SINGLE PERSON POSE RECOGNITION AND TRACKING

Javier Barbadillo Amor

AKNOWLEDGEMENTS

When I decided to come to The Nederlands to do my Master Thesis I wanted an interesting topic, regarding that I was going to spend a lot of time with it. Also I wanted something quite new for me so I could learn the most and, if possible, an area currently demanded by companies. Out of all proposals they offered me, Computer Vision had all the three requirements. It is an interesting and useful science for many applications, completely new for me as I am a Telecommunication Engineer and we just study some general signal processing and, doubtless, a widely demanded area by companies, which are continuously developing new applications, some of them for commercial use.

As I was new to the topic I tried to assist to as many presentations and colloquiums as possible in order to keep me up to date with the state of the art. I also was invited by my supervisor E.A. Hendriks to join GATE meetings either in Delft or in Utrecht, where I learned really interesting techniques under research and I got inspired with some ideas that I later applied to my thesis.

I also had the opportunity, together with Feifei Huo, of testing the game with some children as part of a project in the Science Center for studying children's reactions to games, and they liked it and voted it for being part of the Exhibition Room, although they still noticed some limitations in the control. Watching them enjoying such a simple game behind a complex video processing system was a big motivation for improving the performance of this game during this whole term, and also I got some clues about the main shortcomings of the system.

All that I have learned through these 9 months was worth the big effort.

OUTLINE

CHAPTER 1

INTRODUCTION

Computer vision can be described as the design and implementation of systems capable of processing videos and images like a human eye would do, detecting and tracking objects on videos and interpreting the scenes in order to take decisions. As we don't know exactly how our brain processes the visual information through our eyes it becomes a challenge to design methods for detecting and tracking objects on videos and many times we have to deal with changing lighting conditions, foreground colors similar to the background and self occlusions of the objects, e.g. Thanks to the fast computers many commercial systems are being developed nowadays for computer vision such as interactive games, security systems, control interfaces, medical devices, etc, but there is still a lot to do and most applications only work under certain conditions and for certain situations.

The goal of this thesis is to research detection and tracking of a single person with just one camera and recognize and track human poses in order to improve the performance of an interactive spatial game that uses the horizontal position of the person and the performed poses for its control. Of course the results achieved with this research are not attached to this application only but to any application that can be imagined for using human poses detection as input. The use of one single camera restricts considerably the depth information and it is more challenging for detection and tracking when self occlusions occur but, on the other hand, it has the advantage that you only need a simple camera to run applications of this kind so it is interesting for commercial use on laptops or personal computers with a web camera. These types of games controlled by human poses can develop physical aptitudes and coordination while having fun playing with them and also relieves you from the load of holding a mouse or stick; it is just you and the screen.

This thesis is divided in 8 chapters. Chapter 1 is the Introduction, Chapter 2 is the Literature Review. The most important theories involved in this research are explained in Chapter 3. The first one is background subtraction with a mixture of Gaussians and consists on modeling the background with several Gaussians and then to check every pixel to see if they fit the background model or if they are foreground pixels. This algorithm is state of the art in the field and it is a good choice for the system because it works fine when light conditions are quite stable and it can deal with clothes similar to the background and new objects entering the scene as part of the background. The

particle filter for tracking the torso and head region is the second important theory involved in this thesis and it is based on random sampling from the image in order to fit a predefined shape of the torso and head into the foreground image. The statistical process is explained also in Chapter 3. Although there are more important theories and methods these two are the basics of the system because everything relies on the foreground image and the particle filter for tracking the torso and if one of those fails the rest will probably fail too.

The system used in this research is thoroughly described in Chapter 4 as well as its main shortcomings. Basically it was a single camera recording 25 frames per second, a computer capturing those frames and processing them to extract the torso center and hands positions for calculating the feature space and, finally, a classifier that recognizes one out of 9 predefined poses which are used for controlling a computer game. At the end of Chapter 4 the main contributions of this thesis to the system are presented. The first one is the improvement of hand detection by combining a general skin color model with a skeletonization of the human blob. The second contribution is a classifier able to detect non-poses without adding any new feature. A non-pose is everything that is not a predefined pose ad we assume that being able to detect non-poses will improve game control.

In Chapter 5 the experimental setup is described. Two things have to be considered here. One is the video recordings for processing and testing the system and the other is the pattern recognition toolbox that was used for training and experimenting with classifiers. Video recordings were made with people from different races, heights and clothes and were used both for testing the system and for extracting data to train the classifiers. The PRSD Toolbox is an important part of the experimental setup because it allows you to build features datasets with multiple labeling and export classifiers as an external pipeline which can be easily replaced in the system just by deleting the old one and copying the new one. Pavel Paclick introduced me to this powerful tool that he implemented for his company and he let me use it for my research.

The experiments and results are presented in Chapter 6. In the one hand I present the successful results, which are the improvement of the hand detection by combining the skin color detection with the human skeleton information and the incorporation of the non-pose class to the detector part. Hand detection improves also classification due to the best separation of the different classes. Non-pose detection allows detecting poses only when the user is performing a pose intentionally, and all the other positions will be detected as

non-pose. On the other hand I explain some unsatisfactory results, like those obtained by the particle filter for tracking the hands, because I learned from them too and they gave me ideas about which way I should follow.

Finally I discuss my conclusions in Chapter 7 and in Chapter 8 I suggest some future work that can be done and that I couldn't do due to the lack of time.

CHAPTER 2

LITERATURE REVIEW

In this section the most relevant literature related to the thesis is presented and divided into three topics, which are Background Subtraction, Foreground Segmentation and Classifiers. Following this order will help to understand the system proposed in [1] and the methods chosen.

2.1 Background Subtraction

As we are going to detect and track a single person, we only need the part of the image where the information about the person is. If we remove all the background pixels we can work with the human silhouette without caring about what there is in the background. Several techniques are used by other researchers for removing the background and Mixture of Gaussians is chosen for this system.

Chroma-keying was the first method for background subtraction and it is widely used for filming movies [7] where special effects and different backgrounds are added later to the foreground actors. It performs well when the background is a one-color surface and a lot of hardware devices were developed for this purpose. For using this method the background has to be blue or green in order to be easily removed and this is not challenging and handy for a commercial real time use so it is not appropriate for this system.

Other techniques are based on frame differencing, a simple method that uses image subtraction and a threshold to detect pixels that have changed considerably and are, therefore, foreground. But this simple method has the problem of finding a good reference image and an appropriate threshold, and if objects in the foreground have the same color as parts of the background won't be detected and in many cases shadows are detected as moving objects.

For dealing with the problems of changing backgrounds and light conditions it is necessary an adaptive method. The single Gaussian of Wren [6] is the simplest of this kind and models the color of each pixel with a Gaussian updated recursively with the new values of the pixel in YUV space. A better option for the system used in this thesis is Adaptive background mixture models for real-time tracking method proposed by Stauffer and Grimson [3]. Every background pixel is modeled as a mixture of K Gaussians with different weight depending on how long they have been part of the mixture. Every pixel

of a new frame is checked against the Gaussians in the background model in order to see if it is a background or foreground pixel. Pixels represented by Gaussians that don't fit the Background model will be consider as foreground. Other authors improved this method performance with similar approaches like the method presented in [5] by P. KaewTraKulPong and R. Bowden, which is able to remove shadows from the foreground image. After detecting the foreground pixels the differences in chromatic and brightness are checked. If they are within some thresholds they will be considered moving shadows. As background subtraction has an important role in the system an algorithm based on this method is used in this thesis. Although the foreground image resulting from this has some noise it is enough for later segmentation and can deal with real-time performing.

2.2 Foreground segmentation

Once we have the foreground image we can concentrate on detecting and tracking different body parts like hands, torso and face using the silhouette, colors, contours and other cues.

A very common method for detecting and tracking curves in clutter is the Particle Filter, explained in ConDensation of Isard and Blake [2]. This algorithm approximates the posterior of the state of the curves to be tracked by randomly sampling particles from the image. Different weights are assigned to particles depending on how high is the fitness coefficient computed from this set. Particles with low weights are discarded and particles with high weights are used again for the next iteration in order to make the filter to converge the most likely state. Thanks to the random sampling instead of analyzing all the pixels in the image we can deal with real-time, and for improved speed an integral image is used as in [9]. As this method for tracking the torso also includes the head region it is not necessary a face detection algorithm. Anyway, we considered the Haar cascade available in OpenCV thanks to Paul Viola et al [10], and some tries for sampling skin color from the face were made in order to apply it later to the adaptive skin color detection for hands.

For detecting the hands a general skin color model is used, as in [4], although an adaptive Gaussian model sampling color from the face was tried with similar results so it was discarded. The foreground pixels are classified as skin color pixels or non-skin color pixels by using a skin color mask that gives the likelihood of a color vector of being human skin.

As an improvement for hand detection, the idea of using a particle filter for tracking the hands explained by Antonio S. Miscilotta in [9] was tested but due to the distance of the user from the camera and to the occlusions and different hand shapes the hand regions weren't good enough for tracking robustly. Finally a skeletonization of the silhouette using the Distance Transform [11] was applied combined to the skin color detection to correct the position of the hands. Using the skin color we can detect the skin region and then use information of the skeleton to match the skin region with the extreme part of the arm. Although more advanced methods for skin detection are developed [12], this one was chosen for its simplicity and speed.

CHAPTER 3

THEORETICAL BACKGROUND

In order to completely comprehend the methods and techniques used in the system for detecting and tracking the poses of one person it is convenient to explain with more detail two theories of vital importance for this thesis in which the complexity of the method makes them more difficult to understand. The first one is Adaptive Background Subtraction using a mixture of Gaussians, as proposed by Stauffer and Grimson in [2], while the second theory is ConDensation algorithm of Isard and Blake [3], although we followed the explanation in [9] as it is more understandable and visually clear.

3.1 Background subtraction with Mixture of Gaussians

Background subtraction consists in thresholding the error between an estimate of the image without moving objects (the background model) and the current frame. Having in mind this definition the basics of this method is to model the background with a mixture of Gaussians and update it over time in order to get a binary image with only the pixels of the moving objects (in our system will be the human shape). Figure 1 shows an example of the current frame (left picture) and the resulting binary image (right picture) after applying background subtraction.

Fig. 1. Original frame and binary image after Background Subtraction

As we have to deal with changing lighting conditions, colors in the foreground similar to the background and boundary pixels that belong to different surfaces we need to model each pixel of the background as a mixture of Gaussians which better describes the recent history of the values for that pixel.

Every new frame, an on-line approximation is used to update the model as follows; the parameters of the Gaussians are updated (mean, variance, weight) according to the observed pixel, and basing on those parameters we can determine if the pixel is foreground or background. If the pixel value doesn't match one of the pixel's background Gaussians, it is considered foreground.

The values of a particular pixel over time are called a "pixel process", and we can write it as a time series of pixel values. We know the recent history of a particular pixel:

$$\{X_1,...,X_t\}=\{I(x_0,y_0,i),1<i<t\} \quad (1)$$

The intensity for a particular pixel is I, and represents a measurement of the radiance in the direction of the sensor of the first object intersected by the pixel's optical ray. x_0 and y_0 are the coordinates of the pixel and i is the time instant.

So the recent history of each pixel in the background, $\{X_1,...,X_t\}$, is modeled by a mixture of K Gaussian distributions (usually from 3 to 5), and the probability of observing the current pixel value is:

$$P(X_t)=\sum_{i=1}^{K}\omega_{i,t}*\eta(X_t,\mu_{i,t},\Sigma_{i,t}) \quad (2)$$

K is the number of Gaussians, $\omega_{i,t}$ is the weight of the ith Gaussian at time t, $\mu_{i,t}$ is the mean value of the ith Gaussian at time t, $\Sigma_{i,t}$ is the covariance matrix and η is a Gaussian probability density function.

$$\eta(X_t,\mu,\Sigma)=\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}e^{-\frac{1}{2}(X_t-\mu_t)^T\Sigma^{-1}(X_t-\mu_t)} \quad (3)$$

Different Gaussians are assumed to represent different intensities. The weight parameter of each function it set and updated according to how long has been part of the mixture.

When we observe a new pixel value we check against the K existing Gaussians in order of fitness (from the most probable to the least probable) until we find a match.

A match is defined as a pixel value within 2.5 standard deviations of any of the K distributions.

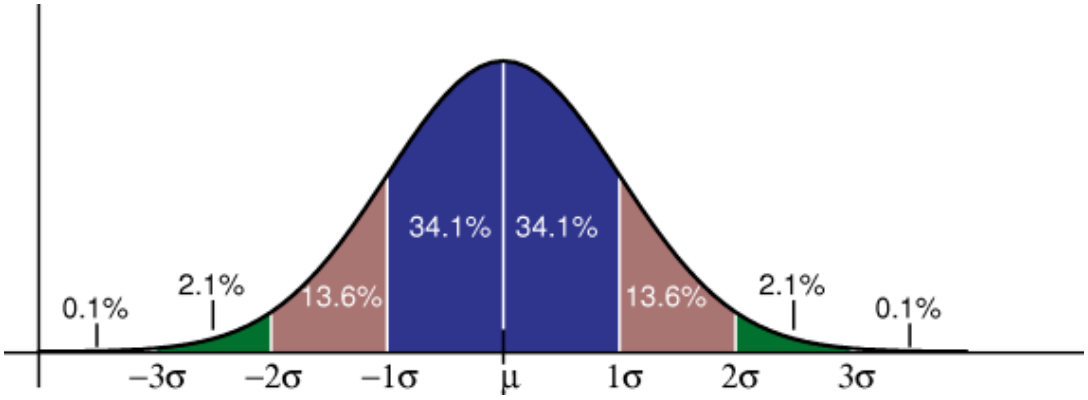


Fig. 2. This Gaussian Distribution could represent an intensity value of mean μ . Any new pixel value will be considered belonging to this distribution only if its value is within -2.5σ and 2.5σ .

If we find a match the parameters are updated, but if we don't, the least probable Gaussian is replaced with a new one with the pixel value as its mean. After updating, we decide which pixels are background and which are foreground.

3.2 Particle filter for torso detection and tracking

This is a method for tracking curves in visual clutter using a random sampling algorithm and applying probabilistic models of object shape and motion to find the posterior of the state.

The CONDENSATION algorithm uses "factored sampling", which improves speed by choosing some particles from the image, predicting the posterior and, depending on the fitness coefficient different weights are assigned to the particles so when the filter converges particles with more weight will be chosen again and again.

CONDENSATION uses learned dynamical models, together with visual observations, to propagate the random set over time.

The main goal is to find the maximum of a posterior distribution in the image that we don't know. That distribution can be parameterized as S_x. At the beginning initialization is done by taking N random samples from the region of

interest, also called particles. Those samples are dispersed along the parameter space x.

We can represent the prior probability of S_x at time t=0 as

$$\left\{S^{(n)}, \pi^{(n)}, n = 1...N\right\} \tag{4}$$

where S is the sample set and π is the weight of each particle calculated as follows

$$\pi_t^{(n)} = P(\omega_t^{(n)} \mid S^{(n)}) \tag{5}$$

The observed value is $\omega_t^{(n)}$. At time t=0 all weights are equal: $\pi_{t=0} = 1, \forall n$. Then, observations are updated using particles weights

$$\omega'^{(n)}_t = \omega_t^{(n)} \times \frac{\pi_{t-1}^{(n)}}{\sum_{n=1}^{N} \pi_{t-1}^{(n)}} \tag{6}$$

so we get the normalized observations. Then new weights are computed for estimating the posterior (or prior of the next iteration):

$$\pi_t^{(n)} = \frac{\omega'^{(n)}_t}{\sum_{n=1}^{N} \omega'^{(n)}_t} \tag{7}$$

That means that sampled particles corresponding to higher values of the parameterized distribution S will get higher weights. The higher the weight of the particle the higher the number of duplicated particles will be generated, while low weight particles are discarded

$$G_t^{(n)} = \pi_t^{(n)} \times N \tag{8}$$

G is the number of particles created from a father particle.

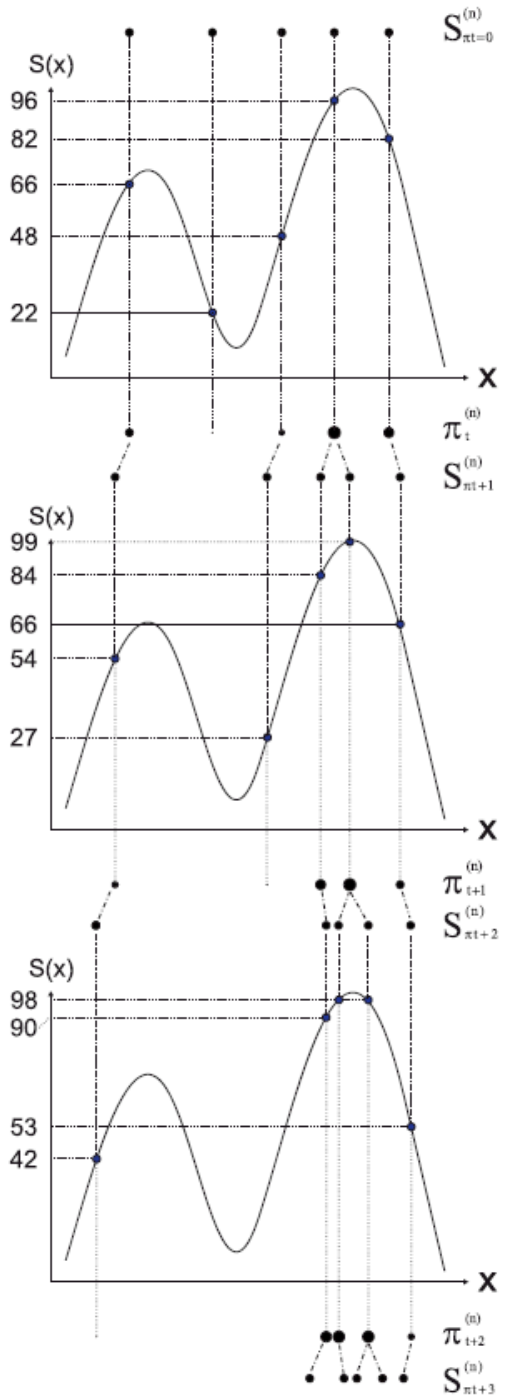


Fig. 3. Particles propagating through time [17]. The particle with highest measured value, 96, is duplicated in the second step. As two big values are measured for duplicated particles, in the third step three particles sons of the first one are in the maximum of S(x), with measured values of 98,98 and 90. After several iterations the filter will converge and particles will concentrate on the highest probability regions.

In our system we need to match the torso and head primitive shown in Fig. 4 with the torso and head region in the image.

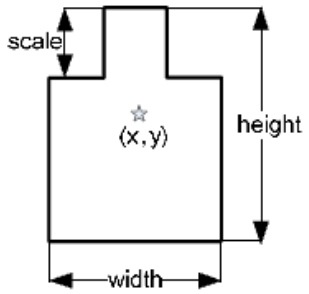


Fig. 4 Torso primitive with the Vitruvian man proportions.

The particles will be sampled from the image across the 2D space (x,y) and across the range of scales. We need to measure the values for those samples in order to assign a weight so we use the binary image with the human silhouette. For this purpose an inner (F) and outer (B) regions are used (Fig. 5).

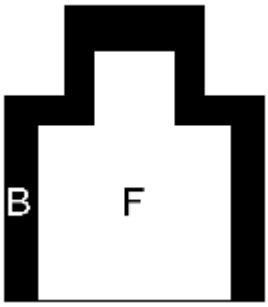


Fig. 5 Inner and outer regions for torso

Inner region follows the Vitruvian proportions so it must fit the torso and head region, while the outer region must only contain background pixels. Both regions have the same area so the right match will happen when the fitness coefficient is a global maximum

$$\omega = \frac{1}{Area(F)} \times \begin{Bmatrix} \sum F_{pixels} - \sum B_{pixels}, & \text{if } \sum F_{pixels} > \sum B_{pixels} \\ 0, & \text{otherwise} \end{Bmatrix} \tag{9}$$

Every foreground pixel sums 1 while every background sums 0. If F is matching perfectly the foreground region $\sum F_{pixels}$ will be maximum and $\sum B_{pixels}$ will be minimum so the expression in (9) reaches its maximum.

As the measurements for Particle Filter are basically areas, improved performance for real time is achieved when using the Integral Image instead of the original image. The Integral Image is a representation in which the value of a pixel (x,y) is equal to the sum of all the pixels above and to the left in the original image [10]. The following expression shows how to compute the values of the pixels in the integral image:

$$Integral(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i, j) \tag{10}$$

I(i,j) is the value of the pixel (i,j) in the original image. Integral(x,y) is the value that the integral image will contain after computing the expression. Figure 6 shows how to compute the area inside a rectangle using the Integral Image.

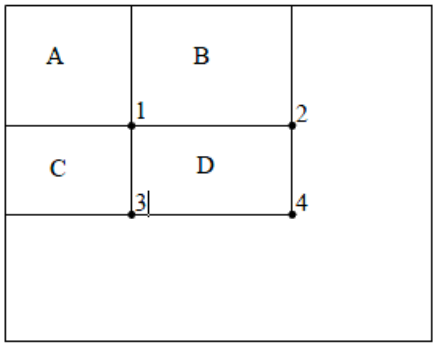


Fig. 6. The sum of the pixels inside rectangle D can be computed with the expression 4 + 1 - (2 + 3). This is because the value of the integral image in 1 is equal to the sum of the pixels inside rectangle A. The value at 2 is A+B. The value at 3 is A+C and the value at 4 is A+B+C+D.

3.3 Classifiers

Some explanation is needed for classifiers seeing that a particular application, PRSD Studio [8], was used for this purpose and some of the methods are not obvious.

PRSD Studio is a professional pattern recognition software for Matlab that allows the user to construct classifiers and embed them in other applications. Given a dataset containing features, each feature set can be indentified by adding one ore more labels. Using the labels it is easy to split the data into training and test sets, train a model, create a classifier and finally test its performance. Most relevant models are explained below.

K nearest neighbor classifier:

The well known algorithm for k nearest neighbor checks the k nearest neighbors of the new observation and returns the fraction of the class most represented by those neighbors. It is assuming that observations close in distance are likely to belong to the same class. The PRSD studio uses a different implementation that measures the distance to the k^{th} neighbor. The output is not a fraction but a distance, so this allows using this classifier as a detector, too.

Gaussian Classifier

This classifier generates a Normal distribution with the training data. 95 % of data is between -2σ and 2σ in one dimension. New observations are checked to belong to the distribution and if they are further than a threshold they are considered a different class. This method assumes that the data follows a Normal distribution and if that is not the case it won't be able to model the class.

Parzen Classifier:

Parzen is a non-parametric method that divides the feature space in bins of fix volume and estimates the contribution of each bin with a defined kernel. The final model is the sum of all bin's contributions.

CHAPTER 4

PEOPLE DETECTION, TRACKING AND POSE RECOGNITION

As the goal of this thesis is improving the performance of detection and tracking of human poses for controlling a real time spatial game, in this chapter we will describe the system, its main shortcomings and how this research contributed to improve it.

4.1 Overview of the system

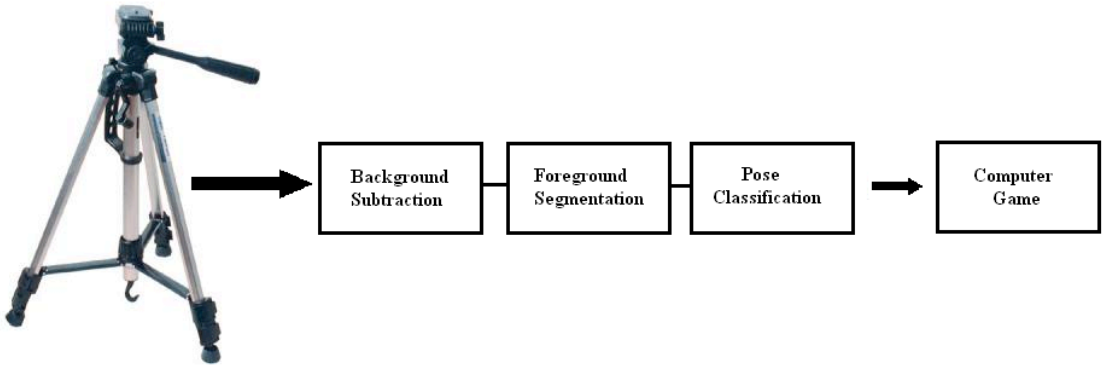


Fig. 7. Flowchart of the system. A video camera captures frames real-time. Then, background subtraction is performed to get the binary image of the human silhouette. Foreground segmentation will give torso and hands positions in order to calculate the features and classify the observed pose. Finally, pose numbers are used for controlling a spatial game.

The system consists of a video camera for recording frames in real time; a C++ code for performing Background Subtraction, torso and hands segmentation and feature extraction; an external pipeline linked to the codes for the pose classification and, finally, the spatial game which uses the pose number and the horizontal position of the person in the image for its control. In the following subsection a quick explanation of the system presented in [1] and its main shortcomings is given.

4.1.1 Background subtraction

We have a fixed camera in an indoor scenario so we can detect motion comparing pixels values from different frames. We first have to build a model of the background with no moving objects and this is done by a mixture of Gaussians. This takes a few seconds and then any moving object that enters the scene will be detected. The background is updated over time so it can deal with

lighting changes and objects entering the scene and staying there, as explained in Chapter 3.

As it is designed for detecting a tracking a single person, only one person is supposed to be in front of the camera. If that is the case the background subtraction will segment the human silhouette from the rest of the scene, creating a binary image with the foreground moving person.

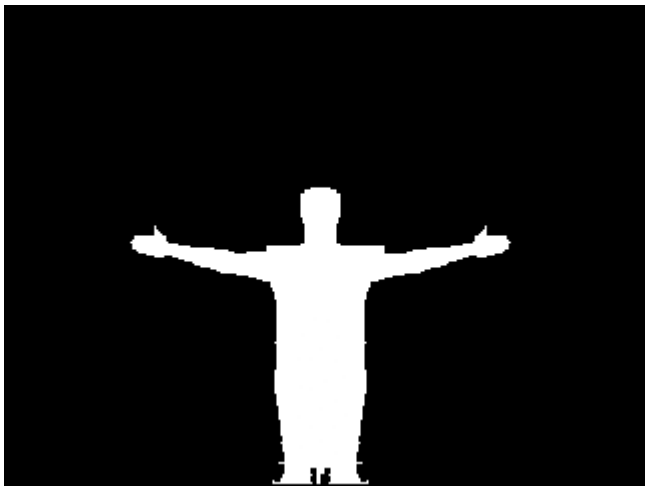


Fig. 8. Binary image with foreground pixels.

For a good performance a quite stable illumination is needed although the method can deal with light changes. Another limitation occurs when wearing clothes similar to the background. This causes sometimes holes appear in the silhouette, but might be solved by adjusting the parameters of the MOG.

4.1.2 Foreground segmentation

Having the human body blob we can concentrate on it for the segmentation of the different body parts. Particle filter is used for tracking torso region as explained in Chapter 3, and a general skin color model is applied for hand detection.

The particle filter performs well under almost every condition but there are some cases where it fails.

When the background subtraction is not good enough to generate a silhouette without holes (if clothes are similar to the background, for example) the particle filter might not be able to match the torso properly if there are holes in

that region. This is because the expression (9) will not become maximum since the pixels inside the holes don't contribute.
If another person enters the scene the particle will try to track both of them. This is specially annoying when playing the game because the control is lost. As the focus of this research is to improve performance for one person we didn't consider this problem.

Assuming that we have a good foreground silhouette, the particle filter will match the torso properly and the next step is to find the hands. The foreground pixels are now segmented into skin color and non-skin color pixels using this general definition

$$\left|\arctan\left(\frac{b}{r}\right)-\frac{\pi}{4}\right|<\frac{\pi}{8}$$
$$\left|\arctan\left(\frac{g}{r}\right)-\frac{\pi}{6}\right|<\frac{\pi}{18}$$
$$\left|\arctan\left(\frac{b}{g}\right)-\frac{\pi}{5}\right|<\frac{\pi}{15}$$
$$0.1<r+b+g<0.9 \qquad (11)$$

Although the method is simple it works fine when people are wearing long sleeves. The problem is when wearing short sleeves or clothes similar to skin color. The hand position is placed in the center of gravity of the skin region, but the center of gravity in the case of e.g. short sleeves will be probably on the forearm or elbow. This is probably the main shortcoming of the system and also affects to the training of the classifier.

4.1.3 Pose detection and spatial game

The feature space is constructed using the 2D coordinates of the torso center (which we obtained with the particle filter), the right hand and the left hand. A 6-feature-space of normalized distances and angles is built in order to deal with scale invariance and positions in the scene.

$$c_1=\frac{\left(x^l-x^t\right)}{s},c_2=\frac{\left(y^l-y^t\right)}{s},c_3=\frac{\left(x^r-x^t\right)}{s},c_4=\frac{\left(y^r-y^t\right)}{s},c_5=\arctan\frac{y^l-y^t}{x^l-x^t},c_6=\arctan\frac{y^r-y^t}{x^r-x^t} \qquad (12)$$

The scale parameter *s* is obtained from the particle filter, so we find that the features are depending on how the torso primitive fits the human blob. The positions are (x, y) coordinates and letters *t*, *l* and *r* stand for *torso, left* and *right* hand, respectively.

There are 9 poses defined for controlling the game, all of them easy to perform and remember, as shown in the picture.

Fig. 9 Predefined poses from [1].

The game is based on the proposal of Phong in [13]. The user controls a chameleon that has to bounce the balls and eat the flying bugs. The position of the chameleon is controlled with the horizontal position of the torso, so when the user moves to the left the chameleon will do the same. To be able to bounce the ball the color of the chameleon has to be the same as the color of the ball. For changing to the desired color the correct pose has to be performed.

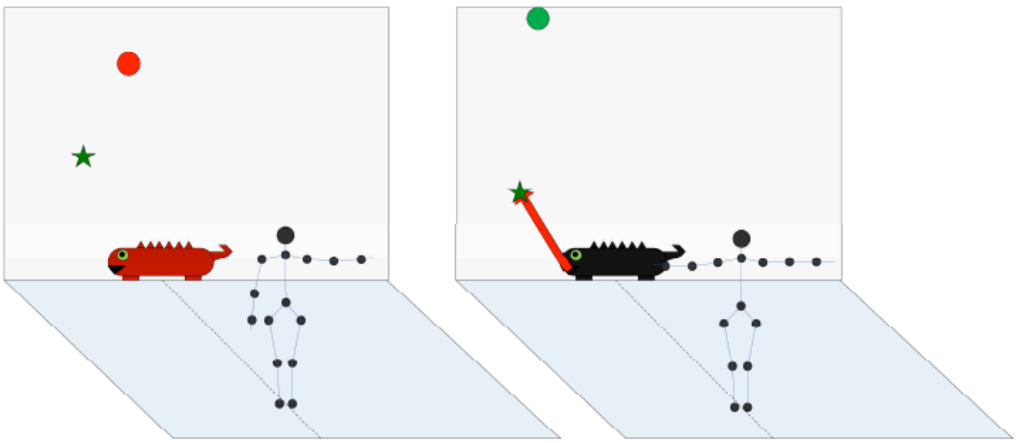


Fig. 10. Example of game control through pose detection and user position.

All the problems related to segmentation and pose detection will directly affect the game control. Hand detection when wearing short sleeves can be the reason for detecting a wrong pose, which will drive to a wrong color and therefore losing the ball. Another shortcoming is that the current classifier is trained only for the 9 classes so one of them will be detected, even if the user is performing something completely different. It will be interesting to have the chance to decide if the user is performing a pose or something different, let's call it a non pose.

4.2 CONTRIBUTIONS TO THE SYSTEM

From the point of view of a player the most interesting problems to solve are hand detection and adding a non-pose class. Solving the hand detection would also improve pose classification because the classifier could be retrained again using only samples where the hand is correctly detected. Adding a non-pose class without using a new feature is also an interesting improvement because we assume that it will give us more freedom to decide when to perform an action as response to a pose class detected.

4.2.1 Hand detection

The flowchart in Fig. 11 shows how the hand detection performs.

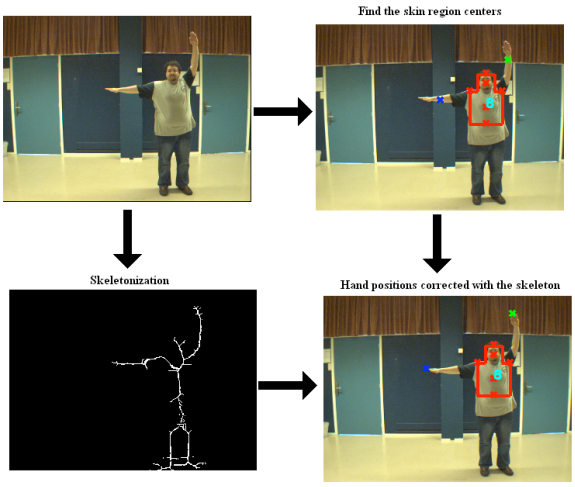


Fig. 11. Skeletonization combined with skin color for hand detection. From the current frame being processed torso is detected, the same as the center of gravity of the skin regions. A thinning algorithm based on Distance Transform is applied to the person's silhouette. Finally, the position of the hand is placed in the most extreme pixel of the skeleton using information from the skin blob detected previously.

Once we have the human silhouette the general steps for the improved hand detection are:

1- Find skin color regions and the center of gravity.
2- Apply Distance Transform to the foreground blob: get the skeleton.
3- If the skin blob is lower than the shoulder:
    - The hand is the point of the skeleton most far from the torso.
  If not:
    - The hand is the highest point of the skeleton

In order to make the searching of skeleton pixels faster we define a region of interest for each hand. For the right hand the region will be to the right of the torso while for the left hand it will be to the left. The region's size depends on the scale of the torso.

4.2.2 Adding a Non-Pose class

In order to make the classifier capable to detect non-poses a new 10-NN classifier was trained with a dataset build with samples from 17 videos (shown later in Chapter 5, figure 12).

The current classifier trained on 9 poses was replaced by this one trained also on Non-Pose class. The input features for the improved classifier are the same allowing embedding it easily in the system.

Clear Non-Pose samples were selected as explained in section 6.3 and two different datasets were built. Finally the classifier was trained with Dataset 2, which is a version of dataset 1 with more samples per pose class.

CHAPTER 5

EXPERIMENTAL SETUP

In this research, computer vision methods are implemented in C++ to be able to run in real time, although Matlab was also used for training and testing classifiers.

There are two main experimental setups. One part is related to the videos recorded, which were used for processing and feature extracting. The other part is related to the dataset built from those videos and the training of the classifiers with PRSD studio for matlab.

5.1 Video Recordings

For the experimental procedure 17 videos from 17 different people were recorded in Shannon Room in TU Delft. A single digital camera was used, connected to a computer.

The goal was to have different heights, skin colors, clothes, arms lengths, body shapes and different way of moving. In these videos people were told to perform 9 predefined poses, changing from one to another, while walking through the scene and sometimes performing other gestures.

5.2 PRSD Studio and Datset

A dataset was built processing all 17 videos and extracting the features from torso and hand detection, as shown in (12). Each feature set was labeled in PRSD Studio with the name of the person, the number of the frame and the class. This multilabeling allows to set different training sets and cross validate them, knowing all the time from which frame and person the samples are.

The dataset consists of 10 classes: the 9 predefined poses and the non-pose class. There are samples from 17 different people although not every person has samples for every class. Figure 12 shows a table summarizing the number of samples per person and per class.

It can be observed that not all the Pose Classes have the same representation but all of them have more than 100 samples. There are also more samples from Poses in total (2986) than from Non-Poses (543).

Person	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8	Pose 9	Poses	Non-pose
Berend	74	209	30	14	0	37	34	22	40	460	107
Carmen	13	39	13	5	0	7	4	0	0	81	21
Chunxia	31	34	3	13	0	27	11	0	0	119	10
Ewin	122	69	26	17	0	0	0	0	0	234	4
Hasan	28	40	21	15	15	12	3	15	128	277	60
Jinrong	0	21	13	0	0	0	0	0	0	34	0
Michiel	40	223	0	13	0	10	25	0	19	330	47
Ranish	67	51	0	0	0	0	0	0	0	118	10
Ronald	40	29	0	8	0	0	3	8	26	114	40
Saleem	125	261	18	38	191	8	109	159	171	1080	145
Shabir		38	0	0	0	0	5	0	0	43	11
Shenjie	12	1	0	0	0	0	0	0	0	13	14
Wanghua	20	7	0	0	0	13	7	16	20	83	69
Marion	0	0	0	0	0	0	0	0	0	0	5
Karfee	0	0	0	0	0	0	0	0	0	0	0
Umut	0	0	0	0	0	0	0	0	0	0	0
Yunlei	0	0	0	0	0	0	0	0	0	0	0
TOTAL	572	1022	124	123	206	114	201	220	404	2986	543

Table 1. Dataset 2.

The dataset is, therefore, composed by samples with 4 basic labels. One label is Person and allows selecting all the samples from a particular person or a group of people. Other label is Frame and contains the frame number, but also the Person is needed to discern from other frames with the same number. Finally we have two labels for classes. On the one hand, label Pose divides the data intro Pose class and Non-Pose Class and this is used for training detectors that can reject the Non-Pose class. On the other hand, label Class divides the data into 10 classes, from Pose 1 to Pose 9 and also Non-Pose class. See Figure 11 for a visual idea of labeling.

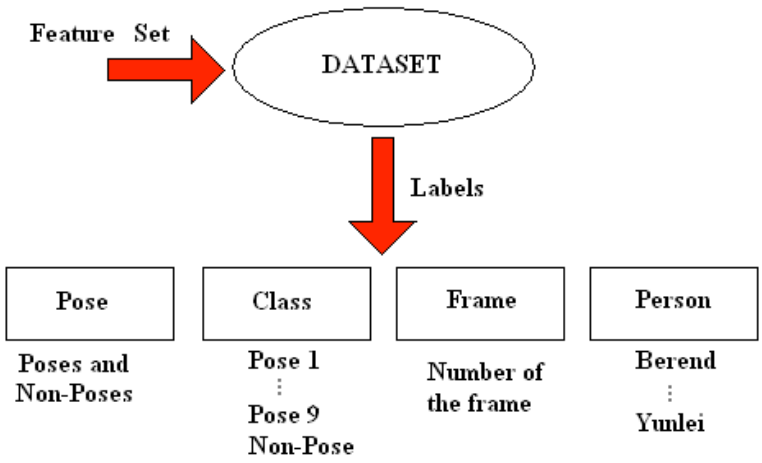


Fig. 12. Each sample consists of 6 features, defined in (9), and 4 labels.

CHAPTER 6

EXPERIMENTS, RESULTS AND DISCUSSION

In this chapter several experiments are explained in separated subsections. The first one is related to the implementation of the particle filter for tracking the hands and, although the results were not robust, some ideas were learned from that. The second part shows the experiments for improved hand detection with the combined method of skeletonization and skin color. Finally, in the last section two different datasets are used for training and testing classifiers. Each section consists of hypothesis, results and discussion. The conclusions from this research are presented in Chapter 7.

6.1 Particle filter for tracking the hands

Particle filter is a fast and accurate method for tracking curves if a mathematical model for shape and dynamics can be described. The particle filter was already implemented for tracking the torso in the current system and other researchers were using it successfully for tracking hands [9], so the idea of improving hand detection by particle filter tracking was considered as a good option and, therefore, investigated.

The same way the silhouette is used for measuring the state of the torso, the skin color regions can be used for measuring the state of the hands. Two regions are defined. The region A is the part that should match the skin region of the hand while region B should not contain any foreground pixel. These regions have the same area and are joined together in the bottom part to be able to deal with short sleeves. Figure 13 shows two pictures.

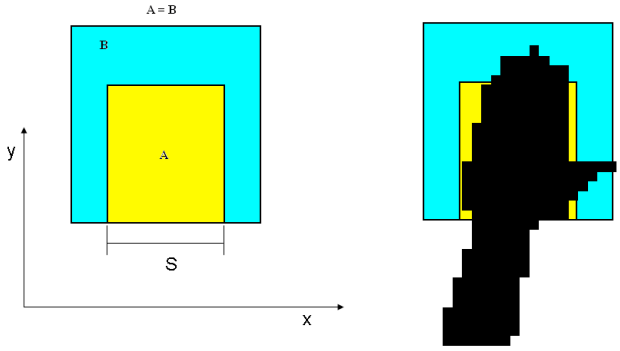


Fig. 13. Primitives for hand tracking with particle filter.

The first one is the primitive for the hand, where the parameters measured are x, y and the scale s, like for the torso. In the second picture the region A is matching the hand.

The way of calculating the probability is similar as for the torso, although the primitive is different. Foreground pixels sum 1 while background pixels sum 0. The fitness coefficient is computed as in expression (9) but changing F and B by A and B, so it will be maximum when region A has as much foreground pixels as possible and region B has as less foreground pixels as possible. In the bottom part both regions are joined together. This way, if the person is wearing short sleeves, skin color pixels from the forearm won't fall inside any of the two regions, as in Figure 13.

The initialization step is very important for the particle filter and it must know where the hands are at the beginning. For that purpose, a predefined pose (Pose 1) will launch the initialization process. As we know torso center position and scale, we define a bounding box for each hand containing the area where hands should be when performing pose one (see Figure 14).

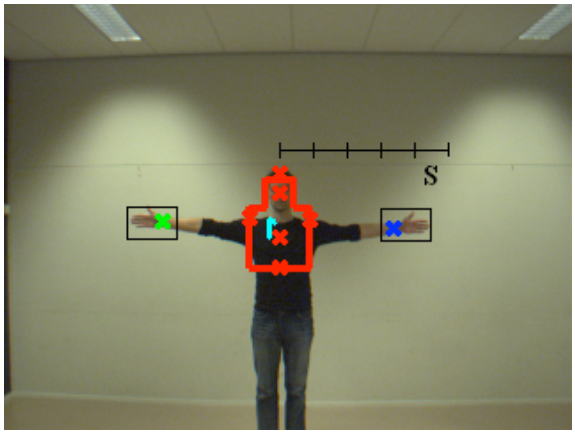


Fig. 14. For initialization hand position is tracked with skin color detection until Pose 1 is detected. Then, the particle filter tracks the hands. The ROI (Region of interest) for initialize the filter should be between 3 scales and 5 scales from torso center position when doing Pose 1, and 1 scale height.

In order to use the integral image for calculating the area as fast as possible, three rotation angles for the primitive corresponding to 0, 90 and 180 degrees were defined, so we can use the rectangle's vertexes in the integral image for calculating the area, as explained in Chapter 3.

Several videos were processed and different dynamic models were tested, including speed and acceleration. The particle filter was able to track the hands

until they were occluded or close to the torso region. This is because hands regions are not enough resolution for the particle filter to track them. As we can see in Figure 15 the region of the hands is confusing, noisy and small. When the hand gets close to the torso the particle filter might start tracking white blobs inside torso region as the hand region is occluded by the torso.

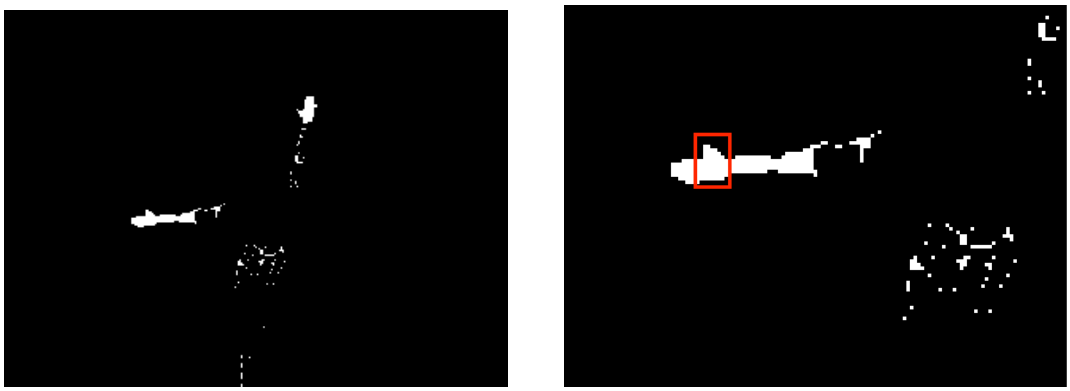


Fig. 15. Binary image containing skin color pixels in white, after a general skin color model was applied to the original image.

Using different angles for rotation of the primitive didn't help and the particle could track the same way without rotation. Also square shape of the primitive was the most robust, more than rectangular ones. As users don't keep the hands opened and they turn them during performance, square primitive adapts better to every situation, while rectangular primitives are only more efficient when the hand is open and the thumb is close to the hand. If the thumb is perpendicular to the hand and rotation is being used the primitive might be matching the hand as shown in Figure 15, at the right.

As no configuration managed to track hands robustly, particle filter for tracking the hands was discarded for this system. The hands closer to the camera and better resolution are needed for robust tracking.

6.2 Hand detection combining skeleton and skin detection

In order to improve hand detection a thinning algorithm was implemented using the Distance Transform in a similar way as in [11], which combined with skin color is able to find the correct hand position even if wearing short sleeves, like described in Chapter 4.

Assuming that right hand is always in the right side and left hand is always in the left side (reasonable assumption, regarding that all the poses are defined this way in this system), the skin color model described in Chapter 4 will detect two main blobs, one for each hand. With the current implementation this might not be enough to discern the hand from the forearm or elbow in the case of short sleeves, so more information is needed. Applying the Distance Transform to the grey scale foreground human blob and thresholding it will result into a 'skeleton' of the human. The method for finding the hand position is based on [14], and the following steps are required:

1- Get the binary image with the foreground human blob.
2- Apply Distance Transform.
3- Take the second derivative (Laplacian) of the image in step 2.
4- Threshold the image in step 3 to obtain the skeleton.
5- Using the blob's position, go through the skeleton until the farthest point from the torso. That will be the hand.

Figure 11 from Chapter 4 illustrates the process.

Several videos with different people wearing short sleeves were tested and the detection of the hands was a success. All the cases were the hand was detected somewhere in the middle of the arm are correct now.

There is still an unsolved problem related to hand detection. When arms are in a relaxed position, hanging and close to the torso (See Pose 2 in Fig. 9.) the thinning algorithm cannot separate the arm from the torso so the correction of the position of the hand has no sense. When this is the case the hand is detected using the general skin model.

The main difficulties come when the clothes of the upper body are similar to skin color and the hands are close to the body. In those cases the hand position jumps from the clothes to the arm intermittently and affects to the classification. In Chapter 8 some suggestions are discussed in order to deal with this shortcoming.

6.3 Non-Pose Class

Until now the system was always detecting one of the predefined 9 poses just because the k-NN classifier was trained on those 9 poses. Any incoming observation will be classified as one of the 9 poses although it might be something really different. It would be interesting to solve this limitation by adding another class called non-pose which represents everything different from a pose. This way the classifier would be able to discern between the 9 predefined poses and Non-Poses, and we assume that the game control will be improved taking advantage of this new class.

It was a problem to define what a Non-Pose is because a Non-Pose can be anything different from a pose and anything is not easy to define. For the purpose of defining Non-Poses it helped having a look to the feature definition in expression (11) and see that features are **relative distances** between hands and torso center and **angles**.

Some preliminary experiments were made recording a video in which a person changed from one pose to a contiguous one slowly and they showed that poses changed within certain angles in between. Those angles are called boundary angles. They are a good example of a clear Non-Pose because they are just in the middle angle of two predefined poses.

Having this in mind we can define two types of Non-Poses:

1- Clear Non-Poses: the poses in which one or both arms are in the middle way of two predefined poses.

2- Wrongs detections: the poses in which the torso detection is wrong or the hand detection is wrong.

As we want to give some freedom to the user the poses shouldn't be strict. The definition of poses includes therefore variations of 10 degrees up and down (more or less). It is important to mention that for the labeling process the decision is visually, so the definition of Pose and Non-Pose is just a clue. It might be the case that two people performing Pose 1 have a difference bigger than 10 degrees due to the matching of the torso, although both of them have their arms perpendicular, because angles are computed using torso center as reference. This will be important for the results discussion.

In figure 16 some examples of clear Non-Poses are shown.

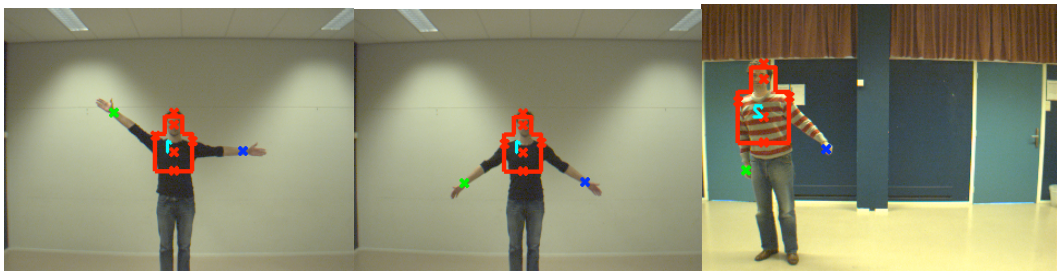


Fig. 16. Examples of clear Non-Poses, or poses right in the middle angle of two predefined poses.

Also notice that some wrongs detections are obvious Non-Poses for the 6-feature classifier but some others are describing a predefined pose using this feature set. With the current features it will be impossible to detect some wrongs detections like in Fig. 17.

Fig. 17. Example of a wrong detection. Hand positions are being detected wrongly because clothes are detected as skin color and a part of the background was detected both as foreground and skin color. Regarding that the classifier only gets features from hand and torso positions it is impossible to detect this as a Non-Pose.

In Chapter 8 some solutions to this problem are discussed, but in this thesis wrong detections are not included due to the lack of time.

In the following subsections two different datasets are analyzed and compared, presenting the results, with the objective of finding the best classifier that includes the new Non-Pose class. Only clear Non-Poses are used for testing and training the classifiers. The labeling was made by visual decision, labeling only what we assume to be very clear examples in order to build a robust classifier.

6.3.1 Dataset 1

The following table (Table 2) shows the Dataset 1. It was created with samples from 17 different videos from 17 different people performing the 9 poses. We processed the videos and extracted the 6-feature set for each frame. Every 6-feature set is labeled with the name of the person, the number of the frame and the class. The possible classes are poses from 1 to 9 and clear non-poses, although another label allows to distinguish between Pose (in general) and Non-Pose. It is important to notice that not all classes are equally represented due to problems with the background subtraction in some videos. Dataset 2 is a completed version of Dataset 1 and it is discussed in the next section.

Person	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8	Pose 9	Poses	Non-pose	W
Berend	74	209	10	14	0	37	34	22	40	440	107	
Carmen	13	39	13	5	0	7	4	0	0	81	21	
Chunxia	31	34	0	4	0	27	11	0	0	107	10	
Ewin	122	69	0	0	0	0	0	0	0	191	4	
Hasan	28	40	1	0	15	12	3	15	128	242	60	
Jinrong	0	21	0	0	0	0	0	0	0	21	0	
Michiel	40	223	0	13	0	10	25	0	19	330	47	
Ranish	67	51	0	0	0	0	0	0	0	118	10	
Ronald	40	29	0	0	0	0	3	8	26	106	40	
Saleem	125	261	0	10	191	8	109	159	171	1034	145	
Shabir		38	0	0	0	0	5	0	0	43	11	
Shenjie	12	1	0	0	0	0	0	0	0	13	14	
Wanghua	20	7	0	0	0	13	7	16	20	83	69	
Marion	0	0	0	0	0	0	0	0	0	0	5	
Karfee	0	0	0	0	0	0	0	0	0	0	0	
Umut	0	0	0	0	0	0	0	0	0	0	0	
Yunlei	0	0	0	0	0	0	0	0	0	0	0	
TOTAL	572	1022	24	46	206	114	201	220	404	2809	543	

Table 2. Dataset 1.

This dataset is not reliable for testing performance with poses less represented because there are not enough samples for testing some Poses in some people and could get, e.g., a test set with 2 samples from one pose, so a 50% error (1 match and 1 error) wouldn't be telling us anything reliable. Anyway, it is useful to learn about the behavior of the classifiers with these classes and to realize the influence of some people in the training/testing.

In order to find the best classifier to detect the new Non-Pose class, 2 main approaches were investigated. The first one is a **cascade of two classifiers**, where the first classifier is used as a detector to discard non-poses, so the second classifier can focus on Poses. The second approach is a **classifier trained on the 9 Poses and the Non-Pose class**.

Both of them have the advantage that we only need to replace the classifier pipeline (only one file) by the new one, so we are not adding new features or changing anything in the system, just replacing the classifier. This is important to consider because any improvement will be very welcomed if we just can

change a single file and don't need to create new features or make the system more complex.

a) First approach: Cascade of detector and classifier.

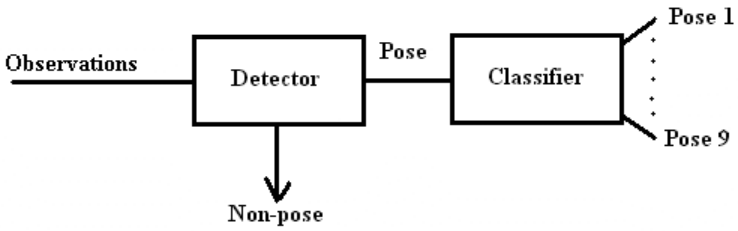


Fig. 18. Cascade of detector and classifier. In the first step Poses are detected and pass to the next step. All the observations detected as Poses by the detector enter the classifier, which is trained on the 9 predefined Poses.

The first part is a detector that can be trained either on Poses or on Non-poses. All the observations detected as poses will continue to the next step, which is a classifier trained on the 9 poses. We first separately trained detectors to find the best one and then we trained classifiers for the same purpose. Then we tested both together in a cascade.

Detector:

We want a detector to separate Poses from Non-Poses so later we can apply a classifier trained on the 9 predefined poses to the output, which will be the detected Poses. The goal is to detect most true poses, although some non-poses are also passing the filter. What we don't want is to lose true poses. A balanced solution has to be found giving more relevance to detect most predefined poses.

For obtaining realistic results we have to train the detector (whatever it is the type; Parzen, k-NN, Gaussian) manually and select the operating point so it minimizes the error on pose, without letting the error on Non-Pose grow much. The steps are:

1.- Split the data into training and test set: the test set should be one person if we want to apply leave one out method.
2.- Split the training set into 80% and 20% making sure that every class is equally represented.
3.- Train the model with the 80% and predict the ROC with the 20%.

4.- Select the operating point so it minimizes the error on pose.
5.- Build the confusion matrix with the test set.

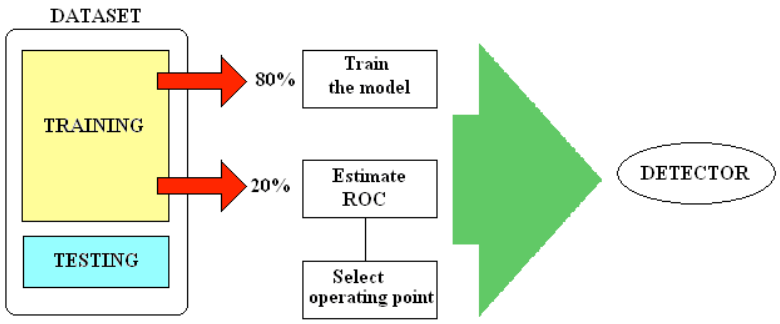


Fig. 19. Splitting the dataset for training the detector.

Following this method several detectors were trained and tested applying LOPO. They were trained both on Pose class and on Non-Pose class, obtaining similar results. Table 3 shows some results for detectors trained on Pose class. Training on Non-Pose class is discussed in section 6.3, for Dataset 2.

	Pose	Non-Pose
5-NN	0.0707	0.3146
9-NN	0.0599	0.2477
10-NN	0.0468	0.3202
11-NN	0.0433	0.3414
15-NN	0.0624	0.3362
20-NN	0.0494	0.3480

Table 3. Mean error for detectors trained on Pose class, cross validated with LOPO.

Although it is difficult to tell, it looks that best results are for 10-NN because ir minimizes both error on Pose and Non-Pose. Almost 70 % of Non-Poses are correctly detected while only the 5% of the Poses are missed (detected as Non-Poses). This 5% of errors in Pose detection are mostly due to 'Hasan' and 'Saleem' samples. Although with 10-NN this 5% can be considered acceptable, when investigating the Parzen detector some interesting results were obtained and high individual error rates from 'Hasan' and 'Saleem' were found. Depending on the experiment 'Saleem' might have more than 95% samples of Pose 5 detected as a Non-Pose, as shown in Table 3.

	Decisions		
Labels	**Pose**	**Non-Pose**	**Totals**
Pose 1	125	0	125
Pose 2	259	2	261
Pose 3	9	1	10
Pose 5	3	188	191
Pose 6	6	2	8
Pose 7	102	7	109
Pose 8	158	1	159
Pose 9	169	2	171
Non-Pose	33	112	145
Totals	864	315	1179

Table 3. One experiment with Saleem as test set with Parzen Detector.

In the case of Hasan, both Pose 9 and Pose 5 were having problems if Parzen detector was used, and sometimes 50% or more samples were classified as Non-Pose (Table 4).

	Decisions		
Labels	**pose**	**non-pose**	**Totals**
Pose 1	28	0	28
Pose 2	40	0	40
Pose 3	1	0	1
Pose 5	3	12	15
Pose 6	8	4	12
Pose 7	3	0	3
Pose 8	14	1	15
Pose 9	64	64	128
Non-Pose	3	57	60
Totals	164	138	302

Table 4. One experiment with Hasan as test set with Parzen Detector.

As shown in Table 2, Hasan has 128 samples of Pose 9 and Saleem has 171. They are the ones with more samples of Pose 9. If we have a look to the feature space we can see that for feature 3 Hasan is separated from the rest of the people and that is what makes the detector to misclassify Pose 9 sometimes (Figure 20).

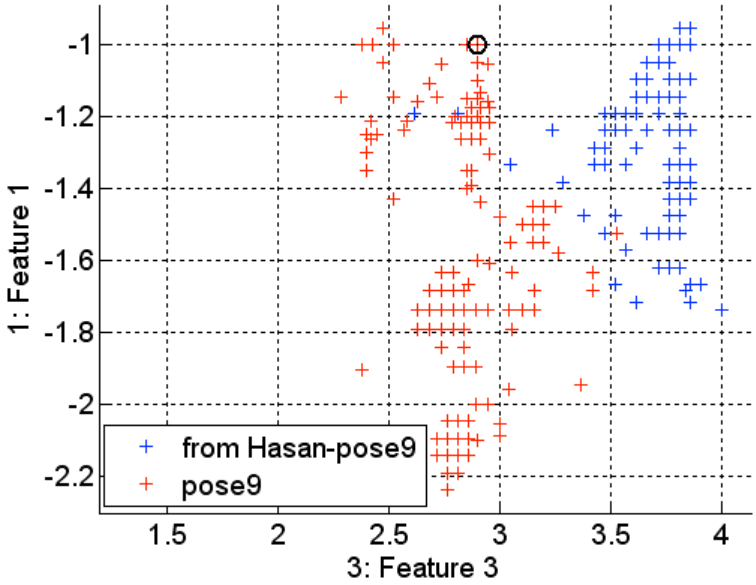


Fig. 20. Comparison of Hasan's Pose 9 (in blue) with the rest of the people in the dataset (in red).

Feature 3 is the scaled horizontal distance between left hand and torso (See expression (11)). Looking back into the frames we can take some samples to see that he is performing correctly pose 9, the same as Saleem, so we have two different people performing pose 9 in a right way and obtaining quite different results for feature 3. This is due to their different physical appearance and their way of performing the pose. Hasan raises his arm less than Saleem and has longer arm, so the distance is bigger. Saleem performs the pose with more energy keeping his arm more perpendicular. Both ways are correct because the game should be permissive with different people and slightly different ways of performing the pose. Also, at the beginning of the game people will be able to perform the poses better than after 10 minutes playing because they will be tired.

But those errors on detection not always happen, there is a high variance on the results. This is due to the random splitting data. Although there are always 80% of Pose 9 samples in the training set, depending on which samples are on that 80% the class can be more or less separated from Hasan's observations.

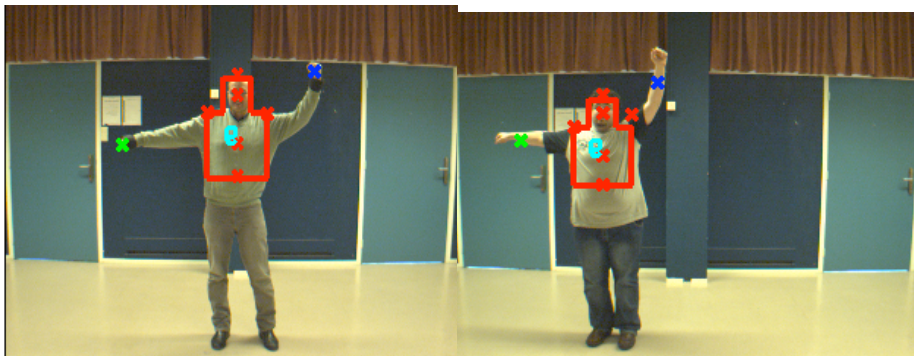


Fig. 21. Hasan and Saleem performing the same pose, but slightly differently.

Classifier on 9 poses for the output of the detector:

In order to decide which classifier is the most appropriate for classifying the 9 predefined poses several methods were trained and tested by LOPO, only using samples from Pose classes. This technique shows quite realistic results because none of the person's samples in the test set are in the training set, so it is the more similar case to real life. Here is a table with the mean errors per pose and per classifier of most relevant experiments.

	Mean error of Leave One Out method								
	5-NN	6-NN	7-NN	8-NN	9-NN	10-NN	Gaussian	Parzen(0.05)	Nearest Mean
Pose 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Pose 2	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
Pose 3	0.00	0.00	0.00	0.00	0.00	0.00	0.51	0.00	0.60
Pose 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
Pose 5	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Pose 6	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.05	0.00
Pose 7	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
Pose 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.81
Pose 9	0.07	0.07	0.07	0.07	0.08	0.08	0.01	0.07	0.08

Table 5. Mean error per class testing on LOPO. Best results are achieved by k-NN, especially with k lower than 10.

Finally 5-NN is selected for its best performance. As k becomes higher, the error in some classes raises due to the small number of samples in the test set. As K-NN computes the distance to the k^{st} neighbor, K should be lower than the minimum number of samples per class in the test set. Gaussian, Parzen, and Nearest Mean cannot describe this class distribution as well as K-NN because classes are overlapped sometimes and quite spread over the feature space, so k-NN can model better this distribution.

Cascade of Detector and Classifier:

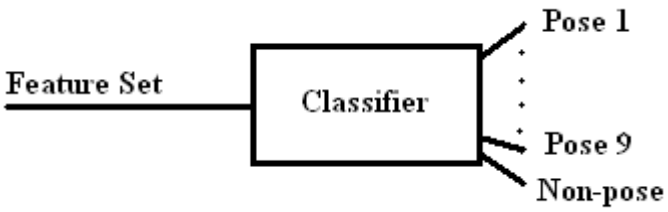
Having selected the best detector (10-NN) and classifier (5-NN) they can be tested together. Now some Non-Poses will be passing through the detector to the classifier, so we expect higher errors on the classifier part.

	Error
Pose 1	0,00
Pose 2	0,02
Pose 3	0,40
Pose 4	0,06
Pose 5	0,13
Pose 6	0,06
Pose 7	0,10
Pose 8	0,17
Pose 9	0,15
Non-Pose	0,30

Table 6. Error from LOPO testing the cascade. Observations from Poses and Non-Poses enter the detector, which lets Poses out and reject Non-Poses. Poses go to the classifier and a pose from Pose 1 to Pose 9 is decided. The mean error per Pose class is presented in the table.

The error is higher than when testing the classifier only with poses (almost on every class). Pose 3 is the class with the highest error, a 40%. This high rate is caused by error when Carmen samples are testing, which are detected as Non-Pose. Carmen has 13 Pose 3 samples and the training set is built with 11 samples. Probably having more samples will solve the high error rate.

b) Second approach: Classifier on 10 classes.



It is a classifier trained on the 9 poses and the non-pose class.

Using LOPO method we can test different classifiers to see which one performs better. As we saw that K-NN models the classes better than the other types we only show k-NN results on Table 7.

	Mean error of Leave One Out method					
	5nn	6nn	7nn	8nn	9nn	10nn
Pose 1	0.02	0.02	0.02	0.02	0.02	0.02
Pose 2	0.01	0.01	0.01	0.00	0.00	0.00
Pose 3	0.00	0.00	0.00	0.00	0.00	0.00
Pose 4	0.02	0.00	0.02	0.02	0.02	0.02
Pose 5	0.04	0.04	0.03	0.03	0.00	0.00
Pose 6	0.06	0.06	0.07	0.07	0.09	0.09
Pose 7	0.02	0.01	0.01	0.01	0.01	0.01
Pose 8	0.03	0.03	0.03	0.03	0.03	0.03
Pose 9	0.08	0.08	0.08	0.08	0.08	0.08
Non-Pose	0.22	0.21	0.21	0.22	0.22	0.22

Table 7. Results from several K-NN, trained and tested on 9 Poses and Non-Pose.

The new Non-Pose class makes classes less separable because Non-Pose class is not a compact class. Non-Pose class mixes with Poses and is not possible to separate it completely. The important issue is that most poses are correctly classified, being the maximum error 8% on Pose 9. Although a 22% of the Non-Poses are being classified as Poses, this is because they are very close to the boundary and it is difficult to tell if they are Poses or Non-Poses. Many of the wrong classified samples were labeled as Non-Poses, but could also have been labeled as the nearest Pose due to the small difference, and vice versa, many wrong classified Poses were labeled as Poses but could also have been labeled as Non-Poses. Figure 22 shows an example of a Pose that could perfectly have been labeled as a Non-Pose.

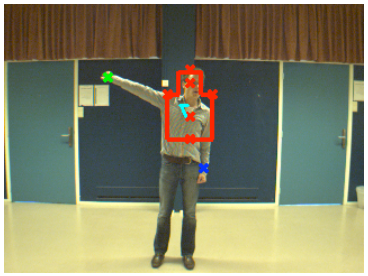


Fig. 22. Example of a person performing Pose 7 in a way that could be called a Non-Pose. During the whole video, this person performs poses raising his arms more than most people, so when labeling samples from his video they were considered as Poses. Compared to other people, this way of performing the pose is more a Non-Pose than a Pose.

The 6-nearest-neighbor minimizes the error on Poses and also on Non-Poses, although all classifiers in the table give similar results.

Analyzing the results on 6-NN, person by person, we can understand better the results. As the error is a mean value in many cases a high error value is caused

just by one person testing, so knowing why that person causes that error can help to solve it. Saleem has the biggest error on Pose 9 and it is 46,2%. This is because he has most of the samples for pose 9 and his way of performing pose 9 is quite different from Hasan's way, who has almost the other half of the samples. This result could be interpreted as Saleem is not a good source of samples for Pose 9, but when Saleem is part of the training set, the error on Pose 9 for the other people is almost 0%. That's why the mean error is 8%, and that means that Saleem makes more robust class 9, at least for the case of short sleeves people playing the game.

Two main conclusions can be learned from this Dataset 1. The first one is that some classes have more errors than other classes, and they are the ones with less number of samples, so the reason is probably that more samples are needed. The second conclusion is that many errors occur because the boundary between a Pose and a Non-Pose is too thin. Sometimes it is difficult to say whether the labeling was wrong or the classifier is performing badly, as in the case of Figure 22. Maybe the samples where it is difficult to tell which class they belong to should just be removed from the training.

Finally, the 10-class classifier shows better results on classification than the Cascade. Even Non-Pose class is classified with a 21% error while with the Cascade the error rate was more than 30%. This shows that a detector on Pose or Non-Pose is not able to separate better than a classifier trained on the 10 possible classes. Non-Poses seem to be easier to classify when comparing them to individual Poses (Pose 1, …, Pose9) than to the whole Pose group, probably because Poses have nothing in common among them.

6.3.2 Dataset 2

Seeing that the unbalanced Dataset 1 had more errors when classifying those classes with less number of samples representing them, we manually labeled some more samples for Pose 3 and Pose 4, having now more than 100 samples per class. The goal is to see if the results are improving, now that we have more than 100 samples per class. Table 1, in Chapter 5, shows the Dataset 2.

In this section the two previous approaches are not investigated thoroughly, but only the cases where Dataset 2 can be improving performance due to the more balanced dataset.

a) Detector on pose class:

As explained in 6.2, the data is split into Training and Testing set and an operating point is selected to minimize the error on Pose class detection.

The constrain for selecting the operating point can be moved for improving the detection of Non-Poses, but this way we pay the price of worse Pose detection. We don't want to miss Poses because we assume that they are more important for playing the game, but some Non-Poses can be missed and that will be acceptable. Table 8 presents the mean error on Pose and Non-Pose for several detectors.

	Err(Pose)	Err(Non-Pose)
5-NN	0.0585	0.2714
9-NN	0.0317	0.2933
10-NN	0.0304	0.2879
11-NN	0.0318	0.3284
15-NN	0.0387	0.3168
20-NN	0.0413	0.3463
Gaussian	0.3052	0.3478

Table 8. Mean error on detector for Dataset 2. Compared to table 3, which shows the results for the same detectors but trained on Dataset 1, we can tell that errors are reduced, so adding more samples from class 3 and 4 made the detector better. Although Poses 3 and 4 are not being treated as classes, having more samples from them make it possible to have a more compact region on the feature space corresponding to those classes. This will help when training the classifier.

We chose the 10-NN detector for its better performance, minimizing both errors. It is interesting to have a look to the mean errors per person on the 10-NN detector:

	Err(Pose)	Err(Non-Pose)
1	0.0109	0.1589
2	0.0988	0.4286
3	0	0.1000
4	0	0.2500
5	0.0505	0.1333
6	0	NaN
7	0.0030	0.1915
8	0	0.4000
9	0.2456	0.2000
10	0.0157	0.2207
11	0	0.2727
12	0	0.7857
13	0.0964	0.1449
14	NaN	0

Table 9. Errors per person on 10-NN detector. The table shows that a mean error of 3% on Pose detection was hiding an error of 24% on person 9. Also an error of 78% occurs on Non-Pose detection for person 12.

We can see that for person 9 (Ronald) the highest error on Pose happens and for person 12 (Shenjie) the biggest error on Non-Pose takes place. In the first case 28 poses out of 114 from Ronald are detected as Non-poses. This result is not good because we are missing quite a few true poses.

	Pose	Non-Pose	Totals
Pose	86	28	114
Non-Pose	8	32	40
Totals	94	60	154

Table 10. 28 poses out of 86 are detected as Non-Poses. As explained on section 6.2, this happens because samples from Ronald are confusing and different way of performance than most people.

It is not an unexpected result, as discussed in 6.2, that Ronald has a high error rate. He doesn't perform with the arms perpendicular to the torso, but with a slightly opened angle. As the definition for Pose cannot be mathematically precise in order to allow some freedom we added those Ronald's poses as samples of his Poses. As the rest perform poses more perpendicular, when we test with Ronald the detector classifies some Ronald poses as non-poses.

Fig. 23. More samples from Ronald performing other Poses.

Going back to the error rate on Non-Pose for person 12 (Shenjie), the 78,5% error rate is caused by 11 Non-Poses out of 14 detected as Poses. That is because for Shenjie some samples of Non-Pose were taken which are quite close to be Pose 1 (The opposite situation than for Ronald). Visually, it looks like a non-pose but not for the classifier. So again we have that the highest error rates are caused by confusing samples. This means that most errors are misclassified because the samples are close to the boundary, which is probably impossible to notice when playing the game.

b) Detector on Non-pose class:

Now the detector is trained on Non-Pose class instead of being trained in the Pose class. This means that if the two classes are spread and mixed, the detector will build the model focusing on the target class Non-Poses. The operating point is set using a constrain to 0.2 of the error on Non-pose (we don't want it to be higher than that) and minimizing the error on Pose, which is the most important class.

	Err(Pose)	Err(Non-Pose)
5-NN	0.0308	0.3607
9-NN	0.0375	0.3025
10-NN	0.0336	0.2980
11-NN	0.0482	0.2823
15-NN	0.0508	0.2604
20-NN	0.0326	0.3033
Gaussian	0.4696	0.2758

Table 11.

We can appreciate that the differences in performance are not big compared to training on Poses. The error on Pose is in general a bit higher except for the Gaussian, where it is a 15% higher. This is because it is not possible to model these classes with a Gaussian due to the nature of the features, which are mixed through the space. Training on Pose is not improving detection because it is an 'ill class', made from everything that is not a Pose. And, therefore more difficult to model.

c) Cascade of Detector and Classifier:

Cascades were discussed for Dataset 1 in 6.2 and Pose 3 was having a 40% error because Carmen was testing and most observations were misclassified. Now the error rate is 0 and all observations from Carmen are correctly classified with the cascade. This means that the detector part is improved and now is not detecting Pose 3 as a Non-Pose. More samples from individual Poses make more robust detection of the whole Pose class because all Poses contribute to model the distribution.

d) 10-class Classifier:

In this section a Classifier is trained on the 9 Poses and Non-Poses, having 10 class in total. The Dataset 2 does not improve classification for this 10-class classifier as shown in Table 12 and it also looks to be worse for 5-NN.

	Mean Error	
	5-NN	10-NN
Pose 1	0.02	0.02
Pose 2	0.01	0.00
Pose 3	0.01	0.01
Pose 4	0.00	0.00
Pose 5	0.04	0.00
Pose 6	0.06	0.09
Pose 7	0.02	0.01
Pose 8	0.03	0.03
Pose 9	0.08	0.08
Non-Pose	0.22	0.22

Table 12. Mean error for 5-NN and 10-NN classifier with dataset 2.

Adding more samples from Pose 3 and 4 didn't improve the classifiers, which were having low errors for those classes. Actually it improved performance on detectors. This is because for training the detector the 80% of the training data is taken with the same representation per class. Before having the Dataset 2, the 80% of Pose 3 when Carmen was testing the detector was the 80% of 11. This means training the detector with 8 samples from Pose 3, so it is reasonable that Pose 3 is detected as a Non-Pose due to the low number of samples for training. With Dataset 2, when Carmen is testing the 80% of Pose 3 is 88 samples, so now Pose 3 has more chances to be detected as a Pose.

CHAPTER 7

CONCLUSIONS

This thesis focused on the research of improved methods for a Single Person Pose Detection and Tracking System. This chapter summarizes most important conclusions, especially from experiments presented in Chapter 6.

The particle filter for tracking the hands is not appropriate for this system due to the low resolution of the hand region and the frequent occlusions and self occlusions of the hand, different possible shapes, etc.

Distance transform combined with skin color is a simple method but accurate and robust detecting the real hand position when arms are separated from the body. This method is suitable for this system because all the Poses except Pose 2 are performed with the arms separated from the body. The main contribution of this improvement is the correct detection of the hand when wearing short sleeves, which now allows detecting poses in which the wrong detection of the hand was causing the detection of a contiguous pose.

Non-Pose detection is a difficult task to solve because it is a class difficult to define. Results show that it is possible to detect Non-Poses when they are in between Poses and their labeling is clear. Most errors on detection occur because the labeling is confusing or because people perform differently, but still 78% of Non-Poses are detected in the best case, which is the 10-Class classifier.

The differences between Dataset 1 and Dataset 2 were notable for training the detectors and Dataset 2 reduced the error both on Pose and Non-Pose detection. Dataset 1 had not enough samples for training a detector able to model robustly Pose 3 and Pose 4 as part of the Pose class. With the completed Dataset 2 on those two classes, the rate of Poses 3 detected as Poses when Carmen was on the Test set increased from 60% to the 100%. This shows that having a complete dataset with enough samples (more than 100, at least) is very important for building good methods.

CHAPTER 8

FUTURE WORK

Building new dataset with more samples per person and per class is one of the easy tasks that can be done, also including the new hand detection. Due to the lack of time it wasn't possible to investigate classification with samples only regarding hand detection in the hand, not in the forearm, but this should improve classification because classes will be more compact if the hand position is restricted to the hand area.

Wrong detections are also part of Non-Poses, but more features are neede to describe them. A face detector could be an interesting feature. The torso detected should be related to the face position. This is, therefore, a good clue about if the torso is properly detected. If the face matches the torso, then we probably are tracking the body properly but if not, we have a wrong detection of the torso and, therefore a Non-Pose. The face detector can be also used for extracting skin color from the face in order to build an adaptive skin color model.

Another improvement that can be considered is defining the poses again. With the improved hand detection, more reliable, elbow possition can be found also useing the skeleton (although improved skeletonization will be helping). Elbow detection can be a new feature and also allow including different poses.

BIBLIOGRAPHY

[1] E.A. Hendriks and F. Huo. Detection, Tracking and Recognition of Human Poses for a Real Time Spatial Game.

[2] M. Isaard and A. Blake. CONDENSATION – Conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1):5–28, 1998.

[3] Stauffer and Grimson. "Adaptive background mixture models for real-time tracking". Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, pp. 246-252, 1999.

[4] Fatih Porikli and Oncel Tuzel. Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis.

[5] P. KaewTraKulPong and R. Bowden. An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection

[6] C. Wren, A. Azarbayejani, T. Darrel, and A. Pentland. Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 19(7):780–785, 1997.

[7] S. Wright. Digital Compositing for Film and Video. Focal Press, 2001.

[8] PRSD Studio http://prsdstudio.com software packages for Matlab.

[9] Antonio S. Micilotta. Detection and Tracking of Humans for Visual Interaction. Centre for Vision, Speech and Signal Processing School of Electronics and Physical Sciences University of Surrey Guildford, Surrey GU2 7XH, U.K. September 2005

[10] Paul Viola et al. Rapid Object Detection using a Boosted Cascade of Simple Features. CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001.

[11] Frank Y. Shih and Christopher C Puff. A Skeletonization Algorithm By Maxima Tracking On Euclidean Distance Transform. Pattern Recognition, Vol. 28, No. 3, pp. 331 341, 1995.

[12] J.F. Lichtenauer, M.J.T. Reinders, E.A. Hendriks. A SELF-CALIBRATING CHROMINANCE MODEL APPLIED TO SKIN COLOR DETECTION. Proceedings of the 2nd International Conference on Computer Vision Theory and Applications (VISAPP), March 2007.

[13] Berendsen, B. (2008). Tracking and 3D body model fitting using multiple cameras. Master's thesis, UU, Department of Computer Science, INF/SCR-2007-066, 2008.

[14]http://pinkjade8848.spaces.live.com/blog/cns!E4159959CD42C507!195.entry

[15] R.P.W. Duin, M. Loog et al. Advanced Pattern Recognition. Ph.D Course, TU Delft, 16 May 2 June 2008.

[16] http://en.wikipedia.org/wiki/Main_Page