Planning Grasps for Robotic Hands using a Novel Object Representation based on the Medial Axis Transform

Markus Przybylski, Tamim Asfour and Rüdiger Dillmann

Abstract— **Many supporting activities that future service robots might perform in people's homes depend on the capability to grasp and manipulate arbitrary objects. Easily accomplished by humans, but very difficult to achieve for robots, grasping involves dealing with a high-dimensional space of parameters which include hand kinematics, object geometry, material properties and forces. We believe that the way a robot grasps an object should be motivated by the object's geometry and that the search space for stable grasps can be dramatically reduced if the underlying object representation reflects symmetry properties of the object that contain valuable information for grasp planning. In this paper, we introduce the *grid of medial spheres*, a volumetric object representation based on the medial axis transform. The grid of medial spheres represents arbitrarily shaped objects with arbitrary levels of detail and contains symmetry information that can be easily exploited by a grasp planning algorithm. We present the data structure as well as a grasp planning algorithm that exploits it and provide experimental results on various object models using two robot hands in simulation.**

I. INTRODUCTION AND RELATED WORK

Life expectancy in our society increases and elderly people need assistance in their homes in order to lead self-determined lifes despite diminishing physical strength. In order to help people doing the household chores future intelligent domestic robots have to be able to grasp and manipulate all kinds of objects. Due to the large variety of objects it is impossible to predefine the way objects are grasped. Consequently grasp planning has to be done in an automated fashion.

A. Grasp Planning

A variety of concepts and methods for grasp planning has been presented over the last decades. One branch of research focused on grasping at the contact level, where the primary goal is to find a predefined number of contact points, regardless of hand kinematics [1]. Some work [2] examines measures for dexterity, equilibrium, stability and dynamic behavior and investigates how grasps with these properties can be synthesized.

Over the past years simulators such as GraspIt! [3], OpenRAVE [4] and Simox [5] were introduced and now a large body of work exists where grasp planning algorithms have been developed and evaluated in simulators, taking advantage of the fact that these simulation environments allow hand kinematics, geometries of hands and objects, physical and material properties and environmental obstacles to be taken into account. Vahrenkamp et al. [6] presented an integrated approach that combines searching for a feasible grasp with searching for collision free grasp motions. Berenson et al. ([7],[8]) introduced a grasp scoring function that considers grasp quality as well kinematic reachability and obstacles in the environment. Ciocarlie et al. [9] proposed to plan grasps in a low-dimensional subspace of the hand configuration space, resulting in the concept of *eigengrasps*. This work was continued by Goldfeder et al. [10] who built a database of grasps for many hands and objects and used shape similarity to plan grasps by retrieving similar objects from the database. In [11] they applied their method to novel objects by aligning and matching partial 3D data to objects in the database in order to find feasible grasps.

There exists a family of simulator-based grasp planning methods that decompose objects into parts and plan grasps on the individual parts. Miller et al. [12] presented the first of these methods. He manually decomposed objects into boxes, spheres, cylinders and cones and planned grasps on the individual primitives. Goldfeder et al. [13] represented objects as a tree of superquadrics. Huebner et al. [14] presented a method to approximate objects by a set of minimum volume bounding boxes. While these methods based on the *grasping by parts* paradigm effectively allow to prune the search space of possible candidate grasps, their common limitation is the fact that the result of the decomposition process cannot approximate the original shape of the object with arbitrary precision, which in turn may sacrifice potential high-quality candidate grasps.

Recently, Aleotti et al. [15] presented a skeleton-based approach that uses the Reeb Graph as an object representation. The object is topologically segmented into parts and classified as member of a specific class of objects. Then grasps are planned on the part queried by the user. The work is aimed at planning grasps across similar objects on the assumption that an ontological description of the class is given a priori.

In our previous work [16], we presented a first step toward an object representation for grasp planning that follows the idea of the *grasping by parts* paradigm but avoids sacrificing potential high-quality candidate grasps because of poor object geometry approximation. We introduced a grasp planning algorithm based on the medial axis that evaluates local symmetry properties of objects in order to generate geometrically meaningful candidate grasps that have a high probability to result in force-closure grasps. The algorithm identified structures in slices of the medial axis and generated

This work was supported by EU through the project GRASP.

All authors are with the Institute for Anthropomatics, Karlsruhe Institute of Technology, Karlsruhe, Germany {markus.przybylski, asfour, dillmann}@kit.edu

candidate grasps depending on which slice structures were present in the object. This approach had the limitation that candidate grasps could only be generated for objects whose medial axis contained a predefined set of slice structures. In order to deal with more complicated object geometries, the user had to define additional medial axis slice structures.

In this paper we introduce a novel grasp planning method that eliminates the limitations of our previous method [16] in the sense that it works for arbitrarily shaped objects without the need for the user to define structures in the medial axis to be identified by the grasp planner. We achieve this by representing the object as a three-dimensional grid of medial spheres, an object representation based on the medial axis transform, which approximates object geometry with arbitrary precision, allows for geometric analysis of individual regions of the object geometry and contains object symmetry information which is exploited by our grasp planner. Our contribution is the object representation and the associated grasp planning algorithm.

The remainder of this paper is organized as follows. In section II, we present our object representation and its construction. In section III, we explain the details of our new grasp planning algorithm. In section IV, we show experimental results using two robot hand models and various object models in the OpenRAVE [4] simulation environment. In section V, we conclude the paper with a summary and some ideas for future research.

II. An object representation for grasp planning

In this section we describe a novel object representation which will serve as a basis for our new grasp planning method explained later in this paper. The core idea of the work presented in this paper is that a grasp planning method should benefit from an object representation that describes an object's geometry in terms of volume as opposed to common mesh representations that only describe the surface of the object. Our novel object representation is based on the medial axis transform (MAT) of an object. The medial axis (MA), originally introduced by Blum [17], is a topological representation of an object that represents objects by inscribing spheres of maximum diameter, i.e. spheres that touch the boundary of the original shape at two or more distinct points. While the MA is the union of the inscribed spheres' centers, the MAT additionally contains the radii of the spheres. The MAT is a complete shape descriptor, i.e. it contains all necessary information to reconstruct the original object's shape. For the experiments in this paper, we first compute the MA and then reconstruct the MAT from the MA point cloud and a surface point cloud of the respective object.

A. Reconstructing the MAT from the MA

In this section we explain a simple algorithm to reconstruct the MAT from the MA. In a first step, we need to obtain a surface sampling point cloud of the object. This can be achieved in at least two different ways. If the surface mesh of the object is sufficiently dense, we can simply use the vertices of the mesh as a surface point cloud. Otherwise, a sampling method based on ray collision as described in [7] can be used. In a second step we compute the medial axis (MA) of the object. Computing the MA in a robust fashion is quite hard [18]. Therefore, we use the *Tight Cocone* tool [19] for this purpose which calculates a medial axis point cloud, based on the surface point cloud from the previous step. The third step consists in the reconstruction of the MAT of the object based on the MA point cloud and the surface point cloud. As described above, the MAT inscribes spheres of maximum diameter into the shape. The spheres have to touch the shape boundary at least at two different points. For each medial axis point, we are interested in the corresponding sphere radius and the nearest points on the object's surface. A brute-force approach would be to compute the distances to all surface points. Given a surface point cloud with n points and a MA point cloud with m points, nm distance computations would have to be performed, which becomes prohibitively expensive for bigger input data sets. To avoid exhaustive computation of distances, we use a cell-based algorithm for nearest neighbor search inspired by [20]. Bentley et al. [20] describe an algorithm which uses a precomputation step to sort the point cloud and the query point into a cartesian grid of cells. This can be achieved in linear time with respect to the size of the input point cloud. During the subsequent nearest neighbor search, first the cell containing the query point is identified. Then, only neighbor cells to the first cell are examined for possible nearest neighbor points. The search proceeds from inner to more outer layers of neighbor cells around the first cell until the first neighbor point p has been found. In the concluding step some more cells are examined to check if there is a closer neighbor than p. We apply this algorithm to our problem of finding the closest surface points for each medial sphere's center. We use the following criterion to sort the surface points as well as the medial axis points into a three-dimensional cartesian grid structure:

$$\begin{pmatrix} i_x \\ i_y \\ i_z \end{pmatrix} = \begin{pmatrix} \lfloor n_x(x - x_{min})/(x_{max} - x_{min}) \rfloor \\ \lfloor n_y(y - y_{min})/(y_{max} - y_{min}) \rfloor \\ \lfloor n_z(z - z_{min})/(z_{max} - z_{min}) \rfloor \end{pmatrix} \quad (1)$$

In equation (1) the variables i_x, i_y, i_z are the indices of the cell in the grid where a point $q = (x, y, z)$ is stored. n_x, n_y, n_z denote the number of cells of the grid along the respective coordinate axes. $x_{min}, y_{min}, z_{min}$ denote the minimal coordinates, $x_{max}, y_{max}, z_{max}$ the maximal coordinates of the axis-aligned bounding box around the surface point cloud. Bentley et al. [20] used in their work a grid of size $(n/C)^{\frac{1}{2}}$ by $(n/C)^{\frac{1}{2}}$ cells for their two-dimensional point cloud, where n denotes the number of points in the point cloud and C is referred to as the *cell density*, i.e. the expected number of points in each cell. In our case, the surface point cloud is contained in a cuboid whose side lengths are not necessarily equal. Yet, in order to allow for Euclidean distance computations on our grid, the individual cells have to be cubes. Therefore we choose our grid to be an array of size $k_1(n/C)^{\frac{1}{3}}$ by $(n/C)^{\frac{1}{3}}$ by $\frac{1}{k_1}(n/C)^{\frac{1}{3}}$ cells, where k_1 is the ratio of the longest edge to the shortest

edge of our surface point cloud's axis-aligned bounding box. Good parameter values are $C = 2$ for thin objects and $C = 4$ for the general case. Smaller values of C lead to less points per cell but more cells to be searched, making the process of searching cells more expensive than the distance computations between MA points and surface points. Given this cell-based data structure, for each MA point we are now able to reconstruct the sphere radius and the points where the sphere touches the object's surface simply by determining the cell c_0 which contains the MA point and then searching for nearest surface points by considering neighbor cells with growing distances to c_0.

B. The grid of medial spheres

We represent our object as a collection of medial spheres which we reconstructed as explained in the previous section. Each individual sphere has the following attributes:

- The position X of the sphere's center.
- The sphere's radius r.
- The points p_s where the sphere touches the object's surface.
- The object angle α_o [21], also referred to as the bisector angle [22], solid angle [23], medial angle [24] or separation angle ([25],[26]), which describes the maximum angle included by two vectors from the sphere's center to two different nearest surface points p_{s_i} and p_{s_j} touched by this sphere (see Fig. 1b, 1c).

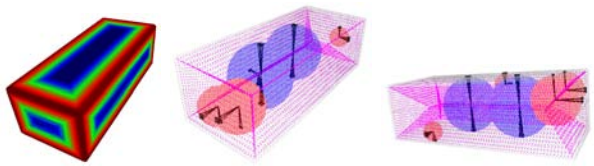
In order to allow for efficient evaluation of the spheres in each sphere's neighborhood we use again the grid data structure presented in the previous section. By sorting the spheres with respect to their center coordinates X_i into the grid cells, we obtain a grid of medial spheres on which we will perform grasp planning in the next section.

III. Grasp planning on a grid of medial spheres

Our approach for grasp planning consists of three steps. In the first step we select the spheres from the grid that should be considered for grasp planning. In the second step, we examine the local neighborhood of each of the selected spheres and analyze each neighborhood's symmetry properties. In the third step, we use the results of the second step to generate candidate grasps which are then tested for force-closure. We begin with the selection of spheres for further consideration.

A. Selection of spheres to analyze for grasp planning

We now have a three-dimensional grid of possibly thousands of spheres describing our object's geometry. Which of these spheres contain valuable information that should be considered for generating candidate grasps? As explained earlier in this paper, the MAT is a complete shape descriptor, i.e. it contains all information to reproduce the original shape. However, for grasp planning, we do not need to consider all the details of the object's geometry, but only the most relevant properties. A very basic class of grasps are grasps consisting of two virtual fingers which oppose [27] each other and touch the object at two opposing sides. The two virtual fingers can be the two jaws of a parallel jaw gripper as well as the thumb of an anthropomorphic robot hand opposing the other fingers of the hand. Our goal is to identify spheres in our object representation that can be used to generate this kind of grasps. In our approach we consider two key parameters of the spheres: the object angle α_o and the diameter d.



(a) (b) (c)

Fig. 1: Shape approximation by inscribing spheres: Rectangular box with all spheres (a), spheres with object angle $\alpha_o \approx 180°$ (blue) and $\alpha_o \approx 90°$ (red) highlighted (b),(c).

We use the object angle α_o as an indicator of a sphere's significance for grasp planning. Spheres with big object angles contribute to a rough approximation of the object's shape and volume whereas spheres with small object angles rather describe surface details of the object. In our object representation, a box-shaped object, for example, consists of spheres with $\alpha_o \approx 180°$ which describe a rough approximation of the object's shape and volume and of spheres with $\alpha_o \approx 90°$ which describe the edges and corners of the box (Fig. 1). For our grasp planning method we are more interested in the rough shape of the object, which reflects the object's thickness and its symmetry properties. In the remainder of this paper, we consider only spheres with an object angle of $\alpha_o \geq 120°$, for the following two reasons: On the one hand, this eliminates edges and other surface details as well as possible instabilities of the MA. On the other hand, it preserves the overall symmetry properties of the object to be evaluated by our grasp planning algorithm.

The sphere diameter d is important for the following considerations: First, spheres might be too big for the hand to grasp. In this case we can still use them to analyze the rough shape of the object, but it does not make sense to generate candidate grasps on them. Second, spheres might be too small to be interesting for grasping. Third, small spheres may contribute to the object's surface details. Therefore, it can be useful to discard them in some cases, as we will see later in this paper.

B. Analyzing the symmetry properties of a sphere's local neighborhood

After having selected a reduced set of spheres with $\alpha_o \geq 120°$ from our object representation, our next goal is to extract symmetry properties of the object from this reduced set of spheres. We accomplish this by using the following method: For each sphere s in the remaining set of spheres we consider the spheres s_N in a local spherical neighborhood N of search radius r_{search} around s. As the spheres are stored in a grid, we can efficiently retrieve these neighbor

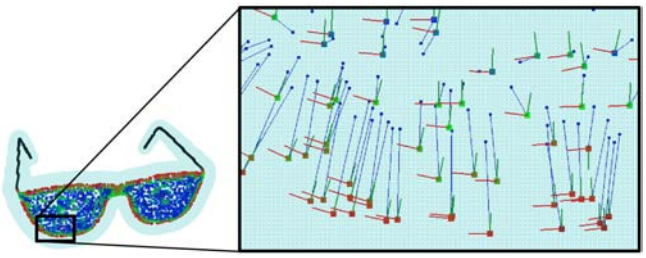


Fig. 2: Estimating the local neighborhood of spheres. Whole object (left), magnified details (right). Big dots indicate sphere centers, with colors from black over red and green to blue symbolizing growing values of ρ_{ev}. Red and green lines indicate first (red) and second (green) eigenvectors $\vec{e}_1$ and $\vec{e}_2$. A small blue dot indicates the center of gravity of each sphere's local neighborhood, connected to the sphere's center by a blue line. The light blue area indicates the size of the local neighborhoods.

spheres by considering neighbor cells of the cell containing the currently considered sphere s. We perform principal component analysis (PCA) on the sphere centers of s_N. Let $\vec{e}_1$ and $\vec{e}_2$ be the first two eigenvectors and λ_1 and λ_2 the corresponding eigenvalues resulting from the PCA. To decide which method should be used to generate candidate grasps, we evaluate the ratio ρ_{ev} of the first two eigenvalues λ_1 and λ_2:

$$\rho_{ev} = \frac{\lambda_2}{\lambda_1} \tag{2}$$

By comparing the value of ρ_{ev} to two constants ρ_{axis} and ρ_{plane} we classify the sphere s into one of the following categories:

- If $\rho_{ev} \leq \rho_{axis}$ we assume that the sphere centers of s_N are part of a local symmetry axis.
- If $\rho_{axis} < \rho_{ev} < \rho_{plane}$ we assume that the sphere centers of s_N are part of a local symmetry plane, and that the currently considered sphere is located at the rim of this plane.
- If $\rho_{ev} \geq \rho_{plane}$ we assume that the sphere centers of s_N are part of a local symmetry plane, but that the currently considered sphere is located inside the plane.

For all experiments in this paper, we used the values $\rho_{axis} = 0.01$ and $\rho_{plane} = 0.4$, which were determined empirically. Fig. 2 illustrates the results of the PCA, where the sphere centers are colored differently depending on the respective value of ρ_{ev}. There is one special case where the symmetry properties of a sphere's local neighborhood N cannot be analyzed. This is the case if N does not contain any further spheres. This occurs if the object's shape or parts of it are perfectly spherical. In our current implementation, we do not treat this case.

C. Generating candidate grasps

In the previous section we stated how we use PCA to identify symmetry structures in local regions of the object's MAT. In this section we use this information to generate candidate grasps that can later be tested for force-closure. Before we explain our method for candidate grasp generation, we define which parameters we use to describe a candidate grasp. From Berenson's grasp parameters [7] we adopt the approach direction P_d of the hand toward the object and the preshape P_p which is a vector of joint angles of the hand. We introduce the following new parameters:

- A 3D grasp target point P_g the hand should approach during grasping. This is similar to Berenson's P_t, but in our case it is not necessarily located on the object's surface.
- A hand orientation vector P_o which can be thought of as an axis the hand is supposed to wrap around during a parallel grasp for a cylindrical object. For each candidate grasp, P_o completely defines the hand's target orientation.

As already stated above, we distinguish three kinds of outcomes of the PCA of a sphere's local neighborhood. In the following, we explain how we use the results of the PCA to generate candidate grasps. We only generate candidate grasps for spheres that are not too big to be grasped by the respective robot hand, which can be easily determined from the respective sphere's diameter. In all cases, the sphere's center is used as the grasp target point P_g for the hand.

1) Local Symmetry Axis: For spheres with a local symmetry axis, we generate multiple candidate grasps with approach directions P_d of the hand orthogonal to the symmetry axis. The direction of the symmetry axis is the first eigenvector $\vec{e}_1$ from the PCA. It serves as the hand's orientation vector: $P_o = \vec{e}_1$.

2) Rim of a Local Symmetry Plane: For spheres on the rim of a local symmetry plane, we generate candidate grasps with approach directions P_d of the hand orthogonal to the rim. The two eigenvectors $\vec{e}_1$ and $\vec{e}_2$ from the PCA describe the orientation of the local symmetry plane. $\vec{e}_1$ is parallel to the rim of the local symmetry plane and serves again as the hand's orientation vector: $P_o = \vec{e}_1$. The approach direction P_d of the hand can be computed from $\vec{e}_2$. In order to make sure that the hand's approach direction points toward the rim from the outside of the local symmetry plane, we consider the vector $\vec{v}_{COG}$ from the sphere's center to its local neighborhood's center of gravity COG_N. If $\vec{e}_2$ and $\vec{v}_{COG}$ include an angle $\beta \geq 90°$, then we set $P_d = -\vec{e}_2$, otherwise $P_d = \vec{e}_2$.

3) Interior of a Symmetry Plane: For spheres located inside a local symmetry plane, we do not generate any candidate grasps.

IV. EXPERIMENTS

In this section we present some experiments for the grasp planning algorithm described above. We use models of the Barrett hand and of the hand of our humanoid robot ARMAR-III [31] to test our grasp planning algorithm in OpenRAVE. In our experiments we assume that the biggest sphere the ARMAR-III hand can grasp has a diameter of 9.4cm, whereas the corresponding value for the Barrett hand is assumed to be 14.0cm. We use a parallel preshape for all

Fig. 3: Object set 1: objects taken from the Chen mesh segmentation benchmark database [28].

Fig. 4: Object set 2: objects scanned using a 3D laser scanner([29],[30]).

candidate grasps. We perform experiments on two different sets of objects.

Object set 1 (Fig. 3) contains some objects from the Chen mesh segmentation benchmark [28], whereas object set 2 (Fig. 4) contains some objects that we scanned with a 3D laser scanner ([29],[30]). The upper rows of Fig. 5 and 6 show the grid of medial spheres representations of some of our test objects with colors ranging from red over green to blue for increasing sphere diameters. The middle and lower rows depict the respective reduced set of sphere centers for grasp planning, with estimations of the symmetry properties of the spheres' local neighborhoods and a set of derived candidate grasps. The candidate grasps are represented by lines indicating their approach directions P_d. Green lines depict candidate grasps originating from a local symmetry plane, whereas orange lines depict candidate grasps originating from a local symmetry axis. Short magenta lines at the end of the approach directions indicate the respective hand orientation P_o. The depicted candidate grasps are those candidates generated for the ARMAR-III hand. Clusters of bright pink dots in the pictures indicate centers of spheres that are too big for the ARMAR-III hand to grasp. Therefore no candidate grasps were generated for these spheres.

In both of our two experiments we randomly pick 200 spheres from the reduced set of spheres, generate candidate grasps for these spheres and test these candidates for force-closure. The test procedure for the candidate grasps follows the approach described in [7]. For each candidate grasp, the hand is first set into the grasp target point P_g with the hand's palm direction aligned with the candidate grasp's approach direction P_d and the hand rotated around the approach direction according to the orientation vector P_o. The hand is then retracted from the object along $-P_d$ until it does not collide anymore with the object's surface mesh. Then the

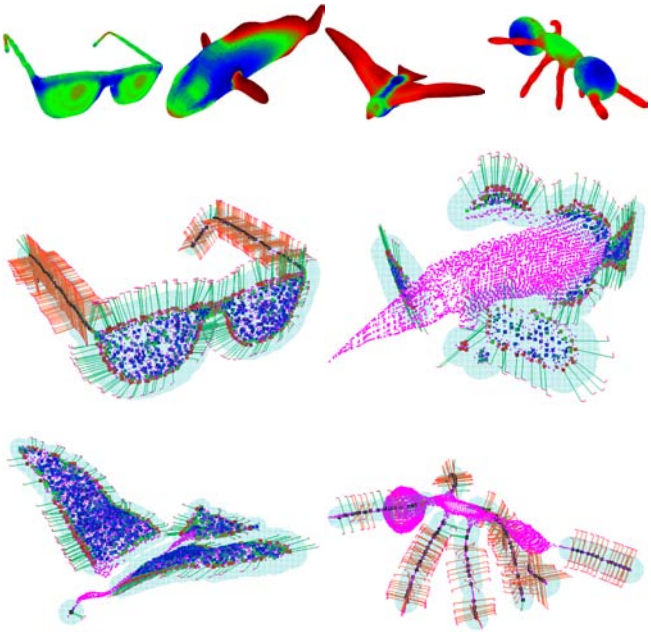


Fig. 5: Upper row: Grid of medial spheres representation for some objects from object set 1. Middle and lower row: reduced set of sphere centers with generated candidate grasps for these objects.

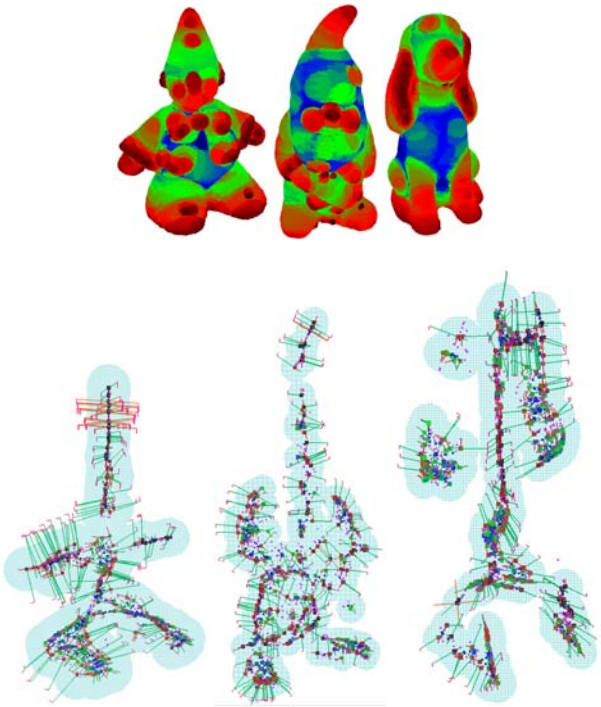


Fig. 6: Upper row: Grid of medial spheres representation for some objects from object set 2. Lower row: reduced set of sphere centers with generated candidate grasps for these objects.

fingers are closed around the object until collision occurs or the joint limits are reached. Based on the contacts between the object's surface mesh and the hand we compute the epsilon measure for force-closure [32]. For all experiments, we use a point contact model with Coulomb friction and assume a friction coefficient of 0.5.

In our first experiment we use object set 1 (Fig. 3). Most of these objects are complex objects, decomposable into distinct parts. Most of them have relatively flat surfaces with few or no details. They are big relative to the robot hands, so not all of their components can be grasped. It matters which part of the object is approached by the hand, from which direction and with which orientation of the hand. For all objects we test a full-size version as well as a version which has been scaled down to 50% of the original size. The results of the experiment are given in Table I, where in the ID column the index of the respective object in the Chen mesh segmentation benchmark is given. For most of the full-scale objects, well above 50% of the tested candidates are force-closure grasps. For the half-sized objects the results are similar, though in some cases the percentage of force-closure grasps drops significantly. This is the case for the glasses, and in case of the Barrett hand, also for the monster and tea kettle. The reason for this is that for these objects many candidates where generated for parts of the object that are now too small to be stably grasped, like the earpieces of the glasses, for example. In our second experiment we consider object set 2 (Fig. 4). These are more compact objects, not easily decomposable into basic parts. In contrast to object set 1, these objects are about the same size as our robot hands. However, the majority of these objects has considerable surface details. In this experiment we consider only those spheres of the reduced sphere set for candidate grasp generation that have a diameter $d \leq 0.3 d_{max}$, where d_{max} is the diameter of the biggest sphere of the respective object. This keeps the planner focused on the rough geometry of the object and avoids consideration of surface details. The effect can especially be seen for the clown, dog and lawn gnome objects in Fig.6 where the planner reliably identifies the vertical main axis of the objects and generates candidate grasps accordingly. The results of the experiment are given in Table II. Again, for most of the objects, well above 50% of the tested candidates are force-closure grasps. Some typical grasps generated by our planner are depicted in Fig. 7.

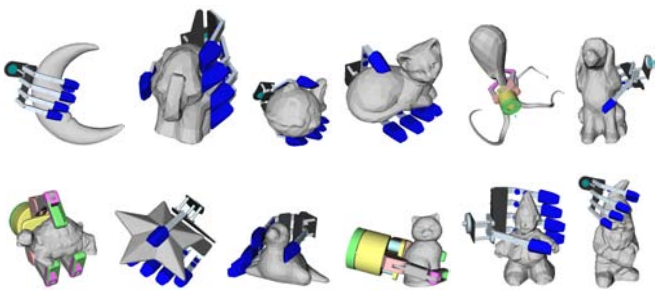


Fig. 7: Some example grasps generated by our method.

TABLE I: Percentage of force-closure grasps among the tested candidate grasps for object set 1

Objects		ARMAR-III hand		Barrett hand	
ID	name	scale 1.0	scale 0.5	scale 1.0	scale 0.5
1	Female doll	71.3%	54.6%	53.13%	37.9%
41	Glasses	93.9%	7.8%	73.7%	10.7%
81	Ant	94.4%	71.1%	61.3%	45.7%
101	Chair	89.6%	49.2%	73.9%	72.2%
125	Octopus	53.7%	55.2%	26.9%	44.7%
141	Table	91.9%	92.5%	94.6%	85.0%
161	Teddy	100.0%	83.3%	86.7%	51.2%
225	Fish	76.5%	83.3%	68.4%	81.1%
245	Bird	75.0%	68.3%	75.0%	65.6%
290	Monster	70.5%	64.7%	67.8%	38.2%
305	Bust	50.0%	70.0%	100.0%	92.9%
361	Vase	76.8%	65.3%	69.6%	55.1%
379	Tea kettle	78.9%	63.2%	75.7%	31.3%
390	Giraffe	85.5%	68.3%	71.4%	56.0%

TABLE II: Percentage of force-closure grasps among the tested candidate grasps for object set 2

Objects	ARMAR-III hand	Barrett hand
Clown	63.5%	61.2%
Elefant	75.3%	76.0%
Owl	78.0%	68.2%
Spheric fish	59.0%	78.3%
Lawn gnome	53.1%	57.7%
Heart	89.0%	77.0%
Dog	63.7%	69.2%
Sitting cat	64.9%	59.5%
Lying cat	80.7%	80.7%
Moon	58.9%	64.4%
Mushroom	80.0%	55.5%
Turtle	57.1%	70.3%
Seal	73.5%	59.2%
Star	44.4%	66.7%

In the following, we would like to illustrate some advantages of our method. Our method generates grasps only on parts of the objects that are not too thick for the hand. Consider for example Fig. 8. For the full-scale giraffe (Fig. 8a) the algorithm generates only grasps on the head (Fig. 8b), the neck, the legs, the tail (Fig. 8c), but not on the animal's torso. For a 25% scale version of the animal also the torso becomes graspable (Fig. 8d). We can also tell the planner to ignore thin parts of the object and to grasp only parts where the object's thickness is close to the robot hand's maximum graspable diameter (Fig. 8e, 8f). The algorithm also reliably finds grasps for objects where the ways the object can be grasped are very restricted due to size, like for the bust in Fig. 9. Especially for big objects, like the tea pot or the vase in Fig. 10 our approach also finds mainly grasps at the handles, without any semantic knowledge such as task dependency, but only due to geometric considerations, as in both cases the hollow body is too thick to grasp.

We used an Intel Core 2 Duo T9300 @2.5Ghz with 4GB RAM and Ubuntu 8.04 LTS as a test platform. Performance of the individual methods was as follows. MA computation via Tight Cocone (section II-A) took about 5s to 30s per

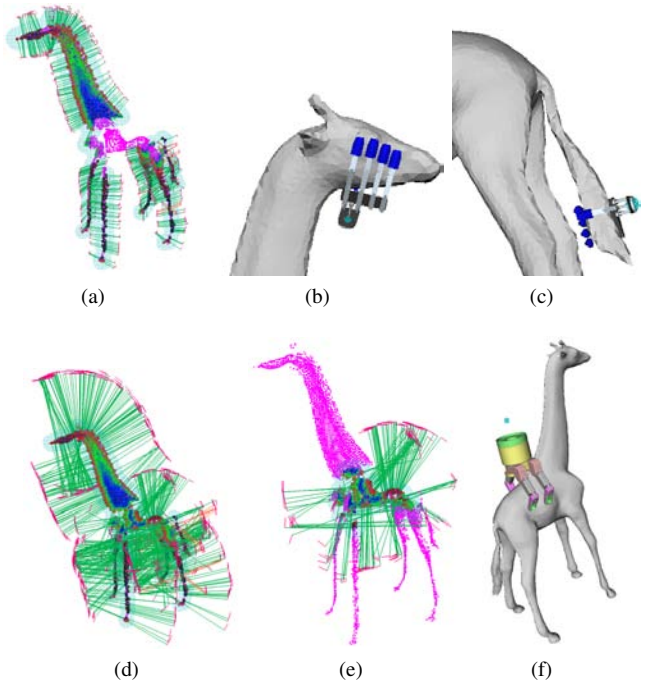


Fig. 8: Candidate grasp generation depending on sphere diameter. For the full-scale giraffe (a) candidate grasps are only generated on the extremities (b,c). For the 25% scale giraffe also the torso can be grasped (d). Considering only big spheres for candidate grasp generation (e,f).

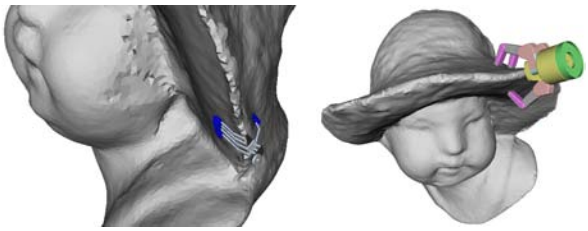


Fig. 9: The algorithm also finds grasps for objects where the ways the object can be grasped are very restricted.

object, depending on the actual size of the input surface point cloud. MAT reconstruction (section II-A) took beetween 13s and 2313s for the objects of object set 1, with an average of 692s, and 127s to 314s for the objects of object set 2, with an average of 228s, depending on the size of the MA point cloud and the size of the surface point cloud. Given the grid of medial spheres, symmetry analysis and candidate grasp generation (sections III-B and III-C) took about 10 to 100ms per sphere, depending on the number of spheres in the local neighborhood. Testing of candidate grasps (section IV) took about 0.5s per candidate grasp. At first glance, some of these computation times might seem quite long. However, we would like to point out some further facts. For the experiments in this paper, we implemented the methods for MAT reconstruction, candidate grasp generation and candidate grasp testing in Python. Considerable speedups

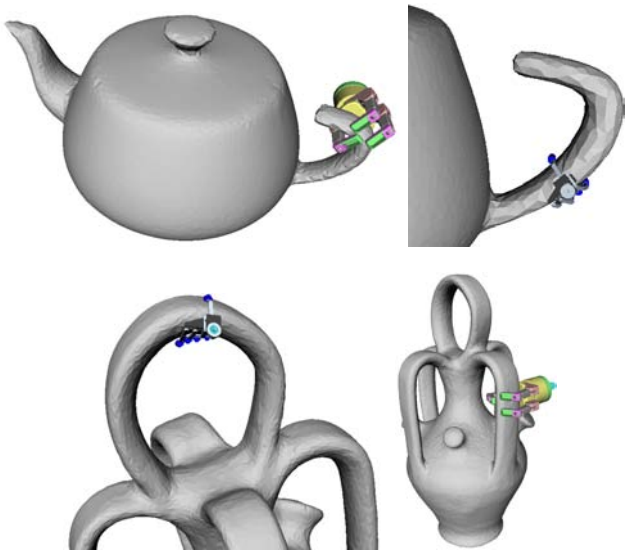


Fig. 10: The algorithm finds grasps on handles simply due to geometric considerations, without semantic knowledge.

should be possible by reimplementation of these methods in C++. Above that, a special case is the MAT reconstruction. For the sake of simple implementation we chose to reconstruct the MAT from the MA. However, we point out that during MA computation, Tight Cocone computes all the MAT parameters (nearest surface points, sphere radii, object angles) internally [19], but unfortunately does not report this information in its output. Therefore, we are optimistic that direct computation of both the MAT and the grid of medial spheres should be possible with computation times comparable to the MA computation time with Tight Cocone, or even faster, if GPUs are used for parallelization.

V. DISCUSSION AND CONCLUSION

In this paper, we presented a novel object representation for grasp planning, the grid of medial spheres, which is based on the medial axis transform as well as a novel grasp planning algorithm that operates on this representation. We performed experiments on two different sets of objects and demonstrated that many of the generated candidate grasps result in force-closure grasps, while at the same time the generated grasps are geometrically meaningful.

The approach presented in this paper overcomes several restrictions and limitations of our previous work in [16]. As our new algorithm considers only local regions of the object geometry, it is not restricted to objects with special geometric properties, but can be used for objects of arbitrary shape. As the grid of medial spheres representation is based on a complete shape descriptor, it can approximate the object's shape very accurately. Yet, the representation allows the grasp planner to ignore surface details of the object in order to exploit local symmetry properties and to focus on parts of the object where the object's local thickness is suitable for grasping with a given robot hand. This way, the grid

of medial spheres representation allows to effectively reduce the search space for force-closure grasps without sacrificing potentional high-quality candidate grasps due to poor object geometry approximation.

In theory, for each of the spheres in our representation one or more candidate grasps can be generated. While generating candidate grasps on a per-sphere basis produces high-potential candidate grasps for arbitrary shapes, there are some special cases, where this approach does not generate those grasps a human being would anticipate. This is the case for objects with concavities, where the typical grasps used by human beings would be grasps where the fingers encompass the concavity without reaching into it. For a coffee cup, for example, our approach would generate grasps at the handle and at the rim of the concavity, but no grasps that encompass the concavity of the cup from the outside without reaching into it. A possible solution to that limitation might be to compute the medial axis transform of the concavity, using the original medial axis transform of the object as input, so the concavity's properties could be exploited for candidate grasp generation. We leave the investigation of this idea as well as the validation of the generated grasps on our real ARMAR-III robot to future work.

VI. ACKNOWLEDGMENTS

We wish to thank Professor Tamal Dey from The Ohio State University for providing us with his software and Rosen Diankov from Carnegie Mellon University for excellent support in all questions regarding the OpenRAVE simulator. We wish to thank Alex Kasper for his help during scanning of the test objects.

REFERENCES

[1] Y. H. Liu, M. Lam, and D. Ding, "A complete and efficient algorithm for searching 3-d form-closure grasps in discrete domain," *IEEE Transactions on Robotics*, vol. 20, no. 5, pp. 805–816, 2004.

[2] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *The Int. Journal of Humanoid Robotics*, vol. 15, no. 3, pp. 230–266, 1996.

[3] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110 – 122, Dec. 2004.

[4] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robotics Institute, Tech. Rep. CMU-RI-TR-08-34, July 2008. [Online]. Available: http://openrave.programmingvision.com

[5] [Online]. Available: http://www.sourceforge.net/projects/simox

[6] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated grasp and motion planning," in *ICRA*, Anchorage, USA, Mai 2010.

[7] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 29 2007-Dec. 1 2007, pp. 42 –48.

[8] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, Dec. 2008, pp. 189 –196.

[9] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dimensionality reduction for hand-independent dexterous robotic grasping," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 29 2007-Nov. 2 2007, pp. 3270 –3275.

[10] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 1710 –1716.

[11] C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. K. Allen, "Data-driven grasping with partial sensor data," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct. 2009, pp. 1278 –1283.

[12] A. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, Sept. 2003, pp. 1824 – 1829 vol.2.

[13] C. Goldfeder, C. Lackner, R. Pelossof, and P. K. Allen, "Grasp planning via decomposition trees," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 4679 –4684.

[14] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 1628 –1633.

[15] J. Aleotti and S. Caselli, "Grasp synthesis by 3d shape segmentation using reeb graphs," in *Intelligent Robots and Systems, 2010. IROS 2010. IEEE/RSJ International Conference on*, 2010.

[16] M. Przybylski, T. Asfour, and R. Dillmann, "Unions of balls for shape approximation in robot grasping," in *Intelligent Robots and Systems, 2010. IROS 2010. IEEE/RSJ International Conference on*.

[17] H. Blum, *Models for the Perception of Speech and Visual Form*. Cambridge, Massachusetts: MIT Press, 1967, ch. A transformation for extracting new descriptors of shape, pp. 362–380.

[18] D. Attali, J.-D. Boissonnat, and H. Edelsbrunner, *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, ser. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, ch. Stability and Computation of Medial Axes: A State-of-the-Art Report, pp. 109–125.

[19] T. Dey and S. Goswami, "Tight cocone: a water-tight surface reconstructor," in *Proceedings of the eighth ACM symposium on Solid modeling and applications*. ACM, 2003, pp. 127–134.

[20] J. L. Bentley, B. W. Weide, and A. C. Yao, "Optimal expected-time algorithms for closest point problems," *ACM Trans. Math. Softw.*, vol. 6, pp. 563–580, December 1980.

[21] B. Miklos, J. Giesen, and M. Pauly, "Discrete scale axis representations for 3d geometry," in *ACM SIGGRAPH 2010 papers*, ser. SIGGRAPH '10. New York, NY, USA: ACM, 2010, pp. 101:1–101:10.

[22] D. Attali and A. Montanvert, "Modeling noise for a better simplification of skeletons," in *Image Processing, 1996. Proceedings., International Conference on*, vol. 3, Sep. 1996, pp. 13 –16 vol.3.

[23] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," in *Proceedings of the sixth ACM symposium on Solid modeling and applications*, ser. SMA '01. New York, NY, USA: ACM, 2001, pp. 249–266.

[24] T. K. Dey and W. Zhao, "Approximate medial axis as a voronoi subcomplex," *Computer-Aided Design*, vol. 36, no. 2, pp. 195 – 202, 2004, solid Modeling and Applications.

[25] M. Foskey, M. C. Lin, and D. Manocha, "Efficient computation of a simplified medial axis," *Journal of Computing and Information Science in Engineering*, vol. 3, no. 4, pp. 274–284, 2003.

[26] A. Sud, M. Foskey, and D. Manocha, "Homotopy-preserving medial axis simplification," in *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, ser. SPM '05. New York, NY, USA: ACM, 2005, pp. 39–50.

[27] T. Iberall, "Human prehension and dexterous robot hands," *The International Journal of Robotics Research*, vol. 16, no. 3, pp. 285–299, 1997.

[28] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3d mesh segmentation," *ACM Trans. Graph.*, vol. 28, pp. 73:1–73:12, July 2009.

[29] A. Kasper, R. Becher, P. Steinhaus, and R. Dillmann, "Developing and analyzing intuitive modes for interactive object modeling," in *International Conference on Multimodal Interfaces*, 2007.

[30] R. Becher, P. Steinhaus, R. Zöllner, and R. Dillmann, "Design and implementation of an interactive object modelling system," in *International Symposium on Robotics*, 2006.

[31] T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "Armar-iii: An integrated humanoid platform for sensory-motor control," in *Proceedings of the 7th IEEE-RAS International Conference on Humanoid Robots*, 2006.

[32] C. Ferrari and J. Canny, "Planning optimal grasps," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, May 1992, pp. 2290 –2295 vol.3.