Mind the Gap - Robotic Grasping under Incomplete Observation

Jeannette Bohg, Matthew Johnson-Roberson, Beatriz León, Javier Felip,
Xavi Gratal, Niklas Bergström, Danica Kragic and Antonio Morales

***Abstract*— We consider the problem of grasp and manipulation planning when the state of the world is only partially observable. Specifically, we address the task of picking up unknown objects from a table top. The proposed approach to *object shape prediction* aims at closing the knowledge gaps in the robot's understanding of the world. A completed state estimate of the environment can then be provided to a simulator in which stable grasps and collision-free movements are planned.**

The proposed approach is based on the observation that many objects commonly in use in a service robotic scenario possess symmetries. We search for the optimal parameters of these symmetries given visibility constraints. Once found, the point cloud is completed and a surface mesh reconstructed.

Quantitative experiments show that the predictions are valid approximations of the real object shape. By demonstrating the approach on two very different robotic platforms its generality is emphasized.

I. Introduction

Many challenging problems addressed by the robotics community are currently studied in simulation. Examples are motion planning [1], [2], [3] or grasp planning [4], [5], [6] in which the knowledge of the complete world model or of specific objects is assumed to be known. However, on a real robotic platform this assumption breaks down due to noisy sensors or occlusions. Consider for example the point cloud in Fig. 1 that was collected with an active stereo head from a table top scene with several objects standing on it.

Due to occlusions, no information about the backside of objects or about the region behind them is available from pure stereo reconstructed data. There are *gaps* in the scene and object representations. Additionally, noise causes the observed 3D structure of an object or scene to deviate from its true shape. Although the previously mentioned planners might be applied on real robotic platforms, a mismatch between the real world and the world model is usually not dealt with explicitly.

Exceptions are reactive grasping strategies that adapt their behavior based on sensor information collected during execution time [7], [8]. Other approaches aim at predicting unknown parts of the world and plan the robot's course of action based on this [9].

In this paper, we consider the scenario of picking up unknown objects from a table top. Therefore, the robot is confronted with scenes similar to those in Fig. 1. To accomplish grasp and motion planning based on this information, we follow the idea of filling in the gaps in the scene representation through predicting the full shape of each object. We make use of the observation that many, especially man-made objects, possess one or more symmetries. Given this, we can provide the simulator with an estimated complete world model under which it can plan actions or predict sensor measurements.

Fig. 1. Example for a point cloud representing a whole scene merged from different view points. Left: View from the side. Right: View from the top.

The contributions of this paper are (i) a quantitative evaluation of the shape prediction on real-world data containing a set of objects observed from a number of different viewpoints. This is different from related work by Thrun and Wegbreit [10] in which only qualitative results in form of point clouds are presented rather than quantitative results on polygonal meshes. (ii) We reduce the search space for the optimal symmetry parameters through a good initialization. And (iii), we demonstrate the applicability of the predicted object meshes in a service robotic scenario by supporting execution in the real-world with grasp and motion planning in simulation.

The paper is outlined as follows. In the next section, we will motivate the use of the symmetry assumption and explain the method for object shape prediction and polygonal mesh reconstruction. Thereafter, the two experimental platforms are described. In Section IV the simulating environment OpenGRASP is presented. In the experimental section, we show how the proposed prediction mechanism produces valid complete object models and is advantageous for grasp planning and execution.

II. Predicting Object Shape through Symmetry

Estimating the occluded and unknown part of an object has applications in many fields, e.g. 3D shape acquisition,

This work was supported by the EU through the project GRASP, IST-FP7-IP-215821. J. Bohg, M. Johnson-Roberson, X. Gratal, N. Bergström and D. Kragic are with CVAP/ CAS at KTH, Stockholm, Sweden. {bohg, mattjr, gratal, nbergst, dani}@csc.kth.se.
B. León, J. Felip and A. Morales are with the Robotic Intelligence Laboratory at the Department of Computer Science and Engineering, UJI, Castellón, Spain {len, jfelip, morales}@uji.es

3D object recognition or classification. In this paper, we are looking at this problem from the perspective of service robotics and are therefore interested in the advantages of shape completion for collision detection and grasp planning.

Psychological studies suggest that humans are able to predict the portions of a scene that are not visible to them through *controlled scene continuation* [11]. The expected structure of unobserved object parts are governed by two classes of knowledge: i) Visual evidence and ii) completion rules gained through prior visual experience. A very strong prior that exists in especially man-made objects is symmetry.

In [10] it has been shown that this symmetry can be detected in partial point clouds and then exploited for shape completion. The authors developed a taxonomy of symmetries in which *planar reflection symmetry* is the most general one. It is defined as the case in which each surface point P can be uniquely associated with a second surface point Q by reflection on the opposite side of a symmetry plane. Furthermore, in a household environment, objects are commonly placed such that one of their symmetry planes is perpendicular to the supporting plane. Exceptions exist such as grocery bags, dishwashers or drawers.

Given these observations, for our scenario we can make the assumption that objects commonly possess one or several planar symmetries of which one is usually positioned perpendicular to the table from which we are grasping. By making these simplifications, we can reduce the search space for the pose of this symmetry plane significantly. As it will be shown in the experimental section, our method produces valid approximations of the true object shape in very different viewpoints and for varying levels of symmetry.

A. Detecting Planar Symmetry

Since we assume the symmetry plane to be perpendicular to the table plane, the search for its pose is reduced to a search for a line in the 2 1/2D projection of the partial object point cloud. This line has 3 degrees of freedom (DoF): the 2D position of its center and its orientation. We follow a generate-and-test scheme in which we create a number of hypotheses for these three parameters and determine the *plausibility* of the resulting mirrored point cloud based on visibility constraints.

We bootstrap the parameter search by initializing it with the major or minor eigenvector e_a or e_b of the projected point cloud. As shown in Section V, this usually yields a good first approximation. Further symmetry plane hypotheses are generated from this starting point by varying the orientation and position of the eigenvectors as outlined in Fig. 2. In the following, we will describe the details of this search.

1) Initial Plane Hypothesis: The first hypothesis for the symmetry axis is either one of the two eigenvectors of the projected point cloud. Our goal is to predict of the unseen object part. We therefore make the choice dependent on the inverse viewing direction v: the eigenvector that is most perpendicular to v is used as the symmetry plane s.

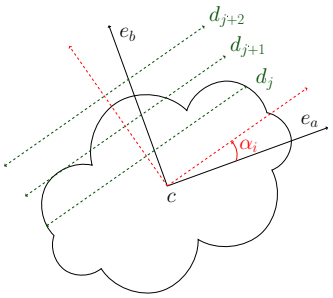

Fig. 2. A set of hypotheses for the position and orientation of the symmetry plane. e_a and e_b denote the eigenvectors of the projected point cloud. c is its center of mass. α_i denotes one of the variations of line orientation along which the best pose of the symmetry plane is searched. The (green) lines at positions d_j to d_{j+2} are three further candidates with orientation α_i.

$$s = \begin{cases} e_a \texttt{ if } e_a \cdot v \leq e_b \cdot v \\ e_b \texttt{ if } e_a \cdot v > e_b \cdot v \end{cases} \tag{1}$$

where e_a and e_b denote the major and minor eigenvectors of the projected point cloud, respectively.

2) Generating a Set of Symmetry Hypothesis: Given this initial approximation of the symmetry plane s of the considered point cloud, we sample n line orientations α_i in the range between $-20°$ and $20°$ relative to the orientation of s. The 2D position of these n symmetry planes is varied based on a shift d_j in m discrete steps along the normal of the symmetry plane. Together this yields a set $\mathcal{S} = \{s_{(0,0)} \cdots s_{(0,j)} \cdots s_{(i,j)} \cdots s_{(n,m)}\}$ of $n \times m$ hypotheses.

3) Mirroring the Point Cloud: Let us denote the original point cloud as $\mathcal{P}$ containing points P. Then given some symmetry plane parameters $s_{(i,j)}$, the mirrored point cloud $\mathcal{Q}_{(i,j)}$ with points Q is determined as follows:

$$Q = R_{\alpha_i}^{-1}(-R_{\alpha_i}(P + d_j - c)) + c \tag{2}$$

with R_{α_i} denoting the rotation matrix corresponding to α_i.

4) Computing the Visibility Score: The simplest source of information about visibility constraints are the binary 2D segmentation masks $\mathcal{O}$ that separate an object from the background. The mirroring process aims at adding points Q to the scene that were unseen from the original viewpoints.

Let us assume a mirrored point cloud $\mathcal{Q}_{(i,j)}$ has been generated, then there are three cases for where a reflected point Q can be positioned. i) If it coincides with a point in $\mathcal{P}$ (the original point cloud), then it supports the corresponding symmetry hypothesis. ii) If Q falls into previously occluded space, it provides information about potential surfaces not visible from the original viewpoint. And finally iii), if Q is positioned into the space that has been visible before, then it contradicts the symmetry hypothesis.

Based on this intuition, the vote $v_{(i,j)}$ consists of two parts. First, we back project each $\mathcal{Q}_{(i,j)}$ into the original image giving us a set of pixels. For all these pixels q that do not lie inside the original object segmentation mask $\mathcal{O}$, we compute the squared distance $\delta_1(q,p)$ to the nearest pixel p in the segmentation mask using the distance transform. Second, for all q that lie within the segmentation mask and have a

smaller depth relative to the camera than the corresponding (occluded) pixel p, we compute the depth difference $\delta_2(q,p)$ between them. Summarizing, the vote $v_{(i,j)}$ is computed as the sum of the expected values of these two distance measures:

$$v_{(i,j)} = \mathop{\mathbf{E}}_{q \notin \mathcal{O}} [\delta_1(q,p)] + \mathop{\mathbf{E}}_{q \in \mathcal{O}} [\delta_2(q,p)]. \quad (3)$$

In case the plane parameters are chosen such that there is a large overlap between the original point cloud $\mathcal{P}$ and the mirrored cloud $\mathcal{Q}_{(i,j)}$, the second part of Eq. 3 will be relatively large compared to the first part. The bigger d_j, i.e. shift of the symmetry plane as visualized in Fig. 2, $\mathbf{E}_{q \in \mathcal{O}} [\delta_2(q,p)]$ will increase and $\mathbf{E}_{q \notin \mathcal{O}} [\delta_1(q,p)]$ decrease. We are searching for the global minimum in the space of all votes

$$\hat{s}_{(i,j)} = \arg\min_{\mathcal{S}} v_{(i,j)} \quad (4)$$

that corresponds to a reflected point cloud $\mathcal{Q}_{(i,j)}$ with the smallest amount of points that contradict the symmetry hypothesis.

B. Surface Approximation

After the prediction of the backside of an object point cloud, we create a surface mesh approximation to support grasp planning and collision detection.

We use Poisson reconstruction proposed by Kazhdan et. al [12] as a solution to the problem of surface reconstruction from oriented points. To determine the normals, we use a kd-tree as proposed in [13]. A local plane fit is estimated for the k-nearest neighbors of the target point. This plane is assumed to be a local approximation of the surface at the current point. More advanced normal estimation techniques have been proposed which could perhaps increase the performance of the surface reconstruction at the cost of computation speed [14], [15]. Following normal estimation, we ensure that the normals of the mirrored points are consistent with the mirrored viewing direction reflected across the plane of symmetry.

Finally the Poisson reconstruction is performed. The single steps are briefly outlined below. For details, we refer to [12].

1) Inputting the points and normals to an octree.
2) Computing an implicit function over this adaptive grid.
3) Finally using marching cubes to extract an iso-surface as a watertight triangular mesh.

It should be noted while Poisson surface reconstruction is robust to some noise, it is sensitive to normal direction. We therefore filter outliers from the segmented point cloud prior to the reconstruction step.

III. Experimental Platforms and Grasp Cycle

To emphasize the generality of the proposed approach, we evaluate and demonstrate it on two robotic platforms being either used at the Royal Institute of Technology (KTH) or at the Universitat Jaume I (UJI). They differ in both, hardware and implementation of a grasp cycle.

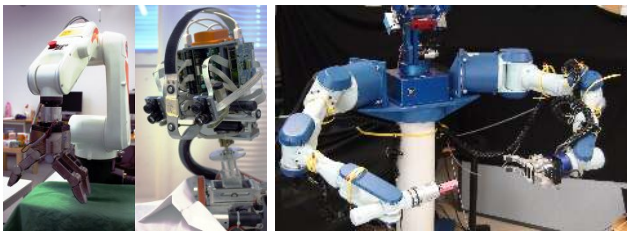

Fig. 3. Left: KTH robot. Right: UJI robot (Tombatossals)

A. Hardware

1) KTH: The platform (see Fig. 3 left) consists of an Armar III robotic head [16] equipped with two stereo cameras, a peripheral (wide-field) and a foveal (narrow-field) one. The robotic head has 7 DoF. Five of these are used for controlling the viewing direction while the remaining two mainly vary the vergence angle between the left and right camera systems, thereby enabling fixation on objects. As a manipulator, we use a 6 DoF Kuka arm[1] that is equipped with a three-fingered Schunk Dexterous Hand 2.0 (SDH)[2].

2) UJI: The torso system, called *Tombatossals* has 23 DOF (see Fig. 3 right). It is composed of two 7 DOF Mitsubishi PA10 arms. The left arm has a 4 DOF Barrett Hand[3] and the right arm has a parallel jaw gripper. Each arm has a JR3 Force-Torque sensor attached on the wrist between the arm and the hand. The visual system is composed of a TO40 4 DOF pan-tilt-verge head with two Imaging Source DFK 31BF03-Z2 cameras. Attached to the center of the pan-tilt there is a Videre DCSG-STOC stereo camera. For this work only the left arm, the pan-tilt head and the Videre stereo system are used.

B. Grasp Cycle

For the scenario of grasping unknown objects from a table top, a grasp cycle is outlined in Algorithm 1. The functions serve as place holders for operations that are common to both robotic platforms but are implemented differently. Exceptions to this are the function `PredictObjectShape`, the simplification of the triangular mesh through `ConvexDecomposition` and the grasp and arm trajectory planners named `PlanGrasp` and `PlanArmTrajectory`. They are identical in both systems and are explained in Section II and Section IV, respectively. The remaining functions of Algorithm 1 are not the focus of this paper. Therefore, in the below we will only briefly outline how they are implemented in each robotic system and refer to our previous work.

1) KTH: As a step prior to the shape analysis of an object and the grasping of it, the robot needs to segregate potential objects from the background. In the KTH system, the function `GetObjectHypotheses` to obtain a segmented point cloud is implemented as explained in detail in our previous work [17]. Given this point cloud which only represents the visible part of an object, its backside can be predicted as described in Section II.

[1] `http://www.kuka-robotics.com`
[2] `http://www.schunk.com`
[3] `http://www.barrett.com`

Algorithm 1: Pseudo Code for a Table Top Scenario

```
Data: Embodiment, Table Plane
Result: Cleaned Table Top
begin
    S = GetObjectHypotheses()
    for i = 0; i < |S|; i++ do Grasp Cycle per Object Hypothesis
        ŝ_i = PredictObjectShape(s_i)
        ŝ_i = ConvexDecomposition(s_i)
        G = GetGraspCandidates(ŝ_i)
        for j = 0; j < |G|; j++ do
            success = PlanGrasp(g_j)
            if success then Grasp Candidate gives Stable Grasp
                t_(i,j) = PlanArmTrajectory(g_j)
                if t_(i,j) ≠ 0 then Collision-Free Trajectory Exists
                    T = InsertTrajectory(T, t_(i,j), g_j)
                end
            end
        end
        k = 0
        repeat
            success = ExecuteGrasp(T, k)
            k++
        until success
    end
end
```

With the complete object shape as an input, `GetGraspCandidates` implements the technique presented in [18] to detect two grasping points. These points are the target positions for the fingers of the SDH. In detail, we apply a pinch grasp in which the thumb is opposite the two fingers. Then for applying a grasp to an object, the vector between the thumb and the two fingers has to be aligned with the vector between the two grasping points.

These grasp candidates are then simulated on the predicted object shape and a collision-free path for the arm planned. Details on this will be given in Section IV. After a suitable grasp and arm trajectory has been selected through simulation, it is executed by the robot in an open loop procedure.

2) UJI: The vision system on the UJI platform implementing `GetObjectHypotheses` is quite simple. The Videre stereo system gathers images and produces an unsegmented 3D point cloud of the scene. The table and background are black to simplify the segmentation. The point cloud is segmented finding its connected components using a clustering method as implemented in ROS PCL [4].

The complete object shape is predicted as described in Section II. `GetGraspCandidates` is implemented such that the vector between the thumb and two fingers is going through the object's centroid (see Section IV for more detail).

Approaching the pre-grasp position is also executed in open loop following the collision free trajectory obtained from the simulator. For the grasping action, a reactive sensor based strategy is used. This algorithm is fully described in [7], basically it tries to align the hand with the object and performs a power grasp adapting the hand pose to the object pose using tactile and force feedback.

IV. OpenGRASP

The simulation platform chosen to perform the experiments is OpenGRASP [19], a simulation toolkit for grasping and dexterous manipulation. It is based on the OpenRAVE [20], an open architecture targeting a simple integration of simulation, visualization, planning, scripting and control of robot systems. It enables the user to easily extend its functionality developing their own custom plugins.

The simulator is used to perform the grasp before the real robot makes an attempt. It allows for testing different alternatives, choosing the one with the highest probability of success. This will not only take considerably less time than performing it with the real hardware but also prevents damaging the robot by avoiding collisions with the environment.

OpenRAVE implements a combined motion and grasp planner plugin [2]. This BiSpace planner has elements from the bidirectional RRT and the RRT-JT algorithms [3]. The use of RRTs [1] is a well known approach to the arm motion planning problem. It has been the starting point for most of the current state of the art motion planners. For collision detection that is necessary for motion planning, the simulator needs complete object models. When known objects are used, an accurate model of the objects can be created off-line using different technologies, like laser scans. In the case of unknown objects, an approximate model has to be created on-line. In Section II, an approach to create this model was presented and it is evaluated in Section V. The experimental results show that even if the object model is not an exact representation of reality, it is close enough to enable the simulator to try different grasp alternatives and select an appropriate one.

The obtained triangular mesh has thousands of vertices which makes the collision detection process computationally expensive. To ameliorate this problem, we pre-process the mesh by `ConvexDecomposition`, a library that was originally created by John Ratcliff [21] and that is implemented in OpenRAVE. It approximates a triangular mesh as a collection of convex components. This process takes only a few seconds and drastically speeds up the grasp and motion planning.

A. Generation of Grasp Candidates

The first step for selecting an appropriate grasp consists of creating a set of grasp candidates and evaluating them using OpenRAVE. This set can be stored and used later, anytime the same robot has to grasp the same object. Each grasp candidate is simulated moving the end-effector until it collides with the object; then the fingers will close around it and finally the contacts are used to test on force closure. In Algorithm 1, this process is referred to as `PlanGrasp`.

Each grasp candidate has the following parameters: the approach vector, the hand pre-shape, the approach distance and the end-effector roll. The number of different values that these parameters can take has to be chosen considering the time-constraints imposed by the on-line execution of `PlanGrasp`.

OpenRAVE has a default algorithm to generate a set of approach vectors. It first creates a bounding box of the object and samples its surface uniformly. For each sample, a ray is intersected with the object. At each intersection, an approach

[4] http://www.ros.org/wiki/pcl

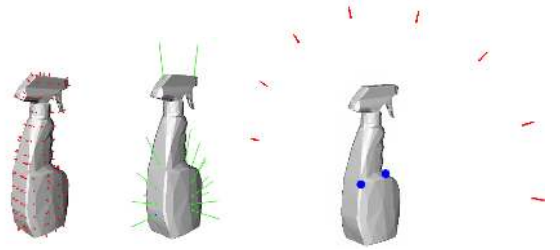

Fig. 4. Example of the approach vectors generated for a spray bottle by Left) the OpenRAVE grasper plugin, Middle) the UJI proposed algorithm using the object's centroid and Right) the KTH proposed algorithm using the grasp points in blue.

vector is created that is aligned with the normal of the object's surface at this point. An example output is shown in Fig. 4 Left). Dependent on the choice of the other parameters, the time to simulate all the corresponding grasps can vary from few minutes to more than an hour. These execution times are acceptable for objects that are known beforehand because the set of grasp candidates can be generated off-line. When the objects are unknown, this process has to be executed on-line and long waiting times are not desirable.

For this reason, we use two methods to reduce the number of approach vectors. The first one, applied on the KTH platform, computes two grasp points as in [18]. To grasp the object at these points, there are an infinite number of approach vectors on a circle with the vector between the two grasping points as its normal. We sample a given number, typically between 5 and 10 of these between 0° and 180° degrees. Figure 4 (Right) shows the detected grasping points along with the generated approach vectors. The second method, used at UJI, calculates the approach vectors in a similar way only that the center of the circle is aligned with the object's centroid and its major eigenvector e_a. Another circle, perpendicular to the first one, is added in order to compensate the possible loss of vector quality due to the lack of grasp points. Figure 4 (Middle) shows an example.

Having the list of approach vectors reduced, the other parameters were also adjusted for our purposes. As a hand pre-shape, we defined a pinch grasp for each hand. The approach distance is varied between 0 to 20 cm. Finally, the roll is chosen dependent on the two grasping points (such that the fingers are aligned with them) or on the orientation of the circle on which the selected approach vector is defined. Using these parameters, we were able to reduce the amount of time taken to generate and save the set of grasp candidates to less than a minute.

B. Grasp Execution Using Motion Planners

The next step `PlanArmTrajectory` in Algorithm 1, consists of selecting a stable grasp from the set of grasp candidates that can be executed with the current robot configuration without colliding with obstacles. For each stable grasp, it first moves the robot to the appropriate grasp pre-shape, then uses RRT and Jacobian-based gradient descent methods to move the hand close to the target object, closes the fingers, grabs the object, moves it to the destination while avoiding obstacles and releases it.

If the robot successfully grabs the object and moves it to the destination, the stable grasp is returned for execution with the real robot. Otherwise, the next stable grasp from the set is tried. Fig. 5 shows snapshots of the grasp execution using the simulator and the real robots.

V. Experiments

In this section, we will present quantitative experiments showing that the completion of incomplete object point clouds based on symmetry produces valid object models. Furthermore, we will show how the estimated complete object model helps when using a simple grasp strategy based on the center of an object.

A. Evaluation of the Mesh Reconstruction

In this section, we will evaluate how much the reconstructed mesh differs from the ground truth mesh.

1) Dataset: The database we used for evaluating the point cloud mirroring method is shown in Figure 6. For each of the objects in this database, with the exception of the toy tiger and rubber duck, we have laser scan ground truth[5]. The test data was captured with the KTH vision system and contains 12 different household or toy objects. Four of them are used both, when standing upright or lying on their side. Thereby, the database contains 16 different data sets. Each set contains 8 stereo images showing the object in one of the following orientations: 0°, 45°, 90°, 135°, 180°, 225°, 270° or 335°. An example for the toy tiger is shown in Figure 7. As an orientation reference we used the longest object dimension when projected down to the table. 0° then means that this reference axis is parallel to the image plane of the stereo camera. This can be observed in Figure 6 in which all objects are shown in their 0° pose. Therefore, the database contains 128 stereo images along with their point clouds.

From all point clouds, we reconstructed the complete meshes based on the method described in Section II. We used the two different values of 5 and 7 as the octree depth parameter of the Poisson surface reconstruction. By limiting this parameter, we enable mesh reconstruction in near real-time. With a tree depth of 5, the meshes are more coarse and blob-like but less sensitive to noise in the point cloud and normal estimation. With a depth of 7, the reconstructed surface is closer to the original point set. However, outliers strongly affect the mesh shape and it is more sensitive to noise.

To obtain the ground truth pose for each item in the database, we applied the technique proposed in [22]. It allows to register the laser scan object meshes to the incomplete point clouds.

2) Baseline: As a baseline, we reconstructed a mesh without mirroring. To do this, we applied a Delaunay triangulation[6] to the projection of a uniformly sampled subset of 500 points from the original point cloud. Spurious edges are filtered based on their length in 2D and 3D. Furthermore,

[5]The ground truth object models were obtained from `http://i61p109.ira.uka.de/ObjectModelsWebUI/`

[6]http://opencv.willowgarage.com

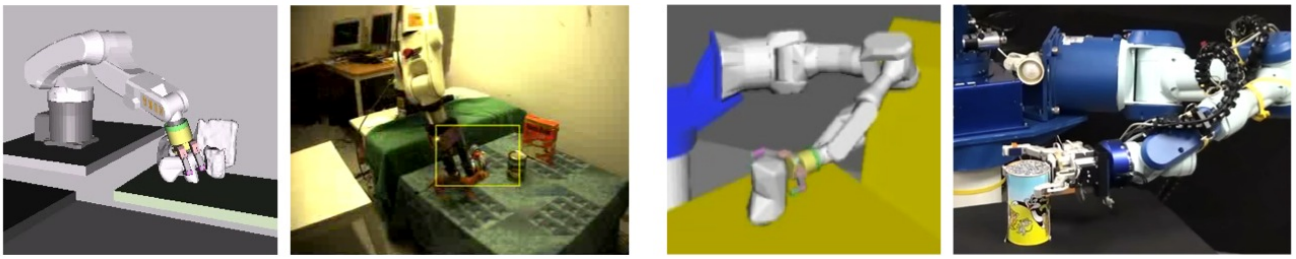

Fig. 5. Example of the grasp performed by the simulated robot and the real one, using Right) KTH platform and Left) UJI platform.

Fig. 7. One of the Datasets from Figure 6 shown in Orientation 0°, 45°, 90°, 135°, 180°, 225°, 270° and 335°.

Fig. 6. The 12 Objects in the Database in varying poses yielding 16 Data Sets. Objects are shown in their 0° position, i.e, with their longest dimension parallel to the image plane. Ground truth meshes are existing for all objects except the toy tiger and the rubber duck.

we extract the outer contour edges of this triangulation and span triangles between them which produces a watertight mesh. Figure 8 shows the result of this Delaunay based mesh reconstruction for the toy tiger.

3) Mesh Deviation Metric: To assess the deviation of the Delaunay meshes and the mirrored meshes from the ground truth we use MeshDev [23]. As a metric we evaluated geometric deviation, i.e., the distance between each point on the reference mesh to the nearest neighbor on the other mesh. We applied the uniform sampling of the surface of the reference mesh as proposed in [23] to calculate this deviation.

4) Results: Figure 10 shows the mean and variance of the mesh deviation between the ground truth mesh and the reconstructed meshes for all object orientations over all objects. We can state that the mirrored point clouds are on average always deviating less from the ground truth than

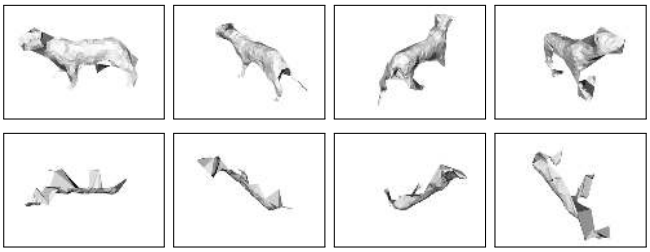

Fig. 8. Delaunay based Meshes of Toy Tiger in the following Orientations: 0°, 45°, 135°, 225°. 1-4: Front View. 5-8: Top View.

the Delaunay based meshes. The average deviation for the mirrored meshes over all orientations amounts to $7mm$.

Figure 11 shows the same error measure for each object independently averaged over all its orientations. The deviation measure is not normalized to the overall object size. Therefore, for bigger objects, like the *Brandt* box or the *Burti* and *Spray* bottle, the mean geometric deviation between the Delaunay meshes and the ground truth exceed 20mm. The mirroring yields a significant improvement for most objects. The green and white cup pose a challenging problem to the Poisson surface reconstruction because they are hollow. Modelling the void is difficult due to viewpoint limitations. On the other hand, holes in the point clouds due to non-uniform texture are usually closed by the surface reconstruction.

Since, we do not have the ground truth models available for the toy tiger and the rubber duck, we show their reconstructed meshes with tree depth 7 in Figure 9. The overall shape of the quite irregular toy objects is well reconstructed. However, because of the complexity of the objects if an incorrect mirroring plane is chosen, we obtain toy animals with either two heads or two tails. In such cases, there are strong violations of the visibility constraints. Thresholding of the vote in Equation 3 could address this. This is considered as future work.

B. Deviation of the Object Centroid

A very simple but effective grasping strategy of unknown objects is to approach the object at its center. However, estimating the centroid is not a trivial problem when the

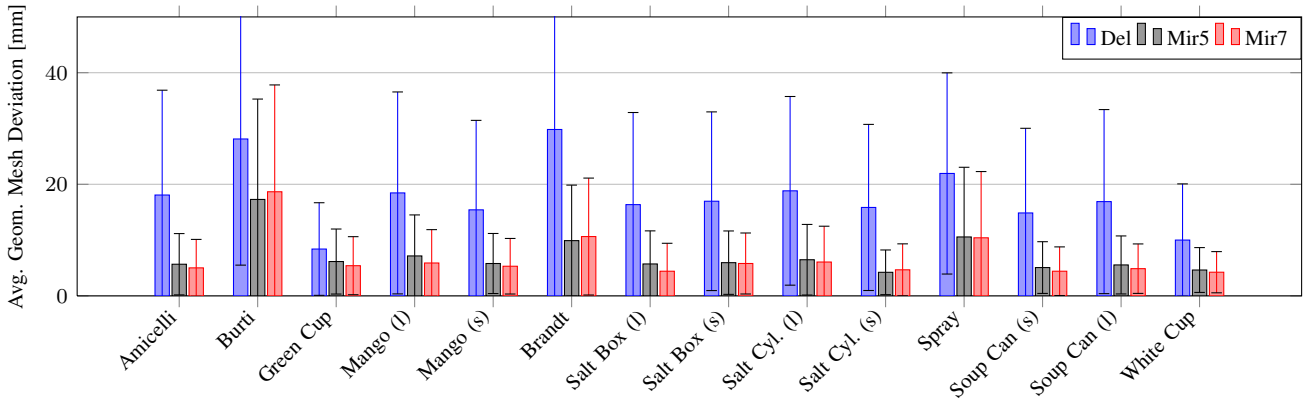

Fig. 11. Evaluation of the Deviation between the Ground Truth Mesh and i) Mesh based on 3D Point Cloud only (*Del*), ii) Mesh based on mirroring and Poisson Surface Reconstruction with Tree Depth 5 (*Mir5*) or iii) with Tree Depth 7 (*Mir7*). Pose of mirroring plane chosen over a set of different positions and orientations. Mean and Variance are computed for each object over all eight orientations.

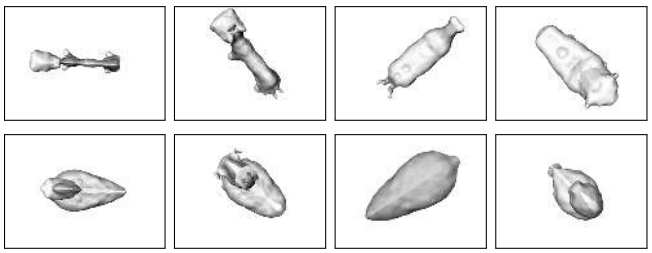

Fig. 9. Meshes based on Mirroring. First Row: Toy Tiger. Second Row: Rubber Duck.

object is unknown. Reactive grasping strategies are proposed to cope with the uncertainty in the object information during the grasp [8], [7]. The method proposed in [7] is applied for the grasp execution on the UJI platform. The more accurate the initial estimate of the object centroid, the fewer unnecessary contacts with the object occur in a reactive grasping scheme.

In this section, we therefore evaluated the accuracy of object center estimation as a simple placeholder for grasp quality. We compared the center of mass of the Delaunay mesh and of the mirrored mesh with the ground truth center. To render this comparison independent of the distribution of vertices (especially for the ground truth meshes), we applied the same uniform sampling of the mesh surface as in the previous section [23]. The center of mass is then the average over all the samples.

Figure 12 shows the error between the estimated and real centroid of an object per viewing direction and averaged over all objects. The deviation is normalized with the length of the diagonal of the oriented object bounding box. We can observe that the deviation ranges from approximately 5% to 10% of the total object size.

C. Real-World Experiments

We demonstrated the approach proposed in this paper on the two robotic platforms described in Section III. A video of the experiment can be found at `http://opengrasp.sourceforge.net/Videos/BohgICRA11.mp4`. Fig. 5 shows snapshots of the grasp execution in simulation on the predicted objects and with the real robots on the

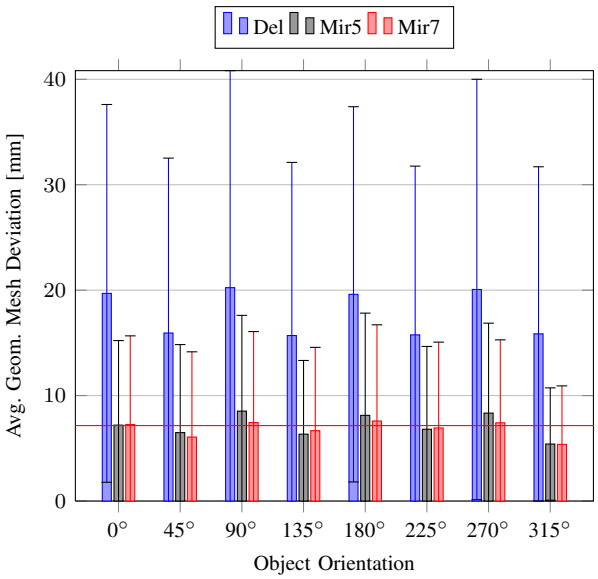

Fig. 10. Evaluation of the Deviation between the Ground Truth Mesh and i) Mesh based on 3D Point Cloud only (*Del*), ii) Mesh based on mirroring and Poisson Surface Reconstruction with Tree Depth 5 (*Mir5*) or iii) with Tree Depth 7 (*Mir7*).

real objects. At KTH, several objects were placed on the table emphasizing the benefit of object shape prediction for motion planning. Furthermore, it shows that the prediction mechanism can deal with some occlusions. This is due to the enforced visibility constraints. One of the main differences between the runs at UJI and KTH is the resolution of the point clouds that is due to the use of camera systems with different focal lengths. While the KTH point clouds usually consist of 40000 points, UJI point clouds contained around 3000 points. However, a suitable mesh could still be generated with the advantage of a lower runtime. When running the whole generate and test procedure (with $n = 6$ and $m = 5$ yielding 35 hypotheses) on a single core of an I7 CPU with 2.8 GHz, we achieved the following run-times: 16.46 seconds for a point cloud with 39416 points and 0.31 seconds for a point cloud with 2100 points. Please note, that we have not exploited the possibility to parallelize this

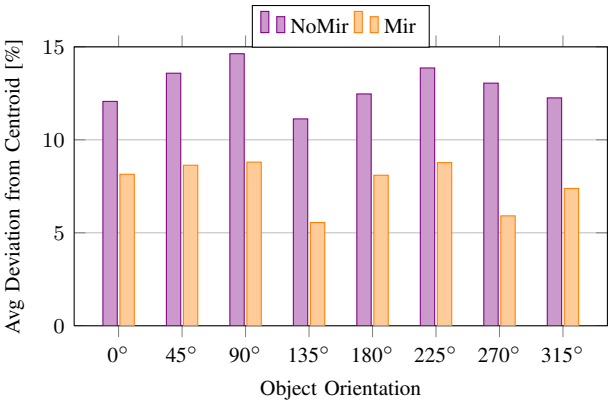

Fig. 12. Deviation of Estimated Object Centroid from Ground Truth Centroid.

process yet. These runtimes also show that downsampling the point clouds before mirroring can speed up the shape completion without a big loss of precision. To investigate these optimizations is considered as future work.

VI. Conclusions

In this paper, a method that estimates complete object models from partial views is proposed. We have validated these complete models using laser scan ground truth. The results show effectiveness of the technique on a variety of household objects in table top environments. Furthermore, the proposed technique has been demonstrated on two different robotic platforms, validating the feasibility of the predicted mesh for grasp and motion planning.

We feel that the proposed technique is a first step towards bridging the gap between simulation and the real world. In future, we hope to develop planners that take uncertain shape information explicitly into account to generate better grasp hypotheses and motion plans.

References

[1] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA'2000: Proceedings of the 2000 IEEE/RSJ international conference on Robotics and Automation*. Piscataway, NJ, USA: IEEE Press, 2000.

[2] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, "Bispace planning: Concurrent multi-space exploration," in *Robotics: Science and Systems*, June 2008.

[3] M. Vande Weghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, nov. 2007, pp. 477 –482.

[4] M. Ciorcarlie, C. Goldfeder, and P. Allen, "Dexterous Grasping via Eigengrasps: A Low-Dimensional Approach to a High-Complexity Problem," *Robotics: Science and Systems Manipulation Workshop*, 2007.

[5] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp Planning Via Decomposition Trees," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 4679–4684.

[6] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic Grasp Planning Using Shape Primitives," in *IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 1824–1829.

[7] J. Felip and A. Morales, "Robust sensor-based grasp primitive for a three-finger robot hand," in *IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1811–1816.

[8] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," in *IROS'10: Proceedings of the 2010 IEEE/RSJ international conference on Intelligent robots and systems*, October 2010.

[9] J. Bohg, M. Johnson-Roberson, M. Björkmann, and D. Kragic, "Strategies for multi-modal scene exploration," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei,Taiwan, October 2010.

[10] S. Thrun and B. Wegbreit, "Shape from symmetry," *Computer Vision, IEEE International Conference on*, vol. 2, pp. 1824–1831, 2005.

[11] T. P. Breckon and R. B. Fisher, "Amodal volume completion: 3d visual completion," *Comput. Vis. Image Underst.*, vol. 99, no. 3, pp. 499–526, 2005.

[12] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 61–70.

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *COMPUTER GRAPHICS-NEW YORK-ASSOCIATION FOR COMPUTING MACHINERY-*, vol. 26, pp. 71–71, 1992.

[14] D. Cole, A. Harrison, and P. Newman, "Using naturally salient regions for SLAM with 3D laser data," in *International Conference on Robotics and Automation, SLAM Workshop*. Citeseer, 2005.

[15] N. Mitra and A. Nguyen, "Estimating surface normals in noisy point cloud data," in *Proceedings of the nineteenth annual symposium on Computational geometry*. ACM, 2003, p. 328.

[16] T. Asfour, K. Welke, P. Azad, A. Ude, and R. Dillmann, "The Karlsruhe Humanoid Head," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Daejeon, Korea, December 2008.

[17] X. Gratal, J. Bohg, M. Björkman, and D. Kragic, "Scene representation and object grasping using active vision," in *IROS 2010 Workshop: Defining and Solving Realistic Perception Problems in Personal Robotics*, Taipei, Taiwan, October 2010.

[18] M. Richtsfeld and M. Vincze, "Grasping of Unknown Objects from a Table Top," in *ECCV Workshop on 'Vision in Action: Efficient strategies for cognitive agents in complex environments'*, Marseille, France, September 2008.

[19] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, and R. Dillmann, "OpenGRASP: A toolkit for robot grasping simulation," in *SIMPAR '10: Proceedings of the 2st International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, 2010.

[20] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, July 2008.

[21] J. Ratcliff. (2009) Convex decomposition library. [Online]. Available: http://codesuppository.blogspot.com/2009/11/convex-decomposition-library-now.html

[22] C. Papazov and D. Burschka, "Stochastic optimization for rigid point set registration," in *ISVC '09: Proceedings of the 5th International Symposium on Advances in Visual Computing*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 1043–1054.

[23] M. Roy, S. Foufou, and F. Truchetet, "Mesh Comparison Using Attribute Deviation Metric," *International Journal of Image and Graphics*, vol. 4, 2004.