CHALMERS

2D Finger Motion Tracking, Implementation for Android Based Smartphones

Master's Thesis in the Intelligent Systems Design

Master of Science Thesis. Thesis work in Intelligent Systems Design

Maryam Khosravi Nahouji

Department of Applied Information Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2009
Report No. 2012:001
ISSN: 1651-4769

MASTER'S THESIS 2012:001

2D Finger Motion Tracking, Implementation for Android Based Smartphones

2D Finger Motion Tracking, Implementation for Android Based Smartphones

Master's Thesis in the Intelligent Systems Design

MARYAM KHOSRAVI NAHOUJI

Department of Applied Information Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012

2D Finger Motion Tracking, Implementation for Android Based Smartphone
Master's Thesis in the Intelligent Systems Design

MARYAM KHOSRAVI NAHOUJI

Master's Thesis 2012:001
Department of Applied Information Technology
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Chalmers University of Technology, Department of Applied Information Technology
Göteborg, Sweden 2012

2D Finger Motion Tracking, Implementation for Android Based Smartphone

Master's Thesis in the Intelligent Systems Design

MARYAM KHOSRAVI NAHOUJI
Department of Applied Information Technology
Chalmers University of Technology

ABSTRACT

Human computer interaction grows day by day and nowadays it has been entered many aspects of the life. Through this growth it has covered not only the desktop and laptop PCs but also mobile phones and specifically Smartphones. Therefore it worth to consider this topic and try to make a new functionality for these kinds of interactions. The intention is to find a practical way to make a new use of the video camera of the Android based Smartphones and track a single finger using the camera as sensor. To track an object in Android platform it needs some special prerequisite. It means that there should be a way to make the tracking applicable on Smartphones. for this purpose the solution this work has come up with is to use OpenCV library which is an open source library specialized for computer vision applications. This library is applicable on Android platform and therefore can be considered for our intention. Also considering different methods applied for hand tracking purpose in earlier works which has been studied in this work, the intention was to implement a method which was not been used by later studied works. Therefore the method of tracking used in this work is called "Motion Templates". This method is a method represented by OpenCV library and used in this work. To reduce the complexity of the work the first mile-stone which is implemented in this work is to track the finger on PC and then try to port the method and algorithm to Android platform.

The tracking is done successfully on the PC using this method and it can be recommended as a possible way of tracking to be considered on Android platforms as a future work.

Key words:

Detection, Tracking, Android, Smartphone

Contents

Preface

The preface should be on a right hand page. Therefore, this page should have an odd (roman) page number. If necessary, insert a blank page before the preface. Example of preface text:

In this study, an investigation is done on the practical methods of finger movement tracking for Android base Smartphones. The study has been carried out from April 2011 to January 2012. The work is a research project concerning methods of tracking applicable for Android base Smartphones. The project is carried out at the Department of Applied Information Technology, IT University, Chalmers University of Technology, Sweden.

The project has been carried out with Maryam Khosravi Nahouji as Master student and Associate Professor Morten Fjeld as supervisor. My supervisor Morten Fjeld is highly appreciated for his comments on my work. I would like to thank my family for their supports during my study. I also thanks the ones of my good friends who have given me help while I was doing my thesis work.

Göteborg January 2012

Maryam Khosravi

1 Introduction

It is not more than few decades that computers have entered into the human's life and in fact the phrase "personal computer" is not a very ancient phrase!

From that time computers importance in our lives have been growing fast and their popularity has been rising in the same speed. Today it is almost impossible to manage our everyday tasks without the help of computers and they are deeply rooted in our daily lives. Computers have been maturing quickly, their functionalities have been growing and their impact on human's life has been increasing. One could say that the growth of the computers has being too fast for some of us, creating a gap between generations when it comes to the knowledge of how to use them.

With the rise of internet, computers took the next step and went from being a personal computing device to the connectivity hub to the mass of information available anywhere. This connectivity is bringing us closer to each other, entertaining us in a way that was never possible before and changes our behaviour in terms of how we socialize, do shopping, and more.

While computers have become faster, smaller and cheaper, the ways of interacting with them have been basically unchanged. A keyboard, small or big has been the main interaction interface between us and computers. Researchers have been trying to change the ways of interaction between the human and the machine more human like.

Computers are today following us wherever we go. The interaction is not limited between us and our personal desk top or laptop computers anymore. The new portable devices, for example smartphones put more requirements on the ways of interaction between the machine and us. A big keyboard is no longer a solution. A small keyboard is not satisfactory. We need ways more human like. We would like to use gestures and even our language to communicate with our new devices.

Touch screens, methods of face recognition, speech control, are examples of wide variety of new ways of interaction between the computers and people. Interactive games are one of the examples in this category. Microsoft Kinect has taken one more step and made it possible to interact with the computer games through body movements. Also a popular game named Fruit Ninja could be played by simple movement of one finger.

My attempt in this thesis work is to contribute to the huge efforts that are made to improve our way to interact with our portable computers, smartphones.

2 Objective

One of the examples of human computer interaction is body movement tracking and more specifically hand motion tracking. There are many applications where body motion tracking is a useful way of interaction with the computers. There are also applications where body motion is a main purpose. Specific application in which this is a main core for; is robotics, security services (smart surveillance) (places like banks, public parking, department stores), interactive games, sport science and medical diagnosis.

Body motion tracking is about tracking of a specified part of the body like tracking motion of eye, head, hand, feet and etc. Obviously tracking one special part of the body consequents in less complexity and is possible with easier methods.

The objective of any motion tracking is different from case to case and for different applications. Based on the features of one specific application and also based on the situations where the specific application is used, the model of tracking might be different.

For example, if the tracking is done for real time applications the real time aspects of the interaction is important and relevant while for non real time cases the real time concerns are not an issue.

One of the concerns in real time applications is the usage of camera for tracking purpose. The first issue is the number of cameras used for the work. The second is the issue of the stability of the cameras which means that if it is necessary to use dynamic cameras to accomplish the aim or stationary cameras can fulfil the requirements? As much as the situation of the work is more sophisticated, the issues should be taken into consideration would be more and it needs to find more accurate ways of implementation.

As mentioned before, hand motion tracking is a special case of body motion tracking. Hand motion tracking can be considered for different purposes and therefore it can have different functionalities. Followings are few examples where hand tracking is useful.

2.1 Gesture recognition

Gesture recognition is an advanced form of movement tracking. Through gesture recognition we could for example recognize sign language and translate the signs into text. One of the abilities we can earn if we can track the movement of our hand is “gesture recognition”. For gesture recognition, it is needed to track the hand motion in 3D.

2.2 Virtual mouse

The other functionality that can be considered for hand motion tracking is to use hand as a virtual mouse. Tacking the motion of the hand, could replace the need of computer mouse. [1]

2.3 Video games

Video games are another class of applications where hand gesture recognition can be applied to. The purpose is to interact with the game to control the game. [2]

These examples are some of the applications for hand motion tracking. As the number of applications grow and as the computers role in our lives gets more important, human-computer interaction plays increasingly more important role and the usages of hand tracking is becoming more relevant.

Nowadays this tracking can be considered not only to interact with computers but also to interact with other PC base devices such as smartphones. This is a new device that needs different ways of interaction and it is worthwhile to think of.

Considering the fact that nowadays the majority of laptops and mobile phones are equipped with video camera, these cameras could potentially be used as a human-machine interaction interface. This might not replace the traditional interfaces, but could be complementary to touch screens and the mouse. This interface will come more into attention in the cases where there is lack of commonly used devices like mice or even touch screens. Portable devices like smartphones are example of these cases.

The aim of this thesis work is to study and suggest practical ways to make use of phone cameras as a human-machine interaction interface. The intention is to track the movement of human hand/ finger using these phone cameras.

3 Problem Description

Smartphones are the new generation of mobile phones. Smartphones have made a big difference in the world. Nowadays the market of smartphones are growing fast and all the older generations of mobile phones are leaving their place to be replaced by this new generation.

Smartphones functionalities are much more than what regular phones are expected to do. In fact, smartphones today are much more powerful than what we used to call computers just some years ago. Smartphones have enough computing power for advanced graphical games. They are good for internet access, playing video, taking qualitative photo pictures and so on. As the name claims, this generation of mobile phones are getting smarter.

These extra features assign the mobile phones new and much more important roles compared with the traditional mobile phones. Nowadays mobile phones are getting the role of computers in many cases and it is possible to do the same tasks in both.

Smartphones need to be equipped with similar accessories of a typical computer such as mouse and keyboard. Touch screens are the alternative for mice in mobile phones and the hardware keyboards are replaced by touch screen keyboards. But when it comes to the generation of smartphones why not make them even smarter by using their already existed facilities to make new functionalities and make the world easier for human being?

The idea is to use the video cameras which almost all the smartphones are equipped with to track the finger of the user. This tracking is to make it possible to interact with the mobile phone by moving a finger. Potentially this means that touch screens might not be needed any more. It means that not only the touch ability of the screens might not be needed any more but also the difficulties some people might have using these screens will disappear.

The problem is to find a practical way to track the user's finger using the existing cameras. The main problems that this thesis work aims to resolve are:

1. What are the possible methods for detecting and tracking finger movement?
2. How it is possible to implement the methods suitable for Android based smart phone?

In the next chapter these issues are taken into consideration and are discussed.

4 Video Processing

In this chapter first the concept of video processing is defined. Then a brief introduction to different methods of video processing is given.

4.1 What is video processing?

Video processing is a special case of signal processing where both the input and output signals are video stream. [3] (A video signal is a sequence of 2D images which are projected from a dynamic 3D scene onto the image frame of a video camera.)[4] Each image is called a video frame.

The applications of video processing techniques are television sets, DVDs, VCRs, video codecs, video scalars and etc.) [3]

Video processing can be done both on analog and digital video streams. An analog video stream is an analog signal which is continuous in time. A Digital video stream is a physical signal which is a sequence of discrete values. Also the processing could be either in real time or in a saved video file.

As it mentioned video processing is done for different purposes varied from medical, security, traffic control (civil), intelligent transportation, industry, to entertainment and amusement.

Depending on the application; method of processing might differs from case to case. These methods might have some steps in common, but what is common in all video processing works, is to detect an object or objects and take the required information out of it. Sometimes it is possible to use the same methods for different applications.

Considering another concept called "Computer vision"; it is a science in which it is possible to visualize objects of the environment by means of computer and this is done by extracting information from images or frames of videos. Therefore it is possible to say that the mentioned applications of video processing are examples of computer vision science.

4.2 Methods of video processing

Since then there have been so many works done in video processing. Consequently there are many different methods of video processing that due to the case of work and the desired goal, one or a few numbers of methods are suitable for any special purpose.

This report does not intend to go through all the methods and give detail about different methods of video processing. Here just a very brief description of studied methods for this work, is given to clarify and give an insight about video processing.

The methods studied for this thesis work are mostly limited to methods of tracking in which the motion of a target object is tracked. There are a variety of methods for tracking that depend on the purpose of the tracking and the situation some of these methods are applicable in each case.

If the tracking applications are narrowed down to hand motion tracking which is the aim of this project, the following methods will be discussed as possible or implemented methods for hand tracking.

4.2.1 External object

One way to track the movement of the hand is to equip the hand with an external object. This eases detection of the hand from the surrounding environment and therefore tracking is easier.

To do so, using colours for detection purpose is an option. There are some research studies for hand tracking using colours. In fact colour coding and colour tracking algorithms are well-known for tracking purposes in many cases.

To apply colour coding and tracking methods for hand motion tracking application, a good way is to use coloured gloves. The reason of using "coloured" gloves is that generally detection of one or few specific target colours makes the work much easier. Wearing these gloves or coding fingers with different colours give possibility to segment the hand from the back ground. Therefore detection and consequently the tracking will be easier.

One of the methods suitable for this way of tracking is "Blob Detection". Blob detection method is useful when the aim is to detect the areas that are brighter or darker from their surroundings in the image or frame. To detect the colours, each colour is considered as a blob and the algorithm searches for each blob (colour). Generally blob detectors are a good option if the tracking is colour-coded base.

4.2.2 Mean-shift tracking

The other way of tracking is to use "Mean-shift Tracking" method. "Mean-shift is a nonparametric statistical method which shifts each data point to the local maximum in its neighbourhood." [5] This method of tracking does not rely on the colour of the objects but on the density of the pixels. "This method gives a nonparametric density estimate from sample set $\{x_i, 1 \leq i \leq n\}$. The estimation is done with little prior belief and can be used for any density estimate."[5]

This method is applicable in many object tracking applications. Using this method, the image is segmented to different parts and then the desired part is tracked. The method is useful for both colour and gray scale images.

4.2.3 Skin colour detection

One another technique for detection and tracking hand movement is methods which are based on skin colour segmentation. There are many different research studies which their fundamental idea of research is based on skin colour detection. Like other mentioned methods, skin colour detection and tracking methods are applicable in different cases like as security systems, medical diagnostics, retrieval systems and hand tracking for human computer interaction that is our case of work.

(Generally skin colour distribution modelling and segmentation are categorized in three different main groups.

4.2.3.1 Region-based methods

According to region based method used in [6], the results show that this method is suitable for detection of the skin colour in images or for pre-processing steps of video tracking. Pre-processing step is when the video frames get ready to be used for the main processing step where the images are processed to gain the desired outcome.

Therefore these methods are not suitable for real time tracking applications. The reason is that this method detects all types of skin colour tones containing the colours similar to skin colour in the image. Consequently it con not distinguishes between objects moving across the camera and other same colour objects in the back ground. It means it is not possible to track the motion, using this method.

4.2.3.2 Nonparametric methods

These methods are histogram-based. In this category of methods the distribution of the skin colour is estimated out of a set of training data set. In this category of methods there is not any significant model of skin colour for correction of the training data set. The advantages of these methods is that they are fast in computation also they are independent of the distribution shape and colour space selection. But the drawback of them is that they use a large space of memory and also they do not have the ability of correction of training samples.

4.2.3.3 Parametric methods

The last category of skin colour modelling methods is parametric methods. This group of methods uses a single or a multiple Gaussian distributing to approximate the distribution of the skin colour. This modelling gives the possibility to generalize the incomplete training data set. Advantages of the parametric methods are that they use small memory space and they have the ability to interpolate the incomplete training data set but their disadvantages

are that they are slow and they are distribution shape dependent and also they have higher false positive rate.) [6]

The Gaussian model distributions used in these methods are 2D or 3D. Generally in these methods any kind of k-means or c-means clustering algorithms is used for classification of the pixels in the image.

According to [6], the parametric and non-parametric methods have the same results in implementation.

What is given are some methods studied as a literature study for this work. There are many other research works or applications that have used other methods or a combination of presented methods. This thesis work has not possibility to go through the other methods.

In the next chapter some earlier research works done in the field of hand motion tracking are introduced and a brief description is given about them.

5 Earlier Work

There are many works done in the field of hand motion tracking. These works have been done for different applications and they have been implemented with different methods of tracking. Here some earlier works that are studied in this area are introduced.

5.1 Chasing the colour glove

One of the projects done in the field of hand movement tracking is a work done at "Simon Fraser University". The work is called "Chasing the Colour Glove: The Visual Hand Tracking." This work is the first work published in the area of real-time hand tracking done in 1994. [7]

This work is based on the colour coded methods of detection. The work intended to design an algorithm to be applicable on a created model to determine 3D gesture of the hand from 2D input images. The aim of this project is to make a computer interface for sign language signers.

(With help of a colour glove, the colour-coded joints of the fingers are detected as clues. Then a simple skeleton-model of the hand is fitted to the joint locations in the image that are given as clues to the system. This way the 3D shape and its motion are founded by the system.) [7] This work is not in real time.

5.2 Real time 3D hand interaction

The second work is a thesis work done at Chalmers University of Technology in 2007. The work is called "Real-Time 3D Hand Interaction: Single Webcam Low-Cost Approach". [8]

Same as previous work in this project also it has taken help from colours for detection purpose. The project uses colour coded fingers and pattern-based/ colour-based coding for palm and back of the hand to detect and track the moving hand.

As [8] claims pattern-based algorithms are often more exact for tracking purpose. The reason is that they can be fitted onto the palm and back of the hand easily but the problem is that patterns are difficult to be fitted on fingers such that fingers could be tracked by means of typical low-end cameras.

The technique used in this work is blob detection and blob tracking. The algorithm searches for a blob for each colour. Each colour blob is representative of one finger. Also two different coloured markers are used for detection of the palm and back and to measure their orientation. (This approach uses hand pose classification in hand coordinate system rather that webcam coordinate system which is independent in the webcam system. This causes to computation reduction and removes the training need.

The aim of this project is to provide a library that can be used for any 3D application using one or two hands.) [8]

5.3 Real time 3D hand computer interaction

The other project presented here as an earlier work is another thesis work done at Chalmers University of Technology in 2008. The project name is "Real-Time 3D Hand-Computer Interaction: Optimization and Complexity Reduction". [9]

The strategy used in this project for detection and tracking of the hand is almost the same as last work. Here just the hands are equipped with colour coded gloves. (Also a bracelet is used in order to determine the bounding borders of the hand. In this way the 3D position of the hand is determined by combination of the position of the bracelet and palm marker information.) [9] Palm marker is also used for checking the orientation of the hand.

The aim of this project is to develop an algorithm that is capable of use for real time tracking in a way that several frames per second can be processed using minimal system resources.

As it is seen all these works are done based on the colour marker coding methods. Surely there are other works using other methods of tracking. Here just these three are introduced as a sample research work done for hand motion tracking.

5.4 Real time hand-tracking with a colour glove

The last work studied in the area of hand tracking presented in this project is the work done in 2009. The work is called "Real-Time Hand-Tracking with a Colour Glove." [15] In this work the tracking is again based on usage of a colour glove. But this time the colour assignment to the glove is different from before presented works. In this work the colour assignment is not specific for each finger and palm and back of the hand. Instead the glove is divided into different patches and each patch is coloured randomly with a colour. Ten different colours are considered to distinct the patches from each other and the total number of patches is twenty.

The project uses a single camera to do tracking. It can track the individual motion of articulates of the single fingers and the 3-D global motion of the hand in the surrounding environment.

In this approach a database including a set of images of different hand poses is constructed. This database is used to estimate the pose configuration of the input image taken by the camera. To estimate the pose of the hand the database is searched for the nearest neighbour correspond to the input pose. This nearest neighbour is most likely to be close to actual pose of the hand. The obtained nearest neighbour provides the approximation of the pose of the

hand but cannot give the global position of the hand. (Therefore 2D projection constraints are associated with each image in database for the centroids of every colour patch. By transforming of these constraints into coordinate space of the original query image, the global position of the hand is obtained.) [15] To improve the estimated pose an inverse kinematics is used. This inverse kinematics minimizes the differences between the rasterization of the estimated pose and the original image.

Although colour-coded methods ease detection of the hand and consequently the tracking will be simplified, in practical situations these methods needs to carry on some extra equipment (e.g. glove or coloured sticky papers) and more problems appear when knowing that some of these equipments like gloves should be made of special materials to have a better result. Therefore it might be better to go for the methods that are independent of extra objects. These kinds of methods have potential to be more practical in real word not a laboratory environment and therefore they are more reliable for future real applications.

In next chapter the approach implemented in this project for tracking of a single finger is introduces and discussed. Also the solution to make it applicable in Android based smart phone is given.

6 Implementation

In order to implement an object tracking algorithm for a typical Smartphone, the environment such as the operating system, specification of the CPU and the available video camera must be considered.

Android is used in majority of the Smartphones today. Android's footprint is growing very fast from its current strong position. Having this in mind, it makes sense to base this thesis work on Android operating system.

Sub Chapters 6.1 points out tools and features that are available in Android and useful for image processing.

The subsequent chapters describe the steps needed for capturing and processing the data in a normal Smartphone device equipped with a camera and adequate CPU power.

6.1 Android compatibility

The intention of this thesis work is to find a strategy to make it possible to track a single finger using the video camera of Android based smartphones. Therefore the selected method of tracking should be applicable on Android platform.

Applying the tracking approach on Android platform has extra requirement than applying it on the PC. Therefore the first step is that to find a solution for applicability of the approach on Android platform.

The solution found in this project for the mentioned issue, is to use a library designed for computer vision applications.

This library is called OpenCV. OpenCV stand for Open Source Computer Vision. It is an open source library which is born in Intel and then developed further as an open source application.

This library is used in computer vision for different applications. The library is basically written in C and in C++ but it has interfaces to Python and Java as well. This library includes different algorithms that are designed for real time computer vision applications but obviously they could also be used for non-real time cases. The OpenCV usage is getting more demanding and its applications of usability varies in many areas like as satellite and web maps, security systems, military applications, Human-Computer Interaction (HCI), object identification and etc.

(The library is implementable in Microsoft Visual Studio and Eclipse APIs and it supports Windows, UNIX, Mac and Android operating systems. The latest version of this library is OpenCV 2.3.1 right now but it might be released by newer versions in future.) [10], [11]

The version of OpenCV used in this thesis work is version 2.1 and the API used for implementation is MS VC++ 2008 (Express Edition).

6.2 Capturing video stream

Before being able to track a moving object, a number of steps need to be taken. The first step would be the step of capturing video stream. The video stream needs to be in a data format that is suitable for intended video processing.

A typical Android smartphone is equipped with a camera that can support several video formats suitable for the purpose.

In this project AVI format has been used since it is supported by almost all cameras today. There are also tools available in OpenCV for Android, which can be used for AVI data processing.

For the purpose of video capturing, we could use the available camera to make frames (images) of the expected object and its surrounding.

For practical reasons, in this project I used PC environment for the implementation. Using OpenCV the code could be easily ported to Android.

Using PC-based webcam I was able to capture real time video of my finger. But for simplicity and avoidance of the issues of real time tracking it was chosen to use a saved video file of the finger moving around. The video file is an AVI video format. The video camera used to record the video is 307200 pixels (640 $\times$ 480) with the frame rate 30fps.

In order to make the environment simple and to make the detection of the moving object (finger) easier, I chose to have a simple and invariant red background.

With the mentioned choices and simplifications the captured data streams should be adequate for the next steps of the data processing, described in following chapters.

6.3 Detection

Having the video stream in a suitable video format and frame rate, the next step would be the detection of the object of interest and separation of that object from the surrounding environment.

As mentioned before, to simplify the environment and make the detection easier, I chose to have a simple red background. This does not mean that in a normal environment a tracking would not be possible, but it would be more challenging in terms of implementation.

Here a sample image of the captured video stream is shown in Figure 1. This figure gives an illustration of video stream frames processed in order to track the movement of the finger.

Figure1 Sample frame of the video stream used for motion tracking of the finer

There are various methods of detection supported by OpenCV libraries. These methods are useful for different use cases depending on the complexity of the objects, movements and also the complexity of the environment and the background.

I have tried two different methods to test which one is more appropriate in my case. Here these two methods will be described and the output results will be discussed.

6.3.1 Frame differencing

One of the methods that I studied in this project for detection purpose is called Frame Differencing.

As mentioned before an AVI video stream consists of a number of frames.

Frame differencing method is based on the differentiation between two frames. These two frames could be either two subsequent frames or any two distinguished frames.

By subtraction of the frames the background which is the parts of the image which have been constant for a period of time will be removed. The difference of two frames which are big enough is called foreground.

In this case since the finger moves, we could assume that the fingers location in two different frames will be changed and therefore the result of subtraction would be big enough to detect the finger. In other words the subtraction would give a big enough difference to qualify for being a foreground.

The algorithm implemented for frame differencing captures each frame of the video file and first converts it to a greyscale image. Then the second frame is captured and converted to greyscale. Then the absolute difference of two frames is calculated.

To avoid noises and fluctuations, the small differences are ignored. To do so, the output result is threshold to "zero" and "one" values in which the zero values (black pixels) are background and one value (white pixels) shows the foreground. To remove small noises, smoothing and dilating are applied to the frames.

The aim of this process is to catch the edges of the moving object. This implementation could subtract the edges of my finger from its background and the finger borders are detected as foreground in the resulted processed frames. In Figure 2 an example is given to illustrate frame differencing method.

Example:

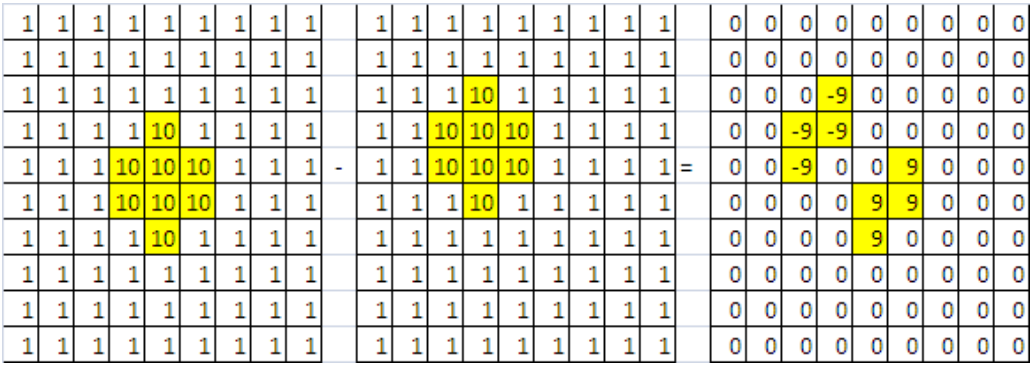

Figure 2 Frame differencing method for foreground detection

Figure 2 is a simple presentation of the frame differencing methodology. It represents two consequent frames. The first grid in the left would be the first frame. The yellow area in this frame is considered as the area of interest. In the second frame the area of interest has moved, representing the movement of the foreground.

Subtracting left and the middle frames, we can see we get the frame on the right side.

For simplicity, the pixels are given two different values "1" and "10".

The values of the pixels are practically irrelevant. Subtracting two consequent frames would result to 0 where the pixels have not changed (same value in both frames). But where the pixel value changes between frames the result is a value other than Zero. This result would indicate movement of an object from the time of first frame to the second frame. The result would indicate the relative position of the movement which should be the relative position of the foreground

Back to the yellow area, it can be seen that the pixels that have different values in two frames are pixels that the yellow area has covered, either in the first or in the second frame. Since the yellow area has moved from frame one to frame two it covers some different pixels in each frame. As it can be seen the difference of each corresponding pixel in two frames if it is part of the yellow area in two frames is zero. If it was part of the area in the first frame but is not in the second frame or vice versa, the difference value of the pixel in two frames is either "9" or "-9". Since the method consider the absolute value the sign is not significant.

Consequently the differencing frame shows the borders of the area where the pixels are covered by the moving object.

The function that calculate the absolute difference of two frames in OpenCv is called "cvAbsDiff()" and mathematical equation of what it does is shown in equation (1).

$$dst(I)_c = |src1(I)_c - src2(I)_c| \qquad \textit{(1)Absolute difference of two frames [12]}$$

Where:

$src1$ and $src2$ are the two frames to be subtracted.

The resulting frame form this equation is shown in Figure 3.

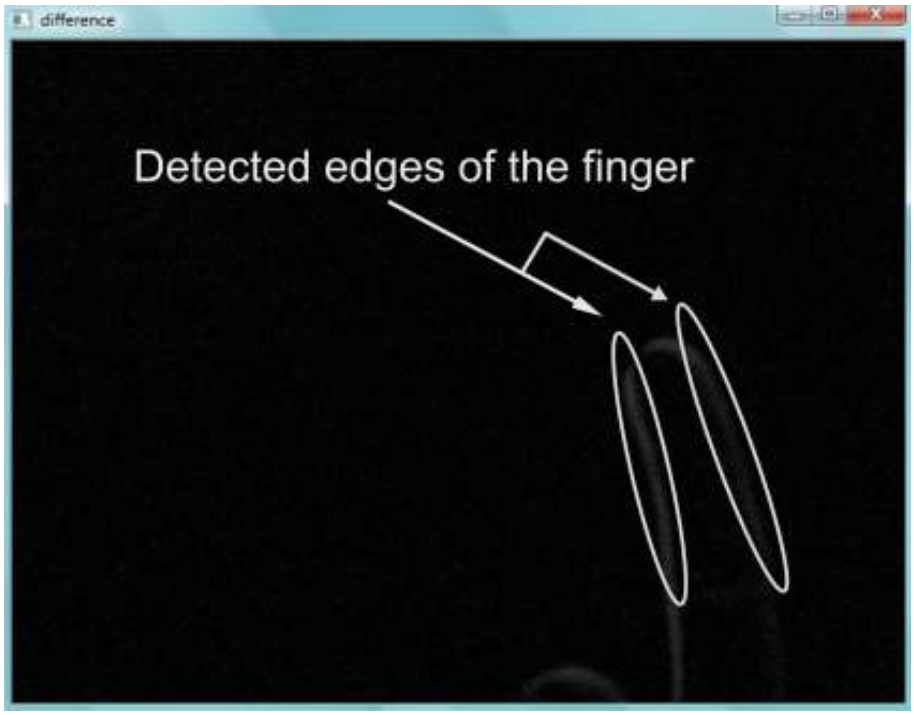

Figure 3 The resulted absolute difference of two frames

6.3.2 Averaging background

This method is also one of the methods that is supported by OpenCV library for detection purpose. In this method first a model of background is created then this model is used in order to segment the original images into foreground and background.

The construction of the model of background is that the average of each pixel and the standard deviation (average difference) of each pixel is calculated for a number of frames and considered as background.

The method first takes a number of frames and accumulates them. Also the absolute difference of these images is calculated and the results are accumulated. The accumulation and the difference calculation could be done for subsequent frames or just a few numbers of frames in each second can be taken and be used for differentiation. This accumulation is typically done for "thirty" to "one thousand" frames. After accumulation of enough frames, they are converted to a model of background.

To do the modelling the average of the accumulated raw images and the accumulated differentiated images are calculated. This way the average of each pixel and the standard deviation of each pixel for a number of frames is calculated.

The next step of creating the model is to determine two threshold values as high and low thresholds. These thresholds are calculated based on the average absolute difference of the frames and are used for segmentation purpose. The threshold calculation procedure is that first the "average frame to frame absolute difference" is scaled by a scale factor.

Then to calculate the high threshold value, the average of the accumulated frames is added to the scaled values. The result is an image in which the value of each pixel is considered as the high threshold value.

For the low threshold value, the average of the accumulated frames is subtracted from the scaled values. The result is an image in which the value of each pixel is considered as the low threshold value. In this way the background is learned.

Once the background is learned it is used to segment the original images into foreground and background.

The segmentation procedure is that first each frame of the video stream is split into its channels. The pixel values in each channel are checked to see if they are within the low and high threshold values? The pixels which are higher that the low threshold and lower than the high threshold are considered as background in each channel. The pixel which are lower that low threshold or are higher than high threshold will be foreground.

In this project the number of frames calculated for learning background is thirty. Also the scale factors for low and high threshold values are "4.0" and "2.0" respectively.

The result of this method of detection is shown in Figure 4.

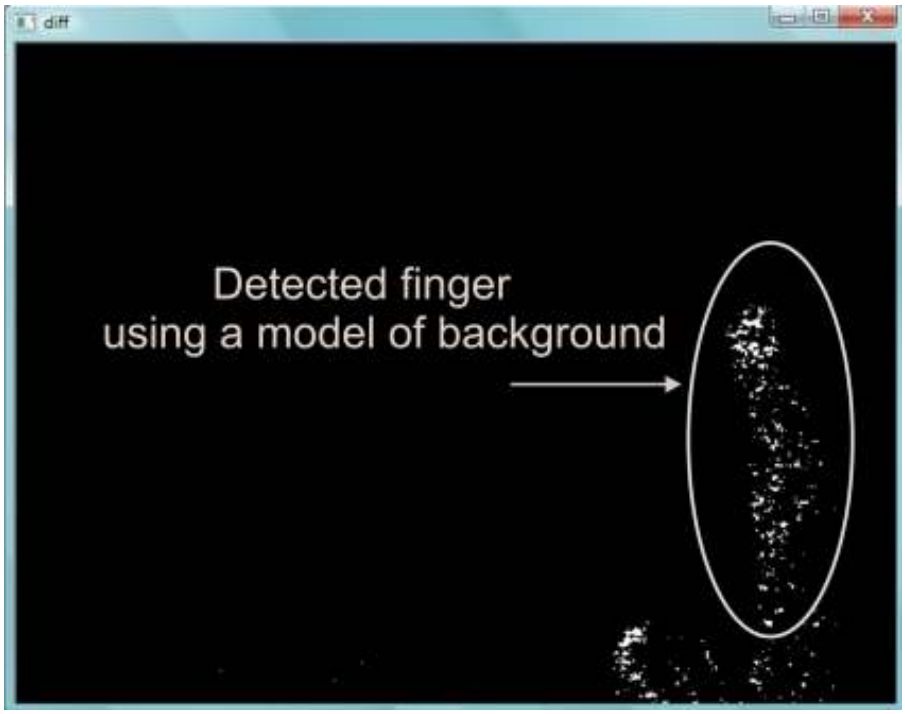

Figure 4 Averaging background method for detection of the moving object

6.4 Tracking

So far the detection purpose has been accomplished. After detection of the tiger object in the video frames it is time to track the object. There are different methods of tracking objects. Depending on the situation, one tracking method might be more suitable than the other.

In this work one method is studied and implemented to track detected moving finger. In the following sub-chapters, this method is discussed and the result of the implementation is shown.

Note: as mentioned before there are many different methods of tracking each of which are applicable depending on the situation of the tracking. The method implemented in this project is among the methods that are supported by OpenCV since this library is the main reference of the work in this thesis work.

6.4.1 Motion templates

This method is a method of tracking which was invented in the media laboratory of the MIT University and then has been developed and used in OpenCV [11]. In this approach the movement of the object is tracked by tracking the direction of the gradient of the motion over time. It means that first gradient of the motion over time is measured and then the overall

direction of the motion is calculated based on the gradient. This direction is the direction of the finger movement.

Motion Template method is based on the gathering of a group of silhouette[1]s of the intended object for tracking in the images. In the next step by using this collection of silhouettes which is gathered over time the tracking is implemented.

There are different ways to obtain silhouette of the object. According to [11] some of these methods are:

- Frame differencing: This method is described in Section 6.3.1. With help of this method it is possible to obtain the edges of the objects in the foreground. Since the moving objects are the objects which do not have the same position in different successive frames, they are considered as foreground. Hence by frame differencing method the edges of the moving finger is obtained and this way the silhouettes of the finger are gathered in different frames.
- Chroma Keying: The second way to obtain silhouette is to have specific colour as a background. For example if the background has a bright specific colour then all the objects moving across it are considered as foreground. By accessing the foreground, the silhouettes are accessed.
- Learning background: This is another way of foreground detection which has already described in Section 6.3.2. This method can also be used to isolate the new foreground as silhouette.
- Segmentation Methods: These methods are the methods which separate a frame into different segments according to the feature of segmentation. By using segmentation methods the frames are segmented to foreground and background and this way the silhouettes are generate. An example of the method which can be used for segmentation purpose is "Mean-Shift Segmentation method".
- The last possible way to generate silhouettes is to use active silhouette techniques. An example is to create a wall of near IR light and use the cameras sensitive to this light then any entering object passing this wall will show up as silhouette.

The silhouettes are saved in time and the collection is updated according to specified time duration. The silhouettes which are older than a time interval are replaced with the new ones to have the updated information of object situation. This way the history of previous movement of the silhouette is available. This motion history is used to measure the motion gradient and the motion direction of the finger.

To know how the collection of the silhouettes is gathered and what is the procedure of motion direction identification, in the following the algorithm of the implemented method is described.

Note: silhouette is the image of any object that appears darker from its brighter background in the photo. They give the representation of the object without detail.

As mentioned, there are many different ways of detecting a moving object (in this case foreground). In this work frame differencing method is used to access the silhouettes. In this method the images (frames) are segmented into foreground and background. The foreground is the part where contains the silhouette of the moving finger.

The silhouette detection procedure is that when a frame is grabbed first it is converted to gray scale then each frame is compared with the first or previous frames captured in the video stream and the absolute difference of them is saved in a one-channel, eight-bit (1ch, 8u) image.

Then the resulted absolute difference image is threshold to a binary format and saved. The threshold value is selected to be "30" and the maximum value is "255". Therefore any pixel which has a value less than the threshold value will be converted to zero and the pixels with bigger value than "30" are converted to "255". The resulted image is the silhouette of the moving object.

To illustrate how the resulted image is till this step, the result is shown in Figure 5.

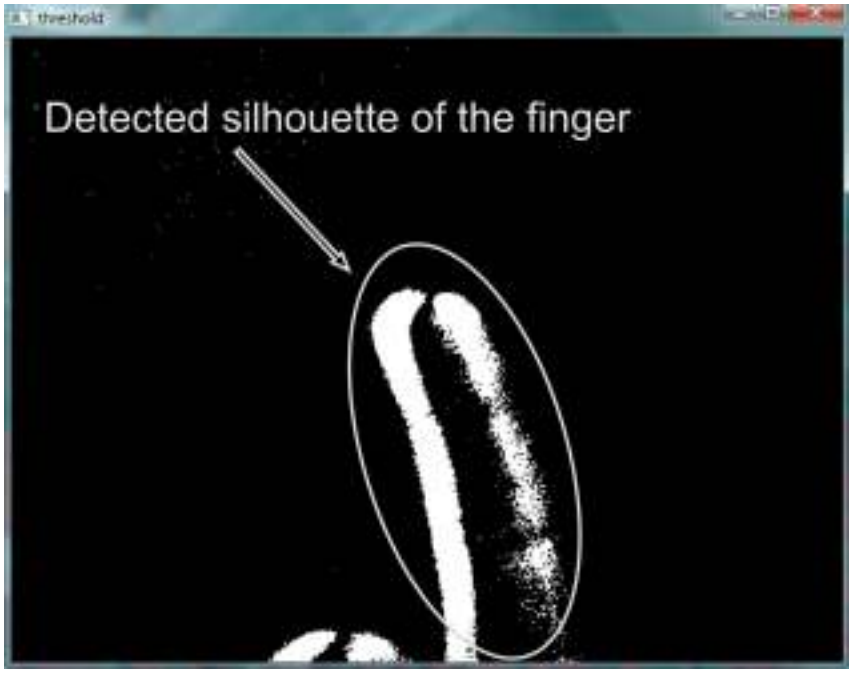

Figure 5 *Image after taking the absolute difference and threshold the difference image*

The next step is to update a motion history image in order to have the most updated silhouette of the moving finger. Motion history image (mhi) is a single-channel, floating point (1ch, 32-bit floating point) image which is updated based on the information gained out of the threshold absolute difference image created in previous step (silhouette image). Update is done in following way for each pixel of the motion history image.

- If the thresholded absolute difference value for one pixel (the value of the pixel in silhouette image) is non zero, it means that motion has occurred for that pixel. Therefore that pixel is updated to value equal to current time stamp which is typically in milliseconds.
- If the thresholded absolute difference (pixel value in silhouette image) is zero, it means that the motion has not been occurred for that pixel. Therefore if the current value of the "mhi" (which is a timestamp) is less than a threshold, the value of this pixel is updated to zero. Since when the motion did not occurred for the pixel in last step and the pixel value is smaller than a defined threshold it is concluded that motion happened for that pixel long time ago and it is not a part of the moving silhouette any more. The threshold defined for comparison is the difference of the "current time stamp" and a specified "duration" value. Therefore if the time stamp (value) of the pixel is older than the current time stamp even more than the specified duration, it is considered as background and gets value zero.
- If none of the above conditions are correct, the value of the pixel in "mhi" image is remained as its previous value. This happens when the pixel did not have motion in last updated information of the silhouette. In other word this pixel is not considered as foreground in the last updated absolute difference image but it is not a long time that this situation happened to it since its current value (mhi value) is not too old.

The mathematical summery of "mhi" image update procedure is given in equation (2).

$$mhi(x,y) = \begin{cases} current\ time\ stamp & if\ silhouette(x,y) \neq 0 \\ 0 & if\ silhouette(x,y) = 0\ and\ mhi(x,y) < (current\ time\ stamp - duration) \\ mhi(x,y) & otherwise \end{cases}$$

(2) Calculation of mhi[12]

Now that the motion history image is updated, the newest information about the silhouettes of the moving finger is available and it can be used for measuring of the gradient orientation step to indicate the overall motion. In this step to take the gradient, the derivatives of the motion history image (mhi) is calculated in two directions "x" and "y". Then the gradient orientation is calculated using these derivatives. The gradient orientation is calculated according to equation (3).

$$Orientation(x,y) = arctang\ \frac{D_y(x,y)}{D_x(x,y)}$$

(3) Gradient Orientation of mhi [12]

Where the signs of both derivatives ($D_x(x,y), D_y(x,y)$) are considered in the equation.

When the gradient orientation is calculated it is checked to identify whether the value is valid for each pixel of "mhi". To check this validity, for each pixel of the "mhi" a maximum and a minimum value over its neighbourhood is founded. Then the validity is checked considering these "max" and "min" and two other values which are the minimal and maximal allowed gradient magnitudes. The gradient is valid if the "max" and "min" difference is between the maximum and the minimum of the gradient magnitude. The mathematical presentation is:

$$\min(MaxMag, MinMag) \leq M(x,y) - m(x,y) \leq max(MaxMag, MinMag) \quad (4)$$

Gradient validity check equatoin

Where

$MaxMag$ and $MinMag$: are the maximum and the minimum defined value for the gradient magnitude respectively.

$M(x,y)$ and $m(x,y)$: are respectively the maximum and the minimum of "mhi" values over each pixel (x, y) neighbourhood.

In this way the magnitude values which are too large are considered as invalid and are eliminated. (The reason that some gradients are invalid is that the parts of the "mhi" which are older or inactive are set to zero and this zero causes large gradients around the borders of the silhouette. Since the time step duration with which the new silhouette is introduced to the "mhi" (as described in previous step) is known, it is predictable how big the gradients should be.) [12] Therefore the very big ones should not be considered in process for tracking purpose.

The output of this step is two images. First one is an image indicating where the gradients were valid. The entries of this image which are not zero indicate where the valid gradients were found. This image is a single-channel, 8-bit image. The other information obtained at the end of this step is the angle of the gradient direction at each point that is in fact the orientations values. This orientation is a floating-point image.

By having this information it is time to calculate the overall direction of the motion of the finger. The overall motion direction is in fact the vector sum of the all valid gradient directions.

To calculate the vector sum, the two images created in the previous step, specified duration and the time stamp used in the step of updating "mhi" and the last updated "mhi" are used in the function in which calculates global direction of the motion (The calculation is that first the orientation histogram is built and a basic orientation as a coordinate of the orientation maximum is founded.

Having this basic orientation, in the next step a shift relative to this basic orientation is calculated as the weighted sum of the all the orientation vectors. As much as the "mhi" is more recent the weight is greater.) [12] (The duration

and the time stamp determine how much motion should be considered from the "mhi" and the "orientation" images.)[11] (The result is an angle between "0" to "360" degrees known as the vector-sum global orientation which is the circular sum of the basic orientation and the shift.) [12]

This angle shows the direction of the global motion of the moving finger and this way the finger movement is possible to track.

To give an illustration about the gradient orientation and the vector-sum, Figure 6 shows a representation of these concepts.

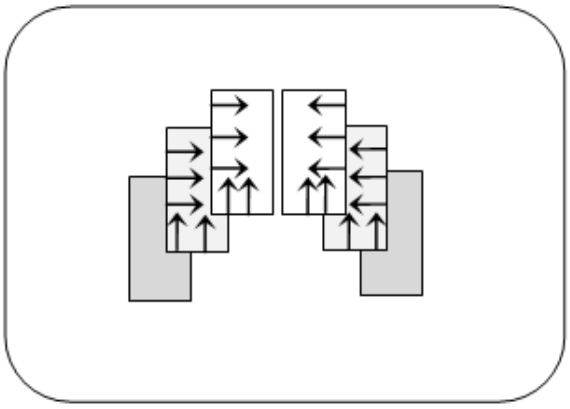

Figure 6 Gradient and directions [11]

What is shown in Figure 6, is the silhouettes and the gradient directions measured in each silhouettes. The silhouettes are shown as rectangles in the figure. The rectangles that are darker are the older ones. The arrows show the gradient directions. As it is shown in the figure, the very old silhouettes are not considered in gradient orientation measurement since their gradient values are too large and in valid. In fact their gradients are eliminated as invalid values.

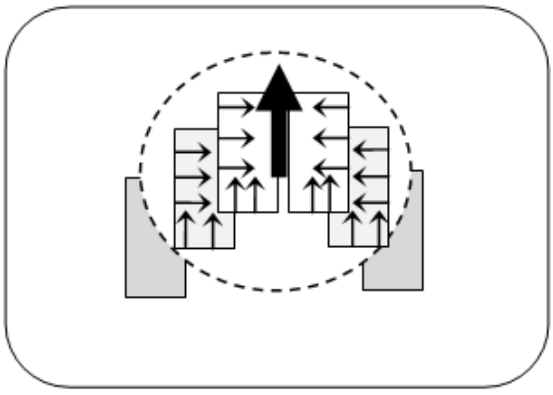

Figure 7 Overall motion detection [11]

In Figure 7, the overall direction of the motion which is the vector-sum of the valid gradient directions is shown. Clearly this direction is shown with the large arrow in the image.

As it is mentioned the overall direction resulted in this approach is an angle between "0" to "360" degrees. To illustrate this angle as the direction of the movement of the finger, a clock with arrow is shown in the output frames. The direction of this arrow shows the direction of the motion of the finger. The sample frame of the output result is shown in Figure 8 and Figure 9.

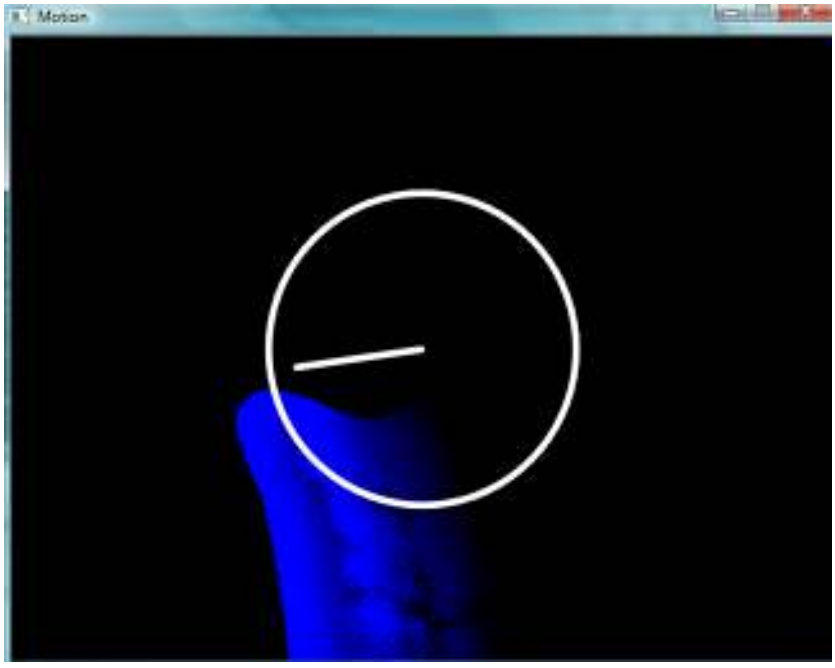

Figure 8 Illustration of the finger tracked by Motion Template approach

Figure 9 Illustration of the finger tracked by Motion Template approach

What is seen in Figure 8 and Figure 9 is the finger moving around which is shown in blue. The white circle and the arrow inside it are the clock representation of the finger direction. As it is shown in the figure, the arrow in the clock is pointing to the finger and shows its direction of the movement.

Motion templates method is a method which in this project is used for tracking of the motion of the finger. The video used for this purpose shows a single index finger moving around with a solid background. The background is red. According to [11], this approach is especially applicable for gesture recognition.

6.5 Implementation in Android

The intention of this project is to find a practical method to track a finger movement using the video camera of the Android based smartphones. This is for usage of possible functionalities that the video cameras of these mobile phones can offer the users.

To do this work the first step was to find a way that can implement the tracking algorithm on Android platforms. It was investigated that OpenCV library is a library which has the necessary image processing functionalities and it offers different methods for tracking purpose.

Also OpenCV has APIs available for implementation on Android platform.

The conclusion was that OpenCV could be considered for this purpose and it is practical for methods of tracking for smartphones.

The milestones were to implement the tracking on a PC in the first step and do the first investigations in the PC environment. Then port the implementation to Android platform in the next step. The reason was to ease the investigation steps and avoid the issues and complexity of working with Android platform as an unknown platform for the implementer.

Also, there is an issue of using OpenCV on Android platforms. OpenCV support for Android is added in version 2.2 and so forth (according to Android-OpenCV community response to this issue). It means that the work that is done on the PC should be switched to a newer version of OpenCV and the code should also be ported to C++ from the current C implementation.

Considering these issues which would complicate the investigations in terms of new versions of OpenCV and Android as unknown platform for the implementer, also considering the time limitations for this thesis work, I decided to refer the implementation on Android platform as a future work.

This project can offer a practical method of tracking using a specific library available for PC environment. This method can be considered as one method to be tested on Android platform to do tracking using the video camera of the Android based mobile phones.

7 Discussion

In this section the methods of tracking used in previous work which are mentioned in Chapter 4 of this report and their level of applicability in Android platform are discussed. Also the method implemented in this project is discussed in the same point of view.

As mentioned in the Chapter 4, there are many different methods and approaches used for object tracking purpose. Some of them are more suitable for hand tracking and hand gesture recognition applications. Since the scope of this project is to find a practical method for tracking implementable by OpenCV library, all the methods considered in this discussion are considered in relation with their accessibility in OpenCV.

7.1 Earlier work

Here the works and studies done by others are discussed.

7.1.1 Colour coded tracking

One of the methods used for hand tracking purpose implemented in earlier works is using an external object as a detector for the hand.

In this technique usually hand is covered with colour coded equipments like colour coded gloves. Or the fingers and palm are coded using different colours.

The technique for detection and tracking in this kind of approaches are usually blob detection method. There are some functions defined in OpenCV library that can be used for blob detection purpose.

My work has not gone to the extent of testing these functions but it can be concluded that there are possibilities of using this technique for finger movement detection in Android platforms.

On the other hand using an external object for the finger to make it possible to track the finger in Android platform based devices such as smartphones does not seem to be practical. Since using extra equipment rather than the device (mobile phone) itself, does not seem comfortable for the user and it causes difficulties. Therefore it decreases the user friendliness of the application which is a disadvantage.

7.1.2 Mean-Shift tracking

The second method used for tracking which is studied in this project is “Mean Shift Tracking”.

This method is a recursive method suitable for detection and tracking of the moving objects. The method is not specially used for hand or finger tracking purpose but it is presented as a method suitable for tracking.

OpenCV library has special functions defined for mean-shift tracking that can be used for tracking purpose. Therefore this method could also be a possible method that can be considered even for Android platform application and for finger tracking. Still since the method is not implemented in this work even on a PC, there is no understanding about the complexity of the work and therefore there is no suggestion to give considering advantages and disadvantages of this method. It is simply presented as an available method in OpenCV library for tracking that could be useful for finger tracking as well.

7.1.3 Skin colour detection

The last category of methods considered as a literature study in this project is the methods based on the skin colour detection.

To use these kinds of methods for finger tracking, the first step is to segment the finger skin colour from the background. The most general method used for skin colour detection is "histogram based methods". It is also common to use "parametric methods" which use single or multiple Gaussian distributions to approximate the skin colour distribution.

In OpenCV library there is no specific method or algorithm offered for skin colour detection and tracking but there are methods and functions based on histogram distribution that could be considered for finger tracking purpose.

Again there is no estimation about practical implementation of these methods and therefore here is not possible to judge about their advantage and disadvantages.

7.2 Method used in this project

In this project the intention is to present a method implementable by OpenCV library that can be used in Android platform.

The method is intended to be used for tracking of the finger movement using the video camera of the Android based smartphones.

In this work the intention is to use a method that has not been discussed by the earlier works for hand tracking. These lead to implementation of a method called "Motion Templates" to track the movement of the finger in an .AVI video.

The method is implemented and the algorithm and the theory behind the method is described and discussed in previous parts of this document. This method is a certain method which can be represented as a possibility to be used for tracking of the finger movement.

Since the method is implemented using the OpenCV library it is reliable to be considered as a method to be implemented in Android platforms. Therefore this work could be seen as a reference for the "Motion Template" method as a method that is applicable for finger motion tracking purpose which could be considered in Android platform.

However this thesis work shows that the method can be used as a general method for hand tracking purpose in smartphones and other applications rather than Android platforms.

8 Conclusion

The aim of this project was to find a practical way to track the motion of a single finger using the video camera of the Android mobile phones.

A number of factors were considered as important for a practical solution.

It is important to use a method that is easy to implement.

It is important to use an environment that exists in Android OS.

It is also important to use an algorithm that is suitable for the H.W platform of mobile devices.

A number of earlier works have been studied. Also OpenCV's functionalities have been studied. Based on the studies and based on the successful implementation during this thesis work, the conclusion is that the method of "Motion Templates" is a reliable method and could be used. This conclusion is based on:

- OpenCV has support for Android platforms
- Implementation could be done in the PC environment and then be ported to Android for ease of development effort.
- The algorithm seems to be appropriate for the computation power of Smartphone.

Surely there are other approaches that can be considered for finger tracking and they probably can be applied in Android based Smartphones. This work has a limited objective as a master thesis work and it cannot go for further investigations due to time limitations. The further investigations and implementation are considered as future works which is described in the next chapter

9 Future work

The objective of this thesis work is to find a solution for tracking of the finger in Android based Smartphone applications using the video camera of the mobile phones.

This aim is accomplished using the OpenCV library which is design for computer vision applications. The method used for tracking is called "Motion Templates".

Since the version of OpenCV library used in this project does not support Android, the implementation is done in PC windows environment. It is known that Android is supported by latest version of OpenCV. The effort of porting the result of this work to Android is considered to be future work.

There are also other methods of tracking that are implementable using OpenCV. Some of those methods have shortly been mentioned in this thesis work. For future work it is recommended to consider those other algorithms to find out about their applicability and usefulness.

The first milestone and main of this thesis work was to track only one single finger, since it eases the investigation process. Also it is a practical and useful application for mobile phones since nowadays many works are done only using a single finger on the touch screen of mobile phones hence tracking of a finger using the video camera of the mobile phones as a sensor can eliminate the need of touch screens in future generation of the mobile phones.

However there are some for future work it could be useful to study and implement as an expanded version of this work and looking at the possibilities of tracking more complicated objects for example a hand or several fingers.

10 References

[1] Changbo HU, Lichen LIANG, Songde MA, Hanqing LU (1994): Virtual Mouse----Inputting Device by Hand Gesture Tracking and Recognition. National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, P.O. Box 2728, Beijing 100080, China.

[2] C. Manresa, J. Varona, R. Mas and J. Perales (2000): Real–Time Hand Tracking and Gesture Recognition for Human-Computer Interaction. Unidad de Gráficos y Visión por Computador Departamento de Matemáticas e Informática Universitat de les Illes Balears. 07122 – Palma de Mallorca – España.

[3] Bovik, Al (ed.). Handbook of Image and Video Processing. San Diego: Academic Press, 2000. ISBN 0121197905. Wang, Yao, Jörn Ostermann, and Ya-Qin Zhang. Video Processing and Communications. Signal Processing Series. Upper Saddle River, N.J.: Prentice Hall, 2002. ISBN 0130175471.

[4] Y. Wang, J. Ostermann, Y.-Q. Zhang (2001): VIDEO FORMATION, PERCEPTION, AND REPRESENTATION. ISBN 0-13-017547-1

[5] Q. Sumin, H. Xianwu (2008): HAND TRACKING AND GESTURE RECOGNITON BY ANISOTROPIC KERNEL MEAN SHIFT. IEEE Int. Conference Neural Networks & Signal Processing, {June} 2008, E-ISBN**:** 978-1-4244-2311-8

[6] L. Sabeti,, Q. M. Jonathan Wu (2008): High-speed Skin Color Segmentation for Real-time Human Tracking. *IEE, {January} 2008,* E-ISBN : 978-1-4244-0991-4 .

[7] B. Dorner (1994): *CHASING THE COLOUR GLOVE: VISUAL HAND TRACKING*. {Thesis Submitted in Partial Fulfilment of the Requirements for The Degree of Master of Science}. SIMON FRASER UNIVERSITY.

[8] F. Duca, J. Fredriksson, M. Fjeld (2007): 3D Hand Navigation. Proc. Workshop at the *IEEE* Virtual Reality 2007 Conference: Trends and Issues in Tracking for Virtual Environments, pp. 1-5.

[9] J. Fredriksson, S. B. Ryen, M. Fjeld (2008): Real-Time 3D Hand-Computer Interaction: Optimization and Complexity Reduction. Proc. ACM NordiCHI 2008, pp. 133-141.

[10] Full OpenCV Wiki Development (2006)

[11] (2008): Learning OpenCV, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, {CA 95472}, {2008}.

[12] OpenCV 2.0 Reference, cv.Image Processing and Compuer Vision {Documentation}

[13] Wan, E.A. Van Der Merwe, R. (2002): The unscented Kalman filter for nonlinear estimation. Adaptive Systems for Signal Processing,

Communications, and Control Symposium 2000. AS-SPCC. *IEEE*, *{Aguest} 2002,* Print ISBN : 0-7803-5800-7 .

[14] G. Welsh and G. Bishop, "An introduction to the Kalman fi lter" (Technical Report TR95-041), University of North Carolina, Chapel Hill, NC, 1995.

[15] R. Y. Wang, J. Popovi´c (2009): Real-Time Hand-Tracking with a Color Glove. ACM SIGGRAPH 2009

Appendix A

```
// Tracking.cpp : Defines the entry point for the console
application.
/*This version of tracking using motion template method,
1- first save a collection of finger silhuoettes in "mhi" over time
2- By having a collection of silhuoettes of finger, it take the
gradient of the motion history image (mhi) and eliminates the
gradient values which are to big and invalid using a gradient
magnitude using "cvCalcMotionGradient()" function. This is
done to take the overal motion. This function gives two outputs:
2.1- a 1ch 8 bit image (mask)which non zero entries of that indicates
where the gradients were valid.
2.2- a floating point image which gives the gradient direction's
angle at each point.
3- In the next step, the global moion orentation which is the vector
sum of valid gradient directions is measured by using
funcion "cvCalcGlobalOrientation". The output of this function is the
vector sum global orientation.
4- check for the case of little motions in silhuotte imageand ignor
them (calculate # of points within silhouette
and checks if this number
is less than a raction of the whole area size of the image then
ignore it.)
5- Finally this globaloreintation of the motion is showed by a clock
and arrow.

*/
#ifdef _CH_
#pragma package <opencv>
#endif

#define CV_NO_BACKWARD_COMPATIBILITY

#include "stdafx.h"

#ifndef _EiC
// motion templates sample code
#include <cv.h>
#include <highgui.h>
#include <cxcore.h>
#include <time.h>
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <iostream>
#endif


// various tracking parameters (in seconds)
const double MHI_DURATION = 1;
const double MAX_TIME_DELTA = 0.5;
const double MIN_TIME_DELTA = 0.05;
// number of cyclic frame buffer used for motion detection
// (should, probably, depend on FPS)
//////////////////////const int N = 4;
const int CycFBuff = 4;
// ring image buffer
IplImage **buf = 0;
int last = 0;
```

```
// temporary images
IplImage *mhi = 0; // MHI
IplImage *orient = 0; // orientation
IplImage *mask = 0; // valid orientation mask
IplImage *segmask = 0; // motion segmentation map
CvMemStorage* storage = 0; // temporary storage

// parameters:
//  img - input video frame
//  dst - resultant motion picture
//  args - optional parameters
void  update_mhi( IplImage* img, IplImage* dst, int diff_threshold )
{
    double timestamp = (double)clock()/CLOCKS_PER_SEC; // get current
time in seconds
      std::cout << "time:" << timestamp;
    int i, idx1 = last, idx2;
    double count;
    double angle;
      double magnitude;
    IplImage* silh;
    CvSeq* seq;
    CvRect comp_rect;
      CvPoint center;
    CvScalar color;

      CvSize size = cvSize(img->width,img->height); // get current
frame size
    // allocate images at the beginning or
    // reallocate them if the frame size is changed
    if( !mhi || mhi->width != size.width || mhi->height !=
size.height ) {
        if( buf == 0 ) {
            buf = (IplImage**)malloc(CycFBuff*sizeof(buf[0]));
            memset( buf, 0, CycFBuff*sizeof(buf[0]));
        }

        for( i = 0; i < CycFBuff; i++ ) {
            cvReleaseImage( &buf[i] );
            buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
            cvZero( buf[i] );
        }
        cvReleaseImage( &mhi );
        cvReleaseImage( &orient );
        cvReleaseImage( &segmask );
        cvReleaseImage( &mask );


        mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );      //1ch
float image of the size "img"
        cvZero( mhi ); // clear MHI at the beginning
        orient = cvCreateImage( size, IPL_DEPTH_32F, 1 );
        segmask = cvCreateImage( size, IPL_DEPTH_32F, 1 );
        mask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
    }

    cvCvtColor( img, buf[last], CV_BGR2GRAY ); // convert frame to
grayscale

    idx2 = (last + 1) % CycFBuff; // index of (last - (CycFBuff-1))th
frame
```

```
    last = idx2; //=2

    silh = buf[idx2];   //1 channel 8bit image

      //*use frame differencing method to obtain silhouette
    cvAbsDiff( buf[idx1], buf[idx2], silh ); // get difference
between frames and put in "silh"

    cvThreshold( silh, silh, diff_threshold, 255, CV_THRESH_BINARY );
// and threshold it
      //cvShowImage( "threshold",  silh);
      //@silh:  Silhouette mask that has non-zero pixels where the
motion occurs
      //@mhi: 1channel, 32-bit floating-point image updated by the
function. It contains either "timestamp" (if the motion occured for
that pixel: silh(x,y)!=0)
      //@or "0"(silh(x,y)=0 && mhi is far old "mhi < timestamp-
MHI_DURATION") or "mhi(x,y)" (o.w)==> MHI pixels where motion occurs
are set to the current
      //@timestamp, while the pixels where motion happened far ago
are cleared.
    cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); //
update MHI

    // convert MHI to blue 8u image
    cvCvtScale( mhi, mask, 255./MHI_DURATION, (MHI_DURATION -
timestamp)*255./MHI_DURATION );
    cvZero( dst );
    cvMerge( mask, 0, 0, 0, dst );  //@insert the singe channel
"mask" image into the "dst" image

    // calculate motion gradient orientation of the motion history
image (mhi)//and valid orientation mask
      //@mhi: Motion history image
      //@mask: Mask image; marks pixels where the motion gradient
data is correct; output parameter
      //@orient: Motion gradient orientation image; contains angles
from 0 to 360 degrees
    cvCalcMotionGradient( mhi, mask, orient, MAX_TIME_DELTA,
MIN_TIME_DELTA, 3 );

    if( !storage )
        storage = cvCreateMemStorage(0);  //@create a memory storage
of the size 64K byte.
    else
        cvClearMemStorage(storage);

    // @segment motion: get sequence of motion components
    // segmask is marked motion components map (Image where the mask
found should be stored). It is not used further
      //@segmask: Image where the mask found should be stored
      //@storage: Memory storage that will contain a sequence of
motion connected components
      //@MAX_TIME_DELTA: Segmentation threshold: the max downward
step (from current time to previous motion) that you'll accept as
attached motion.
    //!!!!seq = cvSegmentMotion( mhi, segmask, storage, timestamp,
MAX_TIME_DELTA ); //@Segments the whole motion into separate moving
parts (isolate and
      //@ compute local motion)
```

```
    // iterate through the motion components,
    // One more iteration (i == -1) corresponds to the whole image
(global motion)
    //!!!!for( i = -1; i < seq->total; i++ ) {
      for( i = 0; i < 1; i++ ) {

        //!!!!if( i < 0 ) { // case of the whole image
            comp_rect = cvRect( 0, 0, size.width, size.height );
            color = CV_RGB(255,255,255);
            magnitude = 100;
        //!!!!}
        //!!!!else { // i-th motion component
            //!!!!comp_rect = ((CvConnectedComp*)cvGetSeqElem( seq, i
))->rect;
            //!!!!if( comp_rect.width + comp_rect.height < 100 ) //
reject very small components
                //!!!!continue;
            //!!!!color = CV_RGB(0,255,0);
            //!!!!magnitude = 30;
        //!!!!}

        // select component ROI
        /*cvSetImageROI( silh, comp_rect );
            cvSetImageROI( mhi, comp_rect );
        cvSetImageROI( orient, comp_rect );
        cvSetImageROI( mask, comp_rect );*/

        // calculate global orientation
            //* output: the vector sum global orientation
        angle = cvCalcGlobalOrientation( orient, mask, mhi,
timestamp, MHI_DURATION);
        angle = 360.0 - angle;  // adjust for images with top-left
origin

        count = cvNorm( silh, 0, CV_L1, 0 ); // calculate number of
points within silhouette ROI

        cvResetImageROI( mhi );
        cvResetImageROI( orient );
        cvResetImageROI( mask );
        cvResetImageROI( silh );

        // check for the case of little motion
        //if( count < comp_rect.width*comp_rect.height * 0.05 )
            if( count < comp_rect.width*comp_rect.height * 0.005 )
            continue;

        // draw a clock with arrow indicating the direction
        center = cvPoint( (comp_rect.x + comp_rect.width/2),
                          (comp_rect.y + comp_rect.height/2) );

        cvCircle( dst, center, cvRound(magnitude*1.2), color, 3,
CV_AA, 0 );
        cvLine( dst, center, cvPoint( cvRound( center.x +
magnitude*cos(angle*CV_PI/180)),
                cvRound( center.y - magnitude*sin(angle*CV_PI/180))),
color, 3, CV_AA, 0 );
            //cvShowImage( "Motion1", dst );
    }
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
      std::cout << "here!"<<std::endl;
      cvWaitKey(0);
      IplImage* motion = 0;
    CvCapture* capture = 0;

    //if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 &&
isdigit(argv[1][0])))
    //    capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' :
0 );
    //else if( argc == 2 )
      //if( argc == 2 )
      {//capture = cvCaptureFromFile( argv[1] );
            //capture =
cvCreateFileCapture("C:/Users/Sony/Desktop/MVI_3985.avi");
            capture = cvCreateFileCapture("G:/Motion Detection
Thesis/MVI_4620.avi");
            std::cout << "here there!"<<std::endl;}

    if( capture )
    {
        //cvNamedWindow( "Motion", 1 );
            std::cout << "here here!"<<std::endl;
        for(;;)
        {
            IplImage* image = cvQueryFrame( capture );
            if( !image )
                break;

            if( !motion )
            {
                motion = cvCreateImage( cvSize(image->width,image-
>height), 8, 3 );
                cvZero( motion );
                motion->origin = image->origin;
            }

            update_mhi( image, motion, 30 );
                  cvNamedWindow( "Motion", 1 );
            cvShowImage( "Motion", motion );
                  cvShowImage( "Main", image );

            if( cvWaitKey(2) >= 0 )
                break;
        }
        cvReleaseCapture( &capture );
        cvDestroyWindow( "Motion" );
    }

      return 0;
}

#ifdef _EiC
_tmain(2,"motempl.c");
#endif*/
```