Computing Convex Grasping Positions for Parallel Jaw Grippers with an Integrated Boundary Layer Mesher

Ludwig Vogt, Tobias Ciemala, Jonas Freitag and Johannes Schilp
Department of Applied Computer Science
University of Augsburg
Augsburg, Germany
Email: {ludwig.vogt, tobias.ciemala, jonas.freitag, johannes schilp}@uni-a.de

***Abstract*—The automated handling of workpieces is one of the main enablers for automatic production systems and requires a stable grasping point determination. To select suitable grasp candidates for STL based models with a planar parallel jaw gripper we propose a two stage algorithm. First, the STL file is remeshed to create a boundary layer containing triangles for every surface of the handling object through a shifting of its edges. In the second step, antipodal grasping positions are determined with the use of local convex hulls, which are generated via a Delaunay3D triangulation, and through an orientation examination to classify the area as convex or concave. Based on this classification all convex points are cross checked to identify pairs of antipodal Points and the associated grasping position. An evaluation of our approach was performed on a test set containing 10 different industrial and everyday objects ranging from gears and shafts to sockets and a freeform body. For all objects a highly diverse solution set could be generated and through the remeshing process the number of grasping positions could be enlarged by at least 38 % for all objects.**

***Index Terms*—robotic grasping, grasping point determination, convex hull, boundary layer mesher**

I. Introduction

In industrial applications planar parallel jaw grippers are commonly used because they are suitable for different objects, e.g., boxes, cylinders, spheres, resulting in a variety of contact shapes, e.g., surface-, line- and point contact [8]. While the determination of surface contacts can be determined via collision detection, shape matching algorithms or Boolean operations [9], a different procedure is necessary for line- and point contacts because their position is preferred to be at a convex place, otherwise a unplanned contact between the gripper and the object is generated. To satisfy this requirement we developed a system for the automatic grasping point determination based on STL files for rigid planar parallel jaw grippers, where we use local convex hulls created from the surface triangles of the CAD model to identify convex grasping positions at the object. In some cases, e.g., extrusions, indentations, holes and notches (cf. Fig. 1) the positioning of the neighbor triangles leads to false negatives and therefore to an incomplete solution set. To identify all solutions, we developed a boundary layer mesher (BLM) which creates a thin boundary layer containing triangles at every surface of the input model and then saves the remeshed model as a STL file again. With the use of the edited STL model the construction of local convex hulls enables us to identify permissible grasping positions on the object even for small and irregular contact shapes.

For the underlying algorithm and especially for the handling of STL files and its tessellated representation containing triangles, a few explanations for the geometrical representation and clustering are introduced [12].

Face: A face represents the most basic geometrical structure in a mesh and is a closed subset of the domain with a Lipschitz-continuous boundary where all nodes are connected through edges – for STL models a face equals a single triangle.

Facet: Adjacent faces of the mesh which have a common normal vector represent a facet.

Surface: Multiple adjacent facets which are clustered together through the angle between their normal vectors γ (cf. Section III-A) are a surface in the mesh.

Mesh: The sum of all constructed surfaces results in a mesh which equals the original STL model.

The paper is organized as follows. Section II covers the past work in the fields of grasping point detection with respect to our problem definition. Afterwards, the methodology and mathematical foundation of our BLM and grasping point determination is formulated. The validity and reliability of our approach is shown in an evaluation setting (cf. Section IV) where we also discuss our results. Lastly, a summary and outlook of our work is given in Section V.

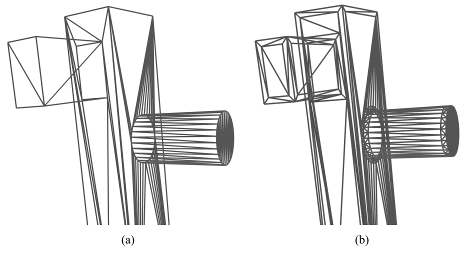

Fig. 1. Comparison of a STL file before (a) and after (b) the remeshing with the BLM.

II. Recent Works

To identify suitable grasping positions two main Methods exist: An analytical approach which calculates the grasping positions via physical and mechanical models [5] and an empirical approach, where historic data is used for machine learning (ML) [3] [16]. In the field of analytical approaches many researchers focus to find a suitable wrench space [14] for human like grasps with robotic hands. The analytical methods also contain camera-based systems, where the grasping of previously unknown objects is possible but because our Algorithm uses CAD models these systems are not described any further. A good summary of existing vison based approaches is given in [6].

A common usage for grasp planning and evaluation are specific physic based simulation environments, e.g., GraspIt! [17] and OpenGRASP [13] where algorithms and grasp evaluation strategies can be implemented. In the GraspIt! simulation environment Miller et al. [18] use primitive shapes to calculate grasping positions. They decompose and model a 3D object with primitive shapes such as cylinders, boxes and spheres with predefined grasping rules. In one of the earlies works Smith et al. [21] used a slicing tool to generate 2D views of the object and calculate grasps for a parallel jaw gripper. Because of the slicing tool their approach depends on the slicing direction and neglects the 3D surrounding of the contact point. Vahrenkamp et al. [22] plan robust grasps by calculating the mean curvature skeletons for 3D mesh objects and align the gripper at suitable positions along the skeleton. Liu et al. [15] propose an algorithm which calculates suitable grasping positions at 3D objects, represented by discrete points, through combining a local heuristic search and a recursive problem decomposition. While their approach is able to generate antipodal contact points for parallel jaw grippers, they disregard the surroundings of the contact point, so their solutions could also be concave areas where a placement of a rigid gripper would be difficult. Honarpardaz and Haschke [9] use tessellated CAD models to build a 3D-kd-tree of face center positions and their associated normal directions. From the kd-tree suitable grasping positions are calculated and ranked. For a selected location the gripping surface is automatically built via a Boolean operation and therefore no knowledge about the geometrical surrounding is needed. A first step towards an automatic unloading process of a 3D printing process is demonstrated in [1]. In their approach, Becker et al. derive the GCode to calculate the position of the object on the build plate and the grasping position. The grippers in their approach are designed for a wide range of product geometries through flexible end pieces at the gripper and therefore a detailed grasping point determination is dismissed.

Harada et al. [7] use a clustering of the faces to identify grasping positions. Their approach was designed for a flexible grasping surfaces which neglects the distinction between concave and convex points at the identified clusters. Recently, Kleeberger et al. [11] used a similar approach but instead of clustering the triangles, their approach is based on a point cloud representation derived from the CAD model. Every datapoint in the point cloud is cross checked with all data points to compute antipodal grips for a parallel jaw gripper. Their strategy results in an extensive solution set but needs $N = \frac{1}{2} \cdot (p^2 - p)$ number of point combinations for p number of points to determine antipodal positions.

III. Methodology

As described previously, the algorithm consists of two main steps, which are a preprocessing step (BLM) and the actual grasping point determination. The BLM (cf. Section III-A) takes the STL file as an input and builds a thin boundary layer into every surface on the handling object. Boundary layer creation tools from available mesh generators which are used in CFD-Simulations were not suitable for our application because these types of boundary layers are created around the selected mesh or surface and are inflating the surface of the object [19]. Whereas in our application the geometry, after remeshing the object, must be identical to the input geometry and therefore a BLM was developed (cf. Fig. 2). Another strategy would be a general smaller tessellation of the STL file similar to a point cloud representation as in [11], but this procedure would enlarge the number of faces significantly and therefore the runtime of the algorithm. After remeshing the handling object, the model is saved as an STL file again and the grasping point determination is performed (cf. Section III-B). The whole algorithm was built in a Python 3.8 environment and besides the vtk library we mainly used Version 3.9.10 of Trimesh [4] to execute mesh operations and manipulations. A model of the used gripper (Schunk Co-act EGP-C 40) is deposited in the algorithm. For sake of representation basic geometries are used in this section, whereas in the evaluation setting more complex geometries are used.

A. Boundary Layer Mesher

The developed BLM consists of 4 main steps: surface subdivision, edge shifting, boundary layer insertion and surface remeshing. Besides not changing the geometry of the object, another criterion is to insert a minimal number of triangles to keep the necessary computational effort as low as possible. For a surface subdivision the triangles $T_k (k = 1, 2, \ldots, z)$, where z denotes the number of triangles in the mesh, are clustered in facets and then in surfaces through the angle $\gamma_{i,j}$ between their normal vectors:

$$\gamma_{i,j} = \arccos \frac{n_i \cdot n_j}{|n_i| \cdot |n_j|}, \tag{1}$$

where n_i denotes the normal vector of a facet from $F_i (i = 1, \ldots, n)$, n_j the normal vector of the clustered surfaces $S_j (j = 1, 2, \ldots, m)$ and n,m denoting the number of facets and surfaces in the mesh. If the angle $\gamma_{i,j}$ is below a set threshold γ_{max}, F_i is added to the surface S_j. After clustering the triangles in surfaces, the enclosing set of edge $E_j (j = 1, 2, \ldots, m)$ of every surface is determined (cf. Fig. 2 a). An edge is defined as the connection of all vertices which

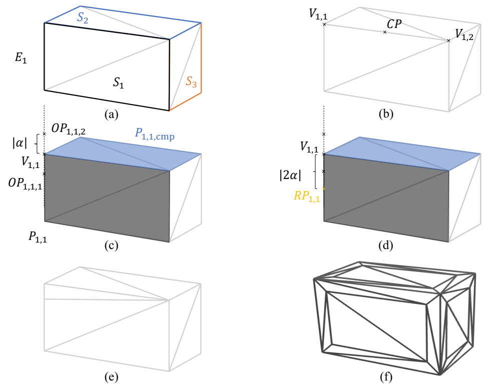

Fig. 2. Workflow of the boundary layer remesher for a hexahedron: (a) clustering of the faces and extracting the surface edge (b) determining edge vertices and center points (c) calculating optimized points and shift edge vertices (d) create a reference point and choose correct optimized point (e) insert triangle (f) loop over all surfaces to create a remeshed STL model.

belong to a minimum of two surfaces and is calculated through the intersection of all facets from the selected surface with all other facets.

$$E_j = \{T_n | (T_n \in F_i(S_j) \cap (T_n \in F_i(\notin S_j))\}. \tag{2}$$

Based on a shallow copy of E_j the edge is modified and shifted inwards. For the insertion of a new boundary, the intersecting vertices $V_{j,l} (l = 1, 2, \ldots, x)$ where x is the number of edge vertices for surface S_j are shifted with two constraints (cf. Fig. 2 c):

Constraint 1: The distance d from the vertex $V_{j,l}$ to the plane $P_{j,l,\text{cmp}}$ of the compared facet is defined by the hyperparameter α which must be set by the operator. The distance from a point to a plane is defined as follows:

$$\begin{aligned} P_{j,l,\text{cmp}} : a_{j,l,\text{cmp}} \cdot x_{j,l} + b_{j,l,\text{cmp}} \cdot y_{j,l} \\ + c_{j,l,\text{cmp}} \cdot z_{j,l} = d, \end{aligned} \tag{3}$$

where $a_{j,l,\text{cmp}}$,$b_{j,l,\text{cmp}}$ and $c_{j,l,\text{cmp}}$ represent the hyperparameters of the plane $P_{j,l,\text{cmp}}$. With this equation the distance d results in:

$$d(V_{j,l}, P_{j,l,\text{cmp}}) = \alpha = \frac{|a_{j,l,\text{cmp}} \cdot x_{j,l} + b_{j,l,\text{cmp}} \cdot y_{j,l} + c_{j,l,\text{cmp}} \cdot z_{j,l} - d|}{(a^2_{j,l,\text{cmp}} + b^2_{j,l,\text{cmp}} + c^2_{j,l,\text{cmp}})^{\frac{1}{2}}}. \tag{4}$$

Constraint 2: The distance from the vertex $V_{j,l}$ to the plane $P_{j,l}$ of its own facet must be zero $d(V_{j,l}, P_{j,l}) = 0$. This equation guarantees that the shifted vertex is in the same facet as before and the geometry of the object is not changing.

To determine the positions of the shifted vertices with a minimum distance to the original vertex an optimization is performed which satisfies the two mentioned constraints. For the optimization a Sequential Least Squares Programming (SLSQP) optimization [20] was used. The optimization results in two points $OP_{j,l,1}$ and $OP_{j,l,2}$ (cf. Fig. 2 c), where one point is located inside the surface and the other outside the mesh. In addition, a created center point CP is used to classify the edge as an outside edge or an inside edge. Depending on the classification, a new reference point $RP_{j,l}$ is inserted with the distance 2α from $V_{j,l}$ in the direction of the inside edge or contrary to the outside edge (cf. Fig. 2 d). The correct vertex of the two optimized points is the point with the shorter distance to the reference point $RP_{j,l}$. After this selection, the calculated vertex is saved as a new node in the STL model and must be provided with a new node id. The allocation occurs in order of the vertices along the edge of the surface (cf. Fig. 2 e). This process is performed for all intersecting vertices resulting in a boundary layer (cf. Fig. 2 f). To ensure a consistent thickness of the boundary layer multiple shifted intersecting vertices are moved equally in all directions through a consideration of their individual unit vectors.

B. Grasping Point Determination

With the updated STL model from the BLM, the subsequent grasping point determination is performed to calculate antipodal convex grasping positions. For a selected triangle T_k in the mesh, the center of gravity is calculated as an auxiliary point AP_k and all adjacent triangles are clustered to span a new surface (cf. Fig. 3 a). For the clustered triangles a convex hull is created with the Delaunay3D triangulation (cf. Fig. 3 c). The next steps of the algorithm aim to classify the clusters as convex or concave. For this purpose, the auxiliary point is shifted along the normal vector n_k of the selected triangle. Through the displacement in this direction with a small value d_{shift} (cf. Fig. 3 d), the auxiliary point is positioned outside the mesh but not necessary outside the convex hull. After displacing the auxiliary point, the triangle is classified as a convex or concave point via two rules:

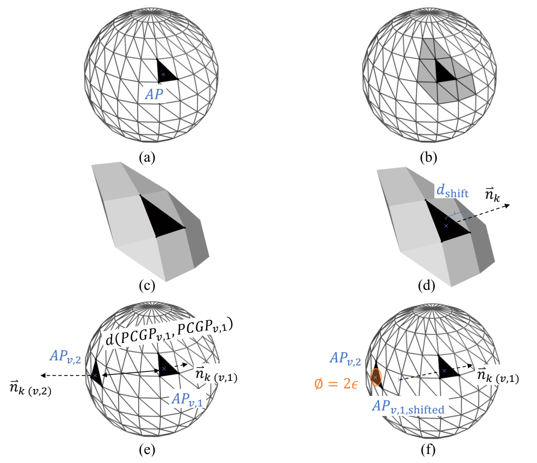

Fig. 3. : Individual steps of the grasping point determination: a) determing the auxiliary point for the selected triangle b) clustering all neighbor triangles c) construct a convex hull via a Delaunay3D triangulation d) shift the auxiliary point and classifiy triangle as convex or concave e) Find antipodal convex triangles f) verify overlapping.

Convex classification: The auxiliary point is positioned outside the convex hull or on its boundary.

Concave classification The auxiliary point is positioned inside the convex hull.

If T_k is classified as convex, the face and its center of gravity are saved as a potential convex grasping position. This procedure is done for all triangles in the mesh resulting in a set $PCGP_v (v = 1, 2, \ldots, s)$ containing all possible solutions s. For the determination of valid grasping positions antipodal combinations from the PCGP set are computed. To represent a grasp, the distance between two points must be smaller than the maximum opening of the gripper $d(PCGP_{v,1}, PCGP_{v,2})$ and the angle between their normal vectors $n_{k(v,1)}$ and $n_{k(v,2)}$ must be below a set threshold $\beta = 180 \pm 2°$ (cf. Fig. 3 e). In addition, both faces are checked for an overlapping to ensure a fit to the parallel kinematics and the geometry of the gripper. For this verification, the auxiliary point $AP_{v,1}$ is moved in negative direction of its normal vector $-n_{k(v,1)}$ with the length $|d(AP_{v,1}, AP_{v,2})|$. After displacing the auxiliary point to the new point $AP_{v,1,\text{shifted}}$ the distance $|d(AP_{v,1,\text{shifted}}, AP_{v,2})|$ is computed. If the calculated value is below a set threshold ϵ, an overlapping is confirmed, otherwise the pairing is dismissed.

IV. Evaluation

In the following, we present the results of our developed BLM and the grasping point determination for a test case. We used 10 different objects from the Fraunhofer IPA Bin-Picking database [10] which is an extension of the Siléane dataset [2] (cf. Fig. 4). The resulting database displays a wide spectrum of different shaped objects. As the gripper parameters we used the specifications of a Schunk Co-act EGP-C 40 which has a max opening of 50 mm in our configuration. Additional default values are $d_{shift} = 0.01$ mm and $\epsilon = 5.0$ mm which proved to be appropriate values for our test objects. Some of the test objects where scaled to fit into the wrench space of the gripper.

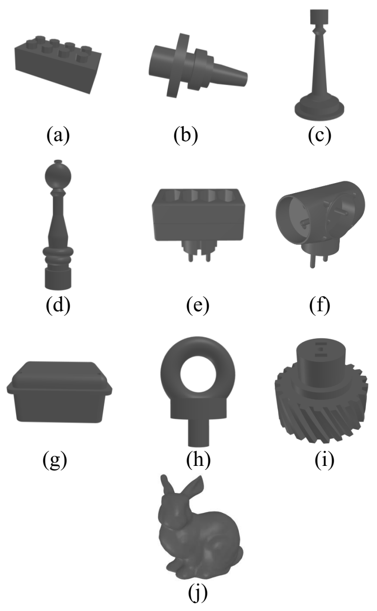

Fig. 4. : CAD models of the 10 test objects: a) SileaneBrick b) IPAGearShaft c) SileaneCandleStick d) SileanePepper e) SileaneTLess22 f) SileaneTLess20 g) SileaneTLess29 h) IPARingScrew i) SileaneGear j) SileaneBunny.

A. Results

For all objects except the SileaneBunny (cf. Section IV-B) we performed the BLM with suitable parameters for α and ϵ and calculated the number of grasping positions for the original STL file and then for the remeshed STL file. The results for all objects are shown in Table I.

A sizeable number of convex grasps, often more than a few hundred, could be determined for nearly all objects only using our grasping point determination algorithm. For all objects the number of grasping positions was significantly enlarged after using the BLM. The number of grasps in all cases increased by at least 38 % and in most cases many times more. Not only was the solution size significantly enlarged but also the distribution (cf. Fig. 5).In two cases (SileaneCandleStick and SileaneTLess29), no feasible grasping positions could be computed with the original STL file and through the BLM a number of positions could be computed resulting in a sizeable solution set. The low number of positions for the SileaneBunny results from its overall large dimensions compared to the Schunk gripper.

B. Discussion

For all test objects our algorithm was successful in finding grasping positions, however a few results need a more detailed discussion. Up until now, the parameters and are set manually for the BLM and identifying their optimal values can sometimes be time consuming (cf. Table I). The best practice right now is to start at values for a coarse mesh ($\alpha = 2$ mm and $\gamma = 95°$) which results in a relatively thick boundary layer and few surfaces and then gradually change the values until a suitable combination is found. The BLM performs good for basic objects and generally for meshes with sharp edges because the surfaces can be identified well. Objects with smooth edge transitions and in general complex curved freeform surfaces are more difficult or in some cases, e.g., SileaneBunny not possible to remesh because the edge detection is much more challenging. Depending on the number of faces in the mesh and calculated convex points, the runtime

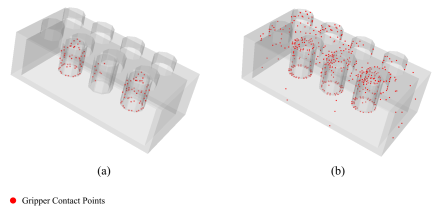

Fig. 5. Contact points (red) for the grasping point determination before (a) and after (b) using the BLM on the SileaneBrick.

TABLE I
PARAMETERS AND RESULTS FOR THE GRASPING POINT DETERMINATION WITH AND WITHOUT THE BLM.

Object Name	BLM	α [mm]	$\gamma_{i,j}$ [°]	Triangles pre-BLM	Triangles post-BLM	# Positions pre-BLM	# Positions post-BLM
SileaneBrick	Yes	0.1	130	532	1636	709	984
IPAGearShaft	Yes	0.5	95	4260	9468	145	16543
SileaneCandleStick	Yes	0.01	1206	2686	120	0	2116
SileanePepper	Yes	0.001	135	3738	4878	57	4488
SileaneTLess22	Yes	0.1	135	4692	9244	3260	5096
SileaneTLess20	Yes	0.01	95	21124	24348	4787	20031
SileaneTLess29	Yes	0.01	100	1152	2608	0	1876
IPARingScrew	Yes	0.001	135	4984	5752	1776	8932
SileaneGear	Yes	0.1	100	4404	5380	25057	46596
SileaneBunny	No	-	-	7000	-	3	-

of the algorithm ranged from a few seconds up to some hours. The most time-consuming part was the search for antipodal points from the set of concave points because every point in the solution set is cross checked for a potential fit. For our test objects this step required at least 50 % of the total time and offers the most potential for optimization. Resulting from the BLM, the size of the CAD files was enlarged between 1.5× and 10× the original size. The biggest file from our test set was the SileaneTLess20 object which had a size of 4.8 Mb.

V. CONCLUSION AND FUTURE WORK

In this work, we presented an approach to determine antipodal and convex grasping positions for parallel jaw grippers with the use of convex hulls and the developed BLM. Based on tessellated CAD models, a triangle boundary layer is inserted into every surface in the mesh and afterwards convex grasping points are calculated. Evidently from Fig.5, not every resulting grasping position is suitable for a gripper because at this point no global collision detection is performed which is planned for the next development stage. While we showed the validity of our proposed grasping point determination algorithm and the benefits of the BLM through our evaluation a friction analysis and component stress calculation will be implemented to allow a detailed classification of the calculated grips and ensure a successful grasp. To reduce the number of grasps, a clustering algorithm will be tested and process parallelization will be implemented to reduce the runtime of the algorithm.

REFERENCES

[1] Pascal Becker, Etienne Henger, Arne Roennau, and Ruediger Dillmann. Flexible object handling in additive manufacturing with service robotics. *IEEE International Conference on Industrial Engineering and Applications*, pages 121–128, 2019.

[2] Romain Bregier, Frederic Devernay, Laetitia Leyrit, and James L. Crowley. Symmetry aware evaluation of 3d object detection and pose estimation in scenes of many parts in bulk. *IEEE International Conference on Computer Vision Workshop*, pages 2209–2218, 2017.

[3] Shehan Caldera, Alexander Rassau, and Douglas Chai. Review of deep learning methods in robotic grasp detection. *Multimodal Technologies and Interaction*, 2018.

[4] Michael Dawson-Haggerty. Trimesh https://trimsh.org/. *version 3.9.10*, 2021-04-01.

[5] João Pedro Carvalho de Souza, Luís F. Rocha, Paulo Moura Oliveira, A. Paulo Moreira, and José Boaventura-Cunha. Robotic grasping: from wrench space heuristics to deep learning policies. *Robotics and Computer-Integrated Manufacturing*, (71), 2021.

[6] Du Guoguang, Kai Wang, Shiguo Lian, and Kaiyong Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review. *Artificial Intelligence Review*, (54):1677–1734, 2021.

[7] Kensuke Harada, Tokuo Tsuji, Kazuyuki Nagata, Natsuki Yamanobe, Kenichi Maruyama, Akira Nakamura, and Yoshihiro Kawai. Grasp planning for parallel grippers with flexibility on its grasping surface. *IEEE International Conference on Robotics and Biomimetics*, pages 1540–1546, 2011.

[8] Stefan Hesse. *Robotergreifer: Funktion, Gestaltung und Anwendung industrieller Greiftechnik*. Hanser, München, 2004.

[9] Mohammadali Honarpardaz, Martin Meier, and Robert Haschke. Fast grasp tool design: From force to form closure. In *IEEE Conference on Automation Science and Engineering*, volume 13, pages 782–788. 2017.

[10] Kilian Kleeberger, Christian Landgraf, and Marco Huber. Large-scale 6d object pose estimation dataset for industrial bin-picking. *IEEE International Conference on Intelligent Robots and Systems*, 2019.

[11] Kilian Kleeberger, Florian Roth, Richard Bormann, and Marco Huber. Automatic grasp pose generation for parallel jaw grippers. *International Conference on Intelligent Autonomous Systems*, 2021.

[12] Mats G. Larson and Fredrik Bengzon. *The finite element method: theory, implementation, and applications*, volume 10 of *Texts in computational science and engineering*. Springer, Berlin and Heidelberg, 2013.

[13] Beatriz León, Stefan Ulbrich, Rosen Diankov, Gustavo Puche, Markus Przybylski, Antonio Morales, Tamim Asfour, Sami Moisio, Jeannette Bohg, James Kuffner, and Rüdiger Dillmann. Opengrasp: A toolkit for robot grasping simulation. *International Conference on Simulation, Modling and Programming for Autonomous Robots*, pages 109–120, 2010.

[14] Shuo Liu and Stefano Carpin. A fast algorithm for grasp quality evaluation using the object wrench space. *IEEE Conference on Automation Science and Engineering*, pages 558–563, 2015.

[15] Y.-H. Liu, M.-L. Lam, and D. Ding. A complete and efficient algorithm for searching 3-d form-closure grasps in the discrete domain. *IEEE Transaction on Robotics*, 20:805–816, 2004.

[16] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, Vol. 4, No. 26, 2019.

[17] A. T. Miller and P. K. Allen. Graspit! *IEEE Robotics & Automation Magazine*, 11:110–122, 2004.

[18] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. *IEEE International Conference on Robotics and Automation*, pages 1824–1829, 2003.

[19] Aleksandr Ovcharenko, Kedar Chitale, Onkar Sahni, Kenneth E. Jansen, and Mark S. Shephard. Parallel adaptive boundary layer meshing for cfd analysis. *International Meshing Roundtable*, 21:437–455, 2013.

[20] Klaus Schittkowski. Solving constrained nonlinear least squares problems by a general purpose sqp-method. *Trends in Mathematical Optimization*, pages 295–309, 1988.

[21] G. Smith, E. Lee, K. Goldberg, K. Bohringer, and J. Craig. Computing parallel-jaw grips. *IEEE International Conference on Robotics and Automation*, pages 1897–1903, 1999.

[22] Nikolaus Vahrenkamp, Eduard Koch, Mirko Waechter, and Tamim Asfour. Planning high-quality grasps using mean curvature object skeletons. *IEEE Robotics And Automation Letters*, 2017.