Journal of *Marine Science and Engineering*

MDPI

Article

A Novel Vision-Based Outline Extraction Method for Hull Components in Shipbuilding

Hang Yu [1,2], Yixi Zhao [1], Chongben Ni [3,4,*], Jinhong Ding [3,4,*], Tao Zhang [4], Ran Zhang [4] and Xintian Jiang [4]

[1] School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; yuhang6464@sjtu.edu.cn (H.Y.)
[2] Shipbuilding Technology Research Institute, Shanghai 200032, China
[3] State Key Laboratory of Ocean Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[4] School of Naval Architecture, Ocean & Civil Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
* Correspondence: nichongben@sjtu.edu.cn (C.N.); ahaha@sjtu.edu.cn (J.D.)

Abstract: The diverse nature of hull components in shipbuilding has created a demand for intelligent robots capable of performing various tasks without pre-teaching or template-based programming. Visual perception of a target's outline is crucial for path planning in robotic edge grinding and other processes. Providing the target's outline from point cloud or image data is essential for autonomous programming, requiring a high-performance algorithm to handle large amounts of data in real-time construction while preserving geometric details. The high computational cost of triangulation has hindered real-time industrial applications, prompting efforts to improve efficiency. To address this, a new improvement called Directive Searching has been proposed to enhance search efficiency by directing the search towards the target triangle cell and avoiding redundant searches. Another improvement, Heritable Initial, reduces the search amount by inheriting the start position from the last search. Combining Directive Searching and Heritable Initial into a new method called DSHI has led to a significant efficiency advancement, with a calculation efficiency improvement of nearly 300–3000 times compared to the ordinary Bowyer–Watson method. In terms of outlines extraction, DSHI has improved the extraction efficiency by 4–16 times compared to the ordinary Bowyer–Watson methods, while ensuring stable outlines results, and has also increased the extraction efficiency by 2–4 times compared to PCL. The DSHI method is also applied to actual ship component edge-grinding equipment, and its effect meets the shipbuilding process requirements. It could be inferred that the new method has potential applications in shipbuilding and other industries, offering satisfying efficiency and robustness for tasks such as automatic edge grinding.

Keywords: intelligent manufacture; machine vision; vision-guided robot; Delaunay triangulation; outline extraction

Citation: Yu, H.; Zhao, Y.; Ni, C.; Ding, J.; Zhang, T.; Zhang, R.; Jiang, X. A Novel Vision-Based Outline Extraction Method for Hull Components in Shipbuilding. *J. Mar. Sci. Eng.* **2024**, *12*, 453. https://doi.org/10.3390/jmse12030453

Academic Editor: Burak Can Cerik

Received: 3 February 2024
Revised: 27 February 2024
Accepted: 29 February 2024
Published: 4 March 2024

1. Introduction

In the shipbuilding industry, most edge-grinding jobs, which is required for all marine components before painting, are still performed manually, costing considerable time and labor, as shown in Figure 1. Meanwhile, metal dust and noise generated by grinding is health-hazardous. This proposes the demand of autonomous grinding. However, components of marine structures are naturally of various shapes and sizes with huge amounts, but the batch size of each type is relatively small. Conventional industrial robots rely on manual teaching or programming, which has to be carried out repetitively for each different product type. The current reliance on manual intervention has significantly impeded the advancement of automation in the shipbuilding industry. In contrast, vision-guided robots that could be automatically reprogrammed on products' shape are free from repetitive manual teaching or model-based programming, which greatly improved the production efficiency

for small batches of multiple types of components. Hence, developing a vision-guided robot grinding system may be an optimized solution.

(**a**) (**b**)

Figure 1. (**a**) Manual edge grinding of ship components; (**b**) Ship components in various shapes and sizes.

There are several key techniques to realize the vision-guided robot system. Some of these techniques, like vision-based calibration, trajectories and programming, have been solved in previous studies. Outline extraction is the last piece to be filled. Vision data, like point clouds from laser scanning or images from a camera, are commonly used to generate workpieces' shapes. Conventionally, edge detection operators like Roberts, Sobel and Prewitt are employed in providing an outline of pixel images, and Graham scan produces a desirable outline for points in convex shapes. Meanwhile, outline extraction from points in concave shapes poses challenges to both academic research and industrial applications. The difficulty appears in two aspects. First, it is expected to produce results that meet human's visual perception for random-shaped components. In practice, the algorithm faces the dilemma of being flexible and stable. The algorithm should be adjustable to provide robust results in different detail levels and will not be over-sensitive, to avoid inevitable errors in points data. Second, the rapidity is crucial since point clouds are mostly in mega-amounts.

Through the research of actual intelligent edge-grinding equipment, the improvement of the recognition efficiency will significantly impact the production efficiency of the overall edge-grinding equipment. For some hull components that require double-sided edge grinding, the effect of improved recognition efficiency will be more significant. For the purpose of visual-guided robotic edge grind of hull components, the Delaunay triangulation-based method is studied here to generate acceptable outline shapes of points from randomly shaped hull components. This paper introduces recent attempts to improve the robustness and efficiency of the algorithm in Section 3. Section 4 verifies the algorithm with point cloud data from test pieces and discusses the potential of a vision-guided working mode in small-batch production.

2. Related Work

In many cases, robots are manually programmed. However, the evolution towards automated robot systems sustains different manufacturing processes. Leo and Selvaraj developed a vision-guided deburring system for rectangle workpieces [1]. Tellaeche and Arana achieved the CAD model-based robotic deburring [2]. Wenlong Li [3] solved the hand–eye calibration problem for grinding. Studies mentioned above are more emphasized on a robot's motion control, while the irregular shape of the workpiece is not mentioned.

For objects in complex shapes, Theodosios [4] proposed and discussed the skeletonization methods, which leads to the Medial axis transform (MAT). The MAT is implemented in mapping and construction because it provides a compact presentation of geometry. However, MAT is easily disturbed by perturbation. Meanwhile, a rare study was conducted from the view of the shipbuilding industry, which emphasizes efficiency and robustness. The gap between mathematical algorithms to the implementation of small-batch production is unfilled.

Research on extracting outlines from visual data poses challenges due to the complexity and variations of target objects. For objects in regular polygon shape, attempts of outline segment segmentation and regularization were performed through Ordered Hough Transform (OHT) by Widyaningrum [5], or through a Ransac-based method by Lucas and Van Tilburg [6]. For objects in complex shapes, alpha shape is well accepted as the outline shape in the visual perception of discrete data points. Edelsbrunner [7] introduced the creation of outlines from a selection of boundary points by the alpha shape algorithm. This leads attention towards the geometry-based outline extraction methods for planar shape, including Voronoi diagrams and Delaunay triangulation [8]. This topic also arises in satellite remote sensing imagery. Zollini and Dominici [9] contributed to the shoreline extraction from SAR and optical images.

The Delaunay triangulation is unique for any fixed set of points. It is studied thoroughly and applied in many fields, like meshing for finite element analysis, mapping for geographical information systems (GIS) [10], and surface reconstruction for reverse engineering [11]. The study focuses on algorithm generating Delaunay triangulation also draws attention. The methodology of previous work on Delaunay triangulation could be classified into several categories, namely incremental insertion algorithm [12], divide-and-conquer algorithm [13], gift wrapping algorithm [14], triangle expanding algorithm [15], sweep line algorithm [16] and convex-hull-based algorithm [17]. According to Su's comparison [18], the algorithms mentioned above have different advantages and disadvantages, suited for various demands in different fields. With the rapid development of computer hardware, algorithm work on parallel computation [19–24] has emerged in recent years. Among these, the workflow of incremental insertion is flexible and friendly to points appending or removing. Thus, it is applied in cases with a demand for frequent point editing. However, the incremental insertion method is associated with a significant computational cost, which is particularly challenging given the mega data involved in point clouds. Enhanced efficiency in this regard would be highly beneficial in expanding its application scope.

3. Methodology

The alpha shape is well known for its capability to preserve small shape details of a finite point set at a required level of detail [10]. The alpha shape is defined in terms of α, which effects the boundary's detail level. As α varies, the alpha shapes can change from the points themselves to the convex hull, as shown in Figure 2.

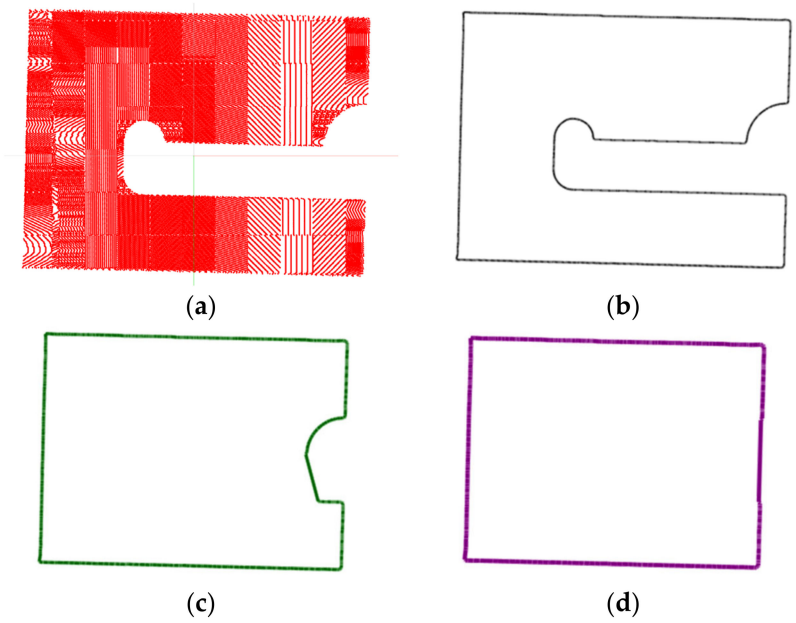


Figure 2. (**a**) Points of concave object; (**b**) Alpha shape of points ($1/\alpha$ = 100); (**c**) Alpha shape of points ($1/\alpha$ = 500); (**d**) Alpha shape of points ($1/\alpha$ = 1000).

Edelsbrunner proposed an empty circle check to determine the boundary of the alpha shape. As shown in Figure 3a, the edge connecting points 6 and 7 is a boundary edge because at least one empty circle at r = 1/α exists. Likewise, all boundary edges given by empty circle check form a complete alpha shape.

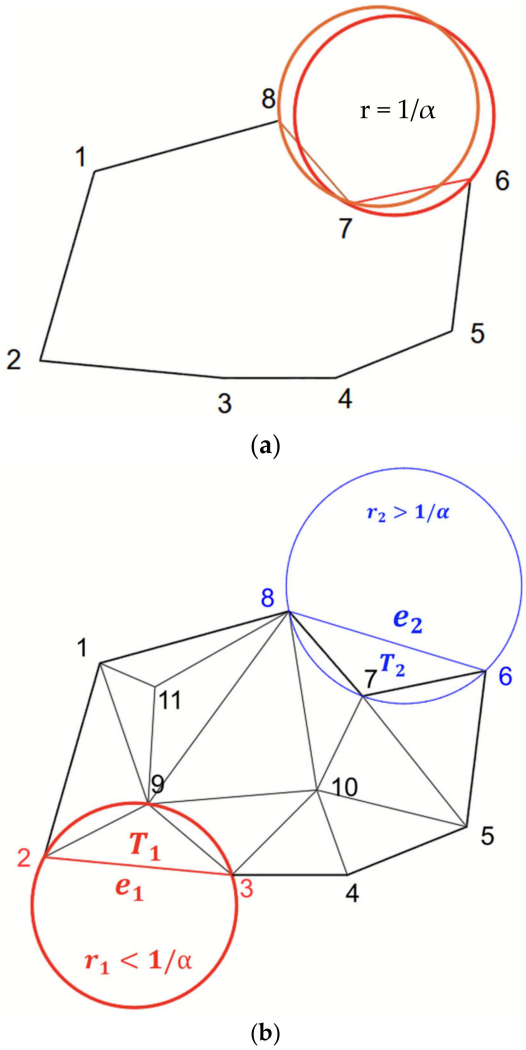


Figure 3. (**a**) Boundary edges from empty circle check; (**b**) Same result of boundary edges (in thick strokes) from circumcircle check.

The time cost of an empty circle check in Edelsbrunner's method is unpredictable. An alternative method is a circumcircle check based on Delaunay triangulation. Figure 3b shows that the circumcircle check provides the same result for boundary edges.

Thus, the general workflow of our proposed method consists of three major steps, as shown in Figure 4. First, a triangulation mesh of the point set is generated. Second, a BFS starts from the super triangles to perform the circumcircle check. Third, the free edges of all outer triangles are found and close the loop of the point set's outline. To provide a clear and coherent narrative, further details on each methodological step are provided in the following.

3.1. Quicker Insertion Algorithm

Given the point set S of the concave object, the outline extraction starts with a Delaunay triangulation mesh DT(S). The Bowyer–Watson incremental insertion is one of the classical mesh generation methods. Because the workflow of incremental insertion is friendly to data editing, it is still a good choice on many occasions.

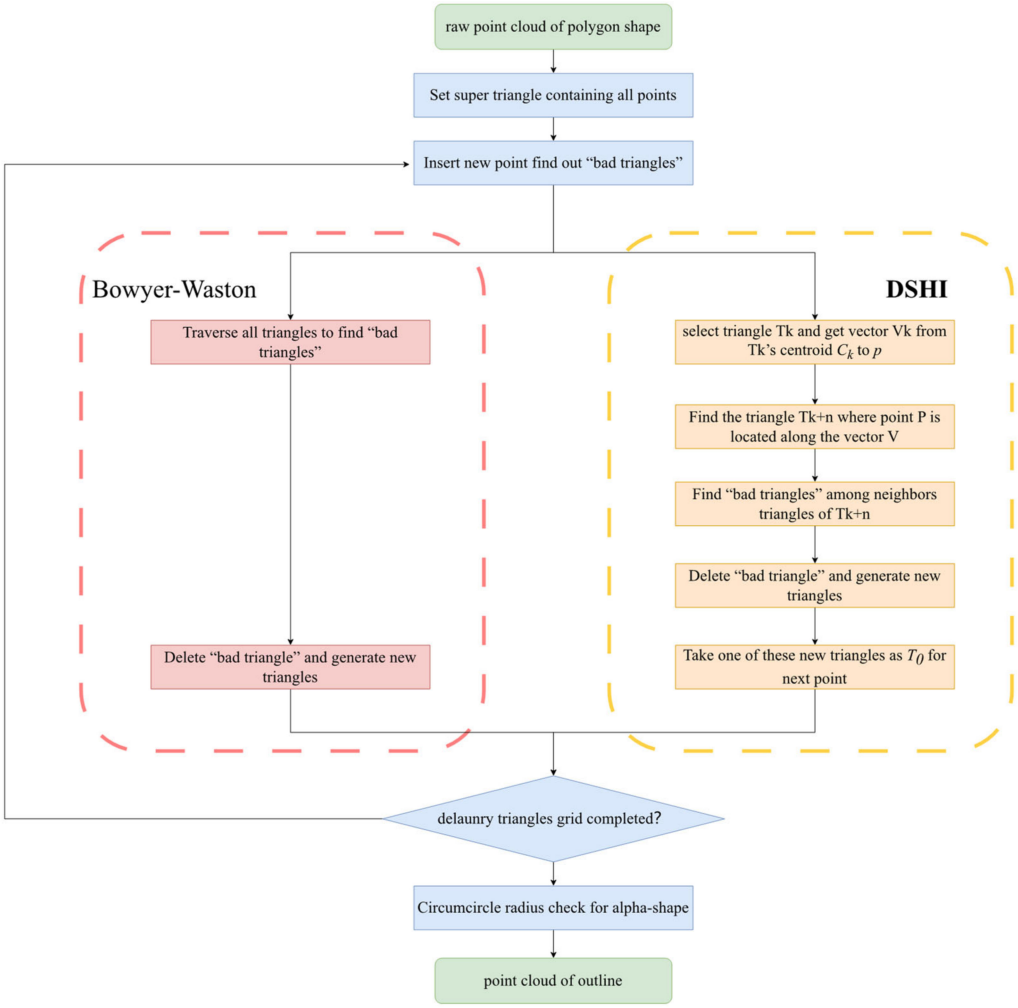


Figure 4. Workflow to extract the outline of the point cloud.

Generally, the Bowyer–Watson scheme starts from a large triangle containing all points within. This triangle is usually called a super triangle. Then, the incremental insertion repeats the following 2 steps until all points are inserted:

Step 1: Insert a new point p. Find out triangles that contain p within their circumcircle. These triangles are marked as "bad triangles".

Step 2: Delete these "bad triangles" and form a polygonal hole. And generate new triangles by connecting the edges of this polygonal hole to point p.

A straightforward implementation of Step 1 is to check all triangles in each iteration. But this method leads to 2 problems:

1. The round-off error is inevitable in practice. In some cases, the round-off error causes triangles' degeneration.
2. It is unnecessary to check all triangles. Since "bad triangles" account for a small fraction of all triangles in each iteration, most calculation expenses of the point-in-circumcircle check are worthless.

Thus, here we propose the first improvement of the incremental insertion algorithm. Step 1 is decomposed into 2 sub-steps:

Step 1.1: Find out the triangle T_p where inserted point p is located.

Step 1.2: Search neighbor triangles from T_p, and find out the triangles of which circumcircle the point p falls in.

The first improvement could avoid the problem of triangles' degeneration, but it helps little in reducing the calculation expense. The majority of the time cost of the Delaunay

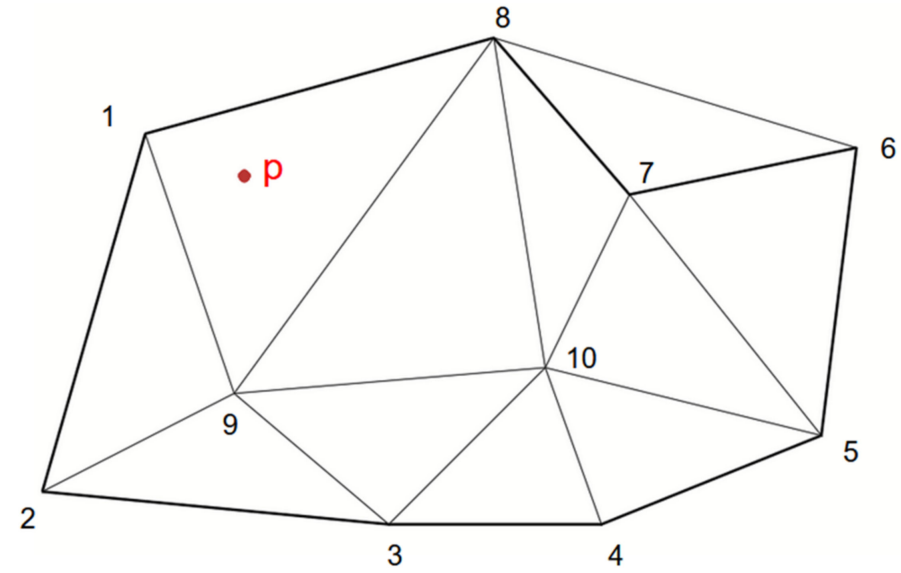
algorithm is the point location search, namely searching the triangle which contains the inserted point [25]. In many cases, step 1.1 requires investigating most triangles to find out T_p. Thus, the second improvement is proposed, providing a quicker search of T_p.

Step 1.1: Select a triangle T_k, where $k = \{0, 1, 2, \ldots, i\}$ denotes the k-th iteration step. Obtain the vector V_k from the triangle's centroid C_k to p.

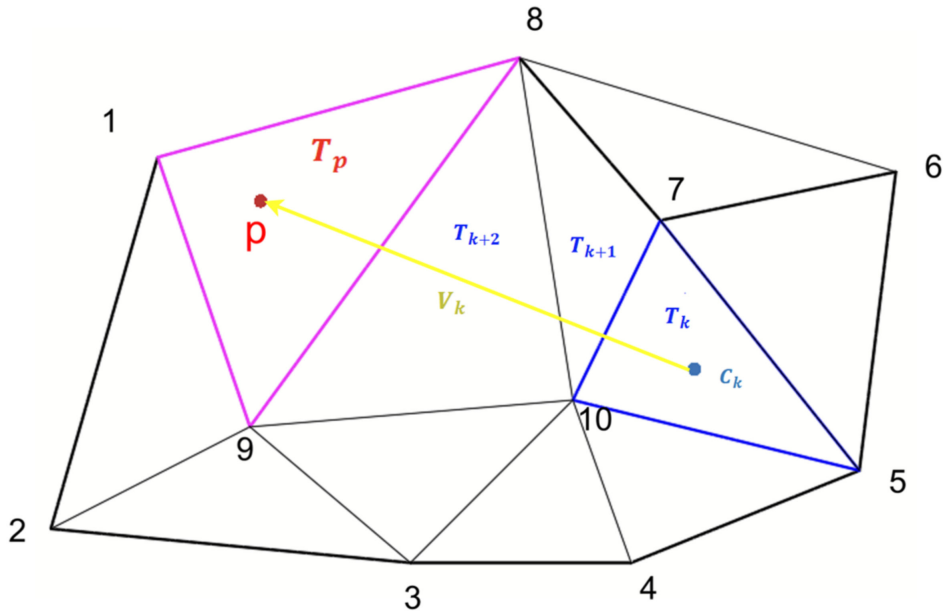
Step 1.2: If p is located in T_k, then T_k is T_p. Otherwise, take T_k's neighbor triangle along the direction of V_k as T_{k+1}, and then repeat Step 1.1.

Step 1.3: Search neighbor triangles from T_p, and find out triangles of which circumcircle the point p falls in.

Figure 5 shows how the directive search works. The search for triangle T_p (in red) starts from T_k (in blue). The neighbor triangle T_{k+1} on the direction of vector V_k connecting C_k and p will be used in the next iteration.



(**a**)



(**b**)

Figure 5. *Cont.*

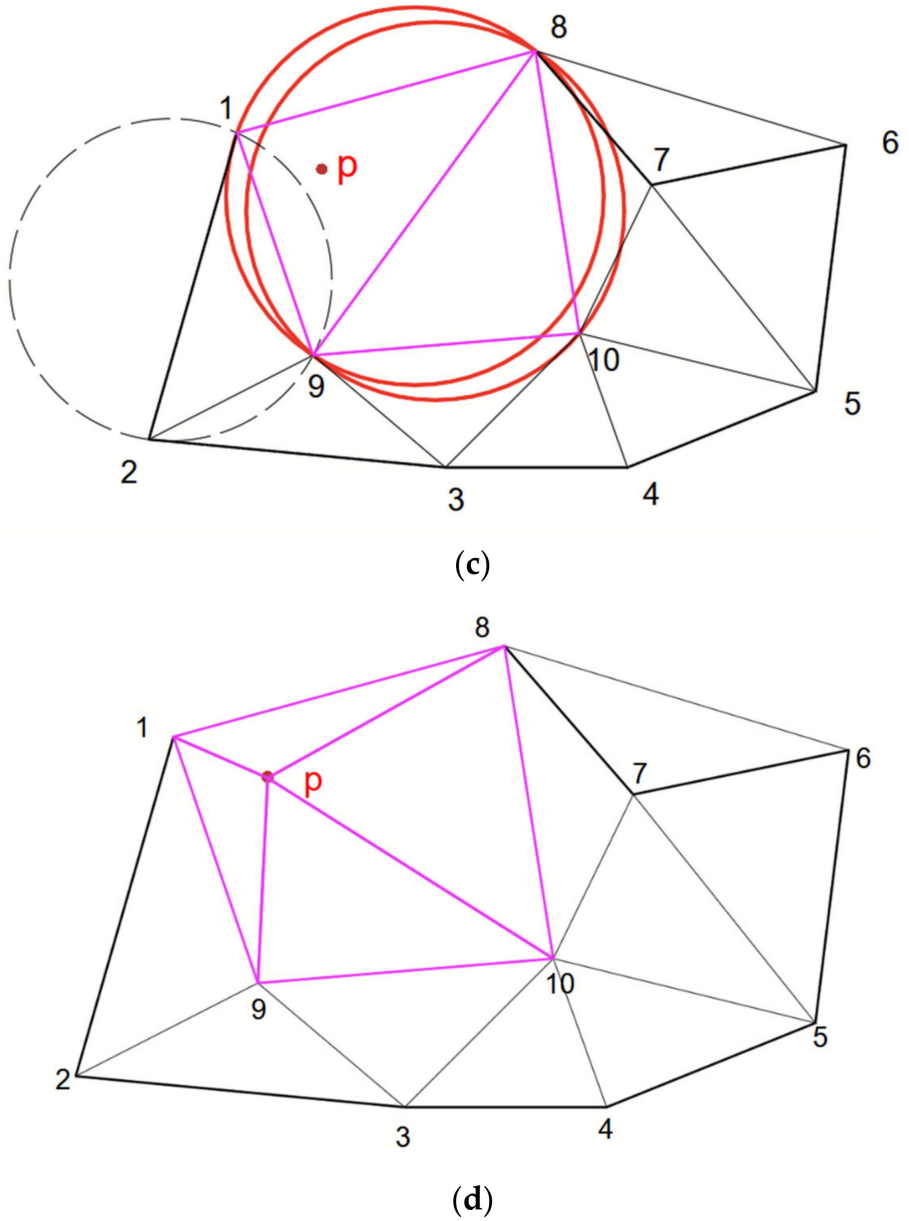


Figure 5. (**a**) Insert a new point p; (**b**) Search along vector V_k to find the triangle containing point p; (**c**) Find triangles of which circumcircle the point p falls in; (**d**) Construct the new Delaunay triangle.

This improvement enables the search to consciously approach T_p, which significantly reduces the number of triangles to be searched during incremental insertion. Here we name it directive search (DS). It is clear that the first iteration triangle T_0 will determine the search process. Thus, deliberately selecting the first iteration triangle T_0 could reduce calculation expenses even further.

In most cases, the point set is stored in a sequence that the next inserted point p_{k+1} is likely near the current point p_k. It could be inferred that the triangle T_{p+1} is likely near to p_k, also. If each iteration starts from a triangle T_0 containing current point p_k, its search process could be quite quicker. Figure 6 presents a typical scenario of this. The search for T_{p+1} (green triangle) starting from the adjacent triangle of the current point (purple triangles) is quicker than from a random triangle (blue triangle) in most cases. Because the start position T_0 is inherited from the current point insertion, this improvement is called heritable initial (HI).

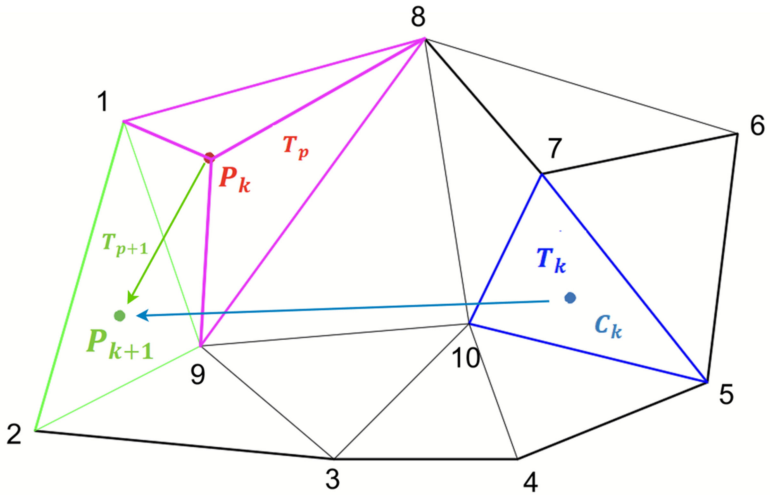


Figure 6. Heritable initial to quicker search for T_{p+1}.

Finally, the previously mentioned improvements, the directive search and the heritable initial position, could be combined. Here, the incremental insertion algorithm ends up being the following procedures:

Step 0: Prepare a super triangle containing all points. The super triangle is T_0.
Step 1: Insert a new point p, and find the "bad triangles". This step consists of 3 sub-steps:

Step 1.1: Select the triangle T_k, where $k = \{0, 1, 2, \ldots, i\}$ denotes the k-th iteration step. Obtaint the vector V_k from the triangle's centroid C_k to p.

Step 1.2: If p is located in T_k, then T_k is T_p. Otherwise, take T_k's neighbor triangle along the direction of V_k as T_{k+1}, and then repeat Step 1.1.

Step 1.3: Search neighbor triangles from T_p, and find triangles of which circumcircle the point p falls in.

Step 2: Delete the "bad triangles" and generate new triangles. This step consists of 2 sub-steps:

Step 2.1: Delete the "bad triangles" and form a polygonal hole.

Step 2.2: Connect the edges of this polygonal hole to point p. And take one of these new triangles as T_0 for the next point.

Step 3: Repeat Step 1, until all points are inserted.

3.2. Circumcircle Radius Check for Alpha Shape

Given the Delaunay Triangulation DT computed from point set S, suppose edge e in the DT of points p and q is a boundary edge of alpha shape S_α. For the definition of alpha shape, an empty circle with its radius $r > 1/\alpha$ must exist containing p and q. It can be inferred that, in the two triangles adjacent to edge e, at least one triangle's circumcircle radius satisfies $r > 1/\alpha$. Thus, we use the circumcircle radius check of Delaunay Triangles instead of the empty circle check. Figure 7a shows an example of a circumcircle radius check. Edge e_1 of points 2 and 3 is a boundary edge of S_α, while the edge e_2 of points 6 and 8 is not. They can be explicitly distinguished by the radius of the circumcircle of their adjacent triangles T_1 and T_2. T_1 is a domestic triangle of S_α since $r_1 < 1/\alpha$, while T_2 is on the outer side of S_α because of $r_2 > 1/\alpha$.

The value of α is determined by point cloud density and the minimal value of hole radius. The circumcircle radius check is implemented in breadth-first search (BFS). All triangles containing the super points are pushed into a to-visit stack. One triangle popped up from the to-visit stack and is labeled as visited, and its neighbor triangles are checked. If the neighbor triangle is unvisited and its circumcircle radius $r > 1/\alpha$, this triangle is pushed into the to-visit stack. Repeat this process until the to-visit stack is empty. All visited triangles stored in set V form the alpha shape S_α, colored by red in Figure 7b.

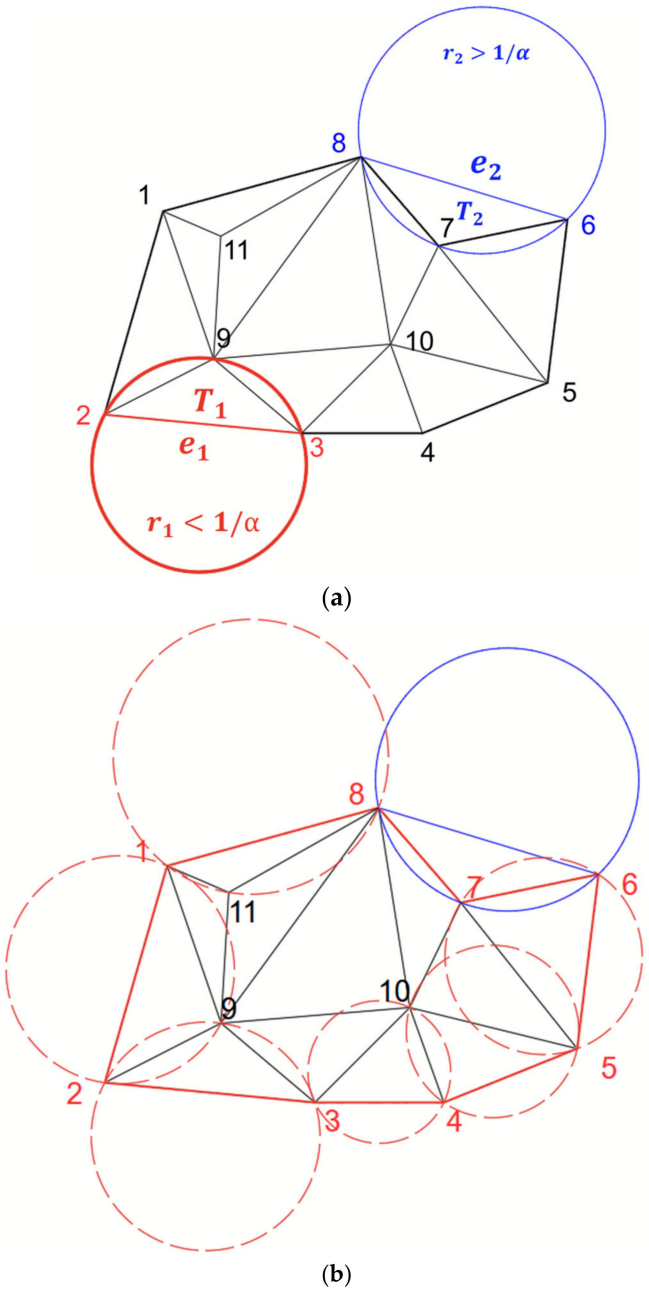


Figure 7. (**a**) Circumcircle radius check to find boundary segment; (**b**) Circumcircle radius check to obtain triangles V within alpha shape.

The set of Delaunay triangles V outlines the alpha shape S_α. Edges of triangles in set V are checked for their shared state. Edges shared by 2 triangles are internal edges, while the other edges are treated as free edges.

The free edges of these triangles may form 2 close loops. One loop contains the super points of Bowyer–Watson incremental insertion and would be discarded. The other loop, without any super points, turns out the outline shape of the concave object. Figure 7b colors the edges of the outline shape in red segments.

4. Experiments

The proposed method is verified in the laboratory. Experiments are carried out on low-density point clouds and high-density point clouds. Each case is to compare the algorithm efficiency and outline extraction speed of DSHI with ordinary Bowyer–Watson scheme and open-source PCL, which is widely used in practical intelligent equipment.

4.1. Case of Low-Density Point Clouds

In this study, 6 test pieces made of acrylic plates are prepared to simulate hull components. An RGBD camera (Percipio PM806-E1 (Percipio, Shanghai, China), 1280 × 960 pixels) is employed to obtain low-density point clouds for test pieces, as shown in Figure 8. Point clouds of test pieces from RGBD cameras are manually separated from the background. Figure 9 shows pictures of 6 test pieces with their point cloud and triangle mesh. Table 1 lists the number of points and triangles of each test piece.

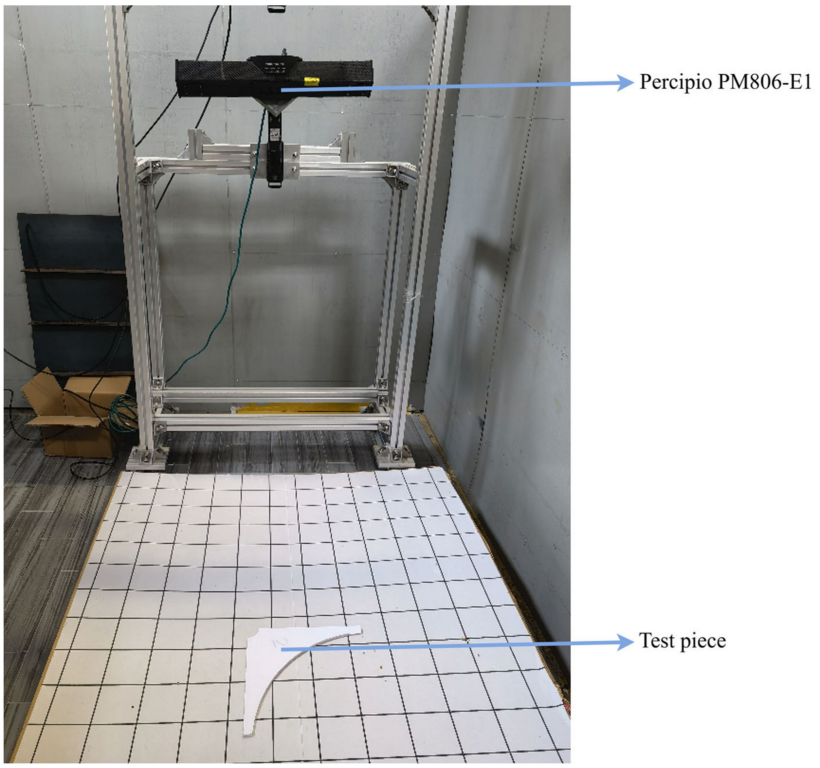


Figure 8. RGBD camera (Percipio PM806-E1).

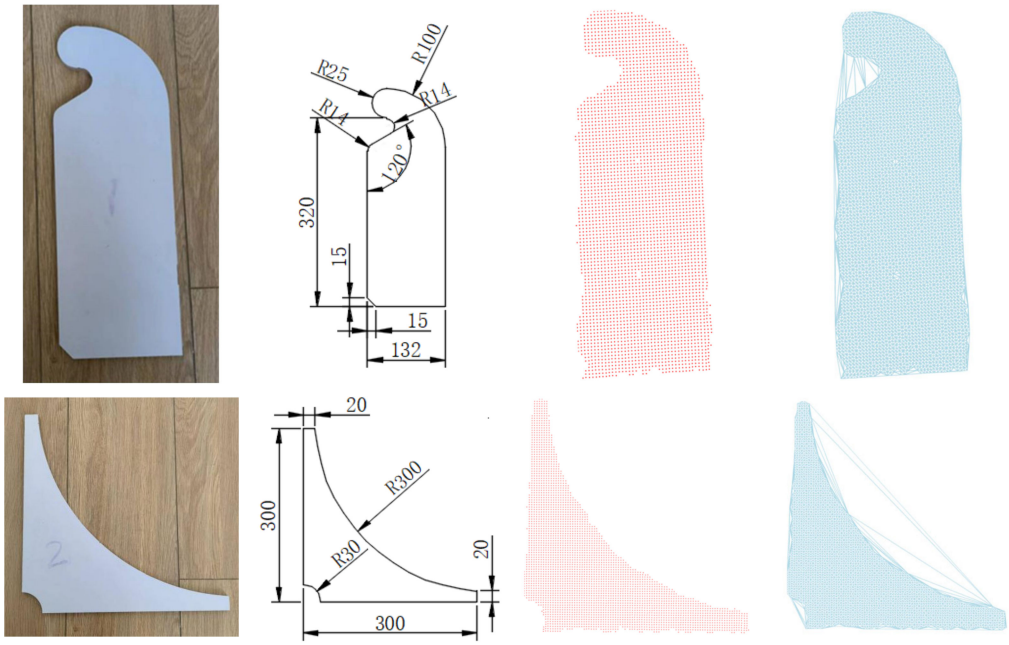


Figure 9. *Cont.*

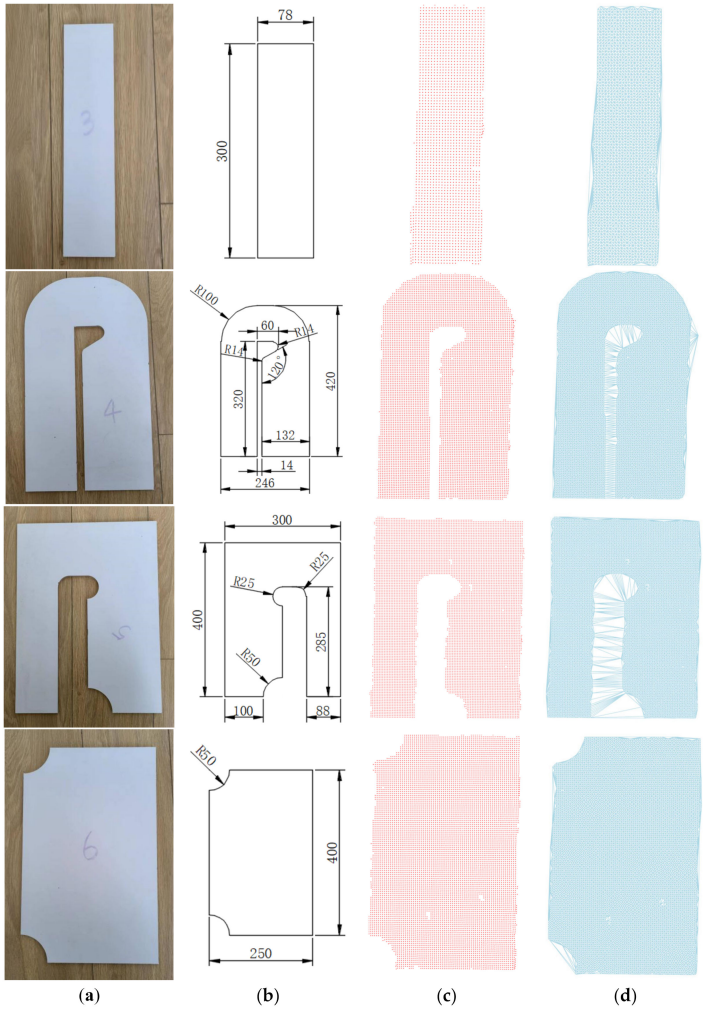


Figure 9. Triangulation test (Percipio). Column (**a**) Photos; Column (**b**) Size; Column (**c**) Point cloud; Column (**d**) Triangle mesh.

Table 1. Number of points and triangle of test pieces (low-density point clouds).

Test Piece	Points	Triangles
1	3716	7356
2	3161	6260
3	2179	4288
4	8258	16,422
5	8828	17,557
6	8696	17,309

To obtain the maximum workspace, this paper uses the minimum 2-norm value of the cable force array as the objective value, combines the static equilibrium equation constraints and the cable force inequality constraints, and solves the cable force using an optimization algorithm.

The incremental insertion scheme for triangulation mesh generation is implemented in Visual Studio C#. In total, there are 3 versions of the incremental insertion algorithm studied here, the ordinary Bowyer–Watson method [25], one with directive search (DS), and the other with both directive search and heritable initial (DSHI).

Here, the number of searched triangles during incremental insertion is taken as the index of search cost, because it is independent of the coding languages, operation systems or hardware configuration. The smaller the number of it is, the lesser the calculation cost paid during mesh generation. Table 2 compares the result of 3 algorithms in this experiment,

the ordinary Bowyer–Watson scheme, one with directive search (DS), and the other with both directive search and heritable initial (DSHI).

Table 2. Search cost of incremental insertions (low-density point clouds).

Test Piece	Search Cost (Ordinary)	Search Cost (DS)	DS/ Ordinary	Search Cost (DSHI)	DSHI/DS	DSHI/ Ordinary
1	11,410,445	255,881	2.24%	29,881	11.68%	2.62‰
2	9,215,350	345,249	3.75%	25,599	7.41%	2.78‰
3	4,096,232	212,507	5.19%	12,761	6.00%	3.12‰
4	62,778,422	722,185	1.15%	72,733	10.07%	1.16‰
5	72,878,539	816,222	1.12%	79,815	9.78%	1.1‰
6	69,991,500	1,112,193	1.59%	77,474	6.97%	1.11‰

Comparing DS and the ordinary scheme, directive search cuts the triangle search cost down to about 1.12%~5.19%. And comparing DSHI with DS, the heritable initial position can further cut the triangle search cost down to 6.00%~11.68%. Comparing DSHI and the ordinary scheme, the search cost is cut down to 1.1‰~3.12‰. It is equivalent to a 300~1000 times improvement in efficiency. This results in a very remarkable improvement in mesh generation efficiency.

The outline extraction test is conducted in 3 groups. The DSHI is compared with the open-source algorithm PCL and the ordinary Bowyer–Watson method in time cost and outline shapes. The time costs of mesh generation on the same points are listed in Table 3 and Figure 10. The test pieces outlines extracted by these three methods are compared in Figure 11. It is shown that the DSHI enjoys better efficiency in most experiments while ensuring the same quality of the outlines. Compared with the PCL, DSHI is 2–3 times faster in most experiments, and compared with the ordinary Bowyer–Watson method, the efficiency improvement effect is more significant, increased by 4–7 times. Meanwhile, under low-density point clouds, the PCL method may not be able to accurately extract outlines in certain experiments.

Table 3. Time cost comparison of outline extraction (low-density point clouds).

Test Piece	Time Cost (DSHI)	Time Cost (PCL)	Time Cost (Ordinary)
1	24.03 ms	34.8 ms	96.12 ms
2	44.05 ms	32.81 ms	216.3 ms
3	40.05 ms	18.77 ms	156.2 ms
4	48.06 ms	96.52 ms	374.67 ms
5	44.05 ms	86.5 ms	336.54 ms
6	36.05 ms	106.9 ms	196.26 ms

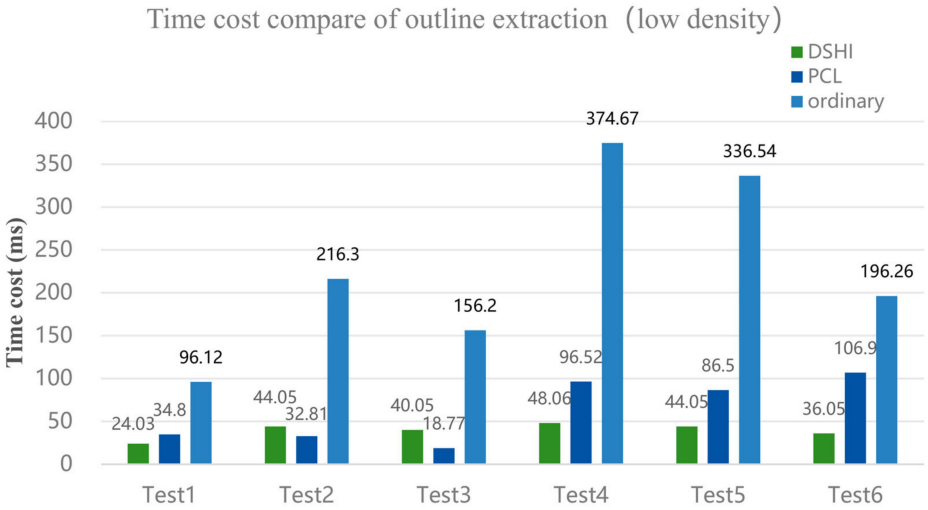


Figure 10. Time cost comparison of outline extraction (low-density point clouds).

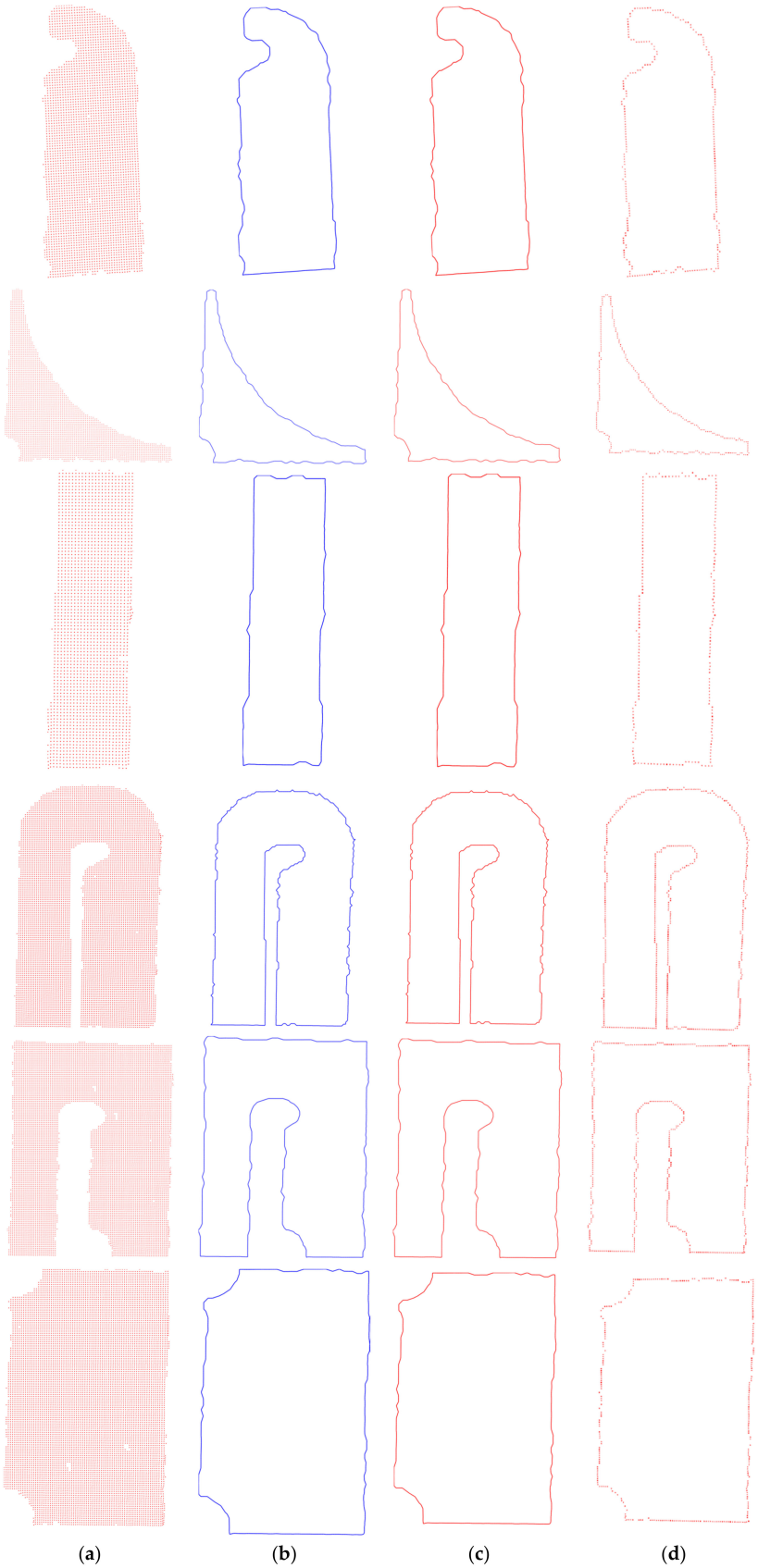


Figure 11. Column (**a**) Points of test pieces (low-density point clouds); Column (**b**) Outline from Bowyer–Watson triangulation; Column (**c**) Outline from DSHI triangulation; Column (**d**) Outline from PCL.

4.2. Case of High-Density Point Clouds

The second group employs a Mech-eye LSR-L (2048 × 1536 pixels) RGBD camera obtaining high-density points cloud of each test piece with the same material as the first group (acrylic plates), as shown in Figure 12. Table 4 lists the number of points and triangles of each test piece's cloud.

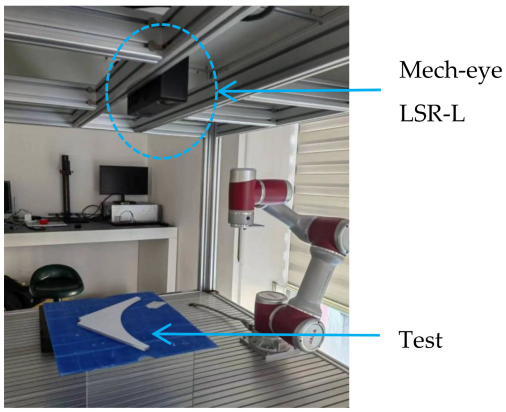


Figure 12. RGBD camera (Mech-eye LSR-L).

Table 4. Number of points and triangle of test pieces (high-density point cloud).

Test Piece	Points	Triangles
1	763,685	1,526,916
2	513,513	1,026,757
3	406,434	812,588
4	1,591,649	3,182,968
5	1,015,985	2,031,739
6	997,571	1,994,959

In experiments, the ordinary Bowyer–Watson crashes in processing the high-density point clouds because of overwhelming computation burdens. Thus, comparison of search cost and outlines are only compared between DSHI and PCL.

This experiment is to verify the capacity of extracting outlines of concave objects in high-density point clouds. Due to the points of test pieces here being in quite large numbers, they are down-sampled to fit the size of Figure 13. The points visualized here are lesser than actual points, but the algorithm still works on the original high-density point clouds. Results of outline extraction are also illustrated in red segments in Figure 13. In comparing with the shape of acrylic plates, it is demonstrated that the outline of high-density point clouds from DSHI successfully meets human's visual perception, while results from PCL have jaggies in some corners or edges.

The running time costs of DSHI and PCL in high-density point clouds are also compared in Table 5 and Figure 14. It is clear that DSHI runs 2–4 times faster in comparing with PCL. This method alleviates the computation burden of the insertion triangulation scheme, providing one more affordable choice of points outline extraction.

Table 5. Time cost of outline extraction (high-density point cloud).

Test Piece	Time Cost (DSHI)	Time Cost (PCL)
1	2854.14 ms	10,267.7 ms
2	3173.56 ms	7072.56 ms
3	1818.06 ms	5323.69 ms
4	9970.89 ms	27,544.8 ms
5	6987.71 ms	13,908.4 ms
6	3189.75 ms	13,399.7 ms

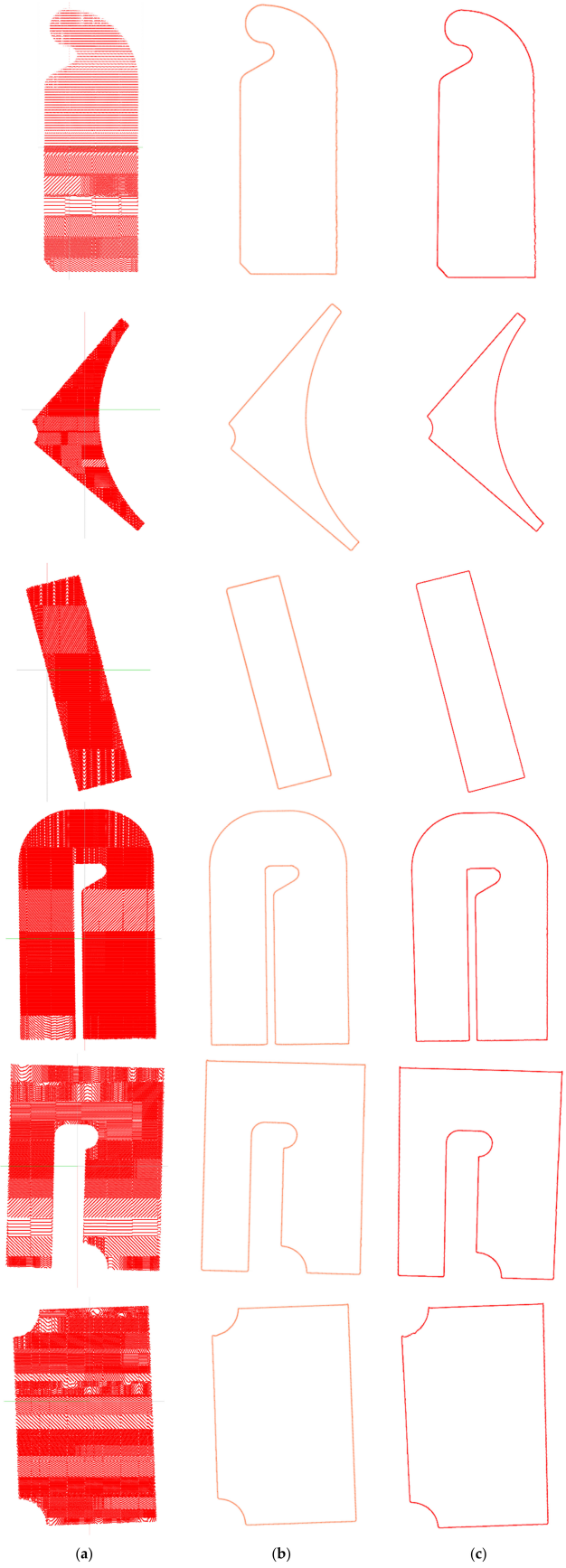


Figure 13. Points and outline test (high-density point cloud). Column (**a**) Points; Column (**b**) Outline from DSHI; Column (**c**) Outline from PCL.

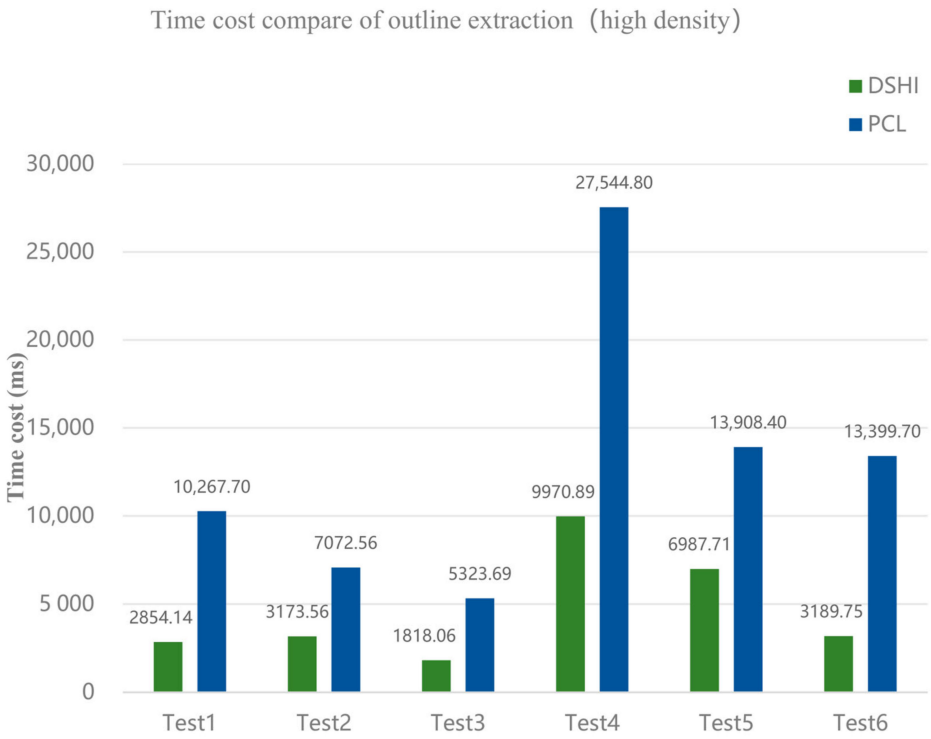


Figure 14. Time cost comparison of outline extraction (high-density point cloud).

Since the ordinary BW scheme fails in dealing with the point clouds of high density, the second case of original point clouds is appropriately down-sampled to compare the efficiency of the three methods and the speed of contour extraction under high-density and high-quality point clouds. Table 6 lists the number of points and triangles of each test piece's cloud after down-sampling.

Table 6. Number of points of test pieces (down-sampled).

Test Piece	Points	Triangles
1	17,604	35,029
2	16,295	32,400
3	12,263	24,381
4	17,130	34,060
5	15,531	30,899
6	14,809	29,478

Table 7 compares the efficiency of 3 algorithms using the same method as in Section 4.2, the ordinary Bowyer–Watson scheme, one with directive search (DS), and the other with both directive search and heritable initial (DSHI). Comparing DS and the ordinary scheme, the directive search cuts the triangles search cost down to about 0.53%~1.65%. And comparing DSHI with DS, the heritable initial position can further cut the triangles search cost down to 2.63%~6.26%. Comparing DSHI and the ordinary scheme, the search cost is cut down to 0.33‰~0.51‰. It is equivalent to a 1000~3000 times improvement in efficiency. The results show that the efficiency improvement of the DSHI method is more significant under high-density point clouds.

Similarly, we used the same method as in Section 4.2 to compare the time cost of outlines extraction of the three methods under high-density point clouds. The results are listed in Table 8 and Figure 15. The test pieces outlines extracted by the three methods are shown in Figure 16. It is shown that compared with low-density point clouds, the outlines extraction efficiency of the DSHI algorithm is more significantly improved under high-density point clouds. Compared with the PCL method, the efficiency is increased by

2–4 times, and compared with the ordinary Bowyer–Watson method, the efficiency is even increased by 7–16 times.

Table 7. Search cost of incremental insertions (down-sampled).

Test Piece	Search Cost (Ordinary)	Search Cost (DS)	DS/ Ordinary	Search Cost (DSHI)	DSHI/DS	DSHI/ Ordinary
1	301,184,708	4,067,904	1.35%	106,970	2.63%	0.36‰
2	259,483,564	2,737,748	1.06%	104,982	3.83%	0.4‰
3	146,277,609	2,410,395	1.65%	74,661	3.10%	0.51‰
4	285,813,272	1,504,871	0.53%	94,157	6.26%	0.33‰
5	232,072,475	2,795,457	1.20%	82,272	2.94%	0.35‰
6	211,203,409	2,815,502	1.33%	83,908	2.98%	0.4‰

Table 8. Time cost of outline extraction after down-sampling (down-sampled).

Test Piece	Time Cost (DSHI)	Time Cost (PCL)	Time Cost (Ordinary)
1	104.28 ms	220.35 ms	925.43 ms
2	78.53 ms	165.68 ms	1325.7 ms
3	47.33 ms	143.52 ms	492.26 ms
4	94.16 ms	207.7 ms	926.27 ms
5	90.14 ms	164.04 ms	1106.24 ms
6	63.55 ms	163.06 ms	455.05 ms

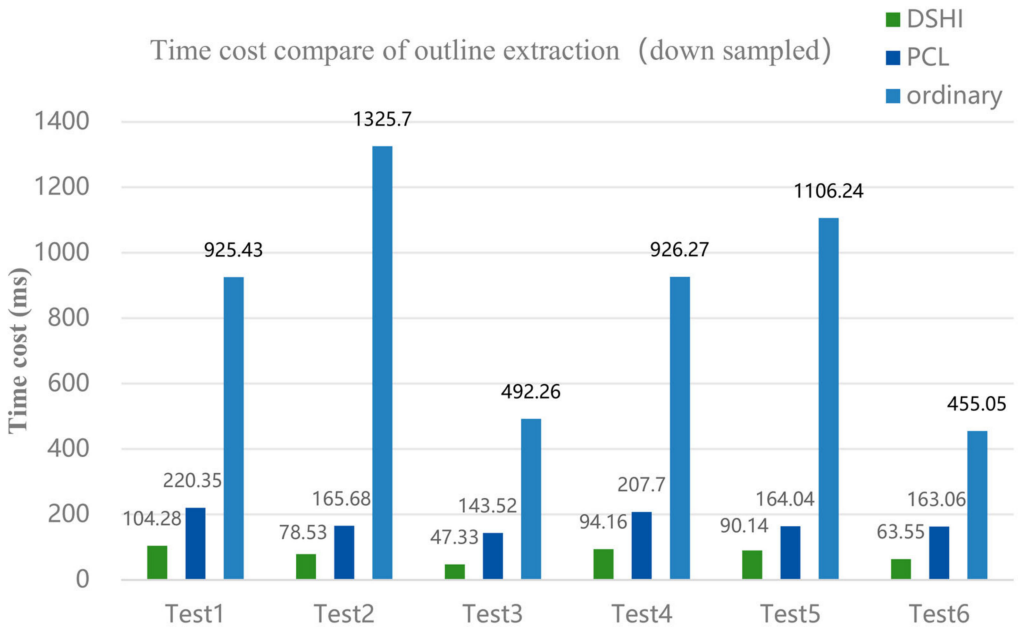


Figure 15. Time cost comparison of outline extraction (down-sampled).

4.3. Robotic Prototype Application

The DSHI method is applied to actual automatic edge-grinding equipment for ship components, as shown in Figure 17. The equipment consists of a carrying robot with an RGBD camera, a grinding robot with a milling cutter and two electromagnetic grinding platforms. Based on the point cloud from the RGBD camera, the vision algorithm detects the center of mass of the workpiece to guide the robot to carry the workpiece to the grinding platform and extract the outline of the workpiece. The robot performs edge grinding based on the workpiece outline information extracted by the vision algorithm.

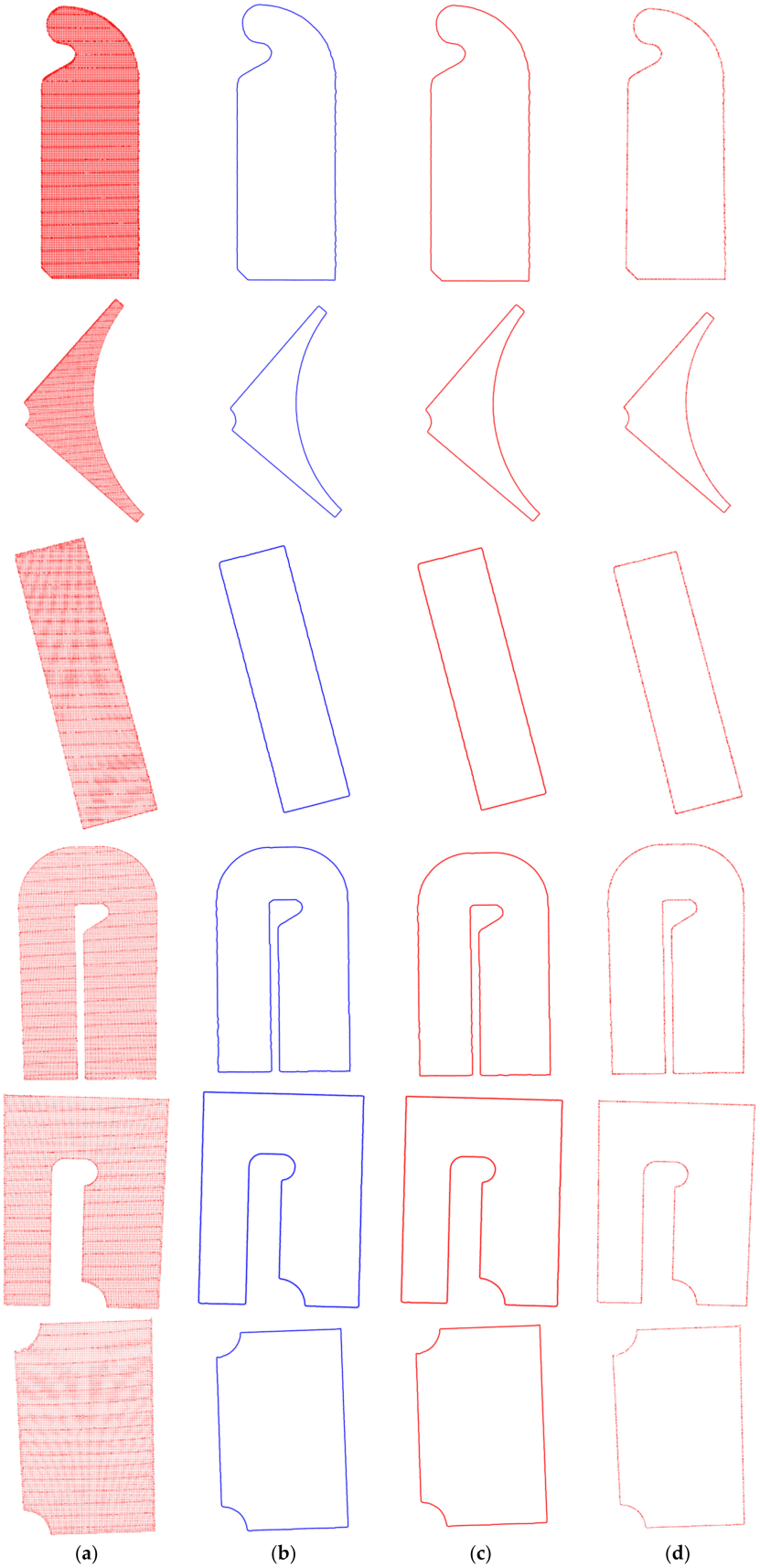


Figure 16. Column (**a**) Points of test pieces (down-sampling); Column (**b**) Outline from Bowyer–Watson triangulation; Column (**c**) Outline from DSHI triangulation; Column (**d**): Outline from PCL.

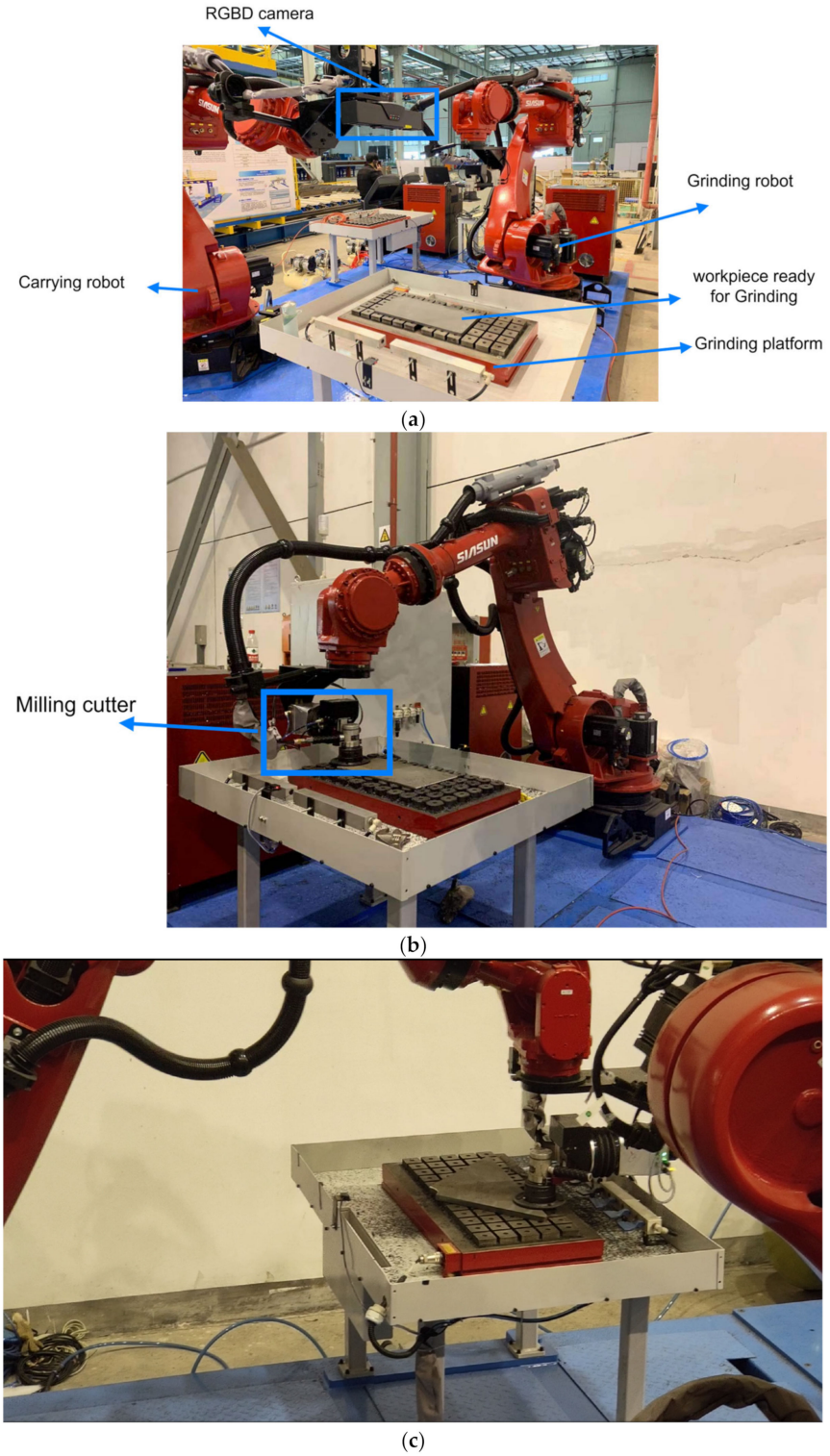
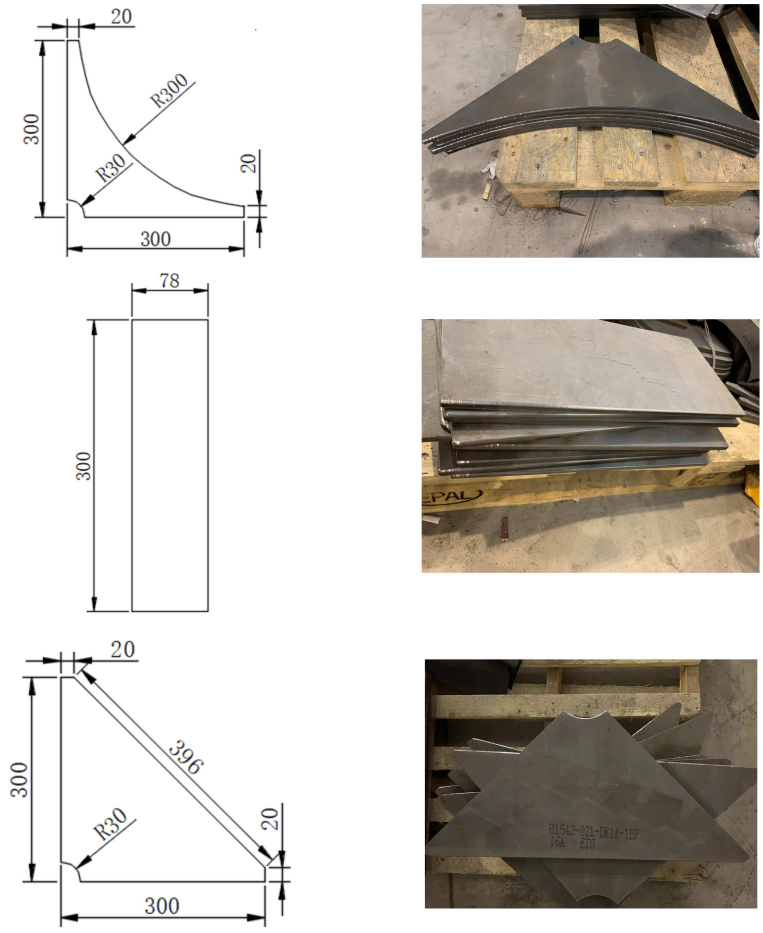


Figure 17. (**a**) Automatic edge-grinding equipment; (**b**) Industrial robot grinds rectangular pieces; (**c**) Industrial robot grinds triangle pieces.

Three types of marine steel workpieces for edge grinding were provided by the shipyard, as shown in Figure 18. The dimensions of the workpieces and the edge-grinding effects are also presented in Figure 18. It can be proved that the DSHI method can accurately guide the industrial robot to perform edge grinding of ship components. The edge treating quality meets the requirements in shipbuilding.



(**a**) (**b**)

Figure 18. Ship components treated by the DSHI method. Column (**a**) Size; Column (**b**) Edge grinding effect.

The edge-grinding robots may significantly save labor costs in shipbuilding. There are even more benefits, including more stable quality and less health-hazardous. Thus, automation in shipbuilding is an inevitable trend.

A major advantage of a vision-guided working mode is that the robot system is self-programmed. It can be finished in seconds when dealing with a new type of product. In contrast, the manual programming working mode requires laborious reprogramming for each new product type. Based on the experience of specialists, the time cost of teaching or reprogramming is by hours or days, depending on the complexity of the products. This explains the dilemma in shipbuilding automation: labor workers are faster than robots in small-batch production. The vision-guided working mode could be an optimized solution, as it imitates the manual operators, from observation, perception, to operation.

The method in this paper is also applicable to other processes, including welding, sorting and assembling. Figure 19 shows a common scenario in shipbuilding, in which the collar plate is lap-welded onto the plate. Currently, welding jobs in this scenario are still performed by labor workers, partly because the collar plate varies in shape and size. With the aid of DSHI, a vision-guided robotic welding is not far off. Therefore, this mode has general applicability in the field of shipbuilding and can be applied to other industries characterized by small-batch, high-variety production.

Figure 19. Lap weld seam surrounds collar plate.

5. Conclusions

In this paper, an attempt has been made to develop a vision-guided robot system for marine components edge grinding. It can save on the cost of labor workers. The novel robot does not require the shape data from the CAD model or manual teaching. Outline extraction from point cloud plays a crucial part in its workflow. The efficiency and robustness of the algorithm will affect robots' productive performance.

The use of the circumcircle radius check in Delaunay triangulation mesh instead of the empty circle check for obtaining the outline of random-shaped point clouds has been discussed. Improvements have been proposed to the incremental insertion method and combined to create a new method known as Directive Searching and Heritable Initial (DSHI). As a result of experiments in outline extraction for test pieces with common hull components' shapes, the following conclusions have been drawn:

DSHI significantly reduces the search cost during incremental insertion, leading to a faster triangulation mesh method than the ordinary Bowyer–Watson algorithm by nearly 300 to 3000 times for low-quality point cloud. As for high-quality point clouds, the ordinary Bowyer–Watson method crashes, while DSHI successfully presents the results.

The use of a circumcircle radius check provides a more accurate fit to the visual perception of workpieces. In terms of outline extraction, DSHI has improved the extraction efficiency by 4–16 times compared to the ordinary Bowyer–Watson method and has also increased the extraction efficiency by 2–4 times compared to PCL. DSHI ensures stable outline results, while the PCL algorithm fails partially in corners for high-quality point clouds.

Application on a robotic prototype proves that it is reliable in real manufacturing scenarios. It is verified in the edge grinding of marine components.

Taking these into account, it can be concluded that the DSHI method has demonstrated potential for intelligent manufacturing applications, such as automatic edge grinding and other processes. This paper's work focuses on flat components grinding. More product types (e.g., workpieces of three dimensions) and more procession (e.g., vision-guided welding) are to be studied in the future.

Author Contributions: Conceptualization, H.Y. and Y.Z.; methodology, H.Y. and C.N.; software, C.N.; validation, H.Y., C.N. and J.D.; writing—original draft preparation, H.Y. and C.N.; writing—review and editing, Y.Z. and T.Z.; visualization, X.J. and R.Z.; funding acquisition, J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Marine Equipment Foresight Innovation Union Project (MEFP), grant number ZCJDQZ202309B02.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Leo, P.F.; Selvaraj, T. Vision Assisted Robotic Deburring of Edge Burrs in Cast Parts. *Procedia Eng.* **2014**, *97*, 1906–1914. [CrossRef]
2. Tellaeche, A.; Arana, R. Robust 3D Object Model Reconstruction and Matching for Complex Automated Deburring Operations. *J. Imaging* **2016**, *2*, 8. [CrossRef]
3. Wen, L.L.; He, X.; Gang, Z.; Si, J.Y.; Zhou, P.Y. Hand–Eye Calibration in Visually-Guided Robot Grinding. *IEEE Trans. Cybern.* **2016**, *46*, 2634–2642. [CrossRef]
4. Theodosios, P. A review of algorithms for shape analysis. *Comput. Graph. Image Process.* **1978**, *7*, 243–258. [CrossRef]
5. Widyaningrum, E.; Gorte, B.; Lindenbergh, R. Automatic Building Outline Extraction from ALS Point Clouds by Ordered Points Aided Hough Transform. *Remote Sens.* **2019**, *11*, 1727. [CrossRef]
6. Available online: https://github.com/Geodan/building-boundary (accessed on 24 January 2024).
7. Edelsbrunner, H.; Kirkpatrick, D.; Seidel, R. On the shape of a set of points in the plane. *IEEE Trans. Inf. Theory* **1983**, *29*, 551–559. [CrossRef]
8. Jayachandra, M.R.; George, M.T. Computation of 3D skeletons using a delaunay triangulation technique. *Comput.-Aided Des.* **1995**, *27*, 677–694. [CrossRef]
9. Zollini, S.; Dominici, D.; Alicandro, M.; Cuevas-González, M.; Angelats, E.; Ribas, F.; Simarro, G. New Methodology for Shoreline Extraction Using Optical and Radar (SAR) Satellite Imagery. *J. Mar. Sci. Eng.* **2023**, *11*, 627. [CrossRef]
10. Elyta, W.; Peters, R.Y.; Roderik, C.L. Building outline extraction from ALS point clouds using medial axis transform descriptors. *Pattern Recognit.* **2020**, *106*, 107447.
11. Liu, Y.J.; Chen, Z.Q.; Tang, K. Construction of iso-contours, bisectors and Voronoi diagrams on triangulated surfaces. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 1502–1517. [CrossRef]
12. Lawson, C.L. Software for C1 surface interpolation. In Proceedings of the Symposium Conducted by the Mathematics Research Center, the University of Wisconsin–Madison, Madison, WI, USA, 28–30 March 1977; pp. 161–194. [CrossRef]
13. Lewis, B.A.; Robinson, J.S. Triangulation of planar regions with applications. *Comput. J.* **1978**, *21*, 324–332. [CrossRef]
14. Dwyer, R.A. Higher-dimensional Voronoi diagrams in linear expected time. *Discret. Comput. Geom.* **1991**, *6*, 343–367. [CrossRef]
15. Green, P.J.; Sibson, R. Computing Dirichlet tessellations in the plane. *Comput. J.* **1978**, *21*, 168–173. [CrossRef]
16. Fortune, S. A sweepline algorithm for Voronoi diagrams. *Algorithmica* **1987**, *2*, 153–174. [CrossRef]
17. Bradford, B.C. Computational Geometry with Imprecise Data and Arithmetic. Ph.D. Thesis, Princeton University, Princeton, NJ, USA, 1992.
18. Peter, S.; Robert, L.; Scot, D. A comparison of sequential Delaunay triangulation algorithms. *Comput. Geom.* **1997**, *7*, 361–385. [CrossRef]
19. Rui, Y.K.; Wang, J.C. A new study of compound algorithm based on sweepline and divide-and-conquer algorithms for constructing Delaunay triangulation. *Acta Geod. Cartogr. Sin.* **2007**, *36*, 358–362. [CrossRef]
20. Lo, S.H. Parallel Delaunay triangulation—Application to two dimensions. *Finite Elem. Anal. Des.* **2012**, *62*, 37–48. [CrossRef]
21. Bi, S.B.; Chen, D.Q.; Yan, J.; Guo, Y. Planar Delaunay triangulation algorithm based on 2D convex hull. *Comput. Sci.* **2014**, *41*, 317–320.
22. Li, J.; Li, D.R.; Shao, Z.F. A streaming data Delaunay triangulation algorithm based on parallel computing. *Geomat. Inf. Sci. Wuhan Univ.* **2013**, *38*, 794–798.
23. Cai, Y.L.; Xiao, S.M.; Qi, L. Locking paralleled GPU-based method research for unstructured mesh generation. *Comput. Eng. Appl.* **2014**, *50*, 56–60.
24. Sloan, S.W. A fast algorithm for constructing Delaunay triangulations in the plane. *Adv. Eng. Softw.* **1987**, *9*, 34–55. [CrossRef]
25. Available online: https://github.com/mapbox/delaunator?ref=gorillasun.de (accessed on 24 January 2024).