HUMAN ARM MIMICKING USING VISUAL DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALGAN USKARCI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet Erkmen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc.Prof. Dr. Aydın ALATAN Co-Supervisor	Prof. Dr. Aydın Ersak Supervisor

Examining Committee Members

Prof. Dr. Aydan Erkmen	(METU,EE)	_____
Prof. Dr. Aydın Ersak	(METU,EE)	_____
Assoc.Prof. Dr. Aydın Alatan	(METU,EE)	_____
Prof. Dr. İsmet Erkmen	(METU,EE)	_____
Haluk Zontul, M.S.	(TÜBİTAK,BİLTEN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name :

Signature :

ABSTRACT

HUMAN ARM MIMICKING USING VISUAL DATA

USKARCI, Algan
M.S., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. Aydın ERSAK
Co-Supervisor: Doç. Dr. Aydın ALATAN

December 2004, 85 pages

This thesis analyzes the concept of robot mimicking in the field of Human-Machine Interaction (HMI). Gestures are investigated for HMI applications and the preliminary work of the mimicking of a model joint with markers is presented. Two separate systems are proposed finally which are capable of detecting a moving human arm in a video sequence and calculating the orientation of the arm. The angle of orientation found is passed to robot arm in order to realize robot mimicking. The simulations show that it is possible to determine human arm orientation either by using some markers or some initial background image information or tracking of features.

Keywords: Robot Mimicking, Computer Vision, Human-Machine Interaction

ÖZ

GÖRSEL VERİ KULLANILARAK İNSAN KOLU TAKLİDİ

USKARCI, Algan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü Bölümü

Tez Yöneticisi : Prof. Dr. Aydın ERSAK
Ortak Tez Yöneticisi: Doç. Dr. Aydın ALATAN

Aralık 2004, 85 sayfa

Bu tez İnsan-Makine Etkileşimi (İME) çerçevesinde robot taklidi konusunu incelemektedir. EMİ uygulamaları açısından el ve kol hareketeleri incelenmiş ve bir model eklemin taklidi üzerine yapılan ön çalışmayla ilgili bilgiler sunulmuştur. Son olarak bir video sekansı içerisinde hareket eden bir insan kolunu bulan ve kolun yönelme açısını hesaplayan iki farklı sistem önerilmiştir. Bulunan yönelme açısı bir robot kola aktarılarak taklit hareketinin gerçekleşmesi sağlanmıştır. Simulasyonlar sonucunda insan kolu yönelme açısının işaretler, fon imgesiyle ilgili bilgi veya imge özelliklerinin takibi kullanılarak bulunabileceği anlaşılmıştır.

Anahtar Kelimeler: Robot Taklidi, Bilgisayarlı Görüş, İnsan-Makine Etkileşimi

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Robot mimicking has been a hot research topic in the previous years and is an important application area for robotic systems. Robot mimicking is a convenient way to teach the robots the operations they are to perform. Instead of cumbersome programming a human may demonstrate the operation of a robot working in an assembly line, or a physically impaired person may direct a robot to accomplish movements he/she is unable to do such as lifting a heavy object. Also robots may be taught to perform actions such as playing musical instruments (e.g. drums). Another advantage, which may be obtained by robot mimicking, is the direct control of the teaching process. If the human, who is being mimicked, is observing the robot at the same time unwanted or dangerous situations may be avoided by actively changing the movements at that instance.

Although early years of robot mimicking research was based on mechanical devices such as data gloves, recent advancements in the capabilities of computers and new algorithms developed in the field of computer vision has made a true mimicking system possible, where robots mimic humans by observing them visually. That kind of mimicking is a more natural and humanoid way since human learning from early childhood begins with visually observing other humans and mimicking them.

In this thesis, a system capable of visually observing a human's arm movements and then mimicking these movements with a PUMA 760 robot arm is proposed.

1.2 Scope of the Thesis

In order to mimic a human arm, first a marker-based approach was followed. Markers were placed on a 2 degree of freedom stationary model joint and two parameters, namely yaw and bending degrees, are obtained by machine vision techniques.

After the research on the model joint, the mimicking of a real human arm without markers is studied. The system is designed to operate on a video sequence, where only one arm of a human is moving. It is also assumed that the largest moving region in the sequence is the human arm to be mimicked.

After being supplied with a background image and a stationary image of the human target, the system detects the moving parts in a video sequence by background subtraction. Then orientation analysis is conducted on the moving part (i.e. arm) in order to obtain information on the posture of the arm. This information is passed to a robot control software frame by frame, and the software directs PUMA 760 to achieve the same posture, thus realizing mimicking.

The third method studied is the tracking of human arm by using Lucas Kanade feature tracking. The arm is detected by considering the features with a movement greater than a threshold, and the movement information is used to obtain the orientation angles through a linear system solution.

1.3 Outline

Introduction is given within this chapter (Chapter 1). Chapter 2 details the concept of human gesture analysis and robot mimicking. Previous research and applications on the subject are also presented in Chapter 2.

Many different computer vision techniques and algorithms are used throughout this research. The details of the mainly used ones are given in Chapter 3 as background information.

Chapter 4 is a detailed analysis of the mimicking of a 2 degree of freedom model joint. Visual analysis of the captured images of the model joint is detailed and results obtained from the tests are presented. Also the robot arm used in this research is detailed in this chapter.

Chapter 5 is where the robot mimicking system is proposed. The visual analysis of the video sequence is detailed step-by-step and the performance of the system is evaluated.

Chapter 6 presents the mimicking system where motion information is used by a Lucas Kanade feature tracker in order to obtain the orientation angles for the upper arm and forearm.

Finally, Chapter 7 concludes the thesis and presents future work directions on the subject.

CHAPTER 2

HUMAN GESTURE ANALYSIS AND ROBOT MIMICKING

2.1 Introduction

One of the most important research fields in the development of successful robotic systems is Human-Machine Interaction area. In the early years of robotics teach pendants and programming were the primary methods for interaction [1]. In fact, mechanical devices such as mice and keyboards are still the most common Human-Computer (Machine) Interaction medium. However, as computer capabilities improve better methods are being developed. In the last several years, there has been an increased interest in adapting means of human-to-human interaction to HMI. An important non-verbal mean of interaction among humans is gestures. Gestures range from simple actions, such as pointing at objects and moving them around to more complex ones, which express human feelings and emotions [2].

The HMI interpretation of gestures requires that the dynamic and/or static configurations of the human hand, arm and sometimes the body be measurable by the machine. Early research on this problem focused on using mechanical devices, which measure human hand/arm position. However these devices required the user to carry heavy equipment, which hinders naturalness. In order to overcome the limitations of the above systems, vision based approach has been proposed in which video cameras and computer vision techniques are used to interpret gestures.

Vision based approaches are promising, since vision is the primary method of interaction among humans [2].

2.2 Human Gesture Analysis

In an HMI application, it is desired that gestures be used in order to perform tasks that mimic both the natural use of the hand as a manipulator, and its use in the human-machine communication (control of human/machine functions through gestures). In a taxonomy given in [2], hand/arm movements are classified as follows:

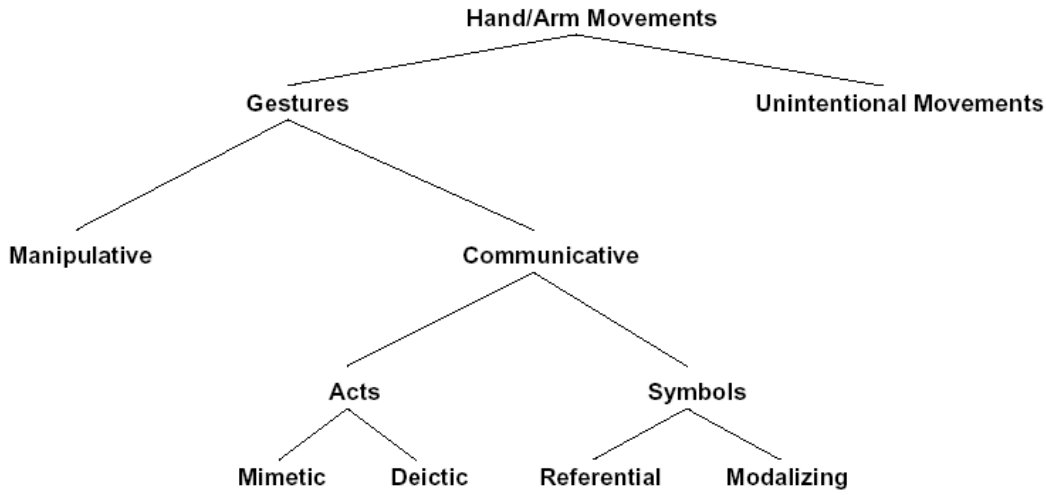


Fig. 2-1 Taxonomy of Gestures

Unintentional movements are those that do not carry any information. Gestures are two subclasses; manipulative gestures are the ones used to affect an object in the real world, such as moving a pencil. Communicative gestures, which are usually accompanied by speech, may be either acts or symbols. Symbols are gestures that have a linguistic

role. They may either symbolize some referential action or used as modalizers. Acts are gestures, which are directly related to the interpretation of the movement itself. These are either mimetic (imitating some actions) or deictic (pointing acts). This thesis is mainly focused on mimetic acts, captured by a camera.

A typical visual gesture analysis consists of three stages:

- Hand/Arm Localization and Segmentation: In this process hands/arms are extracted from the rest of the image. Since this step is a complex task, restrictions on background, user or imaging can be imposed in order to simplify the task. Skin color or motion information is widely used in this stage.
- Hand/Arm Image Feature Extraction: The features to be extracted in this step depend on the parameters, which are to be found in the next step. If a model, which requires finger trajectories, is to be used fingertip positions are extracted or for moment and contour computation image silhouettes may be extracted in this step.
- Hand/Arm Model Parameter Computation: The parameters computed in this step depend on the application. For a tracking system the position of the hand may be sufficient while for a recognition system required parameters for a following recognition step might be computed.

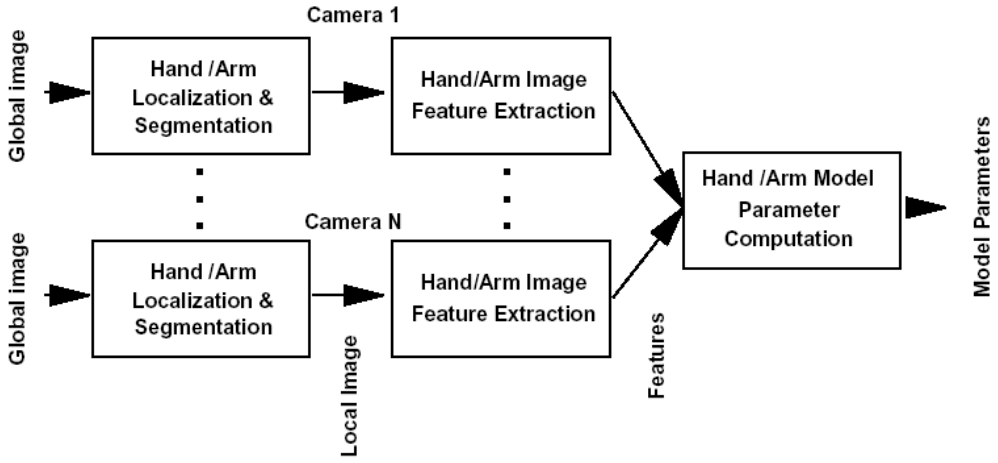


Fig. 2-2 Visual Gesture Analysis Stages

2.3 Robot Mimicking

Webster dictionary defines mimicking as "imitating closely". Hence, robot mimicking may be defined as robots imitating other robots or humans. Since mimicking is the most basic method for learning among humans, applying it to human-machine interaction is desirable in order to develop humanoid robots. Research on robot mimicking covers a large area of robotics, such as artificial intelligence, computer vision, pattern recognition and machine learning.

In order to mimic other robots or humans, a robotic system must be supplied with information defining the movement to be mimicked. For a true mimicking operation, the robot must obtain and evaluate this information by itself. Considering the constraints of mechanical devices explained before, the most natural way for mimicking may be accomplished by visually observing the target of mimicking and analyzing its movement in order to retrieve the information required for mimicking [3].

The target of mimicking may be either a human or another robot. If the target is a human, the parameters to be observed should be carefully chosen since it is impossible to totally mimic human motion. This is due to the fact that humans do not compose of articulated rigid bodies while today's robots mainly consist of articulated rigid bodies. If the target is a robot of the same architecture the mimicking is merely observation of the target's movement and applying computer vision techniques to correctly detect that movement. The target may also be a similar robot but not same kind of robot. In this case, the selection of parameters to be observed gains importance as in the human target case.

2.4 Previous Research on Gesture Analysis and Robot Mimicking

In [4], a system is proposed where human pointing gestures are used in order to control a PUMA 560 robot arm. A human, by using his/her index finger, points to one of the objects placed in the working environment and the robot arm picks that object. Then, the human points to a location within the environment and the robot places the picked object at that location. Two cameras are used to supply the visual information to the system. An artificial neural network realizes the processing of that information in order to detect the pointing human hand, and then the positions it is pointing. The system operates with an accuracy of 1±0.4 cm. in a working environment of 50x50 cm.

A system capable of mimicking the upper body of a human is presented in [5]. The system employs a 24 degree of freedom humanoid robot, which is similar to the upper part of a human body with two arms, head and torso. The system detects the head and arms of a human by using

color, aspect ratio and depth information. Then the motion of these parts is detected by using stereo vision techniques. The system also has auditory input and is able to detect sound sources. Although the system operates at real-time speed, a very advanced hardware structure is used.

In another system [6], a mobile robot is directed by using hand gestures. The system is able to detect commands such as approach, retreat, grasp, release, follow and travel. The gestures are captured by the robot's vision system and classified in order to determine the command by using a Hidden Markov Model. The system is also capable of observing a set of commands and then executing them in order, thus, realizing the robot teaching by gestures concept.

In another approach [7], a human arm tracking system is designed where the arm is modeled as two truncated right-circular cones, constructed with spherical joints. The image of the human arm and an imaginary image of the model are compared and the model joint is iteratively moved until it matches with the human arm. Although the system is successful in estimating the human arm posture, it requires the knowledge of the position of the shoulder and the parameters of the model should be manually configured.

In a research [8], focused on imitation learning, motions of a full human body are visually analyzed and learned by using Hidden Markov Models. Then same motions are realized by a virtual robot, which is a 3D human body simulator software.

Another robot mimicking system is proposed in [9], where visual input of the hand movements of a human are analyzed by using shape and color information. The system determines the trajectory of the human hand and a humanoid robot follows the same trajectory in order realize the

mimicking. Although the system operates successfully the need for using stereo vision is its drawback.

As seen from the work mentioned above, most of the research is directed to obtaining hand postures. The work directed on arm mimicking is usually accomplished by using stereo vision. In this thesis, a visual analysis system, capable of extracting human arm motions from a video sequence of monocular vision, is proposed. In order to achieve this goal, localization of a human arm should be determined by a number of machine vision algorithms.

CHAPTER 3

IMAGE ANALYSIS ALGORITHMS FOR HUMAN ARM LOCALIZATION

3.1 General

This chapter details the specific algorithms and techniques of machine vision field, which are used throughout this research and frequently mentioned in the following chapters.

3.2 Thresholding

Thresholding is a method to convert a gray scale image into a binary image so that objects of interest are separated from the background. For thresholding to be effective in object-background separation, it is necessary that the objects and background have sufficient contrast and one should know the intensity levels of either the objects or the background. In a fixed thresholding scheme, these intensity characteristics determine the value of the threshold [10].

Thresholding may be based on a single threshold or a lower and upper threshold. Let $B[i, j]$ be a gray scale image and T_1 and T_2 be threshold values, then the thresholded image $B_T[i, j]$ may be defined as follows;

$$B_T[i, j] = \begin{cases} 1 & ,if\ B[i, j] \le T_1 \\ 0 & ,otherwise \end{cases} \qquad (3\text{-}1)$$

or for a two level thresholding;

$$B_T[i,j] = \begin{cases} 1 & ,if\ T_2 \le B[i,j] \le T_1 \\ 0 & ,otherwise \end{cases} \quad (3\text{-}2)$$

The threshold values are usually selected on the basis of experience within the application domain. In some cases, the first few runs of the system may be used for interactively analyzing a scene and determining the appropriate values for threshold. There are also more robust automatic thresholding methods such as p-tile, iterative or adaptive thresholding. [10] These methods usually rely on histogram analysis of images and are applied to more general thresholding problems, such as document processing. For a brief review one should revise [11]

3.3 Connected Component Analysis and Flood Filling

A set of pixels in which each pixel is connected to all other pixels is called a connected component [10]. In a grayscale image for two pixels to be defined as connected their values should be within a certain range and there must a traceable path of pixels within this range between these two pixels.

Flood filling [12] is a recursive algorithm, which aims at detecting the connected components (i.e. regions) within an image. The algorithm may be detailed as below:

- Given a seed pixel $[i, j]$ in an image and a pixel value range, check all the unchecked neighbors of the seed pixel for their values. If the difference of value of pixel $[i, j]$ and its neighbor is within the give range mark the neighbor pixel as connected and checked.
- When all the neighbors of pixel $[i, j]$ are checked, pass to a connected pixel and repeat the previous step.
- If all the neighbors of all connected pixels are checked and there is no new pixel fit for the criteria then exit.

If all the pixels in an image, which were not found to be part of connected components in the previous runs of the algorithm, are chosen as seed pixels, then the image may be segmented into connected components.

3.4 Image Analysis by Moments

The definition of moments of the gray value-function $f(x,y)$ of an object is the following [13]:

$$m_{p,q} = \iint x^p y^q f(x,y)dxdy \tag{3-3}$$

The integration is calculated over the area of the object. Generally each other pixel-based feature instead of the gray value could be used to calculate the moments of the object.

Moments are generally denoted by the order of the moments. The order of a moment depends on the indices p and q of the moment $m_{p,q}$ and vice versa. The sum $p+q$ of the indices is the order of the moment $m_{p,q}$. Considering this, moments up to second order may be defined as:

- Zero order moment ($(p,q)=(0,0)$)

$$m_{0,0} = \iint f(x,y)dxdy \tag{3-4}$$

- First order moment ($(p,q)=(1,0)\ or\ (0,1)$)

$$m_{1,0} = \iint xf(x,y)dxdy \tag{3-5}$$

$$m_{0,1} = \iint yf(x,y)dxdy \tag{3-6}$$

- Second order moments ($(p,q)=(2,0)\ or\ (0,2)\ or\ (1,1)$)

$$m_{2,0} = \iint x^2 f(x,y)dxdy \tag{3-7}$$

$$m_{0,2} = \iint y^2 f(x,y)dxdy \tag{3-8}$$

$$m_{1,1} = \iint xyf(x,y)dxdy \tag{3-9}$$

Above definitions describe general spatial moments of the object. From the spatial moments the central moments can be derived by reducing the spatial moments with the center of gravity (x_c, y_c) of the object, so all the central moments refer to the center of gravity of the object. Expressed as formula the central moments are calculated as follows:

$$\mu_{p,q} = \iint (x - x_c)^p (y - y_c)^q f(x,y)dxdy \tag{3-10}$$

The main advantage of central moments is their invariancy to translations of the object. Therefore, they are suited well to describe the form of the object.

The moments are features of the object, which allow a geometrical reconstruction of the object. They do not have a direct understandable geometrical meaning, but usual geometrical parameters can be derived from them [13].

- The zero order moment $m_{0,0}$ is the area A of the object.

$$A = m_{0,0} \tag{3-11}$$

- The coordinates x_c and y_c of the center of gravity of the object are simply described by the first order moments $m_{1,0}$ and $m_{0,1}$ divided by the zero order moment $m_{0,0}$. (i.e. the area of the object)

$$x_c = \frac{m_{1,0}}{A} = \frac{m_{1,0}}{m_{0,0}} \tag{3-12}$$

$$y_c = \frac{m_{0,1}}{A} = \frac{m_{0,1}}{m_{0,0}} \tag{3-13}$$

- The main inertial axis could be derived by calculating the eigenvalues of the inertial tensor [13]:

$$\lambda_{1,2} = \sqrt{\frac{1}{2}(\mu_{2,0} + \mu_{0,2}) \pm \sqrt{4\mu_{1,1}^2 - (\mu_{2,0} - \mu_{0,2})^2}} \qquad (3\text{-}14)$$

The main inertial axes of the object correspond to the semi-major and semi-minor axes a and b of the image ellipse, which can be used as an approximation of the considered object. The main inertial axes are those axes, around which the object can be rotated with minimal (major semi-axis a) or maximal (minor semi-axis b) inertia [13].

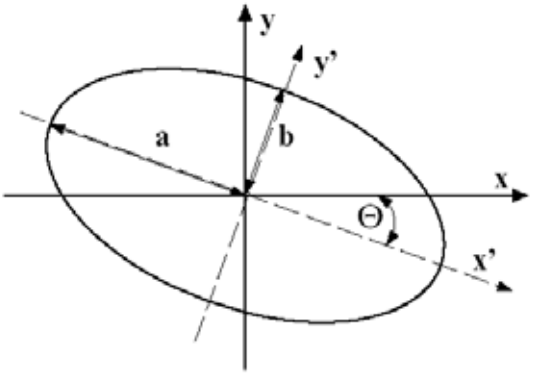


Fig. 3-1 Semi-axes and orientation

- The orientation θ of the object is defined as the tilt angle between the x-axis and the axis, around which the object can be rotated with minimal inertia (i.e. the direction of the major semi-axis a). This corresponds to the eigenvector with minimal eigenvalue. In this direction the object has its largest extension. It is calculated as follows:

$$\theta = \frac{1}{2}\tan^{-1}\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}} \qquad (3\text{-}15)$$

There is an ambiguity in the tilt angle θ of the object, which can be resolved by choosing θ always to be the angle between the x-axis and the semi major axis a (i.e. by definition $a \geq b$). Secondly, the principal value of the arc tangent is chosen such that $-\frac{\pi}{2} \leq \tan^{-1} x \leq \frac{\pi}{2}$ [13].

A summarized tabulation for orientation is given in Table 3-1.

Table 3-1 Summary of Orientation Calculation

$\mu_{2,0} - \mu_{0,2}$	$\mu_{1,1}$	θ	
Zero	Zero	0^o	
Zero	Positive	$+45^o$	
Zero	Negative	-45^o	
Positive	Zero	0^o	
Negative	Zero	-90^o	
Positive	Positive	$\frac{1}{2}\tan^{-1}\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}$	$0 < \theta < 45^o$
Positive	Negative	$\frac{1}{2}\tan^{-1}\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}$	$-45^o < \theta < 0$
Negative	Positive	$\frac{1}{2}\tan^{-1}\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}} + 90^o$	$45^o < \theta < 90$
Negative	Negative	$\frac{1}{2}\tan^{-1}\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}} - 90^o$	$-90^o < \theta < -45^o$

3.5 Morphological Operations

Mathematical morphology gets its name from the study of shape. A morphological approach facilitates shape-based or iconic solutions to computer vision problems. There are four basic morphological operators, which are normally applied to binary images [10].

- The intersection of two binary images A and B, written as $A \cap B$, is the binary image which is 1 at all pixels p which are 1 in both A and B. Thus,

$$A \cap B = \{p \mid p \in A \, and \; p \in B\} \tag{3-16}$$

- The union of A and B, written $A \cup B$, is the binary image, which is 1 at all pixels p which are 1 in A or 1 in B (or 1 in both). Thus,

$$A \cup B = \{p \mid p \in A \, or \; p \in B\} \tag{3-17}$$

- The complement of A is the binary image, which interchanges the 1s and 0s in A. If Ω is a universal binary image (all 1), then

$$\overline{A} = \{p \mid p \in \Omega \, and \; p \notin A\} \tag{3-18}$$

- The translation of binary image A by pixel p is given by

$$A_p = \{a + p \mid a \in A\} \tag{3-19}$$

By using the above operators other morphological operations may be constructed, such as dilation and erosion [10]:

- Dilation: Translation of a binary image A by a pixel p shifts the origin of A to p. If $A_{b_1}, A_{b_2}, ..., A_{b_n}$ are translations of the binary image A by the 1 pixels of the binary image $B = \{b_1, b_2, ..., b_n\}$, then the union of the translations of A by the 1 pixels of B is called dilation of A by B and is given by

$$A \oplus B = \bigcup_{b_i \in B} A_{b_i} \tag{3-20}$$

- Erosion: The opposite of dilation is erosion. The erosion of a binary image A by a binary image B is 1 at a pixel p if and only if every 1 pixel in the translation of B to p is also 1 in A. Erosion is given by

$$A \Theta B = \{p \mid B_p \subseteq A\} \tag{3-21}$$

Often the binary image B is a regular shape, which is used as a probe on image A and is referred to as a structuring element [10].

The basic operations of morphology can be combined into complex sequences. For example, an erosion followed by a dilation with the same structuring element (probe) will remove all of the pixels in regions, which are too small to contain the probe, and it will leave the rest. This sequence is called *opening* [10]. The opposite sequence, a dilation followed by an erosion, will fill in holes and concavities smaller than the probe. This is referred to as *closing* [10].

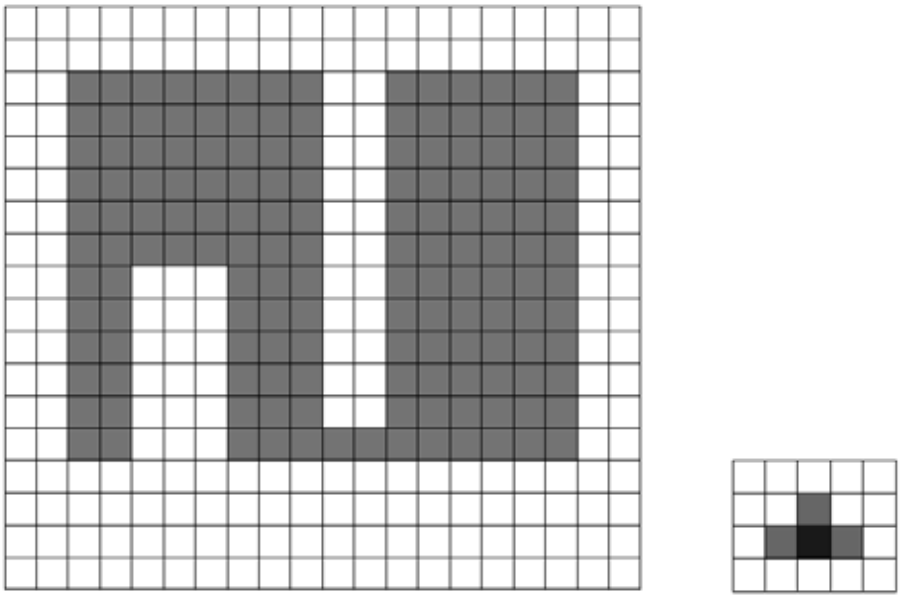


Fig. 3-2 A test image A (left) and a structuring element B (right). The origin of the structuring element is darker than other pixels.

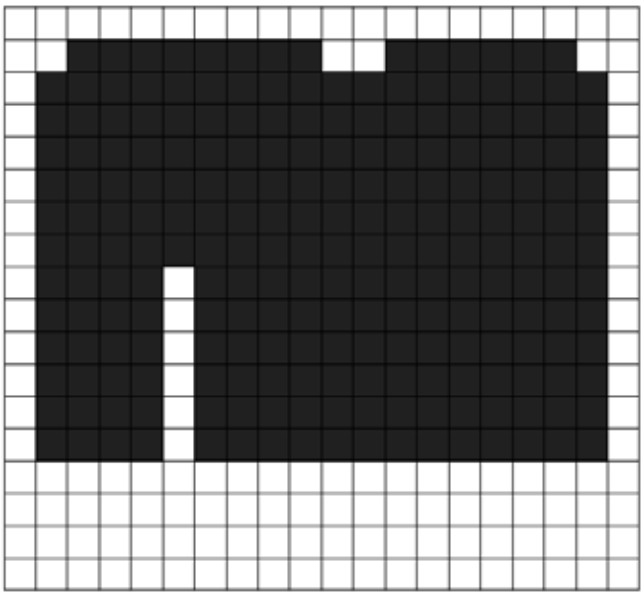


Fig. 3-3 The dilation of A by B

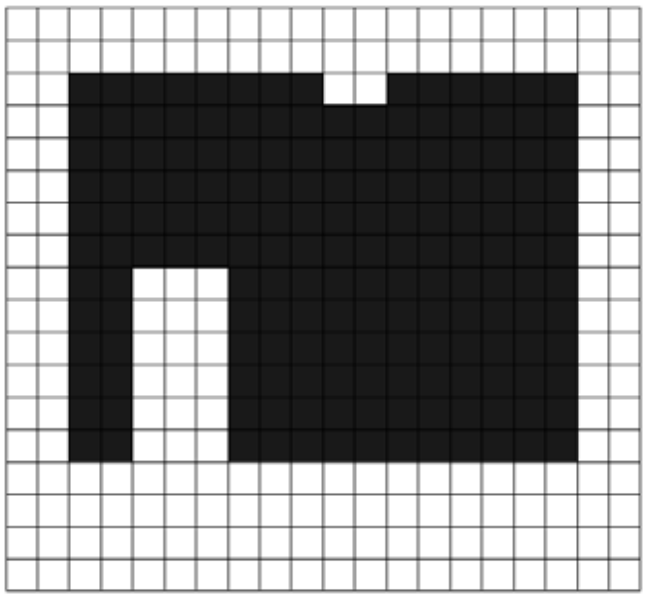


Fig. 3-4 Closing of A by B (Note that, this is erosion applied to Fig. 3-3)

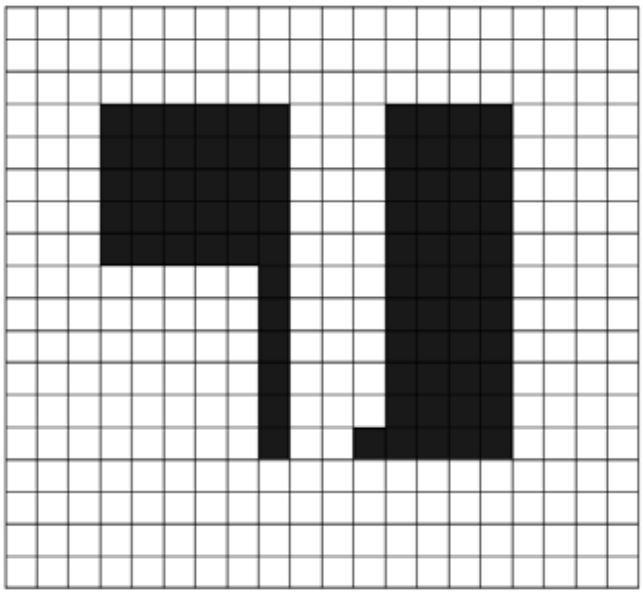


Fig. 3-5 The erosion of A by B

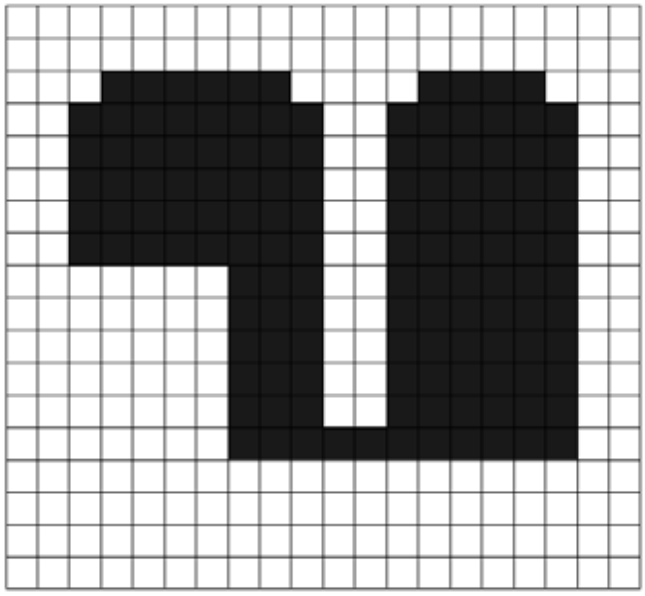


Fig. 3-6 Opening of A by B (Note that, this is dilation applied to Fig.3-5)

3.6 Convex Hull Analysis

In order to define a convex hull, first of all, terms, such as polygon and convex should be defined. A polygon is a closed path of straight line segments [14]. These segments are also called edges of the polygon, and the intersection of two adjacent edges is a vertex of the polygon. Thus, every polygon with n vertices has n edges. A polygon, which has no intersecting non-adjacent edges, is called a simple polygon and a simple polygon is convex if the internal angle formed at each vertex is smaller than 180^{o}. Based on these definitions, the convex hull of a polygon P is the smallest-area simple convex polygon, which encloses P [14].

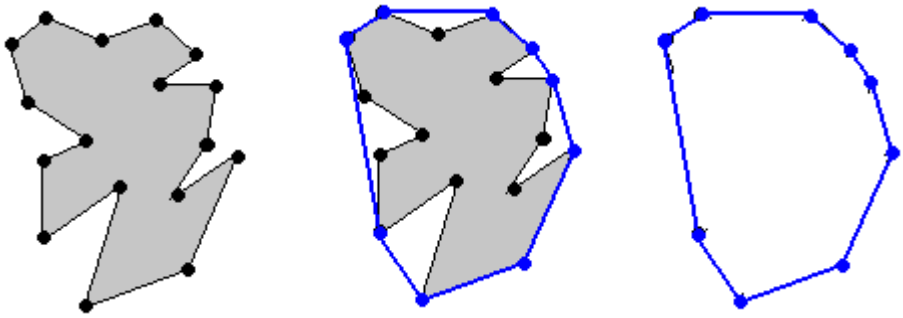


Fig. 3-7 From left to right a simple polygon, convex hull of that polygon and the convex polygon.

Different algorithms have been developed since 1970's in order to find the convex hulls of polygons [14]. One of them, which is used in this research, is the 3-Coins Algorithm or Sklansky's Scan proposed in [15].

Sklansky's Scan (for a n-vertex simple polygon)

- Find a convex vertex and label it p_0. (A vertex p_i is convex if a right turn is made at p_i while going from p_{i-1} to p_{i+1} where the interior of the simple polygon P is to the right)
- Label the remaining n-1 vertices in a clockwise order, starting at p_0.
- Place three coins on vertices p_0, p_1, p_2 and label them "back", "center", and "front" respectively.
- Do:

  If the 3 coins form a right turn (or if the 3 coins lie on collinear vertices),

  - Take "back", place it on the vertex ahead of "front".
  - Relabel: "back" becomes "front", "front" becomes "center", "center" becomes "back".

  Else (the 3 coins form a left hand turn)

  - Take "center", place it on the vertex behind "back".
  - Remove (or ignore hereafter) the vertex (and associated edges) that "center" was on.
  - Relabel: "center" becomes "back", "back" becomes "center".

  Until "front" is on vertex p_0 (starting vertex) and the 3 coins form a right turn.
- The remaining vertices and edges form the convex hull of the original simple n-polygon.

3.7 Median Filtering

Median filters replace each pixel value with the median of the values in the local neighborhood [10]. They are very effective in removing salt and pepper and impulsive noise while preserving edges because the median operation easily rejects outliers, avoiding blurring across edges. In a median filter, a window slides along the image, and the median intensity value of the pixels within the window becomes the output intensity of the pixel being processed [16].

For example, if a 3-by-3 median filter is applied to the window given in Fig. 3-8, the output of the filter will be 37. The pixel being processed, which has an intensity value of 20, will be set to 37.

5	24	68
41	20	92
12	55	37

Fig. 3-8 Median Filtering Example

3.8 Intel's Open Source Computer Vision Library

During the implementation of the computer vision algorithms mentioned above Intel's Open Source Computer Vision Library (OpenCV)[17] is used. OpenCV is a C/C++ library, still in the development phase designed, specifically for computer vision applications. During its early

development it was based on Intel's Image Processing Library (IPL). However, new versions of OpenCV are distributed as a standalone library.

OpenCV offers a variety of functions applicable to image processing and computer vision research and applications. These functions are summarized in 8 groups:

- Basic Structures and Operations
- Image Processing and Analysis
- Structural Analysis
- Motion Analysis and Object Tracking
- Object Recognition
- Camera Calibration and 3D Reconstruction
- Experimental Functionality
- GUI and Video Acquisition

The functions from these groups may be used in many areas from image handling to camera calibration. During this research functions especially from the groups Image Processing and Analysis (`cvFloodFill()`, `cvGetSpatialMoment()`),Structural Analysis (`cvConvexHull2()`), Motion Analysis and Object Tracking (`cvCalcOpticalFlowPyrLK()`), and GUI and Video Acquisition are used quite decreasing the coding burden of such complex algorithms.

CHAPTER 4

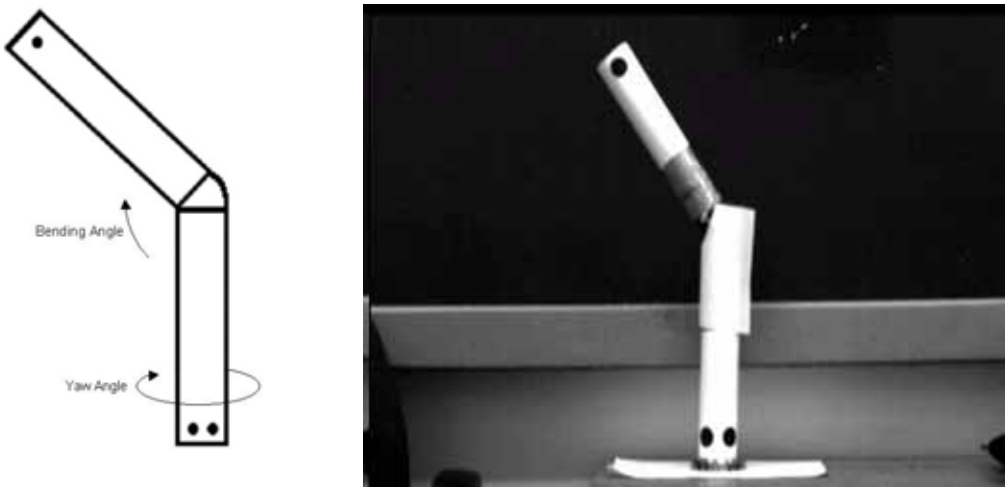
MODEL ARM ORIENTATION ESTIMATION WITH MARKERS

4.1 Motivation

Before advancing to a real human arm mimicking, the mimicking of a model joint with markers to show the orientation was implemented in order to make an introduction to the mimicking concept with such a constrained case. The visual analysis is applied on the captured image of a model joint and the angles of bending and yaw are determined.

4.2 Visual Analysis

Visual processing of captured images is used to determine the bending and yaw angles of the model joint. The model joint consists of two equal length cylindrical parts (Fig. 4-1). In this model, the lower joint can rotate around the base and an upper one can bend. Thus, one has two parameters for this model joint: a yaw angle and a bending angle. In order to be able to determine these two angles by using a visual capture system, two black circular markers are placed at the bottom side of the lower part and one black marker is placed on the upper side of the top part (Fig. 4-1).



(a) (b)

Fig. 4-1 The Model Joint (a) and its image (b)

4.3 Image Processing for Determining Bending and Yaw Angles

In order to determine the bending and yaw angles of the model joint, first an image of the model joint (Fig. 4-1 (b)) is captured by using a camera (COHU 4710 model B/W solid state). The captured image is then passed to an application software (written in Visual C++) (Fig. 4-2). This software uses Intel's Open Source Computer Vision Library (OpenCV) [17] for image processing and machine vision operations. The user can select a previously captured image file or capture a new image by using the program. All the parameters of the algorithm are adjustable through the interface.

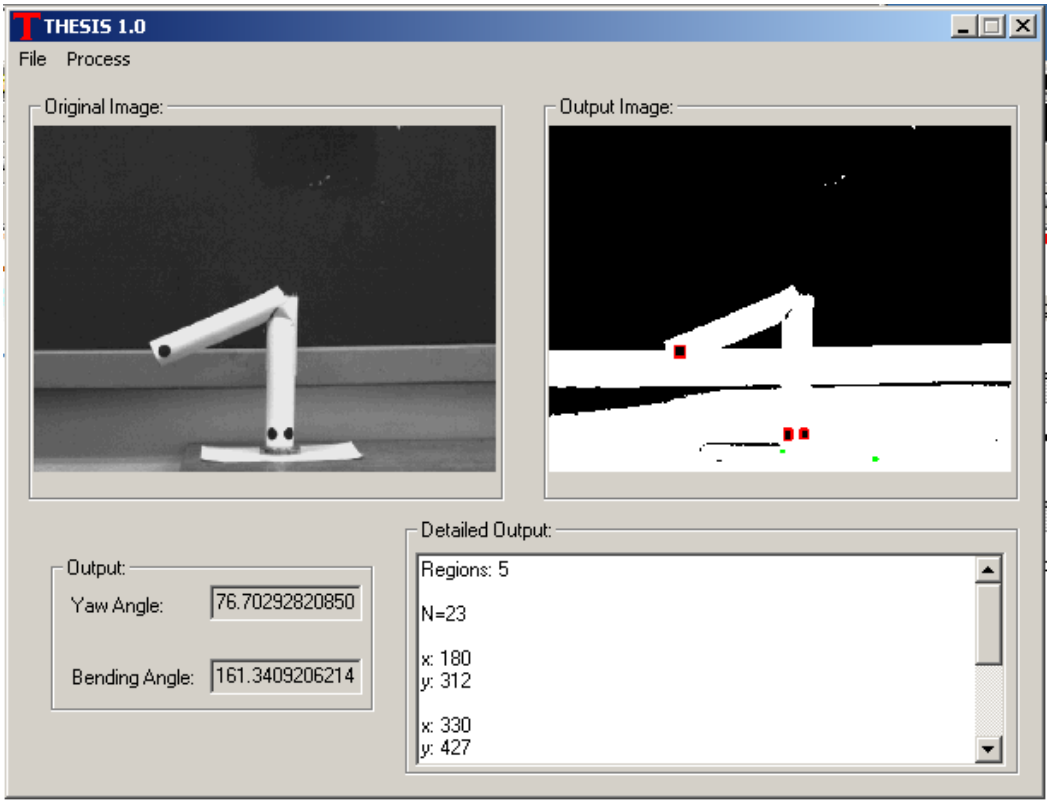


Fig. 4-2 The program interface

4.3.1 Extraction of Marker Points

The major steps in the visual processing stage can be summarized as below:

1. A median filter (5-by-5) is applied to the image in order to remove camera and environment noise. A median filter is chosen for noise removal, since a boxcar or Gaussian filter blurs the image, which leads to the detection of dots to be more difficult.

2. Thresholding is applied, based on the pixel values of the image. Although, there are more sophisticated automatic thresholding methods, the threshold levels in the simulations are chosen

manually after inspecting the environmental effects, such as illumination and background. During the entire experiments, the values for minimum and maximum threshold levels to detect black markers are set to 20 and 80, respectively.

3. Thresholding is followed by a connected component analysis, in order to determine the connected components within the thresholded image. The connected component analysis is realized by using 4-connectedness and a pixel value range of –10 to +10 is chosen to determine the homogeneity of neighboring pixels relative to the seed pixel.

4. Finally, some of the determined connected components are eliminated by some heuristics in order to determine the set of candidate connected components representing 3 markers on the model joint. Elimination heuristics are

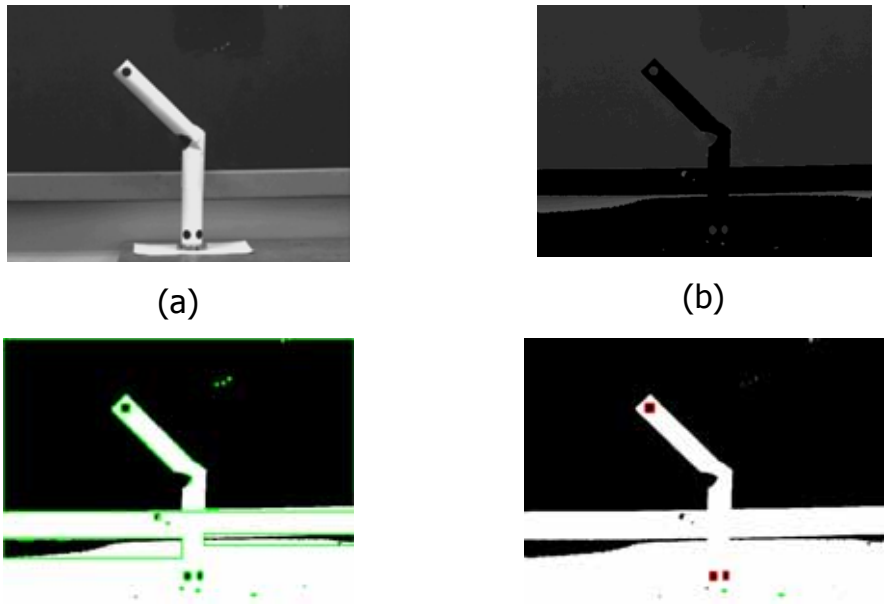
    - Area of the connected component for an upper and a lower limit,
    - Ratio of the width of the bounding box of the connected component to the height of the bounding box,
    - Ratio of the area of the connected component to the area of the bounding box (only a lower limit is utilized).

    For a 320x240-pixel image, typical values of these constraints are selected as

    - Area limits of the connected component = 10 (min), 200 (max)

- Ratio of the width of the bounding box of the connected component to the height of the bounding box = 3 (max), 0.2 (min)
- Ratio of the area of the connected component to the area of the bounding box = 0.7 (min)

From the set of candidate connected components, 3 of them having the largest area are selected as the markers on the model joint. The centers of the bounding boxes of these 3 connected components are assumed to be the centers of the dots. From these 3 dots, the marker having the smallest y-value (origin is the upper-left corner of the image) is chosen as the dot on the upper part of the model joint and the other two dots are chosen as the ones on the lower part of the model joint.



(a) (b)

(c) (d)

Fig. 4-3 Steps for Visual Analysis (a) Input Image, (b) After Thresholding, (c) After connected-component analysis (all the connected components found are bounded with green boxes.), (d) The determined markers (The regions with the largest area are chosen as markers and are bounded with red boxes.)

4.3.2 Determining Yaw and Bending Angles

After the coordinates of the marker are determined, yaw and bending angles of the model joint are determined by using the algorithm in [18]. In this algorithm, first the yaw angle is calculated. Referring to Fig. 4-4 the variable N represents the distance between the lower markers at a zero degree of yaw.

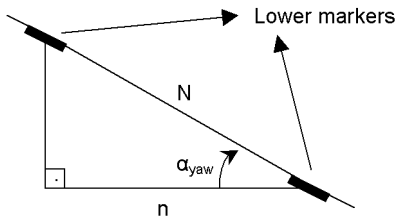
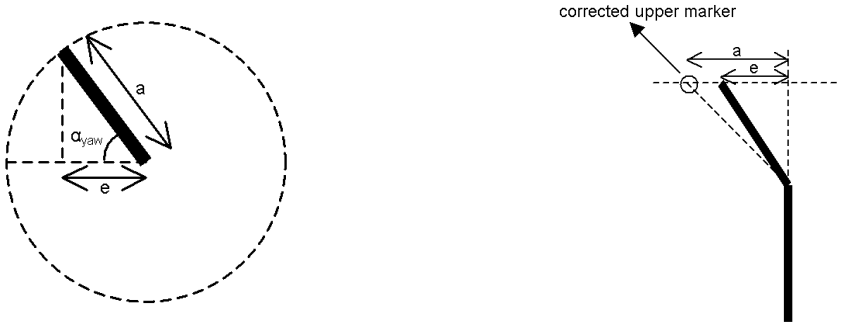


Fig. 4-4 Calculation of Yaw Degree

Forming a right triangle one can show that

$$\alpha_{yaw} = \cos^{-1}(n/N) \qquad (4\text{-}1)$$

where n is the distance between the observed markers at a given yaw angle. Next, the bending angle is determined. Since the image of the model joint will appear skewed for non-zero yaw angles, the bending angle will appear slightly greater than the actual value.



(a) (b)

Fig. 4-5 Adjusting Image due to Yaw (a) Top view, (b) Side view

Referring to Fig. 4-5, this error can be corrected by

$$a = e / \cos(\alpha_{yaw}) \tag{4-2}$$

After the error correction the actual bending angle can be calculated by using the following set of equations. (Fig. 4-6)

$$\alpha_{bend} = 180 - 2\theta \tag{4-3}$$

$$\theta = \cos^{-1}(d / c) \tag{4-4}$$

$$c^2 = a^2 + d^2 \tag{4-5}$$

where a is found previously and d is calculated by using the coordinates of the markers determined by image processing.

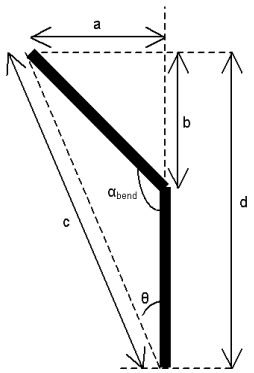


Fig. 4-6 Calculation of Bending Degree

4.4 Results

The simulations are conducted in two phases. In the first of experiments, the visual information is tested for its performance against angle calculation. In the second part of these experiments, the results of the first part are fed to PUMA 760 robot arm for testing the response for the given inputs. Before presenting the results, a brief overview for this robot arm is given next.

4.4.1 PUMA 760

Throughout this research a PUMA (Programmable Universal Machine for Assembly) 760 robot is used for application. PUMA 760 is a model from the PUMA 700 series, which is one of the three main series of PUMA robots with PUMA 200 and PUMA 500 [19]. PUMA 760 consists of two separate parts; the robot arm and the controller computer [20]. Specifications can be found in Appendix A.

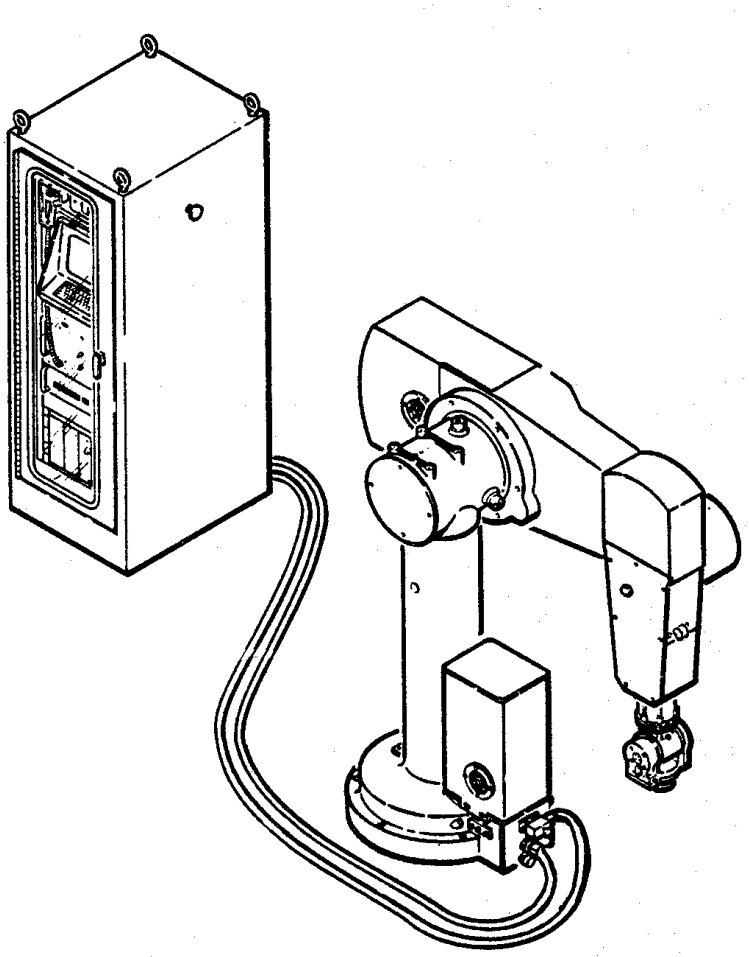


Fig. 4-7 PUMA 760 Robot

4.4.1.1 Controller Computer

The controller computer is a Unimation Mark II controller, which is a typical industrial robot controller. It is the master component of the electrical system. All signals to and from the robot pass through the controller and are used by it to perform real-time calculations to control arm movement and position. Operating controls and indicators are located on the front and top panel of the controller. Connections for the robot arm, terminal, floppy disk drive and accessories are located on the controller rear panel. Software is stored in the computer memory located

in the controller. The software interprets the operating instructions for the robot arm, and the controller transmits these instructions to the arm. From incremental encoders and potentiometers in the robot arm, the controller/computer receives data about arm position. This provides a closed loop control of arm motions. A floppy disk drive is available to record the programs on diskettes [19].

4.4.1.2 Robot Arm

The robot arm, which is connected to the controller, is a mechanical chain of links and joints incorporating 6 degrees of freedom (DOF). A DC servomotor controls each joint. It is similar to a human torso, shoulder, arm, and wrist. The components of the robot arm are the trunk, shoulder, upper arm, forearm, wrist, and mounting flange. The robot arm members contain the various servomotors and gear trains.

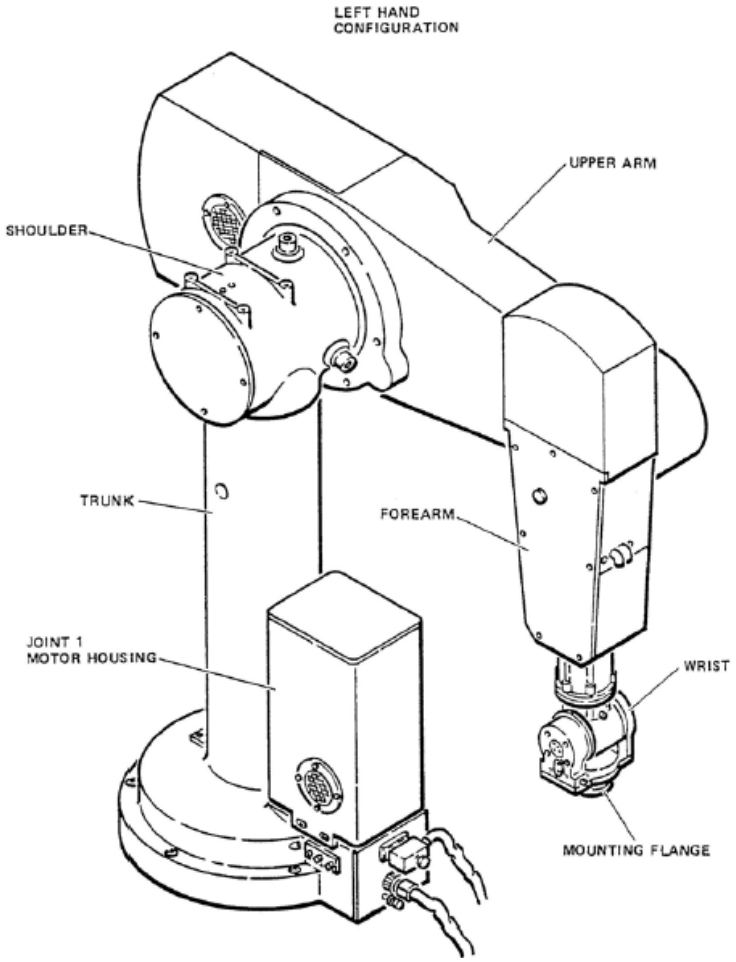


Fig.4-8 The Robot Arm

Each joint is driven by a permanent magnet DC servomotor through an associated gear train, and each motor contains an incremental encoder and a potentiometer driven through a gear reduction. The position at each joint is measured relative to an initially known absolute position. The potentiometers, incorporated in the motors are used to determine this initial absolute position. The incremental encoder mounted on the shaft of each motor provides information about the changes in position involved with the respective joint. The joint velocity is, however, computationally derived from the positional changes. Joint encoder

readings are sampled at every 28 msec. And compared with the calculated positions [19].

Power for the motors is supplied through the cable connecting the robot arm and the controller. This cable bundle, containing the data cables, also carries feedback signals from the incremental encoders and potentiometers.

Table 4-1 Joint Limits and Angular Resolutions

Joint number	Joint Limits	Joint Angular Resolution
Joint 1	320 degrees	0.0050 degrees (min)
Joint 2	220 degrees	0.0035 degrees (min)
Joint 3	270 degrees	0.0092 degrees (min)
Joint 4	532 degrees	0.0082 degrees (min)
Joint 5	200 degrees	0.0080 degrees (min)
Joint 6	532 degrees	0.0111 degrees (min)

As a last element in this mechanical chain structure a pneumatic-controlled hand (gripper) takes place. This hand has two stable states: open or close. A stand-alone air compressor supplies the compressed air necessary to open and close the two fingers of the gripper.

In this part of the thesis, yaw angle is realized by the 1st joint and bending angle is realized by the 2nd joint, since it is assumed that the 1st joint is identical to the rotation of the model joint (i.e. yaw angle) and the 2nd joint is identical to the bending of the model joint (i.e. bending angle). The 3rd joint of PUMA is kept with the same alignment with that of 2nd in order to better visualize the mimicking.

4.4.1.3 The Operating System

The system software that controls the PUMA robot arm is called VAL, which is supplied by Unimation to be used with its controller computer. The software is a sophisticated programming language and a complete robot control system, but has disadvantages mostly due to age of the product. VAL is stored in the controller computer memory. The controller also houses operating controls for the robot system. The VAL programming language consists of a full set of English language instructions for teaching and editing. Work programs are entered into the computer/controller using either of two different procedures, or a combination of both. The programs can be entered with a teach pendant using the teach-by-showing method, or using the keyboard inputs. Full programming versatility can only be achieved through keyboard inputs [19].

4.4.1.4 External Control of PUMA 760

The controller computer has two ports, which are used to communicate with external computers. These are the Alter port and the Supervisory port. Supervisory communication interface is normally used to send information or commands to the controller system. If, however, the command or information, sent in real-time, is to modify the path of the robot while it is moving then the interfacing must be done using the alter port, not through the supervisory port. The communication through alter port is basically for real-time path control.

Throughout this research a program developed in [19] is used in order to control the robot. The program is coded by using MATLAB and uses the supervisory port of PUMA 760. Positions to be taken by the robot may be given point by point, using the interface or as a sequence of VAL

commands assembled previously. The program is also able to simulate the movement of PUMA 760 through its interface without actually operating the robot. The control program interface is given in Fig. 4-9.

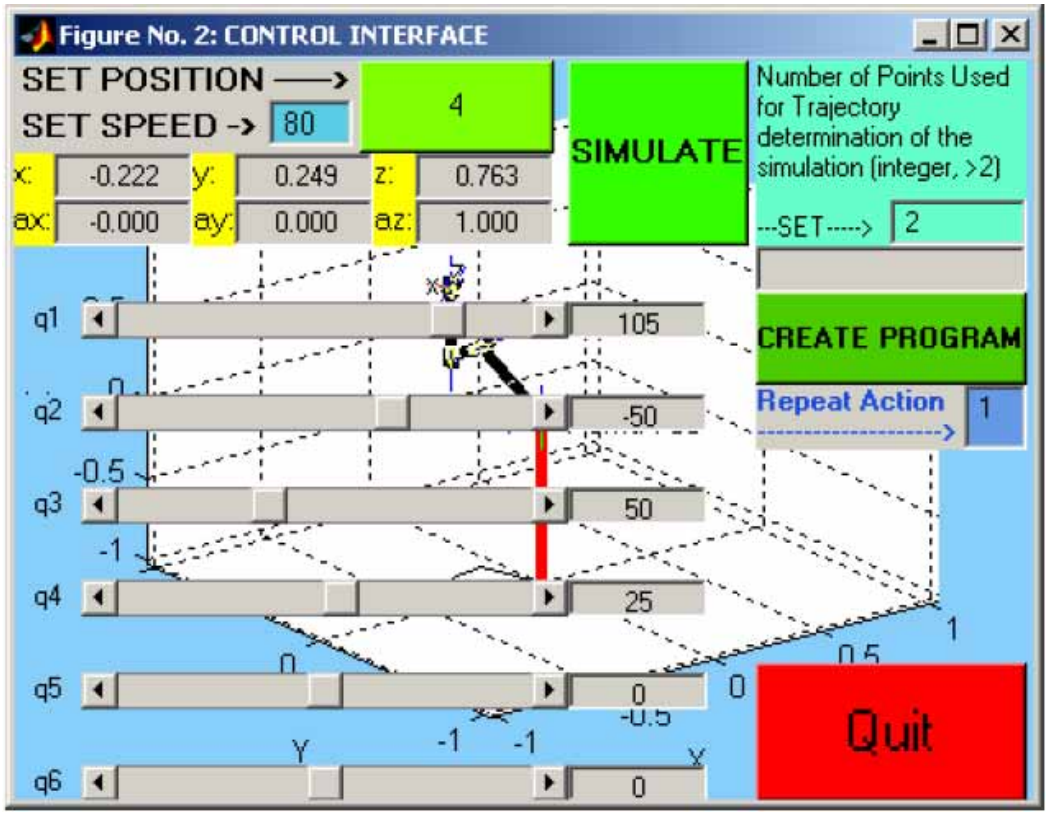


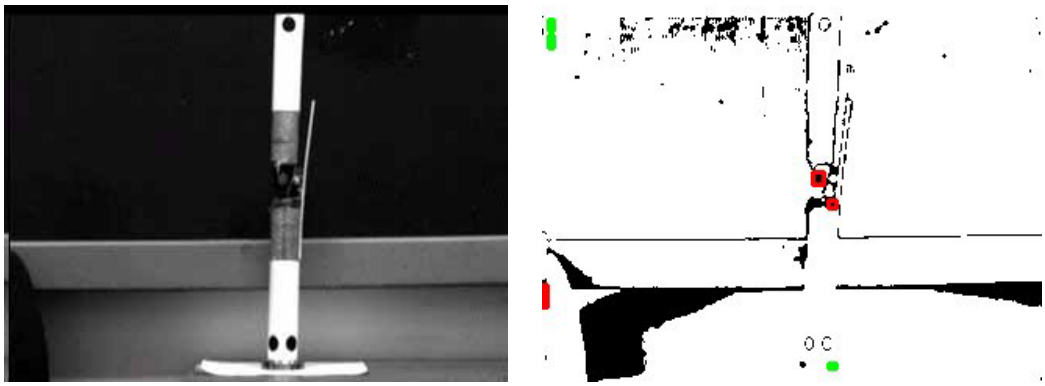
Fig. 4-9 Control Program Interface

4.4.2 Simulations on Determination of Visual Angles

The yaw and bending angles are determined successfully with acceptable errors. After simulations on 25 different images with various yaw and bending angles, the average errors are determined as 1.2 degrees for the yaw and 3.3 degrees for the bending angle. For the same set of

experiments, the maximum errors are measured as 4.5 degrees for the yaw angle and 5.2 degrees for the bending angle. The results are satisfying considering the previous work, where the average error for bending angle is 5 degrees and the average error for yaw angle is 6 degrees [18].

Although the results are successful, it is also observed that the system is susceptible to the changes in the background and illumination. For a case where the algorithm fails with different illumination conditions see Fig. 4-10. Further research might be devoted to make this algorithm robust against illumination and background changes. Another problem is the limitation on yaw angle due to the loss of visibility of the lower markers. If the yaw angle is more than 35º the marker in the side of turning is not visible. Thus, the yaw and bending angles cannot be determined. However, utilization of markers on human arm basically makes this algorithm unfavorable for daily applications.



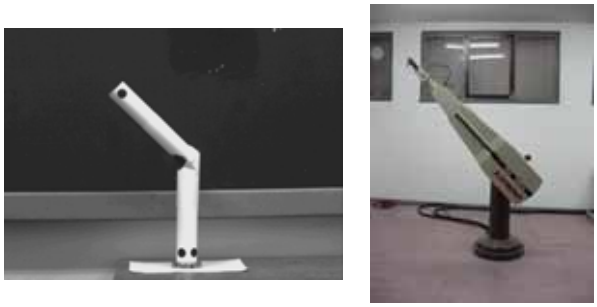
(a) (b)

Fig. 4-10 Failure with a different illumination (a) input image (b) output image

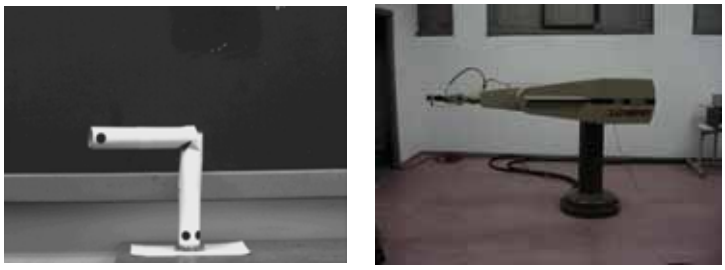
4.4.3 Simulations of Mimicking the Model Joint

After determining the yaw and bending angles correctly, the mimicking step is performed on PUMA 760 robot arm. The mimicking gives satisfying results, as can be seen from some typical results in Fig. 4-11.

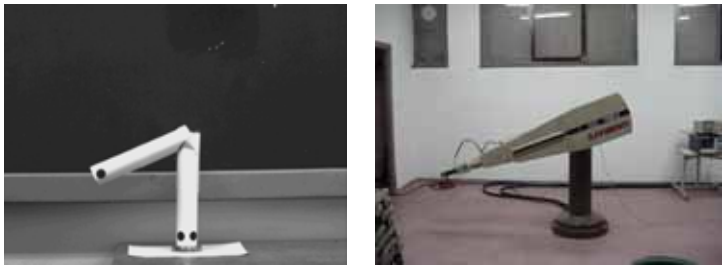
The system works offline so it is possible to process a set of input images and record the resulting yaw and bending angles. Then, these angles may be passed to the robot arm, resulting in not a standstill mimicking but a continuous mimicking motion.



(a)



(b)



(c)

Fig. 4-11 Example Results for Mimicking with (a) 0º yaw, 44º bending (b) 17º yaw, 88º bending (c) 0º yaw, 108º bending

CHAPTER 5

HUMAN ARM ORIENTATION DETERMINATION BY BACKGROUND SUBTRACTION

5.1 Introduction

The method proposed in the previous chapter has certain limitations. First of all, the method requires the use of markers, which limits naturalness. Since naturalness is a basic need for an ideal robot mimicking system this is a major drawback of that algorithm. Also the number of postures, which may be mimicked, is limited by the visibility of markers. As mentioned in section 4.4.2, certain postures with high yaw degrees may not be mimicked since the markers are not visible.

In this chapter a more natural algorithm is proposed. Although this algorithm has also limitations, it does not require the usage of markers and is directly applicable to a real moving human arm. The basic assumption of the proposed algorithm is that the largest moving region in the processed video sequence is the human arm to be mimicked. Also the background image of the operation environment and the human at a standstill are required. The limitations on the possible arm movements are given in section 5.3.

5.2 The Algorithm

In order to detect the human arm in a video sequence, it is assumed that the largest moving object in the sequence is the human arm to be mimicked. The implemented algorithm tries to find the largest moving

object in the sequence by background subtraction and defines it as the human arm. Later, an orientation analysis is conducted in order to find the angles, which are to be used as parameters for mimicking. The algorithm is detailed in the following subsections.

5.2.1 Background Estimation

The background of the operation scene is determined, when there are no humans in the foreground. This is accomplished by averaging the intensities of the frames of the video sequence (for 45 frames). The average is denoted as $mask_1$.

Fig. 5-1 $Mask_1$

5.2.2 Human Body Extraction

The human, whose arm is to be mimicked, should wait in front of the camera (for a duration of 60 frames) before moving his arm. During this time the frames are again averaged as in the previous step, in order to

obtain the human body at a standstill position (See Fig. 5-2). At the next step, the difference of this average image and $mask_1$ is obtained. For every pixel in the difference image, if the pixel value is greater than a threshold that pixel is marked. The mask obtained which is the human body standing in front of the camera is named as $mask_2$. Finally, another image of the human whose arm is to be mimicked is taken in which the arm is at the same plane with the body (i.e. 0 yaw degree), whereas it should be bended away from the body (Fig. 5-3). This image is used in order to find the length of the arm by differencing it with $mask_2$ and obtaining the semi-major axis, in the later stages. These are the only obligatory stages the user should perform before utilizing this system.

Fig. 5-2 Human waiting in front of camera

Fig. 5-3 Arm Length Extraction

5.2.3 Moving Part Extraction

The mask obtained in section 5.2.2 and the image obtained in chapter 5.2.1 is used in order to detect the moving parts in the video sequence (after the 60^{th} frame). The difference of the frame and $mask_1$ is obtained (starting at 61^{st} frame) and thresholding is applied similar to step 5.2.2. The resulting image is the human body probably with a different arm posture than $mask_2$ (Fig. 5-4). Then difference of that image and $mask_2$ is obtained and thresholding is applied in order to obtain an image where only the moved body parts (e.g. arm) are visible. This two-step differencing is conducted in order to better extract the moved parts.

Fig. 5-4 Moved from a standstill position

5.2.4 Removal of Erroneous Segmentation Regions

Region filling is applied to the resultant image from section 5.2.3 in order to obtain the connected components in the image. Among these components the one with the largest area is assumed to be moving arm. The other components are eliminated and morphological closing operation is applied to better outline the moving arm.

Fig. 5-5 Arm detected as the largest connected component

5.2.5 Contour Detection and Convex Hull Analysis

Contour of the arm in the image obtained in the previous step is determined. The contour is required, since the convex hull is defined on a set of points (i.e. a polygon) and the contour (i.e. outline) of the arm constitutes a set points. After the contour is detected the convex hull of the arm is determined by using Sklansky's Scan (Section 3.6). The largest distance of the convex hull defect in the image is assumed to be at the interior of the elbow. Hence, by finding this point the elbow is obtained. If that distance is smaller than a certain threshold (which is %30 of the semi-major axis) the arm is assumed to be straight otherwise it is assumed to be bended.

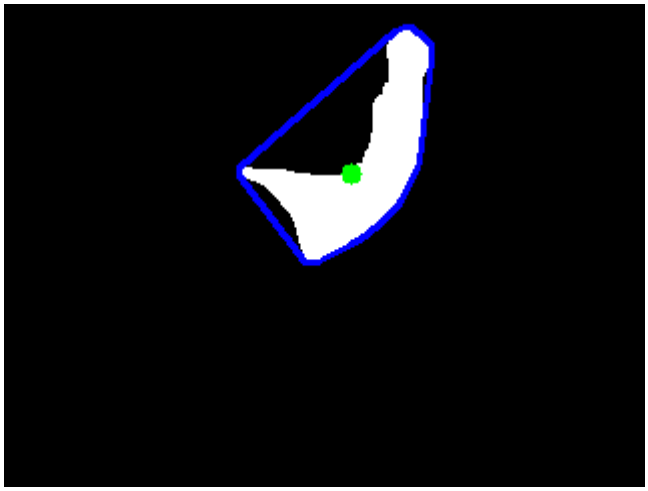


Fig. 5-6 Convex hull analysis (elbow found)

5.2.6 Orientation Analysis

If the arm is found to be straight in the previous step, a single orientation analysis is enough in order to retrieve the angle between the arm and the body. The bending angle of the elbow is assumed to be zero in that case. The yaw angle, θ_{yaw} of the arm is calculated by comparing the semi-major axis length a to the length b found in Section 5.2.2 by using the following equation:

$$\theta = \cos^{-1}\left(\frac{a}{b}\right) \tag{5-1}$$

as illustrated by the top view of a human in Fig. 5-7.

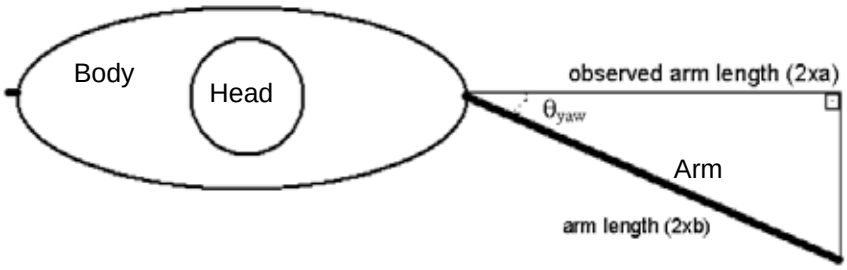


Fig. 5-7 Calculation of yaw angle

On the other hand, if the arm is determined as bending, an orientation analysis is conducted. This time, the arm is divided into two parts by a line, which is perpendicular to the major axis of orientation and passing through elbow point found in the previous step. Thus, the forearm and the upper arm are detected. Then, separate orientation analyses are conducted on both of these parts to obtain the two mimicking parameters, which are the orientation angles of the upper arm and the forearm. A typical result is shown in Fig. 5-8 for the image in Fig. 5-4.

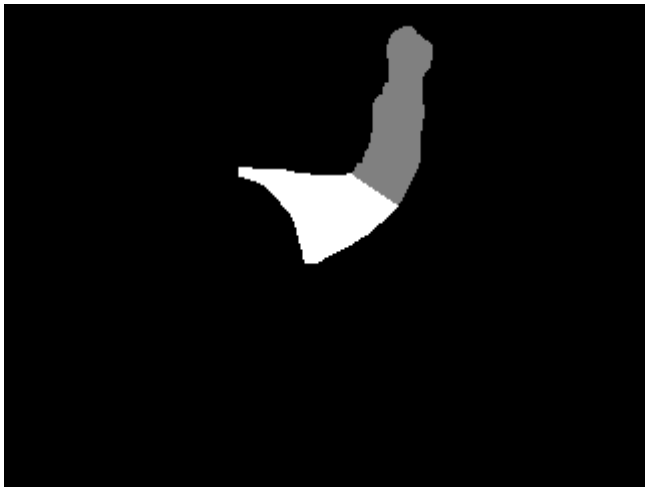


Fig. 5-8 Arm partitioned

5.2.7 Determination of the Shoulder Position

After the orientation of the two segmented arm parts is found, the region corresponding to the upper arm or forearm is not known. In order to overcome this problem the approximate position of the shoulder is determined and the region nearer to the shoulder is assumed to be the upper arm. The approximate shoulder position is found by using the difference image of the human at standstill and the background. On this image, the shoulders are assumed to be at the 3/5 of the height and 1/6 and 5/6 of the width of the bounding rectangle as seen in Fig. 5-9. Among these two shoulder points the one nearer to the center of the image is assumed to be the shoulder of the moving arm.

Fig. 5-9 Determination of the shoulder position

5.3 Performance of the Algorithm

It is observed that the detection of arm by background subtraction yields satisfying results. The main challenges faced during the tests are the detection of bending of the arm at the elbow and the correct separation of the upper arm with the forearm. If the detection and separation is accomplished successfully, the determination of the orientation angles turns out to be a simple application of a well-known moment analysis.

Two example output sets are given in tables 5-1 and 5-2 where the orientation angles are given for the upper arm and the forearm. The error plots are also given in Fig. 5-10 and 5-11 for these two sets. The results at Table 5-1 are over 32 frames and the average error is found to be 11.675. The results at Table 5-2 are over 12 frames and the average error is found to be 4.746. If the two sets are considered together the average error is found to be 9.785, which yields a 5.4% error.

Comparing the above result with similar previous research, it is seen that satisfactory error rates are obtained. For example, the results of the work of [7] states that a 7.5% error is obtained. Thus, the proposed algorithm has a lower error while it is more robust, not requiring as much a priori information as the system in [7].

Although the error rates are satisfactory, problems are faced at various steps of the algorithm. These problems are faced mainly at the separation of the upper arm and the forearm. In the first test sequence, 15^{th} frame constitutes an example of a separation problem (Fig. 5-10). At this frame, the upper arm is not segmented correctly yielding the high error rate.

Table 5-1 Calculated angles and error for the first sequence

Frame	Orientation Angles	Calculated Angles	Error
1	-41 -41	-43.1 -43.1	2.1 2.1
2	8 8	5.7 5.7	2.3 2.3
3	0 90	-13.9 88.3	13.9 1.7
4	0 -51	31.2 -70.2	31.2 19.2
5	-5 60	-25.5 60.7	20.5 0.7
6	-4 -18	-14.4 -14.4	10.4 3.6
7	-56 -56	-63 -63	7 7
8	-63 -49	-58.3 -58.3	4.7 9.3
9	-19 0	-11.2 -11.2	7.8 11.2
10	14 39	22.3 22.3	8.3 16.7
11	46 46	42 42	4 4
12	25 25	20.7 20.7	4.3 4.3
13	-24 0	-8.7 -8.7	15.3 8.7
14	-62 0	7.7 7.7	54.3 7.7
15	-34 84	78.7 78.7	112.7 5.3
16	4 -74	12.2 -76.8	8.2 2.8
17	13 55	-3.1 56.1	16.1 1.1
18	-1 -1	-2.5 -2.5	1.5 1.5
19	-46 45	-58.5 46.3	12.5 1.3
20	-55 -22	-39.5 11.9	15.5 33.9
21	-59 -59	-63 -63	4 4

22	-24 -5	-13.2 -13.2	9.8 8.2
23	-9 85	-17 85.5	8 0.5
24	0 -90	-14.4 -89.8	14.4 0.2
25	10 48	-25 46.1	35 1.9
26	-5 -5	-5.4 -5.4	0.4 0.4
27	-56 -40	-46.4 -46.4	9.6 6.4
28	-50 -36	-44.8 -44.8	5.2 8.8
29	-12 0	-9.8 -9.8	2.2 9.8
30	-10 55	-20.5 61.6	10.5 6.6
31	-5 -51	-66.1 -66.1	61.1 15.1
32	-10 -78	14.4 -81.7	24.4 3.7
		Average error	11.675

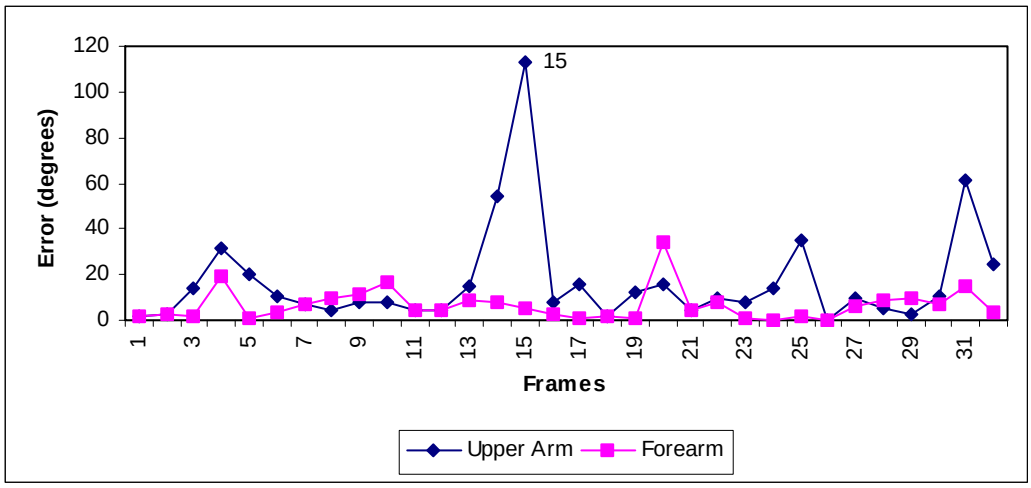


Fig. 5-10 Error plot for the first sequence

Table 5-2 Calculated angles and error for the second sequence

Frame	Orientation Angles	Calculated Angles	Error
1	-39 -39	-41.7 -41.7	2.7 2.7
2	6 6	6.3 6.3	0.3 0.3
3	10 -75	4.9 -69.8	5.1 5.2
4	15 -70	8 -68.4	7 1.6
5	19 19	17.1 17.1	1.9 1.9
6	18 41	30.1 30.1	12.1 10.9
7	18 -45	20.1 -31.7	2.1 13.3
8	7 -52	6.8 -49.6	0.2 2.4
9	0 0	-1.1 -1.1	1.1 1.1
10	-64 -64	-71.6 -71.6	7.6 7.6
11	-73 -73	-81.1 -81.1	8.1 8.1
12	-6 -6	-11.3 -11.3	5.3 5.3
		Average error	4.746

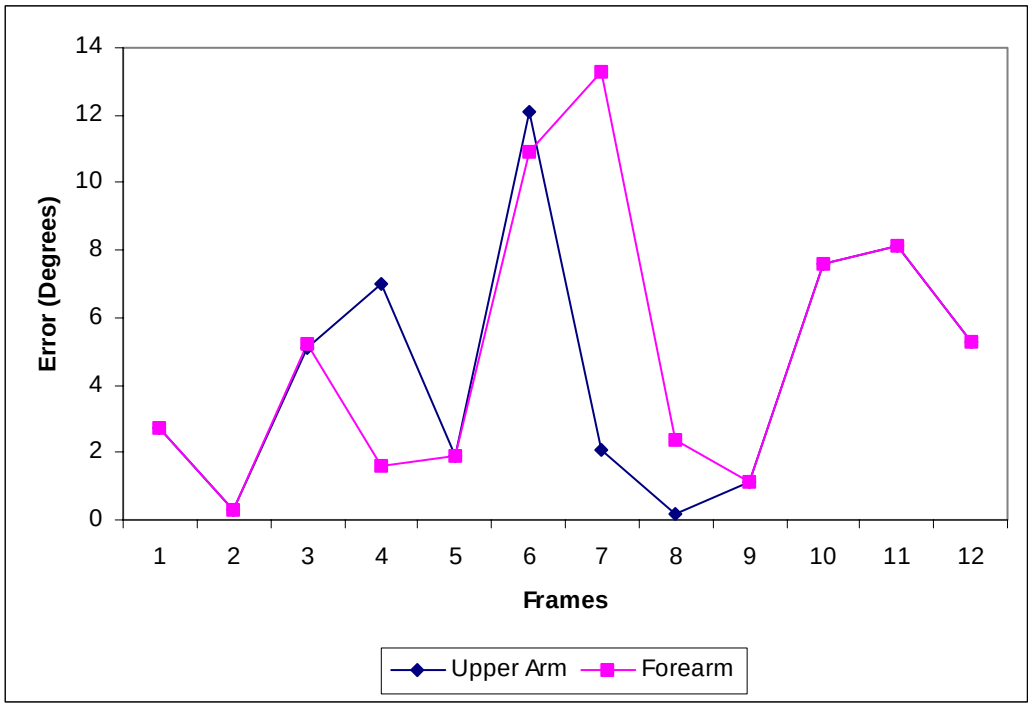


Fig. 5-11 Error plot for the second sequence

The most common problem with the separation of the upper arm and forearm is faced when the arm is bended too much at the elbow. For such a case, the algorithm defines some parts of the shoulder as parts of the forearm. (See Fig. 5-9) This problem might be solved by a further connected component analysis and some heuristics.

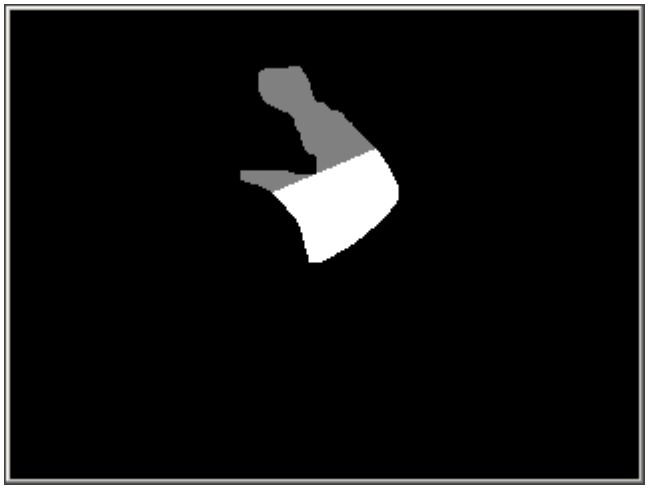


Fig. 5-12 Separation problem

Another possible problem, which is faced rarely, may occur if the human involuntarily moves other parts of his body (such as head) while his arm is stationary. In that case the algorithm also detects that part as the arm since it is the largest region moving in the sequence. Imposing a minimum area constraint on the moving region can reduce the conditions at which this problem may be faced. However, it is still possible to occur since the exact viewed size of the arm cannot be known beforehand.

Since only one camera is used in this research, the determination of the yaw angle is not promising. Normally, stereo vision techniques using at least two cameras are employed for three-dimensional problems. Addition of the calculation of the yaw angle carries the problem to three-dimensional space. The determination of the yaw angle is also considered by observing the shrinkage in the viewed length of the arm. However, the problems, like missing of hands in some frames, make the calculation of viewed arm length unstable. Such problems do not affect the

calculation of orientation angles, but still impose a serious drawback for the calculation of yaw degree.

Although the aim of this research was to develop a real-time system, it is also observed that this is not possible with the high complexity operations applied during the algorithm. With the system, mentioned in Appendix B, the processing rate of the application is approximately 2 frames per second. A higher rate may be obtained by running the application on a computer with higher processing capabilities but still it does not look promising that a real-time processing rate may be achieved.

CHAPTER 6

HUMAN ARM ORIENTATION DETERMINATION BY FEATURE TRACKING

6.1 Introduction

The methods proposed in the previous chapters are based on comparison of a single frame against a background model. Thus, they do not explicitly take into account the motion information, which may be obtained by observing two consecutive frames. In this chapter, an algorithm using the motion information of the human arm is proposed. The algorithm uses Lucas Kanade Feature tracking [21] and finds the orientation of the arm by using the motion vectors of individual features within the image.

As in the previous method, this method also makes use of a background image and the mask of the human at standstill. However, these are used only to find the location of shoulder, which is used in distinguishing between the upper arm and forearm. The detection of the moving arm and determination of orientation angles are realized only by using motion information.

6.2 Lucas Kanade Feature Tracking

Given a point $I(x, y)$ in an image I, Lucas Kanade Feature tracking is used to find the new location, $J(x, y)$ of that point in image J after the

motion. The new location of that point is $(x+d_x, y+d_y)$ where the vector $d=[d_x \ d_y]^T$ is equal to the optical flow at $I(x,y)$. The tracker finds the optical flow vector d, which minimizes the residual function $\in$ on an image neighborhood of $(2\omega_x+1)\times(2\omega_y+1)$ for an image point (u_x, u_y) [22]. The residual function $\in$ is defined as:

$$\in(d_x,d_y)=\sum_{x=u_x-\omega_x}^{u_x+\omega_x}\sum_{y=u_y-\omega_y}^{u_y+\omega_y}(I(x,y)-J(x+d_x,y+d_y))^2 \qquad (6\text{-}1)$$

6.2.1 Pyramidal Implementation of Iterative Lucas Kanade Feature Tracker

The two key components to any feature tracker are accuracy and robustness [22]. Intuitively, a small integration window would be preferable in order not to smooth out the details contained in the images (i.e. small values ω_x and ω_y). The robustness component relates to sensitivity of tracking with respect to changes of lighting, size of image motion, etc. In particular, in order to handle large motions, it is intuitively preferable to choose a large integration window. Therefore, there is a natural tradeoff between local accuracy and robustness when choosing the integration window size. In order to provide a solution to that problem, pyramidal implementation of the classical Lucas Kanade algorithm is proposed [22].

For an image I of size $n_x \times n_y$, I^0 is the zeroth level image, which is the same as the original image I. The next image, in the pyramidal representation is I^1 which has a size of $n_x/2 \times n_y/2$ and is basically the result of sub-sampling of I^0. The following steps are created recursively based on the previous step. The pyramidal representation provides the handling of large pixel motions (larger than the integration window sizes

ω_x and ω_y). The overall pyramidal tracking algorithm proceeds as follows: First, the optical flow is computed at the lowest pyramid level L_m. Then, the result of that computation is propagated to the upper level $L_m - 1$ in a form of an initial guess for the pixel displacement at level $L_m - 1$. Given that initial guess, the refined optical flow is computed at level $L_m - 1$, and the result is propagated to level $L_m - 2$, and so on up to the level 0, which is the original image. Assuming that an initial guess for optical flow at level L, $g^L = [g_x^L \;\; g_y^L]^T$ is available from the computations done at the lower layers. Then, in order to compute the optical flow at level L, it is necessary to find the pixel displacement vector $d^L = [d_x^L \;\; d_y^L]^T$ that minimizes the new residual function $\in^L$:

$$\in^L (d_x^L, d_y^L) = \sum_{x=u_x^L-\omega_x}^{u_x^L+\omega_x} \sum_{y=u_y^L-\omega_y}^{u_y^L+\omega_y} (I^L(x,y) - J^L(x + g_x^L + d_x^L, y + g_y^L + d_y^L))^2 \qquad (6\text{-}2)$$

The optical flow at level L, is the value of $d^L = [d_x^L \;\; d_y^L]^T$ for which the first derivative of $\in^L (d_x^L, d_y^L)$ is equal to zero. That is;

$$\frac{\partial \in^L (d^L)}{\partial d^L} = [0 \;\; 0] \qquad (6\text{-}3)$$

Defining;

$$A(x,y) = I^L(x,y) \qquad (6\text{-}4)$$

$$B(x,y) = J^L(x + g_x^L, y + g_y^L) \qquad (6\text{-}5)$$

Then by dropping the L superscripts,

$$\frac{\partial \in (d)}{\partial d} = -2 \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (A(x,y) - B(x + d_x, y + d_y)) . \left[\frac{\partial B}{\partial x} \;\; \frac{\partial B}{\partial y} \right] \qquad (6\text{-}6)$$

After a set of derivations including a Taylor series expansion of the above term, it is found that;

$$\frac{1}{2} \left[\frac{\partial \in (d)}{\partial d} \right]^T \approx Gd - b \qquad (6\text{-}7)$$

where

$$G = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \begin{bmatrix} \frac{\partial A}{\partial x}^2 & \frac{\partial A}{\partial x}\frac{\partial A}{\partial y} \\ \frac{\partial A}{\partial x}\frac{\partial A}{\partial y} & \frac{\partial A}{\partial y}^2 \end{bmatrix} \tag{6-8}$$

$$b = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \begin{bmatrix} (A(x,y) - B(x,y))\frac{\partial A}{\partial x} \\ (A(x,y) - B(x,y))\frac{\partial A}{\partial y} \end{bmatrix} \tag{6-9}$$

Then, the optimum optical flow vector becomes equal to

$$d_{opt} = G^{-1}b \tag{6-10}$$

The above derivation assumes that the pixel displacement is small in order to use Taylor series expansion. However, that may not always be the case. In order to overcome this problem the algorithm is applied iteratively until the result of each step decreases to a threshold. After each iteration, the found displacement is added to $B(x, y)$ in the form of $B(x + d_x, y + d_y)$ and this new value is used for the new iteration.

6.3 The Algorithm

In order to determine the orientation of the arm, the motion information of two consecutive frames are used. This information is obtained by Lucas Kanade feature tracker. However, suitable pixels (i.e. features) are determined first, for the tracker to operate efficiently. Later, the algorithm is directed on the clustering of motion vectors in order to distinguish the upper arm and forearm. Lastly, the linear system formed by the set of tracked features is solved for the orientation angles.

6.3.1 Finding Suitable Features

Edges and specifically corners are suitable features for tracking within an image. In order to find corners, first the minimal eigenvalue for every source image pixel is calculated [23]. Then, non-maxima suppression is performed only leaving local maxima in 3x3 neighborhood and corners with an eigenvalue, which is less than a threshold, or corners which are too close to another stronger corner are eliminated.

Fig. 6-1 Suitable features determined

6.3.2 Feature Tracking and Separation of the Forearm and Upper Arm

During the conducted simulations, first tracking of features between only two consecutive frames is studied. In this approach, suitable features for tracking are determined for each new frame and these features are tracked in the next frame by using Lucas Kanade feature tracking. Any

feature, which has a movement greater than a certain threshold, is assumed to be on the arm. The position of each feature (x and y) and their movement vectors (d_x and d_y) constitute a 4 dimensional vector. The space generated by all the moving features is clustered into 2 subspaces by 2-means clustering [24]. One of these subspaces form the upper arm while the other forms the forearm. This operation is repeated from the start for each new frame.

The second approach studied, provided a better tracking operation. In this approach, previously tracked features are tried to be followed in the consequent frames. Similar to the previous approach, suitable features for tracking are determined and tracked by Lucas Kanade feature tracking for each new frame. However, the information on features, which are found to be moving are propagated to the next frame, where they are being kept tracked in addition to the new features determined for this frame. Thus, a feature, which is found to be moved in a previous frame, is tracked for all consecutive frames even if its movement is less than the pre-determined threshold. Similar to the previous approach, 2-means clustering is applied with each new frame and upper arm and forearm is determined.

6.3.3 Determination of the Orientation Angles

Once the found features are classified as forearm or upper arm, the orientation angles of these body parts may be determined by using the position and motion information of these features. When a point (x, y) is rotated by θ degrees, the new position (x', y') of the point may be found by the following affine relation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6-11}$$

In this equation, x and y are known terms while x' and y' are found by adding the motion vectors d_x and d_y to x and y.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \end{bmatrix} \tag{6-12}$$

The above equation is modified in order to include error terms. Errors may be caused by camera noise or the movement of the shoulder. The error terms also take into account, the movement of the upper arm while the calculations are made for the forearm.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{6-13}$$

Renaming $\cos\theta$ as a and $\sin\theta$ as b, the above equation my be rewritten as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{6-14}$$

By converting this equation into a linear system of the form $AX = B$ where X is the unknown matrix, we obtain;

$$\begin{bmatrix} x & y & 1 & 0 \\ y & -x & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{6-15}$$

It is observed that there are 4 unknown terms (a, b, Δx, Δy); hence, in order to solve the above linear system at least 2 sets of points are required. Assuming that n different sets of points are available where $n > 2$, the system becomes an over determined linear system.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 \\ y_1 & -x_1 & 0 & 1 \\ x_2 & y_2 & 1 & 0 \\ y_2 & -x_2 & 0 & 1 \\ . & . & 1 & 0 \\ . & . & 0 & 1 \\ x_n & y_n & 1 & 0 \\ y_n & -x_n & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ . \\ . \\ x'_n \\ y'_n \end{bmatrix} \tag{6-16}$$

Since the first matrix is not square, its pseudo-inverse is used in order to solve this system, which results with the following relation;

$$\hat{X} = (A^T A)^{-1} A^T B \tag{6-17}$$

The values of a and b are found by solving this system, and the orientation angle (θ) is obtained by;

$$\theta = \tan^{-1}\left(\frac{b}{a}\right) = \tan^{-1}\left(\frac{\sin\theta}{\cos\theta}\right) \tag{6-18}$$

The above set of calculations determines the change in θ for two consecutive frames. In practice, the human starts moving his arm from a known value of θ (typically 0 or 90 degrees). Therefore, by adding up the found changes in θ up to current frame, the value of θ is found.

These calculations are done for both the upper arm and the forearm and the orientation angles are obtained for each of them.

6.4 Performance of the Algorithm

During the simulations, it is observed that Lucas Kanade feature tracking algorithm is successful while tracking the determined features. Table 6-1 summarizes the results of these experiments and it can be observed that the average error is around 13 degrees. The main problem was faced during the separation of the upper arm and the forearm. As mentioned in Section 6.3.2, two different methods are examined. However, both methods were unable to give a reliable separation.

When the first approach is used, it is impossible to detect the upper arm while it is stationary. Since only the moved features are taken into consideration while determining the angle the features in the upper arm are lost while it is stationary. This situation causes 2-means clustering algorithm to operate on the features from the forearm and the forearm is separated into two parts.

In order to solve the above problem, the second method is developed where not only the moving features between frame pairs but any feature which has moved throughout the video sequence is being tracked. The main disadvantage of this approach is that any feature, whether on the arm or not, is being tracked if it has moved in any frame of the video sequence. This causes points, which has showed up as moving due to error, or other factors such as the shadow of the moving arm to be tracked throughout the video sequence. These kind of features accumulate as the video sequence continues and becomes to constitute a large portion of tracked features (See Fig. 6-2).

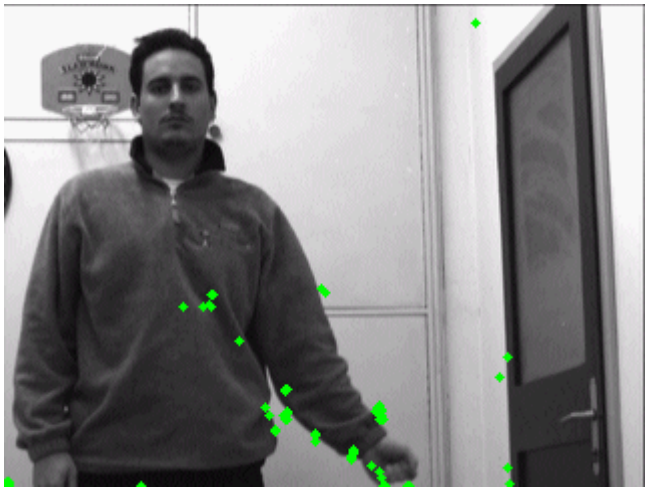


Fig. 6-2 Accumulated moving parts

In order to overcome this problem, the number of consecutive frames for which each point stays stationary is calculated. If a feature is stationary for more than 10 consecutive frames, this feature is not tracked anymore. Although, such a discard reduces the tracking of noisy or unnecessary features, it also eliminates features on the arm, if the arm is stationary for more than 10 frames. Therefore, this approach is also not capable of solving the problem of tracking the arm, while its stationary.

The main problem of separation of the arm is caused by the nature of the tracking of movement information. Since the algorithm totally depends the tracking of moving features, it is natural to experience problems when dealing with non-moving features within the image.

Assuming that the arm is moving at any instant, tests are conducted in order to evaluate the performance of the orientation angle determination algorithm of section 6.3.3. During the tests, the arm is kept straight at all times and moved throughout the sequence. The average error is found to

be 13.621 degrees (7.6% error), which is higher than expected (see Table 6-1 and Fig. 6-3),. However, this result is satisfactory comparing it to the 7.5% error of the research in [7]. The error is high due to the fact that at each frame, the change in orientation angle is found instead of its real value. Then, the real value is constructed by adding up all the changes up to that frame. Therefore, errors within the calculations for each frame accumulate and cause a large total error as orientation angle greatly differs from the initial posture (0 degree for the below example).

Table 6-1 Calculated angles and error

Frame	Orientation Angles	Calculated Angles	Error
1	38	53.6	15.6
2	45	57	12
3	10	25.5	15.5
4	-25	-9.6	15.4
5	-46	-26.5	19.5
6	-29	-10.4	18.6
7	5	10.1	5.1
8	16	26.2	10.2
9	-3	2.2	5.2
10	0	8.3	8.3
11	29	37.1	8.1
12	34	42	8
13	-46	-30	16
14	-55	-36.9	18.1
15	-28	-13.9	14.1
16	26	34.3	8.3
17	-31	-17.5	13.5
18	-71	-47	24
19	-49	-25.7	23.3
		Average error	13.621

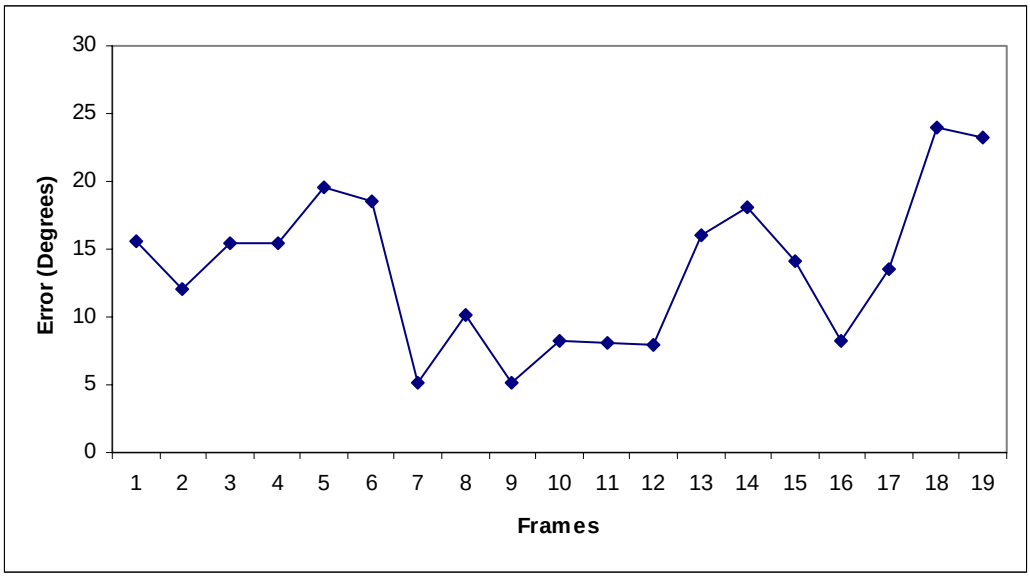


Fig. 6-3 Error plot for the test sequence

CHAPTER 7

CONCLUSIONS

7.1 Conclusions

Robot mimicking by using visual data is an active research area, covering many different fields of science and engineering such as artificial intelligence, computer vision, machine learning, kinematics and image processing. However, when the target to be mimicked is a human, intrinsic difficulties for describing the human dynamics arise. In order to overcome these difficulties and accomplish robot mimicking by using visual data, in this thesis three methods have been proposed.

First, the results of a preliminary research on mimicking are presented, where a 2-DOF model joint with markers is mimicked by a PUMA 760 robot arm. The images of the model joint are processed and the locations of the markers are determined. By using these locations two parameters, which are the yaw and bending angles, defining the posture of the model joint are obtained. The same posture is taken by the robot arm in order realize the mimicking. The results were satisfactory and guided us for the aim of creating a system where the movements of a real human arm are mimicked.

The secondly proposed mimicking system employs basic computer vision algorithms in order to detect a moving human arm in a video sequence. The arm is detected by using background subtraction and the upper arm and forearm are segmented by using convex hull analysis. Afterwards,

the orientations of the upper arm and forearm are determined which are used as mimicking parameters by a PUMA 760 robot arm.

Although problems, limiting the movement of the arm or in finding the elbow, are experienced, it is also observed that the system is satisfactory considering the aforementioned difficulties in this field.

The last proposed system makes use of Lucas Kanade feature tracking in order to track points on the moving human arm. After finding the position and movement of points on the arm, the system finds the orientation by using a simple rotation matrix. Problems were encountered in the segmentation of the arm when the upper arm is stationary, since the only information used in this system is movement.

A comparison of the two proposed systems shows that background subtraction is a more efficient system. This result is due to the fact that the human arm may be stationary during the operation and background subtraction is capable of handling such a case, while feature tracking has still problems with segmentation. However, background subtraction requires preliminary information on the background and the human at standstill before operation, while feature tracking system is capable of operating with small or even no preliminary information. It is also observed that, assuming proper segmentation, background subtraction has a smaller average error.

7.2 Proposed Future Work

The arm detection and orientation estimation algorithms may be improved to be more robust against illumination and background changes or unexpected arm movements conducted by the human target.

Moreover, additions to the employed algorithms might allow detection and identification of both arms in a single sequence, thus nearing the final aim of true mimicking of humans. However, for such a system to be operational a more humanoid robot than PUMA 760 must be used as the mimicking robot.

Based upon the experience gained through this research, a hybrid mimicking system may be constructed using the advantageous properties of each proposed system while reducing their drawbacks.

Another important drawback of both of the systems is their inability to work real-time both due to the complexity of the algorithms and the separation of the visual analysis and robot control programs. The visual analysis may be modified for a reduction in complexity and different algorithms may be exploited to accelerate the operation. Moreover, the integration of the visual analysis and robot control software constitute a good direction for future work.

REFERENCES

[1] Wu, Y.; Huang, T.S.; "Hand Modeling, Analysis, and Recognition", IEEE Signal Processing Magazine, Vol.18, Issue 3, May, 2001

[2] Pavlovic, V.I.; Sharma, R.; Huang, T.S.; "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 19, Issue 7, pp. 677 – 695, July 1997

[3] Triesch, J.; Von der Malsburg, C.; "A Gesture Interface for Human-Robot-Interaction", Proceedings of The IEEE Third International Conference on Automatic Face and Gesture Recognition, April, 1998

[4] Littmann, E.; Drees, A.; Ritter, H.; "Robot Guidance by Human Pointing Gestures", Proceedings of NICROSP, IEEE Computer Society Press, 1996

[5] Cheng, G.; Kuniyoshi, Y.; "Real-Time Mimicking of Human Body Motion by a Humanoid Robot", Proceedings of The 6th International Conference on Intelligent Autonomous Systems, pp.273-280, July, 2000

[6] Rybski, P.E.; Voyles, R.M.; "Interactive Task Training of a Mobile Robot through Human Gesture Recognition", Proceedings of the 1999 IEEE International Conference on Robotics and Automation, Vol. 1, pp. 664-669, 1999

[7] Di Bernardo, E.; Goncalves, L.; Perona, P.; "Monocular Tracking of the Human Arm in 3D: Real-Time Implementation and Experiments", Proceedings of the International Conference on Pattern Recognition, ICPR '96, pp. 622-626, August, 1996

[8] Amit, R.; Mataric, M.; "Learning Movement Sequences from Demonstration", Proceedings of The 2nd International Conference on Development and Learning, pp.203-208, 2002

[9] Ude, A.; Atkeson, C.G.; "Real-Time Visual System for Interaction with a Humanoid Robot", Proceedings of the 2001 International Conference on Intelligent Robots and Systems, 2001

[10] Jain, R.; Kasturi, R.; Schunck, B.G.; "Machine Vision", McGrawHill, 1995

[11] Sankur, B.; Sezgin, M.; "A Survey Over Image Thresholding Techniques And Quantitative Performance Evaluation", Journal of Electronic Imaging, 13(1), pp.146-165, January, 2004

[12] Foley, J.D.; van Dam, A.; Feiner, S.K.; Hughes, J.F.; "Computer Graphics: Principles and Practices", Addison-Wesley, 1990

[13] Kilian, J.; "Simple Image Analysis By Moments", OpenCV Library Document, March 15, 2001

[14] Bronnimann, H.; Iacono, J.; Katajainen, J.; Morin, P.; Morrison, J.; Toussaint, G. T.; "Optimal in-place planar convex hull algorithms," Proceedings of Latin American Theoretical Informatics, LATIN2002, pp. 494-507, Cancun, Mexico, April 3-6, 2002

[15] Sklansky, J.; "Measuring Concavity on a Rectangular Mosaic", IEEE Transactions on Computers 21, pp 1355-1364, 1972

[16] Lim, J.S.; "Two-Dimensional Signal and Image Processing", Prentice Hall, 1990

[17] Intel's Open Source Computer Vision Library (OpenCV), http://www.intel.com/research/mrl/research/opencv, December 2004

[18] Arseneau, S.C.; Nicholls, R.J.; Farooq, M.; Hopkinson, A.; "Robotic Mimicking Control System", Proceedings of the 44th IEEE Midwest Symposium, Vol. 2, pp.547-550, 2001

[19] Gebizlioğlu, Ö.E.; "External Control of PUMA 700 Series Robot Based on the Communication Protocols LUN and DDCMP", M.S. Thesis, Middle East Technical University, Ankara, Turkey, 2003

[20] Dindaroğlu, M. S.; "Control of PUMA Mark II Robot with an External Computer", M.S. Thesis, Middle East Technical University, Ankara, Turkey, 2002

[21] Lucas, B.D.; Kanade, T.; "An Iterative Image Registration Technique with an Application to Stereo Vision", Proceedings of Imaging Understanding Workshop, pp.121-130, 1981

[22] Bouguet, J.Y.; "Pyramidal Implementation of the Lucas Kanade Feature Tracker, Description of the Algorithm", OpenCV Library Document, February, 2003

[23] Harris, C.; Stephens, M.; "A Combined Corner and Edge Detector", Proceedings of The Fourth Alvey Vision Conference, pp.147-151, 1988

[24] Schalkoff, R.J.; "Pattern Recognition: Statistical, Structural and Neural Approaches", John Wiley & Sons, 1992

APPENDIX A

SPECIFICATIONS OF PUMA 700 SERIES ROBOTS

Table A-1. Specification of PUMA 700 series robots.

DOF	6
Drives	DC Motors
Control	Numerical
Positional Control	Incremental Encoders
Coordinates	Cartesian
Configuration	Revolute
Minimum Reach	0.125 mm (Between Joint 1 and 5)
Maximum Reach	360 deg Working Volume
Limit Joint 1	320 deg
Limit Joint 2	220 deg
Limit Joint 3	270 deg
Limit Joint 4	532 deg
Limit Joint 5	200 deg
Limit Joint 6	600 deg
Repeatability	-762 model +/- 0.2mm -761 model +/- 0.2mm
Maximum Speed	1.8 m/s
Auxiliary Processors	6 Slave Microprocessors
Programming	Teach Pendant VAL II language
Serial Interface	RS232 or RS423
Memory Buffer	46KB
Battery Buffer	30 Days
Arm Weight	-762 model 590kg -761 model 600kg
Controller Cabinet Weight	200kg

APPENDIX B

THE APPLICATION SOFTWARE FOR BACKGROUND SUBTRACTION ALGORITHM

The program performing the human arm orientation determination by background subtraction is written in Microsoft's Visual C++ 6.0 by using OpenCV library. The platform used both for coding and running is Intel Pentium3 733 Mhz, with 128 MB of RAM and a Windows 2000 operating system. The only input parameter to be adjusted by the user is the threshold level, which is used in the various steps of the algorithm. The program is capable of processing video files in AVI format or directly capturing from a video camera connected to the computer through a TV-card.

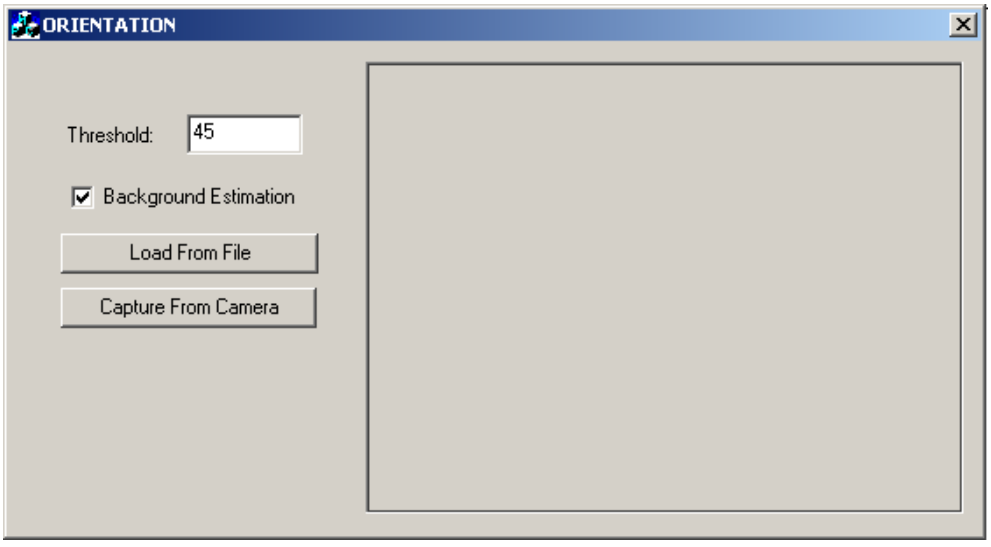


Fig. B-1 The program interface

The program has a single user interface where the threshold value, operation mode and the source of the video sequence may be selected and video sequence being processed may be viewed. If the user desires to make background estimation, he/she should select the appropriate checkbox. After this step, either "Load From File" button or "Capture From Camera" button must be clicked to start the operation. If the "Load From File" button is clicked an open file dialog-box is displayed for the selection of the AVI file to be used. If the "Capture From Camera" button is clicked two consecutive dialog boxes for the selection of video source properties are displayed. When the operation is complete a save file dialog-box is displayed in order to select where the resultant (Bitmap) image will be stored.

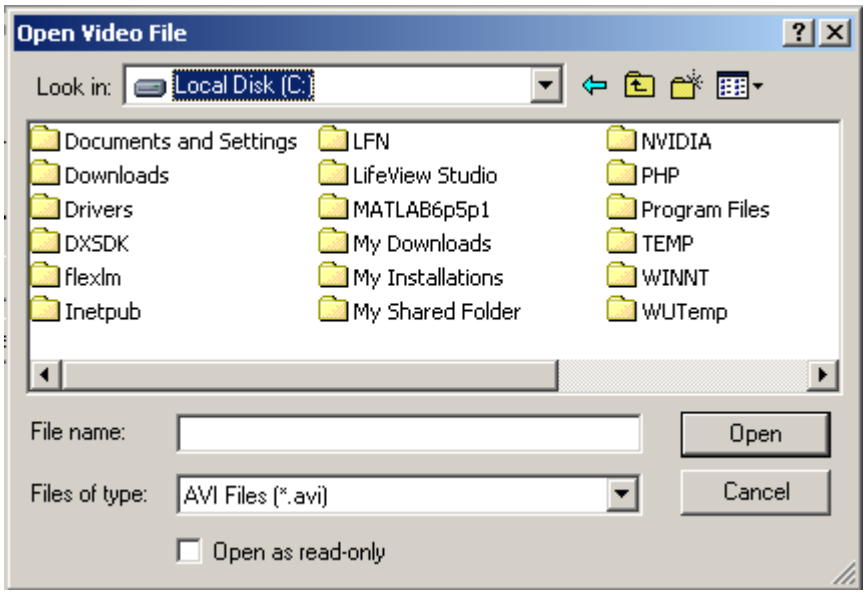


Fig. B-2 Open Video File Dialog Box

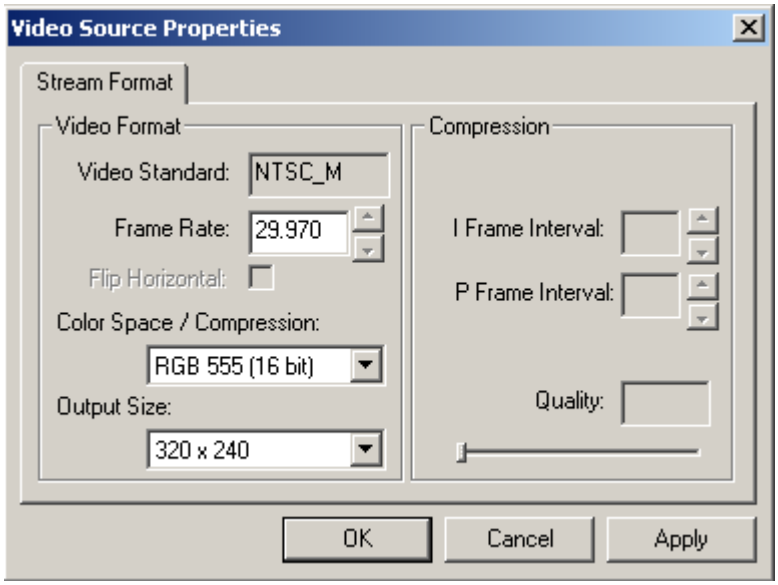


Fig. B-3 Video Source Properties Dialog Box-1

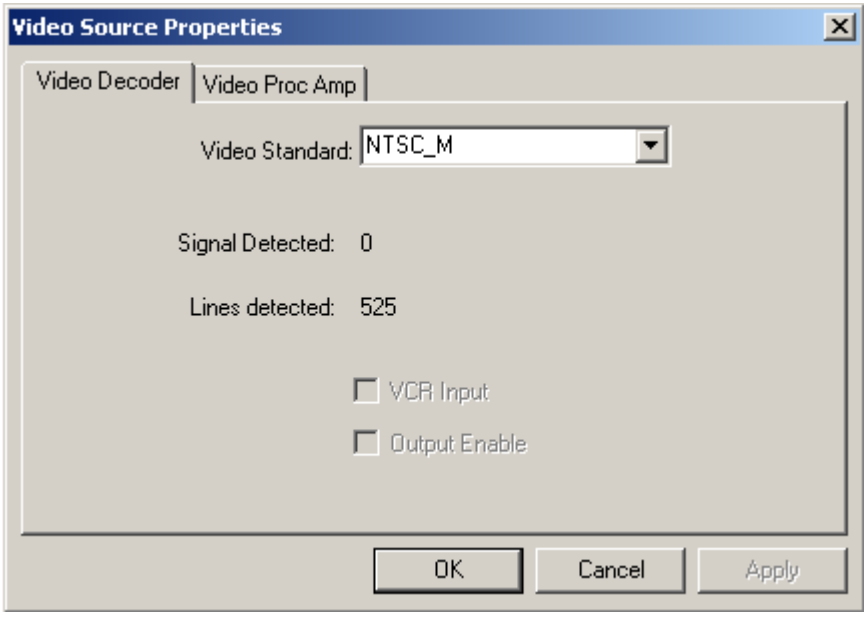


Fig. B-4 Video Source Properties Dialog Box-2

If the user desires to process a video sequence for orientation analysis, he/she should click either the "Load From File" button or the "Capture From Camera" button without selecting the "Background Estimation" check box. Upon clicking the appropriate button two open file dialog-boxes are displayed consecutively in addition to the dialog boxes mentioned above. These open file dialog-boxes enable the user to select the image file to be used as the background estimate and the image file in which the human exposes his arm length for length estimation. When all the files are chosen the operation begins and the processed video sequence can be viewed in the interface with additional information overlaid. This information is

- The frame count on the upper left corner
- The yaw angle on the middle right
- The orientation of upper arm and forearm on the lower right.
- The center of masses of the upper arm and forearm are displayed with circular markers.

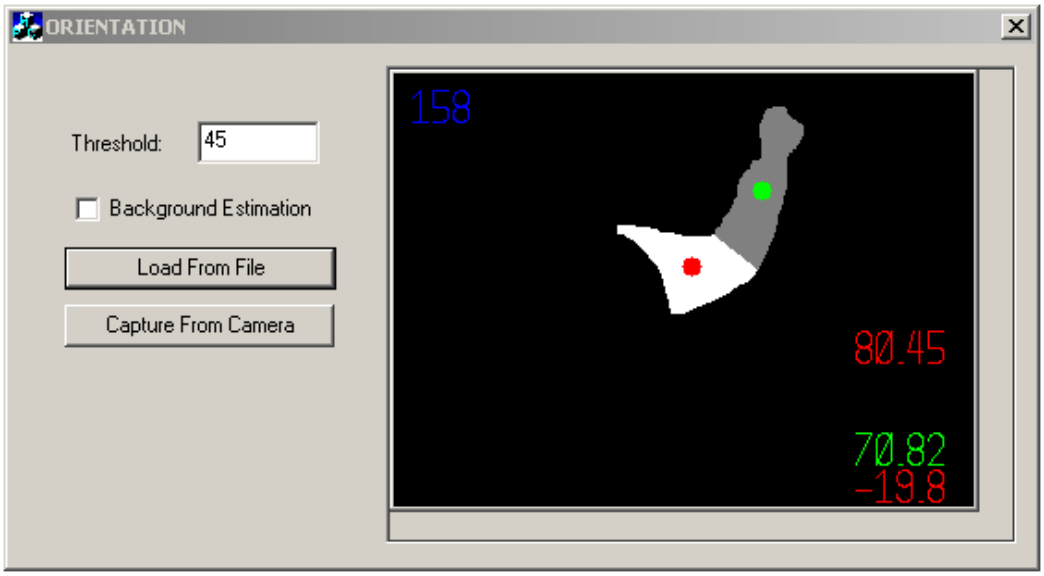


Fig. B-5 Program interface during orientation estimation

During the operation the calculated angles are stored in a text file in order to be used by the robot communication program for actual mimicking.

APPENDIX C

THE ROBOT INTERFACE PROGRAM

A program is developed in order to process the results of the two orientation determination algorithms by using Borland's C++ Builder. The programs implementing the algorithms saves the output orientation angles in the form of a text file. In order to convert these angles into robot position information orientation, orientation is checked for sign changes since two different arm postures may yield the same orientation angle as seen in Fig. C-1.

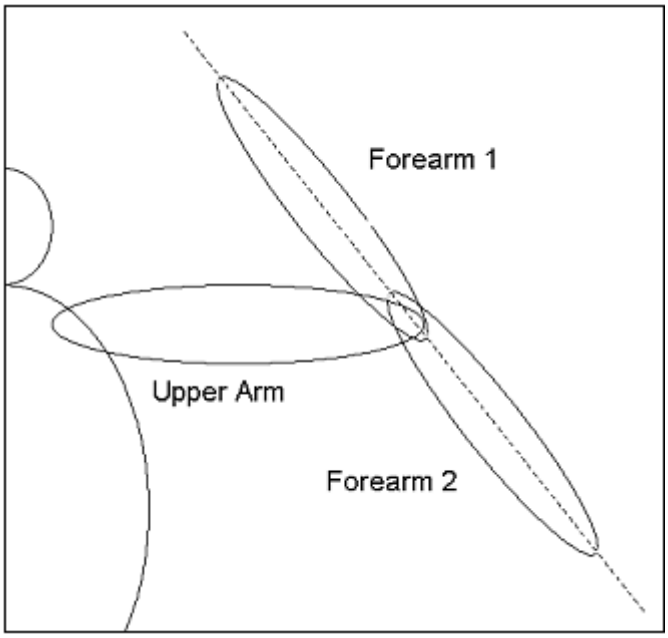


Fig. C-1 Arm posture and orientation

As the initial position of the arm is known the correct orientation is obtained by checking through the angle sequence for sign changes. Following this operation a low pass filter is applied in order to smooth the output and reduce the erroneous orientation angle values. Finally, an output text file is prepared, which is compatible with Ö. Gebizlioğlu's PUMA 760 Control program.

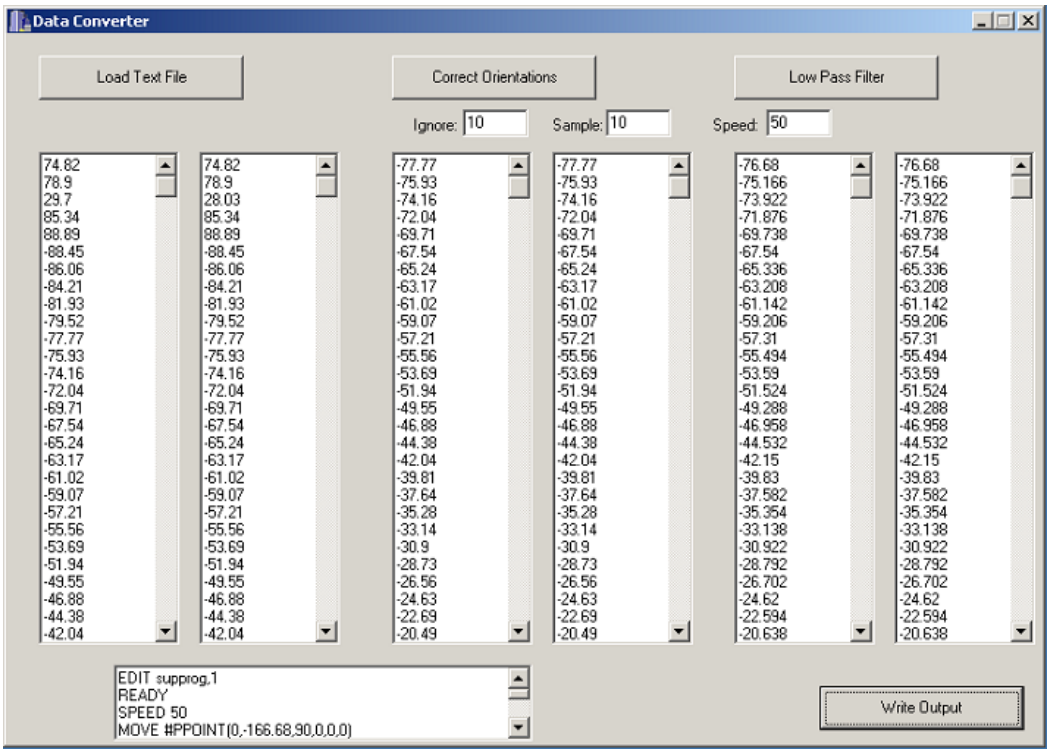


Fig. C-2 Robot Interface Program