BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

System Input Using Marker Based Hand Gesture Recognition

By

Sayem Mohammad Siam (084429)

Jahidul Adnan Sakel (084433)

Supervised By

Md. Hasanul Kabir, PhD

Assistant Professor, Department of CSE, IUT

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
Organization of the Islamic Cooperation (OIC)
Dhaka, Bangladesh

October, 2012

Abstract

Human Computer Interaction(HCI) has been changed in this new era. People want to interact with their devices in such way that has physical significance in the real world and of course want ergonomic input devices. We propose a new method of interaction for the computing devices, having a consumer grade web-cam, with the beck of the hand. Our system is also suitable where touch screen is not feasible like large screen or projected screen. Our method uses two colored marker(red and green) worn on tip of finger to generate eight distinct hand gestures. We have implemented all the usual system commands through different gestures. Our system can easily recognize these gesture and give corresponding system commands.

Acknowledgment

We would like to first thank the Almighty Allah for all the blessings he has bestowed upon us throughout our entire life and throughout our B.Sc program. Without the grace of Allah, We wouldn't be where we are right now. "All thanks and praises be to Allah". We also would like to thank our supervisor Md. Hasanul Kabir, PhD for his valuable suggestion and inspiration. We also would like to thank our classmates and teachers who always have been source of our inspiration.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The field of Human Computer Interaction (HCI) aims at improving interactions between users and computers by making computers more usable and receptive to the users need. This field has developed many input-output techniques including the technique of using hand gesture as input device over the last few years. We propose a hand gesture recognition technique that uses colored markers. The system inputs those can be given by a mouse or touch-pad can be performed using our proposed technique. That is the user will be able to perform system command with the beck of the hand.

Currently we use mostly mouse, keyboard, touch-pad to interact with the computer. These devices are sensor based and had been being used for last few decades as well. Newly emerging technologies in this field are Kinect, Xbox, Ipad, Microsoft surface which mostly focuse on gesture. So the trend of future human computer interaction is shifting toward the sensor less input devices. This is the main motivating factor for our proposed idea. We aim at developing alternative input device to the mouse and touch-pad and in some cases something more.

Use of hand gesture for giving system input is more natural, ergonomic, user friendly and cheap as well. In cases where use of touch-screen is not feasible hand gesture can be a suitable solution. Gesture detection can be done by hand shape detection or by skin color detection or by the detection of marker. For recognizing hand gesture use of marker is better choice because it is cheap, available, user friendly and marker allows accurate detection with simpler algorithms.

1.2 Problem Statement

Developing a cost effective system that allows user to interact with computer using hand gesture with the help of color marker. Our main focus is to implement the commands which can be given through a mouse or a touch-pad.

1.3 Research Challenges

We have several challenges. The first challenge is to detecting the marker in different light intensity. We have worked with hue and saturation values of the image so the effect of light intensity in detection has been reduced. The second challenge is the real time detection. That is we need to detect the marker in a vary less amount of time so that the user's interaction with the computer can be at real time. The third challenge is the accuracy in detection and tracking. The more accurate the detection and tracking is the easier it will be for the user to generate gestures. The fourth challenge is the detection of the lost marker. That is how fast we can detect and track the lost marker again. The next challenge is to segment out the the marker from the same color background. To meet this challenge we define a range of acceptable area for the marker. If the area is greater than specified range then that object is considered as a background not a marker. Again if the area is smaller than the specified range that is not considered as the detection of marker. The sixth challenge is the tracking of marker when some other object having same color as the marker comes in the image. We manage this situation with the use of kalman filter. The next challenge is the accurate gesture recognition which mostly depend on accuracy of detection and tracking. The last but not the least challenge is to make the whole system user friendly. The solution to this challenge lies mostly in the solution to other afore mentioned challenges.

1.4 Motivation

The trend of future human computer interaction is shifting towards the gesture based input devices. In this regard "MIT sixth sense project" [3] has been one of the most motivating works for us.We wanted to develop a system which could enable us to interact with the computer in natural way. Moreover it would be cheap and ergonomic. This system would be best suited in the situation like presentation on big screen where mouse, keyboard, touch-pad are not suitable. The camera that comes with each laptop would be enough to interact with the computer. Again marker can be any uniformly colored object so it will be cheap.

1.5 Scopes

We are working with color marker which is easy to detect in terms of computation complexity. Markers are easy to track as well with the help of kalman filter. As computation complexity is not so high it requires less time to detect and track the marker which results in real time interaction with the computer. Accuracy in detection of the center of the object is not so much

important because that has very less effect on gesture recognition and generation. Because of use of kalman filter the cursor movement has been very smooth which helps user to generate gesture easily.

1.6 Research Contribution

Since our marker size and color is fixed, template matching can detect the marker accurately and template matching is less time consuming. We used HSI(Hue, Saturation and Intensity) instead of RGB because H(Hue) and S(Saturation) values are invariant to different light intensities.We used sliding window mechanism which reduce calculation time. We used Sum of squared differences(SSD) which takes less time to compute than the other template matching techniques. For using Colored marker we are getting greater accuracy and less time complexity for real time detection. For using Kalman filter we are getting smooth detection of marker and getting rid of detecting unwanted position and miss detection of marker, since we do not actually need the exact position of marker. If marker is moved right we must move the cursor right and same for the left and this task is greatly done by using Kalman filter.

1.7 Thesis Outline

In Chapter 1 a brief introduction of the proposed method has been provided. In chapter 2 we provide literature review. We discuss some related Algorithms, techniques of detection and tracking. In Chapter 2 we discuss about existing human computer interaction techniques and what could be possible way of future human computer interaction. In chapter 3 we have discussed our proposed method in details including tracking and detection algorithms and the proposed hand gestures and their recognition techniques. In chapter 3 we provide some system analysis diagram as well. The chapter 4 states the experimental analysis of our proposed method. In chapter 4 we show user friendliness of our system and gesture generation accuracy and we show the average performance and performance in different velocity of hand as well. Chapter 5 states the conclusion.

Chapter 2

Literature Review

The idea of system input using hand gesture is not new. It can be done using sensors and without sensors (that is no physical connection with the system). Our interest is of course the later one. This idea can be accomplished using images taken by a consumer grade web cam attached with the system. The first phase of this technique is the detection. There are several detection techniques. Some system like camera mouse [4] suggests selecting the object first that is to be detected. This selection is done with the help of a mouse. The main advantage of detection in this approach is that under different light conditions it works fine as intensity value of the object is known for particular amount of light at the very beginning of tracking. But major disadvantage is that it is dependent on another input device (why don't we replace mouse?). There are other techniques those detect some predefined objects. In these techniques hand gesture is recognized by the detection of only different shapes of hand itself [5] [6] or with the help of markers [5] of different colors. The markers can be worn on finger tips or in any other parts or even on gloves [5].

2.1 Definition of Marker

A marker is a uniformly colored object usually worn in finger tip or any other parts of human body which can be tracked using different algorithms with a view to obtaining specific goal in specific applications.A marker may be of several colors. The shape of the marker should be unique and easily distinguishable from the background.

2.2 Color Models

A color model facilitates the specification of colors in some standard, generally accepted way [1]. The color model most commonly used in practice are RGB (Red, Green, Blue), CMY (Cyan, Magenta, Yellow), CMYK (Cyan, Magenta, Yellow, Black), HSI (Hue, Saturation, Intensity). The choice of color model depends on the hardware and the application domain environment. We have used HSI color model and worked with Hue and Saturation values.

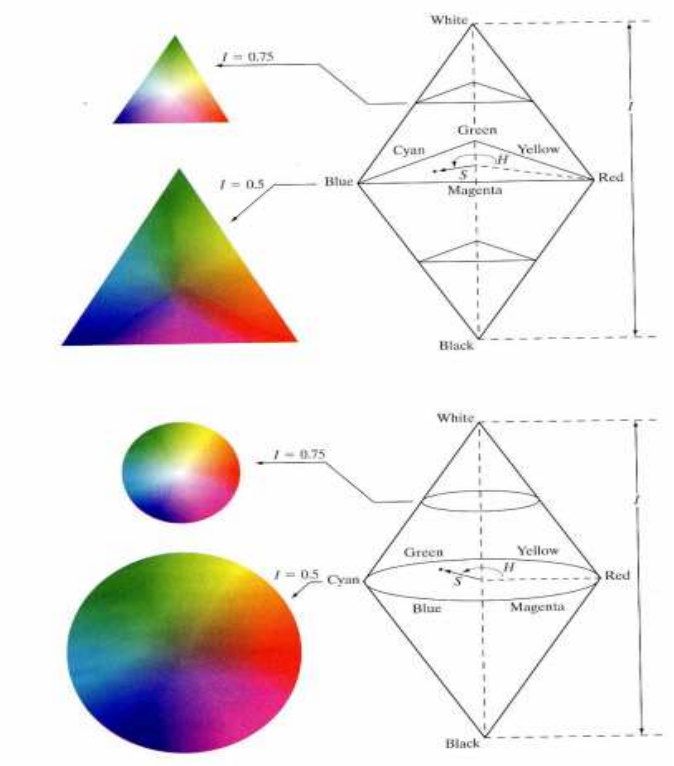

Figure 2.1: The HSI color model based on triangular and circular color planes [1]

2.2.1 HSI Color Model

When we view a colored object we describe the object by its hue, saturation and brightness. Hue is a color attribute that describes a pure color, on the other hand saturation gives a measure of the degree by which a pure color is diluted by white light and brightness is a subjective descriptor that is particularly impossible to measure. The HSI model is an ideal tool for developing image processing algorithms based on color descriptions that are natural and innovative to human [1].

2.2.2 RGB Color Model

In RGB color model the primary spectral components are Red, Green and Blue. This model is based on Cartesian coordinate system. The color subspace of interest is a cube, in which RGB values are at its three corners and the cyan, yellow and magenta are other three corners and balck is at origin. White is at the corner farthest from the origin. In this model the gray scale extends along the line joining those two points (black and white). The different colors are points on this cube or inside this cube [1].

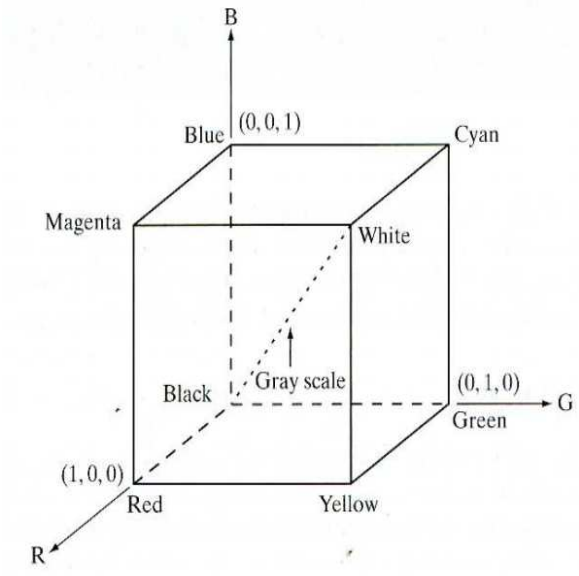

Figure 2.2: RGB color cube [1]

2.3 Human Computer Interaction (HCI)

Human Computer Interaction (HCI) Refers to the design and implementation of computer systems that people interact with. It includes desktop systems as well as embedded systems in all kinds of devices. Although the user interface is the primary element between user and computer, HCI is a larger discipline that deals not only with the design of the screens and menus, but with the reasoning for building the functionality into the system in the first place [7]. One important HCI factor is that different users form different conceptions or mental models about their interactions and have different ways of learning and keeping knowledge and skills (different "cognitive styles" as in, for example, "left-brained" and "right-brained" people). In addition, cultural and national differences play a part. Another consideration in studying or designing HCI is that user interface technology changes rapidly, offering new interaction possibilities to which previous research findings may not apply. Finally, user preferences change as they gradually master new interfaces [8].

2.3.1 Existing HCI Technologies

The existing HCI technologies falls under three or any combination of those three categories. These are visual based, audio based and sensor based. Our main focus is on visual based HCI. The visual based Human computer interaction is probably the most wide spread area in HCI research. Some of the main research areas in this section are:

- Facial Expression Analysis
- Body Movement Tracking (Large-scale)
- Gesture Recognition
- Gaze Detection (Eyes Movement Tracking)

Among the above four Gesture recognition is the Gesture recognition is the main focus of this area and so is of ours. It is notable that some researchers tried to assist or even replace other types of interactions (audio-, sensor-based) with visual approaches. For example, lip reading or lip movement tracking is known to be used as an influential aid for speech recognition error correction [9]. Some example of latest HCI technologies are 10/GUI, Microsoft surface, IpadII and specially KINECT (which enables user to control and interact with X-box 360s without need to touch the game controller) is a motion sensing input device and its working principle is based on image processing.

2.3.2 Future HCI Technologies

Future HCI technologies will be of course visual based and more specifically gesture based. People want flexible and easy way of interaction with the computing devices. In the future days we are going to have Multi-modal (combination of video and speech) HCI technologies.

2.4 Hand Gesture Techniques

There are several hand gesture generation techniques. Some uses shape detection technique [6], some uses marker detection [5] technique and some uses skin color detection to recognize hand gestures. All of the above techniques may involve only on or two fingers or all the five fingers or only the lap of the hand or the whole hand upto the arm of both hand to generate differnt hand gestures.

2.5 Tracking Techniques

Marker tracking technique can be described in two parts:

- Marker recognition : detection of the marker
- Tracking : detect marker in subsequent frame searching neighboring pixel

2.5.1 Marker Recognition Algorithms

Several Marker recognition algorithms are there

- 2D cross correlation
- Normalized correlation coefficient
- Template matching (pixel by pixel)
- Cam shift

2.5.1.1 2D Cross Correlation

The first step is to do a 2D cross correlation on the image using a predefined mask.All these products are added to obtain the correlation value. When the correlation value for a pixel is known, it is checked whether the value is within the expected range. If the value is within range, the coordinate of central pixel of that region is detected. The above approaches are done for each consecutive frames to track the marker. Each image element needs M2 real multiplications and M2 real summations. For the purpose of comparing the cross correlation methods, the assumption will be made that multiplication and summation operation take the same processor clock cycles. The total number of calculations for the total area can now be calculated as 2L2M2 [10]. 2D cross correlation can be done both in time domain and frequency domain. In the time domain cross correlation the cross correlation is done on each pixel in the search area using the following equation

$$R(h,k) = \sum_{y=0}^{y<l} \sum_{x=0}^{x<l} M(x,y)I(x+h, y+k) \tag{2.1}$$

Here $R(h,k)$ is the correlation value of the image point with coordinates (h,k) inside the search area. M is the the area of the image which must be correlated with the mask and l the size of the mask.

2.5.1.2 Normalized Correlation Coefficient

Idea of 2D cross correlation is integrated here. It uses the following formulas [4]

$$R(h,k) = \sum_{y=0}^{y<l} \sum_{x=0}^{x<l} M(x,y)I(x+h,y+k) \tag{2.2}$$

$$r(s,t) = \frac{A\sum s(x,y)t(x,y) - \sum s(x,y)\sum t(x,y)}{\sigma_s \sigma_t} \tag{2.3}$$

$$where, \sigma_s = \sqrt{A\sum s(x,y)^2 - (\sum s(x,y))^2}$$

$$\sigma_t = \sqrt{A\sum t(x,y)^2 - (\sum t(x,y))^2}$$

Here A is the number of pixels in template $t(x,y)$ and $s(x,y)$ is the sub image.

2.5.1.3 Template Matching

Similar to 2D cross correlation. Instead of multiplication operation it compares each pixel value of the template with sub image and keeps a counter to make decision. We discuss it in greater detail later.

2.5.1.4 Cam-Shift

It is an object tracking method which uses histogram back projection technique to detect a known object in an image. This technique can be very useful in skin and face recognition. To better understand Cam shift, it is required to know how histogram back projection works.

- Histogram back projection

  Histogram back projection answers a question,Where in the image are the colors that belong to the object being looked for ? The answer is given in such a way that the color that appears in other objects besides the target are deemphasized so that they are less likely to distract the search mechanism. In this technique the model(target) and the image are represented by their multidimensional color histograms M and I (M¡I) [11]. A ratio histogram R defined as, $R_i = min(\frac{M_i}{I_i}, 1)$

2.5.2 Mass Center Calculation

Using histogram back projection we are done with distinguishing our target object. For proper tracking we need to know the center of the object. Given that I(x,y) is the intensity of the discrete probability image at (x,y) within the search window we do

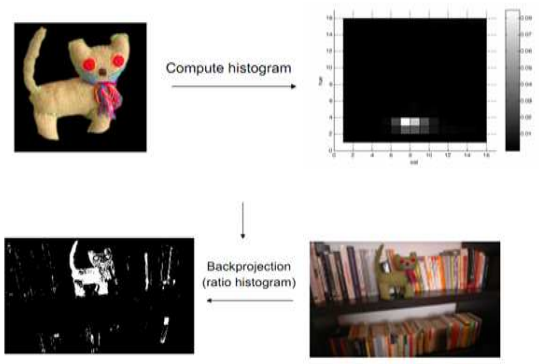

Figure 2.3: Example of histogram back projection [2]

Compute the zeroth moment,

$$M_{00} = \sum_x \sum_y I(x, y) \tag{2.4}$$

Find the first moment for x and y,

$$M_{00} = \sum_x \sum_y xI(x, y) \tag{2.5}$$

$$M_{01} = \sum_x \sum_y yI(x, y) \tag{2.6}$$

Compute the mean search window location,

$$x_c = \frac{M_{10}}{M_{00}}; y_c = \frac{M_{01}}{M_{00}} \tag{2.7}$$

2.5.3 Kalman Filter

We are using Kalman filter [12–14] to track the marker. In the figure 2.4 we have the prediction $\hat{y}^-(t_2)$ and the measurement $z(t_2)$. From these two values we get the optimal estimate of position. Kalman Filter first make prediction based on previous data $\hat{y}^-, \sigma^-$ then take the measurement z_k, σ_z. Then make optimal estimate.

$$Optimalestimate(y) = Prediction + (KalmanGain) * (Measurement - Prediction) \tag{2.8}$$

$$Varianceofestimate = Varianceofprediction * (1 - KalmanGain) \tag{2.9}$$

To predict use initial conditions and model (eg. constant velocity). Use measurement to correct prediction by blending prediction and residual and then optimal estimate with smaller variance.

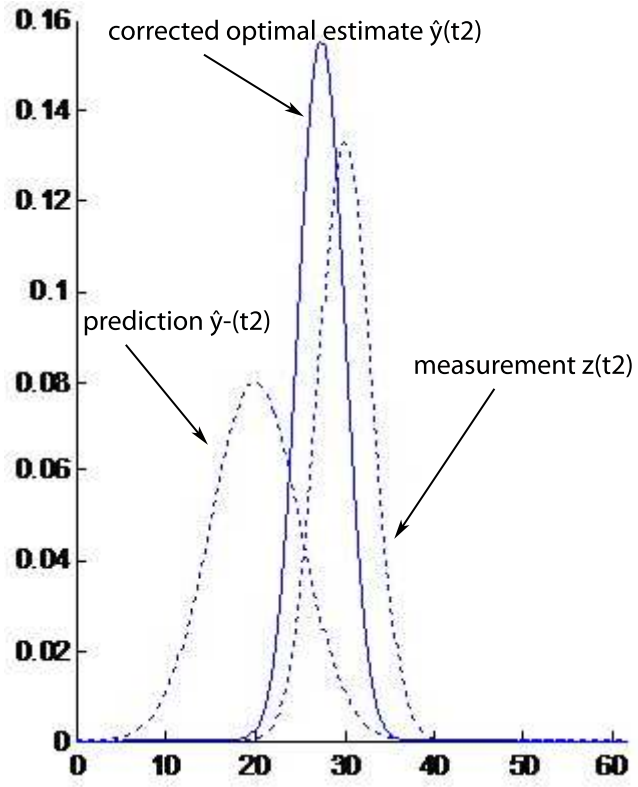

Figure 2.4: Corrected mean is the new optimal estimate of position New variance is smaller than either of the previous two variances

- Process to be estimated:

$$y_k = Ay_{k-1} + Bu_k + w_{k-1}\ where,\ w_{k-1} = Process\ Noise\ with\ covariance\ Q \quad (2.10)$$

$$z_k = Hy_k + v_k \qquad where,\ v_k = Measurement\ Noise\ with\ covariance\ R \quad (2.11)$$

- Theoretical Basis
  - Predicted: $\hat{y}_k^-$ is estimate based on measurements at previous time-steps

    1. Project the state ahead

$$\hat{y}_k^- = Ay_{k-1} + Bu_k \quad (2.12)$$

    2. Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q \quad (2.13)$$

  - Corrected: $\hat{y}_k$ has additional information the measurement at time k

    1. Compute the Kalman Gain

$$\hat{y}_k = \hat{y}_k + K(z_k - H\hat{y}_k) \quad (2.14)$$

    2. Update estimate with measurement z_k

$$K = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad (2.15)$$

    3. Update Error Covariance

$$P_k = (I - KH)P_k^- \quad (2.16)$$

- Blending Factor:
  - If we are sure about measurements:
    1. Measurement error covariance (R) decreases to zero
    2. K decreases and weights residual more heavily than prediction
  - If we are sure about prediction
    1. Prediction error covariance P-k decreases to zero
    2. K increases and weights prediction more heavily than residual

Chapter 3

Marker Based Tracker

3.1 Overall Concept

Several systems have been developed for the interaction with computer via hand gesture or gesture of other body parts of human. Some of those uses cam shift method, which is based on histogram back projection [11] technique, for tracking [5]. Again many tracking methods are based on shape detection i.e works of Chawalitsittikul and Suvonvorn [5], Manresa and et al [6]. We propose a method that uses the marker detection and tracking technique and for detection and tracking we use template matching algorithm. We only use two color markers namely red and green to generate seven hand gestures to give commands to the desktop or laptop computer that has a consumer grade web-cam which could be done by mouse or touch-pad. We have integrated sliding window mechanism with the template matching [15] to reduce its time complexity. For greater accuracy of detection and smooth movement of system cursor we have used kalman filter. We have worked with HSB color model with a view to reducing effect of different light intensity in the detection of the image. We have used only hue and saturation values to detect the object.

3.2 Proposed Tracking Algorithm

3.2.1 Choosing The Color Model

Our most important challenge is that our system should have to work in different environment with different light intensity. If we use the RGB model we will not be able to detect our marker in different light intensity. To work in different light intensity we took HSI model because H and S value are light invariant. We first get the r,g and b values from the input image then we converted it in h(Hue), s(saturation) and i(intensity) using the following equation.

3.2.2 Matching and Traversing Techniques

There are different type of matching techniques for marker recognition. Since marker has fixed color and shape, we can use template matching for detection. There are different types of

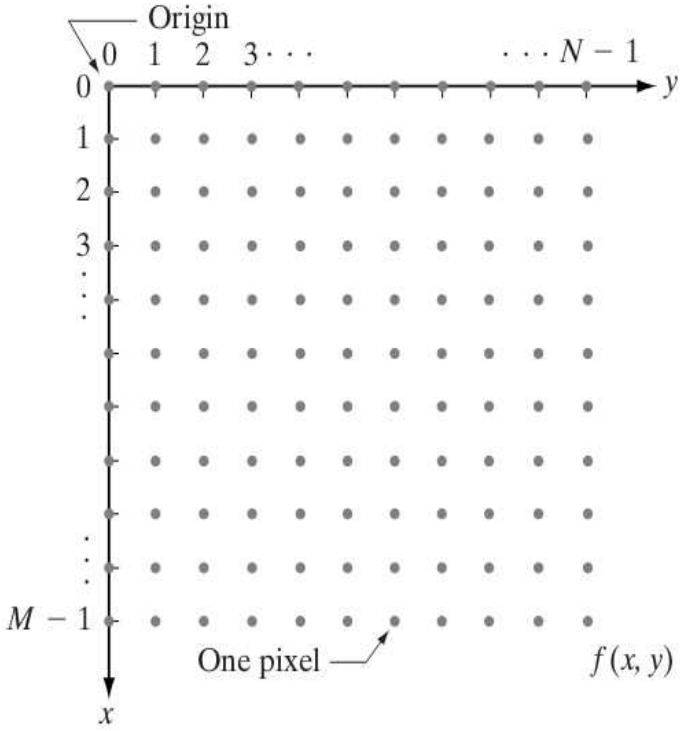

Figure 3.1: Co-ordinates of an input image

template matching discussed in section 2.5.1.NCC(Normalized Cross Correlation) is more time consuming than SSD(Sum of Squared Differences). For matching we have used SSD(Sum of Squared Differences) [16, 17]. Since we are using Hue and Saturation of the HSI model we have to use both Hue and Saturation in our SSD technique.Thats why we modified the equation of general SSD. For a particular pixel in a input image we will get one Hue and one Saturation value. We can find out the response value using the following equation 3.1. When a pixel is actually under the marker then its response value is be less otherwise it will give higher value (i.e) matching pixel will give lower response value and non matching pixel will give higher response value.Let's our input image size is M*N and our mask size is m*n. In figure 3.2.2 show the pixels of a image, horizontal axis is y and vertical axis x.

$$RV_{x,y} = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w1 * (H(x+s, y+b) - h)^2 + w2 * (S(x+s, y+b) - s)^2 \tag{3.1}$$

Where $RV_{x,y}$ is the response value for the point x,y in the input image,mask size is m*n.
$a = (m-1)/2$ and $b = (n-1)/2$ and w1 and w2 are weight co-efficients. In our implementation we use $w1 = w2 = 1$.

- N-th Distant Pixel Scanning:

  We are looking only at the nth distant pixels and calculating the response values.

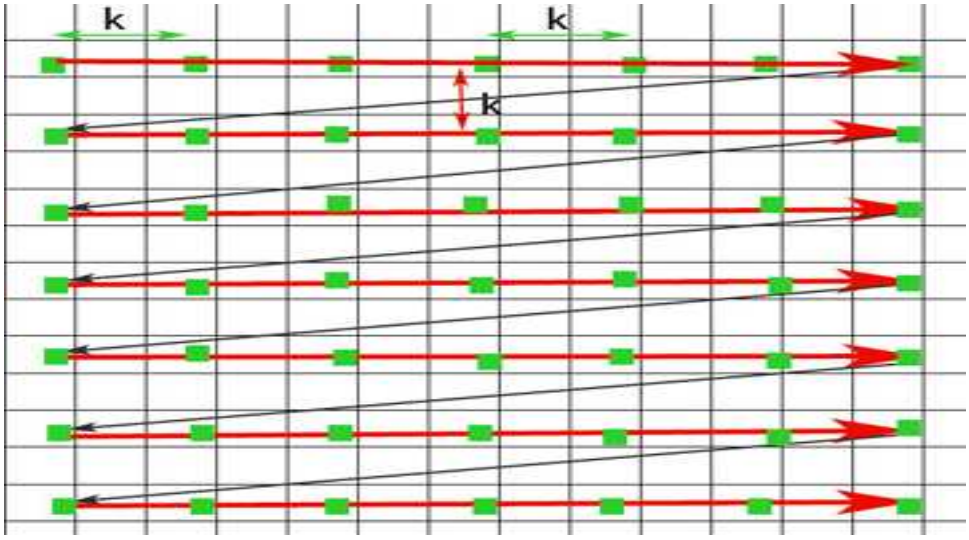

Figure 3.2: Raster Scanning

- Raster Scanning:

  For the first time when when the program wants to find the marker it starts its searching using raster scanning. From the left corner of the image to the right corner of the image. Figure 3.2 show the raster scanning technique.

- Circular Scanning:

  Circular scanning starts only It starts from the previous point and circularly scan the image and return the point if its response value is lower than the a particular response value which is called threshold response value. In circular scanning we define the the boundary and if all the points in the boundary is scanned and no pixel is found whose response value is lower than the threshold response value the circular scanning would be stopped and the marker is looked for by raster scanning. Figure 3.3 show the circular scanning technique.

3.2.3 Center of Mass Calculation

After detection of the marker we have to calculate the center of mass detailed is discussed in 2.5.2. If we do not detect the center of a marker the detection of the marker will not be accurate. Different pixels of a marker will be detected at different time and it will create wrong estimation of the marker position.

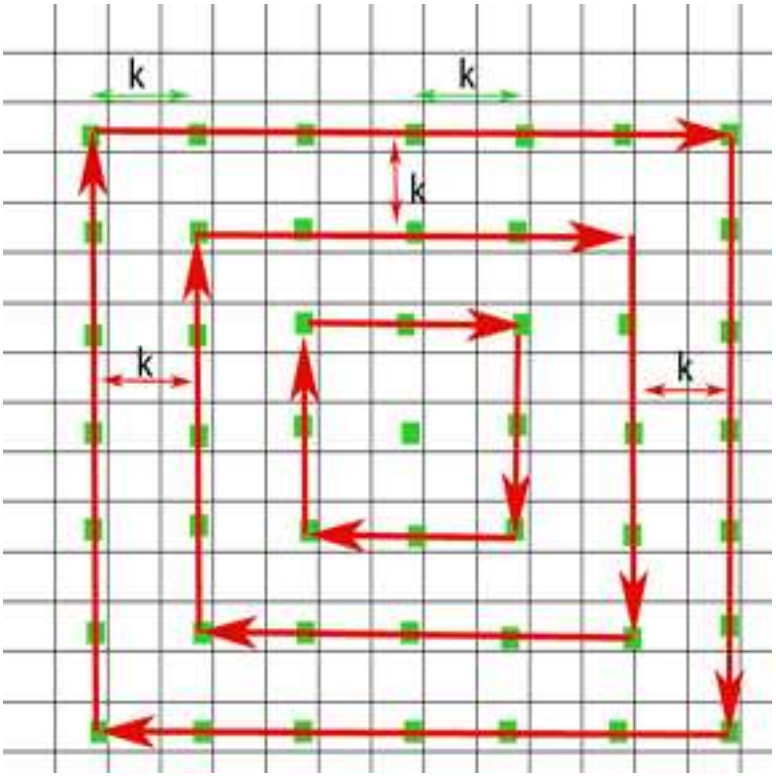

Figure 3.3: Circular Scanning

3.2.4 Sliding Window Mechanism

We are using using sliding window mechanism for calculating the response value. Using the equation 3.1 we can generally calculate the response value.

- Left to Right

  Let, our mask is moving from left to right. Since it is moving from left to right so its left pixel response value($RV_{x-1,y-1}$) value is known. We have to calculate the the current pixel response value($RV_{x,y}$). In figure 3.4 window is slided from left to right. Let we know the response value of the the left window in figure 3.4(a). Now the window is moved one pixel right in figure 3.4(b). Under this new window only the rightmost column is new and and which was the leftmost column in figure 3.4(a) is no more under in figure 3.4(b) so if we only add the rightmost column value and subtract the leftmost column value from the previous response value($RV_{x-1,y-1}$) we will get the new ($RV_{x,y}$). We can easily use the equation 3.2 to calculate the response value when the window is slided from left to right.

7	6	5	4	2	2	5	2
1	2	3	3	4	1	2	3
7	6	5	4	2	2	5	2
2	3	4	5	5	7	2	6
2	3	2	1	2	8	7	6
3	3	1	3	2	6	8	9
7	6	5	4	2	2	5	2

(a)

7	6	5	4	2	2	5	2
1	2	3	3	4	1	2	3
7	6	5	4	2	2	5	2
2	3	4	5	5	7	2	6
2	3	2	1	2	8	7	6
3	3	1	3	2	6	8	9
7	6	5	4	2	2	5	2

(b)

Figure 3.4:

$$
\begin{aligned}
add &= \sum_{s=-a}^{a}\sum_{t=1}^{incr}\{w1*(H(x+s,y+b+1-t)-h)^2+w2*(S(x+s,y+b+1-t)-s)^2\}\\
sub &= \sum_{s=-a}^{a}\sum_{t=1}^{incr}\{w1*(H(x+s,y-b-t)-h)^2+w2*(S(x+s,y-b-t)-s)^2\}\\
RV_{x,y} &= RV_{x,y-incr}+add-sub
\end{aligned}
\tag{3.2}
$$

- top bottom:

  In the same way described in section 3.2.4 when the window is slided from top to bottom we can use the equation 3.3 to calculate the response value.

$$
\begin{aligned}
add &= \sum_{s=1}^{incr}\sum_{t=-b}^{b}\{w1*(H(x+a+1-s,y+t)-h)^2+w2*(S(x+a+1-s,y+t)-s)^2\}\\
sub &= \sum_{s=1}^{incr}\sum_{t=-b}^{b}\{w1*(H(x-a-s,y+t)-h)^2+w2*(S(x-a-s,y+t)-s)^2\}\\
RV_{x,y} &= RV_{x+incr,y}+add-sub
\end{aligned}
\tag{3.3}
$$

- Right to left

  In the same way described in section 3.2.4 when the window is slided from right to left we

can use the equation 3.4 to calculate the response value.

$$
\begin{aligned}
add &= \sum_{s=-a}^{b} \sum_{t=1}^{incr} \{w1 * (H(x+s, y-b+1-t) - h)^2 + w2 * (S(x+s, y-b+1-t) - s)^2\} \\
sub &= \sum_{s=-a}^{-b} \sum_{t=1}^{incr} \{w1 * (H(x+s, y+b-t) - h)^2 + w2 * (S(x+s, y+b-t) - s)^2\} \\
RV_{x,y} &= RV_{x,y+incr} + add - sub
\end{aligned}
\tag{3.4}
$$

- Bottom to up

  In the same way described in section 3.2.4 when the window is slided from bottom to top we can use the equation 3.5 to calculate the response value.

$$
\begin{aligned}
add &= \sum_{s=1}^{incr} \sum_{t=-b}^{b} \{w1 * (H(x-a+1-s, y+t) - h)^2 + w2 * (S(x-a+1-s, y+t) - s)^2\} \\
sub &= \sum_{s=1}^{incr} \sum_{t=-b}^{b} \{w1 * (H(x+a-s, y+t) - h)^2 + w2 * (S(x+a+1-s, y+t) - s)^2\} \\
RV_{x,y} &= RV_{x-incr,y} + add - sub
\end{aligned}
\tag{3.5}
$$

3.2.5 Separating Marker From Background

To meet the challenge of separation of the marker from the background first of all we choose markers as uniformly colored object. Secondly we have defined specific range of size of the marker so that it can be distinguished from a uniformly same colored background as the marker. The range is say [a,b] pixels, where a is the minimum area for the marker and b is the maximum area of the marker. If the marker area is found to be greater than b we simply consider that as background of marker's color and hence no marker is detected. Again if the marker area is found to be less than a we consider that as a red dot in the background and hence no marker is detected.

3.2.6 Tracking Technique

We are using Kalman filter [12–14] to track the marker. Our final goal is to move the cursor smoothly. If we only use the detection points to move the cursor, the cursor will not move smoothly because our marker movements will have some small sharp jerk. If we do not use Kalman filter we will not able to place the cursor for a particular point for a second which is needed for our clicking purposes and most importantly we can not move the cursor in our desired

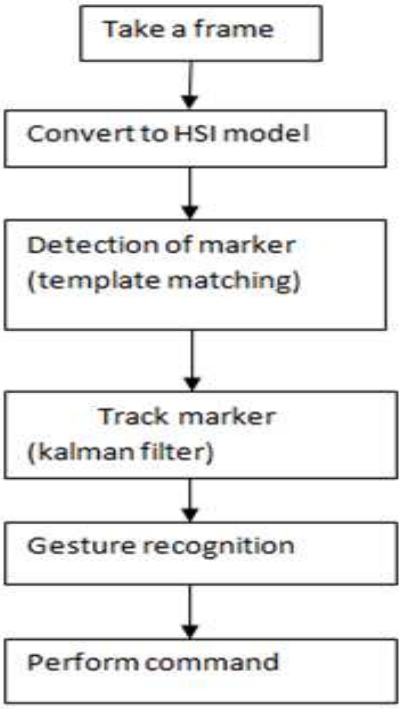

Figure 3.5: System analysis diagram of proposed method

way. If some points is miss detected and if we do not use Kalman filter the cursor will give a sudden jump for that miss detected point. Since kalman does not has any higher time and space complexity we can easily use this for our smooth detection purposes. Kalman filter is a recursive procedure it only remembers the previous state and predict the current state depending on the previous state and system models. Detailed of this is described in section 2.5.3.

3.2.7 System Overview

The figure 3.5 shows the system overview of our proposed method. First the web-cam takes a frame and converts it to HSI model that is we use the hue and saturation information of the image. Then the marker detection is done by the the template matching as we discussed earlier. The tracking is done with the help of Kalman filter. The gesture recognition is done as illustrated by figure 3.7. As soon as the gesture is recognized specific command is performed.

3.3 Proposed hand gestures

3.3.1 Cursor Move

The system cursor moves with the movement of red marker. If the red marker moves right the cursor would move right. If the red marker moves left the cursor would moves left.

3.3.2 Left Click

Only red marker is involved. The red marker has to be within a certain small region (actually user would try to keep the marker still) for some specified time i.e 2 seconds. If the red marker is then moved upward it is considered as left click.

3.3.3 Right Click

Only red marker is involved. The red marker has to be within a certain small region (actually user would try to keep the marker still) for some specified time i.e 2 seconds. If the red marker is then moved right it is considered as right click.

3.3.4 Double Click

Only red marker is involved. The red marker has to be within a certain small region (actually user would try to keep the marker still) for some specified time i.e 2 seconds. If the red marker is then moved downward it is considered as right click.

3.3.5 Zoom In

Both the green and red markers are involved. If the distance between the markers are increased it is considered as a zoom in command.

3.3.6 Zoom Out

Both the green and red markers are involved. If the distance between the markers are decreased it is considered as a zoom out command.

3.3.7 Forward

Only green marker is involved. The green marker has to be within a certain small region (actually user would try to keep the marker still) for some specified time i.e 2 seconds. If the green marker is then moved right it is considered as Forward command.

3.3.8 Backward

Only green marker is involved. The green marker has to be within a certain small region (actually user would try to keep the marker still) for some specified time i.e 2 seconds. If the green marker is then moved left it is considered as Backward command.

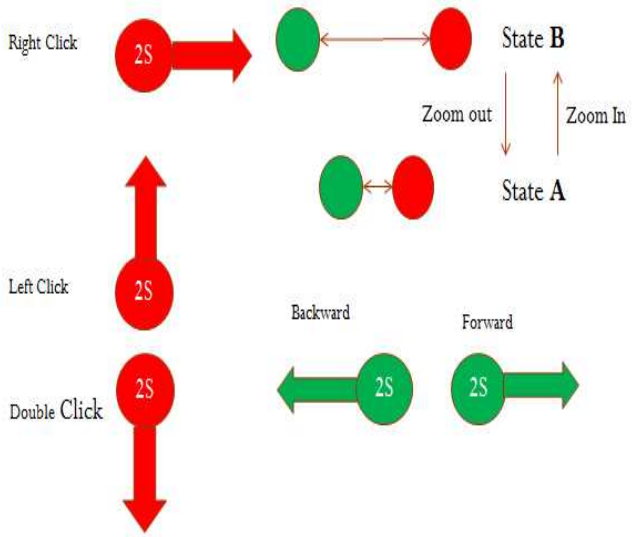

Figure 3.6: Proposed hand gestures

3.3.9 Gesture recognition

The figure 3.7 shows our logic to distinguish each gesture. Initially the system is at start state. The system looks for both red and green marker. If only red marker is detected the system goes to cursor move state. At this state if the system finds that the red marker is found within a certain region for a specified amount of time it saves the center of that region. Now if the cursor moves right the system goes to right click state (performs right click), else if the cursor moves upward the system moves to left click state (performs left click), else if the cursor moves downward the system moves to double click state (performs double click). If the system finds only the green marker while it is at start state the system checks whether the green marker is found within a certain region for a specified amount of time. If found the the system goes to Back/Forward state. Now if the green marker moves left the system moves to Backward state (performs backward command). Else if the green marker moves right the system moves to Forward state (performs forward command). At starting state if the system finds both red and green marker the system moves to Zoom in or Zoom out state. If the distance between two markers are decreased the system moves to Zoom out state (performs zoom out command). Else if the distance between two markers are increased the system moves to Zoom in state (performs zoom in command).

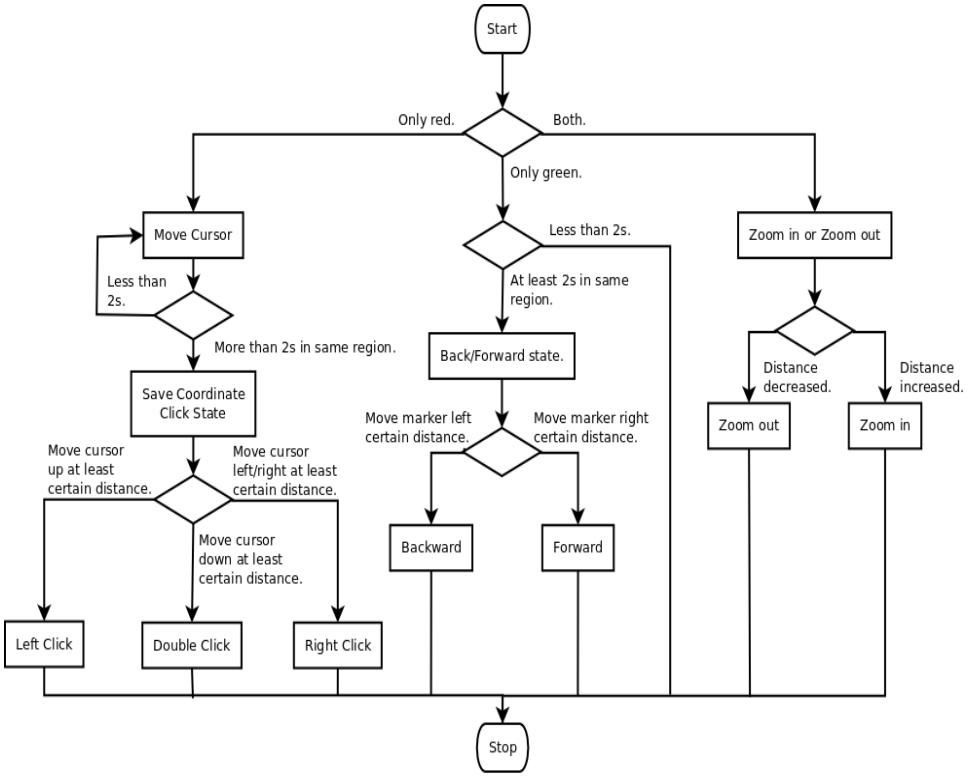

Figure 3.7: System analysis diagram for gesture recognition

Chapter 4

Experimental Analysis

4.1 Experimental Setup

Required hard wares are a desktop and a web-cam or a laptop that has a web-cam with it and red and green colored marker. We have implemented our system in Java. The laptop in which we did our experiment had core-i5 processor and 4 GB of RAM. The web-cam was hp laptop web-cam. We did the experiments at both day light and under the light of tube lights at night.

4.2 Performance Analysis

4.2.1 Average Performance

The table 4.1 shows the average performance. We randomly chose consecutive 20 frames where no miss detection occurred and we calculated the velocity of marker in those frames and the required time to detect the marker as well. We can easily observe from table 4.1 that if the average velocity of marker is about 716 pixel per second then the number of miss detection is almost zero. And at this average velocity our average detection time is 0.0251 second.

4.2.1.1 Impact of Using kalman Filter

In the figure 4.1 the green line shows the actual position of the marker, the blue line shows the position of the system cursor and the red line shows the position of the system cursor if we did not use the kalman filter. the red line is little up and down that shows some jerking that occurs in the absence of kalman filter. On the other hand the blue line is quite smooth as is the green line. This smoothness makes the blue line more similar to the green line which results in good gesture generation.

4.2.1.2 Error Rate Between Experimental Detection and Actual Position

In the table 4.2 the 2nd and 3rd columns show the position of the center of the marker calculated by our proposed technique and the 4th and 5th columns show the real position of the center of the marker. The average distance from the calculated position and the real position of the marker

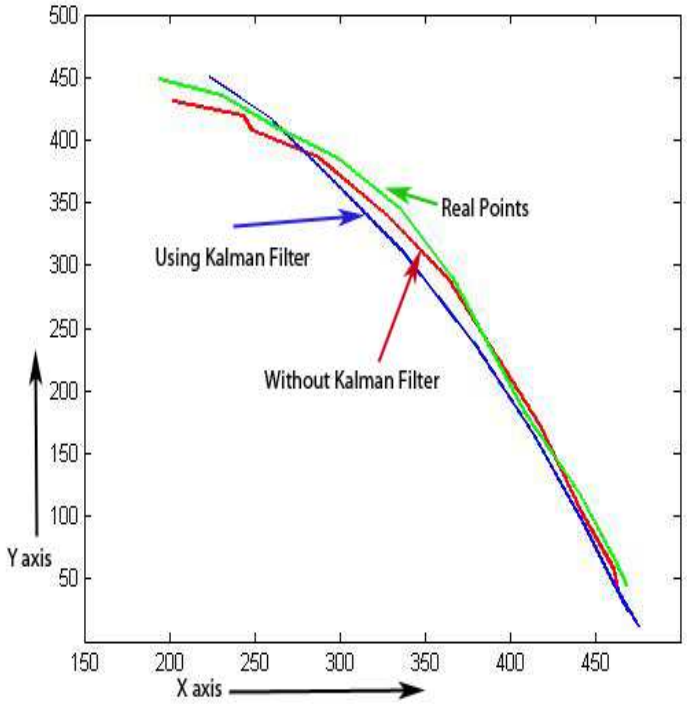

Figure 4.1: Comparing effect of kalman filter in miss detection

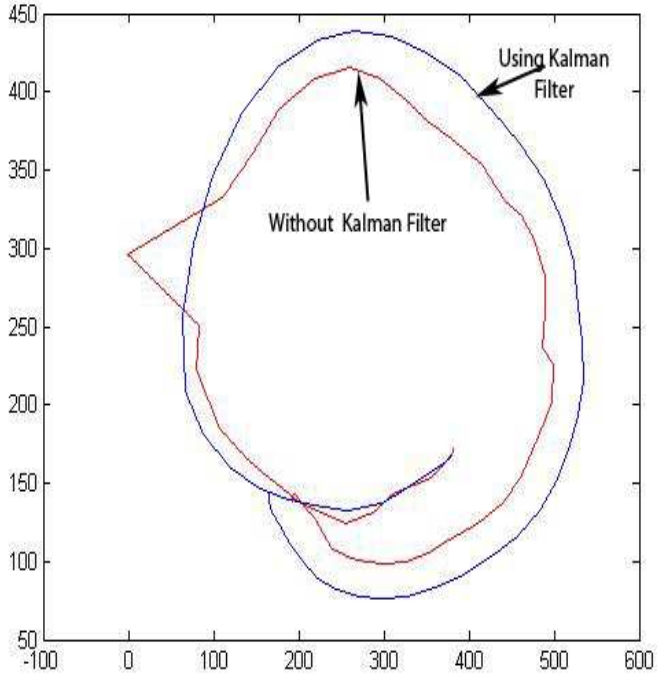

Figure 4.2: Kalman filter helps user to generate smooth gesture. In this case user intended to generate a circle shaped gesture.

Table 4.1: Average detection time of the marker in different velocity

Frame no	Time(s)	Velocity(pixel/s)
40	0.108	623.862
41	0.091	1114.26
42	0.047	1591.93
43	0.006	1206.04
44	0.009	713.616
45	0.027	355.866
46	0.005	52.4845
47	0.028	374.861
48	0.009	669.166
49	0	676.754
50	0.005	767.885
51	0.004	583.323
52	0.005	302.706
53	0.011	456.814
54	0.007	512.144
55	0.096	410.164
56	0.015	2375.09
57	0.011	679.471
58	0.012	509.727
59	0.006	357.03
Avg	0.0251	716.659

is 34 pixels approximately. But this is not a great problem for the tracking and generation of the gestures as the 6th column of table 4.1 shows that the standard deviation of the error is 8.27 which is quite small. This standard deviation shows that the difference of the actual position and the calculated position of the marker remains quite constant and hence cannot create much problem in generation of gestures.

4.2.2 Miss Detection Vs Velocity

In table 4.3 we show the effect of velocity of the movement of the hand of the user on the detection of the marker. In each row the second sub row shows the position of the mis detected marker at time t. The first sub row shows the position of the detected marker at time t_1. The

Table 4.2: Error rate between experimental detection and actual position

frame no	experimental X	experimental Y	Real X	Real Y	Error(pixel)	σ of error
1	476	13	469	46	33.73	
2	470	25	462	65	40.79	
3	261	416	231	436	36.05	8.27
4	224	451	194	449	30.06	
5	295	310	320	350	32.02	

Table 4.3: Miss detection vs velocity

time(s)	X(pixel)	Y(pixel)	Velocity(pixel/s)
4.915	304	277	772.495
5.036	220	236	
5.506	117	264	985.888
5.752	359	248	
8.523	195	212	1077.01
8.744	198	450	
9.810	304	277	890.347
10.026	220	450	
11.812	423	256	793.206
12.011	363	110	

fourth column shows the velocity which is responsible for miss detection. The average velocity responsible for miss detection is greater than or equal to 903.79 pixel/s.

4.2.3 Overcoming Miss Detection

The table 4.4 shows the effect of the boundary which has been discussed in the section 3.2.3. The second column shows the time required to re detect the miss detected marker in second when we do not use the boundary for circular scanning. The third column shows the the time required to re detect the miss detected marker in second when we use the the boundary for circular scanning. We can easily observe that the improvement is almost 12% if we use the boundary in the circular scanning.

Table 4.4: Time required to detect after miss detection

frame no	without boundary box(s)	with boundary box(s)
1	0.124	0.083
2	0.097	0.096
3	01	0.11
4	0.095	0.08
5	0.117	0.082
6	0.089	0.085
7	0.1	0.087
8	0.127	0.081
9	0.08	0.09
10	0.1	0.086
Avg	0.1029	0.0884

4.2.4 Gesture Recognition Accuracy

Gesture recognition accuracy is the comparison between the gesture formed by our tracking technique and the actual gesture that user intended to generate. From figure 4.1 we can observe that the green line and the blue line generates quite similar gesture. Although due to use to of kalman filter some gesture like circle in a grater velocity may not be that much accurate as the other gestures, our proposed hand gestures are not hampered by the kalman filter. Of course in normal velocity like some velocity less the 700 pixel per second circle shaped gesture generation is also possible.

4.2.5 User-Friendliness

To analyze the user-friendliness we asked five users to use our system and perform the commands cursor move, left click, right click of mouse, zoom in, zoom out, Forward and backward. Each user was asked to perform each command 30 times. In the first 10 attempts the overall accuracy was 59.38% as shows the table 4.5. From the table 4.5 we can also see the average accuracy of each command which is at the bottom line of the table. We can also observe that the average accuracy for all the command are 64.2% , 61.4% , 55.7% , 62.8% , 52.8% for the user 1, 2, 3, 4 and 5 respectively. Table 4.6 and 4.7 also illustrate similar informations as mentioned for table 4.5 for the next 10 attempts and next to next 10 attempts respectively. From the table 4.5 ,4.6 and 4.5 we can easily observe that average performance for each command increases as user tries the same command again and again. After 30th attempt the over all average accuracy is 79.4%.

Main point is as much as user will use the system the performance will be increased.

Table 4.5: User performance in 1st attempt

User	Cursor move	Left click	Right click	Zoom in	Zoom out	Forward	backward	Average
1	10/10	5/10	7/10	6/10	6/10	6/10	5/10	64.2%
2	10/10	6/10	5/10	6/10	7/10	5/10	6/10	61.4%
3	9/10	5/10	6/10	7/10	5/10	4/10	5/10	55.7%
4	10/10	5/10	6/10	5/10	5/10	6/10	7/10	62.8%
5	9/10	4/10	5/10	5/10	4/10	5/10	5/10	52.8%
avg	96%	50%	58%	58%	54%	52%	56%	59.38%

Table 4.6: User performance in 2nd attempt

User	Cursor move	Left click	Right click	Zoom in	Zoom out	Forward	backward	Average
1	10/10	6/10	8/10	7/10	7/10	7/10	7/10	75.7%
2	10/10	7/10	6/10	6/10	7/10	6/10	7/10	70.0%
3	10/10	6/10	6/10	7/10	6/10	5/10	5/10	64.2%
4	10/10	6/10	6/10	6/10	7/10	7/10	7/10	70.0%
5	9/10	5/10	6/10	5/10	5/10	6/10	6/10	60.0%
avg	98%	60%	64%	62%	64%	62%	64%	67.98%

Table 4.7: User performance 3rd attempt

User	Cursor move	Left click	Right click	Zoom in	Zoom out	Forward	backward	Average
1	10/10	9/10	9/10	8/10	9/10	8/10	7/10	85.7%
2	10/10	8/10	7/10	7/10	8/10	8/10	7/10	78.5%
3	10/10	7/10	7/10	8/10	8/10	7/10	7/10	77.1%
4	10/10	7/10	8/10	7/10	8/10	9/10	8/10	81.4%
5	10/10	7/10	7/10	7/10	6/10	7/10	8/10	74.3%
avg	100%	76%	76%	74%	78%	78%	74%	79.4%

Chapter 5

Conclusion

Our moto was to develop a cheap hand gesture based input technique. In our proposed method we have used template matching which has been modified with the integration of sliding window technique. We have modified scanning technique of usual template matching as well. Our proposed method processes each image in average at the rate of 1 frame per .025 second which makes it possible to interact with computer in real time. Use of kalman filter helped a lot in generation of smooth gestures. We are using marker which is very available and only the web cam that comes with the lap top is enough to take images. So our system is very cheap. We have used HSI color model instead of RGB color model that has provided better performance in different light intensity. The performance of the system in velocity of some thing around 700 pixel/second is quite satisfactory. We have also checked the user friendliness of our system which is quite satisfactory. So far we have worked with 2D images and 2D hand gestures it can also be possible to work with 3D hand gestures if depth information is available.

Bibliography

[1] R. E. W. Rafael C. Gonzalez, *Digital Image Processing*, 2002.

[2] 8/9/2012. [Online]. Available: http://www.lira.dist.unige.it/teaching/SINA/slides-current/image-processing-point.pdf

[3] 8/9/2012. [Online]. Available: http://www.pranavmistry.com/projects/sixthsense/

[4] I. J. G. M. I. Margrit Betke, Member and P. Fleming, "The camera mouse: Visual tracking of body," *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING, VOL 10, NO 1,*, 2012.

[5] N. S. Pongsatorn Chawalitsittikul, "Real time hand marker tracking as a user input device."

[6] R. M. Cristina Manresa, Javier Varona and F. J. Perales, "Real time hand tracking and gesture recognition for human-computer interaction," 2000.

[7] 7/9/2012. [Online]. Available: http://tinyurl.com/9vmk9qg

[8] 7/9/2012. [Online]. Available: http://searchciomidmarket.techtarget.com/definition/HCI

[9] J. A. S. Fakhreddine Karray, Milad Alemzadeh and M. N. Arab, "Human-computer interaction: Overview on state of the art," 2012.

[10] 2012. [Online]. Available: http://tinyurl.com/94s55vj

[11] D. H. B. MICHAEL J. SWAIN, "Color indexing," *International journal of Computer Vision, 7:1, 11-32*, 1991.

[12] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transaction of the ASME–Journal of Basic Engineering*, March 1960.

[13] P. S. Maybeck, "Stochastic models, estimation, and control, volume 1," *Academic Press, Inc*, 1979.

[14] G. Welch and G. . Bishop, "An introduction to the kalman filter." [Online]. Available: http://www.cs.unc.edu/ welch/kalman/

[15] J. P. Lewis, "Fast template matching," *Vision Interface 95, Canadian Image Processing and Pattern Recognition Society, Quebec City, Canada*, May 15-19, 1995, p. 120-123.

[16] K. G. Derpanis, "Relationship between the sum of squared difference (ssd) and cross correlation for template matching," *York University*, December 2005.

[17] "Fast template matching method based optimized sum of absolute difference algorithm for face localization," *International Journal of Computer Applications (0975 8887)*, vol. Volume 18 No.8, March 2011.