

# Detection Tracking and Recognition of Human Poses for a Real Time Spatial Game

Feifei Huo, Emile A. Hendriks, A.H.J. Oomes  
Delft University of Technology  
The Netherlands  
f.huo@tudelft.nl

Pascal van Beek, Remco Veltkamp  
Utrecht University  
The Netherlands  
phbeek@students.cs.uu.nl

## Abstract

In this paper, we present an approach to detect, track people, and recognize poses. The detected poses are used for controlling a real time spatial game. In the people detection, tracking and pose recognition system, body parts such as the torso and the hands are segmented from the whole body and tracked over time. The 2D coordinates of these body parts are used as the input of a pose recognition system. By transferring distance and angles between the torso center and the hands into classifier feature space, simple classifiers, such as the nearest mean classifier, are sufficient for recognizing predefined key poses. The output of the classifier, that is the identification of the pose, is used to control color and actions of the virtual actor. The position of the virtual actor is steered by the detected position of the user in the image.

**Keywords:** Pose Recognition, Spatial Game, Real Time.

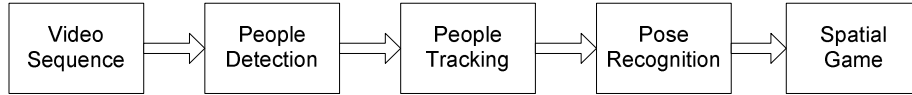
## 1 INTRODUCTION

Nowadays video-based applications have become more and more widespread [1]. A well-known video-based application is man-machine interaction, in which people can use their facial expressions, gestures and poses to control e.g. virtual actors or (serious) games. The essential ingredient for an effective man-machine interaction experience is that the system indicates its level of understanding of the user's movement. Therefore, human motion analysis plays an important role in man-machine interaction. Generally, there are two approaches to obtain the movement of human body. One approach is marker-based, in which users need to wear specific suit with sensors on it. These sensors are used to capture the motion of different body parts. The other approach is vision-based, in which users are totally free of any obtrusive sensors. The movement of users is analyzed from the recorded video data. Compared with the first approach, the second one may have less accuracy of reading motion information. However, it is more convenient and friendly to users, especially for gaming applications. Therefore in this paper, we propose a vision-based people detection, tracking, and pose recognition system. It directly uses the captured video frame as input, then gives the 2D position and pose of the people if there are people appearing in the scene. The position and pose information is connected to a spatial game system, and used as the control command of the spatial game. The remainder of the paper is organized as follows. In Section 2, we give a brief introduction of previous researches. The methodology of the proposed approach is described in Section 3. In Section 4, we show the spatial game application. At the end, the conclusion is drawn in Section 5.

## 2 PREVIOUS RESEARCHES

Although there has been published a significant number of research papers on human body tracking and pose recognition, many research issues still remain to be solved. In [2] the body parts are reliably labeled and located by 2D contour shape analysis. Tracking performance is significantly increased by taking color into account. The limitation of this method is that all the body parts need to be segmented although they may not be needed for certain applications. In [3], an approach to estimate 3D human pose with multiple cameras is proposed. It can deal with cluttered scenes and self-occlusion under some constrains. But the processing time for pose estimation is about 45

seconds per frame, which is not suitable for real-time applications. In [4] an exemplar based approach is used to localize and recognize human poses. However, the pose detector is only learnt from walking poses, so they can not detect people with other poses than walking. In this paper, we present an approach for real-time people detection, tracking, and pose recognition, which can handle a variety of poses and is fast enough to steer a real time game. The output of the pose classifier is used for controlling appearance and actions within this spatial game. Fig.1 gives the flowchart of the proposed system.



**Fig.1.** The flowchart of the proposed system.

### 3 METHODOLOGY

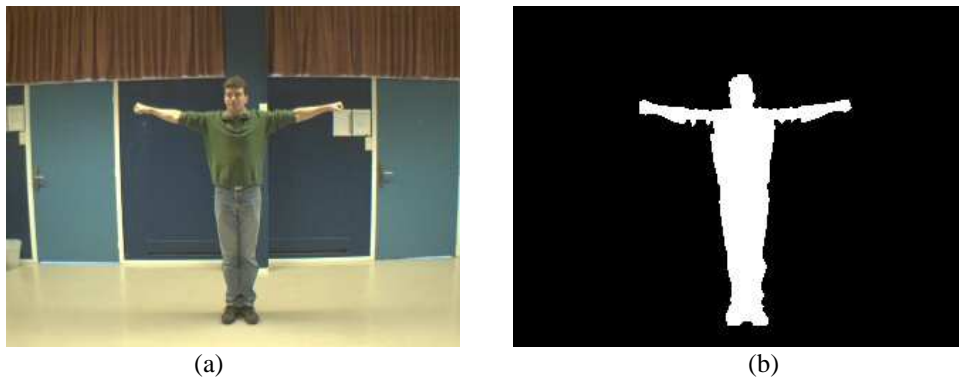
#### 3.1. MOTION EXTRACTION

In an indoor scenario, cameras are usually at fixed locations, so we do not need to consider motion of the scene for foreground object extraction. Therefore, background subtraction is a quite suitable method in this case. Compared to temporal differencing, the background subtraction will not only give the edges but the whole silhouette of the moving objects. The first step of the background subtraction is to build up a background image, which should not include any foreground objects. In our implementation, the background image is built by using a mixture of Gaussian model in [5]. This method is also robust to cluttered scenes. The background image is updated by using current frames, in order to deal with the change of lighting conditions. After the background image is obtained, the pixel-wise difference between the current frame and the background model is used to classify each pixel as either foreground or not. A difference image  $D'_{i,j}$  between an input image  $F'_{i,j}$  and the background image  $B'_{i,j}$  shows only the moving object regions, while the stationary background is suppressed (see eq.1).

$$D'_{i,j} = F'_{i,j} - B'_{i,j} \quad (1)$$

A foreground binary image is obtained by using the difference image  $D'_{i,j}$  and a threshold  $T_d$  (see eq.2).  $T_d$  can be obtained experimentally and is found out to be not very sensitive for different indoor scenes. Fig.2 shows the input image  $F'_{i,j}$  and its foreground binary image  $R'_{i,j}$ .

$$R'_{i,j} = \begin{cases} 1 & \text{if } D'_{i,j} \geq T_d \\ 0 & \text{else} \end{cases} \quad (2)$$



**Fig.2.** Extraction of foreground binary image: (a) input image  $F'_{i,j}$ , (b) foreground binary image  $R'_{i,j}$ .

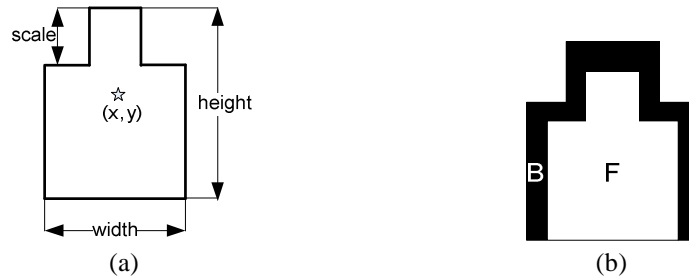
## 3.2. TORSO AND HAND SEGMENTATION

After we obtain the motion-based human body blob, next step is to segment different body parts. This can be based on the selection of various features, such as shape, edge, silhouette, contour and color of human's body. We use a 2D silhouette model to detect the torso and use skin color for the detection of hands.

### 3.2.1 Torso segmentation

After the foreground binary image is built, we use a geometrical characteristic of persons to detect them. The prior knowledge of human geometrical structure that we use is a head-shoulder-upperbody model, shown in Fig.3. The parameters in this 2D human shape model are position and scale, so this model is described as  $P = (x, y, scale)$ . If the scale parameter in the 2D model is determined, the width and length of human upper body can also be derived according to the shape characteristic of persons. Since the recorded video data is from one single camera, the position of the human is given by 2D coordinates, that is x and y coordinate. In the real 3D world, this method can be easily extended to multiple view approaches by fusing the 2D data derived from synchronized cameras into 3D real coordinates.

After the definition of the human model, the problem comes up with how to use this 2D model to determine if there is a person in the image or not. Generally the intuitive solution is exhaustively searching the image for the defined model. Although there are only three parameters in this model, x y coordinate and scale, it still needs a large amount of calculations to find the global optimal solution. In the situation of more than one person appearing in the image simultaneously, this exhaustive search is too far away from real time applications. So it is impractical to use this model directly for template matching. However, this detection and tracking problem can be seen as a state estimation problem. To solve it, statistical methods, such as particle filters, can be used, which is especially suitable for non-Gaussian and multi-model situations. In a particle filter, the probability of detection of a person in the image is represented as the fitness coefficient of the defined 2D shape model. The definition of this fitness coefficient is the same as [6]. The template used to calculate the fitness coefficient is shown in Fig.3 (b). It is composed of two regions: foreground region F and background region B, which is surrounding region F.



**Fig.3.** (a) One sample in a particle filter, (b) two dimensional human model to calculate fitness coefficient [6].

In order to construct the probability, the fitness coefficient should be a value between 1 and 0. This requirement is satisfied by using the following definition (see eq.3).

$$\omega = \frac{1}{Area(F)} \times \begin{cases} \sum F - \sum B, & \text{if } \sum F > \sum B \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Where  $Area(F) = Area(B)$ ,  $\sum F$  is the summation of the pixel values in region F,  $\sum B$  is a summation of the pixel values in region B. Hopefully foreground pixels with a value 1 will fall into region F as much as possible and background pixels with a value 0 are only included in region B. Under this optimal situation the fitness coefficient will be 1, which means the foreground binary image satisfy the 2D model perfectly. We use this fitness coefficient  $\omega$  as the probability of a person present in the image. The larger the value of the fitness coefficient is, the higher the probability of a person existing in the image. By fitting this 2D model on the foreground binary image, people's head and torso can be segmented from other parts.

### 3.2.2 Hand segmentation

In addition to the 2D model mentioned in 3.2.1 for human's torso detection and tracking, foreground pixels are further segmented into skin-color and non-skin-color regions. A skin color model in the RGB color-space is used to select skin color pixels on the foreground image. This human skin color model is similar to the model in [7]. If foreground pixels mapping into the RGB color-space satisfy the following conditions in eq.4 [7], they will be considered as skin-color pixels. People's face and torso region are excluded from skin color detection by masking the head and torso region estimated from 3.2.1.

$$\left| \arctan\left(\frac{B}{R}\right) - \frac{\pi}{4} \right| < \frac{\pi}{8}, \quad \left| \arctan\left(\frac{G}{R}\right) - \frac{\pi}{6} \right| < \frac{\pi}{18}, \quad \left| \arctan\left(\frac{B}{G}\right) - \frac{\pi}{5} \right| < \frac{\pi}{15} \quad (4)$$

After the skin color pixels are selected, two post-processing steps are used to get rid of false positive detections. The first step is to delete regions with a very small size, which are impossible to be hand regions. In the second step, a motion mask is introduced to exclude regions which are far away from previous hand locations. It limits the movement of hands within a certain bounding box. From the remaining two largest blobs, we calculate the centers of gravity and use them to represent the position of the hands.

### 3.3. FEATURE SPACE CONSTRUCTION

The input of the proposed pose recognition system are 2D positions of the torso center and the hands. However, we transfer them into normalized feature space and train the classifier in this new feature space. The reason is that the pose recognition system should be scene invariant. That is, no matter where the person is in the scene, or how far the person is from the cameras, the predefined key poses should be recognized. Therefore the feature space is built by using angles and relative positions between hands and torso center. We construct the following 6 feature components, denoted as  $F_{set} = \{c_1, c_2, c_3, \dots, c_6\}$ :

$$c_1 = \frac{(x_2^l - x_2^r)}{s}, \quad c_2 = \frac{(y_2^l - y_2^r)}{s}, \quad c_3 = \frac{(x_2^r - x_2^l)}{s}, \quad c_4 = \frac{(y_2^r - y_2^l)}{s}, \quad c_5 = \arctan \frac{y_2^l - y_2^r}{x_2^l - x_2^r}, \quad c_6 = \arctan \frac{y_2^r - y_2^l}{x_2^r - x_2^l} \quad (5)$$

Here  $(x_2^l, y_2^l)$ ,  $(x_2^r, y_2^r)$  and  $(x_2^t, y_2^t)$  are the 2D positions of the torso center, left hand and right hand.  $s$  is the scale parameter in 2D model. The classifier will be trained and tested on this 6D feature space  $F_{set}$ .

### 3.4. POSE CLASSIFICATION

The key poses are designed for gaming control, so they should be easy for users to remember and perform. We also choose the number of the poses not too high to make it easier for users. In our system, we defined nine poses in total, as shown in Fig.4. From top row to bottom row and left to right, these nine poses are labeled as pose1 to pose9.

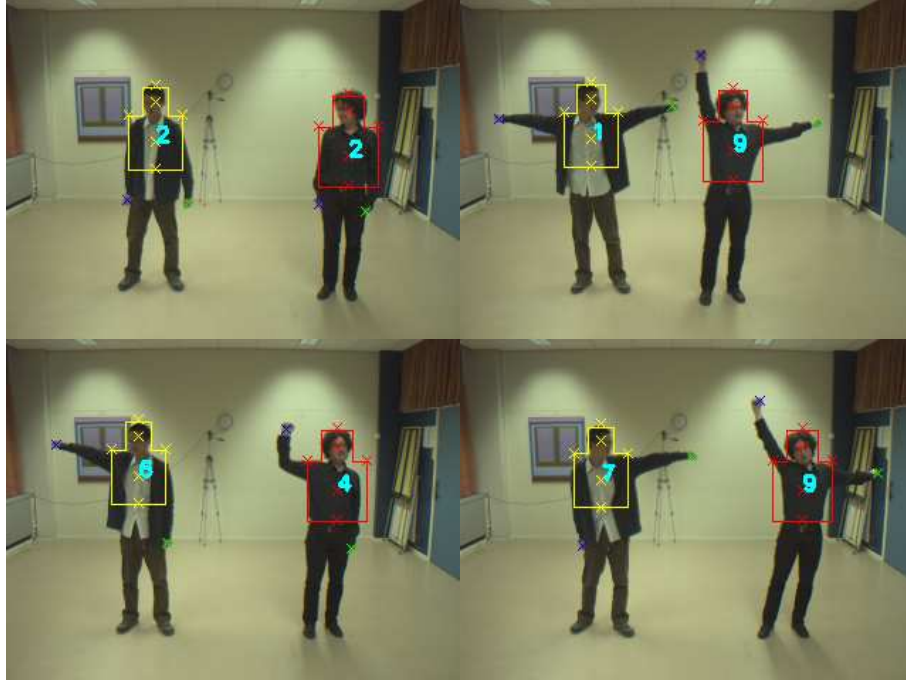
In order to build a classifier, we manually labeled the frames containing the nine poses into nine classes. For each pose, the samples are selected from different persons. Our experimental data set contains 1515 samples of 9 pose types (classes) and 6 features. On average, each pose class is represented by 170 samples. The performances of several statistical classifiers with different complexities are compared. Specifically, we evaluated the nearest mean classifier (NMC), the linear classifier (LDC) and the quadratic classifier (QDC) assuming normal densities and the non-parametric Parzen classifier [8]. We observe that the simplest method (NMC) provides comparable performance to more complex classifiers which need an extra dimensionality reduction step to avoid the curse of dimensionality. We conclude that the extracted features are informative and do not require use of more complex classifiers. Therefore, we chose a simple classifier, 10-nearest neighbourhood classifier (NNC), for the pose classification part of our system. It is easy to implement and also benefits from the computation point of view.



**Fig.4.** Predefined key poses. From top row to bottom row and left to right, these nine poses are labeled as pose1 to pose9.

### 3.5. RESULTS AND DISCUSSION

The proposed system is implemented in C++ with OpenCV libraries [9]. The processing time for each frame is 0.047 seconds, including background subtraction, body parts segmentation and pose recognition. We tested the system with more than 20 users. We also chose the different indoor environment with various settings. Some of the experimental results are shown in Fig.5. An online video demo is also available [10]. People's head and torso are indicated with yellow and red rectangles. Within this 2D model, the location of head top, head center, torso center, torso bottom and both shoulder can be estimated, which are presented by yellow and red cross. People's left and right hand are marked with blue and green cross separately. The output of the pose recognition system is an integer, the number shown in Fig.5. It gives the indication which pose user is performing. This integer and the 1D (horizontal) position of the user will be used as the control command of a spatial game. We evaluated pose classifiers using cross-validation approaches. The average error for all the poses is 6% by using NMC. The result shows that there is a clear separation between pre-defined poses. We also calculate the confusion matrices of the 9-class pose classifier (NMC). The results are promising. Most of the poses can be recognized very well. More details about the pose classification can be found in [11].

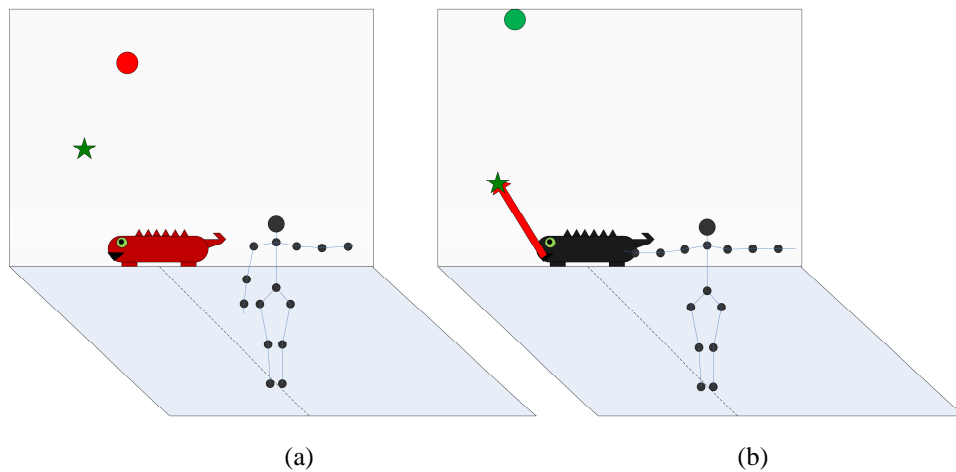


**Fig.5.** Results from people segmentation and pose recognition.

#### 4. SPATIAL GAME APPLICATION

##### 4.1. IMPLEMENTATION

As an application for the pose recognition system we implemented a spatial game, based on the proposal of Phong in [12]. This is a variation of the game Pong [13] in which the player controls a bat to bounce off balls. In Phong the player controls a chameleon which has to bounce off photons, see in Fig.6. The position of the chameleon is determined by the player's position in front of the camera. The photons can have 6 different colors: red, blue, green, yellow, cyan and magenta. The chameleon can change into each of these colors when the player adopts the appropriate pose.



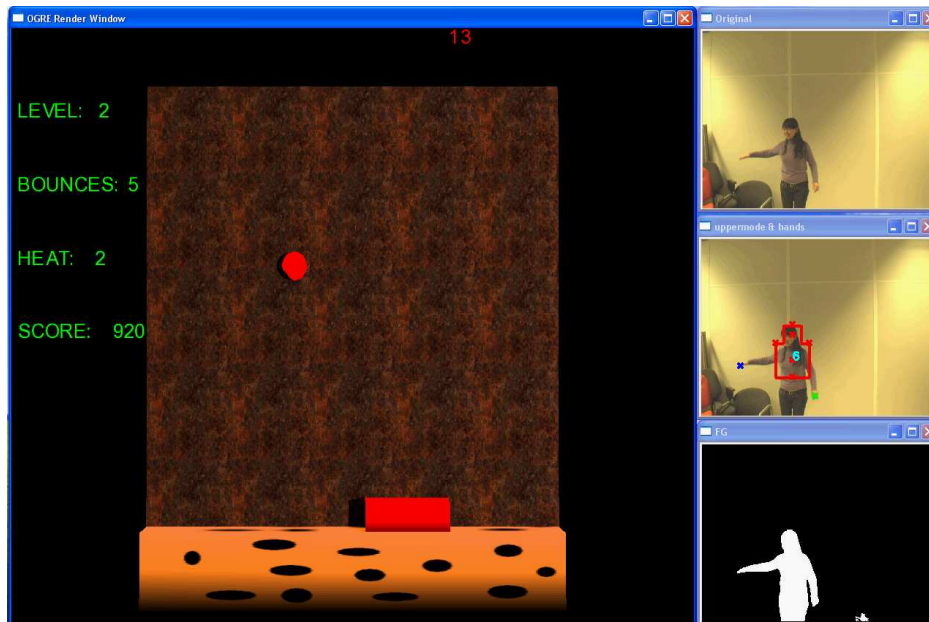
**Fig.6.** (a) Color and position of the chameleon are controlled by pose and position of the player, (b) the tongue of the chameleon is also controlled by a pose to catch flies [12].

When the photon hits the ceiling, it changes color. When the photon is bounced off while the chameleon has the wrong color, the controls flip. Left becomes right and vice versa. When the chameleon has the right color while bouncing the photon off, the score and speed of the photon is increased. When the chameleon misses the photon the ground is heated up. After 4 misses the ground is too hot and the game is over. At random moments a bug flies into the scene which can be eaten by the chameleon when the player adapts to the eating pose. This will increase the score and the ground will cool down.

The game is implemented using the graphics engine Ogre [14]. The input for the game is given by the pose recognition system. The pose recognition system and the spatial game are two separate applications which communicate via sockets [15]. Therefore, it is possible for the two applications to run on different computers and communicate through a network. The pose recognition system sends two types of data to the spatial game in every time step. It sends an integer that represents a pose (1-9) and an integer representing the 1D-location of the player. Whenever the spatial game receives this data, it updates the position of the chameleon according to the position integer and it carries out the action belonging to the pose index that was send. These actions consist of 6 poses for changing the chameleon into the 6 different colors, 1 pose is for eating the bug, 1 pose is for starting the game and 1 pose is to pause the game. Fig.7 gives a screen shot of user playing the game. On the left side is the interface of the game, which shows the level, bounces, heat and score of the player. The three windows on the right side are the results from vision-based analysis. From top to bottom, they are original image, results from body parts segmentation and pose recognition, and foreground binary image.

## 4.2. RESULTS AND DISCUSSION

In our first test runs it became clear that the sensitivity of the pose recognition to detect the change of poses gave a problem for the game player. Whenever the player needs to change from one pose to another there could be a different pose adopted that is “in between” these two poses. When this happens the color of the chameleon in the game is shortly changed into an unwanted color. This problem has been overcome by using a counter whenever a new pose is adopted. The new pose has to be adopted for 4 consecutive time steps until its corresponding action is carried out. This adjustment improved the playability of the game as the user feels having a better control of the chameleon. We did encounter a short delay in handling the players input. The delay is caused by the image processing time. This is mostly noticed with updating the chameleon’s position by the player’s actual location, but the delay is too small to actually cause gameplay problems.



**Fig .7.** Spatial game interface. On the left side is the interface of the game, which shows the level, bounces, heat and score of the player. The three windows on the right side are the results from vision-based analysis. From top to bottom, they are original image, results from body parts segmentation and pose recognition, and foreground binary image.

The implementation of the Phong game showed that the gameplay of the spatial game is interesting. As a next step it is good to reduce the delay to a minimum. After this improvement it is interesting to create a more complex game with an elaborate user interface.

## 5 CONCLUSION

In this paper, we described a real-time computer vision based application. The proposed system is composed of two parts. The first part is video-based people detection, tracking and pose recognition system. It directly uses the captured video frame as input, then gives the 2D position and pose of the people if there are people appearing in the scene. The position and pose information is connected to the second part, a spatial game system, and used as the control command of the game. This pose-driven spatial game is a real time man-machine interaction without obtrusive sensors. It shows the possibility of a new way of interactions in novel computer games and entertainment. The combination of computer vision research and a practical application is quite useful. It allows us to directly test if the proposed algorithm satisfies certain requirements, in a specific application environment. Future work will include improving the robustness of the system (e.g. better skin color detection, more robust feature detection) and developing multiple-user applications. One of the challenges will be to solve the occlusion problem if users are allowed to move freely.

## ACKNOWLEDGMENTS

This research has been supported by the GATE (Game Research for Training and Entertainment) project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie). The spatial game is based on the proposal of Berend Berendsen.

## REFERENCES

- [1] Moeslund, T. B. Hilton, A. and Kruger, V. (2006). A survey of advances in vision-based human motion capture and analysis. In *Computer Vision and Image Understanding*, vol. 104, pages 90-126, 2006.
- [2] Wren, C. Azarbayejani, A. Darrell, T. and Pentland, P. (1997). Pfnder: real-time tracking of the human body. In *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol.19, no.7, pages 780-785, 1997.
- [3] Gupta, A. Mittal, A. Davis, L. S. (2008). Constraint integration for efficient multiview pose estimation with self-occlusion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 493-506, 2008.
- [4] Rogez, G. Rihan, J. Ramalingam S. and etc, (2008). Randomized trees for human pose detection. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, June 2008.
- [5] Stauffer, C. and Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. II, pages 246-252, 1999.
- [6] Micilotta, A. (2005). Detection and tracking of humans for visual interaction. In *PhD. Dissertation, School of Electronics and Physical Sciences, University of Surrey*, 2005.
- [7] Porikli, F.M. and Tuzel, O. (2003). Human body tracking by adaptive background models and mean-shift analysis. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2003.
- [8] PRTools toolbox <http://prtools.org> and PRSD Studio <http://prsdstudio.com> software packages.
- [9] <http://opencv.willowgarage.com/wiki/>
- [10] [http://prsysdesign.net/index.php/html/blog\\_comments/embedding\\_classifi](http://prsysdesign.net/index.php/html/blog_comments/embedding_classifi)
- [11] Huo, F. Hendriks, E.A. Paclik, P. and Oomes, A.H.J. (2009). Markerless human motion capture and pose recognition. In *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, 2009.

- [12] 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.
- [13] <http://www.pong-story.com/rhbaer.htm>
- [14] <http://www.ogre3d.org/>
- [15] [http://en.wikipedia.org/wiki/Internet\\_socket](http://en.wikipedia.org/wiki/Internet_socket)