GPU-based Power-Grasp And Placement Planners For Unknown Environments

Johannes Baumgartl, Universität Bayreuth, johannes.baumgartl@uni-bayreuth.de, Germany
Dominik Henrich, Universität Bayreuth, dominik.henrich@uni-bayreuth.de, Germany

Abstract

A Personal Robot should be able to handle unknown objects in unknown environments. For a manipulation task the question of what to do with an object once it has been grasped, is one of the most essential ones besides the grasping task itself. Moreover, the planning time should be at least as fast as the time the robot needs for its grasping/placing motions. We propose a combination of a fast grasp and placement planner that deals with sensor-modelled objects. They share novel pose computation steps that reduce the amount of computational intensive collision tests. By means of experiments we evaluate the computation time for the pose computation. Furthermore, we present the qualitative results of experiments showing accurate grasps and object placements.

1 Introduction

A flexible and skilled Personal Robot needs to perform various pick-and-place tasks in different domains like cleaning worktops in households, laboratories, or workplaces in general. Therefore, a Personal Robot should be able to elaborate how and where to place objects.

However, there is a wide range of situations, objects, and environments that affect this challenging task. Hence, the robot must be able to handle unknown objects, whose surface models are provided using sensor data. The resulting models are typically incomplete and noisy. Furthermore, we assume that a multi-finger gripper is sufficient to react flexible on different situations.

When the robot has to place an object in an environment, the robot should be able to plan how to grasp the object using a multi-finger gripper and where to put it down. Two algorithms calculating stable grasps and placements help the robot to perform successfully.

What this paper offers, is a solution to plan accurate power-grasps and object placements in the environment, while having only the geometric information available, gained by a depth camera. A special focus is put on the question of how to compute candidate configurations quickly and how to optimise them locally in case of static instability. Furthermore, we set the grasp planning in relation to placement planning.

The remainder of this paper is organized as follows: In Section 2 we outline the related work. In Section 3 we introduce two base algorithms used for the grasp and placement planner in Section 4. Furthermore, we evaluate the performance and quality of both planners in Section 5 and discuss the results in Section 6. Section 7 concludes this paper.

2 Related Work

The related work is composed of two main topics: grasp planning and placement planning.
The field of object placement planning in the context of pick-and-place applications [12] was not focused on as such much as the grasp planning during the past years.
There are several related approaches targeting subproblems of the whole placement planning problem, like computing the upright orientation of objects [5] or finding good planar regions on the placement area [17]. Since these works do not consider finding a stable pose for the object on the placement area, they are not regarded any further.
An approach that targets placement planning in the sense of packing is [1]. Since this approach uses only base areas of the unknown objects and known planar placement areas, it is not applicable for general placement areas and 3D objects, which we are interested in. Another approach considering known objects with an unknown placement area is introduced in [8]. Here, object placement considers the footprint of the contact area. The target location of the object is given as input by a human. Both approaches only consider planar placement areas. Contrary to that, we generalise from planar to complex placement areas.
One of the first overall solutions for placing new objects in complex placement areas is introduced by Jiang et al. [10] and [11]. They outline a learning based framework that requires an object database together with semantic labels. Our work is different from that in so far as we only use geometric information about the object and do not use any kind of learning step.

A detailed overview of grasp synthesis in general is given by [16] and [3]. Since it is sufficient for a service robot to grasp an object stable enough for the given task, we concentrate on approaches that consider a short computation time together with an exact grasp. To cope with these challenges, related approaches apply different heuristics, which differ a lot.

A common solution is to approximate the object by a normal distribution [13], geometrical primitives [9], and super quadrics [6]. These approximations are partly too coarse for an exact grasp or their computation is too expensive. Likewise, approaches using the topology of objects [15] have a high computation time.
We focus on a local sampling approach to build a data-parallel algorithm with a special focus on the positioning of the gripper base. Hereby, we introduce the idea that the positioning of the gripper base is strongly related to the general problem of placement planning of objects in non-planar environments.

3 Base Algorithms

Both algorithms rely on two base algorithms for two complex object models (e.g. gripper fingers or household objects): First, a computation for the translation along a given direction and, second, a rotation angle computation around a given rotation axis. As an object model, we use a 3D surface representation of convex polygonal patches reconstructed from a captured point cloud. The result of our computations is either a translation matrix or a rotation matrix.
Different possibilities to implement these base algorithms exist. Iterative approaches and physical simulation frameworks move one object, step by step, towards an other object including a collision check in every step. The computational effort is comparatively high to our approach, caused by the amount of collision checks. Our approach directly computes the translation and rotation, so that the two objects are in contact with each other. Hence, we only need to perform one collision check to get the contact points.

3.1 Translation Computation

Our translation computation, along a given direction d, uses the vertices v_i of the first object and the polygonal surface patches S_j of the second object. We use a ray casting technique based on the kd-tree restart idea. We locate the leaf of the tree containing v_i and process all S_j of this leaf. Afterwards, we trace the ray along d until it intersects the bounding box of the current leaf. Now, we compute the first intersection of the ray with this bounding box and use the resulting point as the new starting point v_i' of our ray. Please note that v_i and v_i' are not located in the same leaf. This procedure is continued until the kd-tree has been completely traced. For an example see Figures 1(a) and 1(b).

3.2 Rotation Computation

For the rotation computation between two objects around a given axis resulting in a contact between the two objects, we have to distinguish two contact situations: point-surface contacts and edge-edge contacts.
For the point-surface contact, the points p_i of the first object are rotated around the axis a (computed using the current set of contact points) and define circles $C(p_i, a)$ in the 3D space. Rotation angles for point-surface contacts $\angle_a(p_i, p_{ij}^C)$ are between the p_i's and the intersection points p_{ij}^C of $C(p_i, a)$ and all surface patches S_j of the second object. The edge-edge rotation computation is more complex. All edges e_i of the first object are rotated around a. Therewith surfaces of revolution $H(e_i, a)$ are constructed. Using the intersection points $p_{ij}^H := H(e_i, a) \cap e_j$ we construct planes $P(p_{ij}^H, a)$ through p_{ij}^H and a as normal. The rotation angle $\angle_a(p_{ij}^P, p_{ij}^H)$ is between a intersection point $p_{ij}^P := P(p_{ij}^H, a) \cap e_i$ and the previously computed intersection point p_{ij}^H. Finally, the over all rotation angle for the two object is: $\min_{ij}\{\angle_a(p_i, p_{ij}^C), \angle_a(p_{ij}^P, p_{ij}^H)\}$. An example rotation is from the pose in Figure 1(b) to the pose in Figure 1(c).

3.3 Implementation

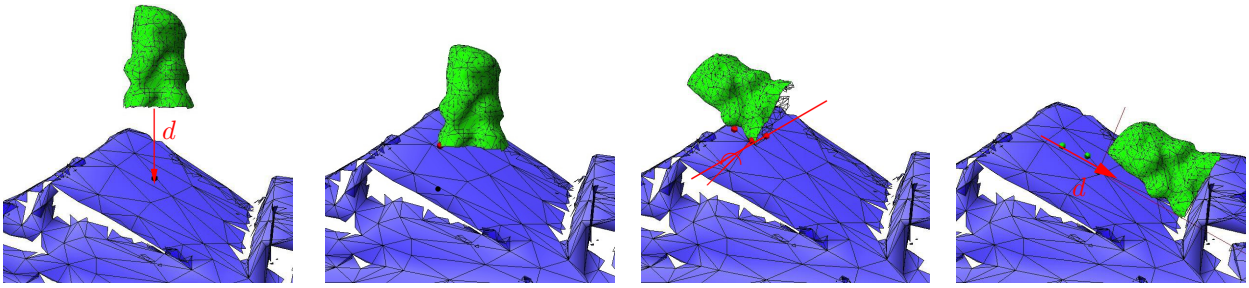
In this section we outline our implementation of the GPU-based and CPU-based implementation and go into the selection of relevant edges and surface patches from an object model.
As mentioned in Section 3.1, we use a kd-tree in combination with the restart idea for efficient selection of relevant parts of an object. In particular, we trace a plain ray during the translational computation and we trace a circular arc during the rotational computation.
One challenge for the implementation[1] is to achieve a good occupation of a GPU. Hence, we batch the computations by type (translation, rotation and collision) to execute them concurrently. Since the data exchange between host and device requires some time, we launch several CPU threads each with one Cuda steam. Our GPU kernels act on a single vertex or edge dependent of the computation.

4 Planner

Using the introduced base algorithms, we design a grasp planner and a placement planner. Notably, the placement planner can place objects on complex, non-planar environments. Hence, it can also be used for online bin packing and dense online placement on planar surfaces. The grasp planner is suitable for complex multi-finger grippers.
In the following, we will introduce the basic steps of our planners starting with the placement planner and followed by the grasp planner.

Placement Planner Figure 1 outlines the main steps of our geometrical placement planner. The algorithm's input is only the two sensor-modelled objects and the direction of gravity. The output of our algorithm is a pose (position and orientation) for the object and several intermediate ones that describe constructive steps to reach a stable goal pose.

[1] Programming languages: NVIDIA Cuda on the GPU and C++ on the CPU



(a) Location of the pre-orientated object above the selected and visible floor point (orange).

(b) Established first contact (red) between object and placement area.

(c) A rotation computation based on the contact points from (b) around the red axis.

(d) A stable pose after the local optimization step.

Figure 1: Examples for the four main steps of the pose computation for the object (green) relative to the placement area (blue) including contact points and rotation axis (red) as well as current translation direction d.

The algorithm has three pose computation steps and a quality rating and optimization step: First, the object and the placement area are provided. Furthermore, we compute initial positions on the placement area. Second, we position the object above those initial positions (Figure 1(a)) and translate the object along the direction of gravity towards the placement area (Figure 1(b)). The result is that object and placement area are in contact with each other. Third, we compute the contact points and based on these decide, whether a translation, rotation ,or quality rating follows up. Furthermore, we compute the necessary translation direction, rotation axis and rotation point using the contact points (Figure 1(c)).

Our quality rating includes an optimization step in case of an unstable pose (Figure 1(d)). Please note, since a rotation can be applied clockwise or counter-clockwise, a sequence of computations for one initial position evaluates to a binary tree. The computation ends, if the tree has reached the maximum allowed depth or the current pose is considered stable.

The quality rating in combination with the local optimization step is based on a heuristic. Since, we deal with unknown objects, we do not have the required input, especially no force values, to apply well known approaches like the FEA analysis, mechanical equilibrium analysis or stability measures from the field of grasp planning.

After every step of our algorithm we have the information about the position and normal direction of every contact point. Additionally, we estimate the center of mass of the object as average value of the coordinates of the mesh vertices.

Our heuristic is grounded on the planar case known form the field of engineering mechanics. If all contact points are located on a common plane, we project the center of mass of the object onto this plane with respect to the direction of gravity. In case this projection is inside of the 2D convex hull of the contact points the pose of the object is stable. In case of instability, the failure direction is the vector from the center of the convex hull to the projected center of mass.

We extend this method, so that we are able to analyse non-planar contact point arrangements. Therefore, we estimate the plane containing the largest convex hull, based on contact points being located on that plane. These points have to be lower than the center of mass of the object. Now we apply the planar case on this set of contact points. In case of instability we use the failure direction as new direction of gravity and reapply the outlined contact point selection.

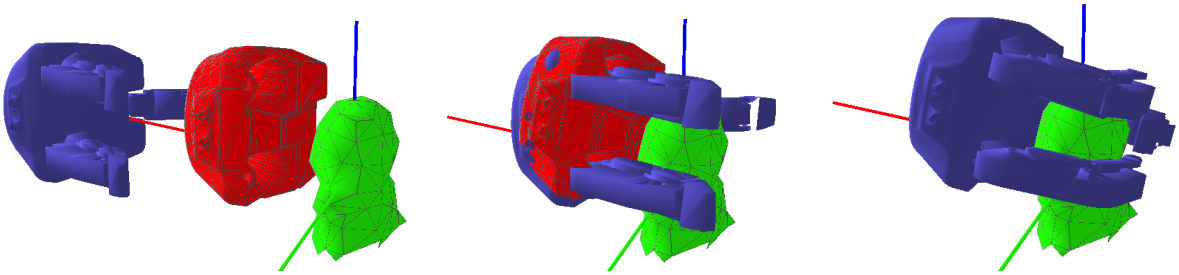
If the current pose is still not considered as stable, we compute a new translation direction and rotation axis, to continue with our pose computation. The new translation direction is the current failure direction. Hence, we compute the length of the translation along this direction. In case there is no significant translation possible we rotate the object using a rotation axis normal to the failure direction. We use the furthest collision point with respect to the failure direction as rotation point.

Our quality rating is also based on the planar case assumption. It is a sum of two quotients. The first one is between the height of the object and the area of the convex hull. The second one is a measure of the centricity of the projected center of mass with respect to the convex hull. A detailed description of the placement planner, including our quality rating, can be found in [2].

Grasp Planner Besides finger closing using our base algorithms, there are more steps to do during grasp planning. We focus on power grasps for under-actuated multi-finger grippers. We distinguish between finger joints that reposition the finger with respect to the gripper base, but do not establish a clamp and joints that fasten the object. Hence, the positioning of the gripper base relative to the object is our first step. For stable power grasps, the base has to be in contact with the object.

The positioning of the gripper base is strongly related to a placement of an arbitrary object, since the degrees of freedom of the fingers are not yet considered. Furthermore, for power grasps it is mandatory that the position of the gripper base with respect to the object is stable enough, so that the relative pose between gripper base and object does not change during the finger closing.

During the positioning of the gripper base we assume, that the gripper fingers are open. Furthermore, we use not only the surface mesh of the gripper base, but also the models of the open fingers. For the approach direction of the gripper we use the centre of every surface patch



(a) Initial pose of the gripper (blue) and translated pose of the gripper base (red) estabilshing the first contact with the object.

(b) Fist contact between gripper (blue) and object and rotated pose of the gripper base (red)

(c) Final optimized gripper pose and finger configurations.

Figure 2: Example for the three main steps of the grasp planning. The coordinate system represents the main axis of the object (green).

and the main axis of the object to place and orientate the gripper base.

We align the negative normal of the surface patch with the approach axis of the gripper coordinate system, assuming that the gripper model follows the NSA coordinate system convention. Furthermore, we position the origin of the NSA coordinate system on the ray defined by the center of the surface patch and its normal. Hence, only one degree of freedom is left for a well-defined start orientation, the rotation around the a-axis of the gripper. This gives us the opportunity to align the rotation axis of the first joint of a fixed finger (without a special joint) of the gripper with the main axis of the object.

To establish contact between the gripper base and the object we utilize the first steps of our placement planner. We use the approach direction of the gripper coordinate system as a first translation direction to establish the fist contact. After the evaluation of the contact points, there are two cases. The first one is that gripper and object share more than three contact points not located on a common line. The second cases is that there are less contact points. In case all contact points share a common line, we rotate the gripper around that line, so that the origin of the NSA coordinate system gets closer to the object. In case there is only one contact point, the rotation axis is the cross product of the a-axis and the position vector of the contact point in the NSA coordinate system. The rotation is performed around the contact point, so that the origin of the NSA coordinate system again gets closer to the object.

Now it comes down to the joints of the finger that only adjust the position and orientation of a finger relative to the object, but do not close the finger to achieve a fixture. These joints increase the available variants for finger configurations dramatically. Eigen grasps [7] or hand taxonomies [4] can be used to overcome this issue.

The angles for the remaining finger joints are computed iteratively, starting at the joint next to the gripper base, using our rotation computation (Section 3.2).

We evaluate computed gripper configurations including the gripper base pose and the angles for the finger joints using the quality rating introduced in [14].

[2]VCG Library: www.vcg.isti.cnr.it

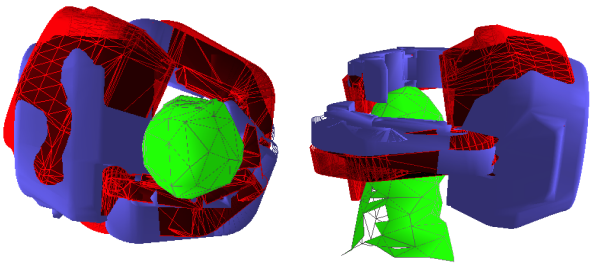
5 Experiments

In this section we introduce our object model, a qualitative evaluation of both planners, and computation time measurements. All computations were performed on an AMD Opteron CPU and a Nvidia GTX Titan GPU. Furthermore, we use a friction coefficient of $\mu = 0.2$ for the quality measure. The initial positions of the placement area are equally spaced with a distance of 0.09 m. Their amount varies with respect to the size of the placement area.

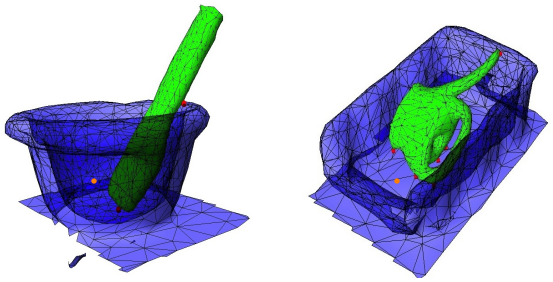
Object Model We now outline our object modelling pipeline used for the experiments. As sensor we use a hand-held depth camera. The relative registration between two camera images is accomplished by using the Iterative Closest Point algorithm. Since we know the initial position of the camera we are able to register the reconstructed object model to the robot's coordinate system.

Each point cloud of a depth image is integrated into a regular grid using a signed distance function. We convert this grid-based model into a triangular surface model using the Marching Cube algorithm. As a last step of our pipeline, we simplify these surface models to a user-specified amount of 1000 triangles using the Quadric Error Edge Collapse Simplification[2] with an quality value of 0.3. Placement area and object are modelled independently from each other.

Results and Evaluation Figure 3(a) and 2 show sample grasps including the grasp with and without the optimized gripper base position. Generally, the quality rating of a grasp improves with our optimized gripper base position. However, in some cases the better base position leads to a poorer finger position relative to the object, which leads to a worse over all rating. But a worse quality rating based on the finger position does not lead to a pose change of the object during the finger closing.



(a) Two grasps: Each with the optimized gripper base position (red) and the initalal translated one (blue). The quality improves from 0.18 to 0.2 (left) and from 0.0 to 0.28 (right).



(b) Stable pose of the green objects on the blue placement areas.

Figure 3: Results of the grasp planner (a) and the placement planner (b).

Figure 3(b) and 4 outline selected results of our placement planning. Our planner is able to deal with non-planar and even box-like placement areas and complex non-convex objects. Our quality rating and optimization step are especially well suited to object placements, where the object leans against some parts of the placement area. The initial pose of the object strongly affects the planned pose. Considering the watering can for example, in case its initial orientation has been upright it would just stand upright inside the box. The final pose shown in Figure 3(b) has been reached with a rotated initial orientation in advance of the placement planning.

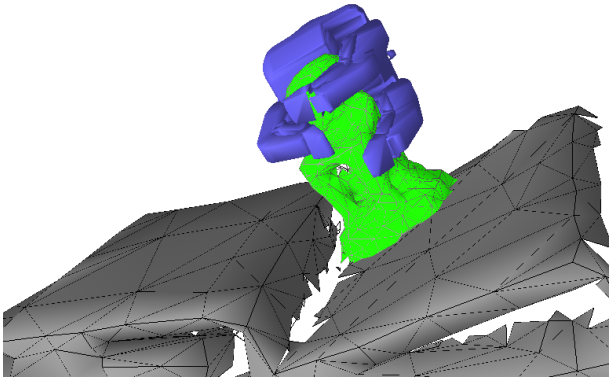


Figure 4: Result of the placement planning followed by a grasp planning of the placed object.

We evaluate the computation time of our experiments with respect to the mandatory parameter, the amount of surface patches of our object model, since both algorithms including our base algorithms strongly rely on this parameter. For one initial position of the object at the placement planning, 32 rotations may occur with a maximum amount of five rotations and for one initial gripper base position 19 rotations may occur for a three finger gripper with three joints for each finger. We have to compute about 32000 rotations and collision point computations for 1000 initial positions during the placement planning. Figure 5 shows the maximum total computation time for all initial positions using the placement area and object displayed in Figure 4. We vary the amount of surface patches from 200 to 100. Please note, that after about a total computation time (including the GPU computations) of 500 ms to 900 ms feasible results are available that result in valid placements. Since the grasp planning has less computation steps, we achieve comparable computation times.

6 Discussion

Our solutions to compute transformation matrices for pose changes differ from well-known static simulations or physic simulations. Since we distinguish between pure translations and rotations, we are not able to change poses using a generic transformation matrix. However, our application does not require a combined rotation and translation matrix. Both components approximate only the real transformation to establish a contact, since we currently do not consider edge-edge contacts during a translation. However, our solution has a major advantage. The most time-consuming computation step in the simulation approaches is the collision check. In our application it is worse, since the objects involved in this step are so close that early-out tests have no effect. Furthermore, if the objects are already in contact with each other and we want to translate or rotate once again, our approach does not require an analysis of the contact points to figure out whether their amount or locations have changed.
Our experiments[3] show that we need about 700 ms to perform all rotation and translation computations. All collision point computations last about 200 ms. To point out the benefit of our approach, let us assume that we want to rotate an object about $45°$ using a static simulation with a step size of $1°$. This results in 45 collision computations for this single rotation. Our approach needs one rotation computation followed up by a single collision point computation. Hence, the break even to our approach is after about 4 to 6 collision point computations or static simulation steps considering the computation time of our components.

7 Conclusions

We introduced GPU-based grasp and placement planners that share the base algorithms (Section 3) and the contact point computation. Both algorithms are data parallel. Hence, a GPU-based implementation is applicable. Additionally, we managed to balance the workload on the GPU. Especially during the placement planning the com-

[3] Computation times for objects with 1000 triangles and 1218 initial poses.

putational effort varies a lot. The placement planner is able to successfully plan object poses in non-planar environments using only geometrical information. Our grasp planner uses our placement planner to position the gripper base before closing the fingers.

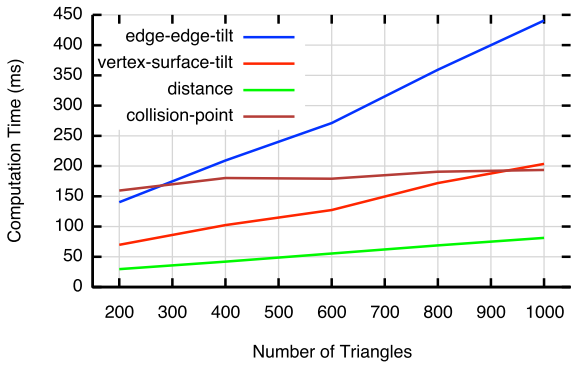


Figure 5: Plot of the maximum computation times of the implemented GPU kernels over the number of surface triangles with 1012 initial poses.

Our base algorithms reduce the amount of collision computations, which are one of the most time consuming components of typical applications. Additionally, the parallel implementation of our base algorithms and the collision point computation enables us to execute concurrently many sample configurations. As a result, we are able to plan exact grasps and placements for sensor-modelled objects in approximately 0.5 sec until good rated configurations are available.
Future work should combine both planners and implement them on a real robot for several pick-and-place operations including a quantitative analysis of both planners.

References

[1] J. Baumgartl and D. Henrich. Fast Vision-based Grasp and Delivery Planning for unknown Objects. In *7th German Conference on Robotics)*, Munich, 2012.

[2] J. Baumgartl, T. Werner, P. Kaminsky, and D. Henrich. A fast, gpu-based geometrical placement planner for unknown sensor-modelled objects and placement areas. In *International Conference on Robotics and Automation*, Hong Kong, 2014.

[3] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-Driven Grasp Synthesis - A Survey. 2013.

[4] T. Feix, R. Pawlik, H. Schmiedmayer, J. Romero, and D. Kragic. A comprehensive grasp taxonomy. In *Robotics, Science and Systems: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation*, pages 2–3, Seattle, 2009.

[5] H. Fu, D. Cohen-Or, G Dror, and A. Sheffer. Upright orientation of man-made objects. *ACM Transactions on Graphics (TOG)*, 27(3):42, 2008.

[6] C. Goldfeder, P. Allen, C. Lackner, and R. Pelossof. Grasp planning via decomposition trees. In *International Conference on Robotics and Automation*, pages 4679–4684, Roma, 2007.

[7] C. Goldfeder, M. Ciocarlie, and P.K. Allen. The Columbia grasp database. In *International Conference on Robotics and Automation*, pages 1710–1716, Kobe, 2009.

[8] K. Harada, T. Tsuji, K. Nagata, N. Yamanobe, H. Onda, T. Yoshimi, and Y. Kawai. Object placement planner for robotic pick and place tasks. In *International Conference on Intelligent Robots and Systems*, pages 980–985, Vilamoura, October 2012.

[9] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann. Grasping known objects with humanoid robots. In *International Conference on Robotics and Automation*, pages 1–6, Kobe, 2009.

[10] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to Place New Objects in a Scene. *IJRR*, 31(9), 2012.

[11] Y. Jiang and A. Saxena. Hallucinating humans for learning robotic placement of objects. In *International Symposium on Experimental Robotics*, Québec City, 2012.

[12] T. Lozano-Pérez, J. Jones, E. Mazer, and P. O'Donnell. Task-level planning of pick-and-place robot motions. *Computer*, 22(3):21–29, 1989.

[13] Alexis Maldonado, Ulrich Klank, and Beetz Michael. Robotic grasping of unmodeled objects using time-of-flight range data and finger torque. In *International Conference on Intelligent Robots and Systems*, pages 18–22, Taipei, 2010.

[14] A. Miller and P. Allen. Examples of 3D grasp quality computations. In *International Conference on Robotics and Automation*, volume 2, pages 1240–1246, Detroit, 1999.

[15] A.r Przybylski and R. Dillmann. Planning grasps for robotic hands using a novel object representation based on the medial axis transform. In *International Conference on Intelligent Robots and Systems*, pages 1781–1788, San Francisco, 2011.

[16] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.

[17] M. Schuster, J. Okerman, H. Nguyen, J. Rehg, and C. Kemp. Perceiving clutter and surfaces for object placement in indoor environments. In *International Conference on Humanoid Robots*, pages 152–159, Nashville, 2010.