2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# Max orientation coverage: efficient path planning to avoid collisions in the CNC milling of 3D objects

Xin Chen[1], Thomas M. Tucker[2], Thomas R. Kurfess[1], Richard Vuduc[1], Liting Hu[3]

***Abstract*— Most path planning algorithms for covering a complex 3D object ignore physical limitations or constraints on a robot's motion. Adhering to such constraints for a given path can slow down the time to cover the path because the motion may need to be adjusted. This work considers a scenario in computer numerical control (CNC) milling applications, where the robot is a cutting tool that needs to cover the surface of a complex 3D object under the following constraint: for every point on the generated path, the robot must be assigned an accessible orientation to avoid collisions between it and other parts of the object. Our proposed approach, which we call max orientation coverage, employs a two-step optimization scheme. It can improve path efficiency with respect to both the length of the path and the cost of dealing with the collision-avoiding constraints. We evaluate our approach through extensive simulation studies on four CAD benchmarks against a state-of-the-art baseline. We show that our proposed approach can improve the efficiency of the path by 29.7% on average compared with the baseline and the improvement goes up to 46.5% for certain complex objects.**

## I. Introduction

Coverage path planning has many applications, including robotic vacuum cleaners, aerial robotic inspection [2], [4], 3D printing [30], [31], [32], and autonomous underwater vehicles (AUV) [25], among others. To facilitate autonomous path planning for complex environments or objects, a robot must be equipped with algorithms capable of computing efficient paths that achieve full coverage while respecting any limitations or constraints on the robot's motion.

We are specifically motivated by problems in computer numerical control (CNC) milling, and Figure 1 gives an example from that domain. The problem is to cover the object (e.g., the head) where the "robot" is a milling tool composed of multiple cylinders. The constraint is that the robot (tool) and the object ought not collide *except* at the pivot point, which is where the end of the tool touches the surface. Any other collision would be detrimental to the milling process, damaging the robot or the target. Each pivot point has an accessibility map (AM) constructed to determine which tool orientations are inaccessible (e.g., the top right side of Figure 1 is the AM and the bottom right is the corresponding orientation). When generating a valid path, the algorithm has to ensure that no collision exists by selecting only accessible orientations at all points.

[1]Georgia Institute of Technology, Atlanta, GA, 30332, USA, {xchen384,kurfess,richie}@gatech.edu
[2]Tucker Innovations Inc., Waxhaw, NC, 28173-9054, USA, tommy@tuckerinnovations.com
[3]Florida International University, Miami, FL, 33199, USA, lhu@cs.fiu.edu

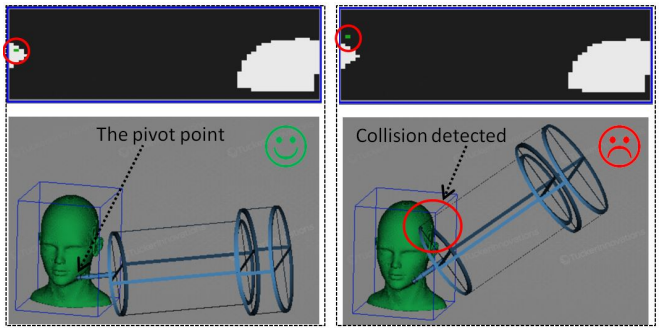


Fig. 1: An example of AM shows two orientations: accessible orientation (on the left) and inaccessible orientation (on the right), corresponding to the white point and the black point in the top AM respectively. An AM at $(r * c)$-resolution is discretized uniformly into $r * c$ points, with each point denoting a $(\theta, \varphi)$ orientation in a spherical coordinate system.

Conventional path planning methods either assume that the robot is under no constraint or focus on the optimization of the path length, and ignore this type of constraint. To satisfy the constraint given a fixed path may require conservative operations; in CNC, for instance, that could be retracting the tool and resetting its position or orientation, which can be very expensive in practice. Focusing on the path length only can underestimate such costs, leading to inefficient paths.

The tool path planning problem differs from conventional coverage path planning (CPP) as it is specifically designed for generating collision-free milling paths. Most prior approaches assume the tool is a single cylinder and only a small piece of the object surface can cause collisions. However, in reality a tool is better-modeled geometrically by multiple cylinders of various sizes, with possible collisions occurring and anywhere on the object. Checking collisions in this setting demands massive computational resources, which is the reason why most tool path planning methods make simplified assumptions. These approaches are hard to extend for the more general coverage task we consider.

We propose a novel path planning algorithm, called *max orientation coverage*, that constructs an efficient path generally covering an arbitrary object. We consider both the cost of path length and the cost of operations handling the constraints. It has two components.

The first is a segmentation algorithm, which we call *max segmentation* (Section IV-A). Prior segmentation algorithms rely on random sampling to achieve a full coverage. Doing so is usually unstable and leads to a large number of segments, resulting in a higher likelihood of redundant coverage. By

contrast, our segmentation algorithm can obtain a stable result and reduce the number of segments. Our approach is to convert the coverage optimization problem into a minimal vertex cover (MVC) problem.

The other component is a new coverage algorithm based on orientation (Section IV-B). Most earlier approaches focus on optimizations of the path length and do not consider the effects of dealing with the constraints on the robot. By contrast, our approach does so, trading off small increases in path length to significantly lower the cost of enforcing the tool constraints and thus yield better overall paths.

Additionally, we conduct several simulations to help validate and evaluate our approach. In particular, we test it on four CAD benchmark objects against a previously proposed random sampling-based coverage algorithm [10], [4], [12], [25]. We observe that our max orientation coverage improves the path efficiency by 29.7% on average and the improvement goes up to 46.5% for one of the more geometrically complex objects, the dragon (Figure 10).

## II. Related work

Many contributions have been made to address the CPP problem. We divide them into three types: the approaches of covering general environments, the approaches of covering 3D objects and the approaches of milling applications.

There are several in-depth and comprehensive surveys on coverage path planning [17], [3]. Gabriely presents an online approach that constructs a systematic spiral path with a spanning tree algorithm [13]. Luo presents an algorithm that utilizes neural network to generate a real-time path [23]. Acar applies cellular decomposition considering sensor-based detector [1]. These approaches focus on the path lengths as the optimization target, while our method concerns an additional cost of handling the restraints.

For the approaches that cover 3D objects, random sampling-based coverage algorithms might be regarded state-of-the-art for handling versatile 3D coverage problems [10], [12], [4], [5], [2], [25], [26], [16]. They operate using a two-step optimization. The first step is a segmentation problem to compute a minimal set of viewpoints using random sampling to achieve a complete coverage of the target structure. This step is equivalent to solving a variant of art gallery problem (AGP), which is NP-hard problem. In the second step, a minimum cost tour over all the viewpoints is searched to optimize the tour length. This involves solving a variant of travel salesman problem (TSP). Although the two problems are NP-hard, there are fast algorithms that approximately solve the two problems for AGP [24], [18] and TSP [11], [27]. Our proposed method follows the structure of the two-step optimization to reduce the computational cost. Instead of randomly sampling, our method collects viewpoints by solving a MVC problem.

The tool path planning problem is another type of problem that concerns avoiding collisions for an efficient milling process [7], [22], [8], [20], [28], [9]. Jun presents a methodology of optimizing and smoothing the tool orientation control 5-axis sculptured surface machining [22]. Hsueh proposed a two-step method for preventing tool collisions in 5-axis machining [20]. Each step adjusts the tilt angle and the yaw angle, respectively.

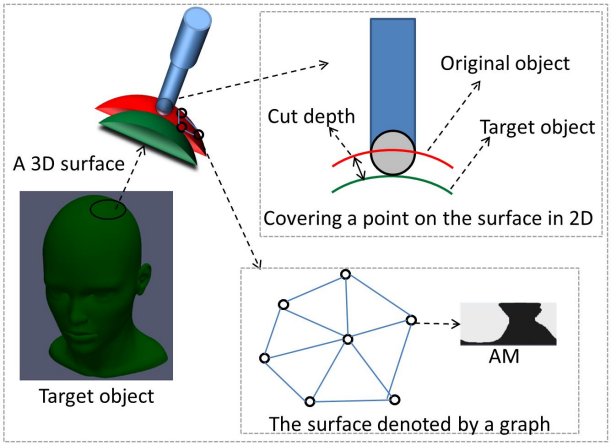


Fig. 2: A solution to the CPP problem is one step of machining one layer in the milling application.

## III. Background and Problem statement

The CPP problem can be used for our CNC milling application, as explained below. We describe the input to our problem, the constraint and the problem statement. Lastly, we define the cost that is our optimization target.

### A. Coverage path planning in CNC milling

In CNC milling, the goal is to cut or shape a given input object (e.g., the stock in Figure 1) into a target (e.g., the head). We assume that the target object fits within the stock. The CPP problem is to cover the surface of a 3D object. To obtain the target object for a milling application, multiple steps of solving a CPP problem are required.

Figure 2 illustrates one step of covering the surface of the head for the milling process. The original object is the object before applying the milling and the target object is the result after it. The space difference represented by the cut depth in Figure 2 denotes the materials that are machined off by the path generated by the CPP problem. Usually the cut depth equates to the size of the robot. For these steps, the milling process uses various robot sizes in a decreasing order. Initially, a large robot size is preferred to get a coarse shape. Finally, a small robot size is used to obtain a high-quality surface.

The surface of the object is represented by a dense mesh as shown in the right bottom of Figure 2. Traversing all vertices in the mesh can achieve the coverage of the object surface. To ensure a safe movement from one vertex to another, each vertex has a corresponding AM, indicating that only accessible orientation can be specified. Note that at a vertex, we only allow the movement to its neighboring vertex following the AM. This paper studies the CPP problem as a single step that covers an arbitrary object.

### B. Accessibility maps as inputs to our problem

An AM is given as an input, since our focus is on how to construct an efficient collision-free path. (Efficiently generat-

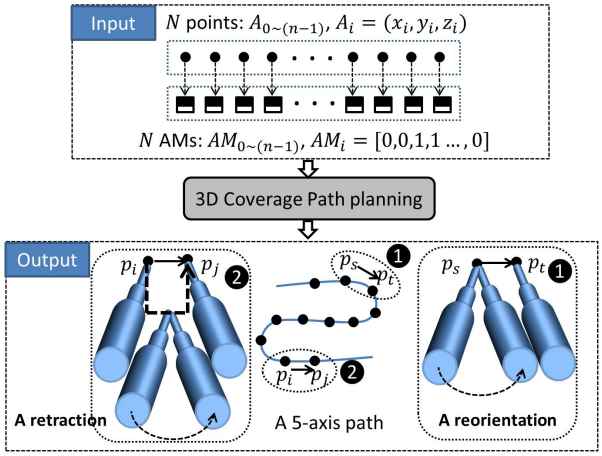


Fig. 3: The input of our problem is the $N$ points on the surface of the target 3D object and the corresponding $N$ AM. The output is a 5-axis $(x, y, z, \theta, \varphi)$ path, where $(x, y, z)$ is the pivot point on the end of the robot and $(\theta, \varphi)$ is an orientation the robot is placed.

ing AMs is addressed in our prior work [6].) As a preliminary task, the AM of all discretized points on the surface of the object are calculated. The optimization task is how to specify an orientation for each point to construct a valid path. Calculating an accurate AM for each point is expensive because the object itself changes during milling. Instead, we calculate the AM of all points only once in a parallel way at beginning of each step of the CPP problem. This approach means our collision-avoidance is conservative: if an orientation has no collision with the original object, we can guarantee that the orientation is safe with the current object undergoing the milling process, because it is a shrunken version of the original.

### C. Constraint of avoiding collisions

This paper targets at the optimization of the cost of dealing with the constraint that the path has to be collision-free, where the robot cannot smoothly move from one point to another without considering its orientation. Even if specifying an accessible orientation, we cannot assume that the orientation can be changed abruptly. Instead, there is a time-cost when changing an orientation to another. Note that we do not concern any joint limits or other kinematic constraints. The purpose of constraint-handling in our particular problem is to have smooth, continuous movement from one point to another. Three possible situations may occur when moving from one point to another.

- 1) Given that the current orientation is accessible at the next point, we directly conduct the movement without worrying about the orientation.
- 2) Given that the current orientation becomes inaccessible at the next point but there exists another accessible orientation on both the next and the current point, we demand a *reorientation*, which is a process that reorients the robot to another accessible orientation and then moves. An example appears in the bottom right of Figure 3.

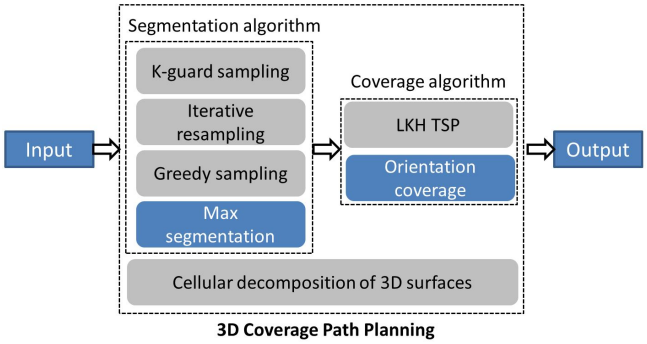


Fig. 4: Candidate ways to solve the 3D path planning problem. Our proposed methods are highlighted in blue color.

- 3) Given that the current orientation becomes inaccessible at the next point and there is no accessible orientation shared between the two points, we demand a *retraction*, which is a process that pulls back the robot to a place far away from the object, reorients to an accessible orientation, and resumes, shown in the bottom left of Figure 3. The point needs to stay far away from the object so that any orientation is collision-free. This operation has three steps for a smooth transition: pull back, reorient, and then push in.

### D. Problem statement

Our problem is to construct an efficient path for a robot that completely covers (physically touches) all points on the surface of a given 3D object (e.g., the head of Figure 1) and guarantee no collisions. The robot is a 5-axis CNC milling tool. Figure 3 shows the input and the output. The focus of this paper is on formulating and solving the optimization problem.

### E. Define the cost function

We define the cost as the machining time, so that an efficient path is one that covers an object quickly. The machining time is assumed to be proportional to the path length. The time of applying the two operations varies significantly, which depends on specifics of the robot, such as how to move and how to reorient. We specify a range as shown in Section V for their cost by normalizing it with the path length. For example, if the cost parameters $(c_1, c_2)$ are (10, 50)mm, it means that each reorientation consumes the time of the robot moving 10mm and each retraction consumes the time of the robot moving 50mm. In the equation below, $\mathrm{num}_1, \mathrm{num}_2$ denote the number of reorientation and the number of retraction respectively. $\alpha$ is a constant (second/mm) determined by the milling robot. Therefore, the optimization goal is to minimize the cost to achieve a fast coverage.

$$\begin{aligned} \mathrm{Mach\_Time} &= \alpha * [\mathrm{Path\_Len} + \mathrm{Cost}(\mathrm{constraints})] \\ &= \alpha * (\mathrm{Path\_Len} + c_1 * \mathrm{num}_1 + c_2 * \mathrm{num}_2) \end{aligned}$$

With the problem defined, Figure 4 outlines candidate solution methods. Most approaches have two stages: running a segmentation algorithm and a coverage algorithm. For

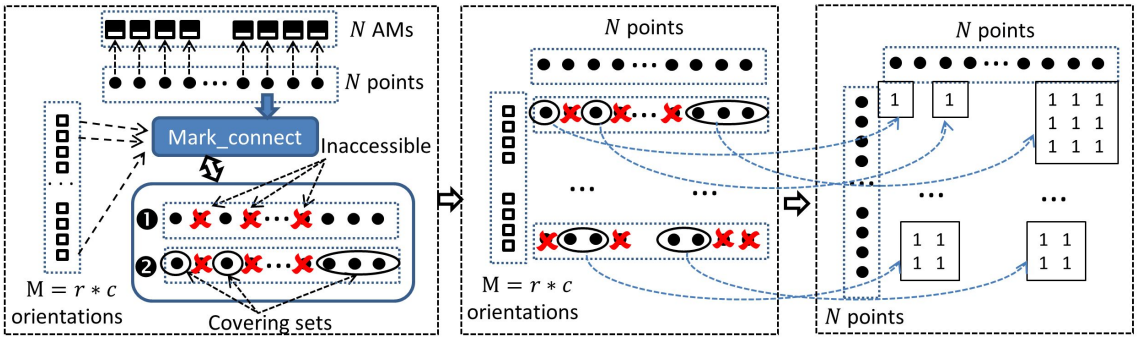


Fig. 5: Construct a graph $N * N$ matrix embedding all candidate covering sets to cover all $N$ points on the surface of the 3D object, where $M$ orientations correspond to the $r * c$ orientation points in AM.

the segmentation algorithm, the K-guard sampling method randomly samples $K$ guards and chooses the one that covers the most points [10]. Iterative sampling chooses points that shorten the path in an iterative way [4]. Greedy sampling finds the points that cover the most points until all points are covered [25]. For the coverage algorithm, Lin-Kernighan heuristic (LKH) TSP constructs a tour covering all the point for the purpose of the path length reduction [10], [4], [12], [5], [25]. Three-dimensional cellular decomposition intersects a slice plane with the object surface to form a loop around the target object and then traces the loop to the next slice plane, until a full coverage is done [14], [15].

## IV. Proposed Approach

Our approach has two components: max segmentation used to reduce the number of segmentation and orientation coverage used to lower the cost of the retraction and the reorientation by sacrificing the path length.

### A. Max segmentation

Motivated by the high cost of the two operations, our segmentation algorithm aims to minimize the total number of viewpoints. We define a set as a series of connected points that can be covered by a single orientation. An orientation on a point is associated with a set, where the robot can cover all the points in this set under this orientation. Our max segmentation algorithm divides all points into sets.

In the process of choosing the sets, we apply the MVC algorithm [19] to reduce the number of sets while achieving a complete coverage, instead of sampling the point or sampling the orientation. Figure 5 illustrates how to construct a graph matrix, where the vertices are the $N$ points and the edges represent the candidate sets. Firstly, a function called "Mark_connect" is used to calculate all candidate sets. Given an orientation, all the points and AM, it has two steps: mark the accessible points under the specified orientation and form a set if the points are connected as neighbors. The outputs are the candidate sets as shown in the middle of Figure 5. Each orientation has a series of sets that can be covered under the orientation, and each point has a series of sets that can cover the point. Lastly, a graph matrix is created, where we let the points in the candidate sets all-connected. Note that each set has a single orientation that covers the points within the set.

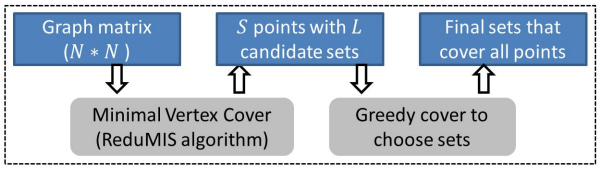


Fig. 6: Our max segmentation approach to choose a small number of covering sets to achieve a complete coverage.

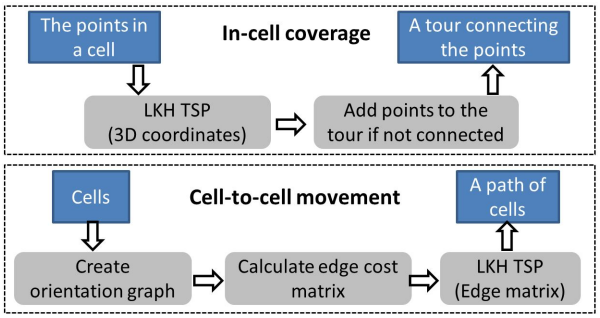


Fig. 7: Steps of in-cell coverage and cell-to-cell movement.

Figure 6 shows the steps of our max segmentation algorithm. The first step is to apply ReduMIS algorithm [19] on the graph matrix to choose $S$ points with a complete coverage. Note that each point is associated with multiple sets. We are sure that the $S$ points' $L$ candidates sets can achieve a complete coverage. The next is to do a greedy coverage that at every iteration, a set that covers the most points is chosen from the $L$ sets until all points are covered.

Compared with the random sampling segmentation algorithms, the benefits of our max segmentation come from two points: $S$ is much smaller than $N$ and $L$ is much smaller than the total number of the candidate sets. Note that if the robot does not need to set the orientation in some applications, the step of doing the greedy coverage can be removed.

### B. Orientation-based coverage

With the covering sets generated from the segmentation algorithm, our coverage algorithm needs to construct a path to cover the points in sets. The algorithm needs to answer two questions: how to cover the points in sets and how to move between sets. We create a graph for the cost reduction, called orientation graph, where each set corresponds to a vertex. For simplicity, the set is called cell from now on.

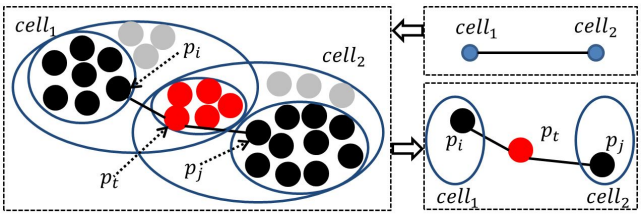


Fig. 8: Calculate the cost of an edge by mapping the edge to three points. Red points are the intersected points. $P_i$ and $P_j$ are the mapped two ends of the edge. $P_t$ is middle point where the robot does reorientation.

To answer the two questions, our coverage algorithm has two parts respectively: in-cell coverage and cell-to-cell movement. Figure 7 shows the steps of the two parts. On the first part, we use LKH TSP to construct a tour to reduce the path length of covering each cell. To avoid redundant coverage, each point on the object surface is assigned to one cell, which might lead to the points disconnected in a cell. Then the next step is to add points between two neighboring points in the tour if they are not connected. On the other part, we create orientation graph and use the cost of an edge to represent the cost of a movement from one cell to another. After all the costs of all possible edges are calculated, we apply LKH TSP to obtain an order of cells.

Figure 8 illustrates how to calculate the cost of an edge by mapping the edge to three points. We add an edge between two cells to the orientation graph as a possible movement, only when the two cells have intersected points. Among all the intersected points, the one that has the shortest distance to exit one cell and to enter the other cell is selected as the middle point between them. The middle point is the place the robot does the reorientation and move to another cell.

Figure 9 shows a real example of an edge connecting two cells in the head object. Due to the requirement of avoiding redundant coverage, each cell has two types of points: the actual points to cover in the current cell and the points already assigned to other cells. The middle point is chosen from the intersection of both types of points while the two ends points are from the actual points. The cost of the edge includes two parts: the sum of the two geodesic distances and the cost of a reorientation.

To further reduce the cost of moving between cells, we apply Floyd-warshall algorithm to calculate the edge cost matrix. Different from the normal calculation of the shortest distance in a graph, when trying to insert a new cell between two cells, the edge cost needs to add the cost of a path passing through the new cell. Lastly LKH TSP is employed to produce an order of cells that has a small cost of moving between cells.

As the order of the cells is determined, we know which edges will be used for the coverage. Through the mapping from the edge to the points, the two end points become the exit and the entrance points. Combining with the path of covering each cell, we can form a 5-axis path.

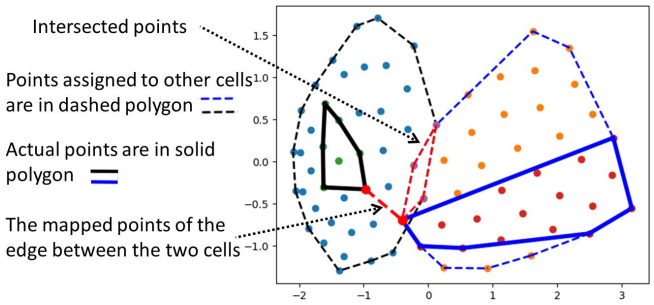


Fig. 9: An actual example of mapping an edge to points in the head CAD benchmark. All the points are projected to 2D using principal component analysis (PCA). The middle point and one end happen to be located at one point.

## V. Simulation Studies

The experimental validation of our approach mainly answers the following three questions: (1) how our max segmentation algorithm performs on the number of sets? (2) how our orientation coverage algorithm behaves on the cost of the two operations? (3) how our proposed method performs on the total cost of 5-axis paths?

**Implementation of candidate algorithms**. We evaluate the candidate algorithms shown in Figure 4. For the segmentation algorithm, K-guard sampling and greedy sampling are implemented for comparison. Because the key idea of the iterative resampling [4] is to reduce the distance among all sets with no need to cover each point in the sets, it does not work for our scenario. Thus the iterative resampling is not presented. In our problem setting, the sampling metric has two dimensions as shown in the middle part of Figure 5: the $N$ points and the $M$ orientations. We implemented K-guard on the orientations and greedy sampling on the points. K-guard algorithm samples $K$ sets as candidate sets at each iteration, and then choose the one that covers the most points, until all points are covered. Greedy algorithm samples a point from the uncovered points and choose the orientation that cover the maximal number of points.

For the coverage algorithm, LKH TSP is our baseline. We also use LKH TSP for the in-cell coverage. The main comparison is on how to construct a path for the cell-to-cell movement. Firstly we calculate the mean coordinates of the points within a cell as the center to represent the cell. Secondly, LKH TSP is employed to generate a short tour passing through all the cells as the cell order. Lastly when combining the in-cell coverage with the cell order, the point in the next cell that is closest to the exit point in the current cell is considered as the next entry point.

For our coverage baseline algorithm, we focus on reducing the path length representing the 3D cellular decomposition approach [14], [15] as well. Regarding lowering the cost of the reorientation and retraction, we always choose an orientation that covers the most points for a given path.

**Benchmarks and configurations.** We use four CAD benchmark models for our evaluation as presented in Figure 10. The AM is generated form SculptPrint, a computer-aided manufacturing (CAM) application for producing CNC

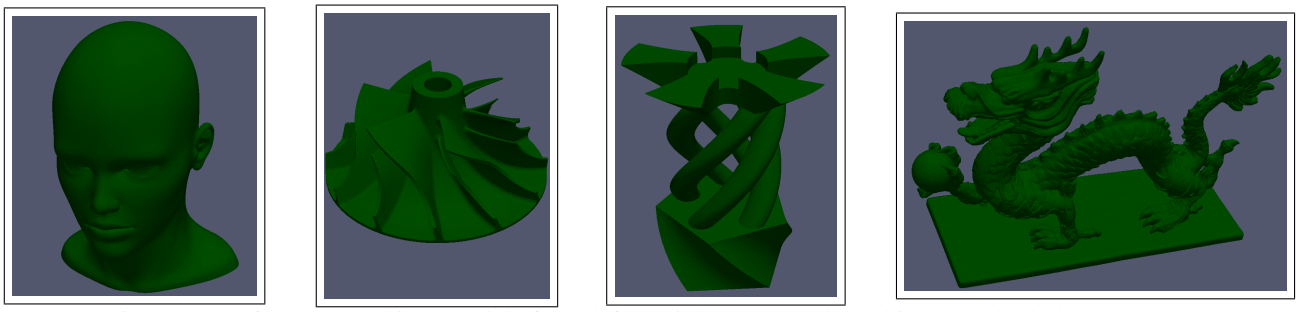

Fig. 10: Our four CAD object models from left to right are: Head, Turbine, Candle holder, Dragon.

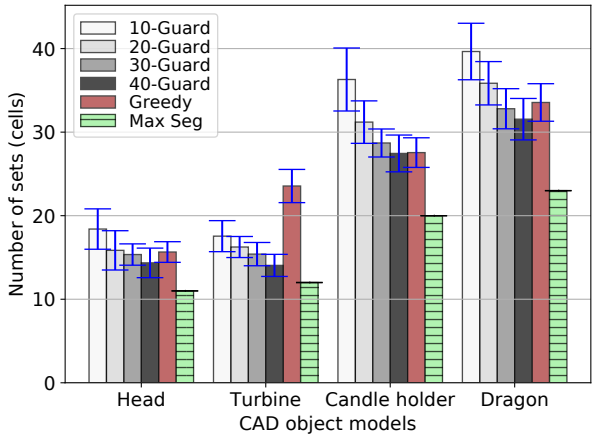


Fig. 11: Using various object models to evaluate segmentation algorithms.

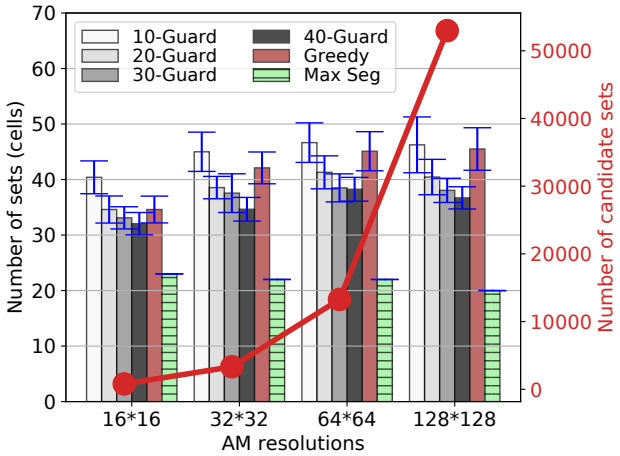


Fig. 12: Segmentation algorithm calculates the number of sets under various AM resolutions with the dragon object model.

tool paths [21], [6], with a tool composed of 4 cylinders, with varying radii (0.79, 1.59, 25, 31.5)mm and heights(2.28, 5.08, 78, 22)mm. For the K-guard sampling algorithm, the $K$ value is ranged from 10 to 40. Both sampling algorithms are run 20 times and report their mean and their std. Depending on the specific application and the robot, the reorientation and retraction cost may vary significantly. Thus we vary the cost of a reorientation and a retraction by normalizing them into the path length as a range from (10, 50)mm to (50, 250)mm. For simplicity, "Ret" and "Ort" are short for the cost of retractions and reorientations. "GreTSP", "GuaTSP", "MaxTSP" and "MaxOrt" are the four candidate algorithms, representing (greedy sampling + TSP), (K-guard sampling + TSP), (max segmentation + TSP) and (max segmentation + orientation coverage).

### A. *Number of sets in segmentation algorithms*

We report the total number of sets (cells) that cover the surface of the four objects. Figure 11 shows the results. We can see that the number of cells required for the K-guard sampling decreases as the $K$ value increases. Comparing the 40-guard sampling with the greedy sampling, they have a roughly same number of cells on the candle holder object, but on the turbine object, the greedy sampling behaves even worse than 10-guard sampling. It is uncertain which one is better because the random sampling is unstable. Our max segmentation gains a smaller number of cells on all four objects. Overall we can reduce the number of cells by 24.5% and 34.2% on average compared with 40-guard sampling and greedy-sampling respectively.

We increase AM resolutions leading to the growth of the number of accessible orientations on each point. Figure 12 shows the results on the number of sets and the number of candidate sets. The number of candidate sets does not grow exponentially as the AM resolution. This is because as the number of orientations raises, it is more likely that multiple orientations correspond to a set as a connected component. From the resolution $16^2$ to $64^2$, both the K-guard sampling and the greedy sampling gain an increasing number of sets, because it becomes less likely that the algorithm chooses the "correct" set as the total number of sets grows. On the resolution $128^2$, the number of cells does not increase because more "correct" sets are added for sampling. In contrast, our max segmentation is not negatively influenced by the AM resolution, but the number of sets decreases slightly. On average, it reduces the number of sets by 38.6% and 48.0% than 40-guard sampling and greedy-sampling respectively.

### B. *Coverage algorithms*

To evaluate how much our orientation coverage can contribute, we fix the max segmentation on the first step, but compare it again LKH TSP algorithm. Figure 13 shows the results of the two algorithms. Our orientation coverage does not have any retractions, because between any two points, we can find an accessible orientation for the movement.

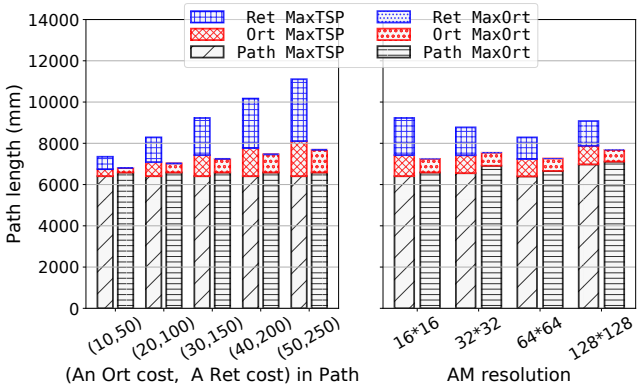


Fig. 13: Orientation coverage against LKH TSP with the dragon object model. On the left side, the cost of reorientation and retraction is converted into the path length. On the right side, the two costs are fixed to (30, 150).

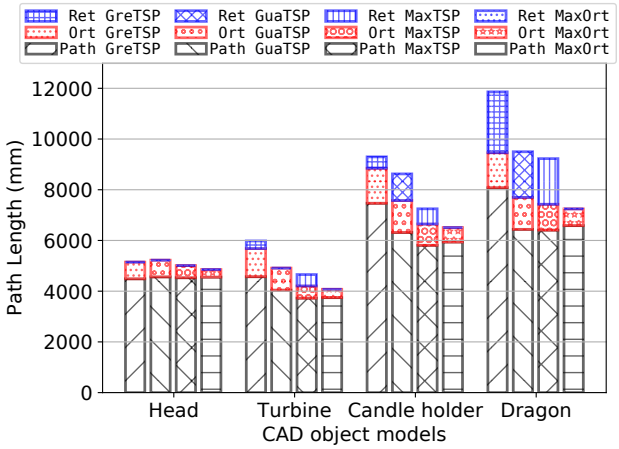


Fig. 14: 5-axis path cost of the four candidate algorithms on the four objects. The cost of a reorientation and a retraction is fixed to (30, 150)mm in path length.

Note that we can not guarantee no retraction if the input is a disconnected graph. Our algorithm has a slight increase, while the "MaxTSP" algorithm has a sharp rise as the cost of the two operations increases. On average, our orientation coverage reduces the total cost by 21.5%.

When varying the AM resolutions, our algorithm always has a smaller number of reorientation. while "MaxTSP" has an unstable cost of both operations, because a shorter path may require a high cost of reorientation and retraction. On average, our algorithm only increase the path cost by 3.5%, and achieve a 15.9% cost reduction in total.

### C. Cost of 5-axis path

We report the total cost of the 5-axis paths generated by the four algorithms. "MaxTSP" algorithm gains a cost reduction of 16.79% and 4.14% on the path length than "GreTSP" and "GuaTSP", because our max segmentation obtains a smaller number of cells, where a larger number of cells leads to a more-likely redundant coverage. Also because of the smaller number of cells, "MaxTSP" decreases the cost of reorientation by 37.8% and 30.4% respectively than the two sampling algorithms. Compared with "MaxTSP", our "MaxOrt" achieves a cost reduction of 67.2% on the cost of the two operations. Figure 15 and Figure 16 present the 5-axis path generated by "GreTSP" and our "MaxOrt" algorithms. For the dragon object, "MaxOrt" performs 38.9% better. The improvement goes up to 46.6% if the two costs are (50, 250)mm. Overall, the path generated by our "MaxOrt" is 29.7% more efficient than "GreTSP" on average.

## VI. Conclusion and Future Work

Focusing on how to avoid collisions when constructing an efficient path for a coverage, our proposed max orientation coverage has two key ideas: a segmentation algorithm called max segmentation Section IV-A and a coverage algorithm called orientation coverage Section IV-B. Our experimental results on 4 CAD benchmarks demonstrate that our method can generate an efficient 5-axis path on both the cost of the path length and the cost of avoiding collisions, and it can improve the path efficiency by up to 46.6% Compared with a state-of-the-art baseline. For future work, we plan to extend our method to online algorithms that require producing fast and efficient paths, because our current work only targets on offline applications that allow several minutes of execution time. We will explore the value of incorporating more special operations, such as considering how to enter and exit a cell and the effects of speed-up and slow-down.

## References


[1] Ercan U Acar, Howie Choset, and Ji Yeong Lee. Sensor-based coverage with extended range detectors. *IEEE Transactions on Robotics*, 22(1):189–198, 2006.

[2] Kostas Alexis, Georgios Darivianakis, Michael Burri, and Roland Siegwart. Aerial robotic contact-based inspection: planning and control. *Autonomous Robots*, 40(4):631–655, 2016.

[3] Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, Jorge Dias, and Guowei Cai. A survey on inspecting structures using robotic systems. *International Journal of Advanced Robotic Systems*, 13(6):1729881416663664, 2016.

[4] Andreas Bircher, Kostas Alexis, Michael Burri, Philipp Oettershagen, Sammy Omari, Thomas Mantel, and Roland Siegwart. Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6423–6430. IEEE, 2015.

[5] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1462–1468. IEEE, 2016.

[6] Xin Chen, Dmytro Konobrytskyi, Thomas M Tucker, Thomas R Kurfess, and Richard W Vuduc. Faster parallel collision detection at high resolution for cnc milling applications. In *Proceedings of the 48th International Conference on Parallel Processing*, page 97. ACM, 2019.

[7] Inhaeng Cho, Kunwoo Lee, and Jongwon Kim. Generation of collision-free cutter location data in five-axis milling using the potential energy method. *The International Journal of Advanced Manufacturing Technology*, 13(8):523–529, 1997.

[8] Chih-Hsing Chu and Jang-Ting Chen. Tool path planning for five-axis flank milling with developable surface approximation. *The International Journal of Advanced Manufacturing Technology*, 29(7-8):707, 2006.

[9] Chih-Hsing Chu, Way-Nen Huang, and Yu-Wei Li. An integrated framework of tool path planning in 5-axis machining of centrifugal impeller with split blades. *Journal of Intelligent Manufacturing*, 23(3):687–698, 2012.

[10] Tim Danner and Lydia E Kavraki. Randomized planning for short inspection paths. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 971–976. IEEE, 2000.

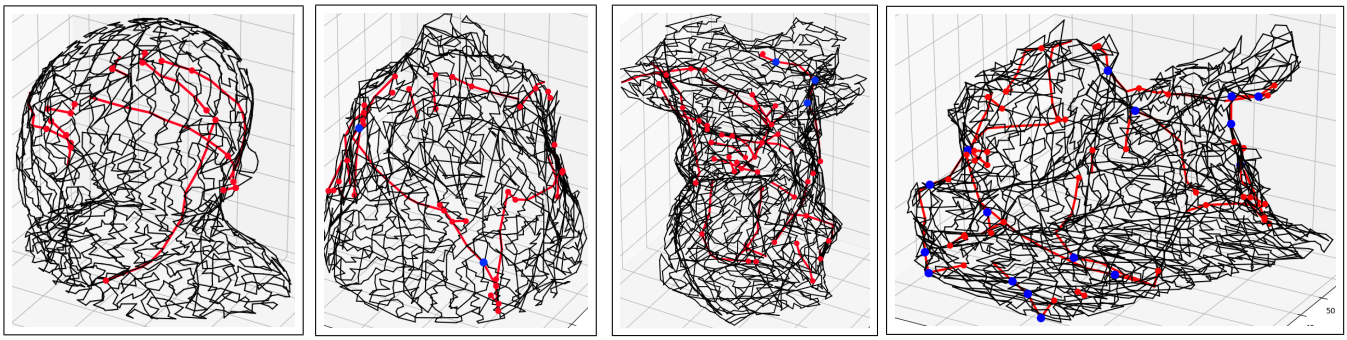

Fig. 15: Path generated by "GreTSP". The black lines represent the path of covering cells. The red lines represent the path of cell-to-cell movement. The red point denotes the point that requires a reorientation. The blue point means a retraction.

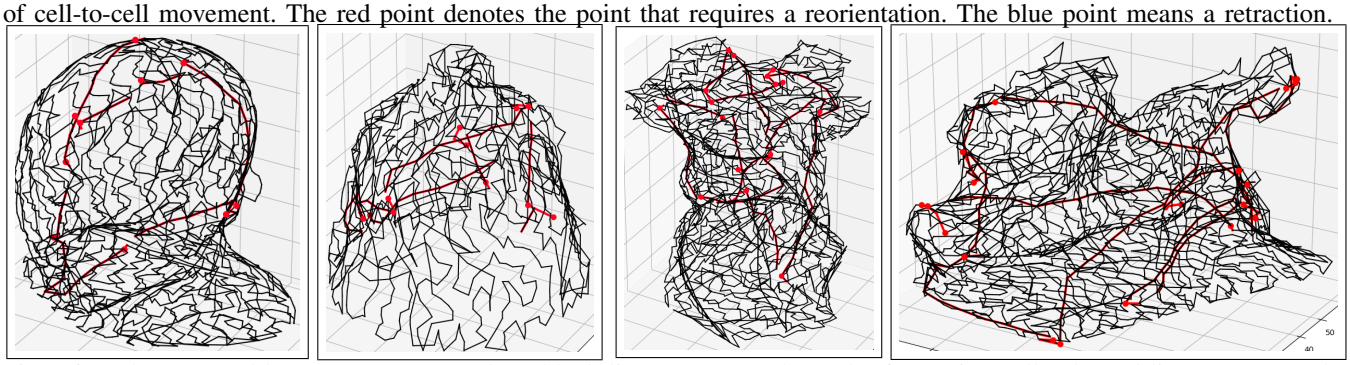

Fig. 16: Path generated by our "MaxOrt", where black lines represent the path of covering cells, the red lines represent the path of cell-to-cell movement. There is no retraction and only reorientation.


[11] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[12] Brendan Englot and Franz Hover. Planning complex inspection tasks using redundant roadmaps. In *Robotics Research*, pages 327–343. Springer, 2017.

[13] Yoav Gabriely and Elon Rimon. Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 1, pages 954–960. IEEE, 2002.

[14] Prasad N Atkar, Howie Choset, Alfred A Rizzi, and Ercan U Acar. Exact cellular decomposition of closed orientable surfaces embedded in/spl rfr//sup 3. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 699–704. IEEE, 2001.

[15] Prasad N Atkar, Aaron Greenfield, David C Conner, Howie Choset, and Alfred A Rizzi. Hierarchical segmentation of surfaces embedded in r3 for auto-body painting. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 572–577. IEEE, 2005.

[16] Enric Galceran and Marc Carreras. Planning coverage paths on bathymetric maps for in-detail inspection of the ocean floor. In *2013 IEEE International Conference on Robotics and Automation*, pages 4159–4164. IEEE, 2013.

[17] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12):1258–1276, 2013.

[18] Héctor González-Banos. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240. ACM, 2001.

[19] D. Hespe, C. Schulz, and D. Strash. Scalable Kernelization for Maximum Independent Sets. In *Proceedings of the 20th Meeting on Algorithm Engineering and Exerpimentation (ALENEX'18)*, pages 223–237, 2018.

[20] Yao-Wen Hsueh, Ming-Hsien Hsueh, and Hsin-Chung Lien. Automatic selection of cutter orientation for preventing the collision problem on a five-axis machining. *The International Journal of Advanced Manufacturing Technology*, 32(1-2):66–77, 2007.

[21] Tucker Innovations Inc. Sculptprint. `http://www.sculptprint.com`. Accessed: 2019-09-30.

[22] Cha-Soo Jun, Kyungduck Cha, and Yuan-Shin Lee. Optimizing tool orientations for 5-axis machining by configuration-space search method. *Computer-Aided Design*, 35(6):549–566, 2003.

[23] Chaomin Luo, Simon X Yang, Deborah A Stacey, and Jan C Jofriet. A solution to vicinity problem of obstacles in complete coverage path planning. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 1, pages 612–617. IEEE, 2002.

[24] Joseph O'rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

[25] Narcís Palomeras, Natalia Hurtós, Marc Carreras, and Pere Ridao. Autonomous mapping of underwater 3-d structures: From view planning to execution. *IEEE Robotics and Automation Letters*, 3(3):1965–1971, 2018.

[26] Georgios Papadopoulos, Hanna Kurniawati, and Nicholas M Patrikalakis. Asymptotically optimal inspection planning using systems with differential constraints. In *2013 IEEE International Conference on Robotics and Automation*, pages 4126–4133. IEEE, 2013.

[27] Renato Tinós, Keld Helsgaun, and Darrell Whitley. Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic. In *International Conference on Parallel Problem Solving from Nature*, pages 95–107. Springer, 2018.

[28] Nan Wang and Kai Tang. Automatic generation of gouge-free and angular-velocity-compliant five-axis toolpath. *Computer-Aided Design*, 39(10):841–852, 2007.

[29] Jeremy A Carter, Thomas M Tucker, and Thomas R Kurfess. 3-axis cnc path planning using depth buffer and fragment shader. *Computer-Aided Design and Applications*, 5(5):612–621, 2008.

[30] Joshua A Tarbutton, Thomas R Kurfess, and Tommy M Tucker. Graphics based path planning for multi-axis machine tools. *Computer-Aided Design and Applications*, 7(6):835–845, 2010.

[31] Roby Lynn, Didier Contis, Mohammad Hossain, Nuodi Huang, Tommy Tucker, and Thomas Kurfess. Voxel model surface offsetting for computer-aided manufacturing using virtualized high-performance computing. *Journal of Manufacturing Systems*, 43:296–304, 2017.

[32] Dmytro Konobrytskyi, Mohammad M Hossain, Thomas M Tucker, Joshua A Tarbutton, and Thomas R Kurfess. Five-axis tool path planning based on highly parallel discrete volumetric geometry representation: Part I, contact point generation. *Computer-Aided Design and Applications*, 15(1):76–89, 2018.