

# Real-time 3D Hand Gesture Interaction with a Robot for Understanding Directions from Humans

Michael Van den Bergh, Daniel Carton, Roderick De Nijs, Nikos Mitsou, Christian Landsiedel, Kolja Kuehnlenz, Dirk Wollherr, Luc Van Gool and Martin Buss

**Abstract**—This paper implements a real-time hand gesture recognition algorithm based on the inexpensive Kinect sensor. The use of a depth sensor allows for complex 3D gestures where the system is robust to disturbing objects or persons in the background. A Haarlet-based hand gesture recognition system is implemented to detect hand gestures in any orientation, and more in particular pointing gestures while extracting the 3D pointing direction. The system is integrated on an interactive robot (based on ROS), allowing for real-time hand gesture interaction with the robot. Pointing gestures are translated into goals for the robot, telling him where to go. A demo scenario is presented where the robot looks for persons to interact with, asks for directions, and then detects a 3D pointing direction. The robot then explores his vicinity in the given direction and looks for a new person to interact with.

## I. INTRODUCTION

As robots move away from industrial settings and closer into our lives, the question arises of how to interact with these robots. A large part of natural interaction happens through hand gestures [13]. Especially if a robot is designed to help humans with everyday tasks, an important ability the robot will have to learn is to understand pointing gestures: pointing at objects or locations of interest, pointing to help explain the direction in which the robot should move, or even helping a robot to find a moving target which humans might have seen (in which case GPS and maps do not help). This requires a system that allows the robot to detect an interacting human and to understand the hand gestures.

In [1], Bauer *et al.* present an interaction system where a robot asks a human for directions, and then interprets the given directions. This system includes a vision component where the full body pose is inferred from a stereo image pair. However, this fitting process is rather slow and does not work in real time.

Waldherr *et al.* [2] present a template-based hand gesture recognition system for a mobile robot, with gestures for the robot to stop or follow, and rudimentary pointing. As the gesture system is based on a color-based tracker, several limitations are imposed on the types of clothing and contrast with the background.

In [3], Van den Bergh *et al.* introduce a real-time hand gesture interaction system based on a Time-of-Flight (ToF)

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)*.

M. Van den Bergh and L. Van Gool are with the Computer Vision Lab, ETH Zurich, Switzerland. [vamichae@vision.ee.ethz.ch](mailto:vamichae@vision.ee.ethz.ch)

D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr and M. Buss are with the Institute of Automatic Control Engineering, TU Munich, Germany. [dc@tum.de](mailto:dc@tum.de)

camera. Both the depth images from the ToF camera and the color image from the RGB camera are used for a Haarlet-based hand gesture classification. Other, similar ToF-based systems are also described in the literature: Breuer *et al.* [5] fit an articulated hand model to the depth data, though not in real time; Kollorz *et al.* [6] use the depth information to segment the hand and then classify the 2D silhouette of the hand; and Soutschek *et al.* [7] use a similar approach in a medical application.

The use of the ToF camera allows for a recognition system robust to all colors of clothing, to background noise and to other people standing around. However, ToF cameras are expensive and suffer from a very low resolution and a narrow angle of view, which is not optimal for pointing-style interaction at reasonable distances. Therefore in this paper the choice was made for a Kinect sensor, which provides a higher resolution at a lower cost, and combines depth and RGB cameras in one compact sensor.

This paper presents a real-time hand gesture interaction system implemented on an interactive robot. A depth sensor (combined with color camera) is used to detect hand gestures, which are translated into interaction commands and 3D pointing directions. In this paper the hand gesture recognition algorithms from [3] are implemented on the Kinect, and

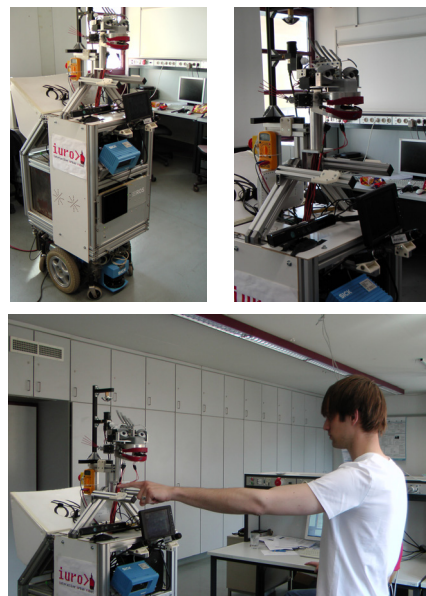


Fig. 1. Prototype robot with Kinect sensor and a typical interaction scene.

adapted to allow for the gestures to be detected in any orientation of the hand. This is useful for detecting gestures while pointing. The system is also extended to extract the 3D pointing direction, which the robot can use to define its goal on a map. The interaction system is integrated on a prototype robot using ROS, shown in Figure 1. A demo scenario is presented where the robot follows pointing directions from humans, and then looks for a new person to initiate further interaction and receive further directions as to where it should go. Our key contributions are:

- 1) An orientation-invariant pointing direction detection and gesture recognition working in real time, making human-robot-interaction more natural;
- 2) An implementation of pointing and gesture detection on the inexpensive Kinect sensor;
- 3) An application of 3D pointing gestures for directed robotic exploration tasks without prior map knowledge.

The remainder of this paper is structured as follows: section II presents the calibration, human and hand detection, section III elaborates the method for gesture recognition, section IV delineates the navigation principle used to explore unknown areas in a given direction, and section V depicts a real-world experiment with the presented system.

## II. INITIAL STEPS

The system presented in this paper aims to build a robot which firstly looks for a person to interact with, then initiates interaction, and uses hand gesture recognition to detect a pointing gesture and 3D pointing direction. This pointing direction is then translated to a goal on the map, after which the robot proceeds to this goal. This section briefly describes the processing steps that are required to initialize the system before hand gestures can be detected.

### A. Calibration

The Kinect sensor comes with a factory calibration between the depth camera and the RGB camera. If not, it would be very difficult to calibrate the Kinect as the depth camera does not register intensity, and thus it cannot 'see' most standard calibration objects. This factory calibration provides us with the necessary parameters to undistort the depth and color images, and map depth pixels to color pixels.

### B. Human Detection

Initially the robot only perceives his environment as an occupancy grid generated from laser data that covers a very constrained area around him. No further knowledge is provided about the area that is to be explored. Therefore the robot has to find an information source he can question for the location of his goal or the direction he should explore. This is solved by a human detection method that incorporates skin, face and additional leg detection. Basically the robot is permanently looking for human beings to appear in his scope, what means that the three methods have to yield successful detections concurrently to appear in a cone with a 60 degree angle of aperture in front of the robot.

A person that appears in front of the robot is at first seen by a SICK LMS 200 laser-range-finder and handled as an obstacle by the navigation planner what forces the robot to stop and provides time for our software to decide if a person has appeared. Since the laser is mounted on the base it only yields a few points ideally in two separate clusters with a semi circle shape corresponding to the legs of a person. The leg detection, which is provided by ROS [15], utilizes the 3D laser scans and splits given point cloud into clusters. From these segments, different features such as curvature and size are extracted and used for classification with a random forest approach [14].

Once a pair of legs is found, an adaptive skin detection [4] operates on RGB images obtained from two wide angle cameras in a narrow stereo setup. The generated result serves as a search space reduction for the subsequent Viola-Jones based face detection [8] within the same images. If the face detection succeeds as well, we assume that there has to be a human in front of the robot that might offer more crucial information about the goal location.

### C. Hand Detection

Once a human is detected, the robot tries to initialize interaction. This requires the detection and tracking of the hand location. 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 [8]. A straightforward approach is to train Haar cascades to detect hands [9], [10]. 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.* [3] detect hands successfully based on skin color and depth. However, the Kinect sensor also comes with a built-in solution that is available in the OpenNI libraries. This built-in hand tracker is initialized by detecting a waving hand as a waving object in the foreground of the depth image. Once detected, the system signals that the user can stop waving, and the hand is tracked. If the hand is lost, the system asks the user to wave again. This system was chosen even though it is not fully automatic, because it is an elegant and robust solution, and allows us to focus on the actual gesture recognition.

Even though other solutions are available, this built-in tracker provides an elegant solution to the task presented in this paper. The robot attempts to initialize the interaction by asking the previously detected human to wave at the robot, which initializes the tracker and indicates that the human is willing to interact. The robot signals that the human can stop

waving, and asks to show where the robot should go, which is where the gesture recognition system can take over.

Note that even though the Kinect can detect a waving hand, it does not come with hand gesture recognition.

### III. HAND GESTURE RECOGNITION

Once the hand location is known, the hand can be normalized for size and orientation, and the hand posture can be detected.

#### A. Orientation Estimation

After obtaining the hand location, the hand needs to be normalized in order to detect its posture. If a human is explaining directions to the robot, it is unlikely that the hand is always in the upright position, so it is important to know the orientation of the hand.

The orientation is found by detecting the position of the wrist with respect to the center of the hand. The wrist is detected in three steps: (1) segmenting the hand based on depth by removing any pixels that are behind the hand; (2) placing a rectangle over the hand/arm; and (3) computing the average point where this rectangle intersects the hand/arm. This process is illustrated in Figure 2. The rectangle size is made relative to the distance between the hand and the camera.

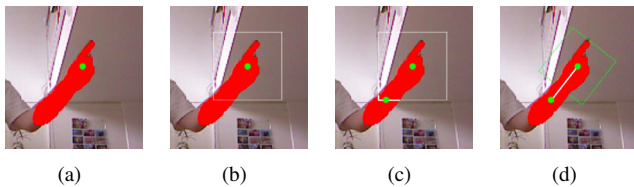


Fig. 2. Estimating the hand orientation: (a) hand center and segmentation; (b) rectangle placed over the hand; (c) intersection with the rectangle and estimated wrist position; (d) estimated hand orientation.

#### B. Hand Posture Recognition

A review of current model-based hand pose estimation methods is presented by Erol *et al.* [11]. Most approaches either use special markers, or are model-based, meaning they rely on articulated 3D body/hand models. In order to be effective they need to have a high number of degrees of freedom, in combination with non-linear anatomical constraints. Consequently, they require time-consuming per-frame optimization and the resulting trackers are too slow for real-time ( $\geq 25\text{Hz}$ ) approaches. They are also very sensitive to fast motions and segmentation errors.

The hand posture recognition system presented in this paper is an example-based approach. Example-based methods benefit from the fact that the set of typically interesting poses is far smaller than the set of anatomically possible ones, which is good for robustness: (1) since the pose is estimated on a frame-by-frame basis, it is not possible to lose track; and (2) not needing an explicit parametric body model is more amenable to real-time implementation. For details on the recognition algorithm we refer to [3]. The classifier relies on a dimensionality reduction based on Average Neighborhood

Margin Maximization (ANMM), which projects the input samples to a lower dimensional space.

The input samples of the classifier are  $72 \times 24$  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 depth image, as shown in Figure 9.



Fig. 3. Examples of input vectors for the classifiers.

These samples are then classified as shown in Figure 4. 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.

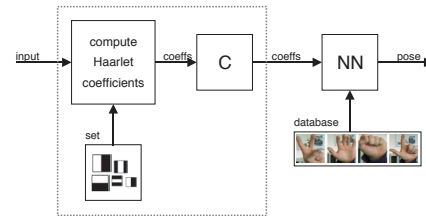


Fig. 4. The ANMM transformation is approximated using Haarlets to improve the speed of the system. Using nearest neighbors search, the coefficients are then matched to hand gestures stored in a database.

In this implementation the classifier is trained with 4 key gestures: open hand (all fingers extended), fist, pointing (index extended) and a second pointing gesture (index and thumb extended). Examples of the gesture classification are shown in Figure 5.



Fig. 5. Examples of the classifier detecting different hand gestures. Green = open hand, red = pointing, blue = fist.

#### C. Pointing Direction

When a pointing gesture is detected, we would like to know the pointing direction. In this system the pointing direction is defined as the vector that connects the wrist with the center of the hand. Other pointing directions can be considered, for example the vector that connects the eyes

with the fingertip. However, for this application, the hand orientation is a better indication of the pointing direction. This vector is then projected onto the horizontal plane, which results in a pointing direction that relates to the 2D map of the surrounding, as illustrated in Figure 6 (green arrow).

#### IV. NAVIGATION

In principle the robot could build a map using SLAM, but to enforce the exploration application the robot only has a 2D occupancy grid of his immediate environment available. This grid map is generated by making use of the SICK LMS 200 laser mounted on the robot's mobile base. The generated obstacle map features inflated obstacles which prohibits the platform from getting too close to objects, yet it can still pass through doors 10cm wider than its footprint.

Utilizing this map a collision-free path can be generated between a goal point and the actual robot position employing the path planners provided by ROS. To be able to explore his vicinity the robot has to set his goals accordingly, meaning that he always aims for the edges of his local map as they correspond to the closest unknown area. The goals are calculated by an openly available ROS package maintained by Robert Bosch LLC<sup>1</sup>. This method searches for the border cells of the grid map which are then clustered. Border-cells belong to the same cluster as long as they are direct neighbors. A split of a border occurs if an obstacle prohibits a direct neighborhood. For each cluster a gradient orientation is calculated that indicates the direction the exploration for the adjacent area should follow.

Thus, in our experiment the robot will try to reach one of these goals and update his destination if one of the following conditions is met: the goal is reached, the goal position is blocked by an obstacle or a human being is found for interaction. The last aspect that has to be looked into is which goal the robot should follow as there will be multiple opportunities given that he only has a constricted local map and no stored map. We address this problem by combining the exploration method with our gesture recognition module.

As it was mentioned above, we receive a 2D vector in the local robot frame pointing into the direction the robot should move. This vector is given in the sensor specific frame which is why it is transformed into the robot's local frame that corresponds to the frame of the occupancy grid. The transformation parameters are kept up to date at any point in time such that the robotic platform always perceives pointing directions the same way. According to this principal direction we can now choose the next exploration goal  $(x_G, y_G)$  from the set of possible goals  $G = [(x_{G1}, y_{G1}), (x_{G2}, y_{G2}), \dots, (x_{Gn}, y_{Gn})]$ . Firstly, we only consider the goals that are located within the half-plane of the robot's local frame the gesture was pointing to. Secondly, we weight the goals by their distance  $d_{Gv}$  to the line given by the vector  $v = (v_x, v_y)$  that originates in the position of the pointing human  $(x_H, y_H)$ . Certainly this distance vector has an intersection point  $\lambda_G \cdot v$  with the

pointing vector line. We can calculate an euclidean distance  $d_{\lambda R}$  between the respective intersection point of each goal  $\lambda_{G1, G2, \dots, Gn} \cdot v$  and the robot  $(0, 0)$  and take it into account in a second weighting step. The following equations apply:

$$\lambda_G = \frac{\begin{pmatrix} x_G - x_H \\ y_G - y_H \end{pmatrix} \cdot v}{|v|^2}, \quad (1)$$

$$d_{Gv} = \left| \begin{pmatrix} x_G - x_H \\ y_G - y_H \end{pmatrix} - \lambda_G \cdot v \right|, \quad (2)$$

$$d_{\lambda R} = \left| \begin{pmatrix} x_H \\ y_H \end{pmatrix} + \lambda_G \cdot v \right|. \quad (3)$$

Accordingly, this means that the closer a goal is to the robot and the pointing direction, the higher its weight will become and the more likely it is to become the next exploration goal. This method favors goals in close proximity rather than the ones where the robot would have to go around walls first.

Figure 6 shows a typical interaction scenario. The first image shows the scene and the second image illustrates how the perceived pointing vector (green arrow) is utilized leading to the next goal (blue arrow) on the obstacle map. The blue, red and green bars at the origin of the planned path (purple line) show the used frames. As one can see the robot only has a local obstacle map (red dots) available which is generated from the base laser scan. The teal points show detected legs in front of the robot and some false positives scattered over the map. The orange blob besides the robot frames indicates a successful face detection.

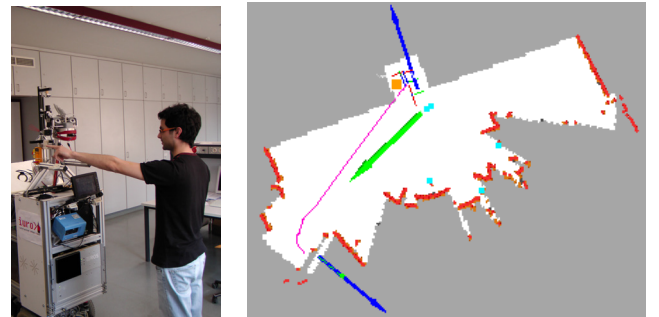


Fig. 6. Person interacting with the robot and example of the path planning. The green arrow indicates the perceived pointing direction, the blue arrows indicate the previous and the new goal, the pink line indicates the planned path and the red dots represent the local obstacle map.

#### V. EXPERIMENT

The platform used in this paper is an extended version of the ACE platform [1], with a new software framework (ROS) and a new head. Figure 7 shows the available hardware. For the experiment we mainly used the base laser for navigational sensing and leg detection and the stereo cameras for face detection. As shown in Figure 1 we also mounted a Kinect sensor to our platform supporting the gesture detection.

The scenario for the experiment consists of two tasks: on the one hand the robot should explore the floor of his laboratory autonomously and on the other hand it must incorporate directions given by human interaction partners.

<sup>1</sup>Copyright (c) 2008, Robert Bosch LLC

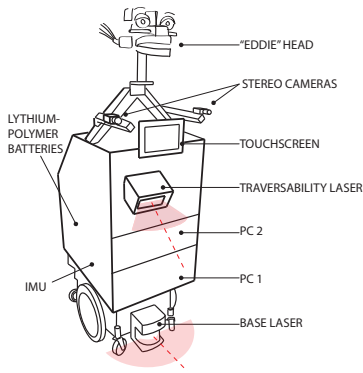


Fig. 7. The prototype robot setup.

The interaction is exclusively based on the above described gesturing, which the robot asks for at the beginning of a human-robot interaction.

To enable the robot to invoke the necessary procedures a state machine is used as a framework with the ability to start and stop methods on demand. Additionally, we implemented modules that manage the activation and deactivation of multiple available services depending on the current needs. This is necessary as the state machine itself can only switch one module at a time. For our application, where leg and face detection need to report a positive result at the same time, a manager module can easily compare the current service responses. In a state machine even concurrency would not be a solution because after a module finished, the state is left independent from the output of the concurrently running module. The state machine framework is shown in Figure 8.

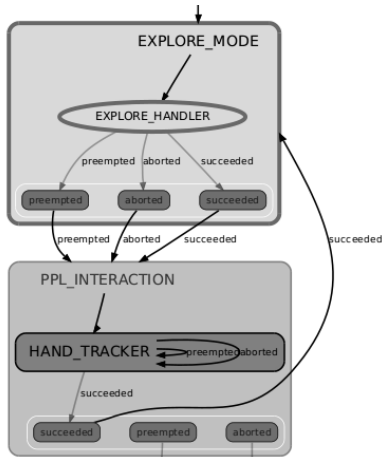


Fig. 8. State Machine framework for action selection.

The robot starts in interaction mode, trying to acquire a first principal direction. On success the given direction is passed to the explore module that starts the exploration mode as well as the leg and face detection services. If the detectors sense a human and report success at the same time, the exploration is stopped by the manager in favor of another interaction. The interaction mode itself consists of multiple states that on one hand let the robot give instructions

using speech synthesis and on the other hand start and stop the gesture detection. Having procedures sleep if the state machine is not in the right state yields the advantage that the power consumption is lowered since the respective sensors are switched off as well.

The result of the experiment is shown in Figure 9 as a concatenation of the local maps available to the robot. We conducted a full run around the lab floor by giving directions to the exploring robot. Gray areas correspond to unknown environment that has to be explored if accessible, and white areas depict known space respectively. There are still gray areas in between some parts of our illustration because of the concatenation of non overlapping local maps. The small red dots show obstacles scanned by the base laser whereas orange points are static obstacles that were observed from multiple points of view and black points are objects that were only detected once by the laser. All the above mentioned information is stored in the occupancy grid which is used by the navigational methods.

Blue arrows in Figure 9 indicate possible exploration goals calculated by the frontier clustering described in section IV. Green arrows depict positions where an interaction with a human occurred, and which direction was given. The image shows occurrences of incorrect leg and face detections as big purple and red blocks. Finally, the pink lines depict the exploration paths planned by the robot during runtime.

For accuracy and precision measurements of the gesture recognition system we refer to [3].

## VI. CONCLUSIONS AND FUTURE WORK

### A. Conclusions

In this paper we presented an integrated and functional system where a robot looks for humans to interact with, asks for directions, understands those directions based on hand gesture recognition, and then translates those directions to goals on a map of the environment which the robot has built.

A number of contributions have been made. A kinect-based hand gesture interaction system has been introduced which supports orientation-invariant gesture recognition and 3D pointing direction. A navigation system was built where the robot builds a map of its known environment and translates directional vectors from the gesture system into navigational goals. The different components have been integrated on a robot, and the functionality of the system has been demonstrated in an interaction and navigation experiment.

The presented gesture interaction system is useful for many applications: a human pointing at objects or locations of interest to the robot, an autonomous robot asking for directions from humans and interpreting those directions, or even a robot in pursuit of a moving target and asking humans if they have seen the target.

### B. Future Work

Thanks to the Kinect, good progress has been made in robust hand gesture interaction. However, the sensor is based on IR and therefore exhibits poor outdoor performance in case of direct sunlight. Thus, it would be interesting to



Fig. 9. Detection results of the experiment. The green arrows represent the detected pointing direction after the robot interacted with a human. This pointing direction is then used to select the next exploration goal (blue arrows).

implement the gesture system with other depth sources, for example stereo vision. In this case the depth information will be far more noisy, and additional segmentation algorithms will have to be put in place to robustly detect the hand location. Furthermore it can be interesting to combine the Kinect with a stereo camera setup, in which case the system can switch to a lower quality segmentation in case sunlight overexposes the IR sensor.

The navigation by incorporation of gestures should eventually be able to align multiple gestures to a route graph, offering the ability to pass comprehensive route descriptions to the robot. Every segment of this description would then need a defined ending point such as 'past the door turn right' requiring additional environment perception which already is being developed. Moreover this scenario allows for a multi modal approach from the route graph acquisition, e.g. gesture and speech. Yet a matching method for multiple route graphs resulting from different sources will be needed.

#### REFERENCES

- [1] A. Bauer, K. Klasing, G. Lidoris, Q. Mhlbauer, F. Rohrmller, S. Sosnowski, T. Xu, K. Khlrenz, D. Wollherr and M. Buss, The Autonomous City Explorer: Towards Natural Human-Robot Interaction in Urban Environments, *International Journal of Social Robotics*, 1 (2009), no. 2, 127-140.
- [2] S. Waldherr, S. Thrun, R. Romero and D. Margaritis. Template-Based Recognition of Pose and Motion Gestures On a Mobile Robot. *Proc. of the fifteenth national/tenth conference on Artificial intelligence/Innovative Applications of Artificial Intelligence*, pages 997-982, 1998
- [3] M. Van den Bergh and L. Van Gool, "Combining RGB and ToF Cameras for Real-time 3D Hand Gesture Interaction, *Comput. J.*, Proc. of the IEEE Workshop on Applications of Computer Vision (WACV 2011), January 2011.
- [4] M. Van den Bergh, F. Bosch, E. Koller-Meier, and L. Van Gool. Haarlet-based hand gesture recognition for 3D interaction. *Proc. of the Workshop on Applications of Computer Vision (WACV)*, December 2009.
- [5] P. Breuer, C. Eckes, S. Muller, "Hand gesture recognition with a novel IR Time-of-Flight range camera: A pilot study", *Proc. of the Third International Conference on Computer Vision/Computer Graphics Collaboration Techniques, MIRAGE*, 247- 260 (2007)
- [6] E. Kollorz, J. Penne, J. Hornegger, A. Barke, "Gesture recognition with a time-of-flight camera", *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4), 334-343 (2008)
- [7] S. Soutschek, J. Penne, J. Hornegger, J. Kornhuber, "3-d gesture-based scene navigation in medical imaging applications using Time-of-Flight cameras", *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 1-6 (2008)
- [8] P. Viola and M. J. Jones. Robust real-time object detection. *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 747, 2001.
- [9] M. Klsch and M. Turk. Robust hand detection. *Proc. of the 6th IEEE International Conf. on Automatic Face and Gesture Recognition*, pages 614619, May 2004.
- [10] E.-J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. *Proc. of the 6th IEEE International Conf. on Automatic Face and Gesture Recognition*, pages 889894, May 2004.
- [11] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle and X. Twombly. Vision-based hand pose estimation: a review. *Computer Vision and Image Understanding*, 108:52 73, 2007.
- [12] E. Sanchez-Nielsen, L. Anton-Canalis and M. Hernandez- Tejera. Hand gesture recognition for human-machine interaction. *Journal of WSCG*, 12(1-3), February 2004.
- [13] Adam Kendon. *Gesture: Visible Action as Utterance*. Cambridge University Press, 2004.
- [14] Leo Breiman, Statistics Department, University of California, Berkley, Machine Learning, Springer, 45, 5-32, 2001
- [15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Ng. Ros: an open-source robot operating system. *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.