

Object Recognition Tasks for Service Robots

Viktor Seib, Susanne Thierfelder, Dietrich Paulus

Active Vision Group, AGAS Robotics
Institute for Computational Visualistics
University of Koblenz and Landau, Germany
{vseib | sisuthie | paulus}@uni-koblenz.de
<http://robots.uni-koblenz.de>

In the last years service robots have moved into the focus of researchers around the world. In this paper we present our service robots Lisa and GiGo who successfully participate in the RoboCup@Home competition. The software and hardware design of our robots is explained. Further, we examine grasping, one typical task in service robotic that highly depends on object detection. We explain how objects are clustered for grasping and how they are recognized. Further, we present an algorithm for color classification.

Introduction

During the last years much effort was put into development of service robots by many researchers around the world (e.g. in the RoboCup@Home league¹). The motivation for these endeavors is the vision that in the future service robot should belong to everyday household appliances. The benefits of having robotic helpers at home are manifold: Not only would they take over annoying and tedious household tasks, but they could also assist disabled or elderly people in helping them with their daily needs [1].

Although many research groups are working on this topic, the household robots available so far are made for simple tasks only (e.g. vacuum cleaning or wiping the floor) and still are not completely free of errors. Another key issue are the costs that have to be reduced significantly to make service robots affordable for a broad mass of people.

For a fully autonomous robot that gets along in a typical household, numerous problems remain to be solved. In this paper, we will focus on one typical task, namely grasping, which requires object detection and recognition. We present the software and hardware design of our robots Lisa and GiGo (Fig.1) that successfully participate in the

RoboCup@Home league. Further, we present the object recognition techniques that we apply in our service robots.



Fig. 1. Our service robots GiGo (left) and Lisa (right)

State of the Art

The annual RoboCup@Home competition aims at fostering the development of service robots and is a rapidly growing league of the RoboCup. This section presents approaches of some participating teams that won the RoboCup finals in the last years.

The current world champion in the @Home league is the German team NimbRo from the university of Bonn. NimbRo detects objects

¹ <http://www.robocupathome.org>

through a combination of depth and vision data. 3D surfaces are extracted for the objects from depth data using RANSAC [2]. The remaining data is clustered and regions of interest are formed. In these regions color histograms and SURF-features are extracted to learn and recognize objects.

eR@sers is a Japanese team with members from three research groups from Tokyo and Kyoto [3]. In their approach, objects are detected by analyzing the scene with structure from motion in multiple images which allows to extract the initial object region. In a subsequent step depth and color information are used to create an object probability map. For the object learning SIFT-features and color histograms are extracted. These are also used afterward to recognize the objects.

The object recognition approach used by team b-it-bots from the Bonn-Rhein-Sieg University in Germany is presented in [4]. First, SURF-features are extracted and a visual dictionary is created that clusters the features by similarity. A support vector machine is trained to classify objects according to a prediction model.

To our knowledge all these teams use own robot software architectures that are not publicly available.

Design and Concept of our Robots

This section introduces the hardware and software design of our service robots.

The main robot, Lisa, is built upon a Pioneer 3-AT platform. A 2 degree of freedom (DOF) gripper in the front allows to grab objects from the floor. On top of the platform is a custom built, lightweight aluminum chassis. Lisa is equipped with a laser range finder that is primarily used for navigation and mapping. A camera in the front part of the robot is used to recognize objects that lie on the floor in front of the robot. A RGB-D camera is mounted on top of the chassis on a pan-tilt unit. It serves multiple purposes like object and face recognition, obstacle avoidance, and gesture recognition. In the chassis, below the RGB-D camera, is a screen showing an animated face of the robot. The face can take a happy, angry or sad expression. To enhance further human-robot communication, the movement of the robot's lips shown on the screen is synchronized with its speech. A microphone

serves as input for spoken language and allows Lisa to react directly to commands given by a human. To interact with its environment Lisa is equipped with a 6 DOF robotic arm. It allows to grab objects up to 90 cm in front of the robot's arm and even from the 2 DOF gripper at the bottom. A small laser range finder mounted on the endeffector of the arm allows for more precise gripping.

The second robot, GiGo, can operate either completely autonomous or according to Lisa's orders which are received over wireless LAN. GiGo is built on top of a commercial Roomba vacuum cleaner robot. He is equipped with a laser range finder for navigation and mapping.

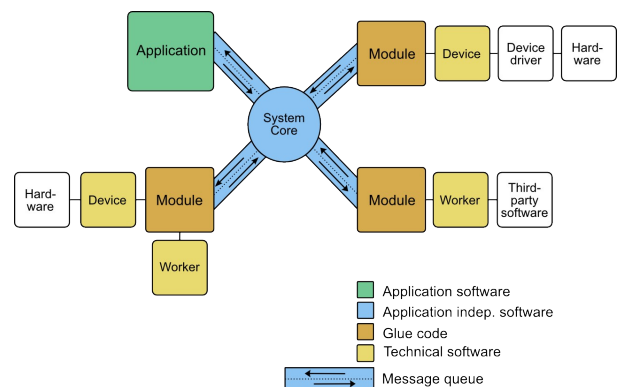


Fig. 2. Software architecture of our robots.

Our software architecture is a modular, message-based system. The main components, called modules, are built around and connected to a generic, application-independent system core (Fig.2). Every module runs in a separate thread. Modules communicate by sending and subscribing to messages. The system core acts as a dispatcher between modules and also takes care of the memory management of the messages. Devices (code responsible for communication with hardware) and Workers (encapsulating algorithms and technical software) are connected to modules. They do not know anything about messages, nor about other parts of the system. Thus, hardware components and algorithms can be easily exchanged without the need to adapt large parts of the system. This software design is therefore perfectly suited for research, as new prototypes and algorithms can be created and tested conveniently.

Recently, we developed an interface that allows our architecture to communicate with the popular robot middleware ROS [5]. This enables us to access the broad variety of

components developed for this architecture, but also to share our algorithms with the rapidly growing ROS community.

More details on the software architecture used for our service robots can be found in [6].

Object Recognition Tasks

In this section we present three different object recognition tasks that typically arise when developing software for service robots. If one would only try to distinguish objects, a visual camera image would be sufficient. However, we also want to be able to manipulate objects. Therefore, the first task we describe is to detect objects using depth data. The second task aims at finding and recognizing previously learned objects. The third task is to comprise color into the recognition step if the recognition is not possible due to insufficient object features.

Object detection. Object detection is performed on a depth image. These kind of images are provided by 3D laser scanners, time-of-flight cameras or by Kinect-like cameras. As objects usually reside on plane surfaces like tables, shelves, or the floor the first step is to find these surfaces. The depth image consists of $n_x \times n_y$ pixels, each containing a 3D point. Here, $\vec{p}_{(x,y)}$ is the 3D point at pixel (x,y) . We use the depth image to compute a normal image where each pixel contains the normal \vec{n} to the corresponding point in the depth image. The normal \vec{n} is computed as

$$\vec{n} = \left\| \left(\frac{\vec{x}_1 + \vec{x}_2}{2} \right) \times \left(\frac{\vec{y}_1 + \vec{y}_2}{2} \right) \right\| \quad (1)$$

with the vectors $\vec{x}_1 = \vec{p}_{(x,y)} - \vec{p}_{(x+1,y)}$, $\vec{x}_2 = \vec{p}_{(x-1,y)} - \vec{p}_{(x,y)}$, $\vec{y}_1 = \vec{p}_{(x,y)} - \vec{p}_{(x,y-1)}$, and $\vec{y}_2 = \vec{p}_{(x,y+1)} - \vec{p}_{(x,y)}$. In the next step, we look for connected components in the normal image that have upwards pointing normals. These components are identified as flat surfaces. Subsequently, the pixels identified as surfaces in the normal image are subtracted from the depth image. The remaining clusters inside the former surface regions are identified as

objects. The results of this step are depicted in (Fig.3).

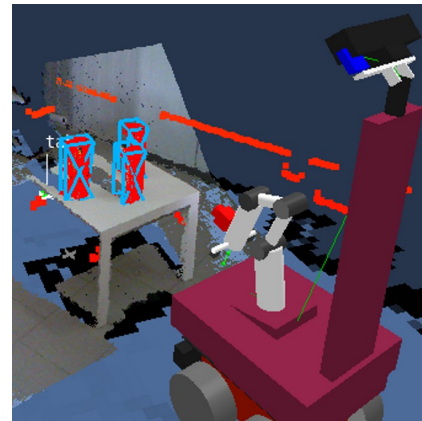


Fig. 3. Detected objects on top of a table (blue boxes).

Object recognition. In the first step we take an image of the background. In the second step, we take several RGB images of the object while rotating it to get different views. The object is separated from its background and the resulting image is used to extract SURF-features. This method allows us to recognize objects reliably even on heterogeneous backgrounds and with partial occlusions. In the recognition step an image of the object is taken. Then, SURF-features are extracted and compared to the stored features. This sparse feature-based approach is explained in more detail in [7].

The original SURF algorithms works on gray level images only. However, different colors can result in the same intensity value, thus leading to information loss. Therefore, we extended the SURF feature detection to the color domain [8]. We analyzed the channels separately for maxima of the determinant of the Hessian, just like the original SURF paper does on gray level images.

Color classification. When dealing with objects of uniform color, the object recognition approach described above can not be used as uniform colors do not provide sufficient SURF-features. In these cases color is the only cue to distinguish objects. We implemented a simple algorithm to provide our robot with the ability to classify colors. The algorithm takes an RGB image as input and converts it to HSV color space. In the first step, we define upper and lower thresholds, to determine whether a gray scale value is sufficient to describe the found color. Low

values for V indicate that the color is black. If V is high and S is low, we have found a white color. In the case that V is between the two thresholds and S is low the color is gray (Fig.4). If none of these three cases applies, the color is determined according to the value of H in the second step.

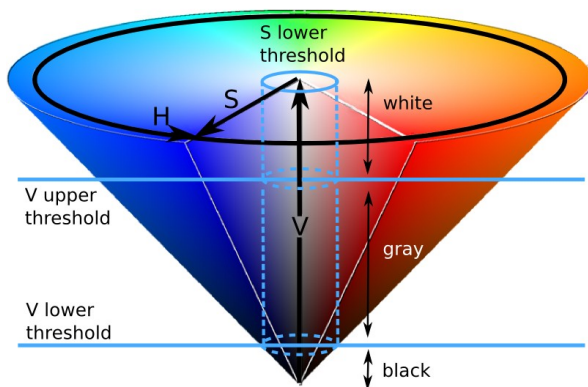


Fig. 4. Color classification.

Results

Our object detection approach works stable: In the RoboCup finals we detected 4 of 5 objects successfully and were able to grasp 3 of them. However, our approach requires the calibration of some parameters. Most important are the parameters in the step of surface detection (determining upwards pointing normals and connected regions). The parameter for the object size is also crucial. If a table is cluttered with many small objects (i.e. below the object size threshold) it will not be recognized as a plane surface.

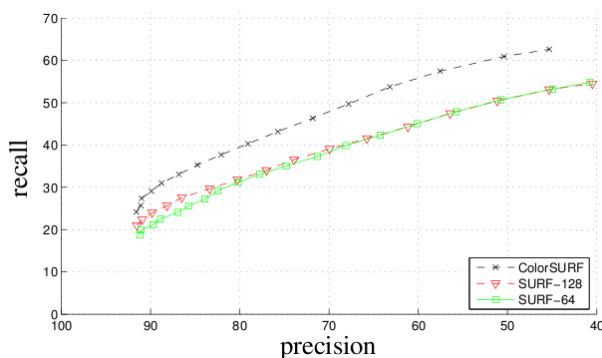


Fig. 5. Comparison between standard SURF and color SURF object recognition [8].

The object recognition yields good results, even on heterogeneous backgrounds. Partially occluded objects are also recognized reliably

[7]. A rotation angle of 30° is optimal in the learning step in order to make the recognition stable. The extension of SURF to the color domain outperforms the original SURF approach (Fig.5), but currently imposes problems due to its increased computation time.

The color classification is a rather simple algorithm. Consequently, the classification results highly depend on the ambient illumination. We currently consider other color spaces and color normalization.

Conclusion

In this paper we have presented three object recognition tasks for service robots. Our proposed solutions not only enable robots to reliably recognize objects, but also to determine their position - a necessary step for object gripping and manipulation.

Due to its modular structure we can use the same architecture for different robots also in RoboCup Rescue.

References

1. Z. Change, Q. Hua. The Domestic and International Research Situation of Humanoid Robot. In: Machine Tool & Hydraulics, vol. 3. - 2006.
2. D. Holz, R. Schnabel, D. Droschel, J. Stückler, S. Behnke. Towards Semantic Scene Analysis with Time-of-Flight Cameras // RoboCup 2010: Robot Soccer World Cup XIV. - 2011. - P. 121-132.
3. H. Okada, T. Omori, N. Watanabe, T. Shimotomai, N. Iwahashi, K. Sugiura, T. Nagai, T. Nakamura. Team eR@sers 2011 in the @Home League Team Description Paper. - 2011.
4. C. Mueller, N. Hochgeschwender, P. Ploeger. Towards Robust Object Categorization for Mobile Robots with Combination of Classifiers // RoboCup Symposium, Istanbul. -2011.
5. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng. ROS: an open-source Robot Operating System // ICRA Workshop on Open Source Software. - 2009.
6. S. Thierfelder, V. Seib, D. Lang, M. Häselich, J. Pellenz, D. Paulus. Robbie: A Message-based Robot Architecture for Autonomous Mobile Systems // Informatik 2011. - 2011.
7. P. Decker, S. Thierfelder, D. Paulus, M. Grzegorzek. Dense Statistical versus Sparse Feature-based Approach for 3D Object Recognition // Pattern Recognition and Image Understanding. -2010.
8. D. Gossow, P. Decker, D. Paulus. Extending Surf to the Color Domain // Proceedings of the Fifth European Conference on Colour in Graphics, Imaging and Vision (CGIV). - 2010. - P. 215-221.