Computing Gripping Points in 2D Parallel Surfaces Via Polygon Clipping

Ludwig Vogt, Yannick Zimmermann and Johannes Schilp

Abstract

To generate suitable grasping positions between tessellated handling objects and specific planar grippers, we propose a 2D analytical approach which uses a polygon clipping algorithm to generate detailed information about the intersection between both objects. With the generated knowledge about the intersection we check whether its shape fits to the set criteria of the operator and represents a valid grasping position. Before the polygon clipping algorithm is applied, a preprocessing step is performed, where appropriate surfaces from the handling object and the gripper are extracted. After rotating all surfaces into a common plane, potential clipping positions are detected and the clipping is performed to get an accurate intersection detection. The validation shows comparable running times to a OBBTree algorithm (0.1 ms per grasping position) while increasing the stability of the results from 30 to 100% for the evaluated test objects.

Keywords

Handling · Polygon clipping · Gripping point determination

L. Vogt (✉) · Y. Zimmermann · J. Schilp
Chair of Digital Manufacturing, Faculty of Applied Computer Science, Augsburg University, Eichleitnerstr. 30, 86159 Augsburg, Germany
e-mail: Ludwig.Vogt@informatik.uni-augsburg.de

J. Schilp
Fraunhofer Research Institution for Casting, Composite and Processing Technology–IGCV, Provinostr. 52, 86153 Augsburg, Germany

T. Schüppstuhl et al. (eds.), *Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2021*,
https://doi.org/10.1007/978-3-030-74032-0_9

1 Introduction

With a rising degree in automated process chains and a continuously demand of smaller lot sizes the requirements for automated product handling are increasing. Additionally, additive manufacturing enables the production of topology optimized parts which can have a very complex and delicate shape or structure and are therefore difficult to handle [1].

The automated handling process and the determination of gripping points has been a popular research field in recent times [2, 3]. For the determination of gripping points two main approaches exist: an analytical approach, taking mechanical and physical properties of the object into account and an empirical approach, trying to replicate the movement of the human hand [2]. While the empirical grasping approach uses neural networks [4] or fuzzy logic [5], the analytical grasping approach includes mathematical models to determine the contact between the gripper and the object [6]. A central part of both strategies is to find suitable grasping positions on the object. To form stable grasps, it is aspired to generate a maximum overlap between the gripping surface and the object. For some handling tasks, only partial overlapping is realizable without changing the grippers. To control this, our main target is to generate a complete grasp set which contains all planar grasps with information about the overlapping between gripper and handling object. Therefore, it is necessary to accurately determine the shape of the contact area. While the empirical approaches show a fast and computational efficient solution they need extensive and high quality training data to generate accurate models for each gripper [7]. Trained models are also gripper specific and difficult to use for other gripper shapes. In analytical methods a collision detection can be used to check for a correct gripper alignment and intersection area, but often contacts are calculated via bounding boxes with primitive shapes [8] and the contact shape is approximated or neglected.

To generate information about the intersection between the object and the gripper a 3D collision detection based on a Oriented Bounding Box Tree (OBBTree) from the vtk library was first used [9]. Although the collision detection was robust and performing well for boundary contacts, the intersection determination was not suitable for our application. Tests showed a good approximation for basic intersection shapes e.g. rectangle, triangle and circle but unsatisfying or no output at all for concave shapes containing holes and crooked intersection planes (cf. Fig. 1).

For this reason, a 2D analytical approach where we use a polygon clipping algorithm to determine area gripping points is proposed.

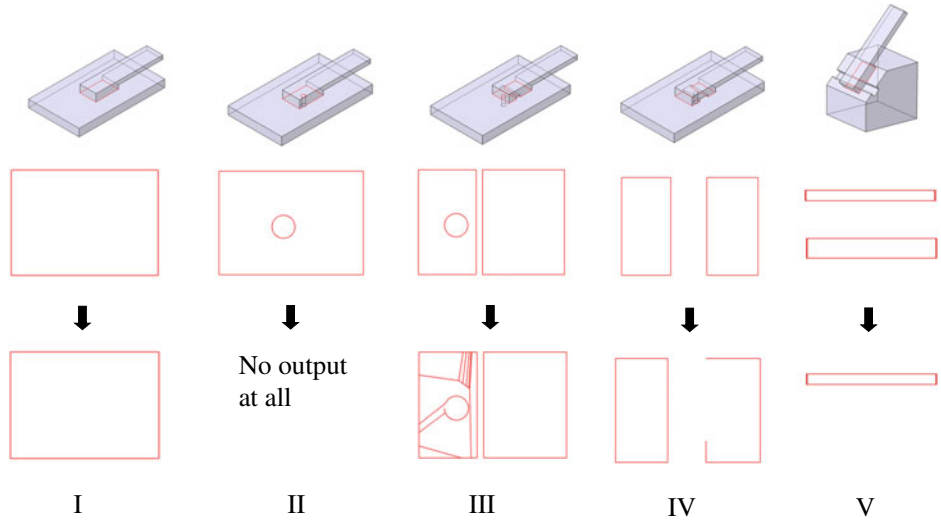


Fig. 1 Results from the intersection detection (red) with an OBBTree algorithm. Second row shows the desired intersection shape, last row shows the computed output. Intersecting cases I-IV lay in the XY plane and case V in a crooked plane

2 State of the Art

Before giving an overview about the various methods to determination gripping points for planar grippers and their intersections, we give a short introduction about the basic concept of polygon clipping and detail the most used algorithms.

2.1 Polygon Clipping

Originally polygon clipping algorithms were developed for basic operations in creating graphic output [10]. Polygon clipping is the calculation of the intersection of two given polygons: a subject polygon and a clipping polygon (cf. Fig. 2] (Rappoport 11]. Because the intersection output from the 3D collision is unsuitable for our application (cf. Chap. 1) and Triangle-Triangle intersection algorithms are either case dependent [12] or perform poorly for bigger applications [13], we develop an alternative algorithm. The usage of polygon clipping algorithms [14] enables a stable and accurate intersection detection between two polygons. In the Literature several algorithms have been proposed to solve this problem, but in the following three of the more prominent ones are described.

Sutherland and Hodgman [15] propose an algorithm which is able to clip a convex clipping polygon and a concave subject polygon against each other but not two concave polygons. The Weiler-Atherton algorithm [14] is able to clip two non-self-intersecting

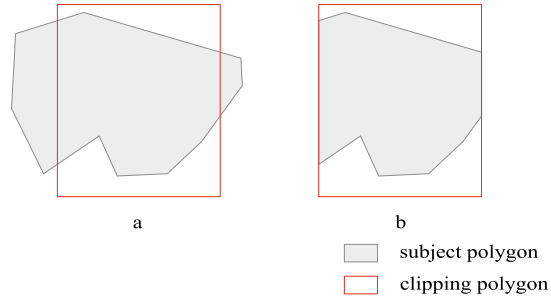


Fig. 2 Initial situation for a polygon clipping algorithm (**a**) and the resulting solution (**b**)

concave polygons against each other. They are even allowed to contain holes. The algorithm is based on classifying each intersection between both polygons and creating the total intersection from it with a runtime complexity of O(n2). The Vatti algorithm [16] is also able to clip two concave polygons against each other. After the determination of the left and right bound of each polygon the intersection is computed via the use of scanbeams with a runtime complexity better than O(n2).

2.2 Gripping Point Determination and Intersection Detection

Because the developed approach is restricted to force closure with area contacts, this paragraph focusses on the relevant literature in this subject area. Some Methods 17, 18] focus on identifying contact points which fulfill the wrench equilibrium without deriving shape information for the intersection. Li et al. [19] calculated a grasp synthesis by shape matching the human hand with the form of the handling object based on a comparison of self-created feature sets. While this approach is suitable to find positions for a full overlapping, it is not applicable for partial overlappings. A triangular clustering is proposed in Harada et al. [20] to determine the contact points of a two finger parallel gripper with soft fingertips. Therefore, neighbor triangles of a surface are clustered via a comparison of their normal vectors and saved to a new rectangular grasping plane. In the next step the gripper plane is matched with the grasping plane on the object but no intersection between the two shapes is derived. Bonilla et al. [21] used bounding boxes to decompose the handling object and find suitable grasps with the use of basic geometries. Their testing showed a robust and flexible method with success rates of at least 77,61% but no shape information for the contact is created and used. Lin et al. [22] decompose objects, represented via a an RGB image, into several ellipses and build a grasping rectangle for each decomposition. The missing depth data prevents a calculation of the intersection area. Spenrath and Pott [23] use a heuristic search to select grasping positions in bin picking applications with the use of predefined contact regions. Trained neural networks [3, 4] show a good performance with high success rates (up to 95%) but need a specific training set for each gripper and do not determine specific shapes for the intersections between the handling object and gripper. Therefore, our main goal is to get a stable and computational

efficient intersection detection for grippers and handling objects even for complex intersection shapes.

3 Gripping Point Determination with an Intersection Detection

The algorithm is built in a Python 3 environment and can be divided in 6 sequences (cf. Fig. 3). To find a stable, planar and accurate matching between the gripper surface and the handling object, the algorithm takes CAD data from both objects as an input and calculates valid gripping points. This means that the intersection surface satisfies the set criteria (cf. Chap. 4.2). Note that the results of the algorithm ignore the reachability and the 3D collision between the gripper and the object. These steps will be implemented in future work.

3.1 Preprocessing

First, the CAD-data of the handling object is imported in a.stl-format and all surface triangles are stored in a $(n \times 9)$ matrix, where n denotes the number of triangles in the mesh. Afterwards a similar approach as in Harada et al. [20] is used to cluster the triangles in 2D surfaces, with the difference that all triangles lay in the spanned surface and don't cut the newly generated surface. From the surfaces in this data set, pairs of parallel surfaces are calculated and stored via a comparison of their plane normal vectors. To take rounding errors into account, or if it is desired to allow a small angle between the surfaces, it is also possible to set a specific threshold. Additional restriction for this step is the geometry of the gripper (max. & min. opening of the gripper). Finally, the contour of the gripper surface is derived from the gripper model (.stl-format). The identification of the gripper surface is done manually at the moment.

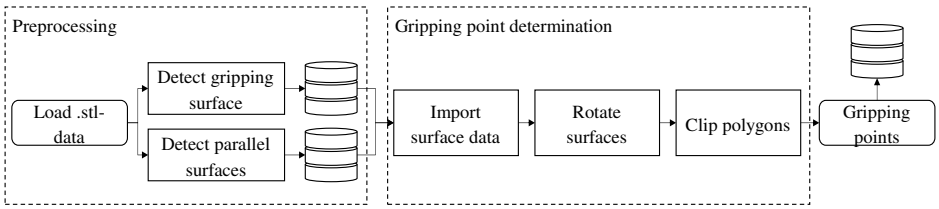


Fig. 3 Workflow of the program to determine gripping points with a polygon clipping algorithm

3.2 Gripping Point Determination Via Polygon Clipping

These two generated data sets are handed over to the gripping point determination algorithm. First, the surfaces are imported and afterwards their contours are extracted. For this, the 3 connection vectors of all triangles n are calculated and saved. For each surface, an empty set is created and afterwards an iteration over the created vector tuples is done. If the vector is already existent in the data set, it is deleted from the set, otherwise it is included. After this step the remaining vectors in the contour are strung together and represent the contour of the surface. Next, is a rotation of all surfaces into the XY-plane. For the following steps the algorithm uses *default* variables which can be set manually by the operator and are further explained in Chap. 4.2. After the rotation, an iteration over the contour area is done and then the clipping is performed (cf. Fig. 4).

To generate different grasping positions, distributed on the object surface, the gripper surface is shifted along the grid from Fig. 4. At every (x, y) combination a rotation of the gripper surface with a *default_roation* value is done. Afterwards the intersection between both polygons is determined with a polygon clipping algorithm and if the following two criteria are satisfied, a valid gripping point is detected:

- The overlapping area is greater or equal than the set *default_roation*.
- At the second object surface, a matching position which also satisfies the *default*_overlapping is found.

The implemented extension for the Weiler-Atherton algorithm enables the clipping of polygons against each other with an arbitrary shape. As long as they are not self-intersecting, the Vatti algorithm can be used directly from a clipper library. In the last step the *gripping_point* data set is rotated back to its original orientation. The whole algorithm is summarized in the following:

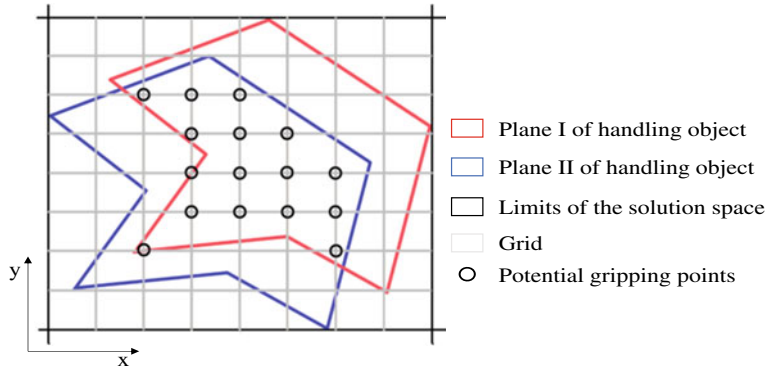


Fig. 4 Determination of potential gripping points (circles) from the two surfaces on the handling object (red, blue) with a superimposed grid in the XY-plane

Gripping point determination via a polygon clipping algorithm

```
## Contour detection
for i in 1:n do
append {p[i][1]-p[i][0], p[i][2]-p[i][1], p[i][2]-p[i][0]} to vectors
contour = []
for i in 1:n do
  for j in 1:n do
    if vectors[i[j]] in contour then
      delete vectors[i[j]] in contour
    else then
      append vectors[i[j]] to contour
sort contour so that vectors are strung together
## Surface rotation
angle = calculate angle between plane normal vector & XY-plane
rotate contour around origin with angle
## Polygon clipping
max_x, max_y = maximum x & y values in contour
min_x, min_y = minimum x & y values in contour
for i in max_x:min_x with step size default_distance do
  for j in max_y:min_y with step size default_distance do
    for k in 1:360 with step size default_rotation do
      set position of gripping_surface to x = i, y = j
      rotate gripping_surface with k°
      if polygon clipping of contour and gripping_surface ≥
       default_overlapping & counterpart at second object surface
       also, then
        append [x = i, y = j, rotation = k] to gripping_points
rotate back gripping_points around the origin with −angle
end
```

p denotes the points of all triangles in a surface, *vectors* all the calculated connection vectors and *contour* all the derived vectors of the surface outline. Also, *gripping_surface* denotes all the contour vectors of the gripping surface and *gripping_points* denotes the set of calculated grasping positions which satisfy the set criteria.

4 Evaluation

4.1 Test Objects and Evaluation Criteria

To evaluate the algorithm, three self-created handling objects (cf. Fig. 5) are combined with a rectangular gripping surface. The surface of the gripper is 20 mm × 30 mm, while

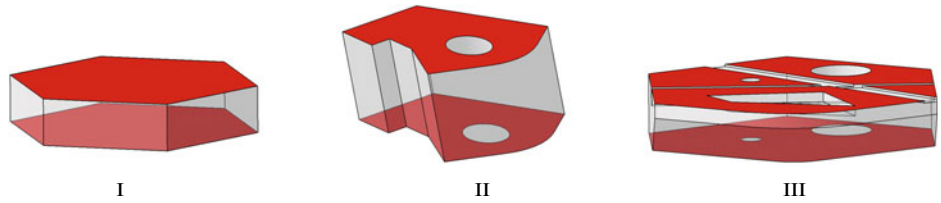


Fig. 5 Representation of the three (I, II, III) handling objects for the test scenario and the derived parallel planes (red) for the gripping point determination

the size of the handling objects is noticeably bigger to enable suitable grasps. The boundary of the 2D surfaces on the handling objects have varying complexity and contain different shapes and holes. To enable a comparison between our approach and existing strategies, we implemented the Vatti algorithm and a custom extension of the Weiler-Atherton algorithm and compared their performance against the Triangle-Triangle intersection detection [13] and the 3D intersection detection from the OBBTree vtk method. As mentioned in Chap. 1, the intersection detection from OBBTree is giving unsatisfying results but is included to act as a reference for computational efficiency as it represents a state-of-the-art 3D collision detection.

We computed the runtime of all algorithms for each test case 30 times and calculated the average computation time per position. The runtime computation is restricted to the core process because it is assumed that the preprocessing for every algorithm is more or less the same. To measure the stability of the methods we check all intersections if they are correct or not, so a score of 50% equals 5 correct outputs for 10 examples.

4.2 Restrictions

First tests of the algorithm without restrictions showed a solution set with an infinite number of grasping positions. Therefore, it is possible to set the following constraints via variables to reduce computational resources and the size of the solution set:

- Distance between two gripping points [mm]: In a radius with this value around a detected gripping point no other gripping point is allowed. Otherwise the algorithm would generate gripping points with an offset which is close to zero.
- Number of rotations at one gripping point [−]: Because a generated gripping point can be the center of many grips if the gripper is rotated around that point, the number of rotations are restricted for each position.
- Minimal overlapping of the gripper surface [%]: A successful grip is also possible if less than 100% of the gripper surface are in contact with the handling object. To realize that, a variable was introduced to set the minimal overlapping between the two surfaces on each gripping side.
- Maximum number of gripping points [−]: For each set of surface pairs a threshold for the maximum number of gripping points can be set.

Table 1 Set default_ parameters for the gripping point determination in the test cases

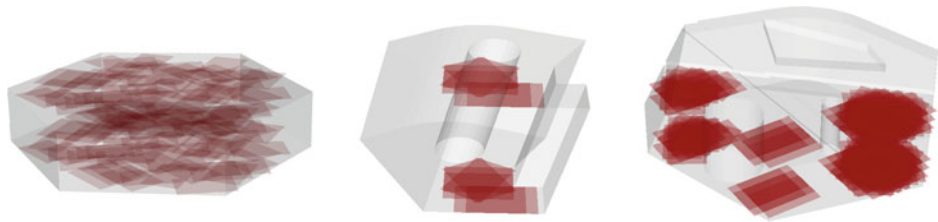
	Default_distance [mm]	Default_rotations [−]	Default_overlapping [%]	Default_#points [−]
Value	5	6 every 60°	100	None

Table 1 shows the *default*_parameters used in the test cases.

5 Results

The results of the proposed algorithm are visualized in Fig. 6 for the three handling objects. Table 2 contains the runtime evaluation for the four algorithms. For sake of a clear visualization, only a fraction of the full solution set of handling object I, containing 2576 grasping positions with the given variables, is visualized.

Comparing the four algorithms, the Vatti- and OBBTree algorithm delivered the best performance followed by the Triangle-Triangle and the customized Weiler-Atherton algorithm. Considering the average clipping time per position, the results show a dependency to the intersection complexity, more specifically the number of intersecting



I II III

Fig. 6 Resulting grasping positions with the Vatti Algorithm on the three handling objects with the marked intersections (red) between the gripper and handling object

Table 2 Performance of the intersection detection with the Weiler-Atherton, Vatti, OBBTree and Triangle-Triangle algorithm. The proposed methods are marked bold

Handling object	Potential grasping positions [−]	Suitable grasping positions [−]	Avg. clipping time per position [ms] (stability)			
			Weiler-Atherton	Vatti	OBBTree	TriangleTriangle
Object I	4068	2576	1.02 (100%)	0.023 (100%)	0.0354 (30%)	0.742 (100%)
Object II	786	12	11.26 (100%)	0.063 (100%)	0.0364 (30%)	1.2961 (100%)
Object III	2484	130	30.83 (100%)	0.11 (100%)	0.0362 (31%)	2.78 (100%)

triangles for the two clipping algorithms and the Triangle-Triangle algorithm. The OBBTree algorithm does not indicate that dependency. These results affirm the suspected performances from Chap. 2. Due to the self-extension of the Weiler-Atherton algorithm, it´s runtime complexity is bigger than (On4), in square the original runtime complexity and in the Triangle-Triangle algorithm 6 inequalities have to be solved, which can be a time-consuming process. Even for object I where a proportionately small number of triangles intersect, the Vatti algorithm and OBBTree algorithm show by far the best performance with computing times smaller than 4/100 ms while the other algorithms took approximately 1 ms to compute one position. While the Vatti algorithm and OBBTree algorithm show comparable running times, their stability shows a clear difference because the OBBTree algorithm correctly detected just 30% of the intersections. The other 70% of the classifications were false and looked comparable to the output in Fig. 1.

6 Conclusion

As shown, our algorithm represents an alternative way to accurately detect area grasping positions and their intersection for tessellated based handling objects and planar grippers. With the graphical representation of the grasping positions for three test cases we showed the validity and robustness of our approach even for complex intersection shapes. A performance comparison between existing approaches from the literature showed the suitability of our algorithm concerning computational complexity as it showed a better performance than a standard Triangle-Triangle intersection determination algorithm. Although the results showed a runtime dependency to the number of intersecting triangles for polygon clipping algorithms, the runtime complexity of the Vatti clipping was still comparable to the OBBTree algorithm and resulting in a higher stability score.

While the algorithm was successfully tested, a few parts were identified for further improvement. The algorithm uses settable *default_* variables which are not optimized because the solution size was sufficient for all test cases. To optimize these parameters, further test cases have to be evaluated. Another point to expand the flexibility of the gripping point determination is to extend the algorithm for grippers with different gripping surfaces at each finger and implement parallel computing. At last, as part of a greater gripping point determination project the algorithm will be implemented into a thorough gripping point determination routine. There, an analysis of the reachability and security of the grips will be done.

References

1. Bonilla, M., Resasco, D., Gabiccini, M., et al.: Grasp planning with soft hands using Bounding Box object decomposition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 518–523. IEEE (2015)
2. Gottschalk, S., Lin, M.C., Manocha, D.: OBBTree. In: Fujii, J. (ed.) Proceedings of the 23rd annual conference on Computer graphics and interactive techniques—SIGGRAPH ‘96, pp. 171–180. ACM Press, New York, USA (1996)

3. Guigue, P., Devillers, O.: Fast and robust Triangle-Triangle overlap test using orientation predicates. J. Graph. Tools **8**, pp 25–32 (2003)
4. Harada, K., Tsuji, T., Nagata, K., et al.: Grasp planning for parallel grippers with flexibility on its grasping surface. In: 2011 IEEE International Conference on Robotics and Biomimetics, pp. 1540–1546. IEEE (2011)
5. Hsiao, K., Chitta, S., Ciocarlie, M., et al.: Contact-reactive grasping of objects with partial shape information. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1228–1235. IEEE (2010)
6. Jiang, S., Zhao, X., Cai, Z., et al.: Single-grasp detection based on rotational region CNN. In: Ju, Z., Yang, L., Yang, C., et al. (eds.) Advances in Computational Intelligence Systems, vol. 1043, pp. 131–141. Springer International Publishing, Cham (2020)
7. Kim, I., Inooka, H.: Determination of grasp forces for robot hands based on human capabilities. Control Eng. Pract. **2**, 415–420 (1994). https://doi.org/10.1016/0967-0661(94)90778-1
8. Lachmayer, R., Lippert, R.B., Kaierle, S.: Additive Serienfertigung. Springer, Berlin (2018)
9. Li, Y., Fu, J.L., Pollard, N.S.: Data-driven grasp synthesis using shape matching and task-based pruning. IEEE Trans. Vis. Comput. Graph. **13**, 732–747 (2007). https://doi.org/10.1109/TVCG.2007.1033
10. Liang, Y.-D., Barsky, B.A.: An analysis and algorithm for polygon clipping. Communications of the ACM. Commun. ACM **26**, 868–877 (1983). https://doi.org/10.1145/182.358439
11. Lin, H., Zhang, T., Chen, Z., et al.: Adaptive fuzzy gaussian mixture models for shape approximation in robot grasping. Int. J. Fuzzy Syst. **21**, 1026–1037 (2019). https://doi.org/10.1007/s40815-018-00604-8
12. Mahler, J., Liang, J., Niyaz, S., et al.: Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics (2017)
13. Nguyen, V.-D.: Constructing force-closure grasps. In: Proceedings, 1986 IEEE International Conference on Robotics and Automation. Institute of Electrical and Electronics Engineers, pp. 1368–1373 (1986)
14. Rakesh, V., Sharma, U., Murugan, S., et al.: Optimizing force closure grasps on 3D objects using a modified genetic algorithm. Soft Comput. **22**, 759–772 (2018). https://doi.org/10.1007/s00500-016-2377-6
15. Rappoport, A.: An efficient algorithm for line and polygon clipping. Vis. Comput. **7**, 19–28 (1991). https://doi.org/10.1007/BF01994114
16. Roa, M.A., Suárez, R.: Grasp quality measures: review and performance. Auton Robot. **38**, 65–88 (2015). https://doi.org/10.1007/s10514-014-9402-3
17. Sabharwal C.L., Leopold, J.L.: A Trianlge-Triangle intersection algorithm. In: Computer Science & Information Technology (CS & IT). Academy & Industry Research Collaboration Center (AIRCC), pp 27–35 (2015)
18. Sahbani, A., El-Khoury, S., Bidaud, P.: An overview of 3D object grasp synthesis algorithms. Robot. Auton. Syst. **60**, 326–336 (2012). https://doi.org/10.1016/j.robot.2011.07.016
19. Spenrath, F., Pott, A.: Gripping point determination for bin picking using heuristic search. Procedia CIRP **62**, 606–611 (2017). https://doi.org/10.1016/j.procir.2016.06.015
20. Su, K.-H., Huang, S.-J., Yang, C.-Y.: Development of robotic grasping gripper based on smart fuzzy controller. Int. J. Fuzzy Syst. **17**, 595–608 (2015). https://doi.org/10.1007/s40815-015-0042-3
21. Sutherland, I.E., Hodgman, G.W.: Reentrant polygon clipping. Commun. ACM **17**, 32–42 (1974). https://doi.org/10.1145/360767.360802
22. Vatti, B.R.: A generic solution to polygon clipping. Commun. ACM **35**, 56–63 (1992). https://doi.org/10.1145/129902.129906
23. Weiler, K., Atherton, P.: Hidden surface removal using polygon area sorting. In: Unknown (ed.) Proceedings of the 4th annual conference on Computer graphics and interactive techniques—SIGGRAPH '77, pp. 214–222. ACM Press, New York, USA (1977)