DELIVERABLE 4.2

Data fusion for robust detection and identification objects and users

Author(s): Ninghang Hu, Ben Kröse (UvA)

Richard Borman

Project no: *287624*

Project acronym: *ACCOMPANY*

Project title: *Acceptable robotiCs COMPanions for AgeiNg Years*

Doc. Status: Final Draft
Doc. Nature: Report
Version: 0.2
Actual date of delivery: 10 September 2012
Contractual date of delivery: Month 12
Project start date: 01/10/2011
Project duration: 36 months
Approver:

DOCUMENT HISTORY

Version	Date	Status	Changes	Author(s)
0.1	2012-09-10	Draft	Initial Draft	Ninghang Hu
0.2	2012-09-30	Draft	Style and spelling	Ben Kröse
0.3	2012-10-10	Draft	Object recognition	Richard Borman

AUTHORS & CONTRIBUTERS

Partner Acronym	Partner Full Name	Person
UVA	University of Amsterdam	Ben Kröse, Ninghang Hu
IPA	Fraunhofer IPA	Richard Bormann

Short description

In this deliverable report, we introduce the system that has been developed in the completion of T4.1: Data fusion for robust detection and identification objects and users.

For object recognition, we fuse data from different modalities to improve the quality of available data for object modelling and detection. Concretely, the colour image data of a colour camera is combined with the depth information gained from stereo vision that is improved with the depth data of a time-of-flight sensor. The result is a dense coloured point cloud at a high resolution. This data is applied in the object recognition system that models the shape and texture of objects to facilitate robust re-detection of those objects in real scenes. In order to avoid the modelling of thousands of objects, the object recognition system is accompanied by an object categorization component that predicts the object's class if no model is available in the recognition module.

In our person detection, tracking and identification system, the ambient cameras are mounted on the ceiling of the Robot House, and their locations are fixed to the world coordinates frame defined by the room. We calibrate the static cameras with OpenCV to facilitate the transfer between the image coordinates and the world coordinates. Since the localization module of the Care-O-bot also runs in the same coordinates system, the data from the ambient camera and the robot can be fused seamlessly. We expect in this way the camera calibration to be more robust than the method that finds correlation of feature points among the different camera images, as the features points are usually rather unstable due to occlusions and rotation of the object.

This report also describes a probabilistic framework for the fusion of data from a laser range finder on a mobile robot and an overhead camera fixed in a domestic environment. The contribution of the framework is that it enables seamless integration with other sensors. For tracking multiple people it is possible to use a probabilistic particle filter tracker. We show that the fusion improves the results of the individual subsystems.

For person identification, the cameras mounted at Care-O-bot's head are used because of their higher resolution. The identification module is based on data fusion between the time-of-flight sensor and a colour camera as well. The depth image is exploited to detect heads in the range of sight of the robot and those regions are inspected in the colour image for the appearance of faces. All detected faces are put into an identification module that asserts the name of the found person. Together with the person tracking via cameras in the environment it is possible to fuse both kinds of information to obtain trajectories of person movements that are labelled with the person's name. Amongst others, this enables the robot to find a target person in the house quicker than with random search.

Table of Contents

AUTHORS & CONTRIBUTERS 1
Short description 1
Table of Contents 3
1. Introduction 3
2. Data Fusion for Object Detection and Identification 3
2.1 Fusion of Stereo Cameras and a Time-of-Flight Sensor 4
2.1.1 Method 5
2.1.2 Results 10
2.2 Object Recognition 14
2.2.1 Methods 14
2.2.2 Results 16
2.3 Object Categorization 17
3. Data Fusion for Human Detection and Localization 19
3.1 Camera Sensory Network 19
3.2 Camera Calibration 20
3.2.1 Intrinsic Calibration 20
3.2.2 Extrinsic Calibration 21
3.3 Data Fusion for Human Localization 22
3.3.1 Related Work 22
3.3.2 System Overview 23
3.3.3 Approach 23
3.3.4 Experiment and Results 28
4. Face Recognition and Tracking 31
4.1 Methods 31
4.1.1 Face detection 31
4.1.2 Face identification 32
4.1.3 Face tracking 32
4.2 Results 33
5. Summary of the Applied Sensor Fusion Techniques 35
6. Conclusion and Future work 37
Bibliography 38
Appendix A 40

1. Introduction

As the baby boom generation is coming to retirement age, the number of elderly citizens over 60 years of age is expected to grow further to a proportion of 1 out of 3 by the year 2030. Alongside this growth in the elderly population, we face short and long-term labour shortages, especially in the health-care sector. Robots may offer a solution for making elderly care affordable by using them for physical [1], cognitive [2] or social [3] support. All these studies share a common foundation that the robots interact intensively with humans, and locations, of both the person and the robot, are estimated robustly.

Robust people detection and localization is a prerequisite for many applications where service robots interact with humans. Future robots will not be stand-alone any more but will operate in smart environments that are equipped with sensor systems for context awareness and activity recognition.

Sensing systems for robot localization or people localization are usually mounted either on the robot or are fixed in the environment. In this report we describe a probabilistic framework for the *fusion* of data from robot and fixed sensors. Here we restrict ourselves to a laser scanner on the robot and an overhead camera fixed in the room. The contribution of our work is that by mapping all information into a probabilistic model, the system can be easily extended with other sensors such as multiple cameras or RGB-D cameras, and is robust to the absence of sensors.

2. Data Fusion for Object Detection and Identification

This section explains the system for object recognition and its prerequisites. To improve the quality of data available to the recognition system, a fusion of different modalities is applied. This preceding step combines the colour image data from a colour camera, computes a depth map of the colour image with the help of the image of a second camera and improves the quality of the depth estimate by integrating the output of a time-of-flight sensor. After the description of this process in Section 2.1 we introduce the system for object recognition in Section 2.2. The object recognition system is able to detect previously learned objects in the scene. However, as the number of occurring objects in a household might be quite large and since nobody is keen to introduce hundreds or thousands of objects to the robot, we also developed an object categorization method. This module is meant to recognize the class of objects that have not been introduced to the robot beforehand. This way the robot can even make sense for many unseen objects without having to learn them in advance. The categorization system gets explained in Section 2.3.

2.1 Fusion of Stereo Cameras and a Time-of-Flight Sensor

The combination of sensor data from different sources aims at creating information that exceeds the quality of each individual source. In terms of quality one usually relates to accuracy, completeness or confidence. The data sources considered in this case are two colour cameras used for stereoscopic vision and one Time-of-Flight sensor that directly deliver 2.5D range data. This work aims at combining both modalities to create accurate 3D

point clouds with associated colour information even in unstructured image areas. In the following, the characteristics of the two sensor modalities are described, beginning with the Time-of-Flight sensor.

Time-of-Flight cameras emit modulated near infra-red light to illuminate a given scene. The reflection of the modulated light is collected in a CMOS matrix. By comparing the returning signal to the camera's source modulation, the phase shift is measured which is a linear function of distance to the reflecting surface. Using the described procedure, the Time-of-Flight sensor is able to operate in real time at about 30 Hz. It creates dense point clouds, however with a limit spatial resolution. As the measurement principle assumes perfectly sinusoidal signals, which are not achievable in reality, the measured distance is subject to noise. It comprises about 1% of the measured distance. Also the measurement principle is biased as a function of object albedo, resulting in poor performance on textured scenes. A prominent example is the distance measurement of a checkerboard, where the black squares seem closer to the camera than the white squares. Additionally, the quality of the measured intensity image is low.

Stereo vision estimates depth through triangulation on point correspondences across image pairs and the knowledge of the cameras' intrinsic and extrinsic parameters. On most textured scenes, stereo is able to provide high resolution point clouds. However, in the absence of features, the system fails to measure depth. Due to the different viewing angles of the two cameras, stereo vision is also prone to occlusions. Additionally, low frequency distortions often disturb the feature association, leading to false depth measurements. Current state-of-the-art stereo matching algorithms achieve accurate dense depth maps only when using global optimization algorithms, needing up to a minute of computation time. Only local correlation based methods are fast enough for real time applications, at the cost of less accuracy and sparse depth maps.

With the combination of a Time-of-Flight camera and a stereo rig, we aspire to unite the advantages of both approaches, that is having dense point clouds with a high resolution within an acceptable computation time. A review of related work has been presented in D4.1. The sensor setup as mounted on the head of Care-O-bot 3 is displayed in Figure 1.

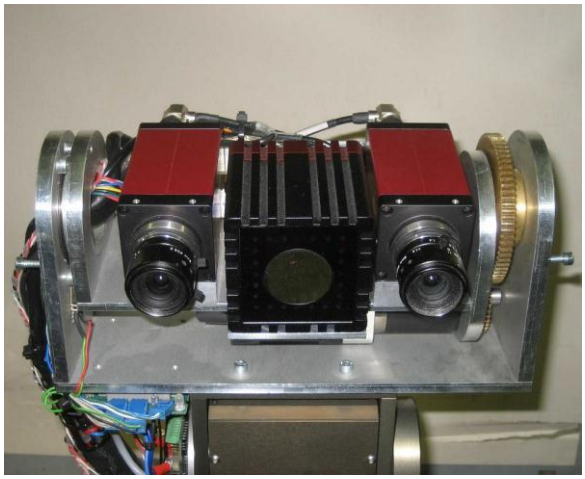


Figure 1 Sensor setup consisting of two colour cameras and one time-of-flight sensor

2.1.1 Method

The first step for successful data fusion from multiple vision sensors is the proper **calibration** of the sensor rig. This process includes the intrinsic calibration of each single device as well as the pairwise calibration between all sensors. For intrinsic and extrinsic calibration the method of Zhang [18] is applied to all cameras using Bouguet's Matlab calibration toolbox [19]. To increase the calibration quality of the time-of-flight sensor we follow the ideas of Lindner and Kolb [20]. The extrinsic calibration results are used to associate depth measurements from the time-of-flight sensor with the corresponding measured disparity from stereo vision or the corresponding colour pixel of one colour camera. The stereo rig is initialized using the extrinsic and intrinsic colour camera parameters. The resulting colour camera's stereo rectified projection matrix enables an association of 3D data with 2D colour image coordinates.

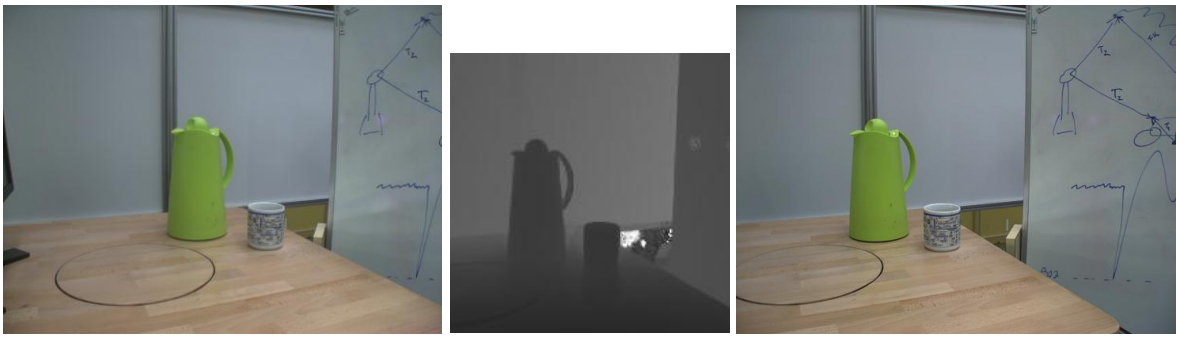


Figure 2 Colour images from the left and right camera as well as the corresponding depth image of the range sensor

Figure 2 displays the source data from the colour cameras and the time-of-flight sensor that becomes combined during the sensor fusion process. Before we can describe the fusion algorithm, some **pre-processing** is necessary as the time-of-flight sensor produces noisy measurements and flying pixels at edges as visible in the side view displayed in the left image of Figure 3. Another source of noise is data that exceeds the non-ambiguity range of the sensor, which lies between 0 m and 5m. To filter the latter source of noise we apply fixed amplitude thresholding [21]. The flying pixels, however, need to be removed with a more sophisticated technique. Therefore, the following wave front propagation

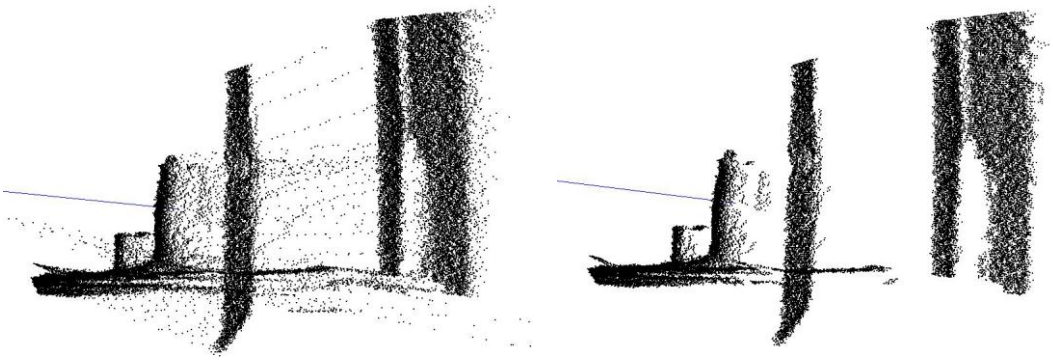


Figure 3 Time-of-flight sensor data before (left image) and after (right image) pre-processing

algorithm is applied after amplitude thresholding. We iteratively expand the neighborhood of each pixel until a maximal depth threshold t_z, relative to the reference pixel is exceeded. The size s_p of the pixel p's neighborhood is compared against a speckle threshold t_s. When $s_p < t_s$ the pixel's range value is labeled as invalid. Otherwise, it is considered as a valid range value. Using wavefront propagation, only range values of pixels with a sufficient number of close-by (in terms of depth differences) neighbors will survive. This directly corresponds to the smoothness assumption made for global stereo vision. The results of filtering the 3D Time-of-Flight data using wave front propagation is shown in the right image of Figure 3.

The filtered 3D measurements from the time-of-flight sensor are now **projected** into the image plane of the left colour camera. Let R_t^l be the extrinsic rotation matrix and T_t^l the extrinsic translation vector that describe the relative pose of the left colour camera to the Time-of-Flight sensor. A 3D measurement $x_t = (x, y, z)^{\mathrm{T}}$ of the time-of-flight camera directly corresponds to the 3D point $x_l = R_t^l\, x_t + T_t^l$ in the coordinate frame of the left colour camera. Based on the intrinsic calibration of the left colour camera we can now employ the 4-by-4 projection matrix Q_l, that transforms 3D points x_l measured in the camera's coordinate frame to 2D points $p_l = (u, v)^{\mathrm{T}}$ measured on the camera's image plane as well as the disparity value d. The calculation $(u', v', d', w)^{\mathrm{T}} = Q_l \cdot (x_l, 1)^{\mathrm{T}}$ uses homogeneous coordinates, that means that the actual values of p_l and d are recovered through division by scale parameter w: $(p_l, d) = (u'/w, v'/w, d/w)$. The projection of the low-resolution time-of-flight measurements to the high-resolution colour image covers only a part of all colour pixels and the coverage has gaps. To be more useful for the optimized depth value computation later, the depth information of the projected time-of-flight measurements is spread to neighbouring pixels at the resolution of the colour image by wave front propagation. To reduce computation time, the propagated range values are not interpolated and copied as they are. However, to express this kind of inexactness an uncertainty value is assigned to the estimated depth measurement that is proportional to the distance of its origin.

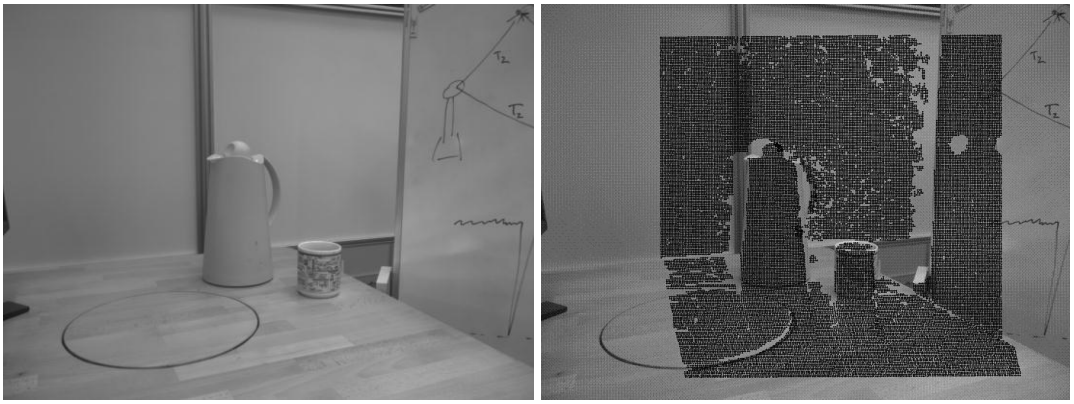


Figure 4 Original greyscale image (left) and time-of-flight measurements projected into it (right).

After the preparation of the raw data from all sensors, we can now **fuse** the information to a high-resolution coloured point cloud. The projected time-of-flight measurements serve as a first guess for the disparity computation from the rectified stereo images. As it is common for the computation of disparities in pure stereo vision algorithms we first have to define **matching costs** that represent how likely two measurements from different sensors are to originate from the same point in the real world. Then we can use this measure to verify the quality of the estimates from the depth sensor. The pixelwise matching costs $C_{BM}(p, d)$ of

pixel $p = (u, v)$ and disparity d are computed as block matching, which is the accumulation of similarity measures within a pixel's square neighbourhood of size $2N + 1$ pixels side length.

$$C_{BM}(p,d) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} C_{BT}(u-i, v-j) \qquad \text{Equation 1}$$

As similarity measure $C_{BT}(p)$ we have chosen the method of Birchfield and Tomasi [22] which determines the difference of intensities in the range of half a pixel along each direction of the epipolar line. When computing the matching costs $C_{BM}(p,d)$ for each pixel p of the colour image and the estimated disparity d from the time-of-flight sensor, we can gauge the validity of the initial depth estimate by considering a pixel's depth estimate as invalid when $C_{BM}(p,d)$ exceeds a fixed threshold t_{BT}. Now we have two cases: first, there are pixels that come with some valid depth estimate, i.e. pixels whose time-of-flight measurement does not contradict with the local texture in the colour image, and second, there are pixels without a valid depth estimate. In the latter case, we do not have a reliable time-of-flight measurement for the pixel and just use the block matching cost function $C_{BM}(p,d)$ from stereo vision. In the first case, we acknowledge that the valid depth measurement is subject to noise. The cost function is chosen as a reversed Gaussian distribution $1 - P_p, P_p \sim N(\mu, \sigma)$ to model this uncertainty, where μ represents the initial depth estimate and $\sigma = \sigma(d)$ stands for the expected sensor noise which is set to be 2% of the measured distance d. Since the depth measurements have been propagated to pixels without depth measurement in the neighbourhood by copying the value instead of interpolation, the uncertainty σ is increased for those cases. To accelerate computation and to incorporate the possibility that Time-of-Flight based disparity guesses may still be wrong, the cost function becomes constant as disparity differences become larger than 2σ. This yields the cost function for matched pixels with valid time-of-flight measurement

$$C_{TOF}(p,d) = \begin{cases} k\left(1 - P_p(d)\right), & \text{if } |d-\mu| < 2\sigma \\ k, & \text{otherwise} \end{cases} \qquad \text{Equation 2}$$

Factor k provides the possibility to scale the maximal costs to a desired value. The total matching cost function is then

$$C(p,d) = \begin{cases} C_{TOF}(p,d), & \text{if time-of-flight data is valid} \\ C_{BM}(p,d), & \text{otherwise} \end{cases} \qquad \text{Equation 3}$$

The optimal disparity $d_{opt}(p)$ for each pixel p is not taken as the value that produces minimal costs $C(p,d)$, because this would yield many false predictions and a very discontinuous disparity image. Instead, we assert a local smoothness assumption and apply the **semi-global disparity optimization** approach of Hirschmüller [23]. The smoothness constraints are supposed to avoid ambiguous matching costs for different disparities and are formulated as the energy function

$$E(D) = \sum_{p} \left(C(p, D_p) + \sum_{q \in N_p} P_1 T\left[|D_p - D_q| = 1\right] + \sum_{q \in N_p} P_2 T\left[|D_p - D_q| > 1\right] \right) \qquad \text{Equation 4}$$

Therein, function $T[\cdot]$ evaluates to 1 if its argument is true and to 0 otherwise. $C(p, D_p)$ represents the cost function from Equation 3 for pixel p with the disparity estimate D_p. The other two terms enforce smoothness of the disparity values within the neighbourhood N_p of point p: the second term penalizes disparity differences of 1 to a neighbouring pixel q with the additional costs $P_1$and the third term penalizes larger differences in disparity with the costs $P_2 > P_1$. Minimizing this energy function over the whole image is a global optimization problem in 2D and known to be NP-complete. The minimization in individual 1D directions is can be computed in polynomial time with dynamic programming, but it generates streaking effects in the direction of optimization. The solution of Hirschmüller performs the 1D optimization from 16 directions surrounding each pixel and hence avoids the streaking effect. An individual cost path is defined by

$$L_r(p,d) = C(p,d) + \min\begin{pmatrix} L_r(p-r,d), \\ L_r(p-r,d-1)+P_1, \\ L_r(p-r,d+1)+P_1, \\ \min_i L_r(p-r,i)+P_2 \end{pmatrix} - \min_k L_r(p-r,k) \qquad \text{Equation 5}$$

where r represents the traversed direction of the individual cost path and i traverses over all disparities except for $d, d+1$ and $d-1$. The first minimization term adds the minimal cost of the preceding pixel for the current optimization direction r, when selecting disparity d for the current pixel. The costs of the preceding pixel are penalized depending on its disparity difference to the currently selected disparity d as explained for Equation 4. To prevent the value of $L_r(p,d)$ from constantly growing while traversing the cost path, the minimum path cost of the preceding pixel is subtracted from the equation. The overall costs are the sum of all cost paths

$$S(p,d) = \sum_r L_r(p,d) \qquad \text{Equation 6}$$

while selecting the minimum $\min_d S(p,d)$ yields the desired disparity for pixel p. We are using the cost function from Equation 3 within this semi-global optimization framework so that the disparity optimization can benefit from both data sources, stereo vision as well as the time-of-flight measurements.

Finally, some standard **post-processing** procedures are utilized to improve the disparity estimates after optimization. First, if there are multiple disparities minimizing $S(p,d)$ the disparity estimate is rejected for its ambiguity. Furthermore, a left-right consistency check ensures that the disparities in the left and right camera image correspond so that the uniqueness of the disparity estimate is likely. Eventually, the final disparity estimate is interpolated based upon the costs of neighbouring disparities at pixel p by

$$d_{new} = d - \frac{S(p,d+1) - S(p,d-1)}{2(S(p,d+1) + S(p,d-1) - 2S(p,d))} \qquad \text{Equation 7}$$

The whole sensor fusion procedure is summarized in the scheme that is displayed in Figure 5.

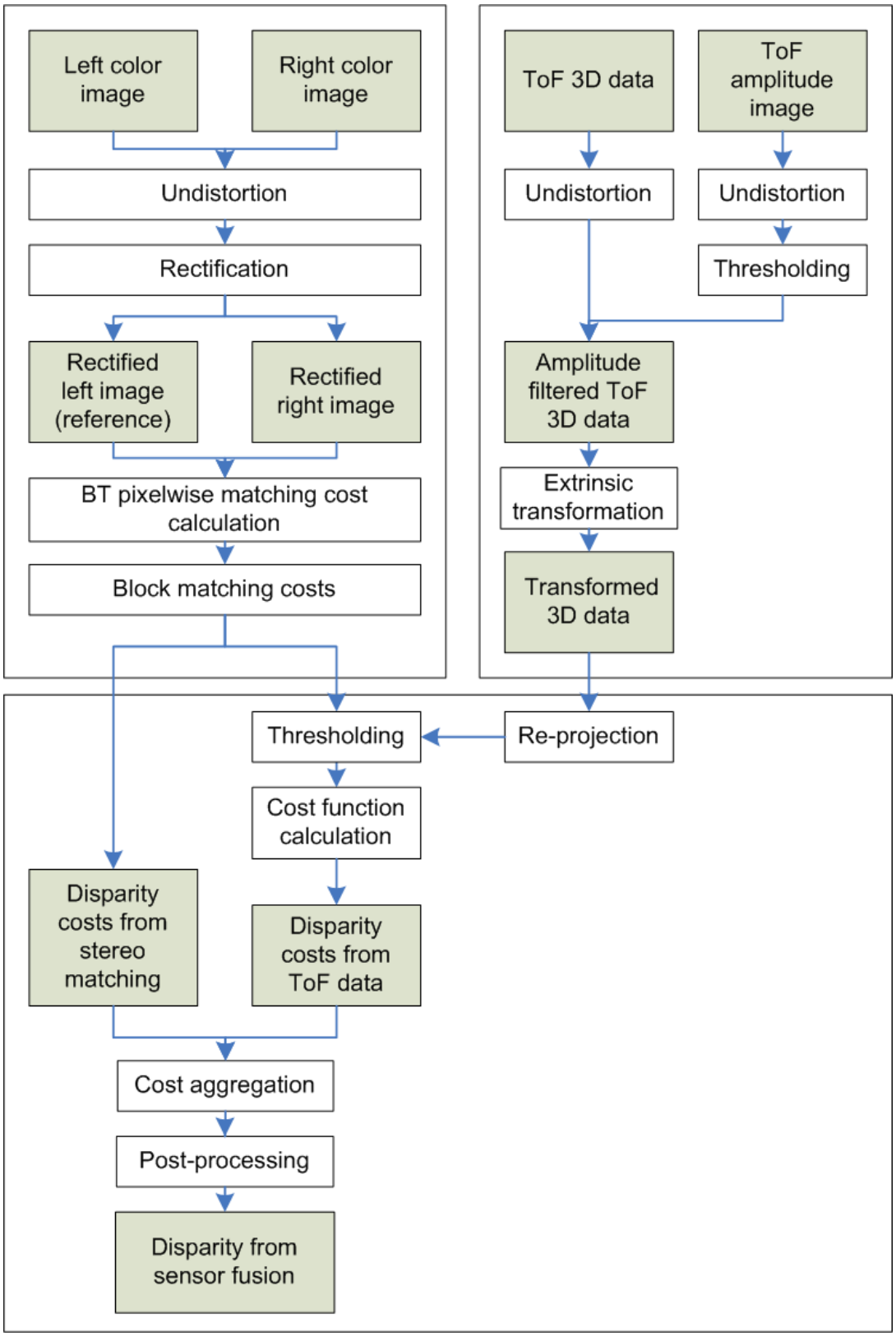


Figure 5: Schematic overview over the sensor fusion of a time-of-flight sensor with stereo vision.

2.1.2 Results

The sensor fusion system has been tested with the SwissRanger SR4000 time-of-flight camera and the AVT Prosilica stereo camera pair mounted on the head of Care-O-bot 3. The two colour cameras have a resolution of 1388x1038 pixels, the SwissRanger SR4000 operates with an image resolution of 176x144 pixels. The cost function has been evaluated for 176 different disparities. Using this setup, the sensor fusion can be computed at a rate of 0.3 Hz when the code is implemented with SIMD based parallelization.

One important factor for the quality of the results is the proportion of projected time-of-flight depth measurements relative to the number of all pixels in the high-resolution fused image. This number can be affected by the size of wave front propagation of time-of-flight measurements to neighbouring pixels without time-of-flight data. Figure 6 illustrates this relation. The upper left image shows a disparity map of the scene from Figure 2 which is purely based on stereo vision. Hence, only textured areas yield depth estimates whereas the depth of plain-coloured surfaces cannot be estimated. The upper right image shows the disparity map when stereo vision is combined with time-of-flight data that was allowed to propagate its depth values in a 1x1 neighbourhood. The depth map is quite similar to the purely stereo vision-based approach since the time-of-flight measurements are outnumbered by the stereo depth estimates so that the latter dominate the semi-global optimization. The numbers of depth estimates from the time-of-flight sensor and stereo vision are better balanced in the lower left image which represents the sensor combination results for a 3x3 propagation neighbourhood of time-of-flight

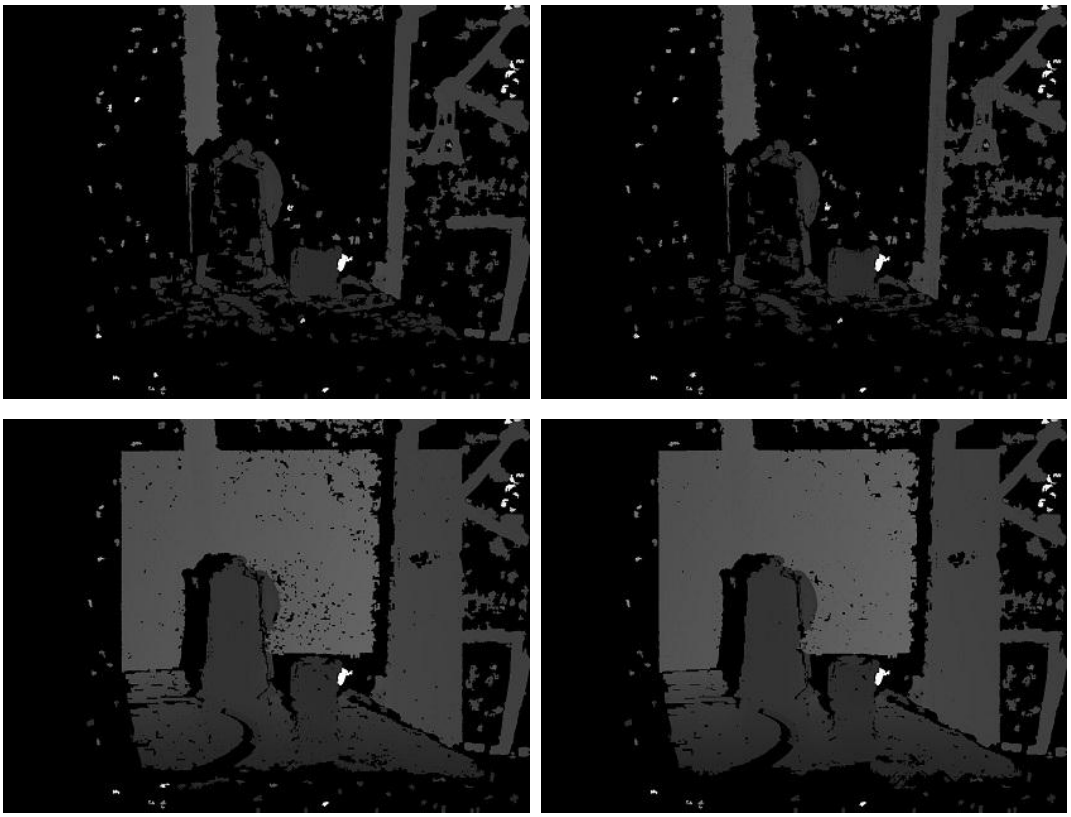


Figure 6 Disparity maps from stereo vision (upper left) and from sensor fusion with time-of-flight data propagation in a 1x1 (upper right), 3x3 (lower left) and 5x5 neighbourhood (lower right).

measurements. This time both sources of information are incorporated to yield a dense depth image. Using a 5x5 neighbourhood renders the stereo map even slightly denser as visible in the lower right image. Nevertheless, the time-of-flight measurements should not be distributed too far as they would overwrite the more accurate stereo depth estimates where they are available. Figure 7 displays several depth maps obtained from the fusion algorithm when using a 5x5 neighbourhood and compares them to estimates from stereo vision only.

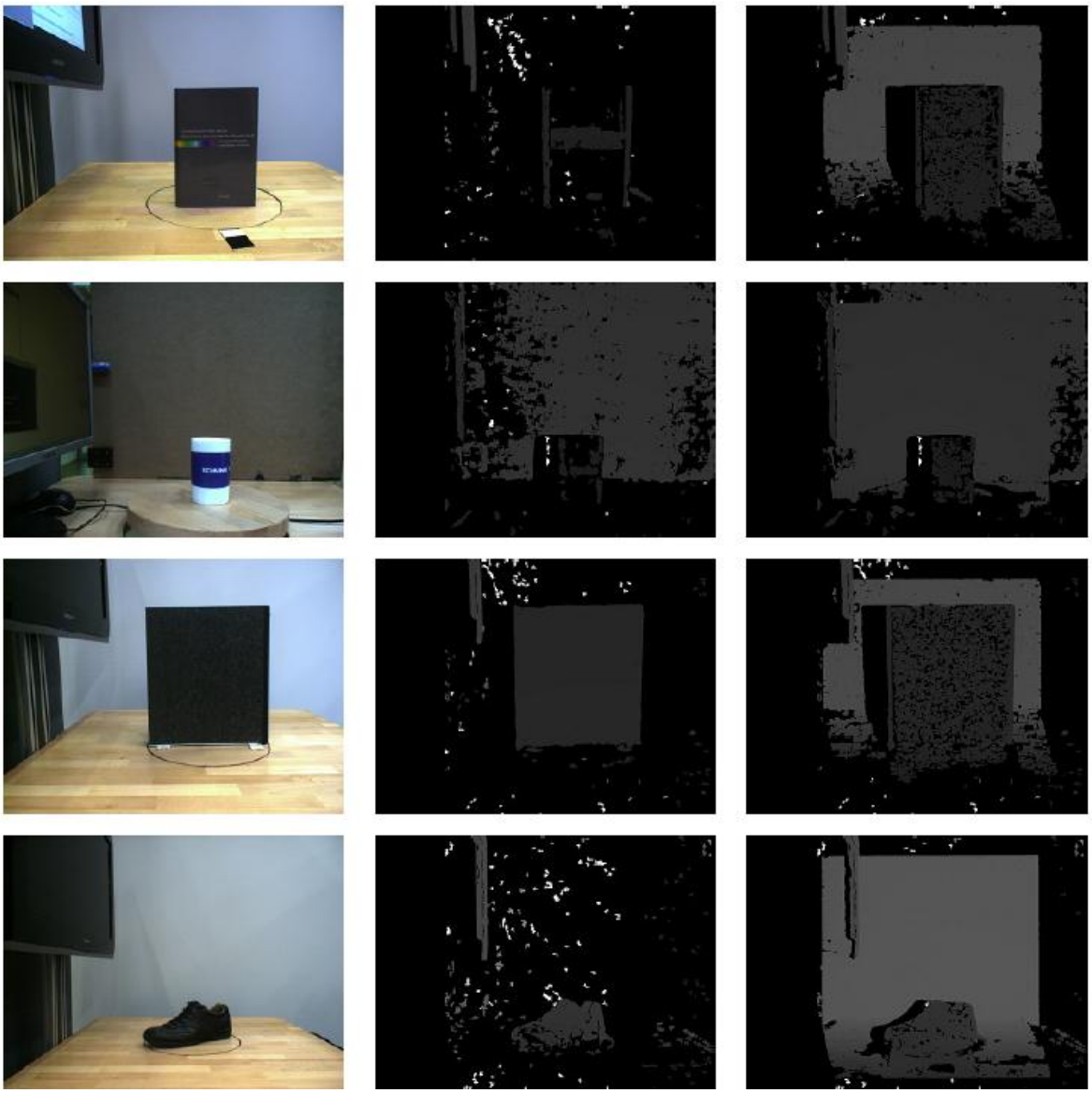


Figure 7 Some examples for the sensor fusion algorithm: the left column shows the original image, the center column displays the depth map from stereo vision and the right column the denser result of the sensor fusion algorithm.

2.2 Object Recognition

This section discusses a system for object recognition that uses the colour and depth data of the previously described sensor fusion algorithm. Object recognition is a necessary prerequisite to enable Care-O-bot for detecting the location of objects and grasping them. The current approach operates on data from a single perspective and is suitable for detecting objects with occlusions and multiple occurrences in highly cluttered scenes

2.2.1 Methods

The detection of objects is based on previously learned models that comprise texture information of outstanding feature points with the 3D location of their occurrence on the object. The **model learning** step requires the object to be placed on a rotary table with attached sensors for model capture or in the gripper of the robot. In both cases, the object is turned so that it can be recorded from different perspectives. For each perspective, STAR [24] feature points are searched within the colour image and a BRIEF [25] descriptor is computed for each feature point (see Figure 8). An accurate 3D location of each feature point is provided by the fusion of stereo and time-of-flight depth information as described in the previous section. The individual perspectives are written into a common object model that contains all detected featured points at their 3D location. Each recording can be related to this common model using the odometry of the rotary table or the robot's gripper. Nevertheless, the odometry is not precise enough to register each view at the correct position in the object model. Hence, an optimization step based on bundle adjustment follows the first model generation pass that is based on pure odometry. The resulting model contains the detected feature points from all perspectives written into a common 3D model of the object. One difficulty for the bundle adjustment is the ambiguity of feature points at repetitive patterns like company labels that are common for many food packages. This problem is tackled by accepting only those feature point registrations that correspond approximately with the measured odometry. Furthermore, the resulting feature point model may contain up to 20,000 points which is a burden for storage and later processing operations. Indeed, many of those points describe the same point on the object but are captured from different perspectives. To reduce the number of feature points, the model points are clustered with mean shift filtering. This results in single centre points for smaller groups of features which relate to the same location. Only the centre points are kept in the model together with the mean descriptor values. The descriptor values, however, are discretized to 0 or 1 after mean shift filtering to keep the data format of the original BRIEF descriptors. This measure reduces the number of model points by around 75%.

Figure 8 Object modeling with the robot (left), colour image image of the object (middle) and detected feature points.

The **recognition of learned objects** in captured scenes proceeds simply spoken by the computation of feature points and their descriptors and by the search for object models that fit clusters of the found feature points in their texture and 3D shape appearance. First, STAR features are searched over the colour image and at each feature point we compute a scale and rotation invariant version of the BRIEF descriptor. Rotation invariance is introduced by using the ORB [26] extension and scale invariance can be obtained when relying on the scale computed by the STAR feature point detector or by relating the measured distance of the object to the patch and kernel size of the BRIEF descriptor. The binary BRIEF descriptor itself is computed by $8n$ comparisons of randomly chosen pixel pairs within the patch of size 48x48 pixels around a feature point. For increased robustness, not only individual pixels become compared but the sum of intensities $p(x)$ within a 9x9 neighbourhood around each sampled pixel x contribute to the comparison. Consequently, the BRIEF descriptor is assembled as follows

$$f_n := \sum_{1 \le i \le 8n} 2^{i-1} \tau(p;\, x_i, y_i) \qquad \text{Equation 8}$$

$$\tau(p;\, x, y) := \begin{cases} 1, & \text{if } p(x) < p(y) \\ 0, & \text{otherwise} \end{cases} \qquad \text{Equation 9}$$

For the feature point matching procedure between feature points from previously learned objects and feature points from the current scene it is not sufficient to go the usual way, this is to take the closest two matches, compute the ratio of their distances and accept the match if the ratio stays below 0.8. We apply a more sophisticated method since the standard approach fails if the same object appears multiple times in the scene. In that case the ratio would always be close to 1 and many good matches would be discarded. Instead, we create two classes of matches: strong matches that have a distance between both descriptors lying below threshold t_{close} are collected in the set M_{close} and weak matches whose matching distance is bigger than t_{close} but smaller than a second threshold $t_{far} \gg t_{close}$ go into set M_{far}. We now check each detected feature point of the scene against every trained object instance O_i whether it is a strong, weak, or no matching and put matches into the sets $M_{O_i}^{close}$ and $M_{O_i}^{far}$, respectively. This allows each feature point to have multiple active object association hypotheses after descriptor matching and before the final object model selection. To localize all existing objects in the scene we then process the sets $M_{O_i}^{close}$ and $M_{O_i}^{far}$ for each object instance O_i individually. First, these sets become sorted by matching distance. Then we down-sample $M_{O_i}^{close}$ spatially to a grid of cell size p (where $p = 4$ in our experiments) by keeping only the strongest matching per cell. This operation yields the set $M_{O_i}^{sparse}$ that contains uniformly distributed matches with a high matching quality. Following, we consider each matching feature point $x_j \in M_{O_i}^{sparse}$ as a seed point for object localization by putting feature points $y_k \in \left\{M_{O_i}^{close}, M_{O_i}^{far}\right\}$ of the local 2D neighbourhood of x_j into the set $M_{O_i}^{local}$ of potential matches for an object. Then, the PROSAC [27] algorithm is employed for guided drawing of point triplets, the model pose estimation based upon those points and verification of the model with the remaining points of $M_{O_i}^{local}$. PROSAC terminates either with a valid estimate that satisfies the non-randomness condition or asserting that there is no valid match between object model and scene points. After processing all seed points $x_j \in M_{O_i}^{sparse}$

according to this procedure, we filter object detections with intersecting bounding boxes by removing the one that fits the model worse.

2.2.2 Results

The obtained object detection and pose estimation algorithm has several desirable properties: it can detect and localize objects in any pose, at different distances to the camera and even if large parts are occluded. If multiple instances of the same object occur in the scene, they can be all detected and localized individually. The procedure can recognize all occurring objects in a scene within a second on a modern laptop computer when the database of known objects contains 10 objects. Some exemplary images of detected objects in a scene are displayed in Figure 9. The bounding box is drawn around every detected object in order to highlight the estimated pose.

Figure 9 Object detection and localization in real scenes mastering variance in object pose and occlusions.

2.3 Object Categorization

In order to enhance the usefulness of the previously discussed object recognition method, we also developed an object categorization algorithm that can determine the category of a formerly unseen object. This method is complimentary to the object recognition approach as is can be used for learning new objects automatically in future applications or as a hint of present objects even when they are not known to the robot, yet. Hence, the versatility of the robot increases because it can deal with a greater selection of objects whose manual training would have been very tedious for the user.

The work on object categorization so far has been published at two conferences:

1. R. Bormann, J. Fischer, G. Arbeiter, and A. Verl, “Efficient Object Categorization with the Surface-Approximation Polynomials Descriptor,” in Spatial Cognition VIII (C.

Stachniss, K. Schill, and D. Uttal, eds.), vol. 7463 of Lecture Notes in Computer Science, pp. 34–53, Springer, 2012.

2. R. Bormann, J. Fischer, G. Arbeiter, and A. Verl, “Adding Rotational Robustness to the Surface-Approximation Polynomials Descriptor,” accepted for publication in Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids 2012), Osaka, Japan, November 2012.

The papers are attached in Appendix A and provide a comprehensive description and evaluation of the developed object categorization system.

3. Data Fusion for Human Detection and Localization

In this section, we introduce the data fusion method of fusion robotic sensors with ambient camera to detect and localize person robustly.

3.1 Camera Sensory Network

The Robot House is the test environment of our project where participants will be invited to perform various activities related to carrying out household chores. After developing and testing the human localization software in local base at the University of Amsterdam (UvA), UvA installed the human localization component to the Robot House in the University of Hertfordshire (UH).

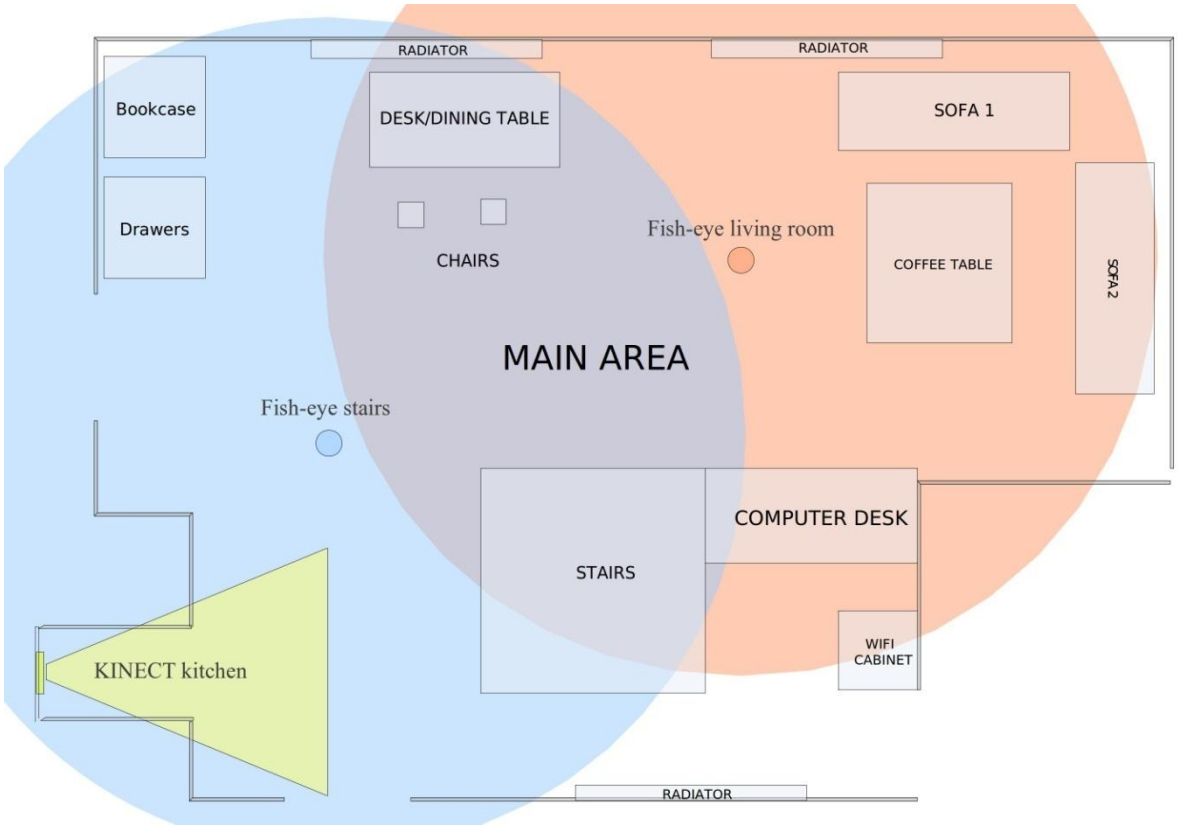


Figure 10: Camera Locations in the Robot House

Two types of cameras have been used in the Robot House. Figure 10 gives an overview of the cameras that are mounted there. The main area is the living room which is the largest area of the Robot House. To localize persons in such an area, two fish-eye cameras (GV-FE421[1]) are mounted on the ceiling, covering the whole living room as well as part of the kitchen and corridor. The fish-eye camera gives an omnidirectional view of the environment and enables us to monitor all angles of a location with less occlusion.

[1] www.geovision.com.tw/english/Prod_GVIPCAMH264Fisheye4.asp

The kitchen area of the robot house is relatively small, and a Microsoft Kinect sensor[2] is considered to be sufficient for monitoring users in the relatively small space. The Kinect sensor is mounted on the left side of the kitchen above the window to avoid the direct sunlight from the window and also keeps a whole field of view of the kitchen. In the current work, only the colour component of the Kinect is used for localizing humans in the kitchen. But we expect the depth component would be more useful in the next step of our research to recognize human activities.

Figure 11 gives an overview of how the devices are connected in our localization system. Each of the camera sensors is connected with a dedicated PC via a fast wired connection. After receiving image streams from the camera, the PC processes the data and generates intermediate products for data fusion. All three dedicated PCs are connected to a network switch and a wireless Router, to enable data communication with the Care-o-bot and other PCs.

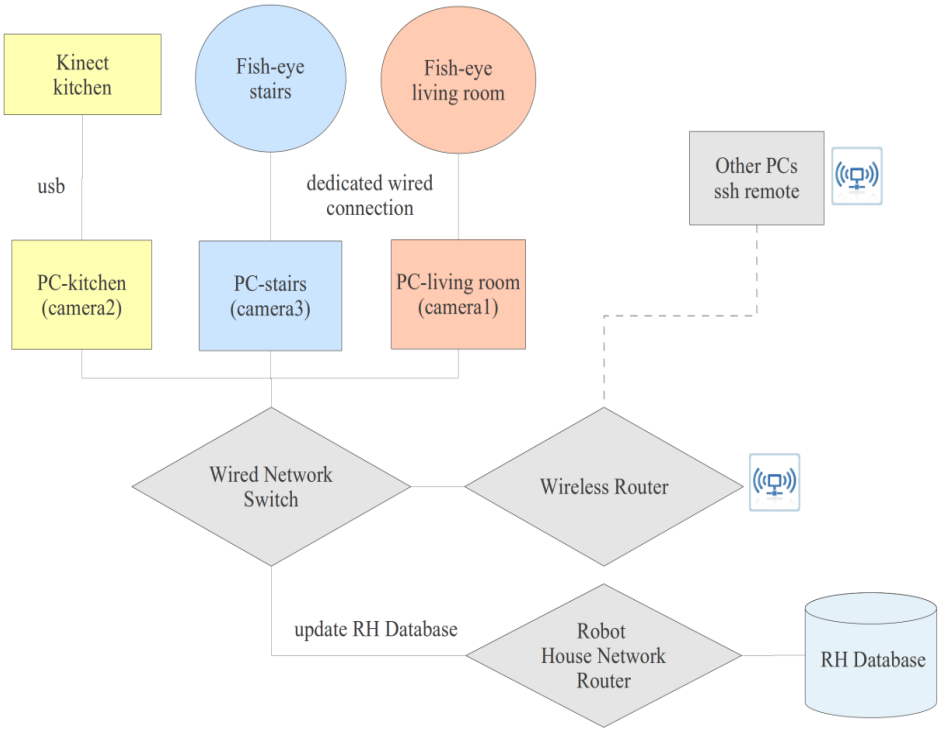


Figure 11: sensory system network

The Robot Operating System (ROS) are used as the software framework when building the human localization module. With ROS, the messages from the Robot can be easily accessed from the dedicated PCs, facilitating the data fusion of ambient cameras and robot sensors.

3.2 Camera Calibration

Benefiting from the fish-eye effect, our cameras have an Omni-directional field of view of the environment. However, the images from the fish-eye cameras are highly distorted and cannot

[2] www.xbox.com/KINECT

be used directly for localizing people. The intrinsic parameters of the cameras, therefore, have to be estimated to eliminate the image distortion. After the intrinsic parameters are calibrated, we then find the rotation and position of the camera in the world coordinate frame by extrinsic calibration. In this section, we introduce our method of estimating the two set of parameters that are necessary for transferring coordinates between the image frame and the world coordinate system.

3.2.1 Intrinsic Calibration

The Intrinsic model of the camera defines the curvature of the lens in the fish-eye cameras, i.e. in which way the lens is distorted. The OpenCV library [4] provides functions that calibrate the intrinsic parameters of the cameras using the checker board pattern[3].

Figure 12: the checkerboard pattern used to calibrated intrinsic parameters of the camera

The intrinsic calibration involves of estimating two parameters of the camera: a camera matrix and the distortion coefficient. The camera matrix defines the location of the optical centre of the camera and the focal length expressed in the pixel coordinates. The distortion coefficient controls how the distortion spreads over the image space.

Figure 12 shows a sample image of the checkerboard pattern that is used in calibrating the intrinsic parameters of the fish-eye camera. Our calibration software finds out where the corner points of the checker board pattern locate in the image coordinates, and use them to approximate the camera matrix and the distortion coefficient. Each pattern gives a set of locations of the corner points and generates a function of the intrinsic parameters. By using a number of such images with different attitude and positions, we can compute the intrinsic parameters that fit the functions optimally.

[3] The documentation for the camera calibration and the environmental setup is available at http://basterwijn.nl/ninghang/robot_house/

3.2.2 Extrinsic Calibration

The extrinsic parameters of the camera define the location of the camera in the room and how the camera is oriented. After apply the image with intrinsic parameters, we are able to transfer coordinates between the undistorted image space and the locations on the floor of the Robot House.

Figure 13: markers are attached on the floor of the Robot House for extrinsic calibration

We made a set of markers on the floor as shown in Figure 13. The markers were put with an interval of one meter and their locations were manually annotated from the image. Combining their corresponding locations in the world coordinate frame, we are able to compute the translation and rotation matrix between the two coordinate systems. Thereby the extrinsic parameters are estimated.

3.3 Data Fusion for Human Localization

3.3.1 Related Work

There is a long tradition of research in the field of people detection and localization in robot applications. Many studies concentrate on people detection using the sensors on the mobile robot. Relatively simple sensors such as laser range finders were used for detection and localization [5][6]. People are extracted from range data as single blobs or found by merging nearby point clusters that correspond to legs. Probabilistic techniques such as multi-hypothesis trackers are used for tracking multiple objects [7].

Instead of using the laser range systems on the robots, vision systems have also been used for people detection. Since robot-mounted cameras are moving, the detection cannot be based on background modelling methods, and local characteristics such as colour

histograms or local features have been used [8][9]. To make detection more robust, the fusion of different modalities of robot sensors is suggested. Leg detection by laser range finders in combination with face detection has shown to be more robust than individual modalities [9] [10]. In [11], Viola-Jones type of visual detectors are used to recognize body parts and are combined with laser range data.

However, future robots will operate in smart homes that are equipped with sensors, and it seems obvious to use these sensors also for person detection. One advantage is that the system may be more robust: noise or deviations in a sensor may be detected and corrected. Another advantage is that the robot does not need to keep monitoring the persons all the time. The robot may be required to finish other tasks from time to time, rather than allocating its resources to the task of tracking each person all the time.

Person tracking systems that are mounted in domestic environments are usually based on vision systems, although there are some exceptions using laser range finders [12] or speech [13]. Overhead cameras are often used, which are usually mounted very high, and have a very wide angle of view, covering most of the areas in the room. Since they look down from above, it turns out that human users are less likely to be occluded compared with cameras mounted on the side. An application in a kids playroom is given in [14].

In our set-up we combine an overhead camera with the laser range finder on the robot. In order to have a sound probabilistic framework we build on the approach of [15], who uses a probabilistic foreground segmentation with a template based detection. The result is a posterior distribution on the locations of the persons in the room. This is combined with a distribution based on the laser range finder.

3.3.2 System Overview

Our proposed system is used to detect and localize the elderly people in chores of robot home-care. With our system, the robot is able to obtain accurate locations of the users in the room, and thereby it can interact with the human users precisely. The robot we use possesses multiple on-board sensors, including a Microsoft Kinect camera, a stereo camera, and a laser range finder.

In the recent work, most of the robots are designed for following the targets. These approaches, therefore, require that the users are always in the range of the robot sensors. In the case of home care, however, the robot moves around in the room to execute a variety of tasks, and at some points the robot sensors will loss the track of the human user, *e.g.* the robot is asked to get an object that is in an opposite direction to the user. To overcome such a problem and enable continuous human localization, we adopt an ambient camera and mount it on the ceiling of the room. The advantage of the ambient camera is twofold: (1) that it gives a top view of the whole room, and (2) that people in the room are less likely to be occluded compared with the robot cameras. Since it covers the whole area of the room, the ceiling-mounted camera is able to localize persons continuously when the users are present in the room, so that when the robot sensors fail to detect the users, the ambient camera is still able to report the correct location to the system. Besides, the robot sensors and the ambient camera observe the persons from different directions, giving complementary cues for the human detection and localization. The fusion system can, therefore, obtain a better estimate of the location of the users compared with the approaches using single modality.

To combine the robot sensors and the ambient camera, we propose a Bayesian fusion framework. Next, we formulate the problem and introduce our fusion framework.

3.3.3 Approach

The Bayesian approach provides an elegant way of fusing between different sensor sources as well as dealing with noise and uncertainty in sensor measurements [16].

Assume I_R is the observed data from the robot sensor, and I_C is observed from the ambient camera, *i.e.* the overhead camera. Given I_R and I_C , we aim to find a robust estimation of the location of multiple persons L_H, the location of the robot I_R , and the orientation of the robot θ_R. In the context of a Bayesian framework, the posterior distribution $P(I_R, I_C, \theta_R | I_R, I_C)$ is the target we would like to know by the end.

Using the Bayesian Theorem, the posterior probability can be derived as

$$P(L_R, L_H, \theta_R | I_R, I_C) \propto P(I_R, I_C | L_R, L_H, \theta_R) P(L_R, L_H, \theta_R) \quad \text{Equation 10}$$

where $\mathrm{P}(\mathrm{L_R}, \mathrm{L_H}, \theta_\mathrm{R}) = \mathrm{p}(\mathrm{L_R})\,\mathrm{p}(\mathrm{L_H})\,\mathrm{p}(\theta_\mathrm{R})$ is the prior distribution that is known before the sensory data is observed. These priors can be estimated either from separate training data, or from prior knowledge of the problem. In our case, we simply assume a uniform distribution over the ground area of the floor, and a uniform distribution over the angles of the orientation. $P(I_R, I_C | L_R, L_H, \theta_R)$ is the likelihood.

By assuming I_R and I_C are measured independent with separate sensors, and I_C is not dependent on the rotation of the robot θ_R, the likelihood probability of Equation 11 can be decomposed as

$$P(I_R, I_C | L_R, L_H, \theta_R) = P(I_R | L_R, L_H, \theta_R) P(I_C | L_R, L_H) \quad \text{Equation 11}$$

where $P(I_R | L_R, L_H, \theta_R)$ is the likelihood of generating the image I_R given the combination of I_R , L_H , and θ_R , while $P(I_C | L_R, L_H)$ represents the likelihood of the ambient camera that generates the observation I_C .

Again, our goal is to find the optimal combination of L_R^* , L_H^* and θ_R^* that maximizes the posterior distribution $P(I_R, I_C, \theta_R | I_R, I_C)$, which is a typical MAP problem that can be solved by particle filtering [17].

The camera likelihood $P(I_C | L_R, L_H)$is used as the proposal distribution to sample particles, and the particles are weighted by the corresponding likelihood of the laser data $P(I_R | L_R, L_H, \theta_R)$. The optimal combination of L_R^*, L_H^* and θ_R^* is considered as the particle that holds the highest weight. In a Bayesian framework, however, we find the expectation of the parameter values rather than the most probable value. Therefore, rather than choosing one particle that maximizes the joint distribution, we compute the solution as a weighted sum of all the particles.

The remaining is to compute the two likelihood terms in Equation 11. In the following two sections, we introduce the methods of estimating the two likelihood items separately. Here, we will focus on modelling the likelihood of the robot sensor. For the camera, we adopt the algorithm from [15].

Measuring Likelihood of Robot Sensors

In our data fusion framework, the state is to be estimated is a triplet of $\{I_R, L_H, \theta_R\}$. The likelihood of the robot sensors measures the probability of generating the observation I_C rather than all the observations that can be possibly generated from such a triplet, given such a state triplet, *i.e.* $P(I_R|L_R, L_H, \theta_R)$.

In this report, we adopt the Laser Range Finder as our robot sensor. The Laser Range Finder scans in a plain and detects the distance to the objects in range. In the context of human localization in a home setting, the detected objects can either be objects that exist in the room or be part of the human in the room. In this report, we use the background model and the human model, respectively, to model the probability that a region is occupied by either of these two objects. Then we can compute the occupancy map of the room, *i.e.* the probability that the area is occupied by either the background object or by a human.

The occupancy maps are used to estimate the probability of the robot sensor generating a certain set of observations, *i.e.* the robot sensor's likelihood.

Probabilistic Background Model

To find out what the room looks like in terms of background obstacles, the robot is first driven around to build a background model of the room.

For each time stamp, the robot sensor fires a set of laser beams $l = \{l_1, l_2, l_3, \dots\}$. Whenever there is an object in the way, the laser is reflected back to the base and thereby the distance to the background objects is detected. Given the coarse location of the robot, we are able to find the approximate locations of these laser detections. But due to the uncertainty in the location of the robot as well as the noise in laser data, these locations are not fully reliable. Therefore, simply giving a Boolean answer to the occupation of the local region is not an elegant solution, and a probabilistic way of modelling the background is required.

In our approach, the ground plane is first discretized into small cells of equal size. We denote *k* as the index of the cell on the ground plane. Then for each cell *k*, we aim to estimate the probability that the cell is occupied by a part of the background. Collectively, these probabilities form the background model $P_b(k)$.

In this report, the background model $P_b(k)$. is measured as the number of times the laser scanner observes an occupied cell normalized by the number of times that the cell is in the range of the laser scanner. To formalize the problem, we define three patterns that can be observed given a scan l and a cell *k*, see Figure 14Figure 14: The relation between a laser beam and a cell can be summarized into three patterns. In the left pattern $\boldsymbol{\alpha_k^l = 1}$, the laser is blocked by the cell, referring that the cell is occupied by certain background objects. The middle pattern $\boldsymbol{\alpha_k^l = 2}$ shows the laser has passed through the cell, indicating the cell is empty. As for the third pattern, however, the laser beam is blocked before it reaches the cell. Therefore, no clue about whether the cell is occupied can be deduced from the third pattern.. We use a random variable α_k^l to denote the index of the three patterns. The first pattern refers that the cell *k* is detected by l as an occupied cell. The second pattern denotes that the cell is observed as an empty cell. As for the third pattern, no information about the cell can be inferred since the cell is either occluded by other background objects that are in front of the cell, or the laser is not fired in the direction of the cell. Therefore, the third pattern does not

contribute to the background model while only the first two do. Next, we estimate the background model by

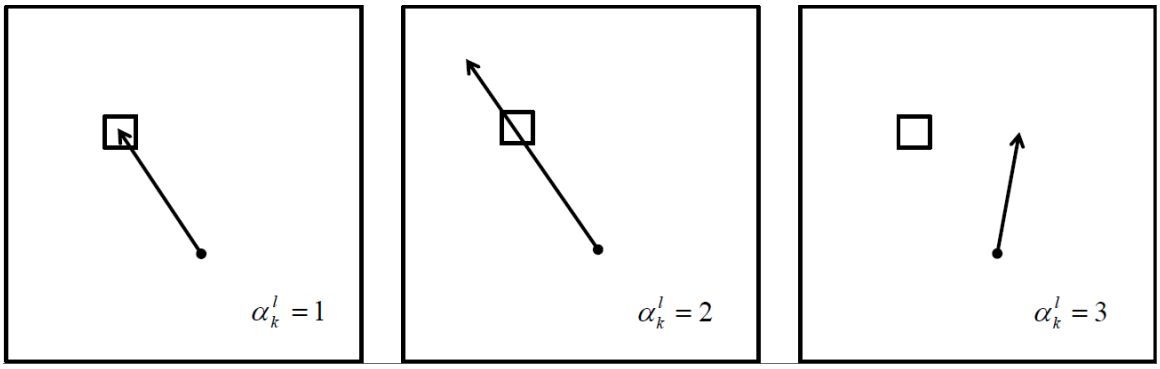


Figure 14: The relation between a laser beam and a cell can be summarized into three patterns. In the left pattern $\alpha_k^l = 1$, the laser is blocked by the cell, referring that the cell is occupied by certain background objects. The middle pattern $\alpha_k^l = 2$ shows the laser has passed through the cell, indicating the cell is empty. As for the third pattern, however, the laser beam is blocked before it reaches the cell. Therefore, no clue about whether the cell is occupied can be deduced from the third pattern.

$$P_b(k) = \frac{\sum_l \delta(\alpha_k^l - 1)}{\sum_l \delta(\alpha_k^l - 1) + \delta(\alpha_k^l - 2)} \quad \text{Equation 12}$$

where δ is a Kronecker delta function, and the equation sums over all the lasers that pass through the cell *k*.

Learning Human Model

The human model P_h reflects how the human looks like from the robot sensors in the world frame. It is learned by accumulating the laser points that locate in a small region around the center of the person. Each pixel in such a region holds a value indicating the probability that the cell is occupied by the person, *i.e.* a higher value means the cell is more likely to be detected by the robot sensor due to the occurrence of the human.

Similar to training the background model, we learn the human model P_h by calculating the number of laser beams that either have a positive detection at the cell or pass through the cell. Again, we adopt the Equation 12 for computing the human model P_h.

Given the person locating in cell *k*, the local human model P_h can be translated into the world frame to generate a human model map $P_h(k)$.

Occupancy Map

Knowing the background model and the human model, we are able to compute the probability of occupancy for each of the cells on the ground plane. Note that the cell cannot be occupied by both the human and the background obstacle at the same time, therefore the occupancy map is computed as

$$\omega_k = \frac{P_b(k)\hat{P}_h(k) + P_h(k)\hat{P}_b(k)}{1 - P_b(k)P_h(k)} \quad \text{Equation 13}$$

where

$$\hat{P}_h(k) = 1 - \mathrm{P(k)} \quad \text{Equation 14}$$

Likelihood of Laser Range Finder

The likelihood of the Laser Range Finder denotes the probability of generating the current observation given the state $\{I_R, L_H, \theta_R\}$. I_R represents a vector of the laser range data. Assume I_R contains N independent measurements $\{i_R^1, i_R^2, \ldots, i_R^n, \ldots, i_R^N\}$. Suppose the direction of the range measurement i_R^n is defined by θ_R^n. Therefore

$$P(I_R|L_R, L_H, \theta_R) = \prod_{n=1}^{N} P(i_R^n | L_R, L_H, \theta_R^n) \quad \text{Equation 15}$$

I_R and θ_R^n define a robot at the location I_R, and the robot fires a laser beam in the direction of θ_R^n. L_H refers to the location of multiple persons.

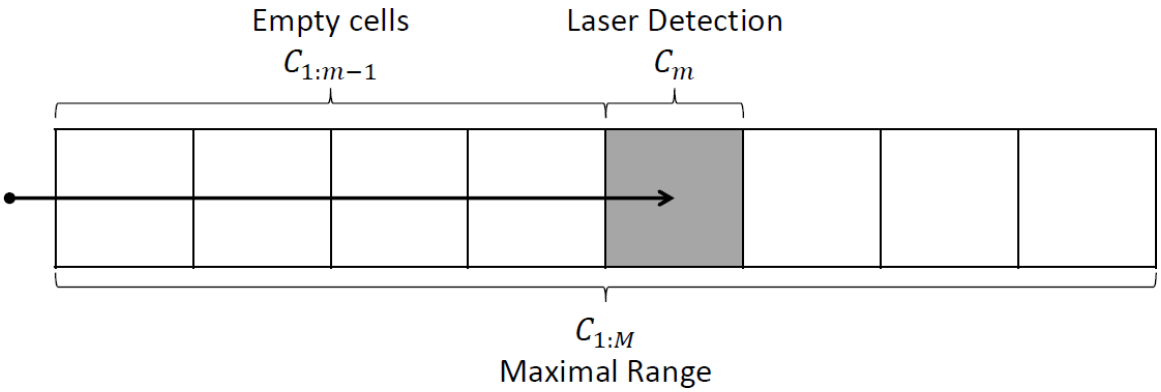


Figure 15: The laser beam (Arrow) passes through m-1 empty cells and finally reaches the cell at c_m. The maximal range of the laser covers M cells

Suppose the laser beam i_R^n passes through a set of cells in a straight line, *e.g.* $\{c_1, c_2, \ldots, c_{m-1}\}$, and then it detects a certain object at the cell c_m. c_m denotes the maximal range that the laser can reach. See Figure 15. Then the probability of obtaining detection at cell c_m rather than the other locations can be computed by

$$P(i_R^n|L_R, L_H, \theta_R^n) = \frac{\omega_{c_m} \prod_{i=1:m-1} \hat{\omega}_{c_i}}{\sum_{j=1:M} \omega_{c_j} \prod_{i=1:j-1} \hat{\omega}_{c_i}} \quad \text{Equation 16}$$

Since multiplications of the probabilities can result in very small numbers which lead to floating point overflows, we compute the log-likelihood instead

$$L(i_R^n | L_R, L_H, \theta_R^n) = \lambda_m - \sum_{j=1:M} \lambda_j \quad \text{Equation 17}$$

where

$$\lambda_m = log(\omega_{c_m}) + \sum_{i=1}^{m-1} log(\hat{\omega}_{c_i}) \quad \text{Equation 18}$$

Likelihood of Ceiling Mounted Camera

The likelihood of overhead camera is computed the same way as in [15]. Assuming the pixels are independent from each other given the image taken by the ceiling mounted camera, the likelihood $P(I_C|L_R, L_H)$can be derived as

$$P(I_C|L_R, L_H) = \prod_{n=1:N} P(i_C^n|L_R, L_H) \quad \text{Equation 19}$$

We build a specific polyhedron to model the 3D shape of both the human and the robot. Given the location of the human L_H and the robot I_R , the polyhedrons are projected into the image space, generating a foreground mask M. For each pixel location $P(i_C^n)$ on the image, we look up in the mask and use M_n to determine whether the pixel is a part of the foreground or background. Then the likelihood can be computed as

$$P(i_C^n|L_R, L_H) = P_f(i_C^n)M_n + P_b(i_C^n)(1 - M_n) \quad \text{Equation 20}$$

where $P_b(i_C^n)$ is the background model which is learned beforehand using the background images. $P_f(i_C^n)$ is the foreground model, and in our case we apply a uniform distribution over the colours.

3.3.4 Experiment and Results

The proposed data fusion framework was evaluated on data collected with a Nomad robot platform and an overhead camera, see Figure 16. The overhead camera is mounted centrally on the ceiling and gives a panoramic view of the room. The frames that are captured with the camera are highly distorted due to the fish-eye effect. The camera's lens parameters are calibrated with the OpenCV module [4].

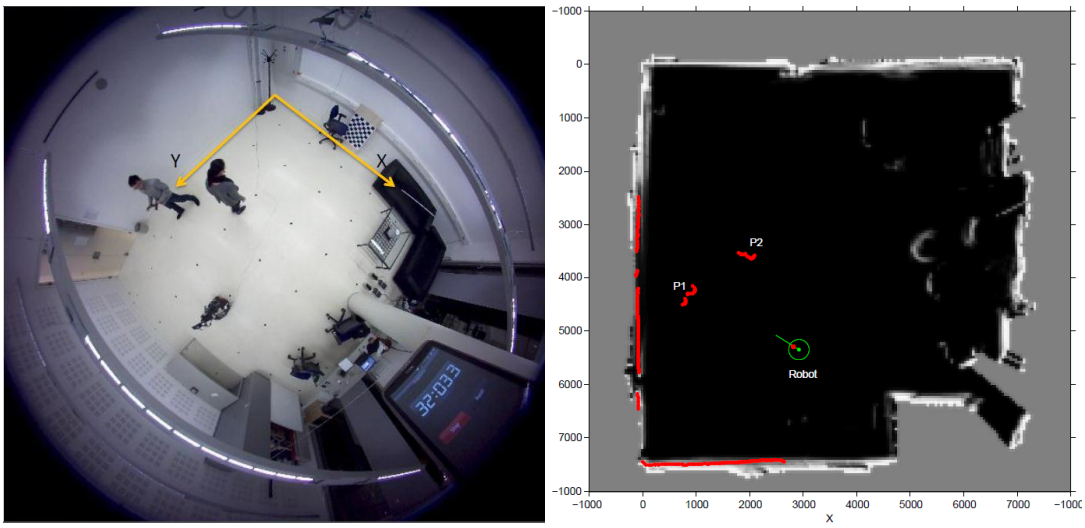


Figure 16: An overview of the experiment room and the observed data. Left: captured by the over head camera; Right: laser detection points (red dots)

On the Nomad robot platform, a Laser Range Finder, a Kinect camera and a stereo camera are mounted on the robot. For the present experiments, we restrict ourselves to test the framework by using the Laser Range Finder, mounted at a height of 20 cm. The robot is remote-controlled and manually driven around in the room. The robot records its odometry information by measuring the rotations of its two wheels. The odometry data are then adopted for generating the orientation and location of the robot. The Nomad robot runs on the Robot Operating System (ROS), and all data captured on the robot site is time stamped in ROS.

The exact time stamp of each frame collected with the overhead camera is obtained by means of a stopwatch mounted close to the camera. We use a nearest-neighbours classifier

to recognize digits in the image to recover the time stamp. We synchronized the robot sensors and the overhead camera based on specific time points, where an event (*e.g.* the puncturing a balloon in front of the Laser Range Finder) was observed by both the robot sensor and the overhead camera.

The ground plane is subdivided into small cells of 50x50mm. In a first training run, the robot was remote-controlled to generate the background model. Second, the human model is trained according to Equation 12 . During testing, the two models are combined probabilistically into an occupancy map given the particles, as depicted in Figure 17. Here each pixel of the occupancy map reflects the probability that that location is occupied, either by a person or by a background object in the room.

Figure 17: The occupancy map is generated from combining the human model and the background model. For each set of locations of persons, *i.e.* a particle, an occupancy map is estimated. Left: Human model. Middle: Background model. Right: Occupancy Map given the hypothetical location of persons (green crosses)

We evaluate the systems by measuring the Euclidean distance between the detection results and the ground truth locations of persons. In this report, three localization approaches are tested and compared: a) with a single Laser Range Finder; b) with a single overhead camera; c) with our proposed fusion framework. We evaluate the proposed system and the single modality approaches on 165 camera frames together with synchronized laser data. For each of these frames, volunteers manually annotated the locations of the persons in the ground plane, based on physical markers that were positioned on the floor during the recording, and these markers were used as reference to compute the ground truth location.

A particle sampling approach is applied both in the single laser and the data fusion approach. An equal number of 800 particles are sampled. Due to the fact that humans are not likely to be too close to each other, we define the safe distance between two persons as 500mm. We incorporate such assumption to reduce the space when sampling particles, *i.e.* the sampled point is always at least 500mm away from each of the points the previous sample set.

The single laser approach detects the foreground laser points by set a threshold to their probability in the background model. The threshold in our experiment is empirically set to 0.3. The particles are sampled from the foreground laser points with a Normal distribution on the location of the points. The weights are assigned by the likelihood of the laser data given the particles, and they are quantized in the sub-divided cells on the ground plane according to the locations of the particles. The human is then localized by recursively finding the cell that has the largest sum of weights as in [15].

In the approach with a single camera, we adopt the human detection algorithm from [15]. For each candidate location of the persons on the ground plane, the likelihood of the camera

frame is measured. The locations of the multiple persons are found by choosing the locations that maximize the likelihood of the camera image.

The proposed approach combines the overhead camera and the robot Laser Range Finder in a probabilistic Bayesian framework. After persons are localized with the single camera, the particles are sampled around the location of the persons with a Normal distribution. These particles are then weighted by the likelihood of the laser observations. The final detection is computed by the weighted sum of the particles that are sampled from the same person.

Figure 18 shows the detection results of our data fusion system comparing with the approach using single modality. The proposed fusion system consistently outperforms the single-camera and the single-laser approach, and approximately 80 percent of the detections are less than 200 mm from the ground truth location. In contrast, only 70% of the camera-only detections and 27% of the laser-only detections are within such distance of the ground truth.

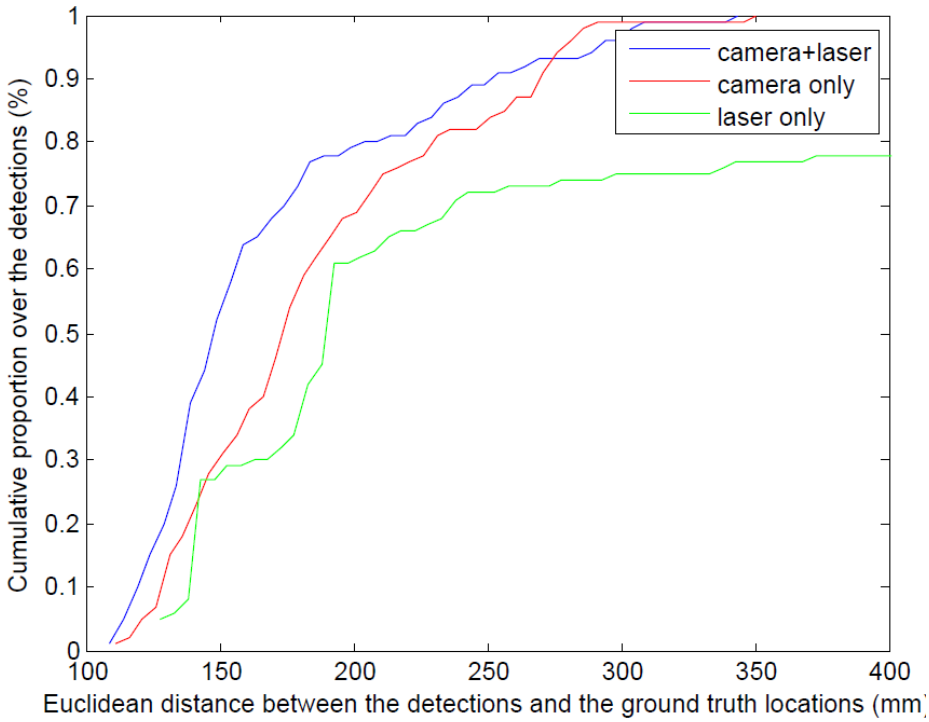


Figure 18: Comparing the proposed data fusion approach and the single modality approach

4. Face Recognition and Tracking

The person recognition and tracking system so far bases upon the data from cameras mounted at the ceiling of the room that have a wide viewing angle. Consequently, it is not possible to recognize the identity of a person because of the low resolution and because of the inappropriate perspective for this task. Nevertheless, person identification is highly desirable for human-centred and individualized tasks like reminding to medicine or bringing some ordered item to the user. The cameras that are placed in the robot's head are well-suited for the identification of faces as they capture persons from an adequate perspective and have the necessary resolution to record details. Therefore, the identification of persons is realized with the movable Care-O-bot. By using the robot's localization in the map of the house and the calibration of the ceiling cameras to that map, the data can be streamed as the trajectories of persons that are attached with name tags. The component described in this section is already available as open-source ROS package at http://www.ros.org/wiki/cob_people_perception.

4.1 Methods

The face identification module has three parts: first, there is the localization of faces in the image, then follows the identification of those detected faces and finally we have a component that organizes some tracking of found faces to render the labels more robust.

4.1.1 Face detection

The first step is face detection, which is the localization of regions in the image that contain a face. The standard approach for this task is the robust and fast method of Viola and Jones [28] that employs a classifier cascade based on Haar-like features to reject non-face regions on an image of a colour camera. We use this approach because of its reliability and efficient computation but modify it to reduce the number of false positives. Our modification uses the possibility to rely on depth data as well that originates from the sensor fusion algorithm presented in Section 2.1 or directly from a RGB-D sensor device like the Asus Xtion Pro live. The idea is to search for heads on the depth image with the procedure of Viola and Jones, first, to lower the chances of false positive face detections in the subsequent application of the Viola-Jones classifier to the colour image. The two classifier cascades are trained with exemplary heads regions found in depth images and face regions from the colour image, respectively. The head regions, however, contain a significant amount of background which is a problem for detection at other places. We hence generated a plethora of random background patterns automatically and placed them in the background of the training images for the cascade classifier. Figure 19 illustrates the process of two-staged face detection: first, the depth image is searched for head regions using the Viola-Jones classifier (1) and only for those regions that probably contain a head, we process the colour image with the Viola-Jones classifier for faces (2).

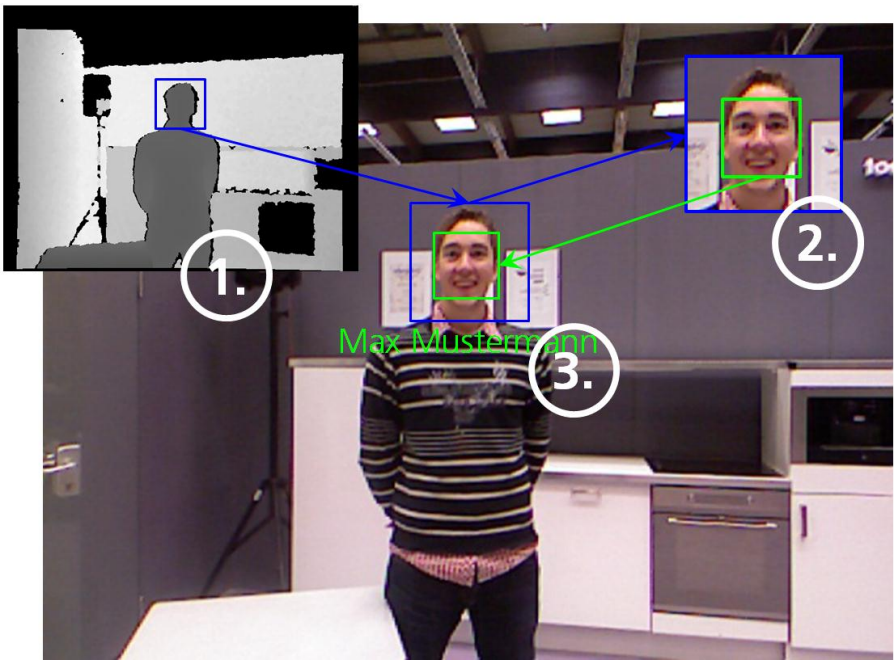


Figure 19 Face detection and identification.

4.1.2 Face identification

After the selection of face regions in the colour image the identification of these image patches follows. We use the Eigenfaces [29] algorithm which identifies a face by projecting it into a space of eigenfaces. Those eigenfaces are obtained as the principal axes when applying PCA to a large set of images depicting human faces. The eigenfaces that correspond to the largest eigenvalues can be considered as the most important prototypes for assembling an image of a human face. For training a recognition model of person A we capture several images of A's face and project them into the space spanned by the N most prominent eigenface vectors. The factors obtained from the projection represent the model of person A. If we are to recognize a person from a detected image patch of a face, this image is projected into the same eigenface space and the factors of the projection are compared with the known models of persons. If the distance to a face model is close enough, the respective label is assigned. However, if no model can be found close to the computed projection, the algorithm asserts that it does not know the depicted person. The identification is illustrated as step (3) in Figure 19 with the placement of a name tag.

4.1.3 Face tracking

Although the detection of faces and the identification only need a single RGB-D snapshot of the scene it is highly desirable to increase the confidence of an identification if it can be verified over several frames or to keep track of a person even though the face is not visible at the moment. The tracking component supports exactly these purposes by assigning detections at time t_i to the previously recognized faces at time t_{i-1}. This allows increasing the confidence for an identified face with each redetection and lowering the confidence at each time when no detection can be associated with that face. In doing so, it is possible to assert face identification not before a certain threshold of confidence is reached. Furthermore, people regularly move their heads in a way that the face is not continuously

visible. Nevertheless, if we can identify the face once and still track the head in the following seconds, there is no reason why the label should not stay attached to the detected head. The tracking component facilitates exactly this behaviour and does not decrease the confidence of identification as long as the head is visible but not the face. The recognition confidence is hence only lowered when neither a head nor a face can be associated with the previous detection.

4.2 Results

Within experiments evaluating the power of the face detection component, a set of 120 faces from 10 persons and 424 non-face regions were used to train the classifiers. It has been taken care to capture different viewing angles and different mimics with the training set. The classification performance of the proposed method has been measured by processing 360 images each containing one face. The classification results are compared with the classification performance of the original Viola-Jones algorithm applied on colour images, only. The results are shown in the left diagram of Figure 20. It becomes visible that our approach has the advantage of a very low false positive rate (1.1%) compared to the 24.1% of the original approach, which means that every fourth face detection would be a false alarm. At the same time, the detection rate of our method is almost equally high as for the original classifier with 98.0%. Furthermore, as depicted in the right image of Figure 20 the computation time of our variant is only little higher than half of the original approach, which is already very fast.

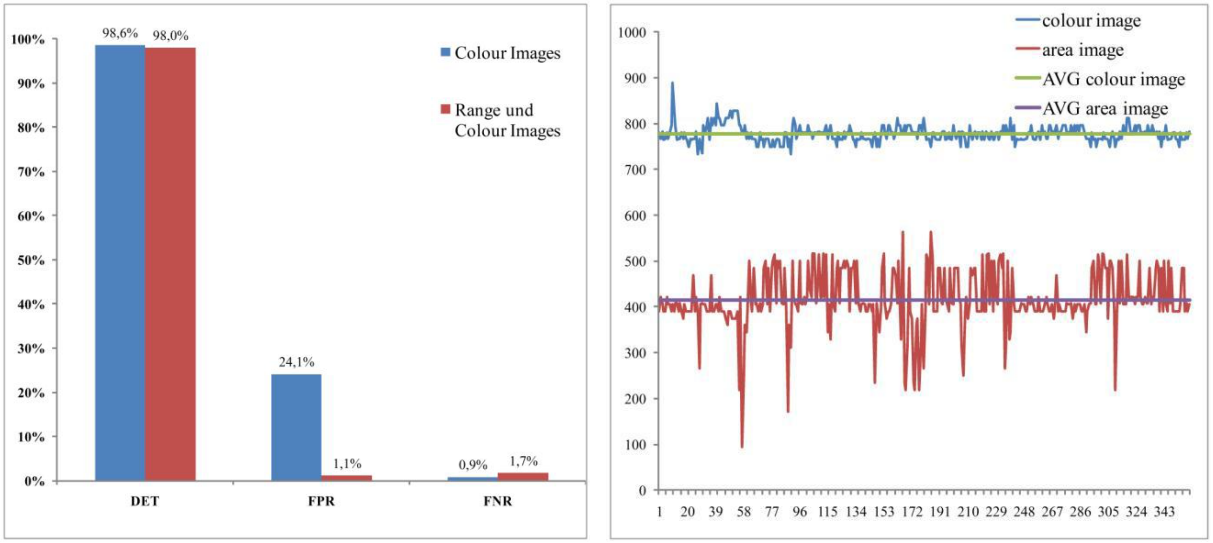


Figure 20 Comparison of detection rate, false positive rate and false negative rate for the Viola-Jones face detector in the original and our version (left) and comparsion of the computational effort of both variants (right).

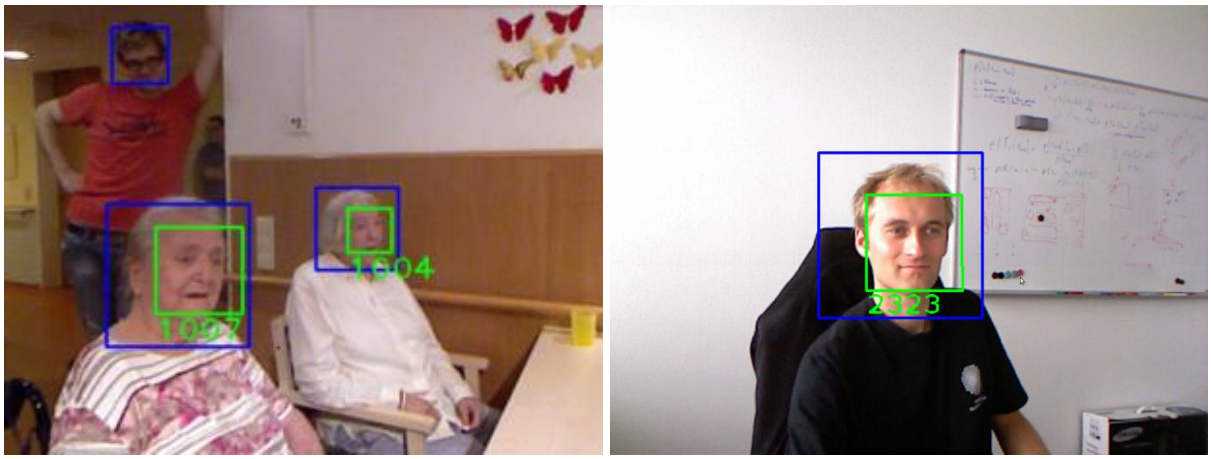


Figure 21 Examples of the application of face detection and identification to real scenes.

Besides these quantitative results, we also demonstrate the successful application of the face recognition system in Figure 21.

5. Summary of the Applied Sensor Fusion Techniques

As the description of work for WP4 puts emphasis on the fusion of sensory information from the sensors placed in the environment with sensory information generated on the mobile Care-O-bot, we like to summarize and highlight the employed strategies in this separate section.

The original idea for the implementation of sensor fusion as stated in the description of work has been identified to be impractical and of very limited use for the following two reasons. First, the continuous calibration of multiple colour cameras is neither robust nor fast to compute. The calibration lacks robustness since the lens characteristics of the objectives put on the fish-eye camera and those of the cameras mounted on Care-O-bot differ by a large extent and furthermore, both kinds of cameras observe the scene from very different perspectives so that it is fairly complicated to find a sufficiently large and stable set of unique correspondence points which is a prerequisite for successful camera calibration. Furthermore, using present people as calibration landmarks, as proposed in the description of work as an alternative, is quite unreliable as well because it is neither guaranteed that there is any person visible in the images of all cameras nor can the pose estimation of body parts of detected people be accurate enough for a good calibration. Besides, the number of correspondence points that originate from people in the scene would be fairly limited and badly-distributed over the image which is another criterion of exclusion for this approach. Concerning the computational speed of continuous calibration it is easy to see that the extraction of robust feature points and their reliable matching are procedures that take at least one second if not more on a modern computer what turns this approach inapplicable for most purposes. Moreover, the computational power on the robot is limited so that this valuable computational effort should rather be available for other tasks. The second reason for abandoning the original idea of continuous calibration between robot and environment sensors is the restricted usefulness. A high precision camera calibration between robot and fish-eye cameras in the environment could only be beneficial for object recognition; however, this application requires high-resolution images of the objects which are to be detected. Because of their wide viewing angle the ceiling cameras are not in the condition to deliver this kind of data. Hence, there is no use for the cost-intense high-precision calibration of the mobile and fixed vision sensors.

Nevertheless, the fusion of higher-level information, which originates from the robot and the cameras in the environment, is indeed of high value but requires a relative calibration with an acceptable accuracy in the range of ± 5cm. This level of accuracy is achievable with simpler technologies that are already available for the environment cameras and the robot. The strategy for data fusion that we pursue in the current work is to fuse data with the help of a map of the robot house that has a fixed coordinate system. This approach is very feasible as all data processing of any sensor delivers information with spatial coordinates that can be easily transformed into that fixed map coordinate frame. At the one hand, the cameras in the environment need to be calibrated against the fixed map coordinate system once in advance, at the other hand, the Care-O-bot is continuously localized with its laser scanners with sufficient accuracy. All the coordinate frames are then put into the transformation tree (tf) of the robot operating system ROS so that coordinates can be transformed easily and computationally cheap from any sensor into the map coordinate frame or between any two sensors.

Specifically, this strategy is implemented for object recognition and categorization in the following way. Although the fish-eye cameras are not useful for object recognition and categorization, the presented systems can deal with the coloured point cloud data from modern RGB-D sensors like the Asus Xtion Pro Live. Hence, object recognition and categorization may run on any PC that has a calibrated RGB-D device attached and deliver object localizations in the detection range of the sensor. The detections can then be transformed to the map coordinate frame and stored in the common object database of the whole system. Consequently, this type of data fusion is transparent to the source of object detections, sensors in the environment may contribute with detections in the same way as the robot.

The fusion of information is even tighter for the application case of person detection as the ceiling cameras deliver a different type of information than the cameras on the robot. The ceiling cameras have advantage of observing the whole house continuously; moving persons are thus always tracked and localized. However, the ceiling cameras do not have the perspective or the resolution for person identification, but the robot can do so with its own sensors. Again, sensor fusion is implemented successfully within the common map coordinate frame based on common spatial coordinates for both sensor's detections at given points in time. This means, we fuse the trajectory and immediate location of a tracked person (information from the ceiling camera) with identity recognized by the robot.

In conclusion, this procedure of information fusion is accomplishable with negligible computational overhead while generating a central map that collects all information from various recognition systems. Higher-level applications can directly benefit from this central map representation.

6. Conclusion and Future work

In this deliverable, we have described a method that allows for the fusion of stereo camera data with the measurements of a time-of-flight sensor. This procedure generates more accurate depth maps than a time-of-flight sensor alone whenever stereo data is available and delivers denser depth maps than those which can be obtained from stereo vision alone. The sensor fusion algorithm is used to yield a better input for our object recognition system, which is able to detect and localize previously modelled objects in cluttered scenes even if they are occluded to larger extents. The object recognition system is accompanied by an object categorization module that can detect the class of objects that are not known to the robot, yet, and hence cannot be detected with object detection. Future work will address the extension of the object recognition method to apply to untextured objects as well. Furthermore, we would like to integrate more properties for the object categorization to model classes. It is also targeted to use object categorization for automatic learning of new objects whose initial label would be related to the detected class and could be specified later by the user.

We furthermore have proposed a novel probabilistic fusion framework for the localization of humans using ambient cameras and robot-mounted Laser Range Finders. Our experiments show substantial improvements in the accuracy of the localization, thus enabling more precise interaction between robot and humans. Due to its probabilistic nature, our framework can deal with occlusions and the absence of measurements in a principled way. As a result, the localization of humans is more robust, and natural interaction becomes possible even in challenging conditions.

In our current experimental work, the orientation and the location are not considered as part of the particle, but only the location of multiple persons are sampled. But we expect the performance can be improved by incorporating robot location and orientation into particles. We plan to specifically address occlusions and missed detections in one of the sensors. We will also extend the method to use more and different sensors, including the robot-mounted Kinect camera, as well as multiple overhead cameras.

Finally, a simple method for multimodal face detection has been implemented that allows decreasing the number of false positives while keeping recall very high. In conjunction with the Eigenfaces approach for the identification of a face and the tracking over several frames, the current system can assert face labels for a low number of people with a high confidence. Upcoming work at this topic focuses on improving the robustness of the face identification approach to different lighting and higher numbers of distinguishable persons either by replacement with a different method or by better data pre-processing.

Bibliography

[1] B. Graf, “Reactive navigation of an intelligent robotic walking aid,” in *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, 2001, pp. 353–358.

[2] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, “Towards robotic assistants in nursing homes: Challenges and results,” *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 271–281, 2003.

[3] K. Wada and T. Shibata, “Living with seal robots—its sociopsychological and physiological influences on the elderly at a care house,” *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 972–980, 2007.

[4] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O’Reilly Media, Incorporated, 2008.

[5] B. Kluge, C. Kohler, and E. Prassler, “Fast and robust tracking of multiple moving objects with a laser range finder,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 2001, vol. 2, pp. 1683–1688.

[6] X. Song, J. Cui, X. Wang, H. Zhao, and H. Zha, “Tracking interacting targets with laser scanner via on-line supervised learning,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 2271–2276.

[7] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, “Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1710–1715.

[8] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Wörz, “Vision based person tracking with a mobile robot,” in *British Machine Vision Conference*, 1998, pp. 418–427.

[9] N. Bellotto and H. Hu, “Vision and laser data fusion for tracking people with a mobile robot,” in *Robotics and Biomimetics, 2006. ROBIO’06. IEEE International Conference on*, 2006, pp. 7–12.

[10] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lomker, G. Fink, and G. Sagerer, “Person tracking with a mobile robot based on multi-modal anchoring,” in *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, 2002, pp. 423–429.

[11] Z. Zivkovic and B. Krose, “Part based people detection using 2d range data and images,” in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 214–219.

[12] A. Fod, A. Howard, and M. Mataric, “A laser-based people tracker,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, 2002, vol. 3, pp. 3024–3029.

[13] N. Checka, K. W. Wilson, M. R. Siracusa, and T. Darrell, “Multiple person and speaker activity tracking with a particle filter,” in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP’04). IEEE International Conference on*, 2004, vol. 5, p. V–881.

[14] A. F. Bobick, S. S. Intille, J. W. Davis, F. Baird, C. S. Pinhanez, L. W. Campbell, Y. A. Ivanov, A. Schütte, and A. Wilson, “The KidsRoom: A perceptually-based interactive and immersive story environment,” *Presence*, vol. 8, no. 4, pp. 369–393, 1999.

[15] G. Englebienne and B. J. A. Kröse, “Fast Bayesian People Detection,” in *proceedings of the 22nd benelux AI conference (BNAIC 2010)*, 2010.

[16] O. Punska, “Bayesian approaches to multi-sensor data fusion,” *Cambridge University, Cambridge*, 1999.

[17] Y. Cai, N. de Freitas, and J. Little, “Robust visual tracking for multiple targets,” *Computer Vision–ECCV 2006*, pp. 107–118, 2006.

[18] Z. Zhang, “A flexible new technique for camera calibration,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 1330–1334, 1998.
[19] http://www.vision.caltech.edu/bouguetj/calib_doc/.
[20] M. Lindner and A. Kolb, “Lateral and depth calibration of pmd distance sensors,” in ISVC. Springer, 2006, pp. 524–533.
[21] U. Hahne and M. Alexa, “Depth imaging by combining time-of-flight and on-demand stereo,” in Dyn3D ’09: Proceedings of the DAGM 2009 Workshop on Dynamic 3D Imaging. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 70–83.
[22] S. Birchfield and C. Tomasi, “Depth discontinuities by pixel-to-pixel stereo,” International Journal of Computer Vision, vol. 35, pp. 1073–1080, 1996.
[23] H. Hirschmüller, “Stereo processing by semiglobal matching and mutual information,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 30, no. 2, pp. 328–341, feb. 2008.
[24] Motilal Agrawal, Kurt Konolige, and Morten Blas, „CenSurE: center surround extremas for realtime feature detection and matching”. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, Computer Vision ECCV 2008, volume 5305 of Lecture Notes in Computer Science, pages 102–115. Springer Berlin / Heidelberg, 2008.
[25] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, “BRIEF: binary robust independent elementary features”. In ECCV (4), pages 778–792, 2010.
[26] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, “ORB: An Efficient Alternative to SIFT or SURF,” in Proceedings of the International Conference on Computer Vision, 2011.
[27] O. Chum and J. Matas, “Matching with PROSAC - progressive sample consensus,” volume 1, pages 220–226. IEEE, 2005.
[28] Viola, P.; Jones, V., “Rapid Object Detection using a Boosted Cascade of Simple Features,” in Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2001, pp. 511-518.
[29] M. Turk and A. Pentland, “Face recognition using eigenfaces,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–591, 1991.

Appendix A

Efficient Object Categorization with the Surface-Approximation Polynomials Descriptor

Richard Bormann, Jan Fischer, Georg Arbeiter, and Alexander Verl

Fraunhofer IPA, Nobelstr. 12, 70569 Stuttgart, Germany
{richard.bormann,jan.fischer,georg.arbeiter,alexander.verl}
@ipa.fraunhofer.de
http://www.ipa.fraunhofer.de/

Abstract. Perception of object categories is a key functionality towards more versatile autonomous robots. Object categorization enables robots to understand their environments even if certain instances of objects have never been seen before. In this paper we present the novel descriptor Surface-Approximation Polynomials (SAP) that directly computes a global description on point cloud surfaces of objects based on polynomial approximations of surface cuts. This descriptor is directly applicable to point clouds captured with time-of-flight or other depth sensors without any data preprocessing or normal computation. Hence, it is generated very fast. Together with a preceding pose normalization, SAP is invariant to scale and partially invariant to rotations. We demonstrate experiments in which SAP categorizes 78 % of test objects correctly while needing only 57 ms for the computation. This way SAP is superior to GFPFH, GRSD and VFH according to both criteria.

Keywords: Object Categorization, Robot Vision, 3D Descriptor

1 Introduction

Mobile service robots which are intended to serve people in natural household environments need to retrieve rich information about their surroundings to accomplish tasks given to them. A major part of this perception problem is the recognition of objects for interaction. Although powerful object detection algorithms exist today they do not suffice for a versatile operation. Neither should every single object occurring in the environment be learned by the robot in advance nor would this even be realizable with respect to current hardware limitations. Object categorization solves this problem by identifying the class of formerly unseen object instances. Hence, the perception problem decreases significantly in size. This work focuses on the categorization of small and medium-scale rigid household objects with a simple shape.

The use of point cloud data is a good starting point towards successful object categorization in this case since many common object classes in households expose strong similarities in shape whereas texture may differ significantly. Moreover, algorithms for categorization should evaluate fast as there is limited computing power and energy supply available on a mobile robot while users expect

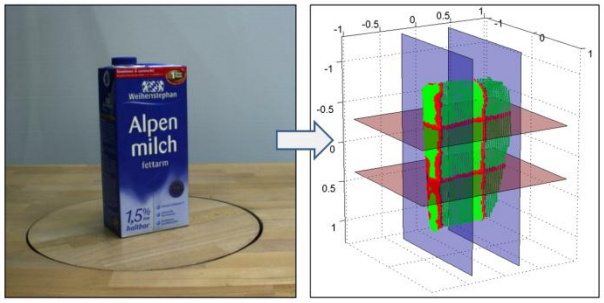


Fig. 1. Computation of the SAP descriptor: the point cloud of the milk box is scale normalized and cut with planes. The red surface points at the cuts contribute to the polynomial approximation.

fast responses. To our knowledge, the best algorithms with respect to runtime use Global Fast Point Feature Histograms (GFPFH) [1], Global Radius-based Surface Descriptors (GRSD) [2] or Viewpoint Feature Histograms (VFH) [3], that describe the properties of an object's point cloud based on point normals. They are computed within less than a second by several complex steps which also include the denoising of the data that is necessary for reliable normal estimation.

In this work, we present the novel Surface-Approximation Polynomials (SAP) descriptor, which is tailored for the goal of fast and normal-free object categorization on point cloud data. It is based on a pose normalization of the object's point cloud and the approximation of polynomials along cuts of the surface (see Figure 1). We successfully apply this descriptor directly to noisy and unprocessed point cloud data generated by SwissRanger SR4000 and PMD CamCube depth sensors. We will show that this descriptor allows for a categorization performance of 78 % correctly categorized objects on a dataset of 14 classes. This is superior to GFPFH, GRSD and VFH by 9.5 % and more. At the same time the SAP descriptor can be computed within only 57 ms which is faster than each of the three aforementioned descriptors. We also demonstrate the scale invariance of SAP and the partial rotation invariance to tilt and pan. Finally, we provide an outlook for the addition of rotational invariance around the camera axis.

The remainder of this paper is structured as follows: Section 2 provides a literature review on object categorization techniques in different contexts and Section 3 explains the algorithm of the SAP descriptor as well as the categorization framework. In Section 4 we present various experiments which demonstrate the performance and properties of the SAP descriptor. We conclude with a summary and an outlook on future work in Section 5.

2 Related Work

Object categorization is a topic of high interest in the computer vision and the robotics community. However, both areas are quite different concerning their data, constraints and objectives. Computer vision approaches usually rely on plain color images and aim at tasks like image retrieval. A good overview over current methods is provided by Galleguillos and Belongie [4]. Modern techniques in this area mostly use derivatives of Bag-of-Words models (BoW) on local descriptors with Support Vector Machines (SVM) as classifier and attain image categorization results of around 70% on the Caltech-101 dataset [5]. While computer vision research rather focused on categorizing large amounts of classes recently [6], high precision in the predictions is actually more important to robotics. We believe that the use of depth sensors supports this goal since 3D shape is often very characteristic for object classes.

Hence, we have to deal with 3-dimensional data from our objects of interest. Similar to local 2D images features, many local 3D descriptors have been devised. Some popular examples are Spin Images [7], Shape Index [8], 3D SURF [9] and SHOT [10]. For a broader overview on local 3D descriptors and a comparison regarding the object classification task we refer to Knopp et al. [11], who present orientation invariant 3D object categorization based on Bag-of-Words from 3D SURF features and a Hough Transform voting method. They evaluate their algorithm on high resolution full 3D models and obtain 95.5% accuracy for 8 categories. However, the processing for one model takes 20.66s on the 3D SURF features and even more on other local features. Likewise, Toldo et al. [12] present up to 100% accuracy on synthetic data for 6 classes while having computation times around 50s. They apply Bag-of-Words on segmented parts of the object using the Shape Index descriptor and classify with an SVM. Examples of global features for the shape matching task are Shape Distributions [13] and Spherical Harmonics [14]. Although these approaches provide very high accuracies, they do not meet several aspects in robotics: first, highly resoluted 3D meshes are normally not available with current 3D sensors attached to robots. Moreover, the object's surface is regularly captured and categorized from a single view instead of a full 360° model. Finally, computation times should be as low as possible but definitely not in the order of seconds or higher. Therefore, we will propose a 3D descriptor for categorizing single shot object surfaces which is very fast to compute and robust to low quality sensory input yet powerful enough for high categorization rates. Pu et al. [15] present an approach for 3D model retrieval that computes the global descriptor from various slices through the 3D mesh. Their key idea is similar to our approach, however, their work focuses on instance retrieval of highly resoluted full 3D models.

Related work which addresses the issues of robotics is available with unsupervised and supervised category learning. The first problem was investigated by Endres et al. [16] who collect histograms of discretized spin images of objects segmented from 3D laser scanner data and cluster them with Latent Dirichlet Allocation (LDA) in an unsupervised fashion. The resulting classes correspond to balloons, boxes, chairs, swivel chairs and humans with 90.38% accuracy. Labels

are assigned within 0.5 s. However, as we like to access the semantic information included in the class labels, we need to use a supervised learning approach.

Based on a linear SVM classifier Bo et al. [17] propose color and depth kernel descriptors which can categorize 86.2 % of the test objects into 51 classes. Further popular descriptors are Surflet-pair-relation histograms [18], Global Fast Point Feature Histograms (GFPFH) [1], which are very similar to the first, and Global Radius-based Surface Descriptors (GRSD) [2]. GFPFH builds on FPFH which computes the sum of angle histograms between angles of normals of each surface point and its neighboring points. These histograms are classified as geometric primitives. Histograms over the occurence of geometric primitives along lines between any two voxels of the object's point cloud yield the GFPFH feature. Its accuracy on a 4 class problem is 96.69% with a computation time of below a second. The GRSD descriptor is composed similarly to GFPFH descriptor from local RSD features, which basically represent the local minimum and maximum curvature around a point. It can categorize 85% of unseen objects correctly into six classes needing around 0.2s for each computation. Another descriptor that is very similar to GFPFH but that also encodes the viewpoint at the visible object surface is the Viewpoint Feature Histogram (VFH) [3]. VFH includes the camera axis in the computation of FPFH histograms to establish viewpoint dependent signatures for the trained objects.

Our descriptor is different from these methods insofar as it does not rely on normal computations and local feature representations. Instead, we construct the descriptor directly in a global fashion on the point cloud data and hence avoid the data preparation and normal computation which can consume quite some time if no GPU is available. This way SAP can be computed very fast within 57 ms. The next section details our approach.

3 Methods

This section describes the concepts of the SAP descriptor and the employed framework for categorizing unknown objects. We present a simple and easy-to-compute descriptor on point cloud data of objects. The SAP descriptor is specially designed for the needs of efficient object categorization on a robot being compact, scale invariant and having little computational demands.

Within the categorization framework, we approach the following two kinds of problems provided that the robot can obtain some descriptor for every object in its surroundings:

1. The robot must find objects of a certain category k in its environment. It will label descriptors as either being members of category k or not. This is a binary classification problem.
2. The robot has to assign a category label to every object found in its surroundings. This problem is essentially a classification problem with multiple classes.

The next section starts with an explanation how the needed data is acquired and preprocessed.

3.1 Data Acquisition and Segmentation

The SAP descriptor solely needs a single shot point cloud $\mathcal{P}$ of the object of interest segmented from the scene. We test the descriptor on two databases captured with a SwissRanger SR4000 sensor and a PMD CamCube, respectively. We did not use the Kinect sensor because it was not yet on the market when the databases were collected. The Swissranger 4000 has a resolution of 176x144 pixels and a depth accuracy of around 1% of the measured coordinates. The PMD CamCube can capture depth images with a slightly higher resolution of 204x204 pixels. Both databases will be introduced in Section 4.1.

For recording, the objects were placed on a rotary disc embedded into a table surface. We placed the depth camera approximately 1 m away from the object center and captured depth images of the objects from a slightly elevated viewing angle in front of a mostly homogeneous background (see Fig. 1). By rotating each object incrementally on the rotary table we recorded point clouds from all sides that can typically be observed by the robot when searching for objects on surfaces of the height of a table. This way, we captured 36 to 72 views per object.

For computing the SAP descriptor the obtained point clouds need to be segmented. We require that objects can only be placed on top of a plane, for example a table as in the case of the database recordings. We use a parametric model fitting algorithm from the PCL library [19] for the iterative estimation and removal of larger planes. This method searches for the plane parameters a, b, c, d and points (x, y, z) satisfying the corresponding plane equation

$$ax + by + cz + d = 0 \quad . \tag{1}$$

Sample points are drawn from the point cloud and associated with a plane according to the RANSAC [20] algorithm. RANSAC iteratively draws triples of points, solves the plane equation and searches for further points supporting this plane. The algorithm terminates when the plane with most supporting points is found with high probability. The volume above this plane is considered as the space of potential objects. Multiple objects inside this volume are separated by Euclidean clustering so that we can only examine simple scenes with this approach. Nevertheless, providing a fancy segmentation algorithm that works in arbitrary situations is beyond the scope of this paper. We refer to the literature for approaches that work properly in many cases [1, 21, 22]. In the following we describe the SAP descriptor and the categorization methods.

3.2 Surface-Approximation Polynomials Descriptor

The underlying idea of the SAP descriptor is to generate a scale normalized view of an object's surface, cut it with 2-dimensional planar subspaces perpendicular to the camera plane and approximate the projections of the cut with polynomials of even order. For a better understanding of the idea and the single steps we refer to Figure 2. To receive a scale-invariant description, the pose of point cloud $\mathcal{P}$ is normalized by computing the centroid $\mathbf{m}$ of $\mathcal{P}$ and its rescaling factor

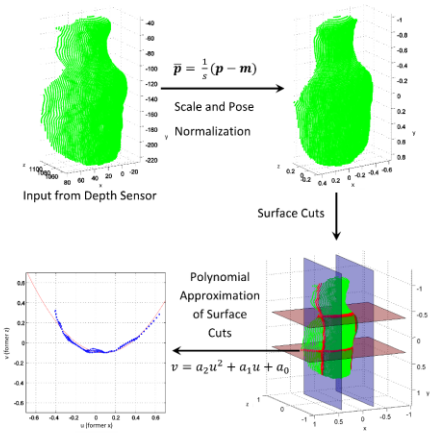


Fig. 2. Computation scheme of the SAP descriptor. The upper left image shows the input point cloud. Then, pose and scale normalization is applied, surface cuts are extracted and finally approximated with a polynomial .

$s = \max_{\mathbf{p}\in\mathcal{P}}\{|p_x - m_x|, |p_y - m_y|\}$, where p_x ans p_y are the x- and y-coordinates of point $\mathbf{p}$ and m_x and m_y are components of the centroid $\mathbf{m}$. The coordinate system is defined with the z-axis pointing from the camera center towards the scene. The x-axis runs horizontal in the image plane, the y-axis is vertical. Every point $\mathbf{p}$ is then translated to shift the point cloud's center into the origin and scaled by $\frac{1}{s}$. Due to the computation of the rescaling factor s, the $\bar{p}_x$- and $\bar{p}_y$-coordinates of each point $\bar{\mathbf{p}}$ of the transformed point cloud $\bar{\mathcal{P}}$ fall into the range $[-1, 1]$. In summary, the transformation is:

$$\bar{\mathbf{p}} = \frac{1}{s} \cdot (\mathbf{p} - \mathbf{m}) \quad , \tag{2}$$

$$\mathbf{m} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}\in\mathcal{P}} p \quad , \tag{3}$$

$$\mathbf{s} = \max_{\mathbf{p}\in\mathcal{P}}\{|p_x - m_x|, |p_y - m_y|\} \quad . \tag{4}$$

Translating the center of the point cloud to the origin ensures translation invariance with respect to the coordinate system of the depth sensor while the scaling operation effects that the point cloud is resized to a common scale.

After normalization, we sample points from the surface which are approximately located on straight lines parallel to the x- and y-axes. This can be thought of as picking points which approximately lie within cutting $x - z$-planes ($y =$ const.) or $y - z$-planes ($x =$ const.). Specifically, we define to sample points for

n_x lines parallel to the x-axis and n_y lines parallel to the y-axis. These lines are equally spaced within the [-1, 1] interval. The cutting planes are illustrated in the lower right image of Figure 2 in rose (y = const.) and purple (x = const.) color. The points associated with cuts are displayed in red.

Following, we approximate the points assigned to each cut with a polynomial of order n_p which essentially comprises the information coded in the point locations into $n_p + 1$ parameters of the polynomial. The polynomials are computed with a standard regression approach: suppose that the coordinates of the points projected into the 2-dimensional subspace are renamed from x or y to u and from z to v. We aim to find the parameters $\mathbf{a} = (a_0, a_1, \ldots, a_{n_p})^{\mathrm{T}}$ of the polynomial $v = a_0 + a_1 u + \ldots + a_{n_p} u^{n_p}$. If we have L sample points on the cutting line, with $L \geq n_p + 1$, we obtain L constraints that can be rephrased in vector notation:

$$v_i = a_0 + a_1 u_i + \ldots + a_{n_p} u_i^{n_p} \ , \ \forall i = 1, \ldots, L \ , \tag{5}$$

$$\mathbf{v} = \mathbf{U} \cdot \mathbf{a} \ , \tag{6}$$

$$\mathbf{v} = [v_1, \ldots, v_L]^{\mathrm{T}} \ , \tag{7}$$

$$\mathbf{U} = \begin{bmatrix} 1 & u_1 & \ldots & u_1^{n_p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & u_L & \ldots & u_L^{n_p} \end{bmatrix} \ . \tag{8}$$

We can easily generate vector $\mathbf{v}$ and matrix $\mathbf{U}$ from the L point samples and solve the linear regression problem with a standard approach, e.g. Singular Value Decomposition. If the available point data is insufficient, we assign zeros to $\mathbf{a}$.

Finally, we concatenate the parameter vectors ${}^{i}\mathbf{a}^{\mathrm{T}}, i = 1, \ldots, n_x + n_y$, of the n_x and n_y approximated polynomials into one vector $\hat{\mathbf{c}} = [{}^{1}\mathbf{a}^{\mathrm{T}}, \ldots, {}^{\mathbf{n_x}+\mathbf{n_y}}\mathbf{a}^{\mathrm{T}}]$. The final SAP descriptor

$$\mathbf{c} = \left[\frac{\lambda_1}{\gamma}, \frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1}, \hat{\mathbf{c}} \right] \tag{9}$$

consists of the parameter vector $\hat{\mathbf{c}}$ and some general size information of the object. The eigenvalues λ_1, λ_2 and λ_3 (in descending order) are obtained from a Principal Component Analysis on point cloud $\mathcal{P}$. They encode the size of the object in the three principal directions. λ_2 and λ_3 are expressed relative to λ_1 to encode the relations between the side lengths. λ_1 instead is saved with its full magnitude, except for the constant scaling factor γ, so that the broad information about the object's absolute size is preserved. The constant γ is solely necessary to rescale the entry of λ_1 approximately into the range $[0, 1]$.

The computational complexity for computing the SAP descriptor is $\mathcal{O}(|\mathcal{P}|)$ where $|\mathcal{P}|$ is the number of points in the point cloud. In detail, the single computations have the following complexities:

scale and pose normalization:	$\mathcal{O}(\lvert\mathcal{P}\rvert)$,	(10)
point assignment to cuts:	$\mathcal{O}((n_x + n_y) \cdot \lvert\mathcal{P}\rvert)$,	(11)
SVD for polynomial approximations:	$\mathcal{O}(L_{max}(n_p + 1)^2(n_x + n_y))$,	(12)
PCA for eigenvalues:	$\mathcal{O}(3^2 \cdot \lvert\mathcal{P}\rvert)$,	(13)

where $L_{max} \ll |\mathcal{P}|$ is the largest number L of points on a cutting line.

In summary, the SAP descriptor is basically a collection of parameters from polynomials fitted into the normalized surface of an object and three size variables. The three parameters that can be tuned are the numbers of cuts n_x and n_y parallel to the x- and y-directions as well as the order n_p of the polynomial.

3.3 Category Learning

After the computation of descriptors we have to apply a method that separates the different object categories by finding the characteristics in their descriptors. Recall that we aim at enabling the algorithm for two tasks: the search for an object instance of a certain class, which is a binary one-against-all classification problem, and the detection of the category of an unknown object, which is a multi-class decision problem. Therefore, the classification algorithm is based on N binary one-against-all classifiers that distinguish each of the N object classes against the others. A straightforward extension of these binary classifiers for multi-class distinction without learning a new classifier will be discussed in Section 3.4.

We use Random Forest classifiers [23] for the N binary one-against-all classification problems because they compared favorably to Support Vector Machines [24] and K-Nearest Neighbors as we will see in Section 4.2. Each Random Forest is trained in regression mode assigning a 1 as desired output if the provided descriptor is from the category the classifier is trained on and assigning a 0 if the sample originates from any other class. Queried with a sample $\mathbf{x}$, the Random Forest for class k will output a number $w_k = w_k(\mathbf{x})$ between 0 and 1. We define a decision boundary $\theta_i \in [0,1], i = 1, \ldots, N$, for each of the N Random Forest classifiers which allows to interpret the outputs w_k: we say that classifier k asserts that sample $\mathbf{x}$ belongs to its class if $w_k \geq \theta_k$. Dependent on the value of the decision boundary θ_k we obtain different results regarding

$$\text{the true positive rate } \rho_{\text{tpr}} \text{ (recall):} \qquad \rho_{\text{tpr}} = \frac{h_{\text{tp}}}{h_{\text{tp}} + h_{\text{fn}}} \quad , \tag{14}$$

$$\text{the false positive rate } \rho_{\text{fpr}}\text{:} \qquad \rho_{\text{fpr}} = \frac{h_{\text{fp}}}{h_{\text{fp}} + h_{\text{tn}}} \quad \text{and} \tag{15}$$

$$\text{the precision } \rho_{\text{pr}}\text{:} \qquad \rho_{\text{pr}} = \frac{h_{\text{tp}}}{h_{\text{tp}} + h_{\text{fp}}} \tag{16}$$

of the classifier. Here we denote the number of true positive classifications h_{tp}. These are positive samples which were actually classified positive. The number of false positive classifications is called h_{fp}, the number of false negative classifications h_{fn} and the number of true negative classifications h_{tn}.

We set each θ_i to the optimal value suggested by the performance on the training set according to the following measure:

$$\theta_i = \arg\min_{t \in [0,1]} d_{\text{ROC}}\left(\rho_{\text{tpr},i}(t), \rho_{\text{fpr},i}(t)\right) \tag{17}$$

$$d_{\text{ROC}}(\rho_{\text{tpr}}, \rho_{\text{fpr}}) = \sqrt{(1-\rho_{\text{tpr}})^2 + \rho_{\text{fpr}}^2}. \tag{18}$$

Hence, we are searching for the θ_i that minimize the distance of the ROC plot to the optimal point with $\rho_{\text{tpr},i} = 1$ and $\rho_{\text{fpr},i} = 0$ for each class i. If we search for an object of class i and the corresponding classifier outputs a value greater or equal than θ_i when provided with a descriptor $\mathbf{x}$ we consider the sample as belonging to the queried class i.

However, this method does not work properly if the category of a sample has to be determined since in some cases more than one binary classifier might assert that sample $\mathbf{x}$ belongs to its class. Therefore, we convert the output w_k of each classifier $k = 1, \ldots, N$ into a likelihood $L(a_k|\mathbf{x})$ of descriptor $\mathbf{x}$ belonging to class k. We define

$$L(a_k|\mathbf{x}) = \frac{e^{\alpha \cdot m_k(\mathbf{x})}}{e^{\alpha \cdot m_k(\mathbf{x})} + e^{-\alpha \cdot m_k(\mathbf{x})}} \quad , \tag{19}$$

$$m_k(\mathbf{x}) = \begin{cases} \frac{w_k(\mathbf{x}) - \theta_k}{1 - \theta_k} \ , & w_k(\mathbf{x}) \geq \theta_k \\ \frac{w_k(\mathbf{x}) - \theta_k}{\theta_k} \ , & w_k(\mathbf{x}) < \theta_k \end{cases} \quad \forall k = 1, \ldots, N \quad . \tag{20}$$

Mapping $m_k(\mathbf{x})$ maps the output value $w_k(\mathbf{x})$ piecewise linearly from the range $[0, \ldots, \theta_k, \ldots, 1]$ to $[-1, \ldots, 0, \ldots, 1]$ where the decision threshold θ_k is mapped to 0. Then for all positive decisions of classifier k, that is $w_k(\mathbf{x}) \geq \theta_k$, it holds that $m_k(\mathbf{x}) \geq 0$ and for all negative decisions we have $m_k(\mathbf{x}) < 0$. Equation (19) is inspired by the conversion of the tree ensemble output to a probability in AdaBoost [25]. This function maps negative values of $m_k(\mathbf{x})$ to low probabilities, positive values of $m_k(\mathbf{x})$ to high probabilities and assigns uncertainty, that is 0.5, if $m_k(\mathbf{x})$ equals 0.Whereas the potentially different values of the decision thresholds $\theta_k, k = 1, \ldots, N$, prevent a direct comparison of the outputs $w_k(\mathbf{x})$, the conversion as shown renders the certainties $L(a_k|\mathbf{x})$ of different classifiers comparable. Parameter $\alpha > 0$ in equation (19) is a scale factor which defines the slope of the mapping and the minimally $\frac{e^{-\alpha}}{e^{-\alpha}+e^{\alpha}}$ and maximally $\frac{e^{\alpha}}{e^{\alpha}+e^{-\alpha}}$ possible probability. α must be carefully adjusted to distribute the occurring $w_k(\mathbf{x})$ over the whole range of probabilities.

The next section details how we can obtain a category decision from the likelihoods of the binary classifiers with respect to their reliabilities.

3.4 Extension for Multi-class Categorization

Most of the popular classification methods of machine learning are essentially distinguishing between two classes. Several approaches exist for a multi-class extension of those binary classifiers, like one-against-all or one-against-one schemes. In one-against-all solutions, there exists one basic classifier for each class which discriminates a single class against the remainder of classes. The decision for a certain class is found either by choosing the result of the classifier with the highest certainty or by constructing a decision cascade beginning with the strongest classifier [26]. Another approach is the one-against-one scheme which contains a basic classifier for each pair of classes. The decision is determined by collecting the votes of all these classifiers. For the reasons discussed in Section 3.3 we

decided for the one-against-all scheme whose size only grows linearly with the number of classes.

Furthermore, each of the N one-against-all classifiers outputs a certainty $L(a_k|\mathbf{x})$ that descriptor $\mathbf{x}$ belongs to class k as explained in Section 3.3. One could be tempted to classify a descriptor with the respective class of the classifier yielding the highest certainty. However, this can easily lead to wrongly biased multiclass decisions since the individual reliabilities of the classifiers are not considered. A simple example illustrates the problem: Suppose the classifier for class j is outputting a probability of 1.0 for every sample so that we would always choose it. Nevertheless, this classifier is only correct in those few cases when the sample is indeed from class j. That is visible on the low precision of classifier j, which indicates that only few of the positive outputs are indeed correct. The multiclass performance of this classifier would not be better than guessing. This finding suggests that we have to incorporate the precision of each classifier, which is a measure of reliability.

Consequently, we apply a probabilistic decision scheme which outputs a probability distribution for the class to choose. The proposed scheme incorporates the different reliabilities of the classifiers in a principled way. Assume there are N different binary classifiers, Random Forests in our example, each for one of the N classes of objects. Presented with a data sample $\mathbf{x}$, their outputs are the likelihoods $L(a_1|\mathbf{x}), \ldots, L(a_N|\mathbf{x})$, where $L(a_k|\mathbf{x})$ stands for the certainty of classifier k that $\mathbf{x}$ is a sample of its class (see Section 3.3). Applying the formula for total probability, the probability that sample $\mathbf{x}$ is a descriptor of class o_i can be expressed as

$$p(o_i|\mathbf{x}) = \sum_{k=1}^{N} p(o_i|a_k, \mathbf{x}) p(a_k|\mathbf{x}) \quad . \tag{21}$$

We approximate

$$p(o_i|a_k, \mathbf{x}) \approx p(o_i|a_k) = \frac{p(a_k|o_i)p(o_i)}{p(a_k)} \tag{22}$$

The decision accuracy term $p(a_k|o_i)$, which describes the probability that the binary classifier of class k considers a sample positive when it is actually a sample of class i, is determined from cross-validating the binary classifiers while the class frequency prior $p(o_i)$ is set to a uniform distribution but could also be obtained from the training dataset. Having these distributions we can calculate $p(a_k) = \sum_{i=1}^{N} p(a_k|o_i)p(o_i)$. The prior $p(a_k|\mathbf{x}) = \beta L(a_k|\mathbf{x})$ is proportional to the certainties $L(a_k|\mathbf{x})$. However, as we are only interested in the object class $\hat{o} = \arg\max_{o_i, i=1,\ldots,N} p(o_i|\mathbf{x})$ with highest probability, we do not have to compute β. In summary, we determine the multiclass decision $\hat{o}$ as

$$\hat{o} = \arg \max_{o_i, i=1,\ldots,N} \sum_{k=1}^{N} \frac{p(a_k|o_i)p(o_i)}{p(a_k)} L(a_k|\mathbf{x}) \quad . \tag{23}$$

This approach allows to weight the probabilities with which the classifiers are voting for their class with the reliabilities of these classifiers.

Table 1. Number of objects captured from each class in the IPA-1 and IPA-2 database.

IPA-1		IPA-2			
class	# objects	class	# objects	class	# objects
ball	3	binder	10	dishliquid	9
book	10	book	11	drink carton	9
bottle	10	bottle	10	computer mouse	8
coffeepot	7	can	10	pen	10
cup	10	coffeepot	10	scissors	5
drink carton	4	cup	10	screen	10
flowerpot	7	dishes	10	silverware	29
plush toy	3				
toy car	3				

4 Evaluation

This section evaluates the categorization performance of the SAP descriptor and compares the results with other approaches. We additionally demonstrate the properties of the SAP descriptor concerning computation time and invariance to camera distance and rotations. All experiments measuring categorization rates were carried out with a 10-fold randomized leave-out-one cross-validation in which we left one randomly chosen object for the test set that did not occur within the training data. Categorization rates are the ratios of correctly classified views of the test objects to the total number of views from test objects. Categorization rates are computed individually for each class and reported as the average over all classes. All results were determined using the following databases.

4.1 Datasets

We recorded two datasets with 9 and 14 classes of household objects, respectively, according to the procedure described in Sec. 3.1. The dataset with 57 objects from 9 classes is called set IPA-1. For this set, each object was placed on a rotary table and captured 72 times yielding consecutive views in 5° steps. The Swissranger depth camera was mounted at the height of a robot viewing slightly downwards onto the table. The average number of points per segmented object is 6144. The left column of Table 1 summarizes the classes contained in the database as well as the number of object instances captured.

The second dataset, named IPA-2, was captured with a PMD CamCube and contains 36 views per object. The average number of points per object is 26491. The middle and right columns of Table 1 provide an overview over the distribution of the 151 objects into the 14 classes. A detailed description of the IPA-2 object database can be found in [27]. This set is publicly available at `http://www.kyb.mpg.de/nc/employee/details/browatbn.html`.

4.2 Results

We compare the performance of our categorization framework and the SAP descriptor on the larger IPA-2 database with the Global Radius-based Surface Descriptor (GRSD), the Global Fast Point Feature Histogram (GFPFH) descriptor and the Viewpoint Feature Histograms (VFH) descriptor, which have been applied to similar tasks. For computing the latter three descriptors, we used the implementations of the PCL library [19] and implemented the supplementary preprocessing according to the descriptions in the original papers [2, 1, 3]. Additionally, the point clusters of the objects were centered in front of the camera before any data processing so that no random deviations of the camera viewpoint could diminish the descriptive power of these descriptors. We found that this measure increased the recall rate of VFH by almost 2 %, for example. Then, the point clouds were downsampled with a voxel filter of leaf size 1.5 cm (GRSD, GFPFH) or 0.5 cm (VFH), respectively, to speed up the following normal estimation. Finally, we called the functions which compute the respective descriptors from the point cloud and its normals. For accumulating the local RSD features, we utilized the GFPFH-function as this is almost exactly the algorithm which also computes GRSD and because no other implementation was available. The labels for each voxel were estimated with the `getSimpleType()` function, which appears to realize the method described in the original paper. As we do not have point-wise labels in our datasets, we could not train the voxel labels w.r.t. to the FPFH descriptors as explained in [1]. Instead, we clustered the FPFH descriptors with k-means into 5 classes and used these classes as labels for the GFPFH computation. Using more clusters resulted in a decrease in performance.

In the following, the naming scheme for the variants of SAP descriptors is SAP-n_x-n_y-n_p, where n_x, n_y describe the number of cuts along the x- and y-coordinate axes and n_p denotes the degree of the approximating polynomial. The multi-class classification rates and their standard deviations on dataset IPA-2 are compiled in Table 2 for all tested descriptors. For the sake of completeness we also cite the results from Browatzki et al. [27] who conducted similar experiments on this database. We can observe that the SAP descriptor appears to be more powerful for categorization problems than VFH, GRSD or GFPFH with an increase in multi-class categorization performance of 9.5 % to 23.5 %. SAP also performs 5 % better than the best of the four descriptors tested in [27]. For SAP, GRSD and GFPFH we report the results when using the categorization framework of this paper whereas the VFH descriptors are categorized with a Support Vector Machine.

Table 3 summarizes the classification rates of these descriptors with respect to the used classifier. We compare a K-Nearest Neighbors (KNN) classifier with $k = 1$, a multi-class Support Vector Machine (SVM)[1] and our proposed framework with Random Forests. It shows that our method can almost always attain the top performance. Only with the VFH descriptor the multi-class SVM is 1.1 % better.

[1] Both classifiers as implemented in OpenCV [28], however the SVM originates from libsvm [29].

Table 2. Comparison of several descriptors regarding multi-class categorization performance, average computation time per view and average throughput in points per second. The evaluation was carried out on the IPA-2 database.

Descriptor	Performance	Computation Time	Throughput
Shape Distributions [27]	25.4 %	31 ms	~ 855 000 pts/s
Shape Index [27]	34.6 %	78 ms	~ 339 000 pts/s
Shape Context 3D [27]	55.2 %	234 ms	~ 113 000 pts/s
Depth Buffer [27]	72.9 %	**16 ms**	**~ 1 656 000 pts/s**
GFPFH	54.4±6.2 %	921 ms	28 928 pts/s
GRSD	56.1±5.8 %	957 ms	27 841 pts/s
VFH	68.4±6.7 %	93 ms	205 883 pts/s
SAP-7-7-2	**77.9±5.5 %**	57 ms	338 534 pts/s

Table 3. Comparison of different classifiers for categorizing the objects from the IPA-2 database with several descriptors.

Classifier	KNN ($k = 1$)	SVM	Random Forests
Performance with SAP-7-7-2	55.4 %	45.5 %	**77.9 %**
Performance with VFH	60.4 %	**68.4 %**	67.3 %
Performance with GRSD	56.0 %	51.5 %	**56.1 %**
Performance with GFPFH	51.6 %	46.8 %	**54.4 %**

The results for KNN do not improve when the number of considered neighbors k is increased. The SVM uses a one-against-one multi-class extension. It was trained with automatic parameter tuning through a 10-fold cross-validation. The margin between the three classifiers is by far the largest with the SAP-7-7-2 descriptor. The reason for this effect is the inhomogeneous descriptor which consists of an absolute size measure, two relative size measures and parameters of polynomials. Random Forests do not require the input data to be normalized to a common magnitude. Consequently, they work well with the data we provide. However, we would need to construct an adequate metric for KNN or SVM since these classifiers rely on normalized data.

Besides the categorization performance we also report average computation times and throughput for the descriptor computation from a single view, including necessary preprocessing. To avoid biased representations the computation time for the classifier is not included. Nevertheless, our Random Forests-based approach evaluates very fast with only 8 ms on average for the SAP-7-7-2 descriptor (45-dimensional). The results for the descriptors Shape Distributions, Shape Index, Shape Context and Depth Buffer were determined by Browatzki et al. [27] and are obtained on a 3 GHz DualCore machine with 2 GB RAM. We estimated the throughput for these values. All code was written in C++. The computation times for GFPFH, GRSD, VFH and SAP were determined on a mobile Intel I7 2.8 GHz Processor using only a single core. It shows that the

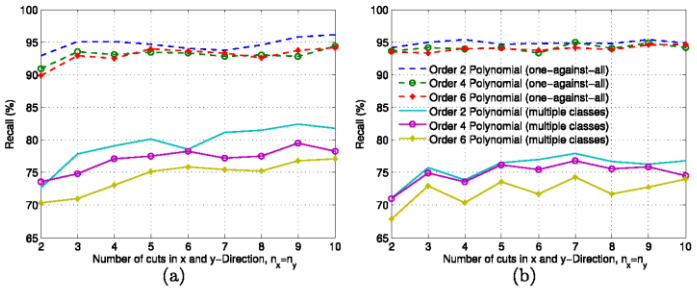


Fig. 3. Comparison of different configurations of the SAP descriptor with varying numbers of surface cuts and degrees of the approximating polynomials (a) on IPA-1 database and (b) on IPA-2 database.

computation time of the SAP descriptor is almost four times slower than the top performer Depth Buffer, however at the gain of 5% better categorization rates. The runtime of SAP allows to compute the descriptor with over 17 Hz. That is SAP could classify up to 17 objects in a scene within one second. VFH suffers from a slightly longer computation time because of the preceding normal computation. GFPFH and GRSD can only categorize one object per second.

4.3 Parameters and Properties of the SAP Descriptor

As explained in Section 3.2 the SAP descriptor has three parameters which are supposed to have a significant influence on its descriptive power and cannot be trivially chosen. Therefore, we examine the categorization performance with respect to the numbers of cuts n_x and n_y parallel to the x- and y-axes and with respect to the degree n_p of the approximating polynomial. The number of cuts is always kept equal for both dimensions, that is $n_x = n_y$, since objects can have the same extent in both directions. It is not possible to reduce n_y for slim objects because the employed classifiers expect descriptors of fixed length. Furthermore, it is not suitable to divide a constant total number of cuts to variable numbers n_x and n_y either, as this approach means comparing parts of the descriptors which contain spatially unrelated data for different objects or at least for objects from different classes.

The dependency of classification performance on these parameters is illustrated in Figure 3(a) for the IPA-1 dataset and in Figure 3(b) for the IPA-2 dataset. Both diagrams report results for the binary classification problem of separating one object class against the others as well as for the multi-class labeling task where each object view has to be assigned one of the class labels. The general trend that polynomials with higher degree n_p cause a lower categorization

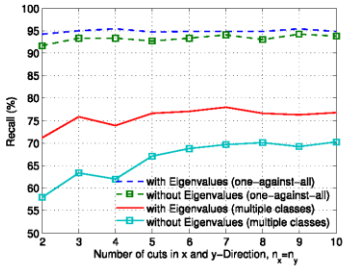


Fig. 4. Recall rates for the SAP descriptor with varying numbers of surface cuts and a polynomial degree of 2 when the descriptor either contains the size information from the PCA eigenvalues. Evaluated on the IPA-2 database.

performance becomes evident in the binary and multi-class case. Manual inspection of the descriptors suggests that higher order polynomials are less stable and tend to model the noise from the sensor. As to expect, increasing the number of cuts allows the descriptors to capture more details and improves the categorization results. Furthermore, dataset IPA-1 contains less object classes than IPA-2 and consequently, the recall rates are higher with fewer classes. Finally, we can conclude that SAP-7-7-2 is a reasonable choice considering the categorization performance on both datasets and the computation time. Therefore, we selected this configuration as standard for the comparison to other descriptors and within the following experiments revealing further properties of SAP.

Besides a suitable configuration we also need to know whether it makes sense to concatenate the absolute and relative size information from PCA with the polynomial parameters to form the SAP descriptor. Thus, we analyzed both components of the SAP descriptor alone. We found that using only the three values of the size component the categorization performance decreases to 62.2 %. On the other hand, there is a similarly significant drop in the recall rates if only the polynomial parameters appear in the descriptor as Figure 4 indicates. Consequently, combining both cues in the SAP descriptor proves to be resonable.

Next, the influence of the parameters of the SAP descriptor on the computation time shall be dissected. Figure 5(a) displays the average computation times for the computation of a SAP descriptor from one object view for increasing numbers of n_x, n_y and n_p. As the theoretical analysis in Section 3.2 predicts there is a linear increase in computation time with rising numbers of surface cuts. However, the influence of the degree of the polynomials is less visible because of the very small differences in computation time. We therefore suppose, that the SVD for polynomial fitting has a significantly lower impact on the computation time than the effort for assigning points to the cuts, which is also linear with $n_x + n_y$ but independent of n_p. We also display statistics about processed model

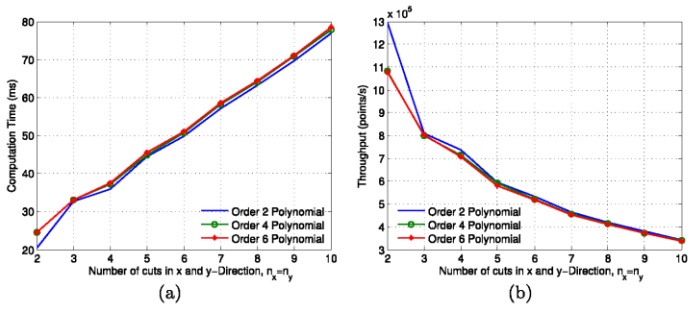


Fig. 5. Dependency of (a) computation time and (b) throughput of the SAP descriptor for increasing numbers of surface cuts and polynomial degrees measured on set IPA-2.

points per second in Figure 5(b) to provide a measure which is independent of the number of points per object view. The general trend coincides with the computation time result as throughput behaves essentially reciprocal to computation time: the more surface cuts and the higher the degree of the polynomials, the lower the throughput.

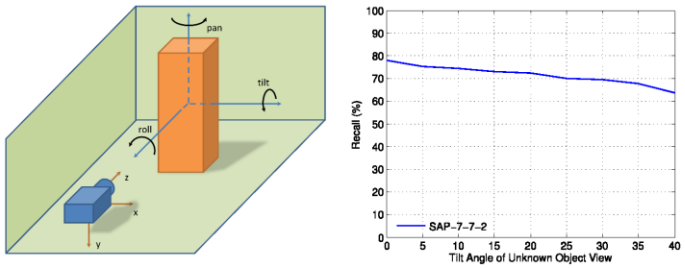
4.4 Scanning Distance and Rotation Invariance

The last part of the evaluation deals with the robustness or invariance of the SAP descriptor against common transformations of objects. First, we analyze the invariance against scale changes. This happens when the camera moves closer to the object or farther away from it. Although the range sensors still capture the real size of the objects because they provide metric measurements, the sampling density of the point clouds decreases quadratically with the distance to the camera. A consequence is that noisy pixels can have more impact since their percentage of the measured points increases. SAP is computing regressions over many points and should therefore naturally expose a high robustness to scale changes. We simulated scale changes by randomly sampling decreasing amounts of points from the original depth data. The recall rates reported in Table 4 indicate that SAP has indeed a high scale invariance. Please notice that sampling 25 % of the original points corresponds to doubling the distance to the camera and sampling 10 % is approximately the triple distance. Up to this distance the categorization performance does not decrease more than 3.0 %.

Rotations are another important kind of transformation that regularly occur between objects in the real world and the camera. In Figure 6(a) we define three kinds of basic rotations: pan, tilt and roll. Arbitrary rotations consist of these three basic rotations. In the following, we examine to which extent SAP can handle each of of them.

Table 4. Effect of lower resolution point cloud data on the SAP-7-7-2 descriptor. The resulting recall rates are a measure of robustness to scale changes.

Percentage of Points	100 %	50 %	25 %	10 %	6.25 %	4 %
Camera Distance Factor	1	1.4	2	3.2	4	5
Performance	77.9 %	78.2 %	77.1 %	74.9 %	74.8 %	73.6 %



(a) (b)

Fig. 6. Analysis of the rotational robustness of the SAP descriptor: (a) definition of the rotational axes and (b) robustness with respect to tilting rotations.

First, we evaluate the robustness to tilting rotations of the object. This kind of rotation occurs for example when the camera watches the object from a different height and angle. To gauge the robustness of the SAP descriptor against tilting rotations, we trained the categorization system with the original data from the IPA-2 database and the same data from every object view tilted by angle α against the camera. Then we measured the categorization performance on object views of objects outside the training set which were tilted by angle $\alpha/2$. This way we can predict how many different tilt angles have to be present in the data of training objects to allow for successful categorization at the intermediate tilt angles. In Figure 6(b) the recall rates are plotted against the tilt angles of the test data. We can see that SAP can still categorize 73.0 % of the test object views, which were tilted by 15°, while the training set only contained object views at tilt angles of 0° and 30°. Consequently, it would suffice to capture object views at different tilt angles every 30° of training instances to successfully model a class.

A similar analysis can be done for pan rotations. As we already have 36 views of each object around the pan direction in the IPA-2 database, an analysis about the rotational stability around the pan axis can be conducted by excluding more and more views from the training set. Testing is done with all views. Figure 7(a)

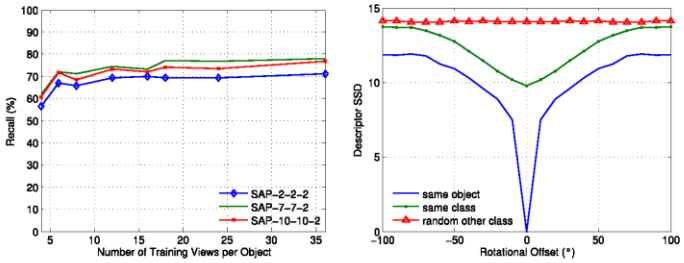


Fig. 7. Analysis of the rotational robustness of the SAP descriptor along the pan axis: (a) recall analyzed against the number of equally distributed views of the training objects for three configurations of the SAP descriptor. (b) Averaged sum of squared differences between SAP-7-7-2 descriptors obtained from different viewing angle offsets for descriptors originating from the same object, objects of the same category and random non-class objects.

shows the relation between utilized training views and recall rates. We learn that 18 views are enough to maintain a high recall rate of 77.0 %. This corresponds to capturing depth images of the training objects in a pan distance of 20°.

Fig. 7(b) furthermore shows the results of descriptor repeatability tests on the IPA-2 database which underline that the descriptor's similarities of neighboring viewing angles are very high. Similarity between two descriptors c_1 and c_2 is measured as the sum of squared differences (SSD) $SSD = \|c_1 - c_2\|^2_{L_2}$. Moreover, with increasing angular offset the SAP descriptors are still much more similar to the original object than to objects of any another class and also very similar to descriptors from other objects of the same class.

SAP, as introduced in Section 3.2, has no means to compensate roll rotations. This implies that SAP can only recognize instances of the learned classes as long as they are standing in a similar upright position as the training objects. Of course, this is not generally the case in reality. Therefore, we devised a rotational transformation which precedes the SAP computation. This transformation compensates roll rotations of the captured object by projecting the 3D points into the image plane and computing a repeatable orientation. Then the point cloud is rolled to a canonical orientation so that the following SAP computation always runs on a well-adjusted point cloud. Several experiments showed the success of this idea with recall rates around 77.0 %. A paper about the analysis of this extension of the SAP descriptor is in preparation.

5 Conclusions

In this paper we introduced the novel descriptor SAP for categorizing simple household objects. SAP directly computes global features on possibly noisy point cloud data. We showed that if SAP descriptors are sampled appropriately from different views of training objects, SAP can obtain very good categorization results of 78% within a short computation time of 57 ms per computation. These results compare favorably to GFPFH, GRSD and VFH. Further experiments proved the invariance of the SAP descriptor to scale, to tilt rotations up to $\pm 15°$ and to pan rotations up to $\pm 10°$. These results provide useful suggestions how to sample views from the training objects to ensure a complete coverage.

We also discussed an extension for full rotation invariance around the camera axis briefly. The results of our experiments are encouraging. Thus, future work will be devoted to a careful evaluation of this approach. Furthermore, we plan to analyze whether it is possible to create some artificial views automatically to decrease the number of views which have to be captured from training objects. Finally, the SAP descriptor shall be tested with the Kinect depth sensor on numerous real world scenes.

6 Acknowledgments

This research was partly funded from the EU FP7-ICT-287624 Acceptable robotiCs COMPanions for AgeiNg Years (ACCOMPANY).

References

1. Rusu, R.B., Holzbach, A., Beetz, M., Bradski, G.: Detecting and segmenting objects for mobile manipulation. In: ICCV, S3DV Workshop. (2009)
2. Marton, Z.C., Pangercic, D., Blodow, N., Beetz, M.: Combined 2D-3D Categorization and Classification for Multimodal Perception Systems. The International Journal of Robotics Research **30**(11) (September 2011) 1378–1402
3. Rusu, R.B., Bradski, G., Thibaux, R., Hsu, J.: Fast 3d recognition and pose using the viewpoint feature histogram. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan (2010)
4. Galleguillos, C., Belongie, S.: Context based object categorization: A critical survey. Computer Vision and Image Understanding (CVIU) **114** (2010) 712–722
5. Zhang, H., Berg, A.C., Maire, M., Malik, J.: SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition. (2006) 2126–2136
6. Deng, J., Berg, A.C., Li, K., Fei-Fei, L.: What does classifying more than 10,000 image categories tell us? In: Proceedings of ECCV. (2010) 71–84
7. Johnson, A., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. IEEE Trans. PAMI **21**(1) (May 1999) 433 – 449
8. Koenderink, J.J., van Doorn, A.J.: Surface shape and curvature scales. Image Vision Computing **10** (October 1992) 557–565

9. Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough transform and 3d surf for robust three dimensional classification. In: Proc. of the European Conf. on Computer Vision. (2010)
10. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: Proceedings of the 11th European Conference on Computer Vision: Part III. ECCV'10, Berlin, Heidelberg, Springer-Verlag (2010) 356–369
11. Knopp, J., Prasad, M., Van Gool, L.: Orientation invariant 3d object classification using hough transform based methods. In: Proc. of the ACM workshop on 3D object retrieval. (2010) 15–20
12. Toldo, R., Castellani, U., Fusiello, A.: A bag of words approach for 3d object categorization. In: Proc. of Int. Conference on Computer Vision. (2009) 116–127
13. Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D.: Shape distributions. ACM Tr. on Graphics **21**(4) (2002) 807–832
14. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3D shape descriptors. In: Symposium on Geometry Processing. (June 2003)
15. Pu, J., Yi, L., Guyu, X., Hongbin, Z., Weibin, L., Uehara, Y.: 3d model retrieval based on 2d slice similarity measurements. In: Proceedings of the 3D Data Processing, Visualization, and Transmission. (2004) 95–101
16. Endres, F., Plagemann, C., Stachniss, C., Burgard, W.: Unsupervised discovery of object classes from range data using latent dirichlet allocation. In: Proc. of Robotics: Science and Systems. (2009)
17. Bo, L., Ren, X., Fox, D.: Depth Kernel Descriptors for Object Recognition. In: IROS. (September 2011)
18. Wahl, E., Hillenbrand, U., Hirzinger, G.: Surflet-pair-relation histograms: A statistical 3d-shape representation for rapid classification. In: 3-D Digital Imaging and Modeling. (2003) 474–481
19. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: Proc. of Int. Conference on Robotics and Automation (ICRA), Shanghai, China (2011)
20. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24** (June 1981) 381–395
21. Marton, Z.C., Rusu, R.B., Jain, D., Klank, U., Beetz, M.: Probabilistic Categorization of Kitchen Objects in Table Settings with a Composite Sensor. In: Proc. of the Int. Conf. on Intelligent Robots and Systems, St. Louis, MO, USA (2009)
22. Collet Romea, A., Srinivasa, S., Hebert, M.: Structure discovery in multi-modal data: a region-based approach. In: Proceedings of ICRA. (2011)
23. Breiman, L.: Random forests. Machine Learning **45** (2001) 5–32
24. Burges, C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2** (1998) 121–167
25. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics **28** (1998) 2000
26. Mozos, O.M., Burgard, W.: Supervised learning of topological maps using semantic information extracted from range data. In: IROS. (2006) 2772–2777
27. Browatzki, B., Fischer, J., Graf, B., Bülthoff, H., Wallraven, C.: Going into depth: Evaluating 2d and 3d cues for object classification on a new, large-scale object dataset. In: Proc. of Int. Conf. Computer Vision Workshop on CD4CV. (2011) 1–7
28. Bradski, G., Kaehler, A.: Learning opencv: Computer vision with the opencv library (2008)
29. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2** (2011) 27:1–27:27

Adding Rotational Robustness to the Surface-Approximation Polynomials Descriptor

Richard Bormann, Jan Fischer, Georg Arbeiter and Alexander Verl
Fraunhofer IPA, 70569 Stuttgart, Germany
{richard.bormann, jan.fischer, georg.arbeiter, alexander.verl}@ipa.fraunhofer.de

***Abstract*—The Surface-Approximation Polynomials (SAP) descriptor has been shown to be an appropriate global surface descriptor for object categorization tasks in robotic applications [1]. Nevertheless, in the original formulation the SAP descriptor is not invariant against rotations around the camera axis. This paper explains and evaluates two methods which pre-process the input data to yield repeatably well-aligned point clouds for the computation of the SAP descriptor. We show that the SAP descriptor can be rendered robust against rotations while retaining almost the full performance of the original approach which is superior to GFPFH, GRSD and VFH.**

I. Introduction

Understanding the elements of the environment is essential for robots that are supposed to assist humans in their homes. Only if robots are able to recognize objects in their surroundings, they can manipulate them in a useful way. However, the large variety of objects in home environments turns instance-based object recognition infeasible as the appearance of each single object would have to be learned individually by the robot. Object categorization instead strives to recognize object classes. Hence, objects of a known class can still be recognized even if a certain instance is completely new to the robot. Moreover, the recognition problem becomes more tractable since there are less classes than individual objects.

The Surface-Approximation Polynomials (SAP) descriptor has been recently introduced as a global 3D surface descriptor that is well-suited for the task of object categorization with a robot [1]. The SAP descriptor approximates the surface geometry of a single-shot view onto an object with polynomials. The categorization system based on the SAP descriptor can determine the category label of unknown objects that are captured with a depth sensing device like a PMD CamCube or a Microsoft Kinect. It has been shown in [1] that the SAP descriptor is robust enough to compensate smaller viewpoint changes up to 15° in pan and tilt direction. However, roll rotations of the object or camera cannot be handled at all with the basic approach. Especially, modeling all possible roll rotations with sufficiently many training views is infeasible as the number of required images would explode. Please consult Fig. 1 for the definitions of rotations.

In the real world objects may occur in any arbitrary pose. Consequently, the SAP descriptor should be able to cover every object pose. In this paper we propose and carefully evaluate two methods which align the input data canonically: a full 6 DOF transformation based on Principal Component Analysis (PCA) over the input point cloud as well as a roll compensation which only aligns the input data to a common roll angle. We furthermore introduce a rule to obtain a repeatable definition of the axis directions of PCA.

Fig. 1. Definition of the rotational axes for the analysis of the rotational robustness of the SAP descriptor and an example image with real categorization results of variously aligned, previously unseen objects.

The outline of the paper is as follows. In Section II we discuss relevant work to the topics of object categorization and pose alignment. Section III explains the employed approaches, which are evaluated in Section IV. We conclude in Section V with a summary and an outlook for future work.

II. Related Work

Object categorization is a topic of high interest in robotics. The most popular global descriptors that can be computed fast enough for using them in robotics are Global Fast Point Feature Histograms (GFPFH) [2], Global Radius-based Surface Descriptors (GRSD) [3], and Viewpoint Feature Histogram (VFH) [4]. GFPFH builds histograms on local Fast Point Feature Histograms [5] which themselves are histograms on the relative pose of local coordinate frames determined at all point pairs within a neighborhood. The GRSD descriptor is composed similarly to the GFPFH descriptor from local RSD features, which basically represent the local minimum and maximum curvature around a point. VFH is very similar to GFPFH but supposed to also encode the viewpoint at the visible object surface. VFH includes the camera axis in the computation of FPFH histograms to establish viewpoint dependent signatures for the trained objects. The recently proposed SAP descriptor [1] instead directly builds a global descriptor without computing local features and produces categorization results superior to the previous descriptors. We will provide a short description of the SAP descriptor in Section III-B.

As stated above, the problem of the original SAP descriptor is the missing inherent invariance to roll rotations. Pose normalization of 3D object models is an important topic in the shape retrieval literature where it is applied to transform objects into a canonical orientation w.r.t. translation, size and rotation for the use with pose variant descriptors. The most popular approach in this community seems to be a PCA-based alignment [6]–[11] because of its simple and fast computation and numerical robustness. However, a serious problem with PCA is the repeatable definition of the coordinate axis into positive or negative direction of the principal axes. In [6] all four possible configurations were tested and the orientation with the best similarity between two query objects was chosen. However, our task does not involve two previously known objects. Therefore, we define the axis definitions according to the distribution of points of the query object in the new coordinate system. This method is similar to the approach of [12] where the axes are directed to the side with a greater total area of the polygons. In [8] continuous PCA is introduced to deal with different triangle resolutions in polygon meshes. These problems do not occur with volumetric or mass-based 3D models as we use.

A second classical method for pose alignment is Extended Gaussian Images (EGI) [13]. This algorithm computes the projections of the surface normals on a Gaussian sphere around the object. In [10] maximum normal distribution is proposed as another normal-based pose alignment method for polygon meshes. The idea is to create a histogram over the total area of surfaces which have the same distance to the object center and the same surface normal. Then the normal direction with the largest total area is picked as first principal axis and the orthogonal normal with next largest area as second. Since our input data does not contain meshes we use a PCA-based full pose alignment with adequate axis definitions and a roll compensation with PCA involved in the computations.

III. Methods

Besides the detailed description of the orientation alignment this section briefly summarizes the principle of the SAP descriptor and the underlying categorization framework. The next paragraph starts with a description of data preparation.

A. Data Acquisition and Segmentation

The SAP descriptor is a global descriptor which describes the surface of objects. Therefore, segmented object data is needed to compute the SAP descriptor. After capturing a depth image the scene is segmented in three steps. First, the amount of points in the input point cloud is reduced with a voxel filter that has a leaf size of 7.5 mm. Then the larger planes are iteratively estimated and removed from the input point cloud. Third, the remainder of points is aggregated with Euclidean clustering. Those clusters which contain more than 50 points are then considered as object candidates and forwarded to the SAP descriptor computation. The functions for clustering base upon the PCL library [14].

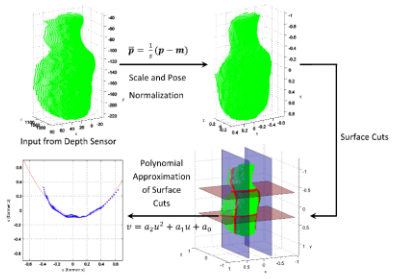


Fig. 2. Computation scheme of the SAP descriptor. The upper left image shows the raw point cloud input. Following the arrows, pose and scale normalization is applied, surface cuts are extracted (red and blue planes cut the surface, cuts indicated as red points) and finally approximated with a polynomial (original points in blue, the red line shows the polynomial).

B. The Surface-Approximation Polynomials Descriptor

The Surface-Approximation Polynomials (SAP) descriptor has been described in detail in [1]. Therefore, we only provide a schematic summary of the algorithm at this place. The basic idea behind the SAP descriptor is to represent object classes by the shape of their surface. As shown in Fig. 2 this is accomplished by normalizing the input point cloud $\mathcal{P}$ to a common centroid and scale, cutting the surface with planes perpendicular to the camera plane and approximating the geometry of the cuts with polynomials via linear regression. Having n_x cuts along the x-direction of the camera plane and n_y cuts along the y-axis this yields n_x+n_y parameter vectors ${}^{i}\mathbf{a}^{\mathrm{T}}, i = 1,\ldots,n_x+n_y$, of the polynomial coefficients. Furthermore, we compute a Principal Component Analysis (PCA) to obtain the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ which serve as a measure of object size within the three principal directions. The SAP descriptor is a concatenation of these three size parameters and the polynomial coefficients

$$\mathbf{c} = \left[\frac{\lambda_1}{\gamma}, \frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1}, {}^{1}\mathbf{a}^{\mathrm{T}}, \ldots, {}^{\mathbf{n_x+n_y}}\mathbf{a}^{\mathrm{T}}\right]. \quad (1)$$

To support a range of object sizes λ_2 and λ_3 contribute only with their relation to λ_1. λ_1 is stored with an optional scale parameter γ to incorporate one measure of absolute size.

C. Extensions for Rotation Invariance

The unaligned SAP descriptor as described in [1] is only invariant against translation and scale but not against rotations, especially around the camera axis (roll). Although it is possible to model viewpoints from different pan or tilt angles with respective training images from a grid around the object, there is no way to capture different poses in roll direction without capturing a vast mass of images. To be able to handle objects in arbitrary poses, rotation invariance has to be accomplished by further measures. Here we propose a full pose alignment with PCA that can compensate pan, tilt and roll rotations of the captured objects as well as a roll compensation method which still has a need for sufficient coverage of training views regarding pan and tilt rotations.

1) PCA-based Pose Normalization: To receive a repeatable, scale- and rotation-invariant description, the pose of the point cloud is normalized by computing the mean $\mathbf{m}$ and the principal axes $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ via PCA. Every point $\hat{\mathbf{p}}$ of the camera coordinate system $\hat{\mathfrak{C}}$ with the axes $\hat{\mathbf{x}} = (1,0,0), \hat{\mathbf{y}} = (0,1,0)$, and $\hat{\mathbf{z}} = (0,0,1)$ is then translated to shift the point cloud's center into the origin, rotated such that the eigenvectors are aligned with the coordinate axes and scaled with the largest eigenvalue λ_1 yielding the normalized point

$$\mathbf{p} = \frac{1}{2\sqrt{\lambda_1}} \cdot \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} \end{bmatrix}^{\mathrm{T}} \cdot (\hat{\mathbf{p}} - \mathbf{m}) \quad . \tag{2}$$

Translating the center of the point cloud to the origin ensures translation invariance w.r.t. the coordinate system of the depth sensor while the rotation compensates for any object rotation around the camera axis and for minor rotations around the other two axes. The scaling operation effects that the majority of the coordinates resides in the range of [-1,1].

As the sign of the direction of the eigenvectors obtained from PCA does not necessarily coincide between several recordings, we have to enforce a repeatable orientation of the new coordinate system $\mathfrak{C}$ with the coordinate axes $\mathbf{x} = \mathbf{v_1}, \mathbf{y} = \mathbf{v_2}$, and $\mathbf{z} = \mathbf{v_3}$. Therefore, we first check that the eigenvectors constitute a right-handed system which is the case if the triple product

$$(\mathbf{v_1} \times \mathbf{v_2}) \cdot \mathbf{v_3} > 0 \tag{3}$$

is positive. If condition (3) is not met, we invert the coordinates of eigenvector $\mathbf{v_3}$ before transforming the point cloud. Then, we obtain a repeatable coordinate system if the following three rules are fulfilled:

1) The new z-axis, which has the coordinates $\mathbf{z} = \mathbf{v_3}$, must point towards the camera. Hence, the condition $\hat{\mathbf{z}} \cdot \mathbf{z} < 0$ must hold since the initial $\hat{\mathbf{z}}$-axis of the camera coordinate system with coordinates $\hat{\mathbf{z}} = (0,0,1)^{\mathrm{T}}$ points away from the camera.
2) The majority of points should have negative x-coordinates in the new coordinate system $\mathfrak{C}$.
3) The new coordinate system $\mathfrak{C}$ is a right-handed system.

These conditions are checked in the given order. If rule 1 is not fulfilled, we only change the signs of eigenvectors $\mathbf{v_2}$ and $\mathbf{v_3}$ before transforming the point cloud to keep the coordinate system right-handed. The second condition can only be verified after the transformation of the points. If it is not met, we have to negate the eigenvectors $\mathbf{v_1}$ and $\mathbf{v_2}$ and the x- and y-components of the transformed points to keep the coordinate system right-handed at the same time. After executing the preceding steps, rule 3 is already fulfilled. Rule 3 is always enforced in step 1 and step 2 by negating $\mathbf{v_2}$, the $\mathbf{y}$-axis, and the y-coordinates. After the verification of all three rules, the eigenvectors $\mathbf{v_1}, \mathbf{v_2}$, and $\mathbf{v_3}$ correspond to the new coordinate axis $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$, respectively. After normalization, the surface of the object is aligned in a way that the two dimensions with the largest extent correspond to the x- and y-axes. We evaluate the success of this measure in Sec. IV-B.

2) Roll Compensation: The second approach to render the SAP descriptor rotation invariant w.r.t. roll rotations does not apply a full 3D transform to the point cloud but only aligns its rotation around the camera axis. This way, roll rotations of objects are made transparent to the algorithm. The roll compensation is motivated by the possible misalignments with full PCA (see Sec. IV-B) and was developed with the goal to transform the point cloud as little as possible.

The roll compensation algorithm works as follows: first a silhouette image is created from the projection of the point cloud onto the camera plane. Then we compute the centroid $\mathbf{m}$ of this 2D silhouette as well as the two principal axes $\mathbf{v_1}$ and $\mathbf{v_2}$ using PCA. Next the silhouette is rotated to be aligned with the principal axes and it is counted whether more points have positive x-coordinates than negative. If this condition does not hold, the directions of the principal axes are negated. This step ensures to have a repeatable definition of the direction of the new coordinate system. Then we compute the angle α between the first principal axis $\mathbf{v_1}$ and the image's x-axis $(1,0)$:

$$\cos\alpha = \frac{1}{\sqrt{v_{11}^2 + v_{12}^2}} \cdot \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad . \tag{4}$$

Finally, we rotate every point $\hat{\mathbf{p}}$ of the original point cloud $\mathcal{P}$ by angle α around the camera axis $\hat{\mathbf{z}}$:

$$\mathbf{p} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} (\hat{\mathbf{p}} - \mathbf{m}) + \mathbf{m} \quad . \tag{5}$$

This yields a repeatable roll compensation for the input point cloud so that the SAP descriptor can then be computed on a point cloud with aligned roll angle. Section IV-C examines how well the roll compensation works in practice.

D. Classification Framework

The object categorization framework is identical to [1]. It is supposed to serve two purposes: first the system should be able to search for instances of a certain class and assert whether test objects belong to it. This is a binary classification task. Second, it should be able to label unknown objects with the correct class. This is a multi-class classification problem. To be able to deal with both problems the classification framework builds on binary Random Forest classifiers [15] which separate each class against the remainder of classes. Binary decisions are obtained by directly querying these classifiers. A probabilistic multi-class extension is employed for the labeling task, which directly computes the labels from the likelihoods of the binary classifiers and their decision reliabilities that originate from statistics.

IV. Evaluation

The SAP descriptor of unaligned point clouds has already been examined in [1]. In this paper we discuss the impact of the PCA-based pose alignment and roll compensation and compare the outcomes with those from unaligned input data. All results reported on database tests are determined with a 10-fold leave out one object cross-validation.

A. Database

Database tests are conducted with the database of household objects named IPA-2 in [1]. It contains 151 objects from 14 classes. Among these classes are binders, bottles, cans, cups, dishes, drink cartons, computer mice, pens, silverware, etc. Each object was captured 36 times with a PMD CamCube from a light tilt angle with an offset of 10° in the pan angle. The average number of points per object is 26491. A detailed description of the IPA-2 object database can be found in [16]. This set is publicly available at http://www.kyb.mpg.de/nc/employee/details/browatbn.html.

B. PCA-based 6 DOF Pose Normalization

This section analyzes the robustness of the SAP descriptor against rotations and camera distance of the captured objects when the input point cloud is aligned with the PCA-based approach (Sec. III-C1). The analysis of the basic SAP descriptor in [1] shows that the SAP-7-7-2 configuration yields very good results. Thus, all experiments in this section will be conducted with this parameter setting if not mentioned else. The naming scheme for SAP descriptors is SAP-n_x-n_y-n_p, where n_x and n_y describe the number of cuts along the x- and y-coordinate axes (after alignment of the point cloud) and n_p denotes the degree of the approximating polynomial.

1) Theoretical Analysis: The function and power of PCA-based pose normalization is demonstrated on a cuboid. Fig. 3(a) displays this cuboid as well as a multitude of camera view points which pan in the range $[5.625°, 84.375°]$, tilt within $[15°, 75°]$ and are depicted as black dots with a black line indicating the camera axes. The black point in the middle of the object is the real centroid of the cuboid whereas the red points with the coordinate frames attached display the object centers that are computed from the three visible surfaces of the cuboid. The offset between the centroid that we can estimate from the visible data and the real centroid has an effect on the chosen translation compensation since the position of the estimated centroid depends on the view point. The locations of the estimated centroids differ since the depth sensor samples the less points from a surface the more the viewing angle onto the surface becomes acute. Let $\mathcal{S} = \{S_1, S_2, S_3\}$ denote the set of visible surfaces of the cuboid. Then the theoretical centroid $\mathbf{x_s}$ of the visible surfaces is computed as

$$\mathbf{x_s} = \mathbf{x}(S_1)A(S_1) + \mathbf{x}(S_2)A(S_2) + \mathbf{x}(S_3)A(S_3) \quad (6)$$

where $A(S)$ stands for the area of surface S and $\mathbf{x}(S)$ for the centroid of S. However, depending on the viewing angle the depth sensor can only capture a ratio of the maximum amount of points that could be captured from a surface if the camera axis was perpendicular to the surface. We model this effect with the following ratios for the visible portions of each area where α is the pan angle and β represents the tilt angle:

$$\begin{aligned} S_1&: \quad \cos(\alpha)\cos(\beta), \\ S_2&: \quad \sin(\alpha)\cos(\beta), \\ S_3&: \quad \sin(\beta)\ . \end{aligned}$$

The view-dependent centroids (red points in Fig. 3(a)) are computed according to Eq. (6) where every area $A(S_i)$ only accounts with the respective view-dependent ratio. It shows that the perceived centroids still lie quite close to each other when the change in viewing angle is below 15°. If the surface of the object is sufficiently smooth this small translation of the centroid will not affect the polynomial approximation substantially given that the rotation can be compensated.

The rotation compensation is supposed to be accomplished by the PCA-based alignment. The idea is to determine the principal axes of the captured object, which are supposed to be stable under minor rotations, and rotate the surface to be aligned with these principal axes. While computing the principal axes we obey the aforementioned ratios of visible points on the object's surfaces to obtain a realistic result. The pose normalized coordinate system is assigned to the principal axes in descending order of corresponding eigenvalues, that is the new x-axis is the eigenvector with the largest eigenvalue. The resulting pose normalized coordinate axes for the cuboid example are displayed for all viewing angles at the position of the estimated centroids in Fig. 3(a). The red axis displays the x-axis, the y-axis is green and the z-axis is blue. It is visible that the estimated principal axes correspond roughly to the real principal axes of the cuboid and all coordinate frames are similarly aligned over a wide range of view points. To illustrate the latter fact, a comparison of the distribution of coordinate frames without and with PCA-based pose normalization is provided in Fig. 3(b) and Fig. 3(c), respectively. While the original coordinate frames scatter a lot, the normalized coordinate frames have little deviation over intermediate viewpoint changes and barely follow the camera movements. Consequently, PCA-based pose normalization will align object surfaces similarly within an intermediate range of pan and tilt rotations and hence yield similar SAP descriptors. Roll rotations are not considered in this analysis because the PCA-based pose normalization and the computation of repeatable axis directions yield the same normalized pose for every roll angle while pan and tilt angles are fixed.

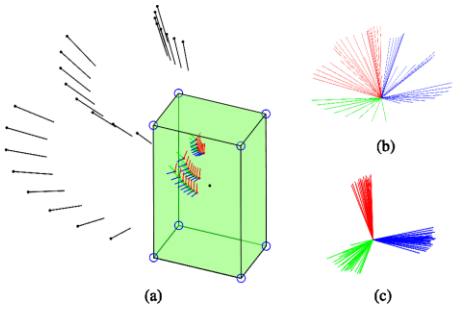


Fig. 3. (a) The normalized coordinate frames are displayed at the estimated centroids for the considered viewing angles, which are displayed as black dots and lines. Collection of (b) the original coordinate frames and (c) the normalized coordinate frames of the cuboid seen from those viewpoints.

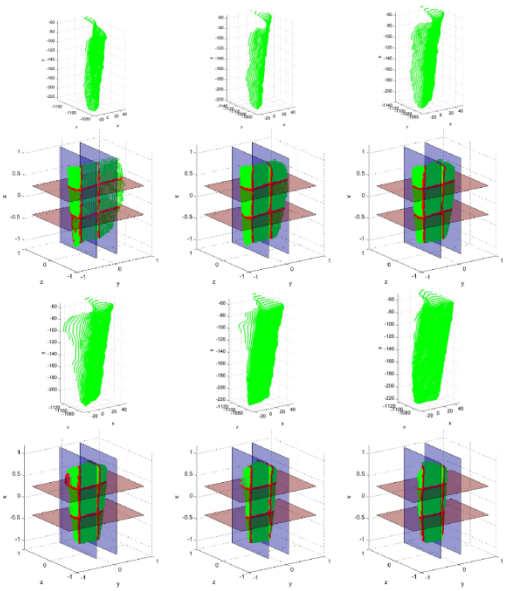


Fig. 4. Point cloud of a milk box captured from six neighboring viewing angles. The first and the third row show the original data from the depth sensor. Row two and four display the corresponding point clouds which are aligned with PCA-based pose normalization. Two exemplary surface cuts are drawn in each cutting direction into the pose normalized views.

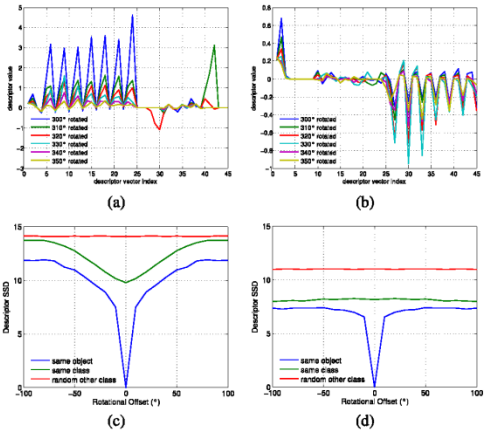


Fig. 5. Analysis of the similarity of SAP descriptors: SAP-7-7-2 descriptors from (a) the original snapshots and (b) the pose normalized point clouds of the example views of the milk box. Averaged descriptor similarity between views with varying angular offset on (c) the original data and (d) the pose normalized data. Average similarity is measured against rotations of the same object, objects from the same class and objects of other classes.

To demonstrate the effect of PCA-based pose normalization on real data Fig. 4 shows a sequence of views onto a milk box. This box rotates on a rotary disc so that the camera movement is effectively a pan rotation with an angular offset of 10° between successive views. The first and third row show the point clouds as captured by the sensor. The second and fourth row display the corresponding views onto the milk box after PCA-based pose normalization has been applied. While the original point cloud rotates by 50° over the sequence the pose normalized views look very similar in all images as predicted by the previous analysis.

To back the claim that the SAP descriptors obtained from pose normalized views are more similar to each other than those obtained from the original views, Fig. 5 provides two pieces of evidence. The first row of images displays the SAP-7-7-2 descriptors of all views of the milk box from Fig. 4. In detail, Fig. 5(a) contains all six SAP-7-7-2 descriptors from the original views whereas Fig. 5(b) displays the SAP-7-7-2 descriptors from the PCA-based pose normalized views. To ensure a fair comparison between both cases the original views are scaled to fit into the unit volume as well. To compare the descriptors of both approaches please consider that the axis definitions change through the pose normalization. In the milk box example, the x- and y-axis definitions swap between original and normalized view and consequently, the SAP polynomial coefficients from the first half of one diagram can be found in the second half of the other diagram. The corresponding coefficients are inverted because the z-axis direction switches through the pose normalization. Only the first three size components of the descriptors correspond in both cases and take the same values. We can observe that the SAP-7-7-2 descriptors from a range of viewing angles of 50° do not differ much when PCA-based pose normalization is applied whereas the descriptors obtained from the original views exhibit a transition in the coefficients in the first half of the descriptor. This steady decrease in magnitude is caused by the pan rotation of the object which lets the surface appear as a backwardly slanted plane at first and transitions to a plane parallel to the camera in the end. The pose normalized views present a parallel plane for all views instead which results in very similar descriptor coefficients in all cases.

To show that this fact holds in general this analysis has to be extended to the whole database. Figure 5 therefore displays two plots in the second row in which the similarity of SAP-7-7-2 descriptors of neighboring views is studied on the whole dataset. Similarity between two descriptors c_1 and c_2 is measured as the sum of squared differences (SSD) $SSD = \|c_1 - c_2\|^2_{L_2}$. We can see in Fig. 5(c) that the descriptor similarity decreases significantly with growing offset between two views if the descriptors are computed on the original point cloud. If computed on the normalized data instead (see Fig. 5(d)), the similarity of descriptors from neighboring views barely increases even for larger rotations. This finding indicates that PCA-based pose normalization helps to keep SAP descriptors computed from neighboring views quite similar.

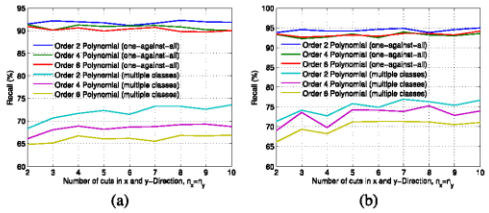


Fig. 6. Comparison of different configurations of the SAP descriptor with varying numbers of surface cuts and degrees of the polynomials. The input data was aligned with (a) PCA and (b) roll compensation.

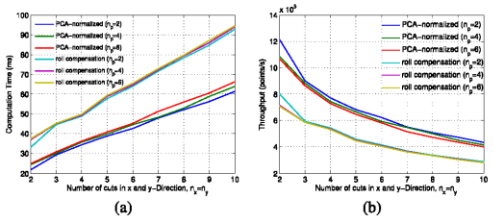


Fig. 7. Dependency of (a) computation time and (b) throughput of the SAP descriptor for increasing numbers of surface cuts and polynomial orders. The input data is either aligned with PCA or with roll compensation.

Although the preceding analysis has pointed out several desirable properties of the chosen PCA-based pose normalization it is well known that this kind of pose normalization is problematic and possibly unstable if applied to objects which do not have such canonical orientations as the cuboid [17]. Therefore, we test the impact of pose normalization by measuring the performance on the object categorization task.

2) Database Tests: According to the analysis in [1] the influence of the numbers of surface cuts n_x and n_y as well as the degree of the approximating polynomials needs to be examined. Figure 6(a) displays the recall rates for the binary classification problem of separating one object class against the others as well as for the multi-class labeling task where each object view has to be assigned one of the class labels. The influence of the number of cuts is as expected and coincides with findings of experiments with the unaligned data: especially in the multi-class problem the recall rates increase steadily with growing numbers of surface cuts. The binary categorization performance, however, remains almost constant independent of the number of surface cuts. Nevertheless, the increasing performance for the multi-class task indicates that the binary decisions become more confident, that is the probabilities for the respective decisions of the binary classifiers grow with the number of surface cuts.

As in the unaligned case approximations of higher order polynomials yield a worse performance. Manual inspection of the descriptors provides an explanation for this observation: it shows that higher order polynomials are less stable and tend to model the noise from the sensor. Besides these qualitative observations we also confirm that the SAP-7-7-2 configuration proves to be among the top performers, however, with slightly lower recall rates than with unaligned data. In the binary classification case the performance drops from 94.9% with unaligned data to 91.7% with PCA-aligned data and for the multi-class labeling task the performance decreases from 77.9% to 73.2%. The good performamce in the unaligned case is not surprising since the objects in the database are already well-aligned. The decrease by almost 5% of multi-class recall indicates that the PCA alignment introduces a significant number of misalignments.

Next, Figure 7(a) shows the average computation time for one SAP descriptor and Figure 7(b) the respective computational throughput. These results resemble those of the unaligned computation very much in qualitative and quantitative aspects. Thus, the additional pose alignment does not introduce significant overhead for the computation. There is a linear increase in computation time with rising numbers of surface cuts. The computation time for the SAP descriptor is quite low as all examined configurations are determined within less than 100 ms on one core of a 2.8GHz Intel I7 mobile processor with 6GB RAM. The runtime of the SAP-7-7-2 configuration e.g. allows to compute the descriptor with almost 21 Hz. That is SAP could classify up to 21 objects in a scene within one second which is a respectable rate.

C. Roll Compensation

We will not provide a similarly extensive evaluation for this approach of orientation compensation as the results strongly resemble those of the unaligned approach. The reason for this lies in the good alignment of objects in the database which renders the input data almost equal for both database tests. However, the success of roll compensation on objects in other poses than those in the training data is proven by the examples in Fig. 1 and 9, e.g. for cans, cups and the binder. The following analysis indicates the equalities and differences to the results of the unaligned approach.

The impact of parameters n_x, n_y and n_p on the categorization performance is shown in Fig. 6(b). The qualitative results correspond with previous findings and the recall rate of 77.0% of the SAP-7-7-2 descriptor with roll compensation comes close to the 77.9% of the unaligned method. The computation times with roll compensation are higher than with PCA-based alignment or without alignment by 20 ms to 30 ms as we can see in Fig. 7(a). Nevertheless, the SAP-7-7-2 configuration still classifies almost 14 objects per second. The linear dependency on the number of cuts remains.

D. Comparison of the Approaches

This paragraph compares the unaligned, PCA-aligned and roll-compensated SAP descriptors according to their categorization performance, runtime and robustness against rotations and scale changes. Table I summarizes the categorization performance and computation times of the three variants of SAP descriptors and other descriptors from literature. It shows that the SAP variant with roll compensation achieves

TABLE I
COMPARISON OF SEVERAL DESCRIPTORS REGARDING MULTI-CLASS CATEGORIZATION PERFORMANCE, AVERAGE COMPUTATION TIME PER VIEW AND AVERAGE THROUGHPUT IN POINTS PER SECOND.

Descriptor	Recall	Time	Throughput
Shape Distributions [16]	25.4 %	31 ms	∼ 855 000 pts/s
Shape Index [16]	34.6 %	78 ms	∼ 339 000 pts/s
Shape Context 3D [16]	55.2 %	234 ms	∼ 113 000 pts/s
Depth Buffer [16]	72.9 %	**16 ms**	**∼ 1 656 000 pts/s**
GFPFH [1]	54.4 %	921 ms	28 928 pts/s
GRSD [1]	56.1 %	957 ms	27 841 pts/s
VFH [1]	68.4 %	93 ms	205 883 pts/s
SAP-7-7-2			
unaligned [1]	**77.9 %**	57 ms	463 439 pts/s
with roll compensation	77.0 %	72 ms	370 314 pts/s
with PCA alignment	73.2 %	48 ms	552 262 pts/s

almost the recall rate of the unaligned SAP descriptor but needs 15 ms of additional computation time. The similar recall rate indicates that roll compensation works with few errors since the performance of the unaligned SAP descriptor is kind of a limit for methods with pose alignment as the database objects are already well-aligned. The recall rate of the SAP descriptor with PCA alignment is almost 5% lower than this limit suggesting that some misalignments occur. The faster computation time compared to the unaligned method is caused by the changed orientation which affects the number of points on surface cuts. All runtime measurements were taken on one core of a mobile I7 2.8 GHz machine with 6GB RAM. A confusion matrix for the categorization with roll compensation is provided in Fig. 8(d). Many class labels are found quite reliably whereas the occurring confusions can usually be easily explained, e.g. bottles and dishliquids have a similar shape and silverware and scissors look the same when seen from the slim side.

The next analysis evaluates the robustness of the variants of SAP descriptors against rotations of the object in pan and tilt direction. For the evaluation on pan rotations we just exclude the respective views from the training data to yield sparser object models sampled only every α degrees in the pan direction. Fig. 8(a) reports on the recall rates obtained with respect to the angular offset $\alpha/2$ of the views of unknown test objects. This means, the angles reported in the diagram correspond with the maximal angular offset to the closest view on another object of this class available in the training set. It is remarkable that in all three cases the performance is still around 60% when the training data only consists of 4 views of each object. We also notice that the recall rate virtually remains constant up to an offset of 15° for the PCA-aligned SAP descriptor and up to 10° for the other two variants. The gap between the PCA-aligned and the unaligned descriptor remains almost constant over the whole range and is surprising as the pose alignment should be benefitial with few views. Apparently the misalignment rate of the PCA-based approach eats up this potential advantage. The performance of roll compensation begins at the same level as the unaligned approach but degrades with growing angular offset towards the performance of PCA alignment. A similar analysis has been carried out for tilt rotations.

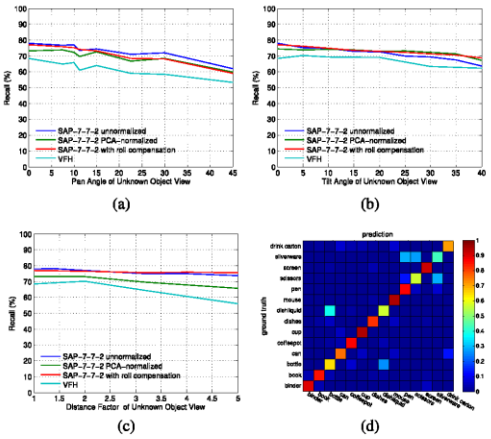


Fig. 8. Robustness of the three variants of the SAP descriptor with respect to (a) pan and (b) tilt rotations as well as (c) camera distance. (d) Confusion matrix for categorization with the SAP descriptor and roll compensation.

Lacking real data from all tilt angles, the experimental setup for tilt rotations is different: the original point clouds are tilted by angle β and because of the changed perspective only a ratio of $\cos\beta$ points are kept in the model. The system is trained with data from tilt angle 0° and β. The test data only contains point clouds tilted by $\beta/2$. Fig. 8(b) displays the recall rates with respect to $\beta/2$. Up to tilt angles of 35° the performance keeps above 67% with all three approaches which is quite high. For PCA-based alignment, the recall rate remains almost constant up to that point, for roll compensation it converges to the level of the aforementioned method. For the unaligned data recall decreases steadily and falls below the other methods at tilt offsets of 15°.

The last experiment evaluates the robustness of the three variants of SAP descriptors against varying camera distance to the objects. To emulate different distances between object and camera we downsample the original point clouds randomly to different distance levels, e.g. to simulate the double distance we only keep 25% of the original points. Fig. 8(c) shows the recall rates for various distances. The unaligned and roll compensated SAP descriptors can retain their performance over almost the whole range of analyzed distance factors. The recall rates of PCA-based alignment, however, decrease significantly after the distance doubles. Apparently, the impact of noise in the point measurements grows larger if less points are available and this affects the stability of the PCA-alignment.

The robustness analysis has also been conducted for VFH to allow for a comparison. The rotational robustness is similar to the SAP descriptors but the robustness to camera distance is lower as visible in Fig. 8(a), 8(b) and 8(c).

Finally, we demonstrate the categorization system on real scenes with previously unseen objects in Fig. 1 and Fig. 9. We selected the roll compensation approach for point cloud

Fig. 9. Exemplary real world scenes with objects from various classes in diverse poses. The objects are not part of the training set. The point clouds are aligned with roll compensation before the SAP descriptor is computed. The right column shows the corresponding object clusters of the point cloud.

alignment to benefit from the higher recall rates and the good robustness against transformations including roll rotations, which are not covered by the unaligned SAP descriptor. We placed the objects in different distances to the camera and turned them in various pan, tilt and roll directions. The recognized object classes are denoted on top of each object with the probability mass for this label in brackets. Although the probability is only in the range of 20% for several objects the alternatives often have significantly lower probabilities. Please notice that a probability of 50% means that no other object can be more likely. Consequently, probabilities of 40% are already strong assertions.

V. Conclusions and Outlook

The analysis of the two proposed pose normalization methods has shown that the PCA-based full pose alignment of the input point cloud is regularly inferior to the approach with roll compensation which can almost achieve the performance of the unaligned SAP descriptor on aligned data. For modeling object classes with the SAP descriptor we therefore recommend to pre-process the input data with roll compensation and capture training images every 20° in pan and tilt direction to obtain optimal performance. If a performance drop up to 5% in the worst case is acceptable, objects can be modeled with 38 views: 12 images per pan rotation at tilt angles -45°, 0°, and 45° as well as one shot from the top and the bottom.

For future research on the SAP descriptor it is planned to substitute the polynomial approximations with splines. Furthermore, we like to add a size parameter to each cut to represent the length of the approximated curves. A transition to part-based models is also planned to cope with occlusions.

VI. Acknowledgments

This research was partly funded from the EU FP7-ICT-287624 Acceptable robotiCs COMPanions for AgeiNg Years.

References

[1] R. Bormann, J. Fischer, G. Arbeiter, and A. Verl, "Efficient Object Categorization with the Surface-Approximation Polynomials Descriptor," in *Spatial Cognition VIII* (C. Stachniss, K. Schill, and D. Uttal, eds.), vol. 7463 of *Lecture Notes in Computer Science*, pp. 34–53, Springer, 2012.

[2] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and segmenting objects for mobile manipulation," in *ICCV*, 2009.

[3] Z.-C. Marton, D. Pangercic, R. B. Rusu, A. Holzbach, and M. Beetz, "Hierarchical object geometric categorization and appearance classification for mobile manipulation," in *Proceedings of the International Conference on Humanoid Robots*, (Nashville, TN, USA), 2010.

[4] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2010.

[5] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *2009 IEEE Int. Conf. on Robotics and Automation*, (Piscataway, NJ), pp. 1848–1853, IEEE, 2009.

[6] M. Novotni and R. Klein, "A geometric approach to 3d object comparison," in *International Conference on Shape Modeling and Applications*, pp. 167–175, may 2001.

[7] J. W. H. Tangelder and R. C. Veltkamp, "A survey of content based 3d shape retrieval methods," in *Shape Modeling Int.*, pp. 145–156, 2004.

[8] D. V. Vranic, D. Saupe, and J. Richter, "Tools for 3d-object retrieval: Karhunen-loeve transform and spherical harmonics," in *IEEE 2001 Workshop Multimedia Signal Processing, Cannes, France*, 2001.

[9] D. V. Vranic, "An improvement of rotation invariant 3d-shape based on functions on concentric spheres," in *ICIP (3)*, pp. 757–760, 2003.

[10] J. Pu and K. Ramani, "A 3d model retrieval method using 2d freehand sketches," in *Proceedings of the 5th International Conference on Computational Science - Volume Part II*, pp. 343–346, 2005.

[11] R. Ohbuchi, K. Osada, T. Furuya, and T. Banno, "Salient local visual features for shape-based 3d model retrieval," in *Shape Modeling International*, 2008.

[12] M. Elad, A. Tal, and S. Ar, "Content based retrieval of VRML objects - an iterative and interactive approach," in *Proc. Eurographics multimedia workshop*, pp. 97–108, 2001.

[13] B. K. P. Horn, "Extended gaussian images," *Proceedings of the IEEE*, vol. 72, no. 2, pp. 1671–1686, 1984.

[14] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. of Int. Conference on Robotics and Automation*, 2011.

[15] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[16] B. Browatzki, J. Fischer, B. Graf, H. Bülthoff, and C. Wallraven, "Going into depth: Evaluating 2d and 3d cues for object classification on a new, large-scale object dataset," in *Proc. of Int. Conf. Computer Vision Workshop on CD4CV*, pp. 1–7, 2011.

[17] J. Pu, L. Yi, X. Guyu, Z. Hongbin, L. Weibin, and Y. Uehara, "3d model retrieval based on 2d slice similarity measurements," in *Proc. of 3D Data Processing, Visualization, Transmission*, pp. 95–101, 2004.