

# Combining RGB and ToF Cameras for Real-time 3D Hand Gesture Interaction

Michael Van den Bergh<sup>1</sup>  
<sup>1</sup>ETH Zurich  
Zurich, Switzerland  
vamichae@vision.ee.ethz.ch

Luc Van Gool<sup>1,2</sup>  
<sup>2</sup>KU Leuven  
Leuven, Belgium  
vangool@esat.kuleuven.be

## Abstract

*Time-of-Flight (ToF) and other IR-based cameras that register depth are becoming more and more affordable in consumer electronics. This paper aims to improve a real-time hand gesture interaction system by augmenting it with a ToF camera. First, the ToF camera and the RGB camera are calibrated, and a mapping is made from the depth data to the RGB image. Then, a novel hand detection algorithm is introduced based on depth and color. This not only improves detection rates, but also allows for the hand to overlap with the face, or with hands from other persons in the background. The hand detection algorithm is evaluated in these settings, and compared to previous algorithms. Furthermore, the depth information allows us to track the position of the hand in 3D, allowing for more interesting modes of interaction. Finally, the hand gesture recognition algorithm is applied to the depth data as well, and compared to the recognition based on the RGB images. The result is a real-time hand gesture interaction system that allows for complex 3D gestures and is not disturbed by objects or persons in the background.*

## 1. Introduction

In [1], Van den Bergh *et al.* introduce a real-time hand gesture interaction system based on adaptive skin color segmentation and Haarlets. This system poses a number of limitations that we hope to lift using depth information obtained from a ToF camera. The system is often disturbed by skin colored objects in the background (wooden floors, cardboard boxes). Furthermore, the hands are not allowed to overlap with the face or with people in the background. Attempts were made to detect the hands based on shape and contrasts rather than just color, by training Haar cascades. However, due to the high variation in appearance of hands, this was deemed infeasible.

Another way to focus on objects in the foreground is using depth. Given a correct mapping from the depth data to the RGB image, everything in the background can be discarded with a threshold on the depth values. The ToF

camera used in this paper is a SwissRanger SR4000. Though this is an industrial camera, similar low-cost IR-based depth cameras are finding their way to the consumer market. The resolution of the ToF camera is rather low (176×144 pixels), which is why it is interesting to pair it up with a higher resolution RGB camera. In this paper an AVT Marlin color camera is used (640×480 pixels). The low resolution of the ToF camera is enough to steer the segmentation, while the higher resolution RGB camera allows for accurate hand detection.

The system is organized as in Figure 1. Both cameras grab frames simultaneously. Both images are undistorted and the ToF image is projected onto the RGB image coordinates. The face is detected and the distance from the face to the camera is measured. Based on this distance, a threshold is applied to the depth image to discard background objects. The remaining pixels, together with skin color detection, are used to detect the hands. Finally, gesture recognition is run on the detected hands.

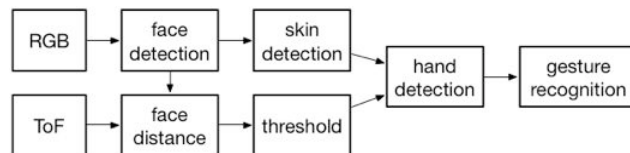


Figure 1: System overview with the RGB and ToF camera.

## 2. Background

The computer vision task in this paper can be divided into two separate problems: (1) detecting the hands and (2) recognizing the hand posture/gesture. Detecting hands is not an easy task. This is illustrated by the fact that there is an abundance of face, mouth, eye detectors available (i.e. in *OpenCV*), but no reliable hand detectors.

One might try to detect hands based on appearance, similar to detecting faces [2]. A straightforward approach is to train Haar cascades to detect hands [3,4]. We evaluated such an approach experimentally, and could not get satisfactory results. This can be explained by the fact that the appearance of hands can change almost indefinitely. Given the 23 degrees of freedom of the hand joints, variation in lighting, shadows and changes in background, the resulting image captured by a camera varies significantly. Faces may vary, but they will always

have eyes above the nose, and a mouth underneath. Except for unusual lighting conditions, the contrasts in the face will also remain constant.

Van den Bergh *et al.* [1] detect hands based on skin color. This approach works well in constrained conditions: no overlap with the face and no skin colored objects in the background. However, the approach copes well with red or brown objects, and with clothing similar to skin color, as the skin color model is adapted to the user on the fly, based on color information taken from the face. However, issues with overlap and persons in the background remain.

Therefore, it is suggested to look into depth, to help segmenting the hand in the foreground from objects in the background. First, we considered stereo. Some experiments were made using real-time stereo vision based on the work by Felzenszwalb *et al.* [5]. Unfortunately, the real-time algorithms don't produce results that are reliable enough for this application. Considering the real-time nature of our application, a better solution is to use a depth camera (*e.g.* ToF). A number of approaches have been published to calibrate multiple color cameras with distance data [6, 7, 8]. The task in this paper is different in the fact that it is a stereo calibration problem: it is limited to two cameras, of which only one RGB camera.

### 3. Calibration

The goal is to map the depth data onto the RGB image in order to help segment foreground objects (hands) from background objects. To maximize the useful depth data, the cameras should be placed as close as possible to each other. In this case, the RGB camera is mounted on top of the ToF camera as shown in Figure 2.

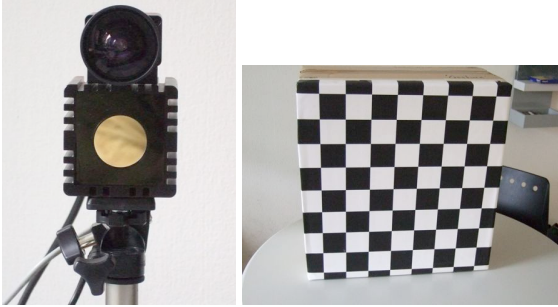


Figure 2: RGB and ToF camera mounted together (left), and calibration object (right).

In this paper we use the Matlab camera calibration toolbox [9] to calibrate one RGB camera and one ToF camera. This can be reduced to a stereo camera calibration problem, except that instead of trying to extract a 3D coordinate from a set of 2D coordinates, we start from a 3D coordinate from the ToF camera, and want to know its corresponding 2D coordinate in the RGB camera image. The calibration is achieved using a large checkerboard calibration object as shown in Figure 2. The ToF camera is able to pick up the intensities of the checkerboard,

however the white is much stronger than the black. Therefore a rectangular erosion filter is passed over the ToF images, to make the corners of the checkers meet.

The first calibration step provides us with the internal camera parameters for each camera: the focal length  $f_c$ , the principal point  $c_c$ , the skew  $\alpha_c$ , and the distortion coefficients  $k_c$ . These allow us to undo the distortion in the input images (using the undistort algorithm in *OpenCV*).

The stereo calibration step provides us with the external camera parameters: a rotation vector  $R$ , and a translation vector  $T$ . The position of the ToF camera chosen as the original of the coordinate system, and these parameters reflect the rotation and translation from the ToF camera to the RGB camera.

For the ToF camera, the pinhole camera model can be written as,

$$s \cdot \begin{bmatrix} u_{ToF} \\ v_{ToF} \\ 1 \end{bmatrix} = \begin{bmatrix} f_{x,ToF} & 0 & c_{x,ToF} \\ 0 & f_{y,ToF} & c_{y,ToF} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad (1)$$

where  $u_{ToF}$  and  $v_{ToF}$  are the coordinates of a pixel in the image captured from the ToF camera,  $s$  is a scale factor, and  $X$ ,  $Y$  and  $Z$  are the coordinates of the corresponding 3D point.

As the ToF camera provides us with the distance of each point to the camera, the scale factor  $s$  can be resolved as this depth value, and the 3D coordinates can be computed,

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = z_{ToF} \cdot \begin{bmatrix} f_{x,ToF}^{-1} & 0 & c_{x,ToF} \cdot f_{x,ToF}^{-1} \\ 0 & f_{y,ToF}^{-1} & c_{y,ToF} \cdot f_{y,ToF}^{-1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{ToF} \\ v_{ToF} \\ 1 \end{bmatrix}, \quad (2)$$

where  $z_{ToF}$  is the depth value for coordinate  $(u_{ToF}, v_{ToF})$ . The  $X$ ,  $Y$  and  $Z$  values can be projected to the coordinate system of the RGB camera,

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = z_{ToF} \cdot R \cdot \begin{bmatrix} f_{x,ToF}^{-1} & 0 & c_{x,ToF} \cdot f_{x,ToF}^{-1} \\ 0 & f_{y,ToF}^{-1} & c_{y,ToF} \cdot f_{y,ToF}^{-1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{ToF} \\ v_{ToF} \\ 1 \end{bmatrix} + T \quad (3)$$

where  $X'$ ,  $Y'$  and  $Z'$  are the 3D coordinates with respect to the RGB camera.

The pinhole model for the RGB camera can be written as,

$$s' \cdot \begin{bmatrix} u_{RGB} \\ v_{RGB} \\ 1 \end{bmatrix} = \begin{bmatrix} f_{x,RGB} & 0 & c_{x,RGB} \\ 0 & f_{y,RGB} & c_{y,RGB} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}, \quad (4)$$

where  $u_{RGB}$  and  $v_{RGB}$  are the coordinates of the pixel in the RGB image, and  $s'$  is a scale factor. Given that  $Z'$  is not zero, the coordinates in the RGB image can thus be computed as,

$$\begin{cases} u_{RGB} = f_{x,RGB} \cdot \frac{X'}{Z'} + c_{x,RGB} \\ v_{RGB} = f_{y,RGB} \cdot \frac{Y'}{Z'} + c_{y,RGB} \end{cases}, \quad (5)$$

which allows us to project any pixel from the depth image to the correct coordinate on the RGB image. Some examples of this projection of the ToF depth image is shown in Figure 3.



Figure 3: Examples of ToF depth images (left), undistorted and projected to the RGB image coordinates (middle), and overlaid onto the original RGB images (right).

#### 4. Hand Detection

Robust hand detection is the most difficult problem in building a hand gesture-based interaction system. There are several cues that can be used: appearance, shape, color, depth, and context. In problems like face detection, the appearance is a very good indicator. The eyes, nose and mouth always appear in the same configuration, with similar proportions, and similar contrasts. This is why appearance-based methods such as Haar cascades lend themselves perfectly to face detection [2]. Unfortunately, this is not the case for hand detection; due to the high number of degrees of freedom, and the resulting shapes and shadows, the hand appearance can change almost indefinitely. The resulting appearance-based detector becomes too general; many parts of the background show similar contrasts a possible hand.

A very important cue in hand detection is color: hands are skin colored objects. However, the perceived color of the skin varies significantly depending on the skin color of the user, and the lighting of the scene. If the skin color model is too broad, it will also detect similar objects such as red clothing or brown floors. If the model is too specific, some skin tones might go undetected, or, under slightly imperfect lighting conditions, the system might break down completely. To overcome this problem we combine a broad, pre-trained skin color model with an adaptive model. The adaptive model is updated on the fly with color information taken from the face, which can be found using a face detector.

This adaptive skin color model helps coping with red and brown clothing and objects in the background, but a number of problems remain. If the hand occludes the face, the algorithm cannot tell where the hand stops and where the face begins. If there are other persons in the background, their hands and faces might throw of the detection, as shown in Figure 4. To overcome this problem, we have chosen to use depth information, measured in real-time using a ToF camera.



Figure 5: Examples of where the skin color-based hand detection cannot distinguish the hand from the overlapping face or hand in the background.

#### 4.1. Skin Color Segmentation and Detection

The skin color segmentation is based on two models. The first is a broad, pre-trained Gaussian mixture model (GMM), which can be trained offline with much more training data. It detects a broad range of skin colors under broad lighting conditions. However, it is easily confused with other colors, such as red sweaters or brown floors.

A second histogram-based model is added, which is trained online while the system is running. It is adapted in real-time to changes in illumination and to the person using the system, by sampling color values from the face region. The model is very specific, but as it is trained with very little training data, it is also very noisy.

The hybrid approach is achieved by multiplying the GMM-based skin color probability with the histogram-based skin color probability. This pseudo-probability is then compared to a threshold,

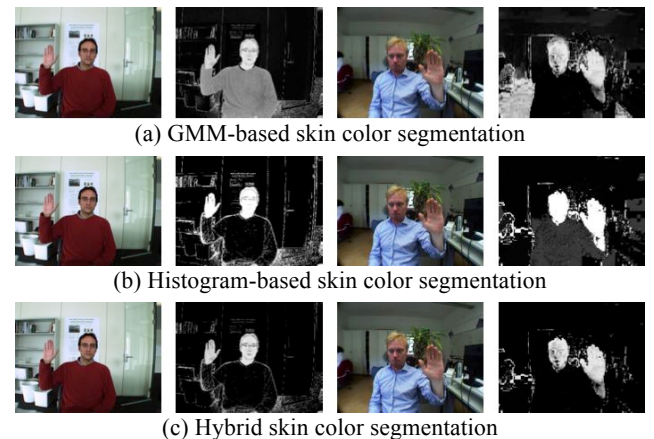


Figure 4: (a) The GMM-based segmentation is too broad and detects the red sweater and brown floor. (b) The histogram-based segmentation is noisy. (c) The hybrid segmentation combines both, and eliminates most of the sweater, the floor and the noise.

$$P_{GMM}(skin|c) \cdot \frac{P_{hist}(c|skin)}{P_{hist}(c|nonskin)} > T, \quad (6)$$

where  $\mathbf{c}$  is the color,  $P_{GMM}$  is the Gaussian probability that  $\mathbf{c}$  is skin color, and  $P_{hist}(c|skin)$  and  $P_{hist}(c|nonskin)$  are the histogram-based probabilities that  $\mathbf{c}$  belongs to the *skin* and *nonskin* classes respectively. The resulting skin color segmentation is shown in Figure 5. The noise from the histogram-based segmentation is filtered out by the GMM-based segmentation, while the adaptive model eliminates the false positives of the GMM-based segmentation.

## 4.2. Depth-based Segmentation and Detection

To further improve the hand detection, depth data is used from the ToF camera. As described in Section 3, the depth image is undistorted and projected onto the RGB image. This provides us with an accurate depth value for each pixel in the RGB image.

Using a face detector (*i.e.* *OpenCV*) on the RGB image, the position  $(x, y)$  and size  $(w, h)$  of the face can be determined. The distance from the face to the camera can be estimated as the average depth values of the face,

$$d_{face} = \frac{1}{wh} \sum_{i=x}^{x+w} \sum_{j=y}^{y+h} I'_{ToF}(i, j), \quad (7)$$

where  $I'_{ToF}$  is the projected depth image. If the face is occluded, the previous position and distance is used. All the objects in front of the face can then be found by applying a threshold,

$$I'_{ToF}(i, j) > d_{face} + t_s, \quad (8)$$

where  $t_s$  is a static parameter. It defines how far the user has to extend the hand towards the camera for it to be accepted as an interaction. An example of this threshold is shown in Figure 6. The search region is limited to the remaining pixels, and the hand is detected based on skin color, as illustrated in Figure 6. If the user is wearing short sleeves, the arm is removed from the hand by the assumption that the arm is underneath the hand. Once the hand is detected, the distance from the hand to the camera can be estimated as well, similar to Eq. (7).

## 4.3. Experiments

To evaluate the hand detection, three test sequences were recorded: a normal test sequence where the hand is next to the face, a sequence where the hand overlaps with the face, and a sequence where a second person tries to disturb the detection behind the user. A total of 362 frames were recorded and annotated for this experiment. A hand is detected correctly if the distance to the ground truth is less than half the width of the hand (center to center), and the difference in scale is less than 25%.

In the first sequence, the hand is always next to the face, with no overlap, and no persons in the background. There are a brown envelope and a brown leather bag in the picture, which might disturb the detection. The results of the hand detection are shown in Figures 6 and 7.



Figure 7: RGB image (top left), depth image (bottom left), skin color probability (top middle), depth image after threshold (bottom middle), skin color probability limited to foreground pixels (top right), and segmented hand (bottom right).



Figure 8: Detection rate on the normal sequence. The color-based detection achieves 92.0% correct detection, while the depth-based detection achieves 99.2% correct detection.

In the second sequence, the hand overlaps with the face in 30% of the frames. Figure 8 and 9 show how the depth-based detection copes with the overlap, while the color-based detection fails on the frames where there is overlap.



Figure 9: RGB image (top left), depth image (bottom left), skin color probability (top middle), depth image after threshold (bottom middle), skin color probability limited to foreground pixels (top right), and segmented hand (bottom right).

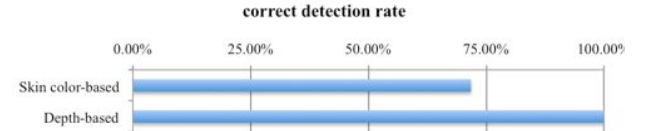


Figure 6: Detection rate on the sequence with face overlap. The color-based detection achieves 71.8% correct detection, while the depth-based detection achieves 100.0% correct detection.

In the third sequence, a second person is in the background trying to disturb the hand detection. The depth threshold allows segmenting the hand in the foreground from the person in the background, as illustrated in Figures 10 and 11. The skin color probability in Figure 11 is accurate, but not enough to detect the hand, as there are several other skin colored regions. However, limiting the detection to the pixels that are close enough to the camera, the hand is segmented correctly.



Figure 10: RGB image (top left), depth image (bottom left), skin color probability (top middle), depth image after threshold (bottom middle), skin color probability limited to foreground pixels (top right), and segmented hand (bottom right).

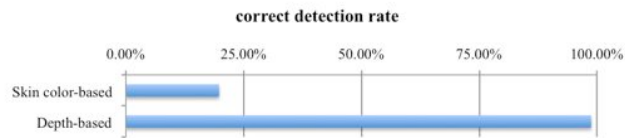


Figure 11: Detection rate on the sequence with a person in the background. The color-based detection achieves 19.8% correct detection, while the depth-based detection achieves 98.8% correct detection.

## 5. Hand Gesture Recognition

The hand gesture recognition system is based on the recognition algorithm described in [1]. The classifier relies on a dimensionality reduction based on Average Neighborhood Margin Maximization (ANMM) [10], which projects the input samples to a lower dimensional space, as shown in Figure 12. The ANMM transformation is approximated using Haarlets to improve the speed of the system. Using nearest neighbors search, the coefficients are then matched to hand gestures stored in a database.

To classify an input hand image, the Haarlet coefficients are computed on the input. This results in 50-100 coefficients depending on the classifier, which are stored in a vector. Multiplying this vector with the pre-trained approximation matrix  $C$  results in a small number of approximated ANMM-coefficients (2-5 depending on the number of trained hand poses). These coefficients are a close approximation of the coefficients that would result from a pure ANMM projection, maximally separating the hand poses and allowing for classification by finding the nearest match in the database of stored hand gestures.

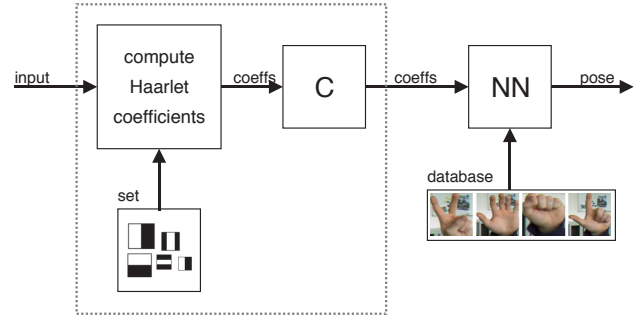


Figure 12: Structure of the classifier illustrating the ANMM transformation (dotted box), approximated using Haarlets. The Haarlet coefficients are computed on the input sample. The approximated ANMM coefficients are computed as a linear combination  $C$  of the Haarlet coefficients.

The system described in [1] relies only on RGB input data. In this paper we also explore the use of depth data from the ToF camera for classification. Depth adds another level of information that might be hard to detect in 2D (*i.e.* user pointing straight at the camera); on the other hand the resolution of the depth data is very low.

### 5.1. Based on RGB Image

The input of the classifier consists of  $48 \times 24$  input vectors. These vectors consist of the concatenation of the intensity values of two input streams: (1) the cropped grayscale hand image; and (2) the skin color segmentation. As described in Section 4, the skin color segmentation is corrected based on the depth image to avoid interference from the face or persons in the background. Examples of this concatenation are shown in Figure 13.

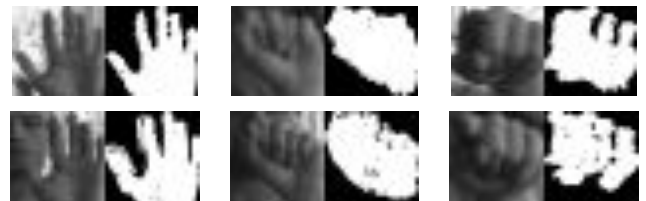


Figure 13: Example classifier inputs:  $48 \times 24$  vectors containing a concatenation of the grayscale hand image and the skin color segmentation.

This input vector was chosen after experiments with different input types. The combination of the grayscale image and the skin color probability allows the classifier to rely on the appearance and shape of the hand: the cropped image contains the appearance, regardless of the segmentation quality, while the segmented image contains the shape, regardless of cluttered backgrounds.

The required hand resolution for good classification is rather low ( $24 \times 24$ ). This translates in the fact that the user can be far away from the camera (3-4 meters) without compromising classification performance.

## 5.2. Based on Depth Image

Another option is to classify the hand gesture based on the depth data. The classifier input is then the cropped depth image of the hand ( $24 \times 24$ ). The hands are cropped based on the hand detection described in Section 4. The same threshold as in Eq. (7) is applied, in the sense that pixels that are too far away are set to 0 (black). The image is also normalized; the average hand pixel distance is set to 0.5. The image values lie between 0 and 1. Examples of these normalized depth images are shown in Figure 14.

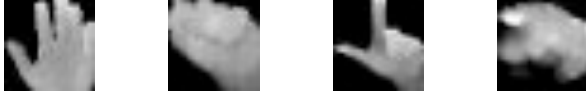


Figure 14: Example classifier inputs:  $24 \times 24$  vectors containing a cropped and normalized depth image.

The depth images benefit from the fact that a lot of information about the hand posture is contained within the image. However, the resolution of the ToF is very low, which requires the hand to be close enough to the camera, and limits the distance from the user to the camera to 1 m.

## 5.3. Combined Image (RGB and depth)

Both the RGB data and the depth data can be combined into a single vector, as shown in Figure 15. A classifier with this input type combines the benefits of both approaches, and should (given sufficient training data) perform as well or better than the individual approaches.

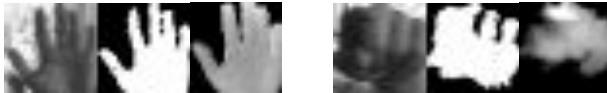


Figure 15: Example classifier inputs:  $72 \times 24$  vectors containing a concatenation of the grayscale hand image, the skin color segmentation, and a normalized depth image.

## 5.4. Experiments

From the three sequences described in Section 4.3, a total of 350 hand samples are extracted, of which half are used for training, and half for validation. The 350 sample images contain a total of 6 different key postures, as shown in Figure 16: open hand, fist, pointing up, L-shape, pointing at the camera, and thumb up.



Figure 16: The 6 key hand postures used in this experiment.

The orientation of the hand is assumed to be constant, but of course some level of rotation is unavoidable (especially around the vertical axis). The samples also include overlap with the face and with the person in the background.

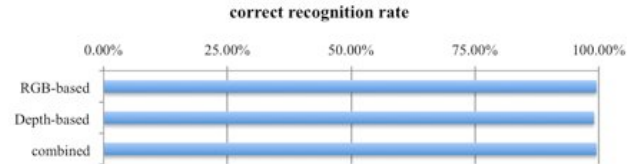


Figure 17: Results of hand gesture recognition experiment: RGB-based recognition achieves 99.54% correct recognition, depth-based recognition achieves 99.07% correct recognition. Combining both achieves 99.54% correct recognition.

The results of the hand gesture recognition experiment are shown in Figure 17. The experiment shows that both the RGB as the depth-based approaches work equally well. The first works slightly better, possibly because of the higher resolution. It seems that in case of a limited set of key gestures, the depth information is not necessary to improve the recognition. It is possible that in another application, with a high number of gestures, the depth information is helpful to disambiguate complex gestures.

## 6. Interaction System

The hand gesture interaction is composed of the key hand postures that were defined in Section 5. The system recognizes the gestures and movements of both hands to enable the manipulation of an object/model. An overview of the gestures and motions used in this paper is shown in Figure 18. The gestures consist of: selecting, panning, zooming and rotating. The movements of the hands are not limited to 2D, but allow for full 3D motion (up, down, left, right, forward, backward). In a system with an RGB camera only, the distance of the hands to the camera is estimated based on the size of the hands. Adding a ToF camera allows for an exact distance calculation.

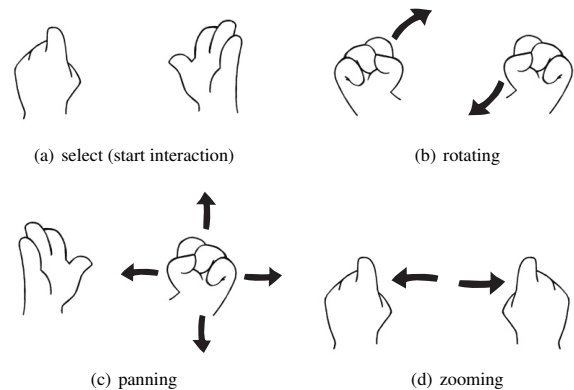


Figure 18: The hand gestures used for the manipulation of the 3D model on the screen.

The cameras are mounted on top of a large screen. The interaction demo system has been implemented as an extension of an open-source 3D model viewer, the *GLC Player*. Pressing a button in the toolbar activates the hand interaction mode, after which the user can start gesturing to navigate through the model. Pressing the button again deactivates the hand interaction model and returns to the standard mouse-keyboard interaction mode.

We conducted experiments in the Value Lab with multiple 3D models, and in particular with a large model created as part of a student urban planning project. This model represents an area of about 0.6 km<sup>2</sup> and is constituted of about 4000 objects (buildings, street elements, trees) with a total of about 500,000 polygons. Despite this size, our system achieved frame rates of about 30 fps. Examples of the user zooming, panning and rotating through the 3D model are shown in Figure 17.

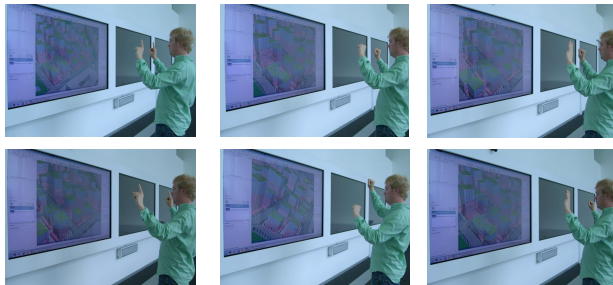


Figure 19: Examples of the interaction demo.

## 7. Conclusion

This paper introduces a hand gesture interaction system based on an RGB camera and a ToF camera. The system is motivated by the fact that IR-based depth sensors are finding their way to the consumer market, and applications such as this one become more feasible. A number of contributions are made. First, calibration of both cameras and real-time projection of the depth data onto the RGB image is implemented. An improved hand detection algorithm is introduced based on adaptive skin color detection and depth. This approach significantly improves the interaction system, as the hands can overlap with the face, and other persons can be in the background. These improvements are also shown in several experiments.

A Haarlet-based hand gesture recognition algorithm is implemented for two approaches: based on the RGB data, and based on the depth data. Both are evaluated against each other, and also a combined approach is proposed. Even though the test data contains gestures that are hard to detect in 2D (*i.e.* pointing at the camera), the experiments show the RGB images contain enough information to detect the gesture correctly. However, the experiments also show the possibility to classify based only on depth information only, which means systems with only a ToF camera are also feasible. In this paper, the depth

information is represented as a 2D depth map, and classification is based on 2D Haarlets. Considering that gesture recognition is possible on this information, in future work it might be interesting to generate 3D hulls from the depth data and classify based on 3D Haarlets. This could enable a level of orientation normalization.

Finally, the algorithms presented in this paper are illustrated with an example application, where the user manipulates a 3D model with hand gestures in real-time, and without the use of special markers. Gesture motions in 3D are made possible based on the depth position of the hand. In future work one could also incorporate the 3D position of the face, to estimate viewing angle, or combined with the hand position: pointing direction. The concepts presented in this paper also open possibilities towards applications with multiple users: detecting multiple faces at different distances, and estimating which hand belongs to which face.

## Acknowledgements

This work is carried out in the context of the Seventh Framework Programme of the European Commission: EU Project FP7-ICT-24314 *Interactive Urban Robot (IURO)*.

## References

- [1] M. Van den Bergh, F. Bosché, E. Koller-Meier, and L. Van Gool. Haarlet-based hand gesture recognition for 3D interaction. *Workshop on Applications of Computer Vision (WACV)*, December 2009.
- [2] P. Viola and M. J. Jones. Robust real-time object detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 747, 2001.
- [3] M. Kölsch and M. Turk. Robust hand detection. *6th IEEE International Conf. on Automatic Face and Gesture Recognition*, pages 614–619, May 2004.
- [4] E.-J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. *6th IEEE International Conf. on Automatic Face and Gesture Recognition*, pages 889–894, May 2004.
- [5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, vol. 70(1), pages 41–54, October 2006.
- [6] R. Reulke. Combination of distance data with high resolution images. *Image Engineering and Vision Metrology*, pages 25–27, September, 2006.
- [7] L. Guan and M. Pollefeys. A unified approach to calibrate a network of camcorders & ToF cameras, *IEEE workshop on Multi-camera and Multi-model Sensor Fusion Algorithms and Applications*, 2008.
- [8] Y.M. Kim, D. Chan, C. Theobalt, S. Thrun, Design and calibration of a multi-view ToF sensor fusion system. *In CVPR Workshop on Time-of-flight Computer Vision* 2008.
- [9] Matlab Camera Calibration Toolbox. [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)
- [10] F. Wang and C. Zhang. Feature extraction by maximizing the average neighborhood margin. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007