Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea • May 21-26, 2001

Exact Cellular Decomposition of Closed Orientable Surfaces Embedded in $\Re^3$

Prasad N. Atkar Howie Choset Alfred A. Rizzi Ercan U. Acar
Carnegie Mellon University
Department of Mechanical Engineering and Robotics Institute
Pittsburgh, PA 15213

Abstract

We address the task of covering a closed orientable surface embedded in $\Re^3$ without any prior information about the surface. For applications such as paint deposition, the effector (the paint atomizer) does not explicitly cover the target surface, but instead covers an ***offset surface*** *— a surface that is a fixed distance away from the target surface. Just as Canny and others use critical points to look for changes in connectivity of the free space to ensure completeness of their roadmap algorithms, we use critical points to identify changes in the connectivity of the offset surface to ensure full surface coverage. The main contribution of this work is a method to construct unknown offset surfaces using a procedure, also developed in this paper, to detect critical points.*

1 Introduction

Conventional path planning determines a path between two points in the free configuration space $\mathcal{S}$ of a system [14]. Recent path planning results describe *coverage path planning* algorithms that determine a path that enables an "effector" to pass over all points in the free configuration space [8], [12], [21]. Applications of this recent work include humanitarian de-mining, autonomous lawn mowing and floor cleaning, all of which are *planar* coverage tasks. This paper takes the first step towards lifting these planar coverage algorithms into three dimensions for applications such as the inspection of complicated surfaces (non-planar), material deposition, material removal, and CNC tool path planning. These applications require coverage of two-dimensional surfaces embedded in three dimensions. For applications such as visual inspection, the camera itself does not cover the target surface; to achieve visual coverage, the camera covers an *offset surface,* a surface that is a fixed distance away from the target surface. This paper describes how a robot equipped with a range sensor at its end effector, can incrementally construct (i.e., explore) an offset surface without any prior information about the target surface.

Our approach to coverage uses a *slice*, a co-dimension one surface (i.e., a plane), that is swept through the free space. We are interested in slices where topological (e.g., connectivity) changes occur; these changes occur at points termed *critical points.* Just as Canny [3], [4], [5] uses critical points to ensure the connectivity of roadmaps, we use critical points to define cells in a cellular decomposition. In each cell, simple motions are sufficient to cover each cell and then *complete* coverage is achieved by visiting each cell in the decomposition. In this paper, we define a cellular decomposition for a large class of two-dimensional surfaces embedded in three dimensions in terms of critical points on the offset surface. To achieve sensor based coverage of the offset surface, we introduce a new method to detect critical points on the offset surface. This method assumes that the robot can measure distance from its effector to the target surface.

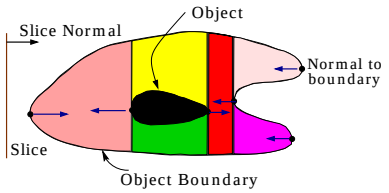

Fig. 1. C llular d composition for flat planar nvironm nts.

2 Prior Work

Our work rests on two tasks: coverage and generating offset surfaces. Most recent work in coverage is geared towards the plane and is either implicitly or explicitly, based on exact cellular decompositions. These algorithms decompose the space into different shapes such as rectangles [2], trapezoids [18], [19], "clumped trapezoids" [6], [8] or fine cells (grids) [10], [20], [21].

Choset et al [7] present a method to achieve an exact cellular decomposition of a planar environment that is formulated in terms of critical points. Between critical points, the number of connected portions of the slice in the free space remains the same. This means that each cell in the decomposition can be covered with simple back and forth motions and complete coverage is achieved by visiting each cell. Acar et al [1] sense critical points using distance information and show that the distance function gradient becomes perpendicular to the slice at the critical points (see Figure 1). We use an analogous approach to detect critical points on the offset surfaces.

Surveys [15], [17] of offset curves/surfaces literature indicate that there has been a vast amount of research in generating offset curves; however this field is still young. Farouki [9] gives procedures to determine a geometric representation for offset surfaces of simple solids, but these methods do not consider offset surfaces of concave objects or curved surfaces. Pham [16] describes methods to generate approximations of the offset surface for the NURB surface, but the method does not yield offset surfaces free from self intersections, discontinuities, and sharp ridges. There have been a few attempts to eliminate self intersections using non-analytical approaches. Kimmel and Bruckstein [13] use the wavefront approach in fluid dynamics to obtain the offset surface, while Gurbuz and Zeid [11] employ the approach of filling closed balls of a fixed radius at each point on the object. However, both methods are based on grid/cell decompositions; hence, their resolution and accuracy greatly depend upon the size of the grid. Since our work uses numerical tracing to construct the "sliced" offset surface of an arbitrarily shaped object, it is free from self intersections and discontinuities.

3 Offset Surface Coverage

A coverage offset surface has two dimensions and is embedded in $\Re^3$. We determine a path that covers this surface by repeatedly intersecting it with a two-dimensional slice. Each intersection generically is one or more loops; we term these loops as *coverage offset surface edges*, or COS_{edges}.

3.1 Coverage Offset Paths

We use a numerical technique that traces the roots of a function to generate the COS_{edge}. This function has two parts: an offset surface component and a slice component. Let x be a point in $\Re^3$. The distance between x and an object C_i is given by $d_i(x)$ ($d_i: \Re^3 \mapsto \Re$). The coverage offset surface, $\mathcal{COS}_i$ is

$$\mathcal{COS}_i = \{x \in \Re^3 : d_i(x) - \Omega = 0\}, \qquad (1)$$

where $\Omega \in \Re$ is the desired fixed distance. By the pre-image theorem, $\mathcal{COS}_i$ is dimension two. Now, we intersect $\mathcal{COS}_i$ with a planar slice $\Sigma_{i,k} = \{x \in \Re^3 : \langle n_s, x\rangle - k = 0\}$, where $n_s \in \Re^3$ is the normal to the plane, and k defines the location of the plane. Varying k has the effect of sweeping the slice. The intersection of slice plane $\Sigma_{i,k}$ and $\mathcal{COS}_i$ defines the COS_{edge}, which can be represented by the pre-image of a function $G_{i,k,\Omega}: \Re^3 \mapsto \Re^2$,

$$G_{i,k,\Omega}(x) = \begin{pmatrix} d_i(x) - \Omega \\ \langle n_s, x\rangle - k \end{pmatrix}. \qquad (2)$$

For concise notation, let $G(x)$ be $G_{i,k,\Omega}(x)$, and let $\mathcal{COS}$ denote $\mathcal{COS}_{\cup_i C_i}$. By the pre-image theorem, we know that for regular values, the pre-image of $G(x)$ is a one-dimensional manifold. We have shown, but omitted its proof due to space limitations, that away from critical points and assuming a generic placement of objects, the offset edges form closed loops.

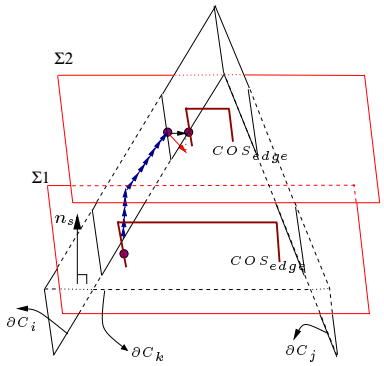

Fig. 2. Switching slic plan algorithm.

Once the planner traces the COS_{edge} loop, it must then shift to the next slice plane. Changing slice planes is not a trivial task. We have developed an algorithm that either determines a point on the COS_{edge} in the next slice plane or concludes that there is no COS_{edge} in the next slice corresponding to the COS_{edge} in the current plane. The algorithm has the following two steps:

1. Shifting the slice planes: The planner moves a point along the slice normal, until it reaches the next slice plane, or until it reaches the boundary of an object. In the latter case, the planner then starts moving along the boundary while increasing the distance to the current slice. In other words, it follows the slice gradient projected onto the surface boundary. See Figure 2. While following the boundary, if the planner comes across a point on the object surface such that it cannot move away from the previous slice, then the planner has reached a critical point on the object.
2. Moving onto a COS_{edge} or detecting that it does not exist: Once in the new slice plane, the planner moves a point away from the closest object while keeping the point in the plane, i.e., it follows $\Pi_\Sigma \nabla d_i(x)$, until it reaches a point on the COS_{edge} or a point on a two way equidistant sheet $SS_{ij} = \{x : d_i(x) = d_j(x), \nabla d_i(x) \neq \nabla d_j(x)\}$. In the latter case, the planner traces $SS_{ij} \cap \Sigma$ looking for a point on the COS_{edge}. After tracing the intersection completely, if the planner does not find any COS_{edge} point, then it concludes the absence of COS_{edge} in the new slice, corresponding to COS_{edge} in the old slice. This conclusion represents that the planner has passed a critical point on the offset surface lying between the new and the old slice. Such a critical point on $\mathcal{COS}$ appears when there is a corresponding critical point on the ob-

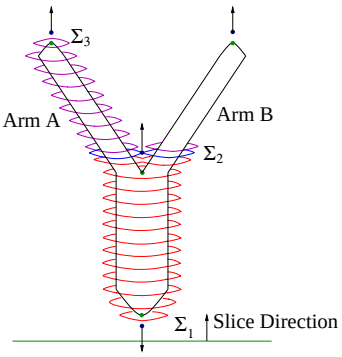

Fig. 3. Multipl loops in a singl slic plan

ject, or when the object has a narrow "neck" which splits the offset surface into two disconnected parts.

3.2 Critical Points and the Adjacency Graph

For convex objects, there is at most one COS_{edge} in any slice plane; hence, repeatedly tracing a loop and then finding a seed point for the next loop in the next slice plane will cover the $\mathcal{COS}$ completely. However, for solids that are not convex, there may be more than one COS_{edge} loop in a single slice plane, therefore, the planner needs to know the number of COS_{edges} in a given slice plane to achieve complete coverage. In Figure 3, as we sweep the slice from bottom to top, at slice Σ_2, the number of COS_{edge} loops in a slice change from one to two. If the planner naively adopts the "trace the loop and jump to next slice" policy, it covers only one "arm" and fails to cover the other arm.

We use critical points on the offset surface to determine when the number of COS_{edge} loops changes. The change depends upon the "convexity" nature of a neighborhood of an offset critical point. Let CP be a critical point on the offset surface , and let $B_\epsilon(CP)$ be the neighborhood of CP, which is an open ball with center CP and of radius $\epsilon > 0$. Since the offset surface is a space-dividing surface, let the volume bounded by the offset surface that contains the object be denoted by $\mathcal{SO}$. Then, the critical points are divided into three classes : convex, concave, and semi-concave.

DEFINITION 3.1 *A critical point CP is*

- ***convex*** *if $B_\epsilon(CP) \cap \mathcal{SO}$ is convex,*
- ***concave*** *if $B_\epsilon(CP) \cap (\ \mathcal{S} \setminus int(\mathcal{SO}))$ is convex,*
- ***semiconcave*** *if neither $B_\epsilon(CP) \cap \mathcal{SO}$ nor $B_\epsilon(CP) \cap (\ \mathcal{S} \setminus int(\mathcal{SO}))$ is convex.*

The critical points correspond to nodes in our adjacency graph; edges correspond to cells whose boundaries are defined by two "adjacent" critical points. In this work, the planner incrementally constructs the adjacency graph by first covering a cell until it detects a critical point. The convexity nature of the critical point determines how many cells are associated with the critical point. In the next section, we show that a convex or concave critical point corresponds to a terminal node (a leaf) in the adjacency graph and a semi-concave critical point corresponds to a node that generically has three edges emanating from it. If the planner encounters a semi-concave critical point, the planner chooses one cell and covers it until the planner encounters another critical point. When the planner encounters a convex or concave critical point, then it returns to a semi-concave critical point with an "uncovered" cell associated with it. When all critical points have no uncovered cells, coverage is complete.

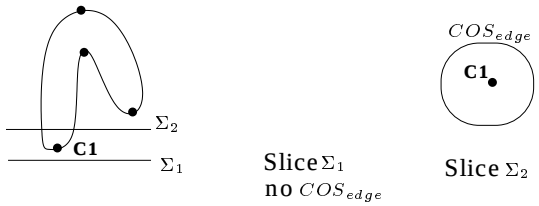

Fig. 4. Conv x critical point: 0-to-1 conn ctivity chang

The challenge then becomes how to detect a critical point and what its type is.

4 Detecting the Critical Points

As we pass the critical point while covering $\mathcal{COS}$, *generically* only four kinds of local connectivity changes are possible: 0-to-1 (the number of loops changes from 0 to 1 as we pass the critical point), 1-to-0, 1-to-2 or 2-to-1.

Note that a slice is defined by the preimage of a scalar-valued function $\lambda: \Re^3 \mapsto \Re$. $\lambda^{-1}(c)$ is a slice, and $\lambda(x)$ denotes the value of the slice function evaluated at a point x on the $\mathcal{COS}$.

LEMMA 4.1 *At a convex critical point X, there is always a 0-to-1 or 1-to-0 change in the number of COS_{edge} loops (see Figure 4).*

Proof: By definition, $B_\epsilon(X) \cap \mathcal{SO}$ is convex. Since X is a critical point, it must be either a local minimum or maximum of the slice function evaluated on the offset surface. Since λ is a convex function and $B_\epsilon(X) \cap \mathcal{SO}$ is a convex set, $\forall x1, x2 \in (B_\epsilon(X) \cap \mathcal{SO})$, such that $x1, x2 \neq X$, either $\lambda(x1) < \lambda(X)$ and $\lambda(x2) < \lambda(X)$, or $\lambda(x1) > \lambda(X)$ and $\lambda(x2) > \lambda(X)$. Since $\mathcal{COS} = \partial\mathcal{SO}$, the same is true for all $x1, x2 \in B_\epsilon(X) \cap \mathcal{COS}$. Thus, there does not exist $x1, x2 \in (B_\epsilon(X) \cap \mathcal{COS})$ such that $\lambda(x1) > \lambda(X)$ and $\lambda(x2) < \lambda(X)$. Hence, all points in $B_\epsilon(X) \cap \mathcal{COS}$ must lie on one side of the slice that contains X, and there is no point in the $B_\epsilon(X) \cap \mathcal{COS}$ which lies on the other side of the critical slice. Therefore, there must be a 0-to-1 or 1-to-0 change at the convex critical point. ■

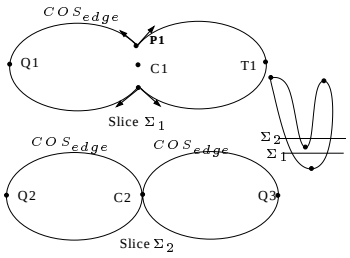

Fig. 5. S miconcav critical point: 2-to-1 conn ctivity chang

LEMMA 4.2 *At a concave critical point Y, there is always a 0-to-1 or 1-to-0 change in the number of COS_{edge} loops.*

Proof: Here, by definition, $B_\epsilon(Y) \cap (\ \mathcal{S} \setminus int(\mathcal{SO}))$ is convex. Again, $\mathcal{COS}$ is the boundary of a convex set $B_\epsilon(Y) \cap (\ \mathcal{S} \setminus int(\mathcal{SO}))$. The proof follows *mutatis mutandis*, as per Lemma 4.1. ■

LEMMA 4.3 *There is a 1-to-m or m-to-1 change in the number of COS_{edge} loops at a semiconcave critical point Z, where $m > 1$ (see Figure 5).*

Proof: For a semiconcave critical point, both $B_\epsilon(Z) \cap \mathcal{SO}$ and $B_\epsilon(Z) \cap (\ \mathcal{S} \setminus int(\mathcal{SO}))$ are not convex. Their intersection, $(B_\epsilon(Z) \cap \mathcal{SO}) \cap (B_\epsilon(Z) \cap (\ \mathcal{S} \setminus int(\mathcal{SO}))$ is $B_\epsilon(Z) \cap \partial\mathcal{SO}$, which is the same as $B_\epsilon(Z) \cap \mathcal{COS}$. Hence, $\exists z1, z2 \in B_\epsilon(Z) \cap \mathcal{COS}$ such that $\lambda(z1) > \lambda(Z)$ and $\lambda(z2) < \lambda(Z)$. Thus, if $z1$ lies on one side of the slice plane that contains Z, then $z2$ must lie on the other. Without loss of generality, if there is only one COS_{edge} on the side where $z1$ lies, since there is a change in connectivity at the critical point, there must be $m(> 1)$ number of COS_{edge} loops on the $z2$ side. Hence, there must be a 1-to-m or m-to-1 change at the semiconcave critical point. A 1-to-m or m-to-1 connectivity change can be seen as $m-1$ number of 1-to-2 or 2-to-1 connectivity changes at the same critical point. In this paper, we will always treat the 1-to-m change as $m-1$ 1-to-2 changes and vice-versa. ■

Now, we present methods to detect critical points on a closed, orientable and *connected* offset surface. When the planner is covering an unknown environment, only detecting a 1-to-0 connectivity change for the convex and concave critical points suffices. A 1-to-0 change is automatically detected by the switching slice plane algorithm. However, the 1-to-2 change at a semiconcave critical point is not very apparent to the planner. The planner uses cusps, or the non-smooth boundary points on the COS_{edge} where a discrete change occurs in the direction of the tangent to the COS_{edge}, to determine the semiconcave critical point. The planner looks for the cusps in the neighborhood of the critical point, and then traces the cusps to reach the critical point. Here,

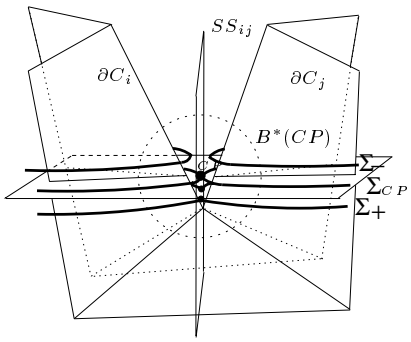

Fig. 6. S miconcav critical point manat s cusps.

we consider a *deleted* neighborhood $B^*_\epsilon(Z) = B_\epsilon(Z) \setminus Z$ of the critical point Z.

LEMMA 4.4 *For every semiconcave critical point CP, there exists a cusp in its deleted neighborhood.*

Proof: From Lemma 4.3, we know that there is a 1-to-2 or 2-to-1 change in the number of COS_{edge} loops at a semiconcave critical point. Therefore, there exists a "critical slice" where two loops intersect non-transversely (i.e. the loops kiss each other). See Figure 6. Let Σ_{CP} denote the critical slice. Since we have assumed that a critical point is isolated, the loops intersect only at one point, the critical point. Clearly, both loops have at least one different convex object closest to them. Let C_i and C_j be the closest objects at the critical point. Then at the critical point, the distance of point CP from C_i and C_j is the same and is equal to Ω. Then, the equidistant sheet SS_{ij} passes through CP and locally splits the $\mathcal{COS}$. Note that the double equidistant sheet intersects the $\mathcal{COS}$ only on the "one loop side" of CP. Also, this intersection is one-dimensional and it locally separates the $\mathcal{COS}$. Let Σ_- be the slice plane with a slice value smaller than that of Σ_{CP}. Similarly, let Σ_+ have a larger slice value than that of Σ_{CP}. Without loss of generality, let the number of COS_{edge} loops change from 2 to 1 as the planner moves from Σ_- to Σ_+. Then, by continuity of the slice function, there exists a slice Σ_ϵ whose slice value is ϵ greater than that of Σ_{CP} such that $\Sigma_\epsilon \cap \mathcal{COS}$ is separated by $SS_{ij} \cap \mathcal{COS}$. Thus, SS_{ij} separates the COS_{edge} in slice Σ_ϵ into two parts (both parts exclude $COS_{edge} \cap SS_{ij}$) such that points belonging to one part are closer to a set of objects $\mathcal{C}_{A\epsilon}$, while points in the other part are closer to a different set of objects $\mathcal{C}_{B\epsilon}$, i.e., $\mathcal{C}_{A\epsilon} \neq \mathcal{C}_{B\epsilon}$. Hence at $SS_{ij} \cap \mathcal{COS} \cap \Sigma_\epsilon = \{K_i\}$, there is a *discrete* change in the gradient vector.

From Equation 2, we know that $Null \begin{pmatrix} \nabla d(x) \\ n_s \end{pmatrix}$ is the tangent to the COS_{edge}. Since there is a discrete change in $\nabla d(x)$ at K_i, the tangent at K_i has a discrete change in direction. Thus, COS_{edge} is non-smooth or,

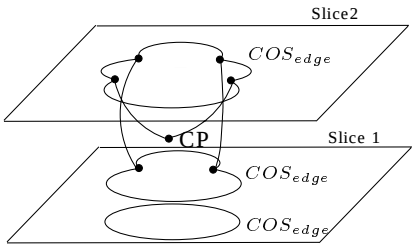

Fig. 7. D t cting th s miconcav critical point : tracing th cusps.

in other words, K_i is a cusp. This argument holds for sufficiently small values of ϵ, hence the proof. ■

Thus, there exists a one-dimensional path, $EquiCOS$, which is the intersection of the offset surface and the equidistant sheet, such that the cusp points and the critical point (when it exists) lie on it. To guarantee that the semiconcave critical point is detected, the planner traces all the $EquiCOS$ edges between the current and the previous slice planes. See Figure 7. Initially, it traces all $EquiCOS$ edges emanating from the cusps in previous slice, and some of these cusps lead to cusps in the current slice plane. It then traces the $EquiCOS$ edges from the remaining cusps in the current slice.

The $EquiCOS$ is the pre-image of the function $EC(x)$:

$$EC(x) = \begin{pmatrix} d_i(x) - \Omega \\ d_i(x) - d_j(x) \end{pmatrix}, \qquad (3)$$

where objects C_i and C_j are the first closest objects at the cusp. While tracing the $EquiCOS$, if at any point, the slice normal n_s lies in the convex hull of vectors $\nabla d_i(x)$ and $\nabla d_j(x)$ [8], then the planner has found the semiconcave critical point. If there is a 1-to-m change at this critical point, then m can be easily found by noting the starting point of each $EquiCOS$ edge in the previous slice. If these $EquiCOS$ starting points lie on n distinct COS_{edge} loops, then $m = n + 1$.

However, tracing all the $EquiCOS$ edges lying between the previous and current slice planes may be computationally very expensive. A very useful heuristic uses the change in the number of cusps between the current and previous slices. Note that it is necessary that the number of cusps changes in the neighborhood of a critical point, but it is not sufficient. So, when such a change occurs, the positions of the cusps in the current slice are compared with the position of the cusps in the previous slice. Only the "new" cusps are traced. A 2-to-1 connectivity change increases the number of cusps in the current slice plane, while 1-to-2 connectivity change decreases the number of cusps. Thus, the planner can detect all three types of critical points and, therefore, can complete the offset surface coverage by constructing the adjacency graph.

5 Simulation

We simulate the critical point detection procedure using known polyhedral environments. For polyhedral environments, we use critical points on the target surface – the boundary of the polyhedral solid – to determine the critical points on the offset surface. We classify the critical points on the polyhedral object similarly to the offset critical points: convex, concave or semiconcave. For non-degenerate cases, the critical points of the target surface appear only at the vertices of the polyhedron. It is easy to determine which vertices are critical points by looking at the positive span of the surface normals that form the vertex. If the slice gradient lies in this positive span, then the vertex is a critical point [8]. Each critical point is then "lifted" to the offset surface, but this "lifted" critical point is not necessarily a critical point on the offset surface. We term these points as candidate critical points. If the distance between the candidate critical point and the closest point on the target surface is the offset distance, then the candidate critical point is indeed a critical point for the offset surface. It is worth noting that for convex, semi-concave, and concave critical points, there will be one, two, and three closest points respectively. Finally, not all critical points on the offset surface are derived from the target surface; these critical points, however, are detected while executing the switching slice plane algorithm. By looking at the target surface and by invoking the switching plane algorithm, we are guaranteed to encounter all critical points of the offset surface and hence ensure complete coverage.

The simulations are carried out by generating the offset surfaces for different offset distances and different slicing directions. We verify that the simulation yields the locations of critical points exactly as predicted. Figure 8 shows the coverage of the offset surface with semiconcave and convex critical points. The simulation shows the 2-to-1 connectivity change at the semiconcave corner. Figure 9 shows an object with a hole in it. For this object, again there are semiconcave and convex critical points on its offset surface. Thus, these simulations successfully demonstrate the critical point detection procedure for offset surfaces.

6 Conclusion

In this work, we introduce complete methods to cover an unknown closed, orientable and connected offset surface in $\Re^3$. The offset surface is the set of points that are a fixed distance away from a target surface (such as an automobile body). The offset surface is covered by incrementally tracing several paths on the offset surface using local numerical techniques. These paths are formed by repeatedly intersecting a slice with the cov-

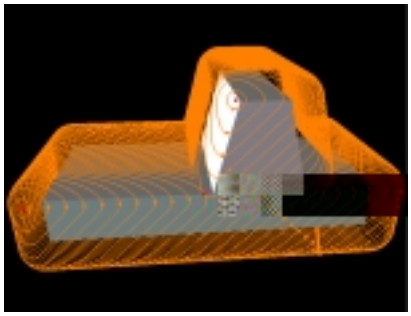

Fig. 8. Compl t cov rag of offs t surfac for a car-shap d obj ct.

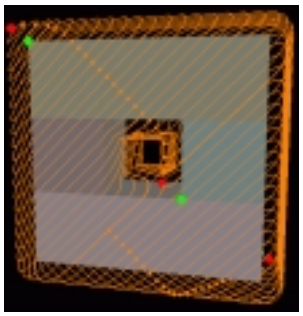

Fig. 9. Offs t surfac for an obj ct with a hol .

erage offset surface and tracing the intersection. Note that these paths do not consider the kinematics of the robot, which will be considered in future work. We also assume that the offset surface is a separator (separating inside from outside); future work will consider scenarios where "obstacles" on the target surface will be not covered, yielding an offset surface that has "holes" in it. This is useful for applications such as paint stripping hulls of ships where the "obstacles" are port-holes.

The offset surface is decomposed into cells where the boundaries of the cells are defined by critical points of a slice function evaluated on the offset surface. Each cell thus generated is easy to cover using our offset path tracing procedure. The primary contribution of this paper is to provide methods which can detect critical points for offset surfaces of unknown environments. The result that semiconcave critical points "emanate" cusps can be particularly useful for surface coverage algorithms. Currently, we do not have a surface crawling robot to test our algorithms, so we have demonstrated the approach in this paper in known polyhedral environments.

The coverage algorithms presented in this work can be useful for a variety of applications such as robotic automobile-body spray painting, paint stripping, robotic inspection or CNC tool path generation. However, for applications like car painting, it is not only necessary that the target surface be covered completely, but it is also crucial that the target surface receive a uniform amount of paint. Our current coverage procedure guarantees complete coverage of an offset surface, but it does not take into account the effect of different deposition patterns. The cellular decomposition obtained by our method needs to be matched with decompositions based on geometrical aspects such as curvature. Our future work will include coverage procedures which consider these issues.

References

[1] E. Acar and H. Chos t. Critical point s nsing in unknown nvironm nts. In *Proc. of IEEE ICRA'00*, San Francisco, CA, 2000.

[2] Z. Butl r, A. A. Rizzi, and R. L. Hollis. Contact-s nsor bas d cov rag of r ctilin ar nvironm nts. In *Proc. of IEEE Int'l Symposium on Intelligent Control*, S pt., 1998.

[3] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Pr ss, Cambridg , MA, 1988.

[4] J.F. Canny. Constructing roadmaps of s mi-alg braic s ts i: Compl t n ss. *Artificial Intelligence*, 37:203–222, 1988.

[5] J.F. Canny and M. Lin. An opportunistic global path plann r. *Algorithmica*, 10:102–120, 1993.

[6] Z. L. Cao, Y. Huang, and E. Hall. R gion filling op rations with random obstacl avoidanc for mobil robots. *Journal of Robotic systems*, pag s 87–102, F bruary 1988.

[7] H. Chos t, E. Acar, A.A. Rizzi, and J. Luntz. Exact c llular d compositions in t rms of critical points of mors functions. In *Proc. of IEEE ICRA'00*, San Francisco, CA, 2000.

[8] H. Chos t and P. Pignon. Cov rag path planning: Th boustroph don d composition. In *Proceedings of the International Conference on Field and Service Robotics*, Canb rra, Australia, D c mb r 1997.

[9] R.T. Farouki. Exact offs t proc dur s for simpl solids. *Computer Aided Geometric Design*, 2(4):257–80, 1985.

[10] Y. Gabri ly and E. Rimon. Spanning-tr bas d cov rag of continous ar as by a mobil robot. *Annals of Mathematics and Artificial Intelligence*, Acc pt d, 2000.

[11] A.Z. Gurbuz and Z id I. Offs tting op rations via clos d ball approximation. *Computer Aided Design*, 27(11):805–10, 1995.

[12] S. H rt, S. Tiwari, and V. Lum lsky. A T rrain-Cov ring Algorithm for an AUV. *Autonomous Robots*, 3:91–119, 1996.

[13] R. Kimm l and Bruckst in A.M. Shap offs ts via l v l s ts. *Computer Aided Design*, 25(3):154–62, 1993.

[14] J.C. Latomb . *Robot Motion Planning*. Kluw r Acad mic Publish rs, Boston, MA, 1991.

[15] T. Ma kawa. An ov rvi w of offs t curv s and surfac s. *Computer Aided Design*, 31(3):165–73, 1999.

[16] B. Pham. Offs t approximation of uniform B-splin s. *Computer Aided Design*, 20(8):471–474, 1988.

[17] B. Pham. Offs t curv s and surfac s: a bri f surv y. *Computer Aided Design*, 24(4):223–9, 1992.

[18] F.P. Pr parata and M. I. Shamos. *Computational Geometry: An Introduction*. Spring r-V rlag, 1985. p198-257.

[19] J. Vand rH id and N. S. V. Rao. T rrain cov rag of an unknown room by an autonomous mobil robot. T chnical R port ORNL/TM-13117, Oak Ridg National Laboratory, Oak Ridg , T nn ss , 1995.

[20] I.A. Wagn r and Bruckst in A.M. Coop rativ cl an rs: A study in ant-robotics. T chnical R port CIS-9512, C nt r for Int llig nt Syst ms, T chnion, Haifa, 1995.

[21] A. Z linsky, R.A. Jarvis, J.C. Byrn , and S. Yuta. Planning Paths of Compl t Cov rag of an Unstructur d Environm nt by a Mobil Robot. In *Proceedings of International Conference on Advanced Robotics*, pag s pp533–538, Tokyo, Japan, Nov mb r 1993.