Heuristic 3D Object Shape Completion based on Symmetry and Scene Context

David Schiebener, Andreas Schmidt, Nikolaus Vahrenkamp and Tamim Asfour

***Abstract*— Object shape information is essential for robot manipulation tasks, in particular for grasp planning and collision-free motion planning. But in general a complete object model is not available, in particular when dealing with unknown objects. We propose a method for completing shapes that are only partially known, which is a common situation when a robot perceives a new object only from one direction.**

Our approach is based on the assumption that most objects used in service robotic setups have symmetries. We determine and rate symmetry plane candidates to estimate the hidden parts of the object. By finding possible supporting planes based on its immediate neighborhood, the search space for symmetry planes is restricted, and the bottom part of the object is added. Gaps along the sides in the direction of the view axis are closed by linear interpolation. We evaluate our approach with real-world experiments using the YCB object and model set [1].

I. INTRODUCTION AND RELATED WORK

Robots become useful when they interact with their environment in a goal-oriented manner, i.e. when they execute practical tasks. Being able to physically manipulate the world distinguishes them from passive artificial systems and enables them to perform production or service tasks. In order to grasp or manipulate objects, shape information is crucial for a robot to plan its actions. The required completeness and precision of the shape information varies, but better knowledge usually leads to better results. Grasp planning algorithms traditionally assume and require that a complete and exact object model is available [2], although some approaches for finding grasp locations on unknown objects have been proposed (e.g. [3], [4], [5]).

When a robot encounters a new object, which will probably happen quite frequently in a realistic open environment, no shape information is initially available. Assuming that the object can be segmented from its environment, the robot can perceive the shape of the side that is facing it using stereo or depth cameras. Depending on the agility of the robot and the availability of sufficient free space around the object, it can possibly be observed from more than one direction. However, making all sides of the object visible would require complex manipulation, which in turn requires comprehensive knowledge about the shape (and preferably other physical properties like mass distribution). A shortcut solution that circumvents this problem is to generate a reasonable hypothesis about the shape of the object based on the available information, and use it to plan the intended manipulation. This is the motivation behind our approach.

Unavoidably, assumptions have to be made when estimating the shape of the hidden object parts. One possibility is to use basic geometric primitives to approximate the object. In [6], the object is classified as being either box- or cylinder-like, and the parameters of the respective primitive are estimated using the perceived points. Using visibility criteria for the generated new parts, the probability of the estimation being realistic can be quantified. In [7], [8] and [9], a superquadric is fitted to the object. Using Levenberg-Marquard optimization, the parameters of the superquadric are estimated from the object points obtained from a depth camera.

Using a database of known shapes, the authors in [10] approximate the object by nonrigid alignment and deformation of known shapes with the perceived points. In [11], a set of shape primitives is fitted into the incomplete observation to fill holes.

Most artificial and natural objects are, at least to some degree, symmetrical. Based on that observation, a popular assumption when aiming to complete partial object shapes is that the object exhibits some kind of symmetry. Early work on completing perceived object shapes exploiting apparent symmetries goes (at least) back as far as 1988 [12]. The authors in [13] propose a hierarchy of symmetry types that they detect efficiently, starting with simple ones and possibly combining them to more complex ones. In [14], the authors model the object as an extrusion, which is a 2D profile that is moved along a trajectory to form a 3D shape. They restrict the possible extrusions to linear or circular paths, thus the object is assumed to be symmetric. Once the symmetry plane is found, the appropriate extrusion parameters can be estimated.

The authors of [15] assume that the object has a plane of symmetry, and try to find it. Presuming that the object stands on a table surface, they generate candidates for possible symmetry planes orthogonal to the table, mirror the perceived object points on them, and rate them using visibility and plausibility criteria. The best plane is retained, and the point cloud resulting from original and mirrored points is used for grasp planning. In [16], first the mirroring approach from [15] is applied and then the completed object shape is approximated by a superquadric. While superquadrics are a rather restrictive subset of symmetric objects, they have the advantage of defining a complete surface without any holes, which is a helpful property e.g. for grasp planning.

The research leading to these results has received funding from the European Unions Horizon 2020 Research and Innovation programme under grant agrement 643950 (SecondHands) and Seventh Framework Programme FP7 under grant agreement 270273 (Xperience).

The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. schiebener@kit.edu, vahrenkamp@kit.edu, asfour@kit.edu

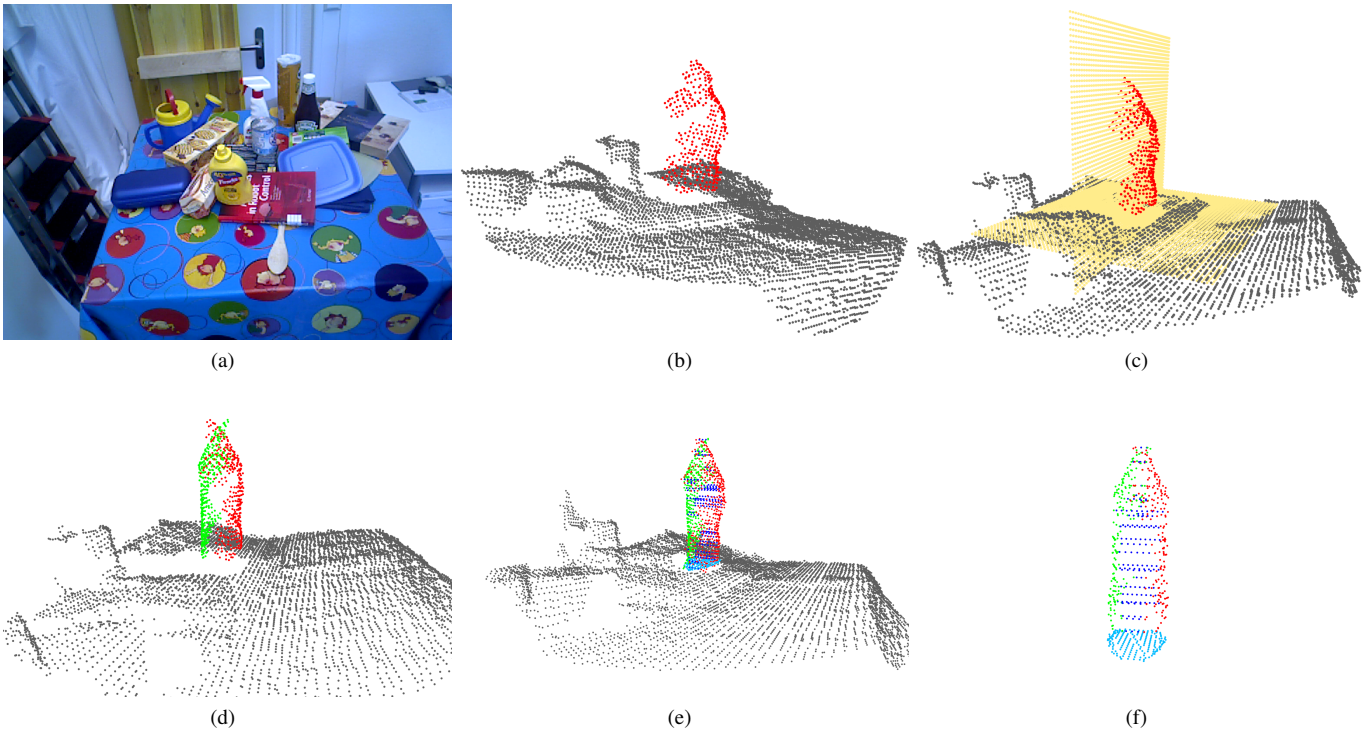


Fig. 1: (a) is the camera image of the scene, in which the yellow mustard bottle has been segmented. (b) shows the point cloud of the object (red) and the part of the environment that is near it (gray). In (c), the estimated supporting plane and the symmetry plane that are used for shape completion are shown (both in yellow). It can be seen clearly that only the front side of the object was perceived. (d) shows the new points (in green) that were generated by mirroring of the original points on the symmetry plane. In (e) and (f), the final completed model is shown after adding the sides (dark blue) and the bottom (light blue).

In this work, we generalize and extend the approach of of the authors in [15] in a way that allows us to generate complete surfaces without significant holes while keeping the full freedom and descriptiveness of symmetric shapes. Using not only the perceived part of the object itself but also the scene around it, we are able to gain more information for the shape completion and also relax some of the assumptions made in their original approach.

II. OVERVIEW

The starting point of our approach is a point cloud, i.e. a set of 3D points, of an unknown object that the robot has perceived. In the worst case the object has only been observed from one perspective, which means that only its front side towards the camera is available. Our second input is the point cloud of the scene around the object (Fig. 1a, 1b).

Thus, we assume that the segmentation problem, i.e. separating the object from its environment, has already been solved. This is in general an extremely hard problem which we solve using in our previous work ([17], [18]. Here, segmentation hypotheses are verified and corrected by the robot by pushing objects slightly and using coherent motion as an additional cue to obtain reliable, grounded segmentations.

Using the partial object and the scene point cloud, we generate candidates for possible supporting surfaces on which the object might be standing. We assume that the object has a planar symmetry and, in order to restrict the search space, that the symmetry plane is perpendicular to the supporting plane[1]. Consequently, for each potential supporting surface, we generate corresponding symmetry plane hypotheses (Fig. 1c).

The plausibility of these hypotheses is rated based on visibility criteria considering the perceived object point cloud. Once the best one has been determined, it is used to mirror the perceived points and thus obtain a back side for the object, i.e. the side of the object that is turned away from the camera (Fig. 1d). Additionally, we close holes on the sides of the object and add a bottom plane using the supporting surface corresponding to the best found symmetry plane. As a result, we get a point cloud that constitutes a complete 3D model of the object of interest (Fig. 1e, 1f).

In the following sections, we will explain all steps of that process in detail, and give the information necessary for re-implementation. We have used the point cloud library PCL

[1]It may happen that the object has a symmetry plane parallel to the surface it is standing on. Mirroring on such a plane would not be very helpful for estimating the back side of the object, which is why we enforce that roughly vertical symmetry planes are determined.

[19], the very lightweight vision library IVT [20] and the linear algebra library Eigen [21].

III. SYMMETRY PLANE HYPOTHESES

Assuming that the object has a planar symmetry, the main challenge is to find it based on the available partial shape information. We use information about the scene surrounding the object to restrict the search space that needs to be sampled.

A. Supporting plane hypotheses

Unless it is held by someone, the object must be standing on some kind of supporting structure. We assume that the robot knows the direction of gravity, which can easily be achieved using an inertial measurement unit, and that the structure supporting the object must be roughly perpendicular to this vertical direction.

We search the immediate neighborhood of the object for surface candidates on which the object is standing. We use the scene point cloud, from which the object has been removed, and consider only those points that lie near the object. With the object radius r_{obj} being defined as half of the maximal diameter of the object point cloud, we remove all points that are further than $3 \cdot r_{obj}$ away from the object center. So with S being the complete scene point cloud, O the object point cloud and C_{obj} the center of the object[2], the neighborhood point cloud N is defined as

$$N = \{p_i \in S \setminus O \mid |p_i - C_{obj}| < 3 \cdot r_{obj}\}.$$

In this neighborhood point cloud, we determine the dominant planes using random sample consensus (RANSAC, [22]). A plane is defined by three non-collinear points, and we repeatedly draw random samples of three points from the scene point cloud. For each of these samples we determine the corresponding plane parameters and count the number of scene points that lie within a tolerance margin of that plane[3]. We keep the plane with the maximal number of inliers, remove them from the scene point cloud, and repeat the process until we find no more planes with a minimal number of inliers[4].

From these planes, we select those that are reasonable candidates as supporting surface of the object. There are two criteria that have to be met by each surface S_i. Firstly, the surface should be roughly horizontal, i.e. its normal n_i should not differ too much from the direction of gravity g. Consequently, we prefer planes for which the angle

$$\alpha = \arccos\left(\frac{|n_i \cdot g|}{|n_i| \cdot |g|}\right)$$

between n_i and g is small. Secondly, as we assume that the object is standing on it, there should not be too many object points lying below the plane. For each point p_j, we calculate its projection p_j^P onto the surface. The sign of the scalar product of $(p_j - p_j^P) \cdot n_i$ indicates whether the point is lying above or below the plane, so the set $O^{b_i} \subseteq O$ of object points lying below the plane is

$$O^{b_i} = \{p_j \in O \mid (p_j - p_j^P) \cdot n_i < 0\}.$$

The ratio of object points below the surface, and the angle between its normal and the direction of gravity, are then used to define an rating for the surface candidate, which is given as the weighted sum of those two cues:

$$r_i = \omega_1 \, \alpha + \omega_2 \frac{|O^{b_i}|}{|O|}.$$

The weights ω_1, ω_2 are chosen so that an angle $\alpha = 90°$ between n_i and g is equivalent to half of the points being below the supporting plane hypothesis. To limit the computational effort later, we only keep the 10 best supporting plane candidates.

B. Generation of symmetry plane candidates

Similar to [15], we make the assumption that the symmetry plane of the object is perpendicular to the supporting surface, with the important difference that we have several hypotheses for that surface. For each of them, we create symmetry plane candidates by uniform sampling. A plane can be defined by a support vector and two orthogonal vectors that span it. Initially, we choose the center C_{obj} of the object point cloud O as the support vector. With O^P being the projection of O onto the supporting surface, we choose the principal axis of O^P which is more orthogonal to the view direction as the first spanning vector, and the normal of the supporting plane as the second one.

Starting from this initial symmetry plane candidate, we generate more candidates by rotating and shifting the plane over regular intervals. We obtain the translation axis a_t by projecting the view direction onto the supporting plane. The translational sampling along this axis is restricted to an interval between the 0.1 and 0.9 quantile of the object point cloud along that axis. For each position, we sample different orientations by rotating the plane around the normal of the supporting surface, with a maximal rotation of $45°$ in both directions.

The required computational effort to analyze and rate all symmetry plane hypotheses is proportional to the product of the number of supporting plane hypotheses, translational samples per plane and rotational samples per translation. Thus, restricting the sampling space efficiently is the key to having a high chance of good results within a reasonable computation time. By testing different potential supporting surfaces, we allow for a higher flexibility compared to [15], for the price of higher effort - or a lower sample density when we keep the effort similar. In section V-B, we show how to to find a good compromise for the sampling density parameters.

[2]Although one might extensively discuss how to best define the center of an object, in this case only a rough approximation is needed and we simply use the mean of the object point cloud.

[3]The appropriate value for the threshold essentially depends on the precision of the depth perception. We set it to $3mm$.

[4]The value of this threshold should depend on the cardinality of the neighborhood. We set it to 3% of the number of points in the neighborhood.

C. Calculation of points mirrored on a plane

Given a symmetry plane candidate with support vector c and normal n, a point p_i is mirrored to the point p_i' by adding twice the vector from the point to its orthogonal projection onto the symmetry plane. With

$$p_i^P = p_i - \frac{(p_i - c) \cdot n}{n \cdot n} n$$

being the orthogonal projection of p_i, we get the mirrored point

$$p_i' = p_i + 2(p_i^P - p_i) = 2p_i^P - p_i = p_i - 2\frac{(p_i - c) \cdot n}{n \cdot n} n.$$

D. Rating of symmetry planes

In order to choose the best symmetry plane candidate, a rating of its apparent plausibility is necessary. It is calculated based on the locations of the points that are generated by mirroring the originally perceived object points on the symmetry plane. Depending on its position, a mirrored point falls in one of these four categories:

1) If it is very close to an original point, this indicates that the symmetry hypothesis is reasonable.
2) If it lies behind the original points, and thus would be invisible for the camera, it may be a correct estimation that introduces information about the back side of the object.
3) If, in contrast, the point lies besides or in front of the original object points, it would be visible to the camera and, if it belongs to the object, should be already included in the original point set. Therefore, it indicates that the symmetry plane hypotheses may be incorrect.
4) If a mirrored point lies underneath the estimated supporting surface, this also reduces the credibility of the candidate symmetry plane[5].

Fig. 2 shows the different possible positions of the mirrored points. To rate a symmetry hypothesis, each point p_i' is categorized to belong to one of them. The first check is for the point to be above or below the supporting plane. If it is below, its distance to the plane is added to the rating with a negative weight. For the remaining points, we first check if they are beside the original point cloud when seen from the camera. This is done efficiently using a binary segmentation mask created by projecting the object point cloud into an image and closing it using morphological operations [23]. If the point is projected outside that segmentation area, it is beside the object, otherwise it may be in front of, behind, or coinciding with the original points.

In any case, the nearest neighbor of of p_i' in the original point cloud can be determined as

$$n(p_i') = \underset{p_j \in O}{argmin} |p_i' - p_j|.$$

[5]As the symmetry planes are perpendicular to the supporting plane, these points are the same for all symmetry hypotheses based on the same supporting surface candidate. However, they indicate that the supporting plane candidate may be bad and are therefore helpful for preferring symmetry hypotheses originating from different potential supporting planes.

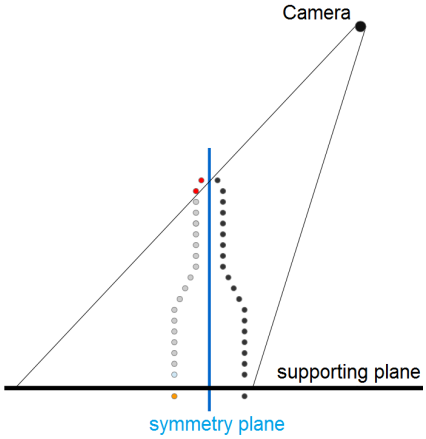


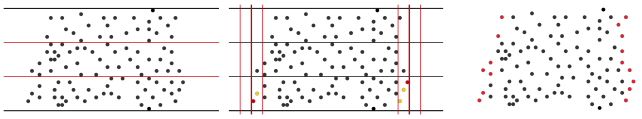
Fig. 2: Visualization of the different regions where the mirrored points may lie with relation to the original object points (black). If the new points lie amongst the original ones, they support the symmetry plane hypothesis. If they lie behind them (gray), they give additional speculative information about the object shape. If they are in front of or beside the original points (red), they would have been visible before, so they indicate that the symmetry plane might be incorrect. The same holds for points that lie under the estimated supporting surface (orange).

If p_i' is projected within the segmentation mask, the distances of p_i' and $n(p_i')$ from the camera are compared to decide whether it lies in front of, within, or behind the original point cloud. If it lies in front of or beside of it, the distance between p_i' and its nearest neighbor is used for the rating with a negative weight. If it lies within or behind the original points, a positive value is added to the rating. These two positive values can be used to balance the preference of symmetry hypotheses that either seem plausible for having many points that coincide with the original ones, or generate more speculative information about the back side of the object. We give the same weight to both categories.

The overall rating of a symmetry plane candidate is the sum of the positive and negative ratings for all mirrored points. As the two negative and the two positive categories get the same weight respectively, the ratio of positive and negative weight doesn't matter for the outcome. We normalize the rating by the number of points, which also makes no difference for choosing the best symmetry plane but allows for comparison between different objects (or different perceptions of the same objects). The symmetry plane candidate with the best rating and its corresponding supporting plane hypothesis are selected, and all original points are mirrored on it.

IV. ADDITIONAL SHAPE COMPLETION STEPS

When testing the shape completion based only on mirroring the object point cloud, we observed that there were often significant holes left in the result. Usually, an object perceived by the robot is seen well from the front and from the top. The back side, and sometimes the rear part of the top



(a) (b) (c)

Fig. 3: Visualization of how the points on the sides of the object are determined. In (a), the point cloud is subdivided into horizontal segments. Within each segment, we find the left- and rightmost points. They, and all points within the segment that are near the vertical lines through them, are considered to be edge points. This is shown in (b) for the bottom segment. In (c), all resulting edge points are marked in red.

when it isn't flat, are reconstructed by the mirrored points. But the sides, when they are roughly aligned with the view direction, tend to be missing. This is due to the fact that with stereo cameras as well as active depth sensors, little reliable depth data can be perceived from these surfaces. Additionally, when an object is segmented, the borders of the segmentation are often imperfect.

The bottom of the object is usually invisible to the camera, except when the object is bowl-shaped or open at the front. Little or no information can be added here by mirroring on a symmetry plane that is perpendicular to the supporting surface. As we want to estimate an object model that is as complete and hole-free as possible to allow e.g. grasp planning algorithms to work on it, we perform additional shape completion steps to overcome these two frequent deficits.

A. Sides of the object

To define what are the sides of the object, we consider its points in the camera coordinate system. Here we are looking for its roughly vertical edges, i.e. the minimal and maximal values along the horizontal image plane axis for any existing value on the vertical axis. As the object is given in the form of a discrete point cloud and not as a filled spatial region, we have to approximate its outline.

First, we project all points into the image plane, i.e. the two dimensions perpendicular to the view direction. One possibility would be to use the two-dimensional convex hull, but this would lead to great errors on concave shapes. Instead, we subdivide the point cloud along the vertical direction to get horizontal segments, in each of which we determine the left- and rightmost point. Here the size is crucial, as too large intervals would eliminate details, while too small ones would carve into the object when no point on the actual border lies within the segment.

We determine the mean distance $\bar{d}$ of a point in the original object point cloud O to its respective nearest neighbor:

$$\bar{d} = \frac{1}{|O|} \sum_{p_i \in O} \min_{p_j \in O / \{p_i\}} |p_i - p_j|.$$

The point cloud, projected into the camera plane, is subdivided vertically into equal intervals between its highest and lowest point, such that the height of the subdivision segments is greater or equal to $2 \cdot \bar{d}$. This gives a high certainty that actual edge points are contained in each segment, while still maintaining a reasonable degree of precision (see Fig. 3a).

In each segment, we determine the minimal and maximal point in horizontal direction. These points, and all points in the segment that are within a distance of $\bar{d}$ from the vertical line through the minimal or maximal point, are considered to be part of the edge (see Fig. 3b). Each of these edge points is connected with its mirrored counterpart on the back side of the object. Along the straight line connecting the original and the mirrored point, we add points in regular intervals of length $\bar{d}$.

B. Bottom of the object

The bottom part of an object standing on something else is usually not visible, so it is a very common case that it is missing in the perceived object point cloud. We rely mostly on the estimated supporting plane to add a bottom part in the last step of our shape completion approach. We estimate the area where the object touches the supporting surface, and close it there. To this end, we select all points of the model generated so far, i.e. original as well as mirrored and side points, that are very close to the supporting plane. The threshold is set to $3 \cdot \bar{d}$.

These points which seem to be close to the bottom of the object are projected onto the supporting plane. We calculate their convex hull and fill it with a regular grid of points at intervals of $\bar{d}$. It is important that only the low object points are used to estimate the bottom area, because otherwise we would create artificial planes under object parts that spread out in higher parts of the object, e.g. the handle of a cup.

There are still cases in which this approach would create a too large bottom part. This occurs when the part of the object that is in contact with the supporting surface is not convex, or even unconnected, e.g. when the object has legs like a table. To limit this effect, we remove all bottom points that would have been visible from the camera, similar to what we do in the rating of the symmetry hypotheses in section III-D.

V. EXPERIMENTAL EVALUATION

We evaluated our approach using the YCB object and model set [1]. Out of the overall set of 75 objects, 18 cannot be perceived well enough by the active RGBD camera that we use because they are transparent, reflective or too small. Another 5 objects are deformable, and 4 have the same shape but different sizes, so we performed our experiments using the remaining 49 unique and well perceivable objects to evaluate our approach.

Each object was segmented by our interactive object learning approach presented in [18], where the object is pushed by the robot and its motion is used as an additional cue to obtain reliable segmentations even in extremely cluttered scenes. This allows us to go beyond the usual lonely-object-on-empty-table setup and perceive the objects in more varied environments (such as shown in Fig. 1a). We used the

segmentation result after performing 3 pushes on the object. From the experimental results in [18], we expect to cover around 95% of the visible object parts, which subjectively seemed to hold for this new object set too.

Based on the point cloud obtained from the segmentation, we apply our algorithm for shape completion. The result is compared to the ground truth, which is available in form of the scanned models of the YCB object and model set.

A. Quality measures

To quantify the accordance of our completed object shape with the model, we first need to align them. Loading the model as a point cloud, we match it with the completed point cloud using the Iterative Closest Point (ICP) algorithm [24]. As this algorithm finds a local optimum, we start it with different initial relative positionings: The two point clouds are placed so that their centers are in the same location, and one of them is rotated so that it takes all six possible orientations that can be reached by rotations of $180°$ around the three axes. We keep the ICP result with the smallest mean error, thus avoiding to use an alignment that got stuck in a local minimum far off the correct matching of the two point clouds. However, the alignment may be imperfect, therefore our ratings for the completed point clouds may sometimes be a bit worse than they deserve.

One measure is the mean distance of each point in one point cloud to its nearest neighbor in the other point cloud. We calculate this score in both directions, i.e. we determine the nearest neighbor for each point of the completed shape in the ground truth model, and vice versa, to make sure the rating penalizes missing or added object parts. This gives a good measure of the distance between estimated and real shape, however it is also influenced by the pixel density and precision of the camera.

Therefore, we also calculate a measure that is intended to quantify the shape similarity in a way that gives a robust and dimensionless correlation value. To this end, we project the completed shape and the model into an image, thus getting a binary silhouette. We do this from all three directions along the coordinate axes. We calculate two measures for the silhouette similarity: The zero-mean normalized cross correlation (ZNCC), and the ratio of the size of the region that is occupied by both to the size of the region occupied by at least one of the two silhouettes, i.e. $\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Both measures give a value in $[0, 1]$ that is 1 for perfect correspondence and 0 for none. We give the average of these values for the three projections from different directions.

B. Quality depending on the sampling density

The main tradeoff to be made in our approach is between the required computation time and the quality of the resulting completed object shape. With $|O|$ being the number of points in the original object segmentation, the expected effort is in $\Theta(|O| \log |O|)$ for the nearest neighbor search, all other calculations are linear in $|O|$. The objects are segmented at a resolution of 320×240, and the original object point clouds contain between 100 and 900 points, on average 312. The resulting completed models contain on average 1251 points, which is about 4 times the initial size.

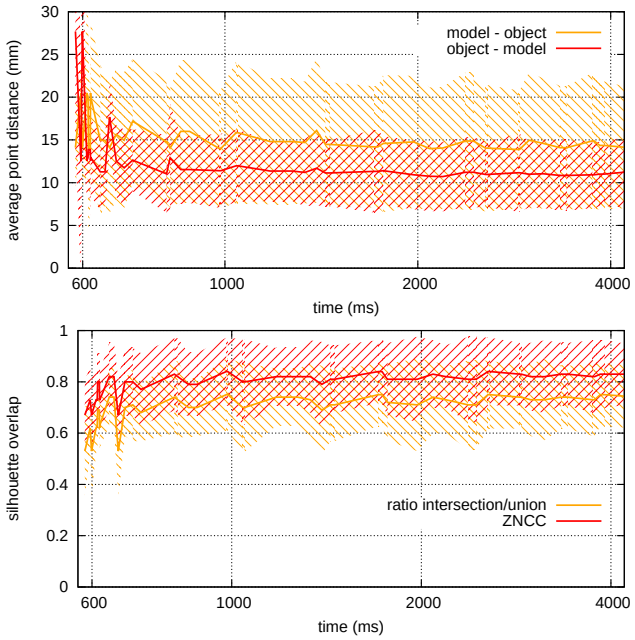


Fig. 4: The upper graph shows the mean distance of a point in the completed object shape to its nearest neighbor in the ground truth model, and vice versa. The lower graph shows two measures for similarity of the projected silhouettes of the completed shape and the ground truth. One measure is the zero-mean normalized cross correlation (ZNCC), and the other gives the ratio of the image region covered by both silhouettes to the image region that is covered by at least one of the two silhouettes. The solid lines show the average for the whole test set, and the striped regions show the standard deviation over the different objects.

As we don't want to downsample the object point cloud, the most effective factor with which we can influence the computational effort is the number of samples for the position and orientation of symmetry plane candidates. We leave the number of potential supporting planes fixed at 10, but vary the number of positions and orientations of the symmetry plane candidates that are tested for each of the supporting surfaces. Fig. 4 shows the dependency of the shape quality on the invested computation time. As can be seen, the quality reaches a certain saturation when more than one second is spent testing symmetry plane candidates. This corresponds to a sampling density of 20 different positions per supporting surface and 10 different orientations per position.

Table I shows the computation times for the different steps of the shape completion using this sampling density. The computationally intensive steps are parallelized, and we run our experiments on a PC with an Intel Core i7 CPU with 8 virtual (4 physical) cores at 3.6 Ghz. There is certainly potential for some more speedup, but we considered the required time to be acceptable for our intended future use as a prerequisite to grasp planning.

TABLE I: Computation times for the different steps of our algorithm (in ms). The values are averages over the whole test set, with the standard deviations given in brackets.

find supporting planes	generate and evaluate symmetry planes	add sides and bottom	total
150.3 (±45.5)	535.4 (±168.3)	144.4 (±82.9)	830.1 (±425.0)

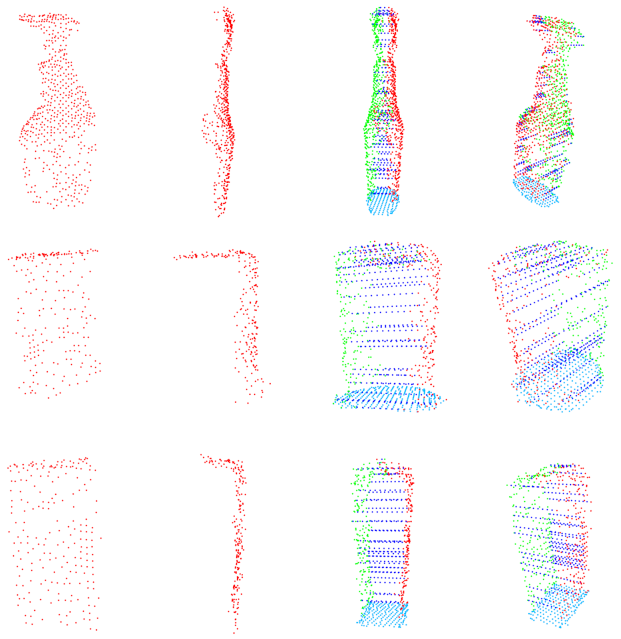


Fig. 5: Results of our shape completion approach for a spray bottle, a cylindrical and a box-shaped package from the test set. The first and second column show the original point cloud, the third and fourth show the result. The original points are red, those generated by mirroring on the symmetry plane are green, the sides are dark blue and the bottom points are light blue.

C. Discussion of the results

The results show that when sufficient computation time is invested, the shape completion is able to create estimations that robustly come close to the real object shapes. While it is hard to judge the absolute similarity between estimated and real shape from the average point distances, the silhouette-based measurements indicate that the concordance is high, but not perfect. The fact that the mean distance of a point in the ground truth model to its nearest neighbor in the completed shape is higher than that of a point in the completed shape to the model indicates that our estimations tend to lack parts of the real object.

We have identified two reasons for this: One is that already in the segmentation that we use, the visible side of the object is not covered completely. This error is propagated through the shape completion, and consequently the resulting shape estimation will be smaller than the actual object, missing the parts whose visible side was not included in the segmentation. Another case where our approach often slightly underestimates the volume is when dealing with rounded objects. Here, the depth camera tends to give no values at the edges of the object where its curvature causes the surface to be nearly parallel to the view axis. In those cases, our method connects front and mirrored back sides by straight lines, which is approximately correct, but omits the bulge on the side. A good example to see this is the cylindrical coffee package in the second row of Fig. 5.

Overall, the shape estimations resulting from our approach approximate the real objects reliably without huge errors. The approach is robust to the objects being placed in non-trivial environments and on tilted supporting surfaces, which is certainly a step forward compared to previous work that, to our best knowledge, only considered objects in isolation from their environment. However, we benefit to some degree from the fact that the objects in the test set were all at least roughly symmetrical. For a totally asymmetrical object, the shape completion error would probably be larger.

In our evaluation, we have only considered the most problematic case in which the object has only been seen from a single point of view. In practice, a robot will usually be able to move its cameras at least a bit and thus perceive more than only one side of the object, to which we informally refer as the ”front side” here. Any additional points on the sides and on the back side of the object will make the estimation of the symmetry plane significantly easier and more stable. When different points of view are taken into account for the visibility criteria in the rating of the symmetry plane hypotheses, that rating becomes much more meaningful, and the potential volume of misestimations of the back side would effectively be restricted to the remaining region that can not be seen from any of the points of view.

VI. CONCLUSIONS AND FUTURE WORK

We have presented an approach for estimating complete 3D shapes of objects that have only been observed partially. Focusing on the worst case, and probably the most common one, where only one side of the object was perceived, we propose robust heuristics that allow to construct a complete model from this limited information. The main assumption is that the object has a symmetry plane that can be used to estimate its back side.

We extend the approach of [15] in several respects: By considering several potential supporting surfaces, we allow for the objects to be placed in nontrivial scenes, instead of requiring them to be placed on an empty table. We also use the estimated supporting plane to close the object at the bottom, and additionally fill the frequently occurring holes at the sides of the object which arise when the sides are roughly aligned with the view direction of the camera. Thus we obtain a complete object surface without sacrificing the flexibility of the approach as compared to e.g. superquadric based approaches. The method has been evaluated using the extensive YCB object and model set [1], and the results indicate that in most cases a very good approximation of the actual object shape can be obtained.

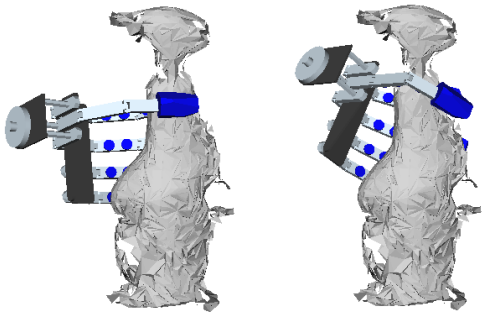


Fig. 6: Two grasps planned on the completed point cloud of the spray bottle, which is one of the rather difficult objects from the test set (see also first row of Fig. 5).

The next step is to use the estimated object model to run a grasp planning algorithm (see Fig. 6) and perform grasping experiments based on the obtained object shape information on a real robot. It will be interesting to evaluate how much the grasp success rate is affected by deviations in the object model, and if and how grasp planning needs to be modified to provide solid grasp configurations under these circumstances. For performing these grasps on a real robot, we hope that a robust grasp execution component that reacts to occurring problems (e.g. as presented in our previous work in [25]) will allow us to compensate for slightly imperfect grasps.

References

[1] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *IEEE International Conference on Advanced Robotics (ICAR)*, 2015.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis - a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[3] G. Kootstra, M. Popovic, J. Jørgensen, K. Kuklinski, K. Miatliuk, D. Kragic, and N. Krüger, "Enabling grasping of unknown objects through a synergistic use of edge and surface information," *International Journal of Robotics Research*, vol. 31, no. 10, pp. 1190–1213, 2012.

[4] D. Fischinger and M. Vincze, "Empty the basket - a shape based learning approach for grasping piles of unknown objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 2051–2057.

[5] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[6] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3d modelling of novel objects from a single view," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 3700–3705.

[7] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: the case for superquadrics with global deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 131–147, Feb 1990.

[8] G. Biegelbauer and M. Vincze, "Efficient 3d object detection by fitting superquadrics to range image data for robot's object manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 1086–1091.

[9] K. Duncan, S. Sarkar, R. Alqasemi, and R. Dubey, "Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[10] M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, and L. J. Guibas, "Example-based 3d scan completion," in *Symposium on Geometry Processing*, 2005, pp. 23–32.

[11] R. Schnabel, P. Degener, and R. Klein, "Completion and reconstruction with primitive shapes," in *Computer Graphics Forum*, vol. 28, no. 2, 2009, pp. 503–512.

[12] R. A. Grupen, T. C. Henderson, and C. D. Hansen, "Apparent symmetries in range data," *Pattern recognition letters*, vol. 7, no. 2, pp. 107–111, 1988.

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

[14] O. Kroemer, H. Ben Amor, M. Ewerton, and J. Peters, "Point cloud completion using extrusions," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 680–685.

[15] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergstrom, D. Kragic, and A. Morales, "Mind the gap - robotic grasping under incomplete observation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 686–693.

[16] A. Huamán Quispe, B. Milville, M. A. Gutiérrez, C. Erdogan, M. Stilman, H. Christensen, and H. B. Amor, "Exploiting symmetries and extrusions for grasping household objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[17] D. Schiebener, J. Morimoto, T. Asfour, and A. Ude, "Integrating visual perception and manipulation for autonomous learning of object representations," *Adaptive Behavior*, vol. 21, no. 5, pp. 328–345, 2013.

[18] D. Schiebener, A. Ude, and T. Asfour, "Physical interaction for segmentation of unknown textured and non-textured rigid objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4959–4966.

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[20] P. Azad, "Integrating Vision Toolkit (IVT)," 2011. [Online]. Available: http://ivt.sourceforge.net

[21] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3," 2010. [Online]. Available: http://eigen.tuxfamily.org

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," in *Communications of the ACM*, vol. 24, 1981.

[23] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[24] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[25] D. Schiebener, N. Vahrenkamp, and T. Asfour, "Visual collision detection for corrective movements during grasping on a humanoid robot," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014, pp. 105–111.