Playing A Virtual Instrument Using Gesture Recognition

student     I.J. SINGER
student-number  0038784
supervisor    DR. R. VAN DEN BOOMGAARD
date     17/06/2004
Abstract

During this project a system has been developed which converts gestures from the user, in front of a camera, into musical notes. This system is being controlled by skin colored areas in the incoming video frames. The skin is segmented using a skin color model in the normalized RGB color space. The gestures are being recognized by first following these skin areas in the video frames. These skin areas are matched between frames using the smallest total Euclidean distance criterium. The notes are triggered when the followed skin areas cross a horizontal line. The pitch is determined by the horizontal offset and the velocity is determined by the Euclidean distance traveled from the previous frame.
Contents

Abstract
1
1 Introduction
2
2 Theory
2.1 Skin Segmentation
2.2 Connected Components Analysis
2.3 Shape Analysis
2.4 Tracking
2.5 Introduction to MIDI
3 System Description
3.1 Introduction
3.2 Image Acquisition
3.3 Image Segmentation
3.4 Connectivity Analysis
3.5 Shape Analysis
3.6 Motion Analysis
3.7 Gesture Analysis
3.8 Sound Production
4 Experiments
4.1 Image Acquisition
4.2 Skin Segmentation
4.3 Quantitative performance analysis
4.4 Qualitative performance analysis
5 Conclusions
6 References
Appendix A Code Overview
1 Introduction

For its PlayStation 2 (PS2) Sony developed the EyeToy [1], a USB camera device which in combination with the processing capabilities of the PS2, is able to detect body movement and allows one to interact with the games. The first application that uses this device is called EyeToy:Play [2]. It offers 12 party games that range from cleaning windows to “fending off fearsome ninja adversaries”. This kind of technology is not new. This game suite released in July 2003 is just an example of what can be done using image processing, image analysis and machine vision. After years of playing with keyboards, mice and joysticks, the EyeToy uses a new form of Human-Computer Interaction, which is according to Hewitt et al.[3] “a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them”. Even more exciting things are possible in this field. For instance, it should be possible to track the movements of a person pretending to play drums. If these movements could somehow be converted into actions on the instrument, this person could get the accompanying sound with a little help from the computer. This is the objective of this bachelor graduation project.

In short the objective is to create a system which allows the user to play an instrument using a digital camera without any accessories. And as far as possible this system should also satisfy the following requirements:

- It should work with cheap hardware (a regular home environment)
- It should work with persons of all types of skin color
- It is supposed to work in real-time
- The output should be of a widely used standard

I expect that human computer interaction will change in the future and that applications like EyeToy and the one that is developed in this project will become more common. The technology is not limited to the gaming and music industry (see website from InteraMotion [4] for interesting examples). It can also be used for, among others, regular computer usage (as an input device), surveillance camera’s and for pornography filters.
2 Theory

This chapter lays the theoretical foundation which is needed to understand the system described in chapter 3. First of all it will explain skin segmentation, which is used for discerning skin and non-skin areas in images. Then connectivity analysis will be explained, which is an approach to group pixels. After that, tracking of objects will be discussed and finally it will end with an introduction to MIDI.

2.1 Skin Segmentation

2.1.1 Introduction

According to Gonzalez and Woods [5] segmentation procedures "partition an image into its constituent parts or objects". In case of skin segmentation this means that an image is partitioned into skin and non-skin areas. The result for each frame is a binary version of the image with zeros for non-skin and ones for skin pixels. There are different approaches to do this segmentation. Vezhnevets et al. [6] state in their survey on skin color detection that there are two approaches for skin detection methods: pixel-based and region-based. Pixel-based segmentation calculates for each pixel the probability (using a model of the skin) of it being a skin pixel independently from its neighbors. It then uses a threshold to separate skin from non-skin pixels. Other approaches involve modeling non-skin pixels too which can for example classify according to highest probability. Region-based segmentation, on the other hand, takes into account the spatial arrangement of skin-pixels to improve the segmentation result. This information can be used in skin segmentation by assuming the fact that skin areas are often found to have specific shapes, for instance faces are ovals. When using such information the detection of human heads can be improved. Yang and Ahuja [7] for instance proposed a method to do this by merging skin areas on multiple scales until an elliptical shape has been approximated. In both earlier mentioned methods it is needed to determine the probability of a pixel being skin or non-skin. For this skin-segmentation there are two important choices to make. First of all the color space that is going to be used and secondly how to model the skin color distribution. An overview and theoretical evaluation is given in Vezhnevets et al. [6]. Basically the color space determines the distribution of the skin and non-skin pixels and the skin color distribution model is "used to give a metric, which measures the distance (in general sense) of the pixel color to skin-tone". The next two paragraphs will explain two of the possible approaches which can be used for skin segmentation.

2.1.2 Normalized RGB color space

According to Vezhnevets et al. [6] one of the important choices when doing pixel-based skin segmentation is the choice of color space. In this section the normalized RGB color space will be discussed. It can be easily calculated from an RGB image by normalization as follows:

\[
\begin{align*}
    r &= \frac{R}{R + G + B} \\ 
    g &= \frac{G}{R + G + B} \\ 
    b &= \frac{B}{R + G + B}.
\end{align*}
\]

The R, G and B values are the color components of a pixel in an RGB image and they represent the intensity of red, green and blue light which mixed together give the pixel its color. The r, g and b values are the normalized R, G and B values. Note that rgb triplets
do not uniquely correspond to RGB values. Because the sum of the three normalized components is known, the third component can be omitted, reducing the color to two dimensions. The special case of black, with all color values being zero, is often not discussed in literature. This is remarkable because according to the formulas it would cause a division by zero which is not possible. Obviously there are two choices for the black pixels: disregard them all or map them to a color. What this normalization actually does is that it regards the RGB values as vectors and then calculates their intersection with the \( R + G + B = 1 \) plane. When you do this for values close to black noise theoretically will play a very big part in determining the normalized color. The brightness of a color can be removed by this normalization \([8]\). This means that a uniformly colored object with soft shadows on it will have a uniform color after normalization. So far only the color space has been discussed, but the skin color distribution is also of importance. Yang et al. \([8]\) found that for one person this resembles a bivariate Gaussian distribution and that it can be used to characterize the color distribution of human skin color. The position of this distribution depends not only on the skin color but also on illumination color on the skin. Sunlight for example will shift the distribution towards blue, because it contains more blue light than the average light bulb.

### 2.1.3 TSL color space

Another color space which can be used for skin segmentation is the TSL color space introduced by Terrillon et al. \([9]\). TSL stands for Tint, Saturation and Lightness and aims just like the normalized RGB color space at separating intensity from color information. Since its introduction in \([9]\) it changed a little and the following formulas are described in \([10]\) and are used for converting an RGB image to TSL color space. The components are calculated as follows:

\[
T = \begin{cases} 
\arctan(r'/g')/2\pi + 1/4 & \text{for } g' > 0 \\
\arctan(r'/g')/2\pi + 3/4 & \text{for } g' < 0 \\
0 & \text{for } g' = 0 
\end{cases} \tag{2.4}
\]

\[
S = \sqrt{\frac{9}{5}(r'^2 + g'^2)} \tag{2.5}
\]

\[
L = 0.299 R + 0.587 G + 0.114 B \tag{2.6}
\]

with \( r' = (r-1/3) \), \( g' = (g-1/3) \), \( r = R / (R+G+B) \), \( g = G / (R+G+B) \). What this conversion actually does is first a conversion to normalized RGB color space, followed by a translation so that the equal energy point (the point \( R = G = 1/3 \) which contains all the grays) lies in the origin. Then the tint is determined by the angle it has relative to the origin and the saturation is determined from the distance from the origin (which are both normalized). The lightness is determined by a weighted average of the original RGB components. There are some special cases to be considered. For the tint the angle starts at the negative \( r' \)-axis and increases counter clockwise until it reaches the positive \( r' \)-axis and then it becomes zero again for some unknown reason. After that it increases normally until it reaches the negative \( r' \)-axis again. Also the same problem what to do with black rises here because the TSL conversion has a normalized RGB conversion in it too. The options are still the same: disregard them all or map all blacks to a different color.

In this color space the skin color distribution has been examined by Terrillon et al. for the components of an unbiased anthropometric scale. This scale is based upon von Luschan’s chromatic scale of 36 different skin colors samples \([11]\) ranging from an unsaturated light color to a saturated brown color and assigns a equal statistical weight to each of the samples, see Figure 2-1. In this case statistical weight means that they all contribute the same amount of pixels to the skin distribution. They noticed that the skin distribution consisted of two clusters one for Caucasian/Asian skin colors and one for dark skins. In
TSL space the distribution for all Caucasian and Asians can be modeled using a single bivariate Gaussian distribution. The black cluster is left for further research. The reason that they try to find a distribution for all the skin colors is that they use it for face detection in images and thus are not focused on training by one specific person but wanted to have a general classifier.

2.2 Connected Components Analysis

When we want to group neighboring pixels in a binary image, because we want identify them as belonging to one object, we need to perform a connected component analysis. The connected component analysis extracts connected components from a binary image, which brings us to the question how a connected component is defined. According to Gonzales and Woods [13] if S represents a subset of pixels in an image then two pixels are said to be connected if there is a path between them consisting entirely of pixels in S. There is a path from pixel \((x,y)\) to \((x_n, y_n)\) if there is a sequence of pixels which are adjacent to each other which go from \((x,y)\) to \((x_n, y_n)\). Adjacency for any pixel \((x,y)\) can be defined as \(N_4(x,y)\) or \(N_8(x,y)\) which is shown in Figure 2-2. For any pixel in S, the set of pixels that are connected to it are called a connected component of S.

When every pixel has been analyzed we can make a labeled image. A labeled image means that all connected components in a binary image have been found and that all members of the same component have the same number. In Figure 2-3 this is shown where the areas belonging to the same component have the same color.

![Figure 2-1 Von Luschan’s chromatic scale for the classification of skin color (taken from Natural History Museum of Florence [12])](image)

![Figure 2-2 Definition of the \(N_4\) and \(N_8\) neighborhood for the pixel at \((x,y)\)](image)

4 Theory
2.3 Shape Analysis

To determine the properties of areas discovered in a connected component analysis a shape analysis can be done. In this case the size of the area and the centre of gravity (often called the centroid) will be explained, but there are more properties which can be considered. Such properties are for example given by the central moments of the two dimensional image [13]. The moment $\mu_{pq}$ is defined for a digital image as:

$$
\mu_{pq} = \sum_{x} \sum_{y} (x-\bar{x})^p (y-\bar{y})^q f(x,y)
$$

(2.7)

With the $x$ and $y$ values being the position in the image, their mean being $\bar{x}$ and $\bar{y}$ and $f(x,y)$ being the value of the image at position $(x,y)$. From this moment other moments could be calculated which are invariant to translation, scale and rotation change [13].

The centre of gravity can be calculated by taking the average of all the $x$ and $y$ values of the pixels in the area and the size can be calculated by the number of pixels as follows:

$$
\bar{x} = \frac{1}{N} \sum_{N} x
$$

(2.8)

$$
\bar{y} = \frac{1}{N} \sum_{N} y
$$

(2.9)

$$
O = N
$$

(2.10)

where only those $N$ pixels are considered for which their value is equal to the group number which is being examined and $x$ and $y$ are the $x$ and $y$ position of pixel $N$. The area calculation can also be done using the previously mentioned moments. One can see that the $\mu_{00}$ is just the adding of function values for all the $x$ and $y$ values. In a binary images this would give the number of ones in the image and thus the size of the area.

2.4 Tracking

2.4.1 Overview

Motion analysis in general involves three major areas: motion detection, moving object detection and localization and derivation of 3D object properties. Tracking is a part of moving object detection and localization which tries to follow an object’s motion through a sequence of frames.
Tracking can be done in several ways. A piece of an image could be followed. This is what happens in template matching. In that case you use a template (taken from a frame or from outside source) and for each frame some properties of the template (for example the color values) are being used to find the same or closest-looking object in the image. By using apriori knowledge (maximum speed for example) the search area can be smaller than the complete image. Another approach is using image features (edges for example) which is information filtered from the image data. This approach reduces the number of data enormously. Features will be extracted for successive frames and these points will have to be matched somehow. During this tracking apriori knowledge, like during template matching, can be used to estimate the next location or other approaches could be used. In the following paragraphs two methods of tracking will be discussed.

2.4.2 Euclidean Method

Given the problem of N points at time k-1 which have to be matched with N points at time k one approach is to minimize the sum of all Euclidean distance between each pair. The (Euclidean) distance $D_i$ can be calculated as

$$D_i = \| P_i^k - P_i^{k-1} \|$$

(2.11)

where $P_i^k$ is the point with index i (of N total points) and k is the step in time. The total cost for all N matches is defined as:

$$D = \sum_{i=1}^{N} D_i .$$

(2.12)

An algorithm calculating the best match will try all combinations of pairs from the successive frames and then select the one with the lowest total cost. This will theoretically give good matches when points are far away from each other and move relatively small or no distances. When trajectories of the objects tracked are near each other the objects the tracking can possibly cross over. This is illustrated in Figure 2-4.

2.4.3 Path Coherence Method

Given the problem of N points from M frames which have to be made into trajectories one approach is to minimize the deviation in a path given as explained by Sonka et al.[14]. It has not been used in the final version of the system as can be read in paragraph 3.6 where the motion analysis which has been used is described. This method uses a cost function $D_i$ which determines deviation for measuring the path’s coherence regarding the angle succeeding points make and the differences in movement. The deviation function $\Phi_i$ is determined by the angle three successive points make and their difference in movement which is calculated using the function $\Phi$. The cost for
\[ d^k_i = \Phi(P^k_{i-1}, P^k_i, P^{k+1}_i) \]  

(2.13)

is defined by the function \( \Phi \) as:

\[
\Phi(P^{k-1}_i, P^k_i, P^{k+1}_i) = \omega_1 (1 - \cos \theta) + \omega_2 \left(1 - 2 \frac{\sqrt{s_k s_{k+1}}}{s_k + s_{k+1}} \right)
\]

\[ = \omega_1 \left(1 - \frac{\|P^k_i - P^{k-1}_i\| \cdot \|P^{k+1}_i - P^k_i\|}{\|P^k_i - P^{k-1}_i\| \cdot \|P^{k+1}_i - P^k_i\|} \right)\]

\[ + \omega_2 \left(1 - 2 \frac{\|P^k_i - P^{k-1}_i\| \cdot \|P^{k+1}_i - P^k_i\|}{\|P^k_i - P^{k-1}_i\| + \|P^{k+1}_i - P^k_i\|} \right) \]

(2.14)

where the motion vectors \( s_k \) and \( s_{k+1} \) and angle \( \theta \) are illustrated in Figure 2-5 and \( \omega_1 \) and \( \omega_2 \) are the weights assigned to reflect the relative importance of coherence in the angle and in the movement. The cost of path \( D_i \) can be calculated by taking the sum of the cost for point 2 to \( (m-1) \) in the path as follows:

\[ D_i = \sum_{k=2}^{m-1} d^k_i . \]

(2.15)

Then the total sum of all \( N \) paths will be the total cost \( D \) which is

\[ D = \sum_{i=1}^{n} D_i . \]

(2.16)

In this method all the combinations could be calculated and using the cost functions the one with the minimal cost can be selected. This trajectory is theoretically supposed to have better tracking when trajectories of objects cross each other but is undefined for objects which are still (hold same position) for two succeeding points or more.

Figure 2-5 The motion vectors and the angle used by the path coherence method (Source: Sonka et al. [14])

2.5 Introduction to MIDI

MIDI is a technology that represents music in digital form. Unlike audio files, such as mp3- and wave files, MIDI messages contain individual instructions for playing each individual note on each individual instrument. According to the MIDI Manufacturer’s Association (MMA) [15]: “Most film and TV scores, as well as popular recorded music is written and performed using electronic keyboards and other MIDI-equipped musical instruments.” They even say that thanks to advances in digital sampling and synthesis technologies, the
orchestra playing behind a major movie is more likely to be the product of MIDI than a real orchestra.

MIDI begins with a midi input device called a MIDI controller. This MIDI controller can for example be a keyboard or a special set of drums. When you play it, actions on the instrument are translated into MIDI data. MIDI data is specified by the MIDI protocol. In this protocol messages are specified which have to be send in order to (re)produce the intended sound. This MIDI data can among other things instruct a sound module which notes to play, for how long and how loud to play them (see Figure 2-6).

Figure 2-6 A simple example of a MIDI system (source: MMA)

Another possible recipient of MIDI data is a sequencer, which records the data so it can be played back later. MIDI data can be send on different channels which can be used in a more complicated MIDI system where each sound module listens to a specific channel and reacts only the sound instructions on that channel (see Figure 2-7 for an example of such of a setup).

Figure 2-7 A more complex MIDI system (source: MMA)

The MIDI messages itself have an eight-bit status byte which is commonly followed by one or two data bytes. There are two main types of MIDI messages: Channel Messages and System Messages. Channel Messages apply to a specific channel which is specified in the status byte and System Messages which are not specific to any channel.
3 System Description

3.1 Introduction

The developed system basically turns video frames into MIDI actions. In order to do this the steps illustrated in Figure 3-1 have to be followed. First of all a frame will be caught from the video stream. After that the position of the hands will have to be identified in the frame. This is done by skin segmentation and means that a binary image will be created which makes a distinction between skin pixels and non-skin pixels in the input image. When it is known what is skin and what is not, those areas will be grouped into meaningful structures using connectivity analysis. Each pixel in an area of skin will belong to the same group and properties of this group will be calculated using shape analysis. A group in this case is a piece of skin for instance where the pixels are connected to each other. Properties which are calculated are the centre of gravity (the centroid) and the size of the area in pixels. At this point the coordinates of objects in one frame are known. To analyze motion those points taken from succeeding frames have to matched. This is done in the motion analysis step and it yields paths of objects (hand and possibly head). During gesture recognition these paths will be examined to determine whether there are any actions on the instrument. Finally these actions are transformed into sound in the sound production step. These actions will be used to send relevant MIDI information which can be used by another program to record the events and/or give audio feedback. All of the previous mentioned steps were implemented in Matlab (supported by C-functions) and will be discussed in the remaining of this chapter.

3.2 Image Acquisition

In the image acquisition step a video frame is retrieved from an outside source. Image acquisition in Matlab for recorded video is easy. After the file has been loaded into Matlab the video frames can be accessed directly. For video from a video device (such as a webcam or a digital camera) Matlab needs a toolkit. This can be its own Image Acquisition Toolbox or one of the other (freely available) toolkits: VFM (Vision for Matlab [17]) or VCAPG2 [18]. Another option has been a media input function, which is currently under development, from Dr. D. Koelma. Video devices such as the webcam deliver their images in various color spaces and dimensions. Due to the fact that Matlab normally works with RGB images (for displaying images) and that it was easier for the color space conversion in the segmentation the RGB color space has been chosen. Although the camera was capable of acquiring images at 640x480, the chosen dimension of 320x240 did not lead to noticeable performance loss. But of course it did lead to an increase of speed.

Figure 3-1 System overview
3.3 Image Segmentation

In the image segmentation step the color image from the video device or recorded video will be translated to a binary image with ones for skin and zeros for non-skin areas. Both color spaces associated with the distribution function mentioned earlier in the theory, chapter 2, were implemented in Matlab. Although this worked, the functions were rather slow. Calibration has to be done to determine the parameters for the skin distribution. The training set for the TSL color space should be a training set of several people ranging from Asian to Caucasian under that lighting in front of the used camera. Although this should yield according to theory a general skin classifier, I’ve chosen not to use it. Because this would not be practical, to need several people with skin from a range of colors, and because black skin is not supported. Training by one person seemed to work relatively well nonetheless but was not supported by theory. That’s why the approach using the normalized RGB color space has been chosen which is specific for only one person. This function has been written in C because it will be called for each frame and the unoptimized version was rather slow, so optimization is of the essence.

Because the input which comes from the image acquisition is an 8-bit RGB image a lookup table is used, a lookup-cube to be more precise, which contains for each possible RGB-triple whether it is skin or not. Smaller sizes for the lookup cube are also possible but have not been tested. Through initialization the lookup-cube can be filled once and stay persistent in memory. The parameters which have to be specified are those of the distribution (mean, covariance matrix) and a threshold. The parameters of the distribution are obtained from a skin sample taken from the user before the system starts. This also ensures calibration under the same lighting as during the running of the system. The user will be asked to fill a rectangle with its skin and after color space conversion these parameters are calculated and using a threshold, which has been determined interactively, the initialization of the lookup-cube can occur.

3.4 Connectivity Analysis

The connectivity analysis finds skin areas in the binary image given by the skin segmentation and does this by giving all pixels in the same area the same label and the background is given the value zero. So each pixel will have a label to which group it belongs to. Because the binary image contains noise and sometimes areas besides the hands and head preprocessing occurs. Morphological filters which can be used to do this are rather expensive in performance terms in Matlab. Therefore although a combination of several operators can give good results the use of them must be limited. Only a small median filter of size 5x5 is used. A median filter replaces the middle pixel \((x,y)\) of an, in this case 5x5, neighborhood with the value at the middle position when the values of the pixels in the neighborhood have been sorted. Of course in binary images it is as simple as counting ones or zeros and determine the one who has more than half of the number of pixels in the neighborhood (for odd sizes of the neighborhood). Median filters remove salt and pepper noise and the result of the operator can be seen in Figure 3-2. Matlab has an implementation of the median filter, although it may not be the fastest for binary images. After preprocessing the connectivity analysis is done which uses the Matlab function \texttt{bwlabel} using N8 connectivity (see paragraph 2.2).

Although some preprocessing has been done the image will probably have lots of noise in it. What is needed is just the hands and the head. Further analysis and selection on shape will take care of better recognition of these objects.
3.5 Shape Analysis

During the shape analysis step the properties of the areas found during the connectivity analysis are being calculated. These properties are their size and their centroid. Because the Matlab shape analysis function regionprops is too slow a function has been written which calculates these properties in C. Due to noise, probably more than three areas are found, and that’s why the three largest areas are taken. We do this to extract the hands and the head from the image. Because, if available, the three largest areas are taken the head will be assumed to be in the image (which will also generate a skin area, although it also contains non-skin elements like hair). If there are less than three skin areas this number is being completed by adding points.

3.6 Motion Analysis

In the Motion Analysis step the properties of individual succeeding frames are connected to represent movement. The algorithms as mentioned in the theory section of this report have been implemented in Matlab. At frame number one nothing can be matched because there is no history. That’s why Euclidean matching, needing only the previous and current frame, starts working from frame 2. The path coherence method needs at least two points and because only the last element needs to be matched the rest of the path is no longer of concern. This is because the cost of the already matched points will stay the same. That is if you are not going to do a re-evaluation of the already matched points. This is a choice which can be made and which I decided to do so. But when two previously matched points are known, then new points can be used to calculated the cost of the angle and the difference in movement. These first two points could possibly be matched using Euclidean matching but has not been used by the system (because a properly working version has not been developed).

3.7 Gesture Analysis

During gesture analysis the paths from the motion analysis are converted in appropriate actions. The primary goal was to make a working system. Therefore the gesture analysis is relatively simple. The method currently used movement through a horizontal line in downward direction will start a note to play. This is sufficient for the current application, where the notes decay automatically. It could be implemented but no obvious solution is available. Stopping a note when movement crosses the line upwards will probably let the user stop the wrong note. Another approach could be to check the pitch for the downward movements only and just stops them when an upward movement is done using this same hand. This would involve using the tracking history for a longer period of time but at this
moment stopping notes has not been implemented.

3.8 Sound Production

The note to be played specified by its pitch and velocity has been recognized by the gesture recognition. This note has to be made audible and to make it flexible that has been done using MIDI. A function has been created in C which after initialization can be called to send a note to the default MIDI device with specified pitch and velocity. This device and the instruments (in this case piano) are hard coded but could easily be changed to let the user select it. The default device on a system like Windows XP is the device configured in the configuration screen as default MIDI playback device. This is like sending MIDI messages through a virtual cable to a sound module inside the computer. Obviously as can be read in the introduction to MIDI it could also be routed to an external sound module. This means that you could let it trigger a professional piano module.
4 Experiments

4.1 Image Acquisition

The first attempt to do the image acquisition was from a LOGITECH QuickCam Express webcam. Unfortunately the number of frames per second which could be extracted from this device was too low due to lighting circumstances and also the color representation was quite poorly by default. After changing the driver settings, the color quality improved somewhat however. Because it should not require extreme lighting (the test situation was lit with a TL-lamp) and because it was not wanted that a user should manually change the device’s settings, a digital video camera from SONY has been used. This camera obtained representative colors at a reasonable frame rate of 25 fps.

4.2 Skin Segmentation

According to theory a skin sample in normalized RGB space should resemble a Gaussian distribution. Skin patches were extracted from photographs of a Caucasian, Asian and an African person (see Figure 4-1) and this seems to be quite plausible. In figure Figure 4-2 this can be verified, where the histograms of the skin samples are shown in both linear normalized form. Although the size and position of the distribution varies it can be seen to approximate a Gaussian distribution. Furthermore in Figure 4-3 the results of the skin segmentation can be seen for a certain threshold and it seems to be working rather well. As can be expected facial hair (for the Asian and African example) and make-up (for the Caucasian example) are not detected as skin because of the different tone of skin color. Also for the Caucasian example the left side of her face has a different color than the right part probably due to lighting circumstances and thus not surprisingly this has its effect on the resulting skin segmentation.

Figure 4-1 The test pictures for an Asian, Caucasian and African skin type (Copyright WHO/ P.Virot, color versions are available at http://gene.science.uva.nl/~isinger/bsc/index.html )

Figure 4-2 Histograms of the skin samples taken from Figure 4-1 in the same order from left to right
4.3 Quantitative performance analysis

The system is meant to perform in real-time so speed is of the essence. To give an indication of the performance and to show the time-expensive steps an overview is given for the time the steps in the system take when they’re tested individually (see Table 4-1). An extra step is included which represents the time it takes to display the image in Matlab. This is not as trivial as it may seem, because in Matlab there are different ways to draw on a figure which can have a huge impact on the performance. It can be seen that the theory is not equal to the reality. Although the recorded input comes close the theory the live input is much slower for some yet unknown reason. Somehow the functions such as the display and preprocessing take much more time than they do while tested isolated. The whole system is supposed to run at 25fps so each iteration should take at most 40 ms. As can be seen this system will not reach this speed on my test system (an AMD Athlon 1800 MHz with 256 MB memory) but it is fast enough to test whether it works or not.

<table>
<thead>
<tr>
<th>Step:</th>
<th>Details</th>
<th>Time (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Acquisition</td>
<td>Recorded video</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td></td>
<td>Live acquisition</td>
<td>9 ms</td>
</tr>
<tr>
<td>Image Display</td>
<td></td>
<td>3 ms</td>
</tr>
<tr>
<td>Image Segmentation</td>
<td>Full lookup table</td>
<td>3 ms</td>
</tr>
<tr>
<td>Connectivity Analysis</td>
<td>For representative images</td>
<td>6 ms</td>
</tr>
<tr>
<td></td>
<td>For noise images</td>
<td>12 ms</td>
</tr>
<tr>
<td>Pre-processing</td>
<td></td>
<td>14 ms</td>
</tr>
<tr>
<td>Shape Analysis</td>
<td>Representative Image</td>
<td>6 ms</td>
</tr>
<tr>
<td>Shape Filter</td>
<td>Filters the three biggest</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td></td>
<td>areas</td>
<td></td>
</tr>
<tr>
<td>Tracking</td>
<td>Euclidean Tracking</td>
<td>3 ms</td>
</tr>
<tr>
<td>Gesture recognition</td>
<td></td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td>Sound production</td>
<td>Only called when sound</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td></td>
<td>production is needed</td>
<td></td>
</tr>
<tr>
<td>TOTAL TIME:</td>
<td>Theoretically (recorded)</td>
<td>About 35-41 ms</td>
</tr>
<tr>
<td></td>
<td>Theoretically (live)</td>
<td>About 45-51 ms</td>
</tr>
<tr>
<td></td>
<td>Reality (recorded)</td>
<td>About 52 ms (27 ms without display)</td>
</tr>
<tr>
<td></td>
<td>Reality (live)</td>
<td>About 108 ms (74 ms without display)</td>
</tr>
<tr>
<td></td>
<td>Skin Detect Only (live)</td>
<td>About 42 ms (without display)</td>
</tr>
</tbody>
</table>

Table 4-1 Performance time of the individual steps of the system
4.4 Qualitative performance analysis

This system has to turn video frames from a person into MIDI actions. This works, but there are some things which do not work alright which can be seen when using the application. The system assumes head & hands are inside the image, because it extracts the three biggest skin colored areas. Not obeying this regulations will cause the system to have unwanted behavior. It will let notes ring which you don’t want to. Also the rate of processing does not yet allow natural behavior from the user which one would expect from a program which allows the user to play an instrument. Furthermore the threshold for skin segmentation will have to be set very low if there are objects which resemble skin color somewhat. If there are no other objects present setting the threshold higher would yield less discontinuities in the skin areas (but also more noise of course). If not too high this will give better results for tracking the hands and heads. Setting the threshold too low on the other hand will cause the skin segmentation to find far less skin then present. Of course some types of input are bound to go wrong in this system. Two hands pressed against each other or moving in front of one another will cause this to be recognized as one object. Of course the same problems will arise with a hand - head combination. At this time my solution will take the three largest shapes and as such this can mess up the matching.
Conclusions

The objective of this project was to develop a system which allows the user to play an instrument using a digital camera without any accessories. The project succeeded into reaching this target. The additional requirements have been met partly. The system will not work in a standard home environment, because people will have cheap image acquisition devices such as webcams. The webcam used turned out to be insufficient, but the system did work with a more expensive digital video camera. This implies that if the quality of the image acquisition device in home situations will improve the system will work. The system will theoretically work well with each skin color under prescribed circumstances after calibration. The prescribed circumstances are that there are no large skin colored objects and that there is enough light so that the colors can be interpreted correctly. The processing of the system on the test machine could not be done in real time for the digital camera (25 fps). Nevertheless it is not very much slower and real time could be probably be achieved by writing it in a low-level programming language with optimized routines instead of using Matlab combined with some C code. Available software as openCV from Intel [18] and directShow from Microsoft [19] could be used for this. Finally the output is done using the MIDI protocol which is supported widely and thus satisfying the criterium earlier stated.

Although the systems functions, there are some other points which could be improved, which are:

- The path coherence function could be fixed which probably means that the problem of still objects has to be solved. Whether or not this will give better results than the current method has to be determined experimentally.

- The graphical user interface of the application could be improved to change settings for several parts of the system when needed (including calibration, skin segmentation threshold, type of tracking).

- The gesture recognition is rather primitive, other types of interpretation are possible. For instance Kalman filtering could be used. By analyzing the user input as a system estimations and predictions can be made of the state of the system. Using this information there are possibly better ways to allow a more natural workflow.

- The absence of head and hands when they leave the borders of the frame should be handled better. This could possibly be done by trying to detect whether and/or where one or more of the objects leave the frame. Another point of improvement could be by using a more advanced shape selection than the current one.
6 References

Appendix A Code Overview

This appendix lists the most important files with code which are used in the developed system and what they do. They are shown with their language in which they were written and what they do in Table A-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Language</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>video2midiFinal.m</td>
<td>Matlab</td>
<td>The main program which can be used for video sequences as well as for live input</td>
</tr>
<tr>
<td>skinDetect.m</td>
<td>Matlab</td>
<td>A program which shows skin colored areas in the input. It can be used to determine an appropriate threshold.</td>
</tr>
<tr>
<td>getSkinSample.m</td>
<td>Matlab</td>
<td>Obtains a skin sample from an input device</td>
</tr>
<tr>
<td>getGaussianParameters...</td>
<td>Matlab</td>
<td>Calculates the covariance and the mean of the given image in the specified color space</td>
</tr>
<tr>
<td>FromSample.m</td>
<td>Matlab</td>
<td></td>
</tr>
<tr>
<td>segmentRGBinC.c</td>
<td>C</td>
<td>After initialization creates a binary image with skin and non-skin areas from an acquired image</td>
</tr>
<tr>
<td>regionPropsInC.c</td>
<td>C</td>
<td>Calculates the mean and size of areas in a labeled image</td>
</tr>
<tr>
<td>getHighestValues.m</td>
<td>Matlab</td>
<td>Returns the indices of the N largest numbers in a vector</td>
</tr>
<tr>
<td>optimalEuclideanMatch.m</td>
<td>Matlab</td>
<td>Matches two list of coordinates based upon the least total Euclidean distance criterium</td>
</tr>
<tr>
<td>gestureRecognition.m</td>
<td>Matlab</td>
<td>Uses the trajectories from the tracking to recognize movements which should be transformed into actions on an instrument</td>
</tr>
<tr>
<td>midiMsgOut.c</td>
<td>C</td>
<td>Plays notes after initialization using the MIDI protocol</td>
</tr>
</tbody>
</table>

*Table A - 1 Code overview*