In Proc. of the 7th German Conference on Robotics (ROBOTIK), Munich, Germany, May 2012

Shape-Primitive Based Object Recognition and Grasping

Matthias Nieuwenhuisen, Jörg Stückler, Alexander Berner, Reinhard Klein, and Sven Behnke

Computer Science Institute, University of Bonn, Germany

Abstract

Grasping objects from unstructured piles is an important, but difficult task. We present a new framework to grasp objects composed of shape primitives like cylinders and spheres. For object recognition, we employ efficient shape primitive detection methods in 3D point clouds. Object models composed of such primitives are then found in the detected shapes with a probabilistic graph-matching technique. We implement object grasping based on the shape primitives in an efficient multi-stage process that successively prunes infeasible grasps in tests of increasing complexity. The final step is to plan collision-free reaching motions to execute the grasps. With our approach, our service robot can grasp object compounds from piles of objects, e. g., in transport boxes.

1 Introduction

Object manipulation is a key capability in many industrial and service robotics applications. We present a new framework to grasp objects composed of shape primitives like cylinders and spheres. These shape primitives are used for both object recognition and grasp planning in an integrated and efficient way. With the proposed approach, our service robot can grasp object compounds from piles of objects, e. g., in transport boxes (see **Fig. 1**).

In the computer vision community, research on the recognition of parts can be traced back to work of Thomas Binford [15]. This early work is based on the use of generalized cylinders for segmentation and recognition. Later work also considered different kinds of shape primitives. A common approach is to hypothesize occurrences of shape primitives and more complex compounds which are subsequently verified against image or range data [4, 7, 6, 11, 12]. These approaches are based on involved and relatively time-consuming segmentations in the image domain (either intensity or range images). In contrast, in our approach we extend a novel robust shape primitive decomposition technique [16] that is able to operate directly on 3D point clouds and is efficient enough to deliver results in a matter of a few seconds even on large, high-resolution point datasets. The method is based on classical RANSAC [8] but employs novel sampling strategies which are responsible for the improved efficiency.

In order to recognize parts composed of shape primitives, we match detected primitives using a probabilistic graph-matching approach to object models. We achieve fast detection of objects by extending the flexible graph-matching method of [17], which can handle optional and repetitive object components, with probabilistic matching strategies were recently proposed for symmetry detection [2, 3].

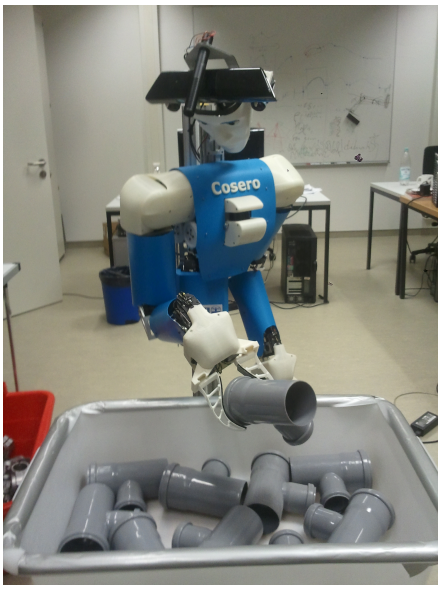

Figure 1: Cognitive service robot Cosero grasps an object from a transport box.

For the perceived objects, we plan feasible grasping motions for a 7-DOF manipulator. In general, grasp planning considers the problem of selecting feasible grasps from the specifications of the object to grasp, the robot kinematics, and the surrounding scene. A variety of approaches determine contact points for each finger for precision fingertip grasps [5, 9]. In contrast, power grasps that exploit contacts with finger surfaces and the palm may yield much more stable grasps. To generate a rich set of fingertip and power grasps, pregrasp shape approaches have been proposed [13, 19].

In previous research, we developed an approach to grasping objects from a table top that is based on sampling feasible grasps on segmented point clouds and fast, but con-

servative, collision check heuristics [10]. In this paper, we present an efficient pregrasp shape approach to plan grasps for objects that are composed of shape primitives. In order to gain efficiency, we plan grasps in a multi-stage process in which we successively prune infeasible grasps in tests with ascending complexity. We plan reaching motions with an efficient multi-resolution sampling-based method.

2 Shape Primitive Detection and Object Recognition

Our method for 3D object recognition is based on subgraph matching: We convert both the searched object and the scanned scene to an annotated graph. Nodes of the graph are instantiated for simple shapes (spheres, cylinders, planes) detected by an extended algorithm that is based on Schnabel et al. [16]. Edges connect those simple shapes which are neighbored in space and also store the relative pose of the primitives. For example, a box might consist of six planar parts, intersecting at 90° angle.
We localize objects in a scene by identifying parts of our search graph in the graph of the scene. Using the established graph correspondences, we calculate a rigid transformation to the assumed position and orientation of the object. Finally, we verify this hypothesis with the original CAD model positioned by this transformation. We generate new object models in advance using a CAD model of the object.

2.1 Object Model Generation

The object we are looking for in the 3D point cloud is given as detailed, exact 3D CAD model. In a first step, we use our primitive detection method on this clean data. Here, we set detection tolerance parameters very tight and get a clean set of shape primitives representing the object and a graph describing the relation of neighboring parts. Please note that for all of these primitives and graph edges we know the exact shape parameters, e. g., the radii for cylinders and spheres or angles between planes.

2.2 Object Recognition

In the following, we describe every part of our object recognition pipeline in detail.

Figure 3: Model generation: Primitive detection and description of spatial relations.

Shape primitive detection: In a preprocessing step, we rapidly calculate surface normals and remove outliers in the data. Next, we detect shape primitives and establish the graph of spatially close primitives.
In order to further increase robustness and performance of the primitive shape detection, we introduce the detection of primitives with predetermined parameters. Since the objects to be identified in our setup are known in advance, the correct parameters are read from the input CAD model.
Graph matching: We efficiently create the annotated shape graph of the scanned point cloud and apply our subgraph matching approach following ideas of [17]. We first choose a random start edge of the query graph of the searched object and find similar edges in the scene. We compare edges by their shape attributes and compute a score for each match. This score is zero if the edge types of primitives do not match and is best if all attributes of the edge and shapes are similar. We go through the list of all similar edges, from best to worst, and for each edge, we try to expand the match to adjacent edges in the query graph and the scene graph simultaneously using our comparison method for edges. We do so until the whole query graph is fitted — or no more corresponding edges can be found. This gives us a number of partially-matched graphs which we rank by the sum of the score of all matched edges. For the best ones, we compute a transformation matrix that encodes the estimated position and orientation of the object in the scene. Then, we use this transformation to verify our graph hypothesis.
Computing position and orientation: We compute a relative 6-DoF transformation towards a common reference shape position and orientation in our object model. Every primitive shape correspondence determines some of these degrees of freedom, for example a sphere-to-sphere correspondence completely characterizes the translation, but does not help at all to determine the rotations. Hence, in the matched partial graph, we use the combination of the matched primitives to compute all unknowns.
We take special care for self-symmetric objects: For symmetric objects we can take any transformation around the self-symmetry axis and compute a valid transformation. It is possible to detect all self-symmetries in the model generation phase (e.g., see [2]).
Graph matching verification: We apply the transformation to the CAD model and check for sufficient overlap with the scanned points. This way, we can also test points where no primitives have been detected. In addition, we verify that no scanned points should be inside the volume of the CAD model. This makes it possible to identify wrong matches that might occur in previous steps. If we can confirm the transformation, we improve its accuracy by employing the ICP method of Mitra et al. [14] to register the CAD model to the points supporting the matched primitives.
Final result and visibility ranking: The previous steps result in transformations to all recognized instances of the object of interest. We estimate the best graspable object

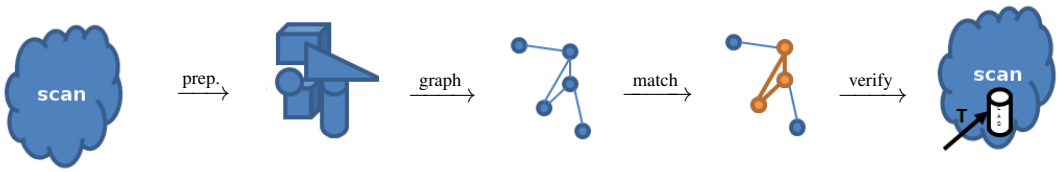

Figure 2: Object recognition: Input scan - fast preprocessing, primitive detection - abstract graph generation - match, transformation estimation and verification.

by ranking our matches according to the number of visible points. The ranked poses of all objects are sent to the grasp planner. We also compute a background point cloud for grasp selection and collision avoidance during motion planning. It consists of all points not in the support of any object.

3 Grasping of Shape Primitive Compounds

We investigate efficient solutions to grasp and motion planning in order to achieve fast planning and short delays between object perception and motion execution. In our approach, we first determine feasible, collision-free grasps at the object. We rank these grasps and find the best grasp that is achievable by a collision-free reaching motion.

3.1 Multiresolution Height Map

To speed up collision checking and allow for dynamically adjusted safety margins between the robot and its environment we employ a multiresolution height-map based on ideas from [1].

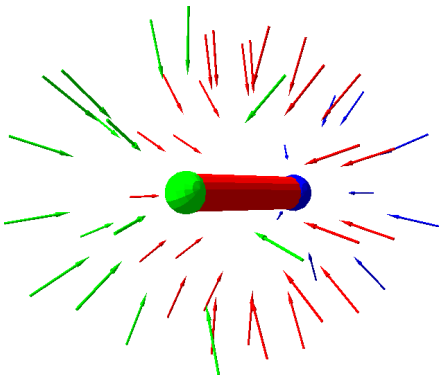

Figure 4: For each shape primitive in an object compound, we sample grasps according to the parametric description of the shape primitives. For the grasps, we determine pre-grasp poses (visualized as arrows pointing in forward direction of the gripper; color codes correspondence to shape primitives). We discard grasps that are in collision within the object.

The grid consists of multiple object-centered grids with different resolutions. With increasing distance to the object, the grid resolution decreases (see **Fig. 5**). This models the trade-off between large safety margins and the need be able to grasp an object in a cluttered environment.
More formally, the environment is discretized into a square $M \times M$ 2-dimensional grid. Recursively, a grid is embedded into the inner part $\left[\frac{M}{4} : \frac{3M}{4}\right] \times \left[\frac{M}{4} : \frac{3M}{4}\right]$ of the grid at the next coarser level. The cell area of the inner grid is a quarter of the cell area of the outer grid.
Each grid cell contains the maximum height value, discretized with the corresponding resolution, measured inside the cell.

3.2 Grasp Planning

For grasp planning, we find feasible, collision-free grasp postures on the object to grasp. We define a grasp as a tuple containing

- the final pose of the gripper when grasping the object (the grasp pose),
- the closure of the gripper (according to object width),
- an initial pose of the gripper (the pre-grasp pose) for approaching the grasp pose, e. g., in a distance of 10 cm to the object,
- a score encoding preferences for certain grasps, e. g., grasping cylinders instead of spheres, and preferring, as the grasp pose, the central part of the cylinder.

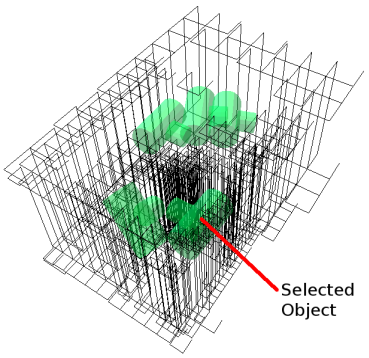

Figure 5: Multiresolution height-map for fast grasp selection and collision checking.

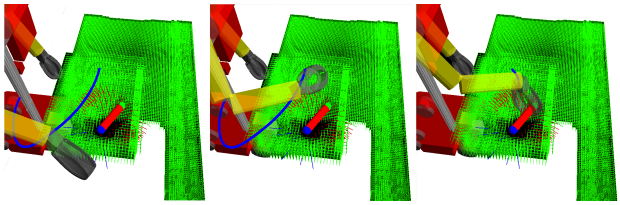

Figure 6: Planned end-effector trajectories for grasping a compound object from a transport box. The trajectories for the approach (left and middle) and grasp (right) phases are shown as blue lines.

In order to gain efficiency, we plan grasps in a multi-stage process that successively prunes infeasible grasps in tests with increasing complexity:
In the first stages, we find collision-free grasp poses on the object, irrespective of the pose of the object and not considering its scene context (see **Fig. 4**). These poses can be pre-calculated efficiently in an off-line planning phase. We sample grasp poses on the shape primitives. From these poses, we extract grasps that are collision-free from pre-grasp pose to grasp pose according to fast collision-check heuristics.
During on-line planning, we examine the remaining grasp poses in the actual poses of the objects to find those grasps for which a collision-free solution of the inverse kinematics in the current situation exists.
Grasps from below the object are considered as infeasible and filtered first. Next, we test grasp and pre-grasp positions against our height-map. The remaining grasps are ranked first by the object ranking and second by the grasp ranking. Collision-free inverse kinematics solutions are searched for the grasps in descending order. If a valid solution was found, we employ motion planning to find a trajectory. If none was found, we continue with the grasp evaluation.
We evaluate the score of the resulting grasp poses and input the grasps to the motion planning method to find the best collision-free grasp.

3.3 Motion Planning

Our grasp planning module finds feasible, collision-free grasps at the object. The grasps are ranked according to a score which incorporates efficiency and stability criteria. The final step in our grasp and motion planning pipeline is now to identify the best-ranked grasp that is reachable from the current posture of the robot arm. We solve this by successively planning reaching motions for the found grasps (see **Fig. 6**). We test the grasps in descending order of their score. For motion planning, we employ LBKPIECE [18]. We split the reaching motion into multiple segments:

- moving the endeffector over the transport box,
- reaching the pre-grasp pose,
- and finally grasp.

This allows for a quick evaluation if a valid reaching motion can be found by planning in the descending order of the probability that planning for a segment will fail.

4 Experiments

4.1 Object Recognition

We tested our object recognition approach using a Kinect sensor. **Fig. 7** shows one result for an object consisting of a cylinder and two spheres. Our method is able to identify the object of interest and selects the topmost object to provide its position and orientation to grasp planning.

4.2 Grasp and Motion Planning

We evaluated our grasp and motion planning pipeline in a physical simulation of the robot which allows for reproducible experiments. We placed an object randomly in a transport box and measured the time for planning and the success rate. In ten runs, our approach requires in avg. 14.9 sec to choose a non-colliding grasp that is within the workspace of the robot. We then plan reaching motions to choose a reachable grasp within 2.45 sec in average. Note, that multiple reaching motions may have been evaluated. A successful reaching motion is planned in 0.45 sec on average. In all test runs, the robot succeeded to grasp the object.
As a second test, we let our robot Cosero clear a transport box with ten objects (see Fig. 1). The used object was a pipe connector. **Tab. 1** shows timings, the number of detected objects per run, and the number of trials needed. In average our approach required 5 sec to detect objects, 19.9 sec to choose a grasp, and 3.8 sec to plan a valid reaching motion. This includes one run, where the robot aborted the execution of a planned motion and replanned for the other arm (see Run 5 in Tab. 1).

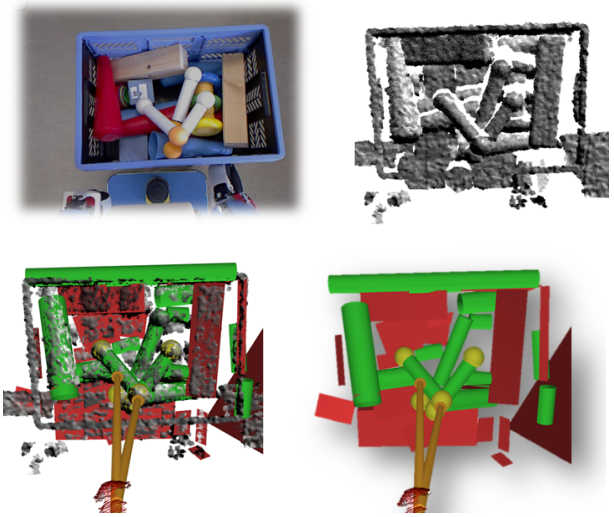

Figure 7: Top: image of the transport box (left) and acquired point cloud (right). Bottom: detected object (left) and primitives only (right).

Run	Detections	Selection	Planning	Trials
1	4 / 10	4.0	1.2	1
2	10 / 9	16.6	2.9	1
3	8 / 8	33.7	3.0	1
4	5 / 7	28.1	6.8	1
5	4 / 6	48.2	9.3	3
6	4 / 5	9.9	4.9	1
7	3 / 4	30.2	3.0	1
8	3 / 3	10.0	1.2	1
9	2 / 2	9.0	3.7	1
10	1 / 1	9.5	1.6	1
Avg.	76%	19.9	3.8	1.2

Table 1: Timings for grasp and motion planning (in sec.) and number of found objects while clearing a transport box. In Run 5, reaching the initially chosen grasp pose was aborted and replanning was necessary, leading to higher evaluation and planning times.

Although the robot cleared the transport box successfully, some issues that need further investigation still exist. E. g., due to noisy measurements with the Kinect sensor, object constellations in the cluttered box are possible, where it is not possible to clearly distinguish to which object a primitive belongs to. This can lead to wrongly connected primitives (see Run 2 in Tab.1). This is not a problem for our approach, as grasps are calculated per primitive, but it slows down the grasp evaluation. Also, the grasp selection still is computational involved and could be sped up by more accurate heuristics.

5 Conclusions

In this paper, we proposed methods to flexibly grasp objects composed of shape primitives. For object recognition, we generate shape compositions from CAD models and perform sub-graph matching with the primitives in the scene to detect and localize objects of interest. We take special care for self-symmetries of the objects.

We solves grasp and motion planning in a multi-stage process with tests of increasing complexity. We divide grasp planning in an off-line and an on-line planning stage: In the off-line phase, we examine the feasibility of grasps irrespective of the actual situation of the object in the scene. In the actual scene, these grasps are further evaluated for collisions with the environment and reachability by the robot.

In experiments, we could demonstrate that our approach is capable of recognizing objects in a transport box. Our method allows to grasp objects in many situations with only short planning times.

In future work, we will consider view-pose planning to further improve the recognition of complex parts in piles of objects. We also investigate the learning of new object models from simple demonstrations to the robot.

Acknowledgement

This research has been partially funded by the FP7 ICT-2007.2.2 project ECHORD (grant agreement 231143) experiment ActReMa.

References

[1] Sven Behnke. Local multiresolution path planning. *Robocup 2003: Robot Soccer World Cup VII*, pages 332–343, 2004.

[2] A. Berner, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel. A graph-based approach to symmetry detection. In *Proc. of the IEEE/EG International Symposium on Volume and Point-Based Graphics*, 2008.

[3] M. Bokeloh, A. Berner, M. Wand, and H.-P. Seidel. Symmetry detection using line features. In *Proc. of the Eurographics Computer Graphics Forum*, 2009.

[4] R. C. Bolles and P. Horaud. 3DPO: A three-dimensional part orientation system. *International Journal of Robotic Research*, 1986.

[5] C. Borst, M. Fischer, and G. Hirzinger. A fast and robust grasp planner for arbitrary 3D objects. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1999.

[6] S. Dickinson, A. Pentland, and A. Rosenfeld. From volumes to views: An approach to 3-D object recognition. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 1992.

[7] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research*, 1986.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[9] R. D. Heester, M. Cetin, C. Kapoor, and D. Tesar. A criteria-based approach to grasp synthesis. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1999.

[10] Jörg Stückler, Ricarda Steffens, Dirk Holz, and Sven Behnke. Real-time 3D perception and efficient grasp planning for everyday manipulation tasks. In *Proc. of 5th ECMR, 2011*.

[11] Whoi-Yul Kim and Avinash C. Kak. 3-D object recognition using bipartite matching embedded in discrete relaxation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1991.

[12] J. Krivic and F. Solina. Part-level object recognition using superquadrics. 2004.

[13] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2003.

[14] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. Registration of point cloud data from a geometric optimization perspective. In *Symp. Geometry Processing*, 2004.

[15] R. Nevita and T. O. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 1977.

[16] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*, 2007.

[17] R. Schnabel, R. Wessel, R. Wahl, and R. Klein. Shape recognition in 3D point-clouds. In *Proc. of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2008.

[18] I. A. Sucan and L. E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *Algorithmic Foundation of Robotics VIII (Proceedings of Workshop on the Algorithmic Foundations of Robotics)*, 2009.

[19] Z. Xue, A. Kasper, J. M. Zoellner, and R. Dillmann. An automatic grasp planning system for service robots. In *Proc. of the 14th Int. Conf. on Advanced Robotics (ICAR)*, 2009.