

Real-Time Hand Gesture Recognition in Complex Environments

by

Milyn C. Moy

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Engineering in Electrical Engineering and Computer Science

and

Bachelor of Science in Computer Science and Engineering

at the

Massachusetts Institute of Technology

May 1998

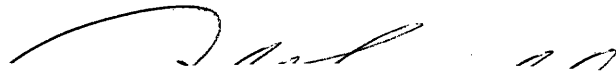
CS June 1998

© Milyn C. Moy, MCMXCVIII. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part, and to grant others the right to do so.

Author
Department of Electrical Engineering and Computer Science
May 22, 1998

Certified by
Rodney Brooks
Professor, Department of Electrical Engineering and Computer Science
Thesis Supervisor



Accepted by
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUL 14 1998

Eng

LIBRARIES
Eng



Real-Time Hand Gesture Recognition in Complex Environments

by

Milyn C. Moy

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 1998, in partial fulfillment of the
requirements for the degrees of
Master of Engineering in Electrical Engineering and Computer Science
and
Bachelor of Science in Computer Science and Engineering

Abstract

As robots become more integrated into humans' everyday lives, it becomes essential for them to perceive and understand people's intentions and actions. People will want a more natural way to interact with robots, closer to the way humans do it among themselves. When humans communicate, they use a rich set of gestures and body language in addition to speech, which significantly enhances their ability to get thoughts, feelings, and intentions across. Hence, robots strongly need the ability to understand human gestures and to interpret body language.

One of the most expressive components of body language are hand gestures, which is why they are so important at the interface between humans and robots. This thesis presents an extensible, vision-based communication system that is able to interpret 2D dynamic hand gestures.

The system consists of three primary modules: hand segmentation, feature extraction, and gesture recognition. Hand segmentation uses both motion and skin-color detection to separate the hand from cluttered backgrounds. The feature extraction module gathers hand trajectory information and encapsulates it into a form that is used by the gesture recognizer. This last module identifies the gesture and translates it into a command that the robot can understand.

Unlike other existing work, this hand gesture recognition system operates in real-time without the use of special hardware. Results demonstrate that the system is robust in realistic, complex environments. Moreover, this system does not require a priori training and allows the hand great flexibility of motion without the use of gloves or special markers.

Thesis Supervisor: Rodney Brooks

Title: Professor, Department of Electrical Engineering and Computer Science

Acknowledgments

This work was performed at the MIT Artificial Intelligence Laboratory. Support for this work was provided by Yamaha Corporation under contract No. 65539.

First and foremost, I would like to thank my parents, sisters, and grandmother for all the support and love they have always given me. Their strong spirit and confidence in me have been a great inspiration to me.

I would also like to thank Rod for giving me the opportunity to work in a field I used to only dream of. His endless support, and guidance have made my past two years much more memorable.

Special thanks to Scaz for all the helpful discussions and suggestions he has offered me and for doing careful revisions of my thesis. My thanks to all the members of the Yuppy and Cog group, specially Junji, and Charlie for all the useful ideas they have provided to my work and for making my experience at the AI lab much more enjoyable. Also, thanks to Chandana and all the people who helped me test this work.

And finally to George, for making me very happy and always being there for me.

Contents

1	Introduction	11
1.1	Motivation	11
1.2	Scope of Investigation	12
1.3	Thesis Structure	13
2	Related Work	15
2.1	Hand Detection Systems	15
2.1.1	Constraints on the Hand	15
2.1.2	Constraints on the Environment	16
2.1.3	More Flexible Approaches	17
2.2	Gesture Recognition Systems	17
2.2.1	Instrumented Gloves	17
2.2.2	Model-based Approach	18
2.2.3	Feature-based Approach	19
3	Hand Gesture Recognition System	21
3.1	Development Platform	21
3.2	Goals	24
3.3	Assumptions	24
3.4	Gesture Lexicon	25
3.5	System Overview	27
3.6	Hand Segmentation	28
3.6.1	Motion Detection	29

3.6.2	Color Detection	31
3.6.3	Combining Motion and Skin-color	36
3.7	Feature Extraction	40
3.8	Gesture Recognition	43
3.8.1	Gesture Matching	45
3.9	True Extensibility	46
3.10	Meeting the Goals	47
4	Experimental Results	49
4.1	Performance	49
4.2	Robustness	57
4.3	Yuppy’s Navigation	58
5	Conclusions	61
5.1	Summary	61
5.2	Future research	62
A	User Manual	69

List of Figures

3-1	Yuppy: an emotional pet robot.	22
3-2	Picture of the external camera.	23
3-3	Basic gestures.	26
3-4	The system's high-level architecture.	27
3-5	Flow diagram for the hand segmentation module.	29
3-6	Image differencing results	31
3-7	The HLS color model	34
3-8	Skin-color detection	35
3-9	Hand segmentation	39
3-10	Example of feature extraction.	42
3-11	Possible orientations of the feature vector in the $dx-dy$ plane.	44
3-12	New gesture classes built from the combination of primitives.	46
4-1	Example of testing environment	50
4-2	Gestures accuracy	51
4-3	Expected vs. obtained accuracy for composite gestures	55
4-4	Robustness to two different skin-colored people: Caucasian (left) and Indian (right)	58

Chapter 1

Introduction

1.1 Motivation

With robots rapidly becoming part of humans' everyday lives, it will become essential to have robots that are able to grasp human intentions. More than just agents, or intelligent computer interfaces, robots are embodied, and are capable of cohabiting with human beings. Thus, it is natural for humans to want to communicate with robots using the same means they would to communicate with other humans beings.

Still the most practical solutions so far for humans to communicate and interact with robots require humans to use an input device that can transfer the necessary information, such as keyboard, mouse, or more specialized devices like trackers, 3D-mice and datagloves. Although using these input devices can contribute to human-robot interaction in the present, a more long term solution is needed. A natural, contactless interface should be devised to avoid the need to wear external devices.

When humans communicate with each other, they use, in addition to language, also gestures, facial expressions, and poses. Children can learn to communicate with gestures before they can even talk [Acredolo and Goodwyn 96]. Adults often use gesture as a supplement or substitute for other forms of communication. Clearly, one way for robots to perceive human's intentions and communicate with people more effectively is for them to be able to interpret human gestures and body language.

1.2 Scope of Investigation

One of the most expressive components of body language is hand gesture. Humans use their hands express their ideas and feelings, to grasp and explore objects, and to act on the world. Thus, hand gestures provide a very natural interface for humans to communicate with robots.

Gesture recognition is a very complex and wide area of research. The general gesture interpretation needs a broad range of contextual information, general knowledge, cultural background and linguistic capabilities, which is beyond the capabilities of this work. Recognition can be performed on static or dynamic gestures. Static gestures are those that do not involve motion (i.e. hand posture). Dynamic gestures, on the other hand, do involve motion. Moreover, two-dimensional (2D) as well as three-dimensional (3D) gestures can be studied.

This thesis will focus on the visual interpretation of 2D dynamic hand gestures which can be performed in real-time and in complex environments. The goal of this work will be to enable humans to communicate and interact with a pet robot, which is capable of exhibiting animal-like behaviors, including emotional ones, as it interacts with the people and objects in the environment surrounding it. The platform that was used for experimentation with this gesture recognition system was Yuppy, a pet robot built at the MIT Artificial Intelligence Laboratory.

This work is intended to provide a step toward a more natural human-robot interaction. Thus, its scope is not to create a full lexicon system that will facilitate this interaction in every possible circumstance. Rather, the system should be extensible toward this goal. This hand gesture recognition system provides a mechanism to extend the gesture database by enabling the recognition of new gestures.

An alternative approach to gesture recognition is speech recognition which represents a similarly demanding research objective. Speech has been widely researched in the past, and very successful results have been produced with the use of Hidden Markov Models[Huang, Ariki, and Jack 90]. However, this research focuses on the study of vision for two main reasons. First, most of human's perceptions of the ex-

ternal world are done through vision. Second, gesture recognition is a much younger area of study compared to speech recognition and it has many more potential applications in areas such as human body motion tracking and analysis, surveillance, video telephony, and non-invasive interfacing with virtual reality applications.

Although hand gesture recognition is the focus of this research, speech recognition and analysis of facial expressions can be combined for multi-modal interaction, contributing to a better communication between humans and pet robots.

1.3 Thesis Structure

The remainder of this thesis will be structured in 5 chapters. Chapter 2 will give an overview of different approaches to both hand segmentation and gesture recognition. Chapter 3 will present the goals and assumptions of this system. In this chapter, the design and the implementation of the system will be described. Chapter 4 will provide an evaluation of the overall performance of the system and a discussion of some weaknesses as well as robustness issues. Finally, Chapter 5 will conclude with a summary of this system and some suggestions for future work.

Chapter 2

Related Work

This chapter will review existing related work. Different approaches for hand detection will be presented first, followed by a description of the most common methods used for gesture recognition.

2.1 Hand Detection Systems

Hand gesture recognition requires the ability to accurately segment the hand from the background. Due to the difficulty of this task, many researchers have found alternate ways to avoid detecting the hand by requiring the user to use a special marker, usually a glove, which is chosen to be very distinct from other objects in the environment. More recently, several glove-free methods have been used, some of which impose constraints on the environment whereas some other ones are more flexible in this respect. These different approaches will be described as follows.

2.1.1 Constraints on the Hand

Many gesture recognition systems have been built using instrumented gloves. The most successful glove was developed by Zimmerman and Lanier [1987]. It is called the VPL DataGlove and it consists of optical fiber sensors placed along the back of the fingers. These sensors are used for posture recognition by detecting how much the

fingers are bent.

Other researchers have controlled the imaging conditions with the use of special colored gloves that are very distinct from objects in the background. Dorner [1993] uses gloves with colored markers to interpret American Sign Language. Starner et al. [1998] built an American Sign Language recognizer using a pink glove for the right hand and a blue glove for the left to simplify the hand segmentation problem. Hienz et al. [1996] developed a video-based hand-arm motion analysis system for German sign language recognition. To facilitate the segmentation of the different body parts, they used a color coded glove and colored markers placed at the elbow and shoulder. These systems are not flexible enough for most real world applications, since the use of gloves is both unnatural and cumbersome.

2.1.2 Constraints on the Environment

Several hand detection systems have been built using a controlled environment. Special backgrounds have been used [Darrell and Pentland 93; Rehg and Kanade 94] where the hand is the only object in the image with a very simple background such as a solid color one.

Others have detected the hand in an arbitrary but static background. To achieve this goal, skin color information as well as a technique called background subtraction is used. This technique, frequently addressed as background separation, identifies the moving object or person in front of the static background by taking a thresholded difference between an image of the background and an image of the object or person in front of the same background at a later time.

Usually, the background statistics are updated continuously to accommodate for changes in illumination, camera signals, and location of moving objects or persons. Darrell et al. [1994] developed an environment that allows a person to interact with autonomous agents in a simulated graphical world, using background subtraction to segment the person from the background. Appenzeller [1997] used this technique to track the 3D motion of people in an intelligent room. Further, by combining this technique with skin-color information, the hands and faces of the people in the room

were identified. Dang [1996] also used background subtraction to detect the different hand postures for the recognition of static gestures.

The obvious drawback of this approach is that the detection of the hand is constrained to a static background, given that an image of the background without the person needs to be taken apriori. The platform for this work is a pet robot that moves around while interacting with people and objects around it. Thus, a static background is too restrictive for this application.

2.1.3 More Flexible Approaches

A few hand segmentation methods have been proposed that do not require the user to wear a marker and operate in uncontrolled environments. Triesch and von der Malsburg [1998] used motion and skin color detection to segment the human hand as it moves toward a stable posture. The classification of the posture was then executed using elastic graph matching. These postures were used for giving simple commands to a robot in a gesture interface system. Limitations exist in the use of motion cues given that they are prone to also detecting the shadows of moving objects. They also appear in the use of color cues given that similarly colored objects will be detected. However, by using these two cues together, the false-positives are reduced considerably; thus, making this a more flexible approach for hand segmentation. This work uses this approach for segmenting the hand from a complex background.

2.2 Gesture Recognition Systems

Recently, there has been a significant amount of research on hand gesture recognition. A description of the most common approaches is offered below.

2.2.1 Instrumented Gloves

With the use of instrumented gloves, many gesture recognition systems have been developed without vision processing. Grimes [1993] developed a Digital Data Entry

Glove to recognize the signed alphabet. A combination of sensor values indicating a particular hand posture, joint flexion or fingertip contact produced a letter of the alphabet. Baudel and Beaudouin-Lafon [1993] used a VPL DataGlove as a remote control for a computer-aided presentation. The great benefits of using instrumented gloves are reliability and accuracy in posture recognition. However, they are unnatural and troublesome to use, due to the wiring that tethers the user to stationary equipment.¹

2.2.2 Model-based Approach

Model-based approaches assume a physically valid model of the hand and attempt to recognize gesture as a variation of the hand articulation [Heap and Hogg 96; Triesch and von der Malsburg 96; Rehg and Kanade 94]. There are several methods that use this approach, such as template matching and neural networks.

Template matching is based on a function, often called the decider, which takes the raw data obtained from the sensor and computes its similarity with different templates of values. There usually is a similarity threshold. If the input is above this threshold, it is classified as a member of its closest template, otherwise it is discarded.

Many template-based systems have been developed, generally for isolated postures. Freeman and Weissman [1995] used normalized correlation between a fixed template and a particular pose of the hand for gesture recognition. An open hand indicates the start of a gesture, whereas a closed hand signals its termination. The motion of the hand determined the user's gesture. Triesch and von der Malsburg [1996] classified ten different hand postures in grey-level images using elastic graph matching. Although robustness to complex backgrounds was achieved, this system is not flexible to large variations in the shape of the hands referring to the same posture. Heap and Hogg [1996] used a 3D deformable Point Distribution Model of the human hand to track different hand postures in real time, but faced occlusion, scale, and rotation problems. Rehg and Kanade [1994] developed DigitEyes, a system that uses

¹In 1997, Wireless DataGloves were introduced in the market by General Reality Company.

kinematics and geometric hand models to recover the 27 degree-of-freedom configuration of the hand.

Neural networks is another common method used for recognizing both static and dynamic gestures. The biggest advantage in using this method is generalization. Recognition is feasible in presence of noisy or incomplete templates via learning. Nowlan and Platt [1992] built a system to track an open or closed hand in a sequence of video frames. For segmentation, they used a neural network to learn the hand template.

Model-based approaches generally suffer from being computationally expensive. The computational complexity is due to the need for high resolution images in order to ensure accurate segmentation, and the need to cope with variations in intensity, scale, orientation (translation and rotation), and deformation. Further, the use of a neural network also has serious efficiency drawbacks. First, many, often thousands of samples need to be used to train the network, requiring a lot of processing power. Second, if too many examples are used in the training phase, the neural net can over learn and discard the previously learned patterns. Finally, if a new gesture is added to the lexicon, the training has to be restarted. Thus, model-based approaches are not very attractive for real-time applications.

2.2.3 Feature-based Approach

Feature-based approaches do not assume a 3-D model of the hand and use instead low-level image features for gesture recognition. Hidden Markov Models (HMM) have been successfully used in the speech domain [Huang et al. 90] and more recently they have been used for gesture recognition systems. Yamato et al. [1992] proposed a Hidden Markov Model based approach for human action recognition from a set of time-sequenced images. This system obtained higher than 90% recognition rate in tennis actions performed by three subjects. Campbell et al. [1996] used 3D data extracted from stereo images and a HMM for the recognition of 18 T'ai Chi gestures. Starner et al. [1998] worked on a wearable computer that can recognize sentence-level American Sign Language using a camera mounted on a baseball cap for input.

Hidden Markov Models have the benefits of being time-scale invariant and offering learning capability. However, a major disadvantage of using this method is that the training phase takes a long time, given that to achieve user independency, a suitable training database must include data from many subjects. Further, when the number of training patterns is small, the recognition rate becomes unstable, and when test pattern subjects and training pattern subjects are different, recognition also suffers; thus, appropriate training patterns should be carefully selected.

Darrell and Pentland [1993] proposed a view-based approach, where gestures were recognized by comparing the correlation pattern of the model image sequence with an input image sequence using dynamic time warping. Although real-time performance and high-accuracy was achieved, this method required user-dependent training of model views, high-resolution images of the hand, and special purpose correlation hardware.

Other feature-based methods include the work of Nagaya et al. [1996] who used a vector representation of the gesture's trajectory as a feature for real-time gesture recognition from motion images. Different background from the fixed one they used, would cause the assumptions of this method to fail. Also, Bobick and Wilson [1995] used a state-transition diagram built from trajectory of a sequence of images to recognize gestures.

These feature-based methods may be suitable for real time gesture recognition because low-level image features can be obtained quickly. However, these methods require an effective segmentation method of the hand from the input images.

Chapter 3

Hand Gesture Recognition System

This chapter will present a new approach to gesture recognition which performs in real-time and is robust to cluttered environments. As opposed to the previous work described in Chapter 2, this gesture recognition system does not require the user to wear a marker and provides the hand with a flexibility of motion that template-based systems do not allow.

The structure of this chapter is as follows: First, a description of the testing and development platform will be provided. Second, the goals that this system attempts to achieve and the assumptions taken to simplify the problem will be stated. Third, the gesture lexicon will be presented. Finally, an overview of the high-level architecture of the system will be described, followed by a detailed discussion of the design and some implementation issues of its main modules.

3.1 Development Platform

The testbed of this system was an emotional pet robot called Yuppy (see figure 3-1). This robot is currently being developed at the MIT Artificial Intelligence Laboratory. In this section, the design of Yuppy as well as the development platform will be described.

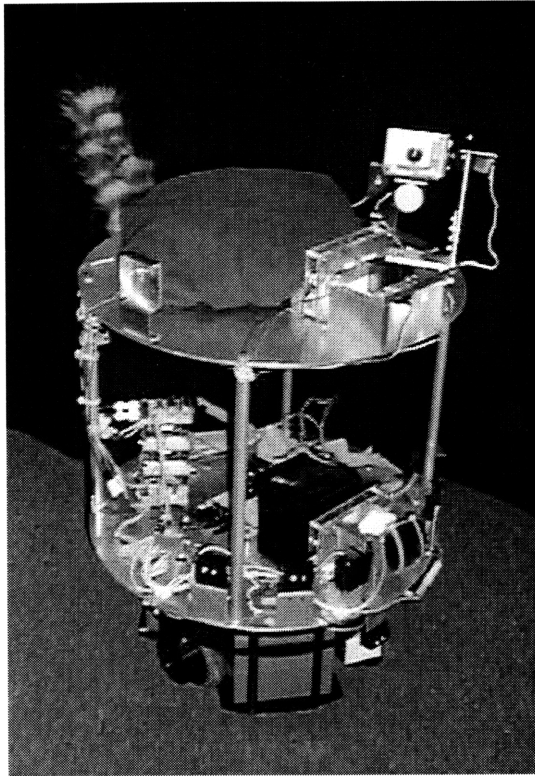


Figure 3-1: Yuppy: an emotional pet robot.

Yuppy is the result of a continuous effort to build a pet robot that will exhibit emotional behaviors and interact with humans in interesting ways. This robot is built on top of a 12 inch synchrodrive base (a B12 built by Real World Interface, Inc.). To increase its awareness of the environment and allow for better interaction with humans, Yuppy is equipped with various sensors which include:

- a lower camera (Chinon CX-062) with a 4mm wide angle lens and one degree of freedom for tilt (provided by a Futaba S148 servo) used for navigation,
- an upper camera (Chinon CX-062) with a 4mm wide angle lens and two degrees of freedom for pan and tilt (provided by a Futaba S148 servo) used for both navigation and interaction with people,
- five one bit IR proximity sensors located in the front and sides of the robot's body,
- a pressure sensor that simulates a patch of touch sensitive skin,

- a pyro-electric sensor mounted parallel to the upper camera, for detecting the presence of people around it,
- a pair of microphones for simple sound localization.

Yuppy uses the motion of its tail, body and cameras to express emotional behaviors. It also uses a breathing simulation mechanism to convey satisfaction.

This robot is meant to be fully autonomous. The sensors and actuators have been integrated using a simple network of 8 bit Motorola 6811 processors programmed in machine code. This low level processing is done on board Yuppy. However, to ease the development task, all other perception and control code temporarily run offboard. All the software is written in C and runs on a standard 300 MHz Pentium Pro workstation.

For development purposes, an external Chinon CX-062 color microcamera was used (see figure 3-2), which is of the same type as the one installed on Yuppy. This camera chosen mainly for price and availability reasons consists of a circuit board containing the photosensitive circuitry and lens attached to a board used to maintain power and produce standard NTSC video output. Similarly, as in Yuppy, the camera's motion is provided through the use of a Futaba S148 low-profile precision servo which allows two degrees of freedom for pan and tilt.

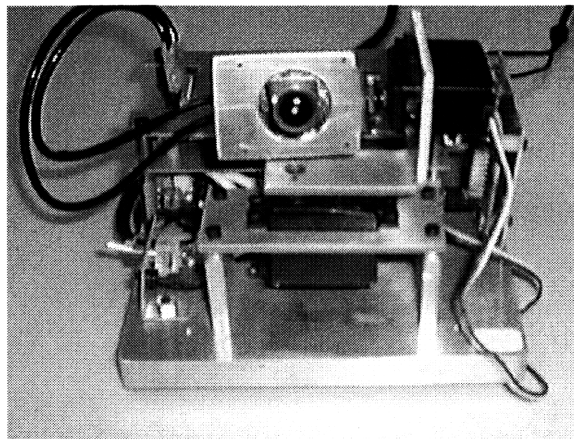


Figure 3-2: Picture of the external camera.

The camera signals produced by the Chinon color cameras are standard NTSC analog signals. To convert them to a digital output, a Matrox Meteor color PCI frame grabber was used. This low-cost frame grabber captures NTSC (640x480) and standard RGB. The software used with this frame grabber is the Matrox Imaging Library (MIL), a high-level C programming library for image acquisition, processing, and display, pattern matching, blob analysis, gauging, OCR, graphics, etc.

3.2 Goals

This system was designed to provide a more natural mean for humans to communicate with a pet robot. There are several goals that this system attempts to achieve to suit this application, such as:

- Real-time performance. The response of the robot to a human's gesture should be fast, otherwise the human's interest will decrease.
- Robustness to variable cluttered environments.
- No requirements for the user to wear cumbersome devices such as colored markers or data gloves.
- Ability to cope with variable lighting conditions.
- User independence.
- Use of standard low-cost hardware, without the need for special purpose equipment which has higher cost and reduces the generality of the system.

3.3 Assumptions

Robust hand gesture recognition is not easy due to the existence of arbitrarily complex backgrounds, variable lighting conditions, and use of uncalibrated cameras. Real-time and user independence requirements usually increase the difficulty of this problem. Thus, several assumptions were made to simplify the problem to be solved:

- The person interacting with the robot will be in constant motion. Since only dynamic gestures will be used, the hand will usually be in motion while the human makes a gesture.
- The robot is static while paying attention to the person making the gesture.
- The user's face does not appear in the image region where the hand gesture is made.
- People rarely wear skin-colored clothes. Users of the system are assumed to wear long-sleeved clothes.
- Gestures are executed with a single hand.
- Although different people might be interacting with the robot simultaneously, the robot will only pay attention to the person that appears closest to it.

3.4 Gesture Lexicon

Researchers have investigated different types of gesture lexicons. Some systems used the American Sign Languages [Starner et al. 98; Dorner 93], while others have chosen to generate their own lexicon [Nishimura and Oka 96; Cohen et al. 96]. Sign languages are attractive because they are well structured and some are very widely used, such as ASL in the United States. However, learning them requires a considerable amount of effort and time. Moreover, the signs should be easy to understand and use. In the case of standardized sign languages, non-signers cannot guess the signs in most cases. Finally, there is a big difference in the communication with other humans than with pets. Whereas a precise “language” is needed in human to human communication with a specific grammar, a small set of gestures simple enough to specify commands will suffice in the case of pets.

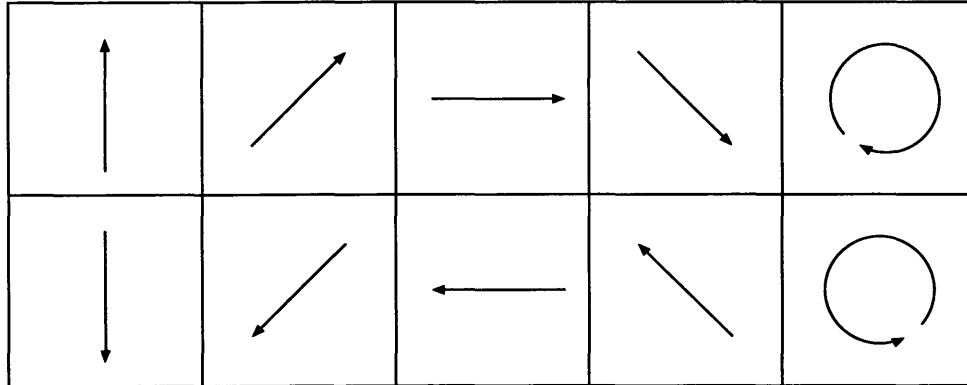


Figure 3-3: Basic gestures.

Because the meanings of gestures depend on culture, education and circumstantial factors, there is no predefined set of gestures that are natural for people to use. Thus, different sets of gesture lexicon suiting a different application can be combined to obtain a more general one. The gesture lexicon developed for this work uses navigation as a scenario of how these gestures will be used with a pet robot. The lexicon consists of a simple set of basic 2D gesture classes (primitives) that include linear (oriented vertically, horizontally, and diagonally) as well as circular (clockwise and counterclockwise) gestures. They are illustrated in figure 3-3. Each gesture class allows for:

- variations in the speed of hand's motion,
- different range of hand's motion,
- small deviations in the trajectory of the hand's motion (orientation, translation).

3.5 System Overview

This system is comprised of 3 independent modules: hand detection, feature extraction, and gesture recognition. The high-level architecture of the system is illustrated in figure 3-4 showing the interaction between the different modules.

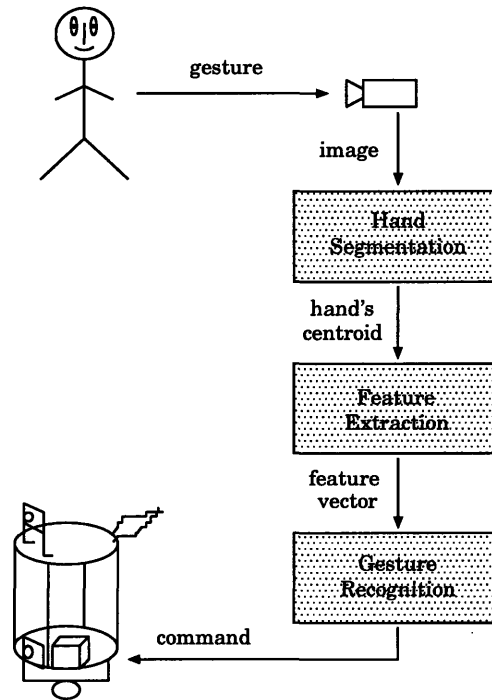


Figure 3-4: The system's high-level architecture.

Human-robot interaction will take place once the human is in front of the robot. As the human makes a gesture to the robot, the frame grabber will capture the images that the robot sees and input them to the hand segmentation module. This module will simultaneously track the motion of the hand and feed it into the feature extraction module, which will create the gesture's feature vector. Once the user finishes the gesture, the feature vector is used by the gesture recognition module to issue a command to the robot for it to perform the desired task.

3.6 Hand Segmentation

The goal of this module is to segment the hand from the cluttered environment and track it in successive image frames. As the hand is detected in each image, the position of the hand's centroid will be used for the feature extraction module to create a feature vector of the hand's motion.

To simplify the hand segmentation and also allow for further uses of this system as a module for multi-modal interaction, hand recognition is performed in a user-specified window of attention. This window does not include the face of the user. Thus, a more complex system, that would perform facial expression analysis besides hand gesture recognition, can be executed in a window of attention that excludes the hand. For simplicity, it will be assumed here that this attention window is the full sized image obtained from the camera.

Hand segmentation can be done based on edges or based on regions. Since this work will focus on dynamic gestures without imposing any 3-D constraints on the pose of the hand, it is virtually impossible to segment the hand using edges. Moreover, compared to edge detection, analyzing regions is more robust against noise and changes in illumination.

To achieve hand segmentation, the combination of two independent low-level detectors are used: motion and skin-color detectors. Two different detectors are used because no single one can provide the robustness required to segment the hand. If skin-color detection is used by itself in a cluttered environment can result in many false positives generated by light wooden doors and pink chairs that can sometimes be detected as skin. So, by noting that the person interacting with the robot is in continuous motion while making the gesture, the segmentation of the hand becomes a simplified problem. Motion detection helps constrain the recognition task by reducing the search space to the moving region, helping discard the false positives that do not move.

The frame grabber used for this work captures NTSC (640x480), but the images input to the hand segmentation module are subsampled to low-resolution (100x100)

RGB color images. These images are processed independently by both the motion and the skin-color detectors.

A data flow diagram of the hand segmentation module is illustrated in figure 3-5. However, before the actual algorithm of hand segmentation is discussed, a detailed description of each motion and skin-color detector will be provided.

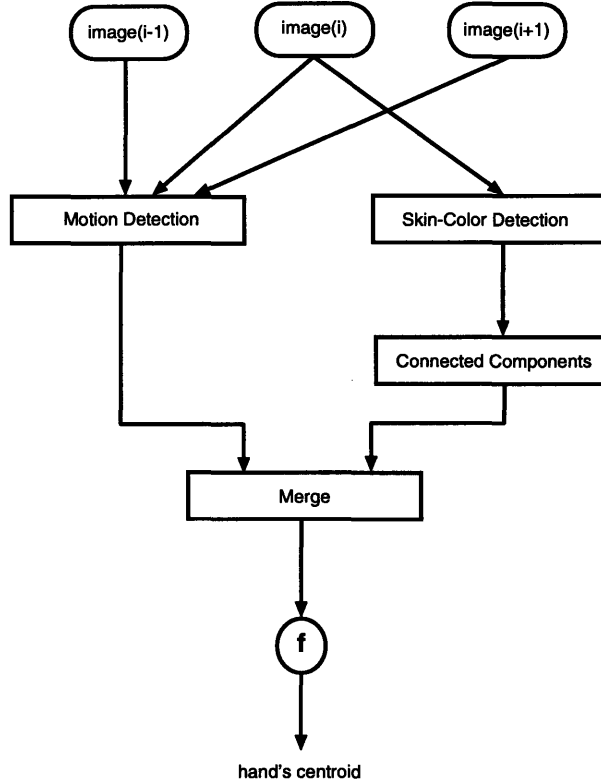


Figure 3-5: Flow diagram for the hand segmentation module.

3.6.1 Motion Detection

Several methods exist for motion detection. Optical flow is a very common method which estimates the relative motion by means of temporal variations of the brightness pattern [Horn 86]. The main limitation of this method is that in the absence of specialized hardware, it is usually very slow, thus inappropriate for real-time applications. Another common alternative is to use image differencing. This method

determines the edges of the moving components, and consists of taking the difference between corresponding pixel values of two subsequent images and selecting the pixels that pass this condition:

$$|I(t) - I(t - 1)| > \theta$$

where $I(t)$ is an image frame taken at time t and θ is a certain threshold. This simple technique includes pixels of the background which are covered in the previous frame in the result. To improve this method, image differencing with three images as opposed to two is used in this work. This modified method takes the difference between the first two and the last two images, and selects the pixels that satisfy:

$$|I(t) - I(t - 1)| > \theta \text{ and } |I(t - 1) - I(t - 2)| > \theta$$

The output of the motion analysis is a binary image (*motion_image*) where the moving pixels are those that pass the threshold θ :

$$motion_image = \begin{cases} 1 & \text{if moving pixel} \\ 0 & \text{otherwise} \end{cases}$$

This method can fail in the presence of shadows. Because shadows move with the entities that create them, using image differencing will detect motion in the shadow as well. However, this problem can be solved with the use of color.

Image differencing was implemented in this system using individual pixel difference between three input images. The same threshold was then used for each image color band (i.e., R, G, B).

The result of applying image differencing to three consecutive images is illustrated in figure 3-6. The image on the left is the last of three images taken. Observe that the bounding box is drawn on the result image on the right to surround the moving regions.

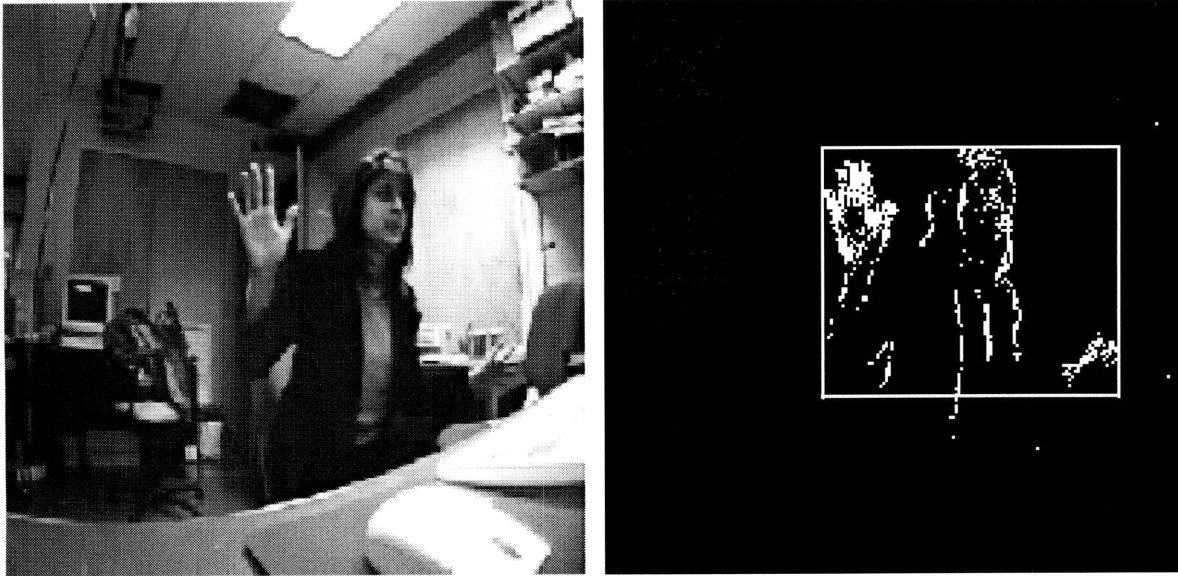


Figure 3-6: Image differencing results

3.6.2 Color Detection

There are several advantages to using color. One advantage is that processing color information is faster than processing other features of the hand for segmentation. Another advantage is that color is orientation invariant in the 2D image plane, assuming that the image intensity does not vary between adjacent image frames. Thus, only a translation model is needed for motion estimation.

Moreover, human skin colors tend to cluster in color space[Yang et al. 97]. One of the characteristics of human skin is that it is rich in lightly saturated red tones. This property is preserved across a wide range of skin colors, intensities, and lighting conditions, making color a reliable basis for skin segmentation.

Despite these advantages, there are also disadvantages in using color. These are due to deviations in the color representation which can occur due to changes in the lighting conditions, specular reflections, motion, and nonlinearities introduced by the camera and the frame grabber.

However, by choosing the right color space, deviations in the color representation can be reduced. Color can be represented in different spaces, which are different ways to create, specify, and visualize color. The position of a color within a particular color

space is specified by three coordinates or attributes.

RGB (Red, Green, and Blue) is the most common color space used in image processing given that it is used in monitors, cameras, computer graphics, etc. Simple RGB thresholding is an easy method to use and implement. Haynes and Taylor [1995] used a neural network with RGB values for skin segmentation. This approach has shortcomings, such as the need for training, which significantly limits the general utility of the segmentation technique (since new weights are required for different users with a different skin tonality), and the fact that it is not robust to changes in illumination. After using RGB for detecting a human's hand, the results obtained were not encouraging mainly due to variations in the lighting conditions.

An alternative was to use normalized RGB values (r , g , b).

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}$$

Scheile and Waibel [1995] and Yang et al. [1997] reported successful results by using these values, which are obtained by dividing the color vector by the brightness component. Only two of r , g , and b are used, since the third component is redundant ($r + g + b = 1$). Theoretically, this intensity normalization reduces the skin-color differences among people. Indeed, using this new color representation provided better results than using simple RGB thresholding. However, this method did not provide a stable hand segmentation for people from different races.

The work presented here uses the HLS (Hue, Lightness, and Saturation) color model (observe figure 3-7). Hue is the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors red, yellow, green and blue, or a combination of two of them. Lightness is the sensation of an area's brightness relative to a reference white in the scene. Saturation is the colorfulness of an area relative to its brightness. The HLS color space is a linear transform of the RGB space and the individual attributes h , l , and s can be obtained using a modified version of the RGB to HLS conversion algorithm from Foley et al. [1990], which is illustrated below. The procedure described in Foley et al. [1990] places red at 0 degrees of hue angle. However, skin color is characterized for its red tone; thus, to

avoid problems with the discontinuity from 0 to 360 in the hue attribute, an algorithm that has blue at 0 degrees of hue angle has been used. The routine below reflects this modification.

```
// Input : rgb each in [0,1] ; Output: h in [0,360], l and s in [0,1]
void RGB_to_HLS (float *r, float *g, float *b, float *h, float *l, float *s)
{
    float max, min, delta;

    max = Maximum (*r,*g, *b);
    min = Minimum (*r,*g, *b);

    *l = (float)((max + min) / 2.0);           // Lightness value.

    if (max == min) {
        // Achromatic case, since r==g==b
        *s = (float)0.0;
        *h = (float)-1.0;                     // Hue is undefined
    }
    else {
        // Chromatic case. Saturation computation.
        if (*l <= 0.5)
            *s = (max - min) / (max + min);
        else
            *s = (float)((max - min) / (2.0 - max - min));
        // Hue computation.
        delta = max - min;
        if (*r == max)
            // Resulting color is between yellow and magenta.
            *h = (*g - *b) / delta;
        else if (*g == max)
            // Resulting color is between cyan and yellow.
            *h = (float)(2.0 + (*b - *r) / delta);
        else if (*b == max)
            // Resulting color is between magenta and cyan.
            *h = (float)(4.0 + (*r - *g) / delta);
        *h = (float)(*h * 60.0);               // Convert to degrees
        *h = (float)(*h - 240.0);             // Shift Blue at zero hue
        while (*h < 0.0)                       // Be sure 0.0 <= hue <= 360.0
            *h = (float)(*h + 360.0);
        while (*h > 360.0)
            *h = (float)(*h - 360.0);
    }
}
}
```

Other linear transforms from RGB are HSL (Hue, Saturation, and Lightness), HSV (value), HSI (intensity), HCI(chroma/colorfulness), etc. Although device-dependent and non-linear as RGB, these color spaces are very intuitive. Moreover, for this particular application, where the goal is to segment a colored region under various lighting conditions, the separation of the luminance component is advantageous. By

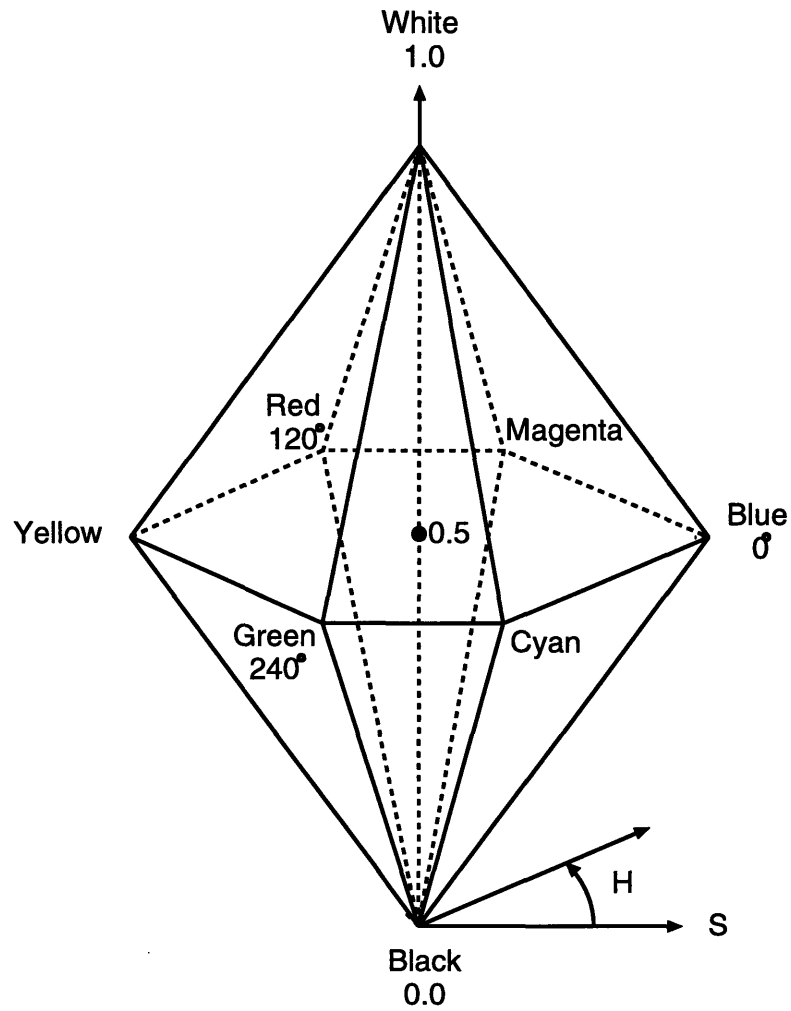


Figure 3-7: The HLS color model

disregarding the lightness component and using appropriately defined domains of hue and saturation, robustness is obtained towards changes in illumination and shadows. These results were corroborated with the work of Saxe and Foulds [1996] and Sobottka and Pitas [1996] who used the HSV color model to identify skin in color images, and by Kjeldsen and Kender [1996] who obtained good results as well by using the HSI color space.

An a priori model of skin color was used. To obtain bounds for the hue and saturation of human skin color, several skin patches with different tonalities and under different lighting conditions were plotted in HS space. Under a certain lighting condition, a skin-color distribution can be characterized by a multivariate normal distribu-

tion[Yang et al. 97]. Thus, the thresholds for hue and saturations ($H_{min}, H_{max}, S_{min}, S_{max}$) were obtained by determining an area (typically two standard deviations) around the mean values of H and S.

The skin-color detector scans the input image, and checks each pixel to determine whether it is skin-colored or not. In HS space, a pixel, which is a color pair (h, s) , is skin-colored if it passes the following condition:

$$(H_{min} < h < H_{max}) \text{ and } (S_{min} < s < S_{max})$$

A simple square representation in HS space is chosen for efficiency in computation. The output of this detector is a binary image:

$$skincolor_image = \begin{cases} 1 & \text{if skin-colored pixel} \\ 0 & \text{otherwise} \end{cases}$$

An example of skin-color detection is illustrated in figure 3-8. The result of applying skin-color detection to the left image is shown in the right image. Observe that there are background objects whose color resemble the skin color such as the wooden door in the back and the pink chair on the left of the image.



Figure 3-8: Skin-color detection

3.6.3 Combining Motion and Skin-color

The order in which the motion and skin-color detectors are run does not affect the results, given that they are totally independent. Because skin-color detection is computationally more expensive than motion detection in this implementation of the system, an optimization can be achieved by having the skin-color detector operate only on the moving regions in the image.

As can be observed in figure 3-5, initially three images ($image(i-2)$, $image(i-1)$, and $image(i)$) are obtained from the frame grabber. The motion detector takes the difference between the first two and the last two images. The result of these differences is stored in a new binary image ($motion_image$).

Given three images of a moving person, the difference of the first two images outputs an image where pixels of the background that were uncovered in $image(i-2)$ as well as pixels of the background that were covered in $image(i-1)$ are considered to have moved. Similarly, the difference of the two last images outputs an image where the pixels that were uncovered by the person's motion in $image(i-1)$ as well as pixels of the background that were covered in $image(i)$ are considered to have moved. Therefore, after taking an image differencing of three consecutive images, the edges of the moving person shown in the result will reflect the edges of the person in $image(i-1)$.

As an optimization for the skin-color detector, a bounding box is placed around all the moving components. This way, skin-color detection does not have to be performed on the whole input image, but only on the specified moving region, given that it is assumed that the hand is moving.

This bounding box is computed by scanning $motion_image$ from the top, bottom, left, and right, until the first pixel with half or more neighboring 1-pixels is found. These pixels will then determine the sides of the bounding box. The constraint on the neighboring pixels helps eliminate the possibility that single noisy pixels, which do not belong to any moving component, are chosen; thus, obtaining a tighter bounding box around the moving components.

The skin-color detector then scans an input image (this will be $image(i-1)$, given that the image differencing reflects the moving components in this image) within the area specified by the bounding box, and determines the skin-colored pixels. The result is stored in another binary image ($skincolor_image$). This image is then smoothed to obtain the skin-colored regions.

To cluster the different skin-colored regions, an 8-neighbor connected components algorithm is used. This algorithm first scans $skincolor_image$ and when it finds the first pixel set to 1, it uses this pixel as a seed and performs recursive region growing by checking its eight nearest neighbors for a 1. Each pixel in this region is assigned a particular label which is the same for all the pixels in that region. Then, the scanning of the image is resumed and other regions are found and labelled similarly.

After the initial processing on the input images, the results of the motion and skin-color detection are combined. The $motion_image$ is superimposed on $skincolor_image$. For each skin-colored region, a bounding box is placed around the moving pixels, to obtain a moving skin-colored region. If there is a skin-color region with very few moving pixels, this region is discarded as it cannot be a hand. This happens when there are objects of skin-color in the background (e.g. pink chair, a light-pink phone, etc.). Clearly, no skin-colored patch will be considered a potential hand if it does not move.

Once the different moving skin-colored regions are detected, they are analyzed individually by a function $f(moving_pixels, skincolored_pixels)$. The result of this function is a score, and the moving skin-color region that obtains the highest score is considered to be the hand.

For this implementation, the function f segments the hand based on the size of the skin-color region and the amount of motion within it.

$$f(moving_pixels, skincolored_pixels) = c_1 * moving_pixels + c_2 * skincolored_pixels$$

where c_1 and c_2 are constants that determine the weight each detector is given in segmenting the hand.

The advantage of using such a function is its flexibility. If a new detector is

added to the system, this function can be changed to accommodate a new argument. Furthermore, probability can also be used to segment the hand instead of a simple score.

Finally, having performed the segmentation, the centroid (\bar{x}, \bar{y}) of the hand is determined using the following formulae[Horn 86]:

$$\bar{x} \int \int_I b(x, y) dx dy = \int \int_I x b(x, y) dx dy$$

$$\bar{y} \int \int_I b(x, y) dx dy = \int \int_I y b(x, y) dx dy$$

where I is the image that combines both the motion and the skin-color detection results and $b(x, y)$ is the value of the pixel at (x, y) in image I .

After the location of the hand's centroid is determined, the process repeats for each new image. The hand's centroid is detected for every frame together with the last two frames that were taken so that the image differencing can operate on three images. If little motion was detected or the moving skin-colored region that attained the highest score was very small, then the last centroid obtained for the hand is used again.

An illustration of the hand segmentation is shown in figure 3-9. Image *a* shows the last image frame taken by the camera as the person makes the gesture in front of the robot. The bounding box in the image delimits the attention window where gestures will be recognized. Image *b* shows the results of the image differencing, and the bounding box surrounding the moving component. Image *c* shows the results of the skin-color detection within the bounding box specified in image *b*. Image *d* shows all the regions which are of skin-color and are moving. Then, the hand is selected from among the patches in image *d* and displayed in image *e*. Finally, the cross is drawn on the centroid of the hand in image *a*.

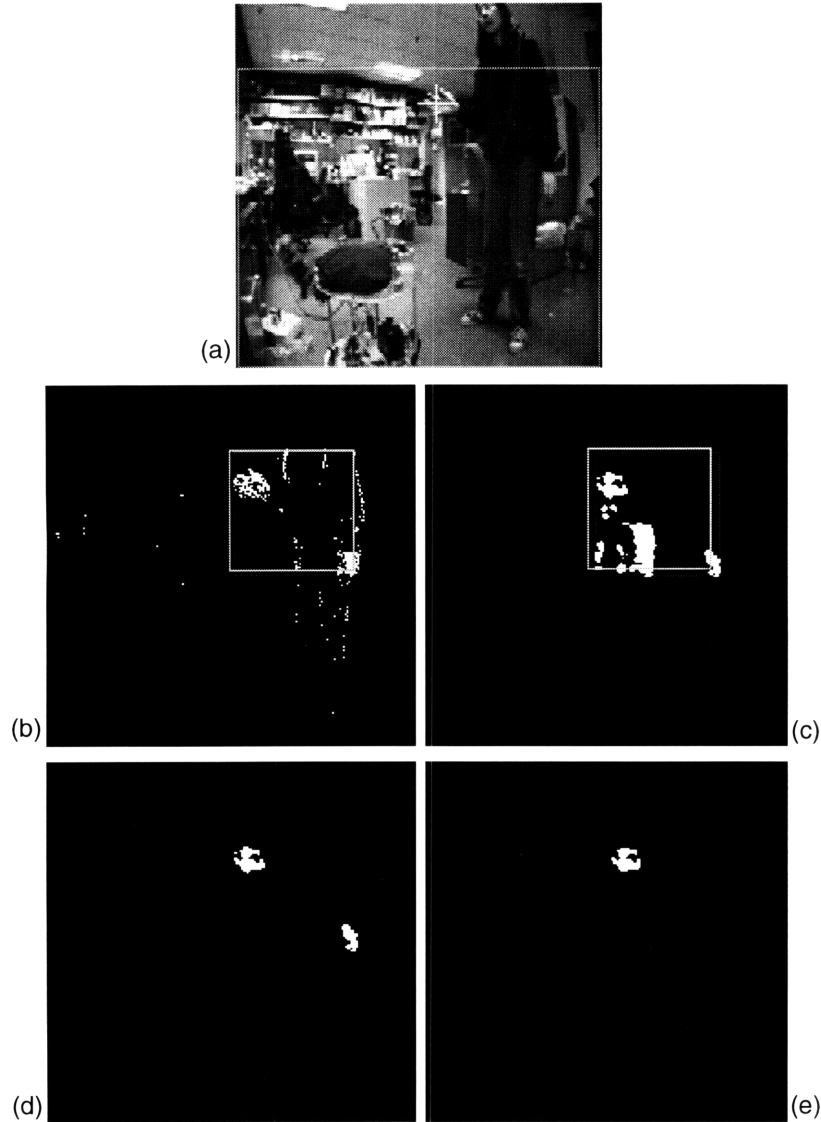


Figure 3-9: Hand segmentation

Once the hand is segmented for the first time, tracking could be used to determine the position of the same hand in the next frame. In this implementation, a simple measure using Euclidean distance between a hand centroid obtained in the previous image frame and the current centroid was enough to discard possible false positives and keep the focus on the hand. If the Euclidean distance between two consecutive centroid locations does not exceed a certain threshold, then the new centroid could be safely assumed to be the new location of the same hand; otherwise, the old location of the hand is kept.

Instead of using both motion and color for segmenting the hand in every image frame, an alternate approach was attempted. This approach assumed that the colored region of the hand from one frame overlaps with the one obtained in the following frame. This technique first obtained the first centroid of the hand using both detectors and then stored this centroid as the seed for color-based growing in the following frame. The drawbacks of this approach are rather obvious. Since motion was not used, a simple region growing of a previous hand centroid may overlap with other skin-colored objects in the background. In addition, the frame rate was not good enough to guarantee the assumption; therefore, this method was abandoned.

A small optimization can also be introduced by searching for the hand in a predicted search area surrounding its last location, however, this was not implemented in the system since the improvement to the overall performance would have been minimal.

3.7 Feature Extraction

The hand segmentation module outputs the hand's centroid in the current image frame as the person makes the gesture. But, before this information about the hand's trajectory is used by the gesture recognition module, it must be encapsulated in a representation that it can use to identify the appropriate gesture. Thus, the feature extraction module gathers the hand's trajectory information and outputs a feature vector that is then used by the gesture recognition module.

The feature vector must encapsulate the trajectory of the hand's motion as the gesture is made; thus, the beginning and ending of a gesture must be determined. This capability of determining when a gesture ends and a new one begins is called segmentation and is a major problem in gesture recognition.

Different approaches have been taken by the research community to solve the segmentation problem. Rubine [1991] studied 2D gesture recognition using a mouse or pen. The start of a gesture was determined by clicking a mouse button and the end was determined by the releasing it. Similarly, Cohen et al. [1996] used the on-

off states of a flashlight to determine the start and stop of a gesture which enabled them to isolate single gestures. However, these systems require the use of external devices which is unnatural and cumbersome. One other approach involves looking at the starting posture of a gesture [Murakami 91; Baudel and Beaudouin-Lafon 93]. This approach is not scalable because every gesture needs to have a different starting posture. Another approach suggested by Takahashi and Kishino [1992] was to have the user maintain a posture for one minute before it could be recognized. But, this method is not feasible for real-time applications. Harling and Edwards [1996] proposed the use of hand tension as a segmentation cue, but this method does not work on a sequence of dynamic gestures. An alternative approach is to always have a known starting gesture followed by the desired gesture. However, this is unnatural. Finally, it was suggested to use Hidden Markov models for gesture recognition given that segmentation between gestures is done during the recognition process. But, the disadvantages of using HMMs were already discussed in Chapter 2.

Given that no final solution has been determined for this problem, to distinguish one gesture from the other, this system assumes for simplicity that once there has been a considerable amount of motion of the hand, the person has started a gesture. At this point, the displacements of the hand's centroids in successive image frames will be calculated and stored in a feature vector until the hand stops moving for a short period of time. The amount of time that the user needs to stop his hand is usually between two to three seconds. The main advantage of this approach is that it is easy to implement and it is natural to the user. On the other hand, unwanted gestures caused by the hand's motion may potentially be interpreted by the robot. In practice, only gestures to which meanings have been assigned a priori will be interpreted by the robot, so not every random hand motion will create a meaningful gesture.

The feature vector used for this work contains the vertical and horizontal displacements (dx, dy) of the hand's centroid in the image space as the hand moves to make the gesture. These displacements are taken in consecutive image frames. An example of how this feature vector is formed is illustrated in figure 3-10. This figure shows on the left the sequence of how the hand's centroid varies in time, and on the right,

the feature vector obtained from the displacements of the hand's centroid. Because the velocity of a motion determines the start of a gesture, it can be noticed that the feature vector starts storing data if either dx or dy of the hand motion exceeds a certain threshold (which is 5 pixels per frame in this example). Moreover, if the hand has come to a stop, the displacements will be very small or zero; thus, they are not included in the feature vector given that they add no extra information.

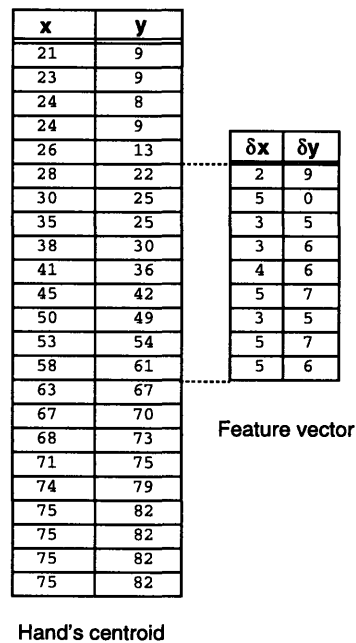


Figure 3-10: Example of feature extraction.

Other representations have been used for gesture recognition, which include the centroid of the hand in the image coordinates, that is, the hand's x , and y positions. Using the absolute position of the centroid requires normalization for translation and scaling. Normalization to accommodate translation can be done using the chest of the human as a point of reference or using the nose, as done by Starner et al. [1998]. Scaling should also be taken into account given that the person interacting with the robot may be closer or farther away from the camera [Yamato et al. 92]. Also other features, such as the second moment (angle of axis of least inertia), and eccentricity of a bounding ellipse considered to be the hand have also been used [Starner et al. 98].

Certainly, combining different local features might improve the robustness of the gesture recognition, however, this work explores the feasibility of gesture recognition by using only essential features.

3.8 Gesture Recognition

This module attempts to identify the gesture made by the user and translate that to a command that the robot can understand. Gesture interpretation is achieved by analyzing the feature vector created by the feature extraction module, which represents the trajectory of the hand's centroid in successive image frames.

In human communication, gestures are accompanied by the attitudes and emotions. Thus, being able to recognize these attributes besides recognizing the type of gesture made by the human would be desirable. This work attempts to provide a gesture recognition system that is capable of interpreting the speed of the gesture, which often represent these attributes.

For linear gestures, the displacements of the feature vector are contained in one of eight possible regions in $dx-dy$ plane (see figure 3-11)

For example, if the gesture to be recognized is a vertical motion, there are only two possible regions in $dx-dy$ plane where the displacements might be, both near the dy axis. If the motion is directed upward, then this region is near the positive side of the dy axis, whereas if the motion is directed downward, this region will be near the negative side of the dy axis. The same argument can apply for all the other linear primitives.

Displacements for linear gestures could potentially cluster in a small region. Nevertheless, gestures are not usually made at constant speed, and in most cases the motion tends to be more abrupt at the beginning of the gesture, whereas the velocity of the hand slows down as the gesture ends. In addition, the platform being used is a multi-purpose machine in which other applications also run in parallel with the system described here; thus, scheduling of other processes may occur, causing the time between frames to be variable. This can lead to variable displacements between

frames, causing the displacements to be scattered around one of the eight possible regions in $dx-dy$ plane which will allow their correct classification.

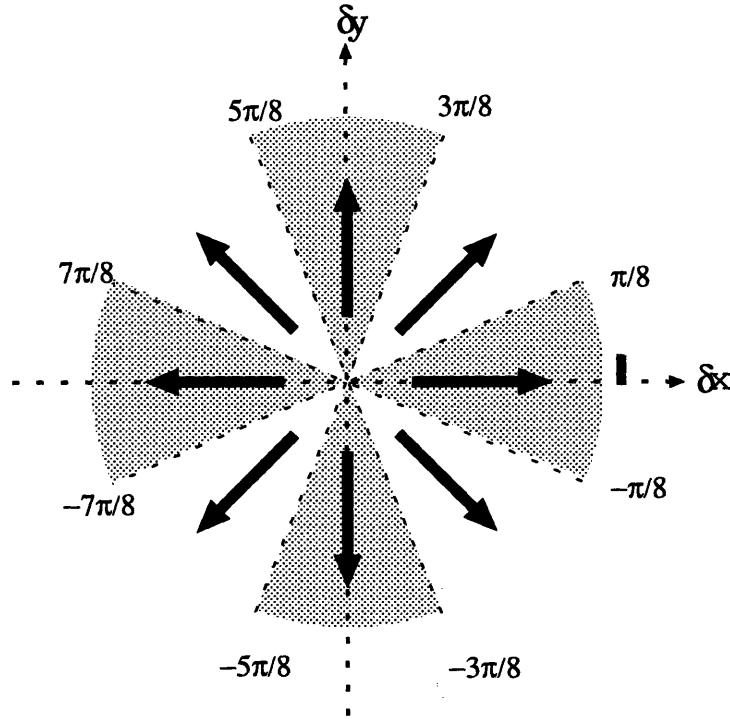


Figure 3-11: Possible orientations of the feature vector in the $dx-dy$ plane.

The centroid of this region, along with the origin of the $dx-dy$ plane, will determine a velocity vector. The orientation of this vector will determine to which of the eight linear gesture classes the user's gesture belongs. The magnitude of the vector will indicate the speed of the gesture.

On the other hand, the displacements produced from circular gestures are not contained in one particular region, rather, they are spread in all different directions in the $dx-dy$ plane. If the centroid of the region determined by the displacements of a circular motion in $dx-dy$ is near the origin, this is an indication that the gesture is not linear but circular. In this case a more careful analysis of the time sequence of the displacements in the $dx-dy$ plane will determine whether the motion was clockwise or counterclockwise.

Imagine dividing dx - dy plane into four quadrants. Quadrant I is where both dx and dy are positive, quadrant II is where dx is negative and dy is positive, quadrant III is where both dx and dy are negative, and quadrant IV is where dx is positive and dy is negative. Thus, observing how the different displacements are placed in these four quadrants as the gesture is made, the direction of the circular motion can be determined. The quadrant in which the circular motion starts is not relevant in determining the direction of the motion.

The process of determining the direction of the circular motion can be described in several steps. First, the feature vector is scanned from beginning to end and the signs of both dx and dy are noted. If two or more consecutive elements of the feature vector contain the same set of signs for both dx and dy , then the repetitive sets displacements are eliminated from the feature vector. This way, only sets of displacements that are different from each other are neighbors in the feature vector. Second, every sequence of four (dx, dy) pairs are compared with a time-sequence model for the clockwise motion and another one for the counterclockwise motion. The time-sequence model for the clockwise motion is $(- + , + + , + - , - -)$ whereas the one for counterclockwise motion is $(+ + , - + , - - , + -)$. The first element in each pair of signs is the sign of dx and the second is the sign of dy . The sequences reflect the way the camera sees the motion. Classification of the direction of motion will be determined by this matching.

3.8.1 Gesture Matching

Each primitive is assigned a label that distinguishes it from other primitives. After determining the basic gesture made by the user through the analysis of the feature vector, this information is represented in a more compact way, using a descriptor. The descriptor of a gesture contains the label of the identified primitive.

Model gestures are stored in a database. The specifics of how to use this database will be described in appendix A. This database contains user-specified model gestures and their corresponding meanings or commands for the pet robot. For example, a circular clockwise motion will command the robot to rotate in place clockwise. The

model gestures are stored as descriptors in the database. Therefore, a matching between a user gesture and a model gesture reduces to a simple matching of two descriptors. Once the descriptor produced from the user gesture is obtained, it is compared with each gesture in the database.¹ If a match is found, then the command associated with the gesture is retrieved and issued to the robot; otherwise, the user gesture is ignored.

3.9 True Extensibility

This hand gesture recognition system does not offer a full lexicon system for recognizing all the potential gestures that might be needed for a human to communicate with a pet robot. However, the simple gesture classes introduced above can be combined to create new gesture classes, allowing for an extensible hand gesture recognition system. Figure 3-12 illustrates some new gesture classes that could be created to specify a variety of new commands.

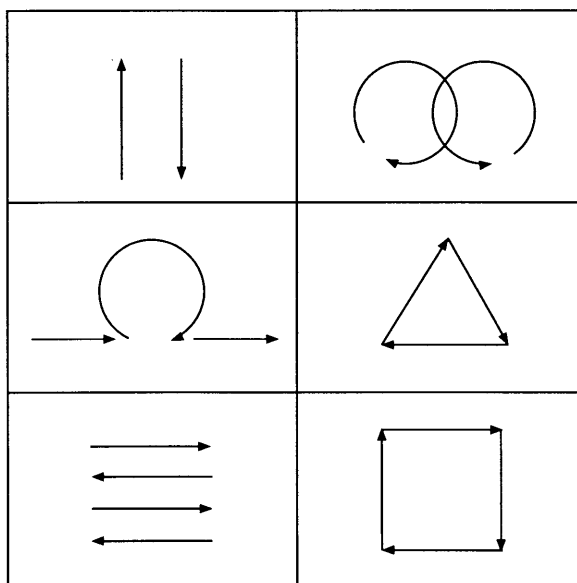


Figure 3-12: New gesture classes built from the combination of primitives.

¹Faster matching is possible with the use of special indexing mechanisms. However, these were not implemented due to the small set of gestures in this system

When the gesture is composite, the feature vector contains the displacements of the hand's centroid for each primitive. To segment these different primitives in one gesture, the same method as for segmenting different gestures is used: a pause is introduced between primitives. The pause is chosen to be shorter for separating primitives within the same gesture than for separating different gestures. The displacements for each primitive gesture are separated from each other in the feature vector by a specific marker.

A gesture with several primitives can similarly be represented as a descriptor, containing sequentially the different labels for the primitives that were identified in the user gesture. Thus, a gesture with one primitive will have a descriptor with only one label whereas a gesture with 3 primitives will have a descriptor with 3 labels.

New gestures containing any number of primitives with new different meanings can be created and stored in the database as needed. This flexibility allows for a truly extensible hand gesture recognition system that can suit the needs of different applications.

3.10 Meeting the Goals

The system presented in this chapter is a starting point toward understanding how vision can be used to recognize human gestures, as a way to enhance human-robot communication. This section will provide a brief evaluation of this system in light of its goals.

The two major goals were the achievement of real-time performance and robustness to complex environments. The different algorithms used for each module as well as the features for gesture recognition were chosen to be simple and fast to compute; thus, guaranteeing real-time performance on standard hardware. Robustness to variable cluttered environments is achieved given that hand segmentation is performed using both motion and skin-color information.

The use of skin-color for detecting the hand eliminates the need for the user to wear cumbersome gloves or markers. Moreover, despite the fact that skin-color varies

among people, they are all rich in lightly saturated red tones which cluster in the color space. This allows the use of skin-color to segment body parts from the background, allowing this system to be user independent. A limitation in using color for hand segmentation is that color appearance tends to vary as the illumination conditions change, but by ignoring the lightness component of the colors perceived in the image, robustness is achieved toward a reasonable range of lighting conditions. Given that pet robots are designed to work indoors, it is reasonable to assume that the robot will not have to face drastic variations in lighting conditions.

Furthermore, this hand gesture recognition system provides a mechanism to be extended to recognize a larger set of gestures. In broad terms, the goals of this system are met with the design and implementation described above; however, a more detailed evaluation of the system will be given in Chapter 4.

Chapter 4

Experimental Results

This chapter provides an evaluation of the hand gesture recognition system described in the previous chapter. The results reflect how well the system's modules perform individually, as well as in conjunction with each other. First, the accuracy of a set of gestures will be shown and discussed together with some weaknesses of the system. Then, several robustness issues of the system will be described. Finally, the improvements in the interaction with Yuppy will be presented.

4.1 Performance

One very important factor that will help determine the overall performance of this system is how accurately it can recognize gestures. For this, an experiment was performed on people of different ethnic origin (with different skin tones) in cluttered environments, under variable conditions, to determine the accuracy rate of a set of sample gestures. The tests were performed on a 300MHz Pentium Pro computer with 128 MB of RAM achieving 15 frames a second.

This experiment was performed on 14 people sitting in front of the camera that was used for development. Among these people, 6 were Asians (3 of which were of Indian origin), 1 was Hispanic, and the rest were Caucasian. The test subjects were dressed with long-sleeved shirts.

To take advantage of its field of view, the camera was positioned such that the

face of the subject was not shown in the image, avoiding the need to use an attention window that would discard the face. Given that the camera was low, pointed downward, sometimes the subject's legs would enter the field of view. If they were wearing shorts, their legs were covered in order to prevent disruptions in the hand segmentation (due to the similarity in skin color).

The conditions chosen for this experiment were variable. The test subjects were sometimes standing and other times seated at 2 to 4 feet away from the camera. The tests were performed in a laboratory setting, with different cluttered backgrounds. The testing room was illuminated by various fluorescent lights and had three large windows on one side; the tests were also performed during different times of the day to demonstrate robustness to variable lighting conditions. Figure 4-1 shows a picture of part of the testing environment.

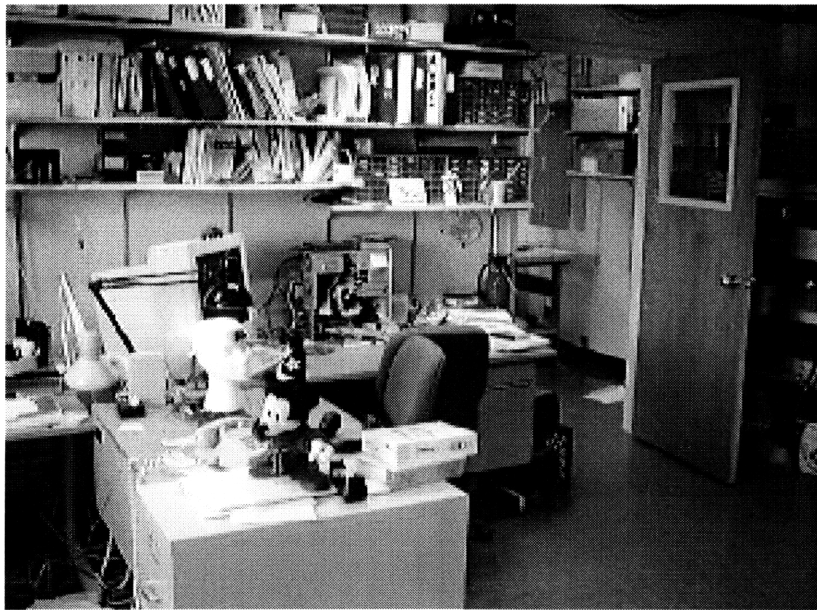


Figure 4-1: Example of testing environment

A group of 16 different gestures were chosen for this experiment (see figure 4-2), consisting of 10 primitive gestures and several gestures composed of 2 to 4 primitives. Although some of the more complex gestures might not be used in a real interaction

with a robot, (e.g. make-triangle, make-square, make-omega, which would require the robot to navigate in such a way as to create a triangle, a square, or the omega symbol) it shows how the system performs on composite gestures. The results were obtained from real-time sequential images taken by having each of the 14 subjects repeat the sequence of gestures 5 times. Thus, the accuracy for each gesture was determined by the following formula:

$$\text{accuracy_rate} = (\# \text{ of correct gestures}) / (\# \text{ of total gestures})$$

where the *# of total gestures* of the same class was $14 \times 5 = 70$.
















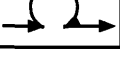
Gesture		Accuracy (%)	Gesture		Accuracy (%)
	front	100.0		rotate-right	90.0
	right	97.1		rotate-left	95.7
	left-back	100.0		calm-down	91.4
	right-front	98.6		dance	80.0
	back	100.0		hi	82.9
	left-front	97.1		make-square	72.9
	right-back	95.7		make-triangle	71.4
	left	100.0		make-omega	80.0

Figure 4-2: Gestures accuracy

As expected, the results demonstrate that, the more natural and simpler the gestures are, the higher their recognition accuracy is. The subjects were asked to perform the gestures naturally, without any significant constraints, and the environment was not set up in any special way. The purpose of this was to simulate expected conditions under which the robot may perform. When performing the tests, it was noticed that

failures of the system were due to three different causes: the subject not performing the gestures precisely, setup problems, and shortcomings of the system's design. Each one of these causes will be explained in detail below.

The way gestures are made varies greatly among different subjects, and for several reasons some people obtained higher accuracy rates for certain gestures than others. The test subjects were asked to make each gesture in the sequence as illustrated in figure 4-2. Most subjects were not trained before performing the tests, but as they repeated the sequence, they became more familiar with the gesture and started getting an overall better recognition rate. A goal of this work is for anyone to be able to interact with the robot without being subjected to previous training. Since any subject could see on the computer screen the results of the tests, together with a real-time image of what the camera was seeing, some subjects used this feedback to improve their gesture performance. This is why subjects tended to improve their overall accuracy as they performed more tests. However, other users, looked at the camera rather than the screen and performed similarly well, given that they were not getting distracted by the events on the screen. A few subjects were asked to perform the sequence of gestures once before starting the tests and these people generated better recognition rates than those who did not receive training, as was expected. In practice, when people interact with robots, this training happens implicitly as the person observes the robot's reactions to their gestures.

The concentration of the subjects in the tests was also very important. A few users tried to exaggerate the motions for easy recognition. However, others did not concentrate too much on the task and instead of moving their hand to the right, for example, they were moving it down and then right. Although the system is reasonably robust to these motions, sometimes the gesture was recognized to have two primitives as opposed to one.

From observing how some gestures were made by the subjects, it was noticed that people are better at making continuous movements with their hands than several discontinuous movements. Although the subjects were asked to perform each primitive sub-component of the composed gestures in an independent fashion, it was often the

case that subjects were combining two or three of the primitives into one continuous motion. Moreover, diagonal motions do not seem to be as natural for people as vertical or horizontal motions are. Very often, when the subjects were making a gesture that involved a diagonal motion, they were making it fairly vertical, thus, reducing the accuracy of these types of gestures.

A problem faced with the experimental setup was that sometimes the hand would leave the camera's field of view while the subject was gesturing. Thus, the full trajectory of the hand was not detected. This is not a weakness of the hand gesture recognition system, so in most cases when this situation was noticed, the subject was asked to repeat the gesture. Nevertheless, this situation had not been noticed at the beginning of the tests. Hence, some of the failures in recognizing gestures made by the first subjects could have been due to this circumstance. A wider angle camera would have been a simple solution to this problem.

The distortion caused by the camera's position was another source of failure in gesture recognition. The camera used for the experiments was not perfectly parallel to the horizontal plane; its angle was not fixed on purpose, given that in practice, while interacting with the robot, the camera would not always be straight either. In addition, it was slightly tilted upwards or downwards in all the testings, suiting the particular position of every subject. Tilting the camera can cause diagonal motions to be distorted. For example, if the camera is tilted upwards or downwards, a diagonal motion would look straighter than it really is, reducing the horizontal component of the gesture. On the other hand, if the user was translated too much to the right or to the left of the center of the camera, the motion would look more horizontal than it really is.

The accuracy differs from primitive gestures to composite gestures. The overall accuracy of the primitive gestures is at least 90%. On the other hand, gestures composed of several primitives obtained lower accuracy, which was slightly above 70%. There are several reasons for why the accuracy was lower on gestures with several primitives, such as the interdependence between primitives in the gesture, the accuracy of individual primitives composing these gestures, and the accuracy in the

segmentation of each primitive. The first is due to how the primitives were combined in the composite gestures, whereas the last two are more relevant to the system's implementation.

In a gesture composed of various primitives, the order in which the primitives should be performed greatly affects the accuracy of the gesture as a whole. Although this problem is not intrinsic to the design or implementation of the hand gesture recognition system, it was a factor that affected its performance. A typical example of this is the *make-triangle* gesture. A major reason for its low accuracy was that subjects found it difficult to make two consecutive diagonal motions correctly. As opposed to using their body (located usually in the center of the camera's field of view) as the center for the two diagonal motions, people would often make the first primitive gesture directly in front of their body, and in this case, the second primitive was done on the right of the user's body. This could have presented three different difficulties while making the second diagonal motion. First, sometimes it was hard to stretch the hand further right even if it was needed to perform a right diagonal motion. Second, the user was very conscious of getting out of the field of view. Third, since the gesture was not made directly in front of the person's field of view, it was hard to determine if the motion was properly diagonal. All these reasons resulted in the gesture *make-triangle* being recognized in some cases as composed of the primitives *right-front*, *back*, and *left*.

Gesture recognition becomes potentially less accurate as the gesture contains more primitives, given that its accuracy depends on the accuracy of the individual primitives composing it. Since these primitives need to be combined, there are more chances of failure if recognition of each primitive gesture has lower individual accuracy. The main reason is the way the database was designed to match model and user gestures. If the descriptors are not matched exactly, then the gesture is not recognized, thus, to obtain a matching, every single primitive needs to be recognized accurately. However, this problem can be reduced by devising a similarity function to match a user descriptor to the most similar model descriptor in the database.

Figure 4-3 shows the obtained and the expected accuracy for the composite ges-

tures. Expected accuracy is computed by taking the product of individual primitives composing the gesture. As can be seen, the obtained accuracy are lower than the expected ones. For simpler composite gestures, the difference between the obtained and the expected accuracy was within 11.4%, which could be explained by the difficulty in segmenting the primitives. For more complex composite gestures such as the make-square and the *make-triangle*, this difference exceeded 20% which can be explained both by the segmentation involved and also by the fact these gestures are not natural in human communication and are difficult for users to perform.



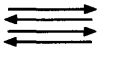



Gesture	Expected Accuracy (%)	Obtained Accuracy (%)
	100.0	91.4
	86.1	80.0
	94.3	82.9
	97.1	72.9
	94.4	71.4
	84.9	80.0

Figure 4-3: Expected vs. obtained accuracy for composite gestures

Gesture segmentation is another cause of the lower accuracy obtained in composite gestures. There is a fixed threshold for the time that the user needs to stop the hand to transition from one primitive to the other, and to transition between different gestures. However, some people are not very good at this task. The results of the tests show that some people were stopping for too long while making a transition

between primitives, and therefore, the system assumed the gesture had ended. On the other hand, some subjects were not stopping to transition between primitives, causing the system to interpret the whole motion as one primitive. In practice, timing is not a real constraint when signalling the end of a gesture given that in most cases, the robot will react after it has interpreted a single gesture; thus, the user can stop the hand until the robot reacts to the gesture.

Another problem was that some people did not start the gesture at an appropriate velocity. Because there is a fixed threshold for determining the initial velocity of a gesture, when someone started a gesture very slowly, or the range of the motion they were covering was very short in a time step, this module would not detect the start of the gesture. This occurred for example when some subjects made the *right* primitive gesture, which explains its low accuracy compared to the other vertical and horizontal primitives.

A limitation exists in the hand segmentation module due to the difficulty in distinguishing the hand from other patches of moving skin in the robot's attention window. Because the body moves with the hand as it makes a gesture, other body parts such as the face, arms, hands, legs, etc. are potential regions to be selected by this module. The face and arms do not represent a problem because this system assumes that the face does not show up in the attention window and the user wears a long-sleeve shirt. The legs usually do not show up in the attention window given that the person interacting with the robot is sufficiently close to it. The problem arises when the user moves both hands together because only one hand is segmented. In a sequence of images, the centroid may end up jumping from one hand to the other. In practice however, as one hand is used to make the gesture, it is closer to the camera and therefore bigger. Moreover, it moves more than the hand which is not being used for making the gesture, thus, the correct hand gets segmented most of the time. Similarly, this problem arises whenever there are several people in the attention window. This problem could be solved by using an appropriate tracking mechanism that would not lose sight of the hand.

Furthermore, hand color detection fails in the presence of red and dark pink colored

clothing. This is because skin is very rich in red tones, so the hue thresholds that define the skin color model are biased toward this color. In addition, the color detector does not perform very well when the images are extremely bright or very dark. Such cases can happen when the camera is tilted upward and images are taken with several fluorescent lights in the background, or when the images are taken in a room with insufficient lighting. Finally, hand segmentation might fail in the cases where the shadow of the moving human is similar in color to the skin.

4.2 Robustness

Despite some of the problems described in the previous section, this system achieves user independence by being fairly robust in recognizing hands of various tonalities. This system was tested successfully on Japanese, Chinese, Indians, Caucasians, and Hispanics. Figure 4-4 shows results for a Caucasian and an Indian. Tests were not performed on African races because subjects were not available during the experiments. Given the results of the tests with two Indians with two fairly dark Indians, the system is expected to perform well in all cases. The change in skin tonality between the different subjects did not appear to have a direct effect on the accuracy of different gestures.

The system also showed robustness to a variety of lighting conditions and complex backgrounds in a laboratory setting. Similarly, the system showed robustness to changes in the background colors even when they influenced the appearance of the skin color.

Robustness was shown in the recognition of gestures even when the centroid of the hand was “lost” during the performance of the gesture. In most cases this was due either to parts of the user’s face entering the attention window, or the other hand moving in the field of view. In other cases it was due to the fact that the hand got out of the field of view while the gesture was performed. In some circumstances, the gestures were still recognized successfully.



Figure 4-4: Robustness to two different skin-colored people: Caucasian (left) and Indian (right)

It is challenging to build a system that is robust enough to account for the fact that people might use it in unforeseen ways. This system exhibited the ability to cope with one such situation. On one occasion, the user who was playing with the system, used his face to give commands to the robot, and then the robot successfully recognized them as if they were made with the hand.

4.3 Yuppy's Navigation

The gesture recognition system has been tested and run on Yuppy. An interface was developed to control the robot's navigation using each primitive gesture as a navigational command for the robot. Thus, an upward hand motion will command the robot to go front, whereas a circular motion directed clockwise will command the robot to rotate in place clockwise. The gesture recognition was done using an external camera. Issues of relative motion between the camera and the robot were therefore limited. This system however, turned out to be quite robust and even a naive observer can easily learn to control the robot this way.

This hand gesture recognition system already allows people to interact with Yuppy in a more natural fashion. Moreover, it provides a basic interface for future behaviors

implemented on Yuppy, such as approaching a person, searching for a bone, fetching the newspaper, etc.

It is clearly necessary for the robot to interpret correctly what the gestures mean and be able to respond accordingly. But, even communication between humans is sometimes ambiguous, and we expect the communication between humans and animals to have even more ambiguities. Thus, unexpected behaviors are welcome. People expect objects to react immediately, predictably and consistently; however, they are more tolerant with humans and animals. Humans can accept that a pet robot might not have perceived or correctly interpreted the request.

Chapter 5

Conclusions

5.1 Summary

This thesis presented a novel approach to hand gesture recognition, which satisfies the need for a natural interface for human-robot communication. The system does not require the user's hand to wear a special marker or that the environment be special in any way.

Although creating an interface for humans to communicate with robots has been researched before as a possible application of a hand gesture recognition system, few have been implemented for this purpose. In fact, most of the systems have been developed for other applications such as virtual-reality, multimedia, intelligent security systems.

This work presented a hand gesture recognition system for visually interpreting 2D dynamic hand gestures. The system explored the use of fast algorithms and simple features to determine the trajectory of the hand in real-time. The system consists of three main modules: hand segmentation, feature extraction, and gesture recognition. The hand segmentation module uses both motion and skin color to segment the hand from a cluttered environment. Once the segmentation is performed, the centroid of the hand is found and used to obtain a feature vector. The feature extraction module consists of determining the dx and dy displacements of the hand's trajectory from when the gesture starts until when it ends and storing this information in a vector

used to interpret the gesture. Finally, the gesture recognition module uses this feature vector to interpret the gesture made by the user and sends the command to the robot.

Single directional linear motion gestures and circular gestures were implemented as primitive gestures and their combination allows for an extensible gesture interpretation scheme. The primitive gestures are recognized by observing in which region the displacements in the feature vector are placed in the $dx-dy$ plane. Circular gestures are distinguished from linear gestures because the displacements in the feature vector are scattered in different regions in the $dx-dy$ plane. Thus, an analysis of the time sequence of the displacements will determine whether the circular motion is clockwise or counterclockwise.

This system performs in real-time and was tested on people of different ethnic origins, placed in a cluttered environment under variable lighting conditions. Robustness was shown in the hand segmentation given these conditions. This system was also tested on Yuppy, an emotional pet robot, and satisfactory results were obtained while navigating the robot around the laboratory using hand gestures.

5.2 Future research

The work presented is a step toward solving the more general gesture recognition problem. Gesture recognition is a recent field of research and much work still needs to be done toward creating natural interfaces for use not only in robotics, but also in virtual reality environments, intelligent rooms, etc.

This system has been designed and implemented in a modular fashion so that components can be substituted and added as needed. This system can be improved in many different ways. To improve the hand segmentation module, a fast and robust implementation of hand tracking would considerably increase the performance of the system by avoiding the detection of the other body parts of the user and preventing the human from getting out of the field of view of the robot's camera. This would also allow several people to interact with the robot at the same time. In addition, using mechanisms for color detection that could learn to adapt in time to different

skin tonalities and different illumination conditions would be desirable.

The feature extraction module can also be enhanced once a more elegant and robust solution to gesture segmentation is found. Also, other features can be used in conjunction with x and y displacements to recover the hand's trajectory more robustly.

Gesture recognition can be improved in several ways. A mechanism that would enable the robot to learn the gestures made by the human (e.g., through reinforcement) could be implemented. Moreover, the use of a similarity measure may improve matching accuracy between user and model gestures in the database.

This system also opens doors to a great number of other interesting work. Given that this system only studies 2D dynamic hand gestures, the next step could include recognition of hand poses and 3D gestures. Also, by allowing the robot to move freely while interpreting the human would greatly improve human-robot interaction.

A multimodal interaction between humans and computers are becoming increasingly important, and a system that recognizes and understands the gestures and voices of more than one person would be highly desirable for building a flexible human-machine interface. Thus, speech recognition, facial expressions, and physical contact between humans and robots could be supported to enhance their interaction.

Besides being able to classify a user's gestures, it would be interesting to interpret the human's attitude and emotions that are implicit in the gestures. This work has provided a way to determine the speed of the gesture, however, the velocity pattern and the way the gestures are made could be studied to learn more about the human interacting with the robot. On the other hand, robots can also show emotions by the way they react to the human's gestures. They do not necessarily have to respond to the human's command, they can act both according to their internal emotional states and to what they perceive from the environment. An even more interesting question is to study the user's reaction to the robot's behavior.

It is clear that the opportunities for interesting interaction with pet robots becomes endless once a suitable communication scheme is in place. This work was a step toward a more natural human-machine interaction.

Bibliography

- Acredolo L. and Goodwyn S., 1996. *Baby Signs: How to Talk with Your Baby before Your Baby Can Talk*.
- Appenzeller G., “Building Topological Maps by Looking at People: An Example of Cooperation between Intelligent Spaces and Robots.” In *IROS*, 1997.
- Baudel T. and Beaudouin-Lafon M., 1993. “CHARADE: Remote control of objects using free-hand gestures.” *Communications of the ACM* pp. 28–35.
- Bobick A. and Wilson A., “A state-based technique for the Initialization and recognition of gesture.” In *International Workshop on Automatic Face and Gesture Recognition*, 1995, pp. 382–388.
- Campbell L., Becker D.A., Azarbayejani A., Bobick A., and Pentland A., “Invariant features for 3-d gesture recognition.” In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 157–162.
- Cohen C., Conway L., and Koditschek D., “Dynamical System Basic Oscillatory Generation, and Recognition of Motion Gestures.” In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 60–65.
- Dang D., “Template Based Gesture Recognition, M.S. Thesis.” Tech. Rep. TR-, Massachusetts Institute of Technology, Cambridge, MA, 1996.
- Darrell T., Maes P., Blumberg B., and Pentland A.P., “A Novel Environment for Situated Vision and Behavior.” Tech. Rep. TR-261, M.I.T. Media Laboratory, Cambridge, MA, 1994.
- Darrell T. and Pentland A., “Space-Time Gestures.” In *CVPR’93*, 1993, pp.

335-340.

Dorner B., "Hand shape identification and tracking for sign language interpretation." In *Looking at People Workshop*, 1993.

Foley J.D., van Dam A., Feiner S.K., and Hughes J.F., 1990. *Computer Graphics, Principles and Practice*. Addison-Wesley.

Freeman W.T. and Weissman C.D., "Television Control by Hand gestures." In *Workshop on Automatic Face and Gesture Recognition*, 1995.

Grimes G.J., "Digital data entry glove interface device." Tech. rep., Bell Telephone Laboratories, 1993.

Harling P. and Edwards A., "Hand Tension as a Gesture Segmentation Cue." In *Progress in Gestural Interaction*, 1996, pp. 75-88.

Haynes C.W. and Taylor C., "Hand Recognition from image data." In *In Human Computer the Conference on Intelligent Interaction*, 1995, pp. 39-42.

Heap T. and Hogg D., "Towards 3D Hand Tracking using a Deformable Model." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 140-145.

Hienz H., Grobel K., and Offner G., "Real-Time Hand-Arm Motion Analysis using a single Video Camera." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 323-327.

Horn B., 1986. *Robot Vision*.

Huang X., Ariki Y., and Jack M., 1990. *Hidden Markov Models for Speech Recognition*.

Kjeldsen R. and Kender J., "Toward the Use of Gesture in Traditional User Interfaces." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 151-156.

Murakami K., "Gesture Recognition using recurrent neural networks." In *Proceedings of CHI'91*, 1991.

- Nagaya S., Seki S., and Oka R., "A theoretical consideration of recognition. Injectory for gesture spotting." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 72–77.
- Nishimura T. and Oka R., "Spotting Recognition of Human Gestures from Time-Varying Images." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 318–322.
- Nowlan S. and Platt J.C., "A convolutional neural network hand tracker." Tech. rep., Synaptics, 1992.
- Rehg J. and Kanade T., "Visual tracking of high DOF articulated structures: An application to human hand tracking." In *3rd ECCV*, 1994, pp. 35–45.
- Rubine D., 1991. "Specifying gestures by example." *Computer Graphics* 25 pp. 329–337.
- Saxe D. and Foulds R., "Toward Robust Skin Identification in Video Images." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 379–384.
- Scheile B. and Waibel A., "Gaze Tracking Based on Face Color." In *International Workshop on Automatic Face and Gesture Recognition*, 1995.
- Sobottka K. and Pitas I., "Segmentation and Tracking of Faces in Color Images." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 236–241.
- Starner T., Weaver J., and Pentland A., "A Wearable Computer Based American Sign Language Recognizer." In *To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.
- Takahashi T. and Kishino F., 1992. "A hand gesture recognition method and its application." *Systems and Computers in Japan* .
- Triesch J. and von der Malsburg C., "Robust Classification of Hand Postures against Complex Backgrounds." In *International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 170–175.

- Triesch J. and von der Malsburg C., "A Gesture Interface for Human-Robot-Interaction." In *Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.
- Yamato J., Ohya J., and Ishii K., "Recognizing Human Action in Time-Sequential Images using Hidden Markov Model." In *Conference on Computer Vision and Pattern Recognition*, 1992, pp. 379–385.
- Yang J., Lu W., and Waibel A., "Skin-Color Modeling and Adaptation." Tech. Rep. CMU-CS-97-146, Carnegie Mellon, Pittsburgh, PA, 1997.
- Zimmerman T. and Lanier J., "A hand gesture interface device." In *ACM SIGCHI/GI*, 1987, pp. 189–192.

Appendix A

User Manual

A simple interface has been implemented to allow this hand gesture recognition system to be easily used. This interface consists of two main tasks. First, the user will have to specify the attention window where the gestures will be recognized, and second, the user will have to define model gestures that will be used and their corresponding meaning.

To specify the attention window for gesture recognition, the user needs to set the values of four global variables: X_{min} , X_{max} , Y_{min} , and Y_{max} which specify the top-left corner and bottom-right corner of the attention window. If these parameters are not specified by the user, the attention window will default to the camera's field of view.

The user will also have to specify the gestures that are intended to be recognized. Gestures are specified using a descriptor. A descriptor is a vector of primitive labels. Each primitive will be assigned an integer label for rapid computation.

The descriptor preserves the order in which the primitives are supposed to be gestured. For instance, the descriptor for a gesture that would make the robot rotate in place clockwise would be:

$$gesture_rotate_right = \{9\}$$

assuming that the label for *rotate-right* is 9. If on the other hand, the user wanted to make the waving gesture which is composed of 4 primitives, then the descriptor

will look like this:

$$gesture_hi = \{4\ 0\ 4\ 0\}$$

assuming that the label for *right* is 4, and the one for *left* is 0. For each model gesture, an action will also have to be specified. The actions are specified by using a function that could specify commands to the robot. An action method for the rotate_right gesture may look as follows:

```
void ActionRotateRight(void)
{
    BOOL result;

    result = SendCommand(TurnRightVeloc, 50);
    if (result == FALSE)
        printf("Error: TurnRightVeloc\n");
}
```

Once these gestures have been specified, they need to be inserted in the database so that they can be used during the recognition stage for matching the user and the model gestures. The model gestures are inserted in during the execution of the InitializeDatabase procedure using the function InsertGesture. The InsertGesture function takes two different arguments, a descriptor of the gesture to be inserted and a pointer to the corresponding action method, which will be invoked if a matching occurs.

5-12-04