{gradient}$, the kernel principal component analysis is applied into the joint basis vectors $\{\varphi_o(x_i) \otimes \varphi_p(y_j)\}_{i=1..d_o, j=1..d_p}$. Let t-*th* component ${\alpha^t}_{ij}$ is learned through kernel principal component analysis, following [5], the resulting gradient kernel descriptor for match kernel in (4.5) has the form:

$$\widetilde{F}^t_{gradient}(P) = \sum_{i=1}^{d_o} \sum_{j=1}^{d_p} \alpha^t_{ij} \sum_{z \in P} \widetilde{m}(z) k_o(\widetilde{\omega}(z), x_i) k_p(z, y_j) \tag{4.18}$$

4.3.3 Extraction of image-level features

Once patch-level features are computed for each patch, the remaining work is computing a feature vector representing the whole image. In [6], the authors used a spatial pyramid structure by dividing the image into cells using horizontal and vertical lines at several layers (Fig.4.5(a)). This structure is generic, therefore does not take into account the specific shape of objects. In our work, as the hand is an object with a specific structure, we propose a new pyramid structure specifically for the hand. In the following, we present in detail each step to build the final descriptor of the image.

Design a hand specific pyramid structure for patch-level features pooling

Fig. 4.5(b) shows the proposed hand pyramid structure. The main idea is to exploit characteristics of hand postures. Let the hand posture image have a size of $w \times h$. We observe that the regions at the image corners often do not contain information. For this reason, we only consider the area inside the inscribed ellipse of the hand image rectangle

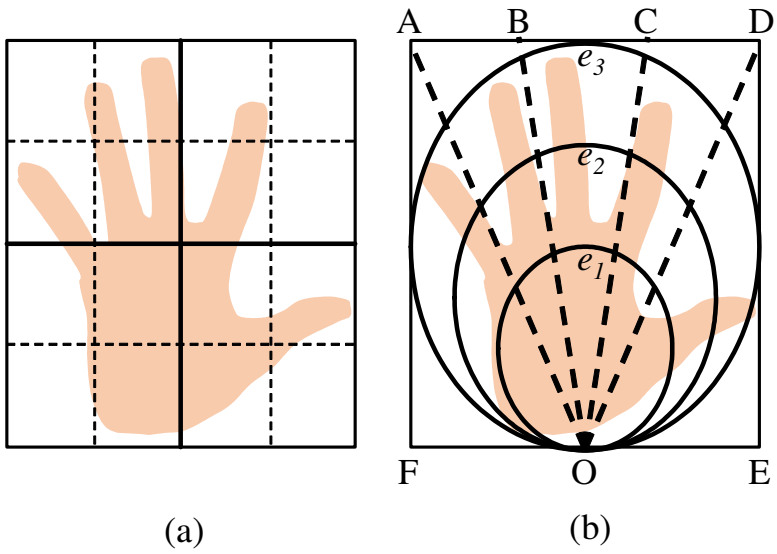


Figure 4.5 – (a) General spatial pyramid structure used in [6]. (b) The proposed hand pyramid structure.

bounding box (e_3). The lines along the fingers converge at the lowest center point of the palm, near the wrist (O). Based on the structure of the hand, the ellipses (e_1, e_2, e_3) and the lines (OA, OB, OC, OD) are used to divide the hand region into parts that contain different components of the hand such as palm and fingers where $AB = BC = CD$. The detail of designed structure is described as: O is the midpoint of $FE(OF = OE)$. The ellipse e_1 is the inscribed ellipse of the rectangle that has a size of $(\frac{1}{2}w \times \frac{1}{2}h)$. The line FE is a tangent line of the ellipse e_1. The contact between the line FE and the ellipse e_1 is O. The ellipses are axis-aligned. In the similarity, the ellipsis e_2 is the inscribed ellipsis of the rectangle that has size of $(\frac{3}{4}w \times \frac{3}{4}h)$.

In a layer, we define a cell as being a full region limited by these ellipses and lines. In our work, the hand pyramid structure has 3 layers, (see Fig.4.6).

- Layer 1: This layer contains only one cell defined by the biggest inscribed ellipse e_3.
- Layer 2: In [5], this layer has four rectangular cells. Unlike this, we create eight cells: three cells created from 3 ellipses and five cells created from the intersection of four lines with the biggest ellipse.
- Layer 3: This layer has 15 cells generated from the intersection between lines and three ellipses.

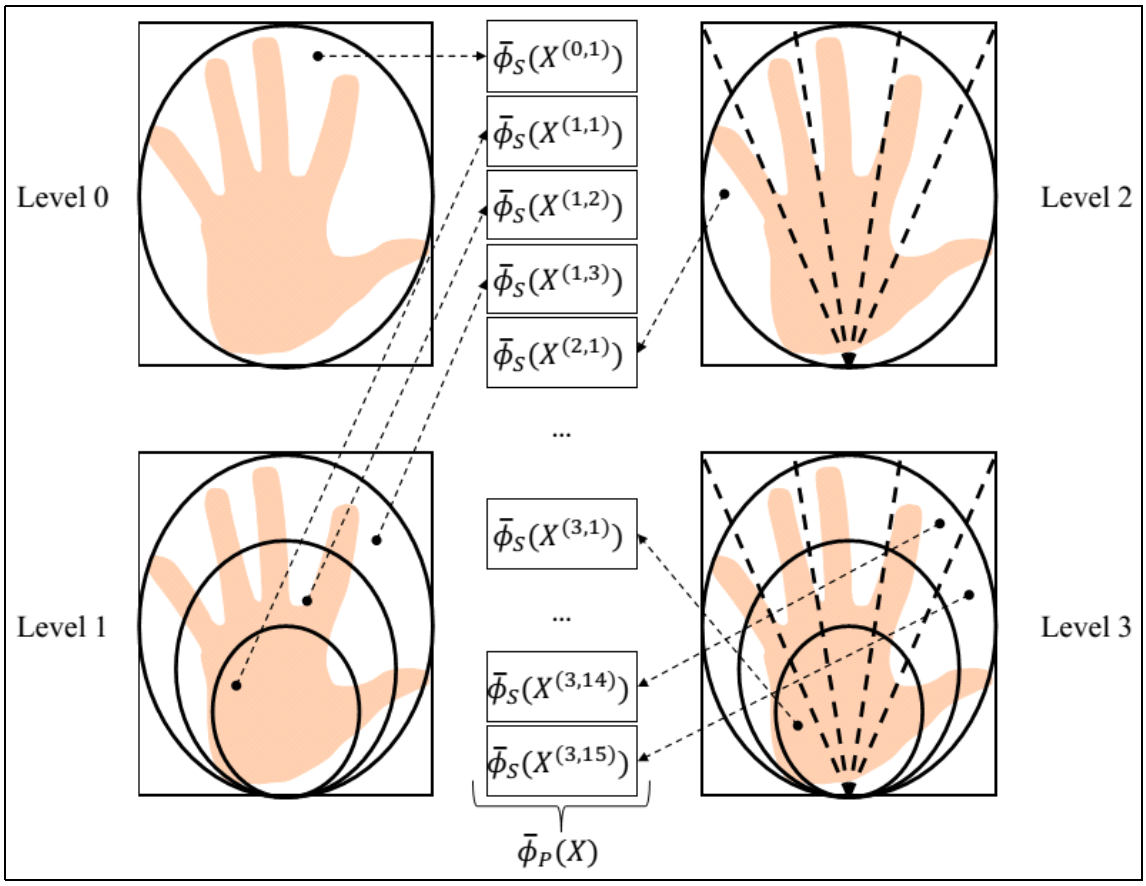


Figure 4.6 – Construction of image-level feature concatenating feature vectors of cells in layers of hand pyramid structure.

Create the final descriptor of the whole image

To create the final descriptor of the whole image, we firstly compute the feature vector for each cell of the hand pyramid structure, and then concatenate them into a final descriptor.

To compute the feature vector for a cell, we use a method of generation feature vector for a region that has a set of patch level features. The method is an adaptation of BoW method using match kernels to measure the similarity between two local features. The most contents of the specific method we use in our work relates to the Efficient Match Kernels (EMK) between sets of features that is introduced in [6].

Let C be a cell with its set of patch-level features is:

$$X = \{x_1, ..., x_p\} \tag{4.19}$$

Where p is the number of patches of C.

In BoW, each patch-level feature vector of image is treated as a word. Suppose that $V = \{v_1, ..., v_D\}$ is the dictionary (a set of visual words). A patch-level feature vector is

quantized into a D dimensional binary indicator vector:

$$\mu(x) = [\mu_1(x), ..., \mu_D(x)]^\top \tag{4.20}$$

where, $\mu_i(x)$ is defined as:

$$\mu_i(x) = \begin{cases} 1, & \text{if } x \in R(v_i) \\ 0, & \text{otherwise} \end{cases} \tag{4.21}$$

where $R(v_i) = \{x : \|x - v_i\| \leq \|x - v\|, \forall v \in V\}$. The feature vectors of an image is defined as a normalized histogram:

$$\overline{\mu}(X) = \frac{1}{|X|} \sum_{x \in X} \mu(x) \tag{4.22}$$

where $|X|$ is the cardinality of X. When BoW is used in conjunction with a linear classifier, the match kernel function is:

$$\begin{aligned} K_B(X, Y) &= \overline{\mu}(X)^\top \overline{\mu}(Y) \\ &= (\tfrac{1}{|X|} \textstyle\sum_{x \in X} \mu(x))^\top (\tfrac{1}{|Y|} \sum_{y \in Y} \mu(y)) \\ &= (\tfrac{1}{|X|} \textstyle\sum_{x \in X} \mu(x)^\top)(\tfrac{1}{|Y|} \sum_{y \in Y} \mu(y)) \\ &= \tfrac{1}{|X|} \tfrac{1}{|Y|} (\textstyle\sum_{x \in X} \mu(x)^\top)(\sum_{y \in Y} \mu(y)) \\ &= \tfrac{1}{|X|} \tfrac{1}{|Y|} \textstyle\sum_{x \in X} \sum_{y \in Y} \mu(x)^\top \mu(y) \end{aligned} \tag{4.23}$$

Let to denote the similarity between two path-level features x and y as (see the definition of $\mu(x)$ in Eq. 4.20 and Eq. 4.21):

$$\begin{aligned} \delta(x, y) &= \mu(x)^\top \mu(y) \\ &= \begin{cases} 1, & \text{if } \exists i \in \{1, ..., D\} : x \in R(v_i) \text{ and } y \in R(v_i) \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{4.24}$$

The Eq. 4.23 is then rewritten as:

$$K_B(X, Y) = \frac{1}{|X|} \frac{1}{|Y|} \sum_{x \in X} \sum_{y \in Y} \delta(x, y) \tag{4.25}$$

The meaning of Eq. 4.24 is that the similarity between x and y is 1 if x and y belong the same region $R(v_i)$, and 0 otherwise. The two path-level features assigned to different (even very close) clusters are considered completely different (see Fig. 4.7). In Fig.4.7, points $v_1, ..., v_4$ represent cluster centers (visual words), and points x, y, z are patch-level

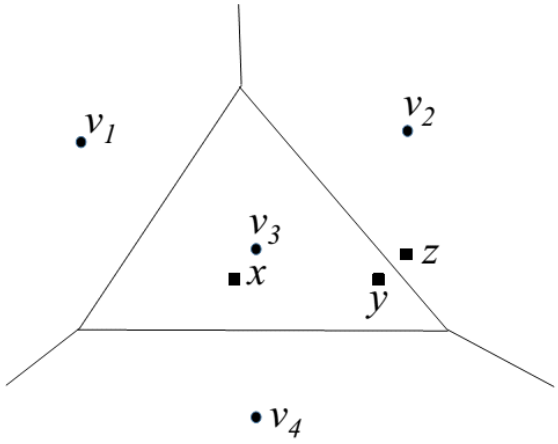


Figure 4.7 – Disadvantages of hard assignment, adopt from Fig. 1 in [74].

features. Here two disadvantages of hard assignment are showed: (i) In hard assignment, the similarity between patch-level features y and z will be 0 that means they are completely different because they are assigned to different visual words despite being close in patch-level feature space. (ii) Features x and y are both assigned to visual word v_3 equally and there is no way of distinguishing that x is closer than y as well as there is no way to know that in fact patch-level feature y is closer to z than x. This quantization provides a very coarse approximation to the actual similarity between the two path-level features, 1 if assigned to the same visual word, and 0 otherwise. This hard assignment can causes errors due to variability in the feature descriptor such as image noise, varying scene illumination, instability in the feature detection process and non-affine changes in the measurement regions. In [74], Philbin *et al* therefore described a "soft assignment" based technique by a weighted combination of visual words. This is an improvement in matching (compared to a hard assignment) of the patches. The term "soft assignment" describes techniques in that the weight assigned to neighbouring words depends on the distance between the descriptor and the centers.

In similar idea, in EMK, $\delta(x, y)$ in Eq. 4.25 is replaced with a continuous kernel function $k(x, y)$ that more accurately measures the similarity between path-level features x and y. The kernel function then become:

$$K_S(X, Y) = \frac{1}{|X|}\frac{1}{|Y|}\sum_{x\in X}\sum_{y\in Y} k(x, y) \tag{4.26}$$

Suppose that $k(x, y)$ is a finite dimensional kernel (see Definition 5), the match kernel is then:

$$\begin{aligned} K_S(X,Y) &= \frac{1}{|X|}\frac{1}{|Y|}\sum_{x\in X}\sum_{y\in Y}k(x,y) \\ &= \frac{1}{|X|}\frac{1}{|Y|}\sum_{x\in X}\sum_{y\in Y}\phi(x)^\top\phi(y) \\ &= \frac{1}{|X|}\frac{1}{|Y|}(\sum_{x\in X}\phi(x)^\top)(\sum_{y\in Y}\phi(y)) \\ &= (\frac{1}{|X|}\sum_{x\in X}\phi(x)^\top)(\frac{1}{|Y|}\sum_{y\in Y}\phi(y)) \\ &= (\frac{1}{|X|}\sum_{x\in X}\phi(x))^\top(\frac{1}{|Y|}\sum_{y\in Y}\phi(y)) \end{aligned} \tag{4.27}$$

Definition 5 (Finite dimensional kernel, [6]). *The kernel function $k(x,y)=\phi(x)^\top\phi(y)$ is called finite dimensional if the feature map $\phi(.)$ is finite dimensional.*

The feature map on the set of vectors is defined as:

$$\overline{\phi}_S(X) = \frac{1}{|X|}\sum_{x\in X}\phi(x) \tag{4.28}$$

The match kernel is rewritten as:

$$K_S(X,Y) = \overline{\phi}_S(X)^\top\overline{\phi}_S(Y) \tag{4.29}$$

The feature vector on the set of patches, $\overline{\phi}_S(X)$, is extracted explicitly. The linear classifier can be then applied on the extracted feature vectors. In Eq.4.28, $\phi(x)$ is approximate feature maps (4.15) for the kernel $k(x,y)$ with the set of basis vector that is generated by constrained singular value decomposition method (CKSVD) [6]. The feature vector on the set of patches, $\overline{\phi}_S(X)$, is extracted explicitly.

Once feature vectors for cells are computed, we concatenate them to construct image level feature vector. Given an image, let L be the number of spatial layers to be considered. In our case $L=3$. The number of cells in layer *l*-th is (n_l). $X(l,t)$ is set of patch-level features falling within the spatial cell (l,t) (cell *t*-th in the *l*-th level). A patch is fallen in a cell when its centroid belongs to the cell. The spatial hand structure match kernel then is:

$$\begin{aligned} K_P(X,Y) &= \sum_{l=0}^{L}\sum_{t=1}^{n_l}(w^{(l)})^2K_S(X^{(l,t)},Y^{(l,t)}) \\ &= \sum_{l=0}^{L}\sum_{t=1}^{n_l}(w^{(l)})^2\overline{\phi}_S(X^{(l,t)})^\top\overline{\phi}_S(Y^{(l,t)}) \\ &= [w^{(0)}\overline{\phi}_S(X^{(0,1)})^\top,...,w^{(l)}\overline{\phi}_S(X^{(l,t)})^\top,...,w^{(L)}\overline{\phi}_S(X^{(L,n_L)})^\top] \\ &\quad [w^{(0)}\overline{\phi}_S(Y^{(0,1)});...;w^{(l)}\overline{\phi}_S(Y^{(l,t)});...;w^{(L)}\overline{\phi}_S(Y^{(L,n_L)})] \\ &= [w^{(0)}\overline{\phi}_S(X^{(0,1)});...;w^{(l)}\overline{\phi}_S(X^{(l,t)});...;w^{(L)}\overline{\phi}_S(X^{(L,n_L)})]^\top \\ &\quad [w^{(0)}\overline{\phi}_S(Y^{(0,1)});...;w^{(l)}\overline{\phi}_S(Y^{(l,t)});...;w^{(L)}\overline{\phi}_S(Y^{(L,n_L)})] \\ &= \overline{\phi}_P(X)^\top\overline{\phi}_P(Y) \end{aligned} \tag{4.30}$$

Where $\overline{\phi}_P(X)$ is the feature map on the spatial hand structure:

$$\overline{\phi}_P(X) = [w^{(0)}\overline{\phi}_S(X^{(0,1)}); ...; w^{(l)}\overline{\phi}_S(X^{(l,t)}); ...; w^{(L)}\overline{\phi}_S(X^{(L,n_L)})] \tag{4.31}$$

In (4.31), the weight associated with level l is defined as:

$$w^{(l)} = \frac{\frac{1}{n_l}}{\sum_{l=1}^{L}\frac{1}{n_l}} \tag{4.32}$$

Fig. 4.6 shows image-level feature extraction on the proposed hand pyramid structure. Until now, we obtain the final representation of the whole image, which we call image-level feature vector. This vector will be the input of a Multiclass SVM for training and testing.

We can see that the hand pyramid gives a suitable representation for upright frontal hand postures in that the difference between postures is the configuration of the fingers (open or closed). In addition, the proposed hand representation has another advantage that is the image level feature is more invariant to slightly rotation/finger distortion. The reason of this ability is that the feature vector of a cell in pyramid is computed based on the set of feature vectors of patches belonging to the cell without concerning the locations of patches in the cell. Moreover, the patch level feature is invariant to rotation. In case of our constraints (upright frontal hand postures with slightly rotation/finger distortion), the hand pyramid with normalized orientation gradient in the patch is suitable for hand representation, see Fig.4.8(a,b).

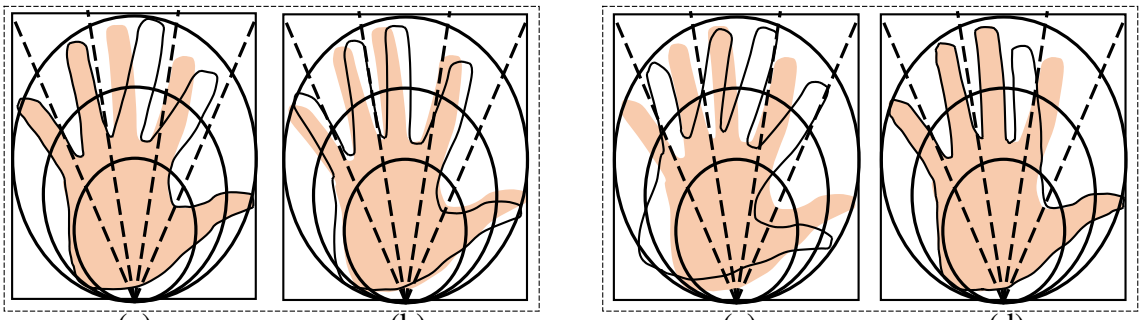


Figure 4.8 – (a) Slight finger distortion and (b) slight hand rotation that do not make the same fingers belong to different cells; (c) heavy finger distortion and (d) strong hand rotation that make the same fingers belong to different cells.

When hands strongly rotate or fingers heavily distort that make the same finger (or part of hand) in two hand images belongs to different cells in pyramid, above advantages of proposed hand presentation are not shown, see Fig.4.8(c,d). In the case of strong rotation

of hand, we could normalize hand image before applying hand pyramid.

4.4 Experiments

4.4.1 Dataset

This section presents the datasets that we use to evaluate the hand posture recognition algorithms. They are two benchmark dataset [75,91], one dataset we collect from [9], and one dataset with more number of hand postures that we build ourself (L3i-MICA dataset). The Table 4.1 recapitulates these datasets. The details are presented in the subsections.

NUS II dataset

The NUS II dataset is introduced in [75]. Fig. 4.9 shows several examples of a subset of this dataset. The hand postures in this dataset were captured in complex natural back-

Figure 4.9 – Sample images from NUS hand posture dataset-II (data subset A), showing posture class from 1 to 10, [75]

grounds. The shapes, sizes of postures and ethnicities of subjects are various. The dataset has 2000 hand posture color images of 10 posture classes performed by 40 subjects, 5 five images per class per subject. The image size is 160×120. To evaluate the hand postures recognition method, we crop manually to build a set of whole hand regions from NUS II dataset. This means we assume that the segmentation step is perfect. Fig. 4.10 shows samples of cropped whole hand region images from NUS II dataset. The hand posture recognition step will be tested on NUS II dataset using 10 fold cross-validation.

Triesch dataset

The Jochen Triesch static hand posture database [91] is one of the most popular benchmark dataset for hand postures recognition researches. The database consists of 10 hand

Table 4.1 – Summary of the datasets for evaluation

Dataset	**Dataset description**	**Comments**
NUS II	• 10 postures • 40 subjects • 5 images per class per subject • 2000 hand posture color image • Image resolution: 160x120 • Complex background	• Low resolution • Hand appears as the biggest object in the image
Triesch	• 10 hand postures • 24 subjects • 17280 8bits grayscales images • Image resolution: 128x128 • 3 backgrounds: Light, dark, complex	• Low resolution • Cropped hand region
Caron	• 5 hand postures • 4 subjects • Each of five postures has 572 images in training dataset and 572 images in testing dataset • Uniform simple background	• High resolution • Cropped hand region
L3i-MICA	• 21 hand postures • 10 subjects • 840 color videos, 30fps • Image resolution: 320x240 • Complex background	• Medium resolution • Hands appears as a small object in the image

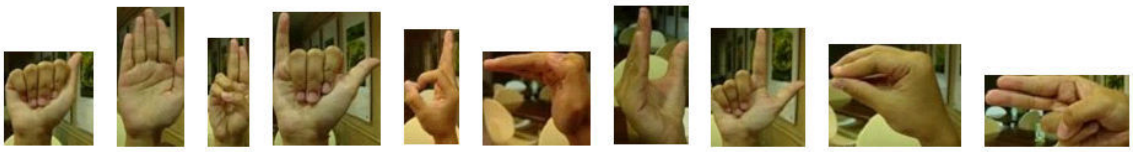


Figure 4.10 – Samples of cropped whole hand region images from NUS II dataset.

signs (a, b, c, d, g, h, i, l, v, y). We also evaluate the hand postures recognition on whole hand regions that were provided in the protocol including in Jochen Triesch dataset. The dataset performed by 24 persons against three backgrounds (light, dark, complex). For each person, the ten postures were recorded in front of uniform light, uniform dark and complex background giving 720 images (see Fig.4.11). The training dataset contains 60

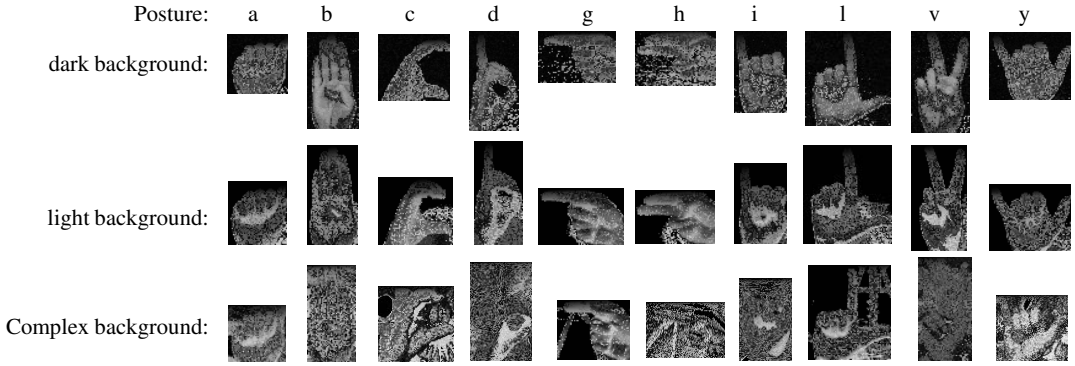


Figure 4.11 – Samples of Jochen Triesch static hand postures dataset.

images of three persons against light and dark background. The remaining of the dataset is treated as a testing dataset. The format of images is 8-bit grey-scale with resolution of 128×128.

L3i-MICA hand posture dataset

L3i-MICA hand posture dataset was presented in Section 3.5.1. We prepare two kinds of dataset for evaluation hand posture recognition methods.

The first one is manual detection dataset. We prepare the training and testing set of frames by uniform sampling of the training videos set and the testing videos set respectively with step of 10 frames. We then manually crop the image in order to have the whole hand regions. The training dataset has 4636 cropped whole hand images. The testing dataset has 4690 cropped whole hand images. The number of training and testing images for each of 21 hand postures are nearly equal. Fig.4.12 shows several samples of L3i-MICA dataset. The size of hand posture images are variant.

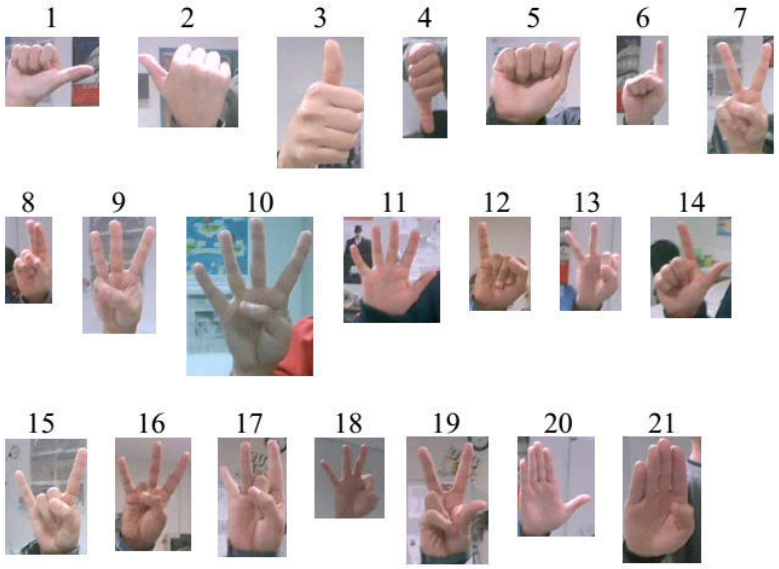


Figure 4.12 – Samples of the set of 21 hand postures in L3i-MICA dataset.

The second one is automatic detection dataset. For automatic hand detection, we apply the hand detection method proposed in Chapter 3 to detect internal center region of the hand. We then expand the detected region to obtain the whole hand region. For this, we keep only true detections (based on the Jaccard index) and discard the false detections since we focus on evaluating the hand posture recognition method. We randomly select 100 examples per posture from the automatic detection results on L3i-MICA dataset for testing. We also create the testing dataset, in the same way. Fig.4.13 shows several samples of the automatic detection dataset. In this case, the hand region contains more background.

Caron's dataset

Caron's dataset contains five hand postures of the 4 subjects with a uniform simple background. This dataset is suitable for the method in [19]. Using this dataset allows to avoid the effect of the complex background.

We collect this dataset from [9]. In [9], Lefebvre-Gascon and Caron implemented the hand postures recognition method that presented in [19]. They also provided a hand postures dataset that is similar to hand postures in [19] (see Fig.4.14). For our evaluation purpose, we use the cropped part of this dataset. Each posture has 572 images for training and 572 images for testing. These images are manually cropped images with a uniform dark background.

Figure 4.13 – Samples of the set of 21 hand postures in L3i-MICA dataset with automatic segmentation.

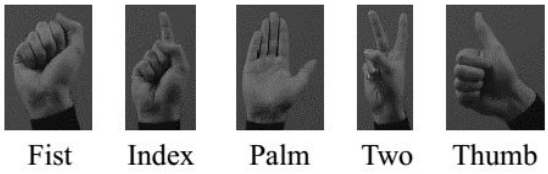


Figure 4.14 – Example images of the five hand postures in [9].

4.4.2 Performance measurement

We use accuracy to measure the performance of the hand posture recognition method. The accuracy is defined by the ratio between the number of correct recognition images and the number of testing images. In our experiments, each input sample will be classified to one of the predefined classes.

$$Accuracy = \frac{\#\{True\ classifications\}}{\#\{Input\ testing\ samples\}} \tag{4.33}$$

4.4.3 Experiment 1: Comparison of gradient with other types of KDES

In [5], the authors introduced three types of KDES that are Gradient-KDES, Texture-KDES, and Pixel-value-KDES. In this experiment, we evaluate the performance of different KDES types: gradient-KDES, Texture-KDES, Pixel-value-KDES and Combination. For these types of KDES, we try 3 different color spaces that are RGB, HSV, and Lab. We firstly present these types of KDES briefly.

Texture-KDES

In Texture-KDES, Local binary patterns (LBP) [68] is treated as a type of **pixel-level feature** that is computed in the manner shown in Fig. 4.15. Each pixel is compared to

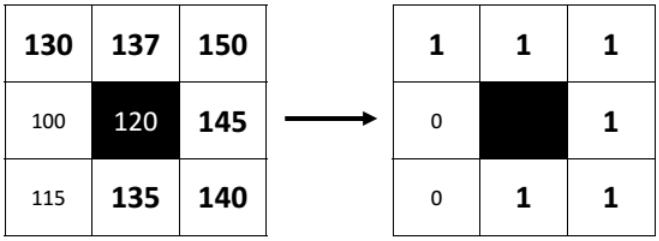


Figure 4.15 – Local binary patterns (LBP)

each of its 8 neighbors. Where the neighborhood pixel's value is greater than the center pixel's value, the resulting LBP value is 1. Otherwise, it is 0. The result is an 8-digit binary number. Let denote the resulting binary 8-dimensional vector at pixel z by $b(z)$, and denote the standard deviation of pixel values in the 3×3 neighborhood around z by $s(z)$. $b(z)$ and $s(z)$ are treated as texture pixel level features.

The **texture match kernel** is constructed from three kernels that are the standard deviation kernel k_s, the binary pattern kernel k_b, and the position kernel k_p.

$$K_{texture}(P, Q) = \sum_{z \in P} \sum_{z' \in Q} k_s(z, z') k_b(\widetilde{b}(z), \widetilde{b}(z')) k_p(z, z') \tag{4.34}$$

where the standard deviation kernel $k_s(z, z')$ is the inner product of two vector $\widetilde{s}(z)$ and $\widetilde{s}(z')$:

$$k_s(z, z') = \widetilde{s}(z)\widetilde{s}(z') \tag{4.35}$$

where $\widetilde{s}(z)$ is the normalized standard deviation of pixel values that is defined as:

$$\widetilde{s}(z) = \frac{s(z)}{\sqrt{\sum_{z \in P} s(z)^2 + \epsilon_s}} \tag{4.36}$$

where $s(z)$ is the standard deviation of pixel values in the 3×3 neighborhood around z, ϵ_s is a small constant.

In Eq.4.34, the binary pattern kernel k_b is a Gaussian kernel that is defined in Eq. 4.8. In this kernel, γ factor is replaced by γ_b. $\widetilde{b}(z)$ is binary column vector binaries the pixel value differences in a local window around z. The meaning of standard deviation kernel k_s is the weight that indicates the contribution of each local binary pattern. The binary pattern kernel k_b provides the similarity between two patches through local binary

patterns.

In the same way of Gradient-KDES extraction, the Texture-KDES for (4.34) has the form:

$$\widetilde{F}^t_{Texture}(P) = \sum_{i=1}^{d_t}\sum_{j=1}^{d_p} \alpha^t_{Texture(ij)} \sum_{z\in P} \widetilde{s}(z) k_b(\widetilde{b}(z), w_i) k_p(z, y_j) \tag{4.37}$$

where where d_v is number of elements of the set of basis vectors $B_v = \{\varphi_t(w_1), ..., \varphi_t(w_{d_t})\}$, w_i are normalized standard deviation of pixel values in an image patch.

Pixel-value-KDES

In Pixel-value-KDES, the intensities of pixels are used as **pixel level features**. The **pixel value match kernel** is constructed from two kernels that are the value kernel k_v and the position kernel k_p.

$$K_{PixelValue}(P, Q) = \sum_{z\in P}\sum_{z'\in Q} k_v(v(z), v(z'))k_p(z, z') \tag{4.38}$$

where $v(z)$ is the pixel value at position z that is intensity of pixel z. k_v is a Gaussian kernel that is defined in Eq. 4.8 with γ factor is replaced by γ_v. The color kernel k_v measures how similar two pixel values are. Therefore, the pixel value match kernel will capture image appearance.

In the same way of Gradient-KDES extraction also, the PixelValue-KDES for match kernel (4.38) has the form:

$$\widetilde{F}^t_{PixelValue}(P) = \sum_{i=1}^{d_v}\sum_{j=1}^{d_p} \alpha^t_{v(ij)} \sum_{z\in P} k_v(v(z), u_i))k_p(z, y_j) \tag{4.39}$$

where d_v is number of elements of the set of basis vectors $B_v = \{\varphi_v(u_1), ..., \varphi_v(u_{d_v})\}$, u_i are value of pixels in an image patch.

Combination-KDES

The combination of the three kernel descriptors (Combination-KDES) is constructed by concatenating the image-level features vectors of three type of KDES.

Results

We performance this comparison on two color dataset that are L3i-MICA dataset and NUS II dataset. In this experiment, we use manual hand segmentation. The results of

this experiment is shown in Table 4.2 (on L3i-MICA dataset) and Table 4.3 (on NUS II dataset).

Table 4.2 – The investigation results on L3i-MICA dataset

Method / Feature	Original KDES			Improved method		
	RGB	HSV	Lab	RGB	HSV	Lab
Gradient	**0.844**	0.624	0.624	**0.912**	0.721	0.857
PixelValue	0.694	0.725	0.719	0.871	0.614	0.741
Texture	0.783	0.576	0.688	0.779	0.751	0.824
Combination	**0.850**	0.741	0.823	**0.913**	0.796	0.880

Table 4.3 – The investigation results on NUS II dataset

Method / Feature	Original KDES			Improved method		
	RGB	HSV	Lab	RGB	HSV	Lab
Gradient	**0.953**	0.921	0.951	**0.974**	0.938	0.959
Texture	0.829	0.803	0.868	0.943	0.912	0.881
PixelValue	0.927	0.886	0.903	0.852	0.833	0.886
Combination	0.961	0.946	**0.973**	**0.968**	0.946	0.964

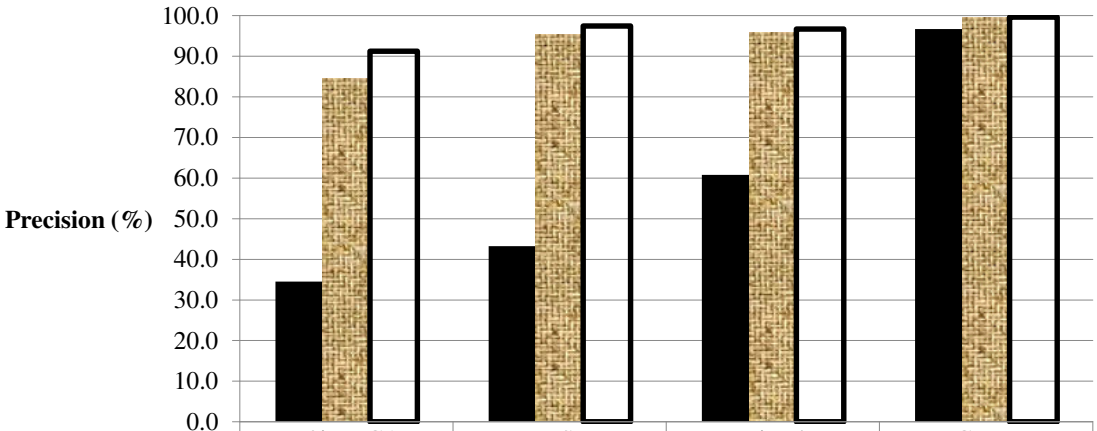
The results indicate that the gradient kernel descriptor obtains the best result on RGB color space on both two datasets. The combination kernel descriptor is slightly better than the best individual descriptor. However, the recognition time takes three times more expensive than the individual descriptor because the feature extraction time plays a critical role in recognition time. In real application, reducing computation time is important. For this reason, the best choice for hand posture recognition is Gradient-KDES on RGB color space.

4.4.4 Experiment 2: Comparison with the state of the art methods

In this experiment, among different approaches proposed for hand posture recognition, we compare our method with a method presented in and in [19] because it is closely related to our work and proved robust for hand posture recognition. The proposed method is also compared with original KDES. We perform this experiment on all four datasets with manual segmentation and automatic segmentation result on L3i-MICA dataset.

Figure 4.16 shows obtained accuracy of three methods on four datasets with perfect hand detection while Fig. 4.17 illustrates the accuracy of the three methods for L3i-MICA dataset with automatic hand detection.

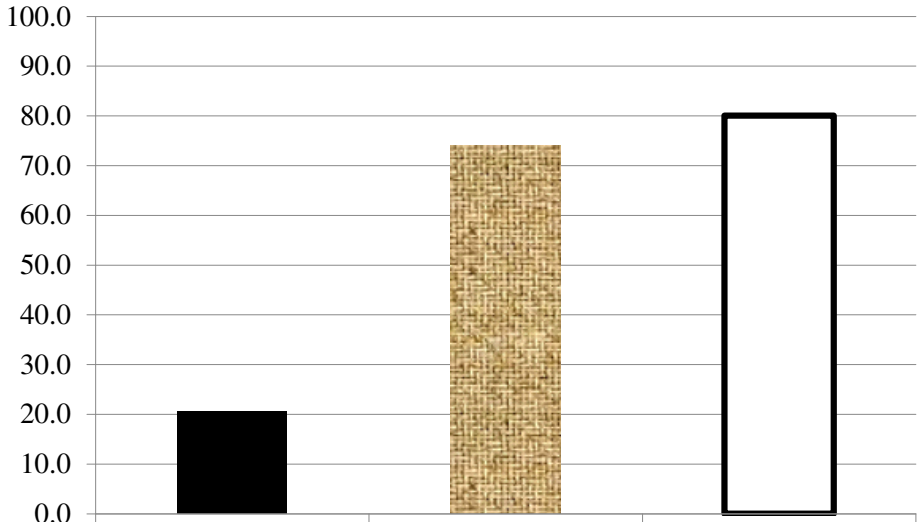
We observe that our method outperforms the state of the art method [19] and original KDES [5] on all datasets for both manual and automatic hand detection. The recognition



	L3i-MICA	NUS II	Triesch	Caron
Dardas' method	34.5	43.2	60.8	96.7
OriginalKDES	84.4	95.3	95.7	99.6
Proposed method	91.2	97.4	96.7	99.6

Dataset

Figure 4.16 – Comparison our proposed method with original KDES and Dardas method [19] on manual segmentations from four datasets.



	Dardas' method	Original KDES	Proposed method
Accuracy (%)	20.6	74.0	80.0

Figure 4.17 – Comparison our proposed method with original KDES and Dardas method [19] on automatic segmentation results from L3i-MICA dataset.

accuracy with automatic hand detection is, of course, lower than with manual detection, but remains relatively good (80%). This suggests that we can combine our recognition method with the hand detection method in order to build a complete human-robot interaction using hand postures.

However, the performance of the method depends on the characteristics of the data which it is applied to. With L3i-MICA dataset, since this dataset contains images of the same hand postures in different scales, our method has been proved its robustness. Our method gets 7% better than the original method based on kernel descriptor. For the two others datasets, the improvement in recognition accuracy is smaller. The method presented in [19] shows limitations when applied to L3i-MICA dataset, NUS II dataset and Triesch dataset, due to the small number of detected key points. With Caros dataset, all three methods obtain high accuracies, and our proposed method get the best with slightly higher. The cause of this result is that this dataset is simple and satisfy properties suitable to Dardas' method. Tab. 4.4 shows the main diagonal of the confusion matrix obtained from Dardas' method, original KDES, and our method with the same dataset (L3i-MICA dataset). Table 4.5, 4.6, and 4.7 are the confusion matrixes obtained with Dardas' method, original KDES, and our proposed method for 21 hand posture classes in L3i-MICA dataset. Our method improves the recognition accuracy for almost hand posture classes (19 over 21). Especially, for class #7 and #8, the recognition accuracy increases 30% after applying our improvement.

Table 4.4 – Main diagonal of the confusion matrix (%) with the original KDES and our proposed method for 21 hand posture classes in L3i-MICA dataset

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Dardas' method	53.1	38.3	51.4	67.7	63.6	41.7	23.2	56.7	15.3	13.8	15.9	24.5	31.3	31	22.6	20.7	25.2	17.3	13.5	37.5	64.5
Original KDES	100	69.4	91.2	95.6	73.2	90.4	66.1	45.2	74.3	84.1	83.4	100	95.8	93.6	98.6	100	91.6	92.2	70.1	93.1	67.2
Our proposed method	100	71.1	99.6	93	80.8	96.1	**83.3**	**74.2**	82.9	92.7	92.6	100	100	100	100	100	94.7	99	77.7	92.6	86.9

Table 4.5 – The confusion matrix of Dardas' method for 21 hand posture classes of L3i-MICA dataset.

P	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
1	**53.1**	0.5	1.0		11.1	11.1	1.0	3.9	1.0		0.5	2.9		6.3	1.0	1.9			2.4	1.4	1.0
2	3.1	**38.3**	1.3	2.6	2.6	6.6	5.3	1.3	1.3	5.7	3.1	1.8	7.0	2.6	0.9	2.2	4.4	6.2	1.3	0.4	1.8
3		1.4	**51.4**	5.6	0.5	1.9	3.7	1.9	5.1	2.8	2.8	1.4	3.7	0.9	2.3	1.4	3.7	4.7	2.3	0.5	1.9
4	0.4	1.8	2.7	**67.7**		1.3		0.4		1.8	5.8	1.3		3.1	0.4	0.9	2.7	2.2	4.4	2.2	0.9
5	10.2				**63.6**	11.6	1.8	0.4				2.7	0.4	0.4					0.4	4.4	4.0
6	3.4	0.5	1.0		13.2	**41.7**	6.9	3.4	2.5	0.5	1.5	2.5	2.0	13.2	1.5	0.5				0.5	5.4
7		2.2	1.8	0.9		4.5	**23.2**	7.1	7.6	4.9	4.5	5.8	4.9	4.5	11.2	7.1	4.5	2.7	1.3		1.3
8	1.4	0.5	3.2	1.8		4.6	5.5	**56.7**	6.0	0.9	0.9	1.4	1.8	2.3	3.2	0.5	3.7	1.8	0.5	0.5	2.8
9	1.7	1.3	3.5	1.3		0.9	**15.7**	5.2	***15.3***	4.8	3.5	3.1	1.3	3.9	7.0	10.5	12.2	3.1	3.5		2.2
10	0.9	2.2	3.4	1.3			3.4	3.9	11.2	***13.8***	5.2	6.5	5.6		5.6	8.2	**14.2**	11.2	2.2	0.4	0.9
11	0.4	4.4	1.3	3.1		0.4	6.2	4.9	9.7	7.1	**15.9**	2.7	4.4	8.4	7.1	5.3	6.2	10.2	1.3	0.9	
12	1.3	1.3	1.3	2.6	0.4	6.6	3.9	7.9	2.6	1.7	0.9	**24.5**	14.4	2.2	10.0	3.5	8.7	2.2	1.7	0.4	1.7
13		0.4	3.6			0.4	3.1	1.8	4.5	2.7		8.0	**31.3**	2.7	2.2	6.7	25.4	6.7			0.4
14	2.6	0.9	0.9	1.7	1.7	6.1	4.8	7.9	8.3	0.9	5.7	4.8	0.4	**31.0**	5.2	0.9	2.6		3.5	5.2	4.8
15		2.5	3.8	0.4	0.4	6.3	2.1	7.5	5.9	3.8	5.0	4.2	6.7	5.0	**22.6**	7.5	9.6	5.4		0.4	0.8
16		0.9	0.5	1.8			8.8	5.1	9.2	8.8	5.5	4.6	5.5	0.9	3.2	**20.7**	18.0	2.3	3.7		0.5
17		4.1	2.3	0.9		0.5	3.6	3.6	9.0	4.5	5.0	2.7	14.4		6.8	3.2	**25.2**	12.2	0.9	1.4	
18		8.6	1.8	1.8		0.5	2.3	1.4	8.2	4.1	4.1	1.4	10.0		3.2	7.7	**26.8**	***17.3***	0.9		
19	3.5	1.7	6.1	0.4	0.4	2.2	3.9	1.7	7.9	3.9	**16.2**	0.9	4.4	7.4	3.9	10.0	5.7	1.3	***13.5***	4.8	
20	0.9	0.5	0.9	2.3		1.4	5.1	4.2	5.1	3.7	9.7	1.4	2.8	6.0	3.7	1.4	1.4	0.5	0.5	**37.5**	11.1
21		1.3	0.4	0.9	0.4	0.4	2.6	4.7	4.7	2.1		0.9	3.4	0.9	1.7	0.9	0.4	2.1		7.7	**64.5**

Table 4.6 – The confusion matrix of original KDES method for 21 hand posture classes of L3i-MICA dataset.

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	**100**																				
2		**69.4**					4.7	2.6	1.7	0.4				7.8							13.4
3		4.4	**91.2**								0.4		4								
4				**95.6**	0.4		3.9														
5	0.4			2.7	**73.2**				11.6	8			4								
6						**90.4**							1.3						8.3		
7		6.3		3.8			**66.1**	8.4	9.2							1.7		0.4	3.8	0.4	
8		7.4			0.5		21.7	**45.2**	3.7		0.9								9.2	3.7	7.8
9		3.2	3.2		5.4		9.9	0.5	**74.3**	0.9									2.7		
10			0.9		0.5				14.1	**84.1**							0.5				
11			2.2			12.7				1.7	**83.4**										
12												**100**									
13					0.5						0.5		**95.8**	3.2							
14													6.4	**93.6**							
15															**98.6**			0.5	0.5	0.5	
16																**100**					
17			0.4	3.1		3.1	0.4										**91.6**	0.9		0.4	
18						3.4												**92.2**	2.9	1.5	
19							1.8	2.2	1.3		4.5								**70.1**	8.5	11.6
20							6.5												0.5	**93.1**	
21		3.9						8.7		0.4									16.6	3.1	**67.2**

Table 4.7 – Confusion matrix (%) of our method for L3i-MICA dataset.

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	**100**																				
2		**71.1**	8.2					7.8		4.3			3.4	0.9							4.3
3			**99.6**										0.4								
4				**93**	0.9		6.1														
5	0.9			4.5	**80.8**				8	5.8											
6						**96.1**													3.9		
7		2.1		8.4			**83.3**	4.2									0.4	1.7			
8		5.1					8.3	**74.2**			0.9								3.7		7.8
9		1.4			5.4		2.7	5	**82.9**	2.7											
10			0.9					0.5	5.9	**92.7**											
11						5.7					**92.6**		1.7								
12												**100**									
13													**100**								
14														**100**							
15															**100**						
16																**100**					
17										0.4		4					**94.7**	0.9			
18																		**99**		1	
19				0.9			2.7				2.7								**77.7**	4.9	11.2
20							0.5				1.8								5.1	**92.6**	
21		0.9						7			0.4								4.8		**86.9**

Table 4.8 – Main diagonal of the confusion matrix (%) obtained with our method for 21 hand posture classes in L3i-MICA dataset for three testing cases

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Origianal KDES	100	69.4	91.2	95.6	73.2	90.4	66.1	45.2	74.3	84.1	83.4	100	95.8	93.6	98.6	100	91.6	92.2	70.1	93.1	67.2
Case 1	100	81	95.1	93.4	82.1	90	76.6	73.7	82.4	86.8	88.2	100	100	97.9	100	100	94.7	97.1	81.7	93.5	90
Case 2	100	75.9	99.1	93.4	78.1	100	84.9	76.5	79.7	81.4	91.3	100	100	100	100	100	93.8	100	75	92.6	90
Case 3	100	71.1	99.6	93	80.8	96.1	83.3	74.2	82.9	92.7	92.6	100	100	100	100	100	94.7	99	77.7	92.6	86.9

Table 4.9 – The confusion matrix with **Case 1** (Apply only adaptive patch) for 21 hand posture classes in L3i-MICA dataset.

Posture	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	**100**																				
2		**81**	6.9				0.4	0.4			2.6		0.4	3.9							4.3
3		1.8	**95.1**										3.1								
4				**93.4**			6.1						0.4								
5	1.8		0.4	2.7	**82.1**			0.4	9.4	3.1											
6						**90**					1.7		0.4						7.9		
7		3.8		5			**76.6**	7.9	4.6									1.7	0.4		
8		7.4					11.5	**73.7**	0.5		2.8								1.4		2.8
9					9.5		5	1.4	**82.4**	1.8											
10			0.5		3.2			0.5	8.6	**86.8**				0.5							
11			3.5			6.1					**88.2**		2.2								
12												**100**									
13													**100**								
14													2.1	**97.9**							
15															**100**						
16																**100**					
17												4					**94.7**	1.3			
18							2.5											**97.1**		0.5	
19				0.4		2.2	0.4	0.9			0.4								**81.7**	6.7	7.1
20																			6.5	**93.5**	
21		0.9						1.7											7.4		**90**

Table 4.10 – The confusion matrix with **Case 2** (Combine both adaptive patch and hand pyramid structure) for 21 hand posture classes in L3i-MICA dataset.

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	**100**																				
2		**75.9**	5.2					3					1.3	6							8.6
3		0.9	**99.1**																		
4				**93.4**	0.9		5.7														
5	1.3			4	**78.1**			0.4	9.4	6.3								0.4			
6						**100**															
7		2.9		8.4			**84.9**	0.8	2.1												0.8
8		6					9.2	**76.5**													8.3
9		0.5			7.2		2.7	7.2	**79.7**	2.7											
10			0.9						16.8	**81.4**				0.9							
11						8.7					**91.3**										
12												**100**									
13													**100**								
14														**100**							
15															**100**						
16																**100**					
17										0.4		3.6					**93.8**	2.2			
18																		**100**			
19				2.2		0.4													**75**	8.5	13.8
20					0.5		0.9				0.9								5.1	**92.6**	
21								1.7			0.4								7.9		**90**

4.4.5 Experiment 3: Evaluation of our improvements

In this experiment, to obtain a detailed analysis of the behavior of our three improvements, we perform different comparisons on L3i-MICA dataset. As described in Section 4.3, our method has three improvements: adaptive patch, normalized gradient orientation, and hand pyramid structure. We observe the performance of the method in the following cases: **Case 1**: Apply only adaptive patch; **Case 2**: Combine both the adaptive patch and hand pyramid structure; **Case 3**: Combine all improvements (our proposed method).

The obtained result is shown in Tab. 4.11. The confusion matrixes for 21 hand posture

Table 4.11 – Effects of our improvements in three cases: Case 1: Apply only adaptive patch; Case 2: Combine both the adaptive patch and hand pyramid structure; Case 3: Combine all improvements

Method	Original KDES	Case 1	Case 2	Case 3
Accuracy (%)	**84.4**	**90.6**	91	91.2

classes in L3i-MICA dataset with the **Case 1**, **Case 2**, and **Case 3** are shown in Tab. 4.9, 4.10, and 4.7 respectively. We can see that the adaptive patch improvement makes a great difference. The performance increases 6% after applying the adaptive patch instead of the uniform patch in the original method. With this dataset, the hand pyramid structure and normalized gradient orientation have a minor contribution. The hand pyramid gives a suitable representation for upright frontal hand postures in that the difference between postures is the configuration of the fingers (open or closed). However, this constraint is not always satisfied in the working datasets so that we can not see the improvement. Even though, the performance obtained with this pyramid is at least equal to that of the general pyramid. Tab. 4.8 provides the recognition accuracy obtained for 21 hand posture classes in three testing cases. From this result, one time again, the adaptive patch improvement shows that it has an important impact on the recognition accuracy. This improvement makes the recognition accuracies of 15 over 21 classes increases. The spatial hand posture is relatively sensitive. Its robustness depends on the characteristics of the hand posture.

4.4.6 Computation time

Concerning computation time, our method takes averagely 0.3s per image when working with 50×100 image using Matlab 8 (R2013a), Window 64-bit Operating System with processor Intel(R) Core(TM) i5-2520M.

4.5 Conclusion

In this chapter, we have introduced a method adapting kernel descriptor into hand posture recognition. The proposed hand representation is proved that it is robust for hand postures recognition. Our main contributions are:

- We propose an adaptive patch that helps the proposed method to be invariant to scale change.
- We propose a normalization for gradient orientation in patch based on soft dominant gradient direction of patch that is sum of all the gradient vectors of pixels in patch. This normalization helps gradient features in patch is invariant with rotation.
- We proposed a spatial hand structure that allow to capture the proper characteristics of the hand postures.
- In overall, we proposed a new and robust method for hand posture recognition based on kernel descriptor.

We have tested our proposed method on four datasets. The experimental results indicate that the proposed method is better than recent success method for hand postures recognition [19] that use SIFT features and SVM. The proposed method based on KDES with our improvements has been proved better than original KDES method when applying into hand postures recognition problem. Our improvements also can apply into generic object recognition. The proposed spatial hand structure is designed and suitable for our constraint. However, the proposed invariant rotation gradient kernel descriptor and the proposed adaptive patch technique can be used for any other objects recognition system. Based on the experimental results, we highly recommend to use our proposed method to build a human-robot interaction using hand postures.

Chapter 5

Application

5.1 Introduction

We have integrated our proposed hand detection and recognition approaches into a fully hand posture recognition system and deployed this system in a human-robot interaction application. The main purpose of this is to evaluate the whole system in a real application and to demonstrate that this system could be applied successfully in real situations. The chosen application is service robot in a library. This application is built in the framework of a research project at MICA Institute, funded by Orange lab in Japan with the aim is to evaluate the behavior of the user while interacting with the robot. In this application, the robot plays the role of librarians in a conventional library. In the following, we will describe in more detail how we build this application and validate it.

5.2 Service robot in library

5.2.1 Investigation of library environment and end user requirements

We have visited the Ta Quang Buu Library of Hanoi University of Science and Technology to understand the organization of a library as well as borrowing and returning book activities. The library spans on several floors of the building and is organized into specific rooms such as storing room, reading room or borrowing room. The users are mainly students. Books and users have been managed through 1D barcode management system (Fig.5.1a). Every year, new students will be provided with an account for using library services. This account will be created based on the student profile including his/her student card number and the barcode onside the card. The library provides different station machines allowing students to lookup:

a) b)

Figure 5.1 – a) Ta Quang Buu Library management using 1D barcode, the consultation is carried out at station machine; b) Robot acts as a librarian, the end user could interact with it using hand postures.

- User information (including user account information and list of books borrowed by this user) by entering a barcode number on the student card.
- Information of a borrowed book by entering the barcode ID of this book or search in the borrowed list.

The library used the library database management software named VTLS (http://www.vtls.com/) to manage the users, books, as well as book borrowing and returning information. When a user wants to get into the library, he/she needs to show his/her student card to a receptionist of the library. In reading rooms, the user can lookup information using OPAC interface (a module of VTLS) on a station machine. In borrow rooms, the user brings books that they want to borrow to the librarian. According to librarians as well as users, the most frequent and important queries are the one about overdue books.

5.2.2 Definition of human-robot interaction scenarios

After investigating the library and its activities, we propose to simulate the library Ta Quang Buu by implementing a room (that plays the role of reading and borrowing room) inside the showroom of MICA Institute. The simulated library is a room of size 3m x 5m

in which we equip with some tables, chairs, bookshelves. All are similar to a reading room in the library so that the human can feel as in a real library (see Fig.5.1b and Fig.5.2). In

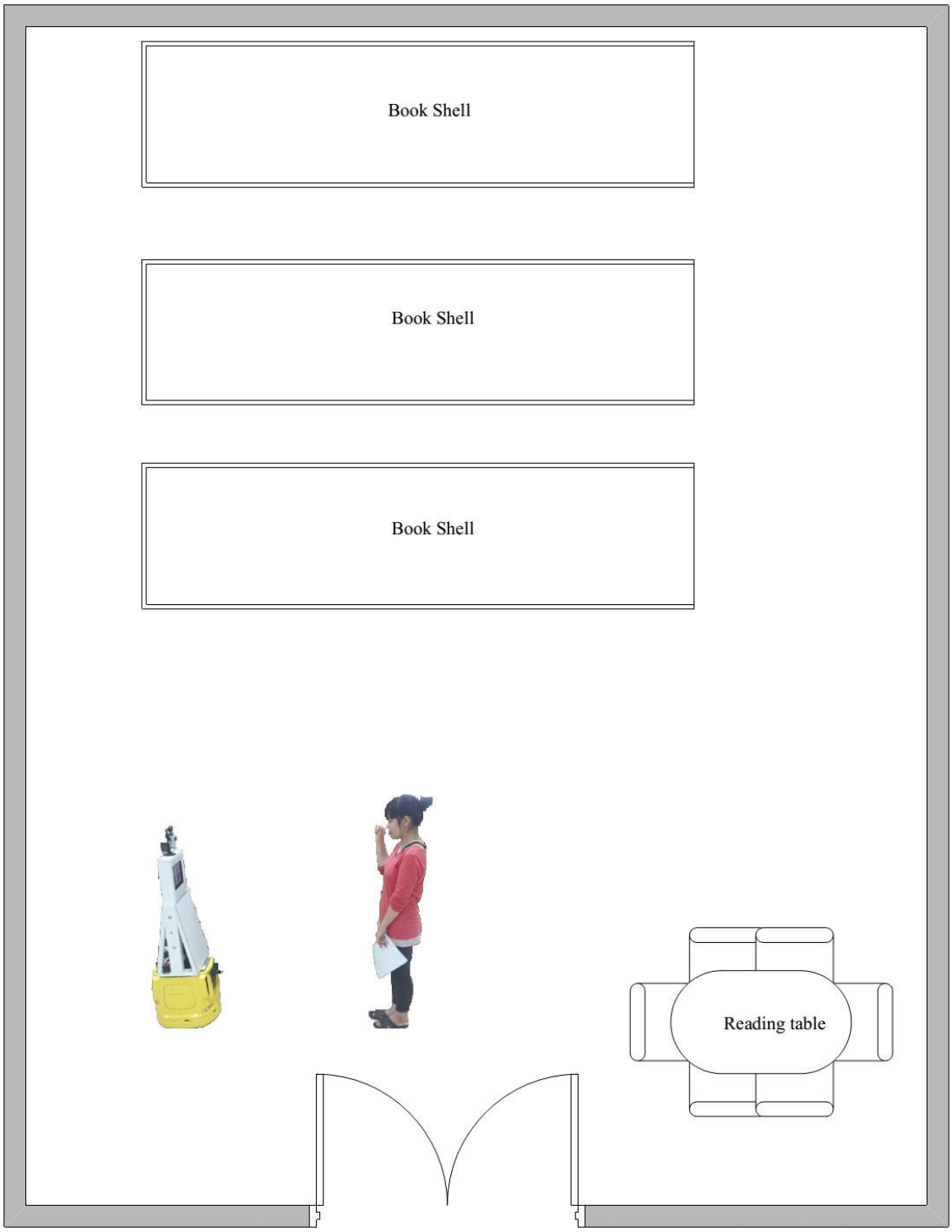


Figure 5.2 – Stimulated library for testing robot services

this context, the scenarios are played by two actors: a human and an assistant robot in the library. We focus on some general and basic requirements that a human needs and often

uses in the library such as search for user information or book.

To define interaction scenarios, we invent situations and assign roles to human and robot as follows (see Fig.5.3):

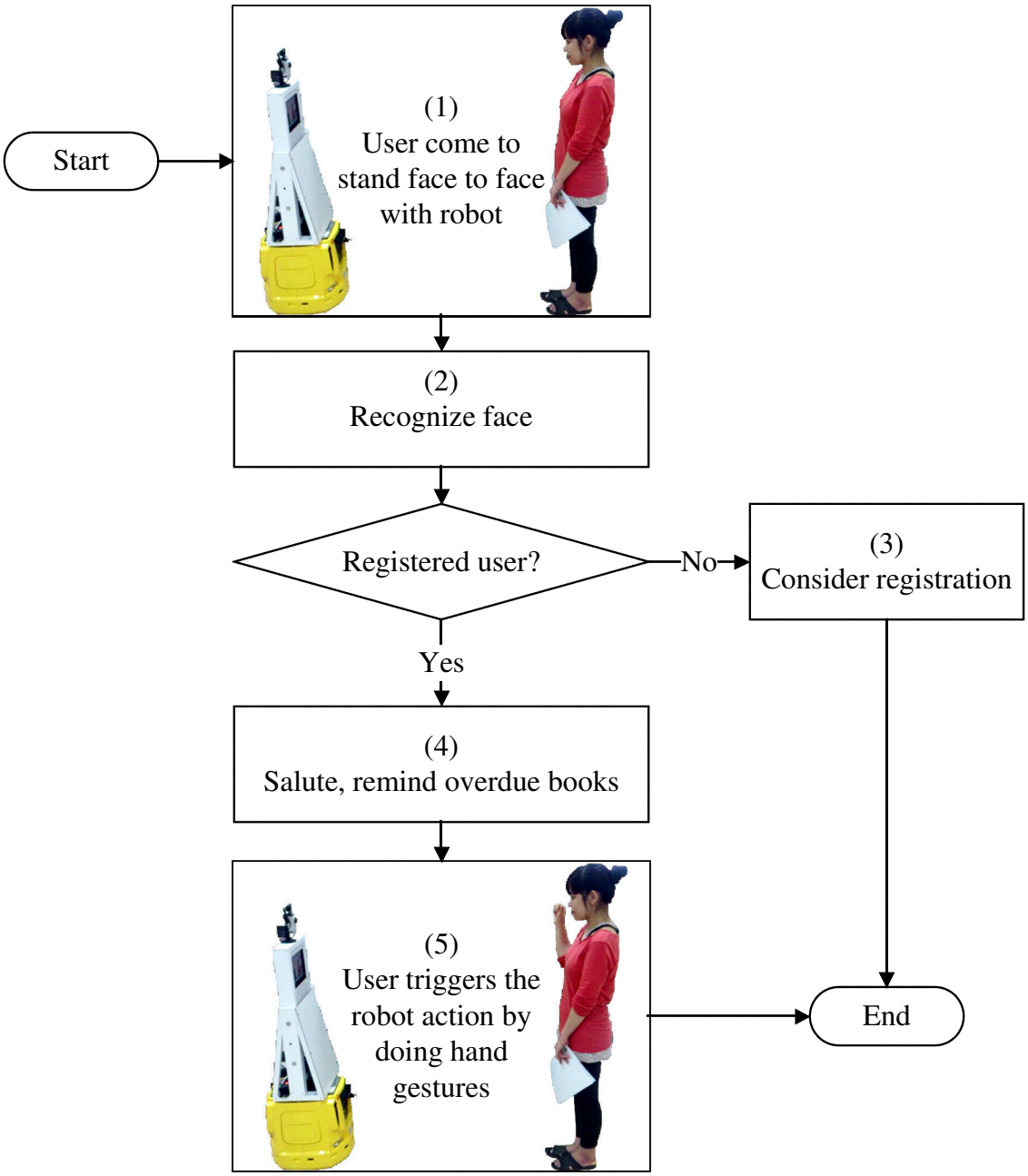


Figure 5.3 – The activity diagram of the system

1. *User* comes and stands in frontal of the robot.
2. *Robot* captures photos of user face and recognizes him.
   (a) If the robot can not recognize the user, go to (3).

(b) If the user is recognized successfully, go to (4).

3. *Robot* considers as the user has not been registered before. The robot will say by synthetic voice: *"Hello, you are not registered. Please do the registration first"*.

4. *Robot* says : *"Hello, welcome to the library! I'm ready for your consultation"*. The robot searches overdue books borrowed by this user and reminds him by displaying all overdue books on the screen.

5. Once getting login, user can trigger the robot commands by hand gestures to:

   (a) Ask for user information

   (b) Lookup borrowed books, select a book

   (c) Listen summary of a selected book

   (d) Stop interaction with the robot

The step 5 is described detail in the statechart diagram (Fig.5.4). Before user login successfully, the robot shows the "standby screen". After login, the robot reminds overdue books borrowed by this user by displaying on the "Borrowed book list (S1)" screen (Fig.5.5). At the S1 screen ("Borrowed book list"), the user can trigger following commands:

- Go to next or Next page (command A1), Back (command A2) to lookup the borrowed book list. The user still stays in the S1 screen.

- Open user information (command A3): the robot shows the "User information" screen (S2) (Fig.5.6).

- Open book information (command A4): the robot shows the "Book information" screen (S3) (Fig.5.7).

At the S2 screen ("User information"), the user can trigger the command Back (command A2); the robot goes back to the S1 screen ("Borrowed book list").

At the S3 screen ("Book information"), the user can trigger following commands:

- Back (command A2): the robot goes back to the S1 screen ("Borrowed book list").

- Read book (command A5): the robot read the summary of a selected book and still stay at the S3 screen ("Book information").

At any screen (S1, S2, or S3), the user can trigger the Stop command (command A6) to stop interactive with the robot.

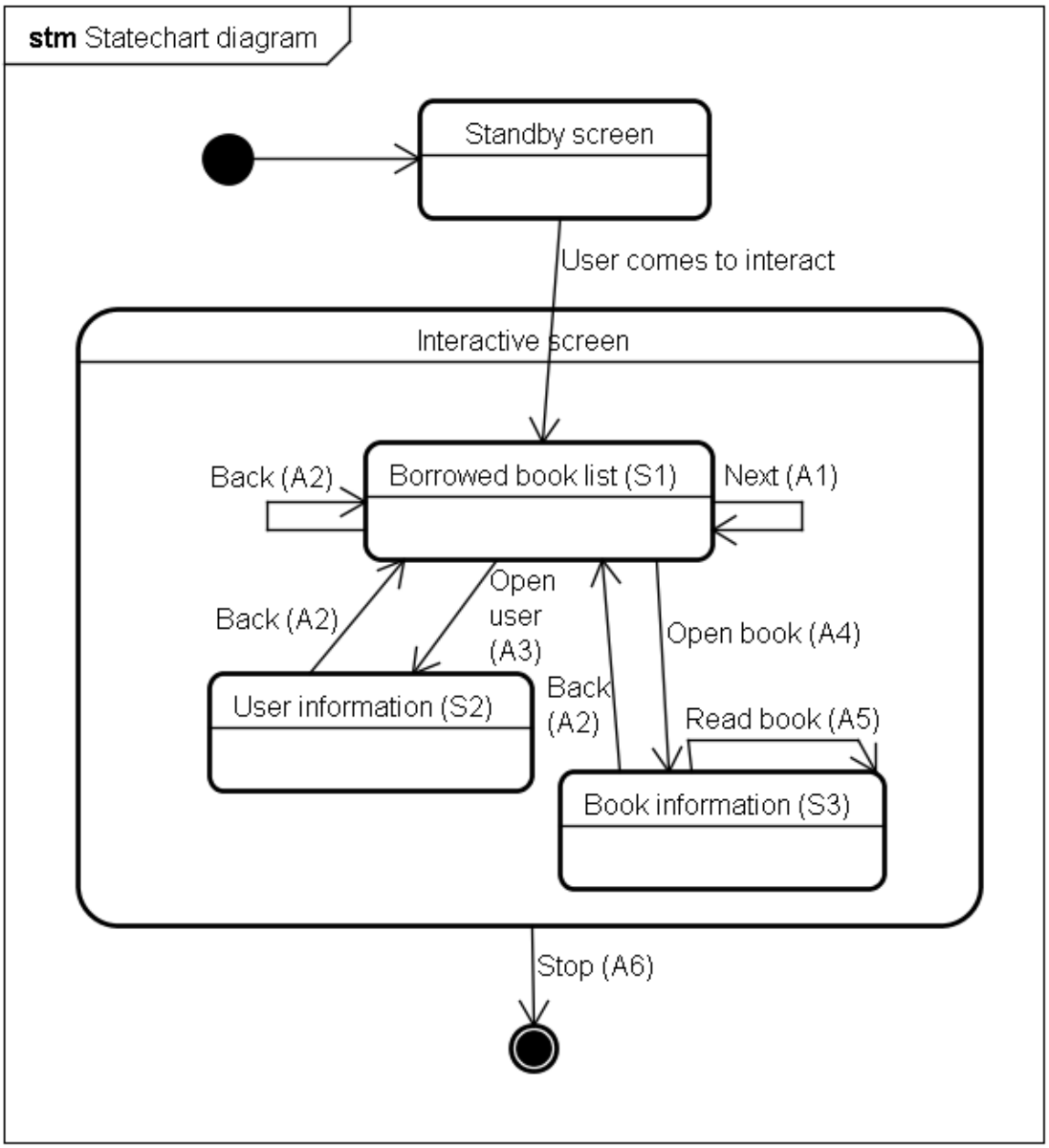


Figure 5.4 – The state chart diagram of the system

5.2.3 Design of postural command vocabulary

With the above scenario, we design a set of hand postures and map them to set of commands for human-robot interaction. It covers two groups of command: Interface Controlling and Consultation. Figure 5.8 shows the commands with corresponding hand postures.

- Interface controlling (four commands): Back, Next, Next page, and End.
- Consultation (three commands): Open user info, Open book info, Read book summary.

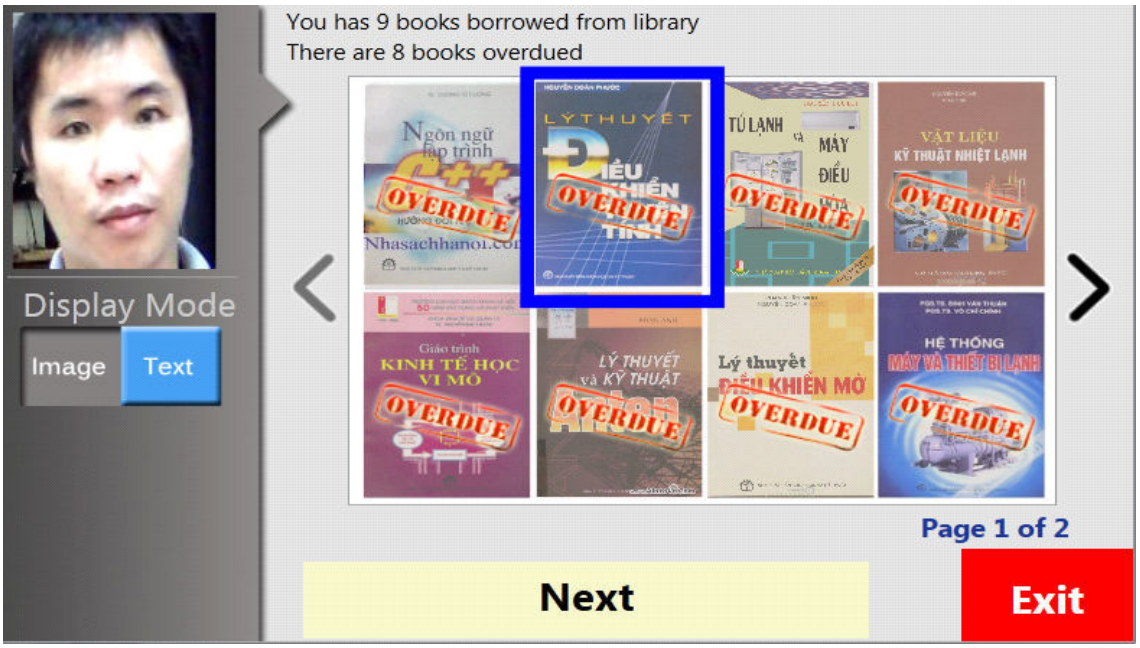


Figure 5.5 – The screen borrowed book list (S1)

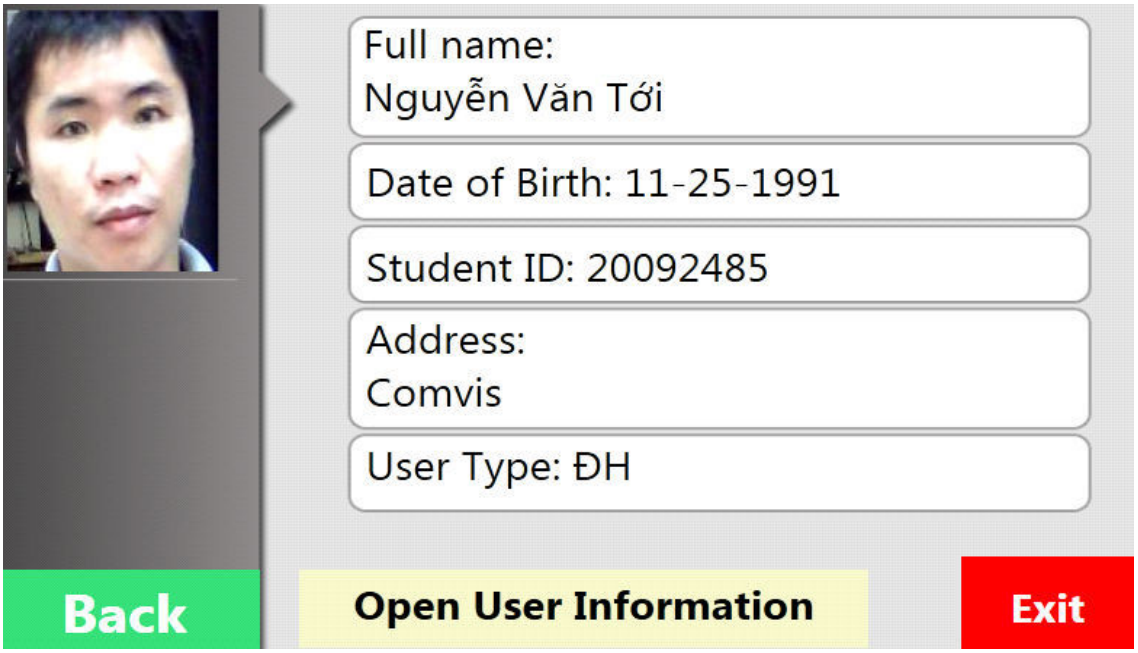


Figure 5.6 – The screen user information (S2)

We did not focus on analyses to find an optimal hand gesture vocabulary (GV) because it is costly and time consuming [85, 86]. In addition, GV design research is not our current goal. We will take account into this task in the near future. In this thesis, we selected hand postures from our set of postures presenting in Chapter 4 with a simple way. Based on performance measures of GV considering in [85] (intuitiveness, comfort, and recognition accuracy), we ourselves selected the GV with regard to our qualitative

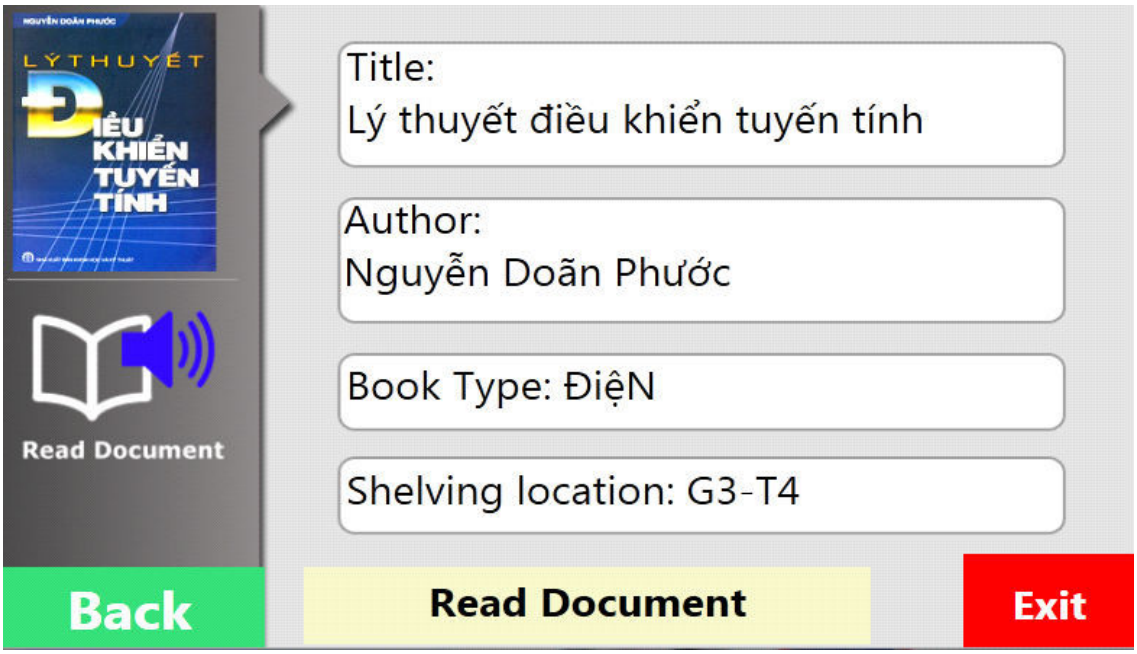


Figure 5.7 – The book information (S3)

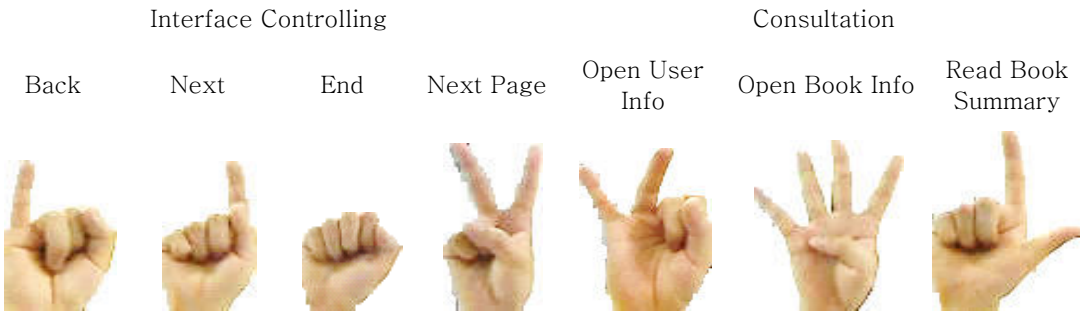


Figure 5.8 – Set of postural commands

intuitiveness&comfort analysis and recognition accuracy in the experiments in Chapter 4.

5.2.4 Deployment of the proposed method on robot for human-robot interaction

To deploy such method for hand posture recognition, we integrate the modules of hand detection and hand posture recognition on a robot. However, these modules are designed to take still images as input. For real application, the camera captures consecutively frames then the question is how to make the decision of recognized command at a certain time in order to control the robot.

To deal with this question, several methods require the user to start controlling the

robot by raising his hand, keep the posture in a certain time (normally from one to two seconds) and finish his command by putting his hand down. By this way, the system can determine more easily the period of playing hand postures then follow a voting scheme on frame by frame recognition results. This approach does not allow the user to make consecutive commands without putting down the hand. In addition, because the command is only executed after segmentation and recognition, the response could be slow.

We propose a strategy that detects and recognizes hand posture every frame and makes the decision based on the number of consecutive similar recognized hand postures and the duration between them (Fig.5.9). This method avoids the mentioned drawbacks and gives satisfied results as shown in experiments section.

Suppose at time k, the system captures frame F_k and does hand detection and posture recognition. If the posture P is recognized, we confirm a command corresponding to the posture P to control the Library Management System and if the following conditions are satisfied:

1. The system detects the same hand posture P at fpc frames before $F_{k_1}, F_{k_2}, ...F_{k_{fpc}}$
2. The system detects nothing at remaining frames counting from F_{k_1} to F_k
3. The number of frames between every couple of frames $F_{k_i}, F_{k_{i+1}}$ dees not exceed fpr frames.

After sending a command to control the Library Management System, we reset parameters to start the process of the upcoming command. Fig.5.10 visualizes how to make a decision in our deployment.

5.3 Experiments

The aim of experiments is to evaluate the performance of the hand posture-based human-robot interaction in library context. The robot used for testing is a PCbot 914. It is like PC under the form of robot with 1 Gbyte of RAM DDR 2. We have mounted a frame to keep a small monitor and a camera on the robot at a height convenient for communicating with human. For evaluation, the robot is put at a corner of the simulated room, neon-lighting condition, and office background. Besides, we had developed a simulated library management system with main functionalities such as VTLS. In addition, as presented Section 5.2.2, face recognition is used to login to the library management system. The face detection and recognition module has been developed in a previous works [89]. Fig.5.11 shows an example image captured from working session. The user makes the command "Open Book Info", the system recognizes and goes to the book info page.

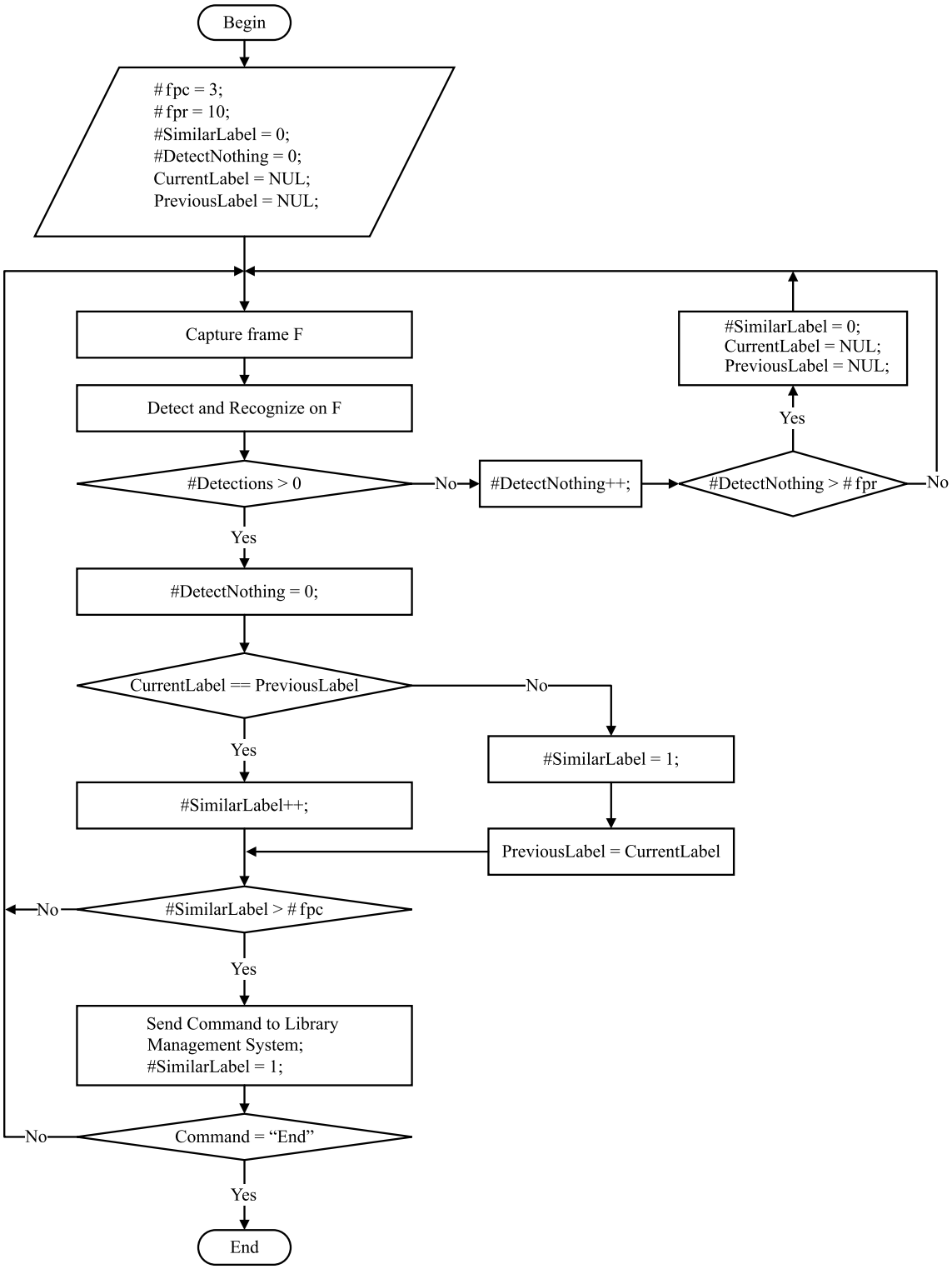


Figure 5.9 – System deployment.

We invite 10 subjects to play the pre-defined scenario. The data coming from the first five subjects are used for training the multi-class SVM for hand posture recognition

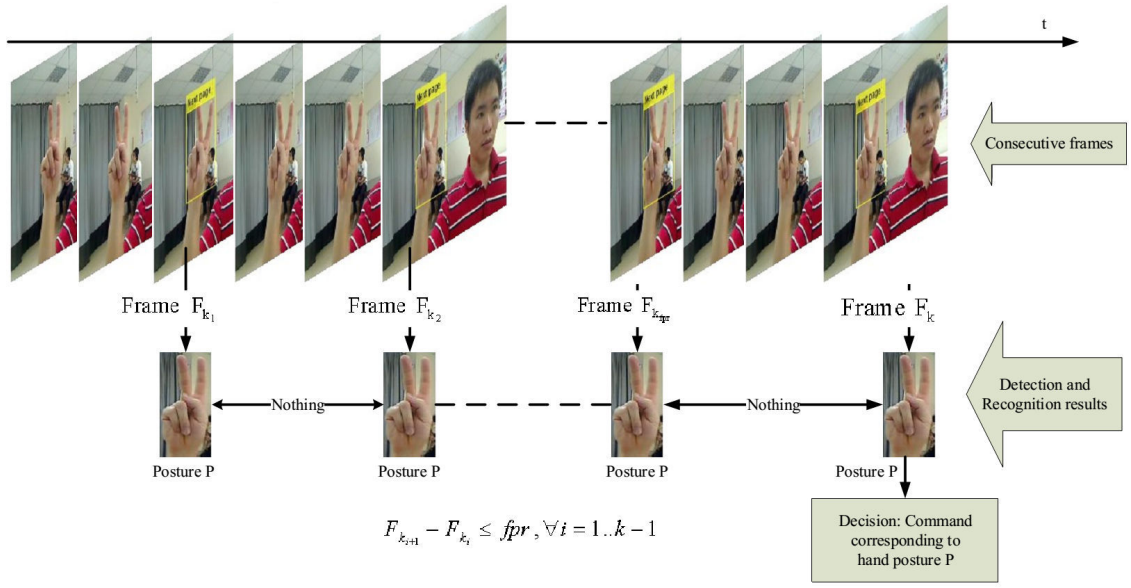


Figure 5.10 – System deployment: making decision.

Figure 5.11 – An example image captured from working session.

module while the remaining subjects participate to evaluate the system. We have to train a new classifier instead of reuse the trained classifier in Chapter 4 because of different numbers of classes. For hand detection module, we use the hand detector that is trained in Chapter 3. Each subject is required to trigger any command using hand postures in order to control the robot so as one posture could appear several times. We collect all the frames the system processed. The ground truth is created manually on the original captured data

without any information of the results generated by the system. We compare the results with the ground truth to evaluate the performance of the system.

Table 5.1 shows the confusion matrix of the results. We add the column "Other" for

Table 5.1 – The confusion matrix of command recognition.

Action	Back	Next	Open book info	End	Open user info	Next page	Read book summary	Other
Back	30	0	0	0	0	0	0	0
Next	0	30	0	0	0	0	0	0
Open book info	0	0	30	0	0	0	0	0
End	0	0	0	30	0	0	0	0
Open user info	0	0	0	0	30	0	0	0
Next page	0	0	10	0	0	10	0	10
Read book summary	0	0	0	0	0	0	30	0
Other	0	0	0	0	0	20	0	0

the missed recognitions and the row "Other" for the false recognition. The system obtains very good performance on most of hand postures except the "Next page". In the overall result, the average Precision is 83%, the average Recall is 91%, and the average F-score is 86%. We can see that the hand pyramid help distinguish hand postures very similar with different open/close fingers. In Figure 5.8, the postures for "Next Page" and "Open User info" are very similar except for the fist posture and the particular fingers involved in the postural command. The confusion matrix in Table 5.1 however, does not reflect this confusion. The confusion matrix shows a perfect performance for the "Open User info". We see a relatively degraded performance for the "Next Page" command. Fig. 5.12 illustrates an example of wrong recognition. The hand is well detected. The hand posture recognition module, however, makes a wrong decision because of the participation of background information in the hand bounding box that makes the gradient of this rectangle region is similar that of "Open book info" posture. Removing the unexpected effects of background on hand posture recognition is a further topic need to research.

In this application, we only evaluate the system on the final command recognition results. We did not individually evaluate the hand detection and hand posture recognition on frames because when a hand candidate is detected on a frame, the average time computation of next steps is 0.6 second. During processing steps after hand detection, a large number of frames are skipped. Therefore, we can not evaluate each module on these

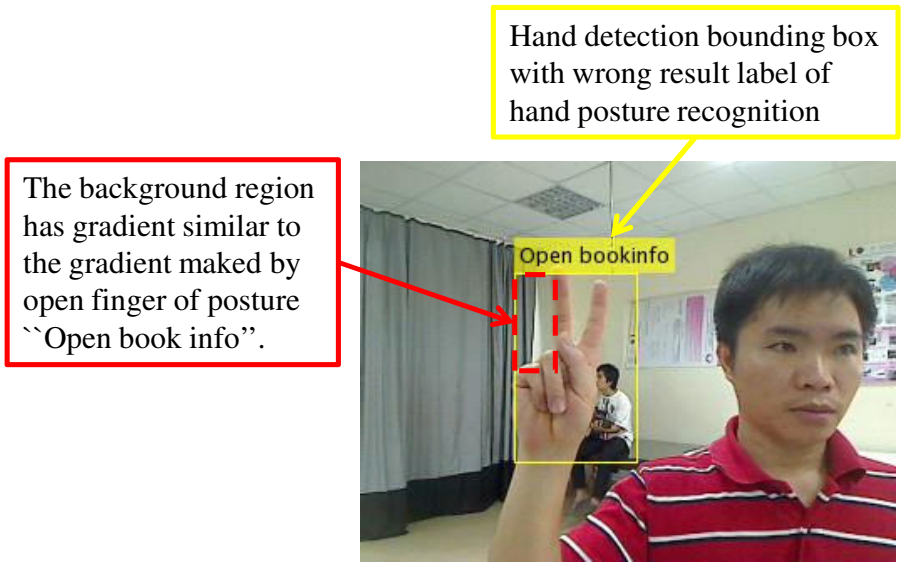


Figure 5.12 – An example of wrong recognition. A “Next page” posture is recognized as “Open book info”.

frames.

All of the users who used this application have following common comments. They are interested in the interacting with the robot using hand posture. However, the robot response to user’s command slight slowly. The system also does not works really smooth. The accuracy of the system is quite high. Sometimes the robot makes a wrong response or no response to user. In this case, user can slightly move to change the position of the hand to make slight background change.

In overall, the application system is acceptable to use. However, it is necessary to improve the response time as well as to reduce the unexpected effects of background. To do this, we need to reduce computation time of hand segmentation and hand posture recognition after detecting hand. To avoid unexpected effects of background, we need to find a better hand segmentation based on detected hand center region. Another approach we can apply is that sliding window on the region expanding from detected hand center region. In this case, the expanded region is large enough to contain whole hand. However, we have to concern about time computation when use this approach. Additional task we have to do is that do research to design a optimal hand postures vocabulary.

5.4 Conclusion and Future works

This chapter shows a fully automatic hand posture recognition system, integrated successfully on a robot for human-robot interaction application in the library context. Extensive experiments show that the proposed system works well in real situation and responses in a

quite satisfied time duration. We also observed that the people feel quite comfortable and funny to use hand postures to communicate with the system. Even thought, it is necessary to do research to improve the performance of the system such as to reduce computation time, to avoid unexpected effects of background, and to design an optimal GV. In the long term future works, we will combine hand postures with other modality such as speech to make the interaction more natural and efficient. The chapter shows a specific context but the framework could be extended for more generic application.

Chapter 6

Conclusions and future works

6.1 Conclusions

This thesis presents a hand posture recognition system consisting two phases that are hand detection and hand posture recognition. An application based on detection and recognition results on frames is also deployed.

The proposed system can work on the input image containing a small hand region with cluttered background in real indoor environment. The proposed method can be applied directly to build human-machine interaction systems. The application in Chapter 5 is an example.

The proposed hand detection method is based on Viola-Jones detector therefore it has all advantages of this good detector. The hand detection method is able to perform in real time, robust under different lighting conditions and invariant to scale changes. Besides, our hand detection method avoids the unexpected effects of background because of using proposed internal Haar-like features. The experimental results indicate that the hand detector using internal Haar-like feature obtains better performance than the hand detector using traditional Haar-like features. On MICA-L3i dataset, internal Haar-like features help the F-score of Viola-Jones detector improve from 0.69 to 0.84. However, when we train a detector using internal Haar-like feature, positive samples without background have less information than those containing whole hand with background. This can causes more false positive detections.

For hand posture recognition, we proposed a new hand representation based on kernel descriptor with three improvements. These improvements makes KDES invariant to scale change and make patch-level feature invariant to rotation. The experimental results show that improved KDES is more robust than original KDES and a state of the art method for hand posture recognition. The accuracies of the proposed method are 6.8%, 2.1%, and 1% higher than those of original KDES and 56.7%, 54.2% and 35.9% higher than the Dardas's

method [19] on MICA-L3i dataset, NUS II dataset, and Triesch dataset respectively.

The proposed hand representation belongs to implicit representation approach, hence it does not require a clear hand segmentation. In spite of this, we designed a special pyramid structure that is suitable for the hand. This structure helps to capture configurations of the hand posture especially in cases of upright frontal hand postures with the different open/close fingers. In other cases, the performance of KDES with hand pyramid is still not lower than general pyramid structure.

We have applied the improved KDES into some other object recognition and obtained better results than original KDES and the current methods in the same fields. However, there are some limitations of hand representation in different images: the unexpected impacts of background and hand positions in images.

To illustrate the applicability of our proposed hand posture recognition system as well as evaluate the proposed system, we deployed a human-robot interaction system in library context using a subset of our hand postures set. With this system, the user can use predefined hand postures to trigger the robot actions. In this system, we use only one normal 2D camera. The success of this real application allow us believe that we can extend to apply directly the proposed hand posture recognition method into a real suitable application. We also can believe that the proposed method can contribute a good role in a RGB-D based hand posture/gesture recognition systems.

6.2 Future works

We proposed some improvements for hand detection and hand posture recognition. These improvements dedicate a small portion to the progress of developing fully automatic human-machine interaction systems using hand gestures. As we presented our objectives and motivations in Chapter 1 Introduction, we want to continue to do some research works based on the results of this works. In addition, there are a number of research questions and ideas come when we have to finish this thesis. In this section, we summary the selected works we would like to do after this thesis.

1. **Short term**:

   In the near future, we will do some technical tasks to improve the performance of the system and complete some experiments to get fuller evaluation for the proposed method.

(a) *Learn to build the set of basis vectors and match kernel parameters from hand images*:

The performance of hand posture recognition based on KDES will improve when the set of basis vectors and match kernel parameters are selected considering the particular characteristics of hand images (selection based on set of hand images instead of ImageNet).

(b) *Complete experiments*:

Due to the lack of time, we have not done some deeper experiments we want to do to evaluate more detail our proposed method such as the role of normalized gradient orientation in the patch. In the near future, we want to spend time to complete these experiments. We also would like to build some hand posture/gesture dataset with evaluation proposals.

2. **Long term**:

(a) ***For hand detection and segmentation***, we proposed new concepts that are Internal features and Internal Haar-like features. We pointed out that Internal Haar-like features have some advantages compare with normal Haar-like features. After studying on this topic, we find some research topics coming as follow:

i. *Design a set of Haar-like feature that is more suitable to hand*:
In this thesis, we used generic set of Haar-like features that is proposed for face and general object detection. A specific set of Haar-like feature designed for hand might make hand detector more robust.

ii. *Develop a hand detection method based on internal features and context information of internal features*:
In this thesis, we only exploit the internal Haar-like features extracting from hand center region. We have not exploited any context information. In the future, we want to develop a hand detection method that added context informations. In this direction, all candidates of rigid components of hand will be detect individually then the best combination of a subset of these components will be found as detected hand/hand posture. A problem we will have to tackle is how to reduce time computation.

iii. *Find a effective strategy to integrate skin color into hand detector*:
In this thesis, we did not use existing combination with skin color detection methods because of the limitation of them. Most of the existing ways to combine skin color cue with Haar-like features is that firstly do

skin color detection then apply Viola-Jones detector on segmented skin regions. This technique still meet a problem that is missed skin regions detection. In the future, we want to do a research to find a effective strategy to integrate skin color into hand detector.

iv. *Develop a method to segment whole hand region based on detected hand*: In this thesis, we use a simple method to expand detected hand center region to find a large region that contains whole hand. Finding a method to segment whole hand region based on detected hand center region is reasonable research task in the future.

(b) ***For hand posture recognition:***

i. *Design better kernel descriptor for hand posture recognition*:
In this thesis, we built hand representation based on kernel functions and kernel match functions in [5] for generic object recognition. In the future, we want to do research to find a better kernels and then build better kernel match function for hand images for example a kernel descriptor based on Haar-like features.

ii. *Develop a method to normalize rotated hand images*: In this thesis, we assume that hand is upright. However, the hand is often rotated even user try to keep it upright. To tackle this problem, we need to develop a method for hand rotation normalization.

(c) ***For applications and extensions based on our proposed methods***:

i. *Apply proposed method into a dynamic hand gestures recognition based on recognized key postures*:
In this case, hand detection step can use additional motion cue to improve performance of hand localization. We also combine hand detection and hand tracking. The proposed hand posture recognition can be used to recognize hand posture on key frames. This research is necessary because a HCI system based on gesture will exploit both static and dynamic hand gestures, even includes facial expressions and body gestures.

ii. *Exploit depth information*:
Nowadays, RGB-D sensors become more popular and cheaper. They provides depth information besides RGB information. Depth information is useful to help us improve performance of the computer vision system. For this reason, we would like to integrate our proposed method into a hand posture/gesture recognition using both color and depth informations.

iii. *Develop applications*:
The application in Chapter 5 mainly aims to illustrate that the proposed hand posture recognition system is feasible to build a real application. In the future, beside above works we want to do, we also want to build real and meaningful applications.

6.3 Publications

Book chapters:

B1. **Nguyen, V.**, Vu, H., Tran, T.: An Efficient Combination of RGB and Depth for Background Subtraction. In: Quang A Dang, Xuan Hoai Nguyen, Hoai Bac Le, Viet Ha Nguyen, and V.N.Q.B. (ed.) *Some Current Advanced Researches on Information and Computer Science in Vietnam.* Post-proceedings of The First NAFOSTED Conference on Information and Computer Science. Springer (2014).

International journals:

J1. **Nguyen, V.-T.**, Le, T.-L., Tran, T.-H., Mullot, R., Courboulay, V., Castelli, E.: Visual interpretation of hand postures for human-machine interaction. Completed manuscript, ongoing process for submission to *Pattern Analysis and Applications* journal in November, 2015.

J2. Doan, H.-G., **Nguyen, V.-T.**, Vu, H., Tran, T.-H.: A combination of User-Guide scheme and Kernel Descriptor on RGB-D data for robust and realtime hand posture recognition. Submitted to *Engineering Applications of Artificial Intelligence* journal.

National journals:

J3. Nguyen, T.-T., **Nguyen, V.-T.**, Nguyen, T.-T.-T., Le, T.-T.: A hand posture dataset (in Vietnamese). *J. Sci. Technol.* Thai Nguyen Univ. Vietnam. 99, 145-150 (2012).

International conferences and workshops:

C1. **Nguyen, V.**, Le, T., Tran, T., Mullot, R., Courboulay, V.: A method for hand detection based on Internal Haar-like features and Cascaded AdaBoost Classifier. The International Conference on Communications and Electronics (ICCE). pp. 608-613 (2012). Hue, Vietnam.

C2. **Nguyen, V.-T.**, Nguyen, T., Mullot, R., Tran, T.-T.-H., Le, H.: A Method For Hand Detection Using Internal Features And Active Boosting-Based Learning Categories and Subject Descriptors. The 4th International Symposium on Information and Communication Technology - SoICT 2013. pp. 213-221 (2013). Da Nang, Vietnam.

C3. **Nguyen, V.-T.**, Le, T.-L., Tran, T.-H., Mullot, R., Courboulay, V.: Hand Posture Recognition Using Kernel Descriptor. Procedia Comput. Sci. 39, 154-157 (2014). Evry, France.

C4. **Nguyen, V.-T.**, Le, T.-L., Tran, T.-H., Mullot, R., Courboulay, V.: A New Hand Representation Based on Kernels for Hand Posture Recognition. The 11th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2015) (2015). Ljubljana, Slovenia.

C5. **Nguyen, V.-T.**, Tran, T.-H., Le, T.-L., Mullot, R., Courboulay, V.: Using hand postures for interacting with assistant robot in library. The 1st International Workshop on Pattern Recognition for Multimedia Content Analysis (PR4MCA 2015) In conjunction with the 7th International Conference on Knowledge and System Engineering Ho Chi Minh City, Vietnam. (2015). Ho Chi Minh city, Vietnam.

C6. Pham, T.-T.-T., Le, T.-L., Dao, T.-K., **Nguyen, V.-T.**, Le, D.-H.: A robust model for person re-identification in multi-modal person localization. The Ninth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2015) (2015). Nice, France.

C7. Le, T.-L., Duong, N.-D., **Nguyen, V.-T.**, Vu, H.: Complex Background Leaf-based Plant Identification Method Based on Interactive Segmentation and Kernel Descriptor. 2nd International Workshop on Environmental Multimedia. In Conjunction with ACM Conference on Multimedia Retrieval (ICMR) 2015. (2015). Shanghai, China.

C8. Tran, T.-H., **Nguyen, V.-T.**: How good is Kernel Descriptor on Depth Motion Map for Action Recognition. 10th International Conference on Computer Vision Systems (2015). Copenhagen, Denmark.

C9. Tran, T., **Nguyen, V.**, Nguyen, V., Midy, Q.: Vision based dynamic hand gesture recognition. 2013 ICT PAMM Workshop on Mobility Assistance and Service Robotics. pp. 43-47 (2013). Kumamoto, Japan.

C10. Le, T.-L., Nguyen, V.-N., Tran, T.-T.-H., **Nguyen, V.-T.**, Nguyen, T.-T.: Temporal gesture segmentation for recognition. 2013 Int. Conf. Comput. Manag. Telecommun. 369-373 (2013). Ho Chi Minh City, Vietnam.

National conferences:

C11. **Nguyen, V.**, Vu, H., Tran, T., Le, T., Technology, C.: Noise suppression in depth map for improved background segmentation. 7th Conference on Fundamental and Applied IT Research (FAIR2014). (2014). Thai Nguyen, Vietnam.

C12. **Nguyen, V.**, Vu, H., Tran, T.: Background Subtraction with KINECT data : An Efficient Combination RGB and Depth. The first NAFOSTED Conference on Information and Computer Science (2014). Hanoi, Vietnam.

The contributions and results of this thesis have published in one book chapter, one national journal, and twelve conference papers including the IEEE conference on Automatic Face and Gesture Recognition that is the premier international forum for research in image and video-based face, gesture, and body movement recognition. We have completed two other journal paper manuscripts, and the publishing processes are ongoing.

The main results on hand detection are presented in papers *#C1* and *#C2*. While, papers *#C3* presents our investigation results of KDES on hand posture recognition problem, and paper *#C4* presents our proposed hand representation method based on KDES. A hand posture dataset is reported in paper *#J3*. The deployment of an application example and its results is presented in paper *#C5*. Manuscript *#J1* presents fully and more detail our contributions in hand detection, hand posture recognition, and application.

We remark that our improved KDES will make other object recognition better than original KDES. According to this remark, we applied the improved KDES that we proposed for hand posture recognition into some other problems, and we obtained good results. These results were presented in published papers *#C6 - #C8*.

In this thesis, it is important to study about dynamic hand gesture recognition, human action recognition because these problems are closely related to our problem, static hand gesture recognition. In terms of sensor, the understanding about RGB-D sensor helps us know the role of color information in a RGB-D system as well as define our motivation of research on color information. Upon the above reasons, we did some related studies and prented results in published papers *#C9 - #C12* and manuscript *#J2*. Manuscript *#J2* also presents the results of our improved KDES on Kinect sensor.

6.4 Awards

1. **Travel Award** to *The 3rd IAPR Asian Conference on Pattern Recognition (ACPR2015)* **Doctoral Consortium**, November 3-6, 2015, Kuala Lumpur, Malaysia.

2. **Best Paper Award** "Using hand postures for interacting with assistant robot in library," *The 1st International Workshop on Pattern Recognition for Multimedia Content Analysis (PR4MCA 2015)* In conjunction with *the 7th International Conference on Knowledge and System Engineering*, 2015, Ho Chi Minh city, Vietnam.

3. **Best Paper Award** "Vision based dynamic hand gesture recognition," *2013 ICT PAMM Workshop on Mobility Assistance and Service Robotics,* 2013, Kumamoto, Japan.

Bibliography

[1] Immersion corporation, cyberglove ii datasheet, [online], available: http://www.immersion.com/3d/docs/cybergloveii

[2] Andre LC Barczak and Farhad Dadgostar. Real-time hand tracking using a set of cooperative classifiers based on Haar-like features. *Research Letters in the Information and Mathematical Sciences*, 7:29–42, 2005.

[3] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine . . .* , 2002.

[4] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[5] Liefeng Bo, X Ren, and Dieter Fox. Kernel Descriptors for Visual Recognition. *NIPS*, pages 1–9, 2010.

[6] Liefeng Bo and Cristian Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. *NIPS*, pages 1–9, 2009.

[7] Nabil Boughnim and Julien Marot. Hand posture recognition using jointly optical flow and dimensionality reduction. *EURASIP Journal on Advances in Signal Processing*, pages 1–22, 2013.

[8] L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 423–428. Ieee, 2002.

[9] Frédérick Caron and Mathieu Gascon-Lefebvre. Projet de session: Reconnaissance de gestes de la main en temps réel. Technical report, École de technologie supérieure (ÉTS), the Université du Québec., 2013.

[10] Chin-Chen Chang, Jiann-Jone Chen, Wen-Kai Tai, Chin-Chuan Han, and Others. New approach for static gesture recognition. *Journal of information science and engineering*, 22(5):1047–1057, 2006.

[11] Ankit Chaudhary, Jagdish Lal Raheja, Karen Das, and Sonia Raheja. Intelligent Approaches to interact with Machines using Hand Gesture Recognition in Natural way: A Survey. *IJCSES*, 2(1):122–133, February 2011.

[12] Qing Chen, Nicolas D. Georganas, and Emil M. Petriu. Real-time Vision-based Hand Gesture Recognition Using Haar-like Features. *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007*, pages 1–6, May 2007.

[13] Junyeong Choi and Byung-kuk Seo. Robust Hand Detection for Augmented Reality Interface. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, volume 1, pages 319–322, 2009.

[14] Yuelong Chuang, Ling Chen, and Gencai Chen. Hierarchical bag-of-features for hand posture recognition. In *2011 18th IEEE International Conference on Image Processing (ICIP)*, pages 1777—-1780, 2011.

[15] Yuelong Chuang, Ling Chen, and Gencai Chen. Saliency-guided improvement for hand posture detection and recognition. *Neurocomputing*, 133:404–415, June 2014.

[16] Andrea Corradini. Real-Time Gesture Recognition by Means of Hybrid Recognizers. *Lecture Notes in Computer Science, Gesture and Sign Language in Human-Computer Interaction*, 2298:34–47, 2002.

[17] R L Cosgriff. Identification of shape. *Ohio State Univ. Res. Foundation, Columbus, OH, Tech. Rep. ASTIA AD*, 254:792, 1960.

[18] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 886–893. Ieee, 2005.

[19] Nasser H. Dardas and Nicolas D. Georganas. Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques. *IEEE Transactions on Instrumentation and Measurement*, 60(11):3592–3607, November 2011.

[20] Liya Ding and Aleix M Martinez. Modelling and Recognition of the Linguistic Components in American Sign Language. *Image and vision computing*, 27(12):1826–1844, November 2009.

[21] Shaoyi Du, Nanning Zheng, Qubo You, Yang Wu, Maojun Yuan, and Jingjun Wu. Rotated Haar-Like Features for Face Detection with In-Plane Rotation. *Lecture Notes in Computer Science*, 4270/2006:128–137, 2006.

[22] Le Dung and M Mizukawa. Fast Fingertips Positioning Based on Distance-based Feature Pixels. In *Third International Conference on Communications and Electronics (ICCE)*, pages 184–189, 2010.

[23] Artur Ferreira. Survey on boosting algorithms for supervised and semi-supervised learning. *Institute of Telecommunications*, 2007.

[24] Hardy Francke, Javier Ruiz-del solar, and Rodrigo Verschae. Real-Time Hand Gesture Detection and Recognition Using Boosted Classifiers and Active Learning. *Lecture Notes in Computer Science*, 4872:533–547, 2007.

[25] William T Freeman and Michal Roth. Orientation Histograms for Hand Gesture Recognition. In *International workshop on automatic face and gesture recognition*, pages 296—-301, 1995.

[26] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[27] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[28] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[29] Pragati Garg, Naveen Aggarwal, and Sanjeev Sofat. Vision Based Hand Gesture Recognition. *World Academy of Science, Engineering and Technology*, 49:972–977, 2009.

[30] Dariu M Gavrila and Vasanth Philomin. Real-time object detection for "smart" vehicles. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 87–93. IEEE, 1999.

[31] Helmut Grabner, Thuy Thi Nguyen, Barbara Gruber, and Horst Bischof. On-line boosting-based car detection from aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(3):382–396, 2008.

[32] Shikha Gupta, Jafreezal Jaafar, and Wan Fatimah Wan Ahmad. Static Hand Gesture Recognition Using Local Gabor Filter. *Procedia Engineering*, 41(Iris):827–832, January 2012.

[33] Haitham Hasan and S. Abdul-Kareem. Static hand gesture recognition using neural networks. *Artificial Intelligence Review*, pages 147–181, January 2012.

[34] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[35] Chung-Lin Huang and Wen-Yi Huang. Sign language recognition using model-based tracking and a 3D Hopfield neural network. *Machine Vision and Applications*, 10(5-6):292–307, 1998.

[36] A. Just, Y. Rodriguez, and S. Marcel. Hand Posture Classification and Recognition using the Modified Census Transform. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 351–356. Ieee, 2006.

[37] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40(3):1106–1122, March 2007.

[38] Benoit Kaufmann, Jean Louchet, and Evelyne Lutton. Hand Posture Recognition Using Real-Time Artificial Evolution. In *Applications of Evolutionary Computation*, volume 6024, pages 251–260. Springer Berlin Heidelberg, 2010.

[39] Daniel Kelly, John McDonald, and Charles Markham. A person independent system for recognition of hand postures used in sign language. *Pattern Recognition Letters*, 31(11):1359–1368, August 2010.

[40] Alireza Khotanzad and Yaw Hua Hong. Rotation invariant image recognition using features selected via a systematic method. *Pattern recognition*, 23(10):1089–1101, 1990.

[41] Roelof Koekoek and Rene F Swarttouw. The Askey-scheme of hypergeometric orthogonal polynomials and its q-analogue. *arXiv preprint math/9602214*, 1996.

[42] Mathias Kölsch and Matthew Turk. Robust Hand Detection. In *The Sixth IEEE international conference on Automatic face and gesture recognition*, pages 614–619, 2004.

[43] P. Pramod Kumar, Prahlad Vadakkepat, and Ai Poh Loh. Hand Posture and Face Recognition Using a Fuzzy-Rough Approach. *International Journal of Humanoid Robotics*, 07(03):331–356, September 2010.

[44] Takeshi Kurata, Takashi Okuma, Masakatsu Kourogi, and Katsuhiko Sakaue. The hand mouse: Gmm hand-color classification and mean shift tracking. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on*, pages 119–124. IEEE, 2001.

[45] Daeho Lee and Seung Gwan Lee. Vision-Based Finger Action Recognition by Angle Detection and Contour Analysis. *ETRI Journal*, 33(3):415–422, June 2011.

[46] Yu-Ting Li and Juan P. Wachs. Recognizing hand gestures using the weighted elastic graph matching (WEGM) method. *Image and Vision Computing*, 31(9):649–657, September 2013.

[47] Yu-Ting Li and Juan P Wachs. HEGM: A hierarchical elastic graph matching for hand gesture recognition. *PR*, (765), 2014.

[48] Rung-Huei Liang and Ming Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 558–567. IEEE, 1998.

[49] Licsár Attila and Tamás Szirányi. User-adaptive hand gesture recognition system with interactive training. *Image and Vision Computing*, 23:1102–1114, 2005.

[50] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Proceedings. International Conference on Image Processing*, pages 900–903. Ieee, 2002.

[51] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. *Lecture Notes in Computer Science*, 2781/2003:297–304, 2003.

[52] Liwei Liu, Junliang Xing, Haizhou Ai, and Xiang Ruan. Hand Posture Recognition Using Finger Geometric Feature. Number Icpr, pages 565–568, 2012.

[53] Raymond Lockton and AW Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *BMVC*, pages 817–826, 2002.

[54] David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, volume 2, pages 1150–1157 vol.2, 1999.

[55] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[56] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Efficient classification for additive kernel SVMs. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):66–77, January 2013.

[57] S Marcel and Olivier Bernier. Hand posture recognition in a body-face centered space. *Gesture-Based Communication in Human-Computer Interaction*, pages 97–100, 1999.

[58] Sébastien Marcel. Hand posture recognition in a body-face centered space. In *CHI EA '99 CHI '99 extended abstracts on Human factors in computing systems*, pages 302–303, 1999.

[59] K Mathias and Matthew Turk. Analysis of Rotational Robustness of Hand Detection with a Viola-Jones Detector. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, pages 07 – 110, 2004.

[60] Kevin McGuinness and Noel E. O Connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, February 2010.

[61] Kuizhi Mei, Lu Xu, Boliang Li, Bin Lin, and Fang Wang. A Real-time Hand Detection System Based on Multi-feature. *Neurocomputing*, 158:184–193, February 2015.

[62] Chris Messom and Andre Barczak. Fast and Efficient Rotated Haar-like Features using Rotated Integral Images. In *Australian Conference on Robotics and Automation (ACRA2006)*, pages 1–6, 2006.

[63] Mokhtar M.Hasan and Pramod K. Mishra. HSV Brightness Factor Matching for Gesture Recognition System. *International Journal of Image Processing (IJIP)*, 4(5):456–467, 2010.

[64] T. Mita, T. Kaneko, and O. Hori. Joint Haar-like features for face detection. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 2:1619–1626, 2005.

[65] Sushmita Mitra and Tinku Acharya. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, May 2007.

[66] Arpit Mittal, Andrew Zisserman, and Philip Torr. Hand detection using multiple proposals. In *Procedings of the British Machine Vision Conference 2011*, pages 75.1–75.11. British Machine Vision Association, 2011.

[67] Van-Toi Nguyen, Thi-Lan Le, Thanh-Hai Tran, RÃ©my Mullot, and Vincent Courboulay. Hand posture recognition using kernel descriptor. *Procedia Computer Science*, 39(0):154 – 157, 2014. The 6th international conference on Intelligent Human Computer Interaction, {IHCI} 2014.

[68] Timo Ojala, Matti Pietikainen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[69] David L. Olson and Dursun Delen. *Advanced Data Mining Techniques*. Springer, 1st editio edition, 2008.

[70] Eng-Jon Ong and Richard Bowden. A Boosted Classifier Tree for Hand Shape Detection. In *Face and Gesture Recognition*, 2004.

[71] S. Padam Priyal and Prabin Kumar Bora. A robust static hand gesture recognition system using geometry based normalizations and Krawtchouk moments. *Pattern Recognition*, 46(8):2202–2219, August 2013.

[72] Vladimir I Pavlovic, Rajeev Sharma, and Thomas S Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *TPAMI*, 19(7):677–695, July 1997.

[73] Minh-tri Pham, Yang Gao, Viet-Dung D.Hoang, and Tat-jen Cham. Fast Polygonal Integration and Its Application in Extending Haar-like Features to Improve Object Detection. In *International Conference on Computer Vision (CVPR) 2010*, pages 942 – 949, 2010.

[74] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[75] Pramod Kumar Pisharady, Prahlad Vadakkepat, and Ai Poh Loh. Attention Based Detection and Recognition of Hand Postures Against Complex Backgrounds. *International Journal of Computer Vision*, August 2012.

[76] Siddharth S. Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, November 2012.

[77] J Ravikiran, Kavi Mahesh, Suhas Mahishi, R Dheeraj, S Sudheender, and Nitin V Pujari. Finger Detection for Sign Language Recognition. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume I, pages 489–493, 2009.

[78] S. Mohamed M. Roomi, R. Jyothi Priya, and H. Jayalakshmi. Hand Gesture Recognition for Human-Computer Interaction. *Journal of Computer Science*, 6(9):1002–1007, 2010.

[79] Yong Rui, Alfred C She, and Thomas S Huang. A modified Fourier descriptor for shape matching in MARS. *SERIES ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING*, 8:165–180, 1997.

[80] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[81] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):411–426, 2007.

[82] Kyriakos Sgouropoulos, Ekaterini Stergiopoulou, and Nikos Papamarkos. A Dynamic Gesture and Posture Recognition System. *Journal of Intelligent & Robotic Systems*, 76(2):283—-296, October 2013.

[83] Ekaterini Stergiopoulou and Nikos Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *EAAI*, 22(8):1141–1158, 2009.

[84] Ekaterini Stergiopoulou and Kyriakos Sgouropoulos. Real time hand detection in a complex background. *Engineering Applications of Artificial Intelligence*, 35:54–70, 2014.

[85] Helman Stern, Juan P Wachs, Yael Edan, and Others. Human factors for design of hand gesture Human-Machine Interaction. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 5, pages 4052–4056. IEEE, 2006.

[86] Nguyen Thanh-Mai, Viet-Son Nguyen, and Thanh-Hai Tran. Designing a hand gesture vocabulary for human-robot interaction applications. *J. Sci. Technol. Tech. Univ.*, (83B/2011):pp. 22—-29, 2011.

[87] Antonio Torralba, Kevin P Murphy, and William T Freeman. Sharing features : efficient boosting procedures for multiclass object detection. In *Computer Vision and*

Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, pages II—-762, 2004.

[88] Kentaro Toyama and Andrew Blake. Probabilistic tracking in a metric space. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 50–57. IEEE, 2001.

[89] Thi-thanh-hai Tran. Face Recognition from Video under Uncontrolled Lighting Condition. In *12th IEEE International Symposium on Signal Processing and Information Technology*, 2012.

[90] Thi-Thanh-Hai Tran and Thi-Thanh-Mai Nguyen. Invariant Lighting Hand Posture Classification. In *IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 827 – 831, 2010.

[91] J.; Triesch and C. von der Malsburg. Robust classification of hand postures against complex backgrounds. In *The Second International Conference on Automatic Face and Gesture Recognition, 1996.*, pages 170 – 175, 1996.

[92] J.; Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1449 – 1453, 2001.

[93] Jochen Triesch and Christoph Von Der Malsburg. A gesture interface for human-robot-interaction. In *10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 546–546, 1998.

[94] Claudio Uras and Alessandro Verri. Sign Language Recognition: an Application of the Theory of Size Functions. In *BMVC*, pages 1–10, 1995.

[95] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):480–92, March 2012.

[96] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, pages 511–518. IEEE Comput. Soc, 2001.

[97] Juan Wachs, Helman Stern, Yael Edan, Michael Gillam, Craig Feied, Mark Smith, and Irving Street. A Real-Time Hand Gesture System based on Evolutionary Search. In *GECCO2005: Tenth Genetic and Evolutionary Computation Conference,Washington, DC, USA.*, volume 1, 2005.

[98] Chieh-chih Wang and Ko-chih Wang. Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction. *Recent progress in robotics: viable robotic service to human*, pages 317—-329, 2008.

[99] Ying Wu and Thomas S. Huang. View-independent recognition of hand postures. In *Computer Vision and Pattern Recognition*, pages 88–94, 2000.

[100] Ying Wu, Qiong Liu, and Thomas S. Huang. An adaptive self-organizing color segmentation algorithm with application to robust real-time human hand localization. In *Asian Conference on Computer Vision*, pages 1106–1111, 2000.

[101] X Yin and M Xie. Finger identification and hand posture recognition for human-robot interaction. *Image and Vision Computing*, 25(8):1291–1300, August 2007.

[102] Gabriel Yoder and Lijun Yin. Real-Time Hand Detection and Gesture Tracking with GMM and Model Adaptation. *Lecture Notes in Computer Science*, 5876:387–396, 2009.

[103] Xiaojin Zhu, Jie Yang, and AlexWaibel. Segmenting hands of arbitrary color. *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 446–453, 2000.