Visibility-Based Pursuit-Evasion: The Case of Curved Environments

Steven M. LaValle, John E. Hinrichsen

Abstract— **We consider the problem of visually searching for an unpredictable target that can move arbitrarily fast in a simply-connected two-dimensional curved environment. A complete algorithm is presented, and is based on critical visibility events that occur because of inflections and bitangents on the environment boundary. By generalizing the notion of inflections and bitangents to polygonal and piecewise-smooth environments, the approach is considered as a step towards developing pursuit-evasion strategies that have little dependency on the representation of the environment.**

Keywords: Motion planning, visibility, active sensing

I. Introduction

Imagine entering a cave in complete darkness. You are given a lantern and asked to search for any people who might be moving about. Several questions might come to mind. Does a strategy even exist that guarantees I will find everyone? If not, then how many other searchers are needed before this task can be completed? Where should I move next? Can I keep from exploring the same places multiple times? This kind of scenario might apply to firepersons engaged in a rescue effort, law enforcement officials in a hostage situation, or soldiers attempting to secure a potentially-hostile area. Since it is always preferable to place robots at risk instead of humans, we might like to determine whether successful searching strategies can be computed automatically for a mobile robot. Such strategies can also provide valuable advice to people as they plan for high-risk operations.

Plenty of applications exist that could benefit from visibility-based pursuit-evasion strategies. They can be embedded in surveillance systems that use mobile robotics with various types of sensors (motion, thermal, cameras, etc.). Small mobile robots with pursuit-evasion strategies can be used by special forces in high-risk military operations to systematically search a building in enemy in territory before it is declared safe for entry. In scenarios that involve multiple robots that have little or no communication, a pursuit-evasion strategy could be used to help one robot locate others. One robot could even try to locate another that is malfunctioning. For remote presence applications, it would be valuable if a robot can locate automatically other robots and people using sensors. Beyond robotics, software tools can be developed that assist people in many applications that involve systematically searching or covering complicated environments. Relevant pursuit-evasion scenarios can be imagined in law enforcement, search-and-rescue, toxic cleanup, and in the architectural design of secure buildings. One limitation of our current work, however, is that the application must provide a complete representation of the environment.

S. M. LaValle (the corresponding author) is with the Dept. of Computer Science, Iowa State University, Ames, IA 50011 USA, lavalle@iastate.edu. +1-515-294-2259

J. E. Hinrichsen is with the Dept. of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213 USA, john4@andrew.cmu.edu

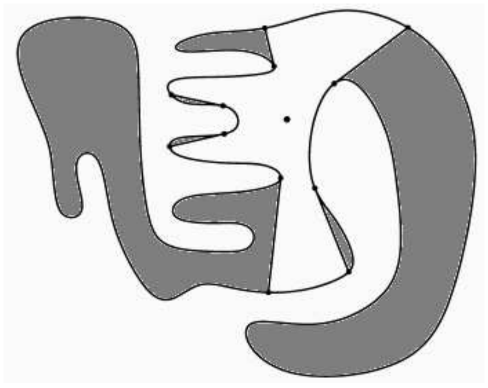

Fig. 1. The pursuer is asked to find any moving evaders in the curved environment.

It is assumed in this paper that there is a single point pursuer in a curved, planar environment, and the pursuer is given the task of searching for any moving evaders, who have unbounded speed. A search strategy is successful if all evaders will eventually fall within the line of sight of the pursuer. See Figure 1. In this paper, we propose a complete algorithm that will compute a path that a pursuer must follow to be guaranteed that all evaders will be seen, regardless of their paths. The approach developed in this paper extends previous work by LaValle et al. [15], from the case of a polygonal environment to an environment with arbitrary curves. In both cases, critical events are found in an information space, and a finite, combinatorial structure is searched that is induced by a special cell decomposition of the environment. The method in [15] would tend toward an infinite number of cells if we considered approximating curved models with arbitrarily-fine polygons. This problem motivated the current work, which identifies a finite set of critical events for a curved environment, and ultimately leads to a simplification of the cell decomposition for polygonal environments. Our approach is also motivated by similar combinatorial representations of environments for related visibility problems [8], [23], [24], [28]. The extension of the pursuit-evasion problem to curved environments is significant because it brings us one step closer to a unified approach to pursuit-evasion problems that is not sensitive to the particular environment representation. Completeness of the pursuit-evasion algorithms can generally be argued in terms of sensor-based representations of the environment, taken from the perspective of a pursuer and its sensors. Several variations of visibility-based pursuit-evasion in a polygonal environment have also been considered in [3], [27], [17], [26], [29].

II. Problem Formulation

The pursuer and evader are each points that move in an open region, R, in the plane. Three types of regions will be considered: 1) a *polygonal environment*, in which the boundary of R is a simple polygon, 2) a *smooth environment*, in which the boundary of R is a simple, closed, smooth curve, and 3) a *piecewise-smooth environment*, in which the boundary of R is a simple, closed, piecewise-smooth curve with only a finite number of nonsmooth points. Note that the third case is a generalization of the first two. The boundary could be expressed either parametrically or implicitly.

Let $e(t) \in R$ denote the position of the *evader* at time $t \geq 0$. It is assumed that $e : [0, \infty) \to R$ is a continuous function, and the evader is capable of moving arbitrarily fast (i.e., it moves at a finite, unbounded speed). Let $\gamma(t)$ denote the position of the pursuer at time $t \geq 0$. The function $\gamma : [0, \infty) \to R$ is also continuous, and is referred to as a *strategy*. For any $x \in R$, let $V(x) \subseteq R$ denote the set of all $y \in R$ such that the line segment that joins x and y does not intersect the boundary of R. Let $V(x)$ be called the *visibility region*.

The task can now be formulated. It is assumed that the pursuer does not know the starting position, $e(0)$, or the path, e, of the evader. Initially, an evader could be anywhere in R that is not visible from $\gamma(0)$. A strategy, γ, is called a *solution strategy* if for every continuous $e : [0, \infty) \to R$, there exists a time $t \in [0, \infty)$ such that $e(t) \in V(\gamma(t))$. In other words, the evader will eventually be seen, regardless of its path.

Two observations should be made. Because the evader has unbounded speed, the existence of a solution strategy does not even depend on the maximum speed of the pursuer. The primary concern is the route taken by the pursuer. Also, arbitrarily many evaders could be considered without changing the problem. In other words, guaranteeing that one evader will be found is the same as guaranteeing that for n evaders, they will all be found.

Note that the set of points not visible, $R \setminus V(x)$, is a finite collection of disjoint subsets of R. In the spirit of [22], any such subset of R that might contain the evader is referred to as a *contaminated* region. If it is guaranteed not to contain the evader, then it is referred to as *cleared*. If a region is contaminated, becomes cleared, and then becomes contaminated again, it will be referred to as *recontaminated*.

III. Critical Changes in Information

Virtually all motion strategy problems that involve sensing uncertainties can be viewed in terms of an *information space*. By analogy to the common use of *configuration space* concepts for path planning [18], [12], information space concepts provide a conceptual tool for characterizing problems. An algorithm may or may not explicitly construct a representation of the information space; however, the concepts can be used to assess the efficiency or completeness of an algorithm, or the difficulty of a problem, much in the same way that configuration space concepts have been traditionally used. Information space and related concepts have appeared in many related planning contexts [1], [5], [6], [7], [13].

For our problem, consider the information available to the pursuer. At a given time, the pursuer knows its executed trajectory, and the collection of all sensor measurements taken along the trajectory. Given some initial information regarding the evader, the trajectory and sensor measurement information can be used to determine a set $S \subseteq R$ of all contaminated points in R. Note that many different trajectories could lead to the same information state; however, information regarding the particular trajectory appears useless once S is known. Let $\eta = (x, S)$ represent an *information state*, in which x denotes the pursuer position for convenience. The set of all possible information states will be referred to as the *information space*, $\mathcal{I}$. Initially, every point not visible is contaminated; hence, $\eta = (x, S)$ in which $S = R \setminus V(x)$.

A definition will be given shortly which indicates an information invariance property that allows the information space to be partitioned into equivalence classes. As the pursuer moves in R, the information state changes. Thus, a path in R leads to a path in $\mathcal{I}$. When the pursuer moves from one position, $x \in R$, to another $x' \in R$, the corresponding information states must be different because $R \setminus V(x) \neq R \setminus V(x')$. If the pursuer moves in a closed-loop path, the initial and final information states may or may not be different.

A connected set $C \subseteq R$ is called a *conservative region* if for any initial information state (x, S) such that $x \in C$, the following occurs: if the pursuer moves along a closed loop path that maps into C, then the resulting information state is (x, S) (i.e., it is unchanged as long as the pursuer does not leave C). The information states in C can be considered as equivalent because no critical change in information occurs during motions in C. Just as in the case of motions in a conservative field, the resulting information states within a conservative region do not depend on the particular path [15].

To help identify conservative regions in R, consider Figure 2, which indicates how the environment might appear to the pursuer in Figure 1. Suppose that the pursuer has a sensor that performs an angular sweep from 0 to 2π and measures line-of-sight distance to the nearest wall, which produces an image as shown in Figure 2.a. There will generally be a finite set of orientations at which there is a discontinuity in the depth data. Let $d : R^2 \times S^1 \to \Re$ denote a real-valued function, called the *depth map*, which corresponds to the ideal distance measurements (here, S^1 denotes the topological space that is characterized by the unit circle). The value $d(x, \theta)$ gives the distance from x to the boundary of R along the ray emanating from x at an angle θ. We call each data discontinuity a *gap* in $d(x)$ ($d(x)$ is considered as a function of θ).

Each gap corresponds to a single connected region of the environment that might be contaminated. These regions are the six shaded areas in Figure 1. For each region, all

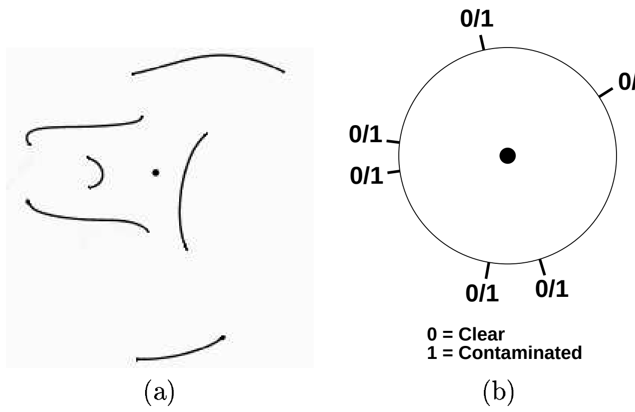

Fig. 2. Discontinuities in depth measurements partition the set of viewing directions. Each discontinuity could hide an evader.

points are either contaminated or all are clear. Therefore, any information state can be completely characterized by assigning a binary label to each one of the gaps in the angular sweep. Suppose that “0” indicates that the region is clear, and “1” indicates that it is contaminated.

Figure 2.b shows a representation of S^1 with the gaps indicated. Imagine how these gaps move in S^1 as the pursuer moves along a path. A test can be defined that determines whether the pursuer stays within a conservative region during the entire execution of its path. In practice, this could be implemented by an omnidirectional range scanner, or omnidirectional camera and edge detector. The test can be expressed entirely in terms of sensor information that is available to the pursuer, as opposed to a particular representation of the environment.

Let $\gamma : [0, 1] \to R$ be a continuous path for the pursuer. Suppose that during the execution γ, the following occur in terms of the depth map, $d(x) : S^1 \to \Re$,

1. The location in S^1 of each gap in $d(x)$ varies continuously during the execution of the path γ.
2. The number of gaps remains constant during the execution of the path γ.

Note that γ can be composed with the first argument of d to yield a function $h : [0, 1] \times S^1 \to \Re$. If the conditions above are satisfied, then h is referred to as a *piecewise homotopy* (except at the discontinuities, γ induces a homotopy on the set of depth maps [14]).

If h defines a piecewise homotopy, then $d(\gamma(t)) : S^1 \to \Re$ and its discontinuities are required to change continuously as $\gamma(t)$ varies. This implies that topology of the image of $d(\gamma(t))$ does not change. Each gap corresponds to a connected component of $R \setminus V(x)$ that is not visible to the pursuer, and has a label “0” or “1”. The connected components of $R \setminus V(x)$ remain preserved (although they will gradually change). This implies that the binary labels cannot change, and hence the information state is the same:

Observation 1: If h is a piecewise homotopy, then the image of γ in R is contained in a conservative region.

Observation 1 is useful for partitioning an environment, R, into a finite collection of conservative regions, regardless of the representation of R. The result is a combinatorial representation that can be searched for a solution to the pursuit-evasion problem. In Section IV it will be applied to the case of a smooth environment, and in Section VI it will be applied to polygonal and piecewise-smooth environments.

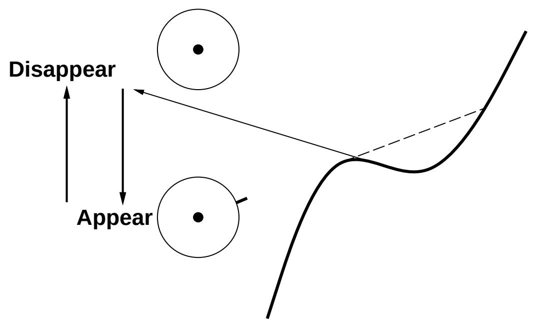

Fig. 3. An inflection causes a gap to appear or disappear, depending on the direction of the crossing.

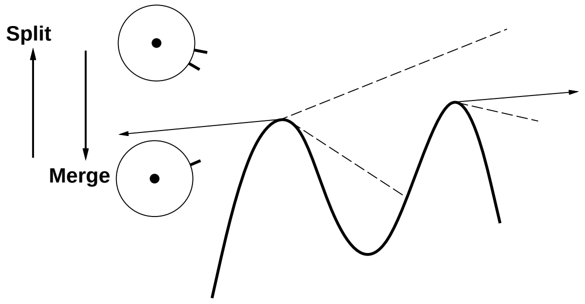

Fig. 4. A bitangent causes gaps to split or merge.

IV. An Algorithm for Smooth Environments

This section covers the case in which the boundary of the environment, R, is described by a simple, smooth, closed curve. Consider the changing depth map as the pursuer moves in this environment. There are four possible ways in which the piecewise-homotopy conditions can be violated: 1) a new gap appears; 2) an existing gap disappears; 3) two or more gaps merge into one; 4) a gap splits into two or more gaps. Each of these conditions will change the number of gaps, resulting in a critical change in the information state. Furthermore, no other possible situations can occur that would violate the piecewise-homotopy conditions. The task is to identify these causes of these four situations in a smooth environment.

Consider extending rays as shown in Figure 3. In Figure 3, a ray is extended outward from an inflection point, and is terminated when it reaches the boundary of R. If the pursuer crosses from above this ray to below it, a gap will appear. If the pursuer crosses the ray in the other direction, a gap will disappear. Thus, two of the four critical changes are caused by crossing the ray extended from an inflection point. In Figure 4, two rays are extended outward from a pair of bitangent points, and are terminated with they reach the boundary of R. If the pursuer crosses from above the left ray to below it, then one gap will split into two. If the pursuer crosses in the other direction, then two gaps merge into one. Thus, bitangents account for the other two cases.

The problem of finding all of the inflections and bitangents reduces to computing roots of polynomial equations. If the ray extensions are performed for all of these critical events, and the intersections between rays are computed, the environment, R, can be partitioned into a finite collection of cells. Each cell is conservative, and the boundary of each cell consists of segments of different extension rays, and possibly parts of the boundary of R. Two cells are adjacent if they share a one-dimensional boundary. From this adjacency, a finite graph, G, can be derived (i.e., the dual) in which the vertices represent cells and the edges represent adjacencies between cells. The resulting cell decomposition is similar to a visibility complex in computational geometry [24] and an aspect graph in computer vision [23]. A similar representation was also computed for a polygonal environment in [8], [15], [28]

The graph G corresponds to a finite collection of conservative cells in R, but what is needed is a finite collection of cells in the information space $\mathcal{I}$. A directed *information graph*, G_I, can be derived from G. Let $B(C)$ denote a binary sequence that corresponds to labelings that are assigned to gaps using the representation shown in Figure 2. For each possible binary sequence, $B(C)$, a different information state is obtained, but if the pursuer stays within C, the binary labels cannot change. This results in a collection of 2^m cells in $\mathcal{I}$ that each corresponds to a possible labeling of m gaps when the pursuer is in C. Let the graph G_I contain one vertex for each combination of cell C and its possible labelings $B(C)$. The vertices of G_I correspond to a partition of $\mathcal{I}$ into a finite set of equivalence classes.

To complete the definition of G_I, the set of edges must be defined. Each vertex of G_I will have a corresponding vertex in G that corresponds to a cell in R. When an adjacent cell is entered, the pursuer must cross an inflection ray or a bitangent ray (assuming general position). In terms of information states, it must be determined which information equivalence class is reached when going from a vertex in G_I that corresponds to C, to another vertex that corresponds to an adjacent cell, C' in R. This reduces to finding the appropriate binary labeling $B(C')$. If an inflection ray is crossed, then either a gap appears or a gap disappears. If the gap disappears, a bit simply disappears when going from $B(C)$ to $B(C')$. If a gap appears, then it always receives a "0" label. If a bitangent is crossed, then gaps either merge or split. If several gaps merge into one, then the corresponding bit in $B(C')$ will be the logical OR of the corresponding bits in $B(C)$. This is correct because one contaminated region could spread to other regions. If one gap splits into several, the corresponding bits in $B(C')$ will receive the label of the corresponding bit in $B(C)$. Note that G_I is directed because inflection and bitangent rays have different effects on the information state when crossed in opposite directions.

The task of finding a solution now reduces to searching G_I for a path between any information state with $B(C) = [1\,1 \cdots 1]$ to any information state with $B(C') = [0\,0 \cdots 0]$. The path in G_I induces a path in G. The path in G corresponds to a sequence of cells that must be visited by the pursuer. A path for the pursuer can be constructed by choosing a point in each cell and constructing a path between adjacent cells in the sequence. The cells are generally not convex; however, it is straightforward to determine a path that connects points between two adjacent cells without entering other cells or crossing the boundary of R.

The following proposition establishes the completeness of the algorithm.

Proposition 1: An algorithm that finds any path to a goal vertex from an initial vertex in G_I is complete for the visibility-based pursuit-evasion problem defined on R.

Proof: The algorithm is complete if the existence of any solution strategy implies that a path exists in G_I between initial and goal vertices. Let $\{D_1, \ldots, D_n\}$ denote the sequence of conservative cells (as we defined them with inflection and bitangent rays) that are traversed by any given solution strategy, γ. Each D_i corresponds to a vertex in G, and $\{D_1, \ldots, D_n\}$ corresponds to a path in G. This in turn corresponds to a path in G_I. Note that the information state does not depend on the path chosen within each region, D_i. From this and the fact that γ is a solution strategy, the vertex obtained at the end of the corresponding path in G_I is a goal vertex. $\triangle$

Proposition 2: If R is modeled by an implicit polynomial equation of degree d, then the size of G is $O(d^6)$.

Proof: Suppose that the boundary of R is represented by the set of solutions to an implicit polynomial equation of the form $f(x, y) = 0$ (here we use (x, y) to denote a point in R, instead of $x \in R$). The number of inflections and bitangents can be related to the degree of f. An inflection corresponds to a change in sign of the curvature. The curvature is

$$\frac{f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_{yy}f_x^2}{(f_x^2 + f_y^2)^{3/2}}, \tag{1}$$

and the inflections are the solutions of $f(x, y) = 0$ and $f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_{yy}f_x^2 = 0$. If the total degree of f is d, then the second equation has degree $3d - 3$. There at most $3d(d-1)$ inflections by Bezout's Theorem. The bitangents occur as solutions to the equations: $f(x1, y1) = 0$, $f(x2, y2) = 0$, $(x1-x2)f_y(x1, y1) - (y1-y2)f_x(x1, y1) = 0$, and $(x1 - x2)f_y(x2, y2) - (y1 - y2)f_x(x2, y2) = 0$. Once again, by Bezout's Theorem, there are no more than d^4 bitangents.

The cell decomposition that is produced can be considered as a variant of the cell decomposition introduced in [8] for localization in polygonal environments. Once the inflections and bitangents are known for a smooth environment, an equivalent polygonal environment can constructed that produces the same critical visibility events. It was shown in [8] that although a quadratic number of rays exist in the worst case, there are no more than $O(n^3)$ cells, if n is the number of edges in the polygon. The key in that argument is to observe that each ray crosses no more than a linear number of other rays. The argument can be adapted to our case, implying that there are no more than $O(d^6)$ cells, in which d is the total degree of $f(x, y) = 0$. $\triangle$

Once this has been determined, it is straightforward to apply the ideas in [15] to incrementally construct the information graph, G_I, by "lifting" each vertex in G for each information state. Simple graph search algorithms can then be employed to search G_I for a path that leads from an initial information state to the goal information state, in which the label associated with each gap is "0".

The graph G_I is exponentially larger than G due to the 2^m possible information states for each cell, in which m is the number of gaps observed by the pursuer at that cell. However, this graph does not need to be completely represented or explored. In an implementation, the nodes of G_I only need to be represented when they are encountered by the search algorithm. Furthermore, at most only one connected component of G_I needs to be constructed. The graph G_I will generally contain large components that are not reachable from the initial state. For queries that have a solution, much of the connected component that contains the initial state does not need to be explicitly constructed because a solution is often found quickly. This is partly accounted for by the fact that most examples contain numerous alternative solutions.

In the implementation described in [15], an information graph for most problems was searched using dynamic programming within a couple of seconds on a simple workstation, and within a few minutes for very complicated examples (involving over 800 vertices in G). Although that work presented an approach to pursuit evasion for polygonal environments, one can construct a smooth environment that contains equivalent critical events. This will be explained further in Section VI. The information graph, G_I, described in this paper is considerably smaller than the information graph generated in [15] for an equivalent polygonal problem. Therefore, we believe that the algorithm presented in this paper is considered at least as efficient, and often more efficient, than the one in [15] for polygonal environments. It remains an open problem to determine if G_I can be searched in polynomial time, even for a polygonal environment. The equivalent version of this problem for polygonal environments has been open for many years [27].

V. An Implementation with Examples

The cell decomposition algorithm was implemented for the case of a smooth environment using Linux, GNU C++ and the Library of Efficient Data Types and Algorithms (LEDA). Instead of representing the boundary of R implicitly, we decided that it would be easier to work directly with a parametric representation. In either case, the bitangent and inflection rays can be found as the solutions to polynomial equations. We chose to find the critical events numerically, without giving major consideration to stability issues. The purpose of our implementation is to gain some further insights to the problem through computed examples. The implementation generates the conservative cell decomposition and G.

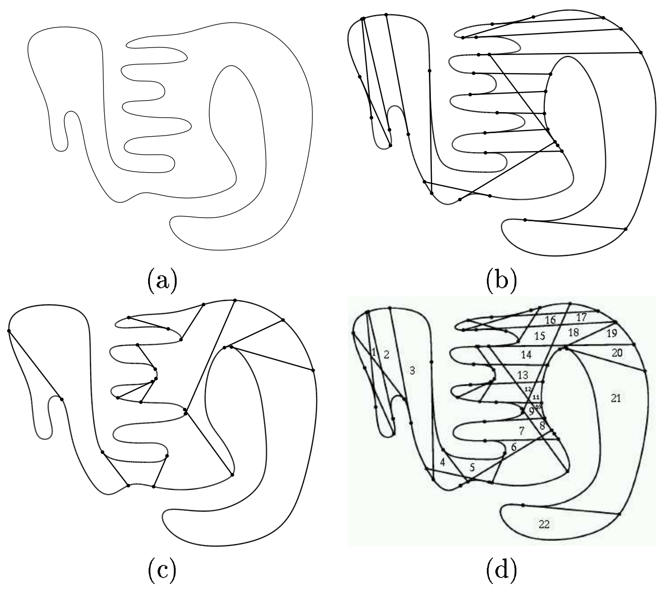

Fig. 5. (a) The input; (b) the set of inflection rays; (c) the set of bitangent rays; (d) the conservative cell decomposition and the sequence of cells visited in the solution.

Figure 5 shows the computed cell decomposition for the example shown in Figure 1. Integers from 1 to 22 show the sequence of cells that are visited in the solution path. Figure 5.d shows the cell decomposition, which is an overlay of boundaries shown in Figures 5.b and 5.c. There are 66 cells in this example. Figure 6 shows four more computed examples. In Figures 6.a and 6.b there are 13 and 26 cells, respectively. In Figure 6.c, the spiral tunnel produces exactly what we would intuitively expect: the pursuer needs to search from end to end. There are only three cells. Figure 6.d shows an example that produced 244 cells.

VI. Polygonal and Piecewise-Smooth Environments

This section presents pursuit-evasion algorithms for the cases of polygonal and piecewise-smooth environments. Both cases are handled in the same way as the smooth case, and the task is simply to find a collection of rays that violate the piecewise homotopy condition when crossed. Once these rays are identified, a cell decomposition of R can be computed, and the corresponding graphs G and G_I can be searched for a solution. For the polygonal case, this leads to a variant of the algorithm presented in [15]; however, the approach proposed here leads to far fewer vertices in G in practice. The piecewise-smooth case can be considered as a generalization of the smooth and polygonal cases, and it has not been considered previously in the context of pursuit-evasion. It is hoped that by showing these additional cases, a greater understanding of commonalities between pursuit-evasion problems under different representations can be obtained.

Polygonal environments. Suppose the boundary of R is described by a simple polygon. Figure 7.a shows how sensor

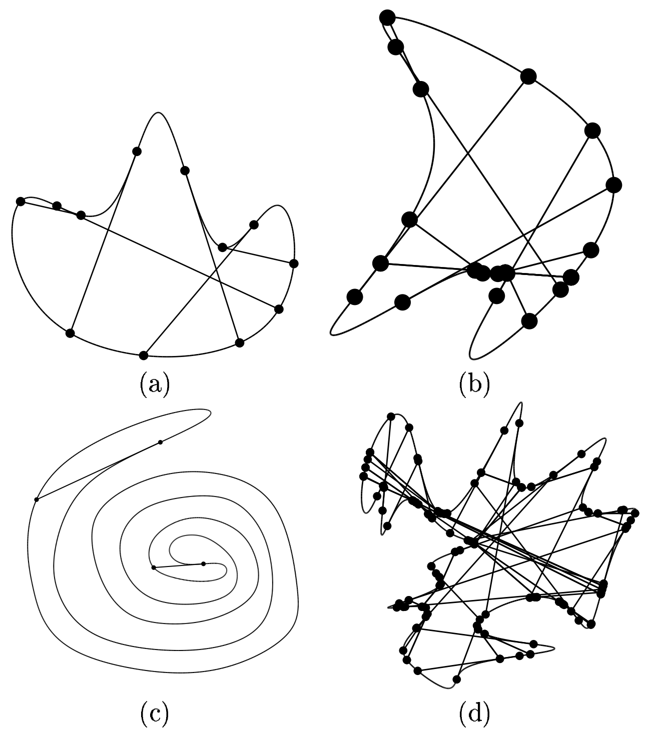

Fig. 6. Each of the figures shows the computed cell decomposition. The line segements crossing through the interior of the environment represent both the inflections and the bitangents. Each black disc indicates a place at which an inflection or bitangent ray is incident to the boundary of R.

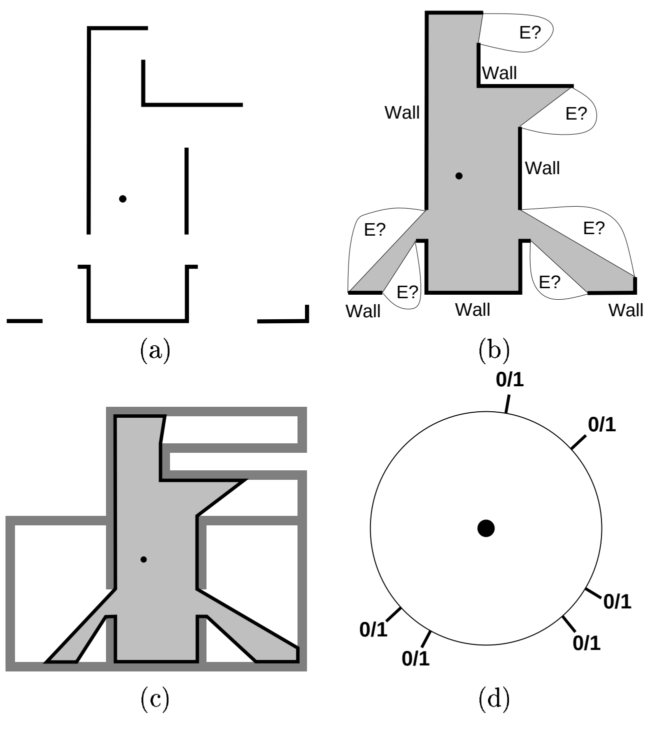

Fig. 7. (a) Distance measurements in a polygonal environment; (b) interpreting the data; (c) a possible environment that is consistent with the data; (d) the sensor-based representation can be applied to this case.

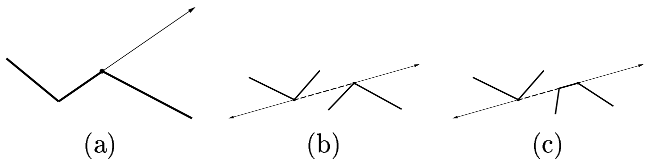

Fig. 8. These criticalities for polygonal environments correspond directly to the inflections and bitangents for smooth environments.

data would appear (ideally) to a pursuer in a polygonal environment. As in the smooth-environment case, each gap corresponds to a region where the evader might be hiding, which is shown in Figures 7.b and 7.c. Figure 7.d shows the gap locations in S^1. By using this sensor-based view, the connection between the polygonal and smooth cases becomes clear. The task is to find the situations that violate the piecewise homotopy. Figure 8.a shows the case, called a *polygonal inflection*, that is analogous to an inflection in a smooth environment. A gap appears or disappears in this case. Polygonal inflections occur when traversing the edges of a polygon in order and either: 1) a right turn is followed by a left turn, or 2) a left turn is followed by a right turn. Figure 8.b shows a *polygonal bitangent*, which corresponds to the splitting or merging of gaps. Each occurs if, for a line drawn through two or more vertices, the following are satisfied for each vertex: 1) the two edges incident to the line lie on the same side of the line, 2) successive vertices along the line are mutually visible to each other. Figure 8.c shows a degenerate case of a polygonal bitangent, which must also be considered; an entire edge can contribute to the polygonal bitangent if the edge coincides with the line, and both adjacent edges lie on the same side of the line. By identifying the criticalities as inflections and bitangents, the number of cells is reduced considerably compared to the algorithm in [15]. In that algorithm, a critical event was based on whether or not the edges in a visibility polygon change; however, this condition is too strong for the pursuit-evasion problem. Many changes in the visibility polygon to not lead to a fundamental change in the information state. The polygonal inflection and polygonal bitangent identify precisely the critical changes.

Piecewise-smooth environments. For the most general case, suppose that R is bounded by a continuous, piecewise-smooth curve. Let I denote a connected, open set of points along the boundary of R. The set I identifies a *generalized inflection* if there exists a line, L, such that I can be partitioned into three connected sets, I_1, I_2, and I_3, in the following order: 1) I_1 is an open set that does not intersect L, 2) I_2 is a closed set that is a subset of L, and 3) I_3 is an open set that does not intersect L, and lies on the opposite side of L from I_1. Furthermore, if I_2 is a single point, then its right derivative (taken in the limit over intervals in I_3) must equal the slope of L. The ray is extended from I_2 in the direction of L, towards I_1.

Let I and J denote two disjoint, connected open sets of points along the boundary of R. The sets I and J identify a *generalized bitangent* if there exists a line L such that both I and J can be each partitioned into three connected sets, I_1, I_2, and I_3, which will be described for I only (the case of J is identical): 1) I_1 is an open set that does not intersect L, 2) I_2 is a closed set that is a subset of L, and 3) I_3 is an open set that does not intersect L, and lies on the same side of L from I_1. Note the only difference in comparison to a generalized inflection is that I_1 and I_3 are on the same side of L, as opposed to opposite sides. Also, J_1 and J_3 must be on the same side of L (although, they both may be on a different side with respect to I_1 and I_3). Furthermore, at least one point of I_2 must be visible from at least one point of J_2. If all of these conditions are met, then rays can be extended outward from I_2 and J_2 in the direction of L.

The generalized inflections and generalized bitangents account for all of the ways in which piecewise homotopy can be violated. Note that the generalized inflection includes both the original inflection from Figure 3.a and the polygonal inflection from Figure 8.a. The generalized bitangent includes both the original bitangent from Figure 3.b and the polygonal bitangents from Figures 8.b and 8.c.

VII. Conclusions

In this paper, we have considered the visibility-based pursuit-evasion problem in curved environment. By viewing the sensing information in terms of moving discontinuities in depth measurements, we were able to find critical events by identifying inflections and bitangents in smooth environments. Furthermore, it appears that inflections and bitangents are fundamental to visibility-based pursuit-evasion problems, and were easily generalized to the case of polygonal and piecewise-smooth environments. It is hoped that by establishing the connections between these representations, pursuit-evasion algorithms can be easily adapted from one representation to another. Several strong assumptions have been made in this paper: the sensor is omnidirectional, the robot is in a 2D environment, and the environment is simply connected. It is hoped that some of the ideas contained in this paper can be extended to other models. For example, the bitangent and inflection insights can be used to adapt the algorithm presented in [16] for pursuit-evasion using a single flashlight in a polygonal environment to the case of smooth environments.

One interesting direction for future research is to develop an on-line pursuit-evasion algorithm. The basic idea is to search for the evader while relying only on sensor data. This general approach has been successful for several planning, search, and exploration problems [2], [4], [5], [9], [10], [11], [20], [19], [21], [25]. We believe the representation shown in Figure 7.d is the first step toward solving the pursuit-evasion problem *without* building a complete map. The correct information state can be computed for any pursuer trajectory using only the gap information. No maps or localization are required. The next step will be to develop and express a motion strategy in purely in terms of the sensor-based representation. This would lead to an approach that has minimal sensing requirements, greatly facilitating the development of an actual robotic system that performs pursuit-evasion.

Acknowledgments

We thank Leo Guibas, Jean-Claude Latombe, David Lin, and Rajeev Motwani, Boris Simov, Giora Slutzki. Many of the ideas contained herein were inspired after interactions with them. Finally, we thank Jean Ponce for his suggestions on bounding the number of critical events.

References

[1] J. Barraquand and P. Ferbach. Motion planning with uncertainty: The information space approach. In *IEEE Int. Conf. Robot. & Autom.*, pages 1341–1348, 1995.

[2] H. Choset and J. Burdick. Sensor based planning, part I: The generalized Voronoi graph. In *IEEE Int. Conf. Robot. & Autom.*, pages 1649–1655, 1995.

[3] D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor – the open edge variant of the polygon search problem. *Int. J. Comput. Geom. & Appl.*, 5(4):397–412, 1995.

[4] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. Available from "http://www.cs.berkeley.edu/~christos/", 1997.

[5] B. R. Donald. On information invariants in robotics. *Artif. Intell.*, 72:217–304, 1995.

[6] M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.

[7] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[8] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Proc. 1st Workshop on Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.

[9] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Trans. Robot. & Autom.*, 13(6):814–822, December 1997.

[10] I. Kamon, E. Rivlin, and E. Rimon. Range-sensor based navigation in three dimensions. In *IEEE Int. Conf. Robot. & Autom.*, 1999.

[11] K. N. Kutulakos, C. R. Dyer, and V. J. Lumelsky. Provable strategies for vision-guided exploration in three dimensions. In *IEEE Int. Conf. Robot. & Autom.*, pages 1365–1371, 1994.

[12] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[13] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.

[14] S. M. LaValle and J. Hinrichsen. Visibility-based pursuit-evasion: An extension to curved environments. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 1677–1682, 1999.

[15] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 737–742, 1997.

[16] S. M. LaValle, B. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. In *Proc. ACM Annual Symposium on Computational Geometry*, 2000.

[17] Jae-Ha Lee, Sung Yong Shin, and Kyung-Yong Chwa. Visibility-based pursuit-evasions in a polygonal room with a door. In *ACM Symp. on Comp. Geom.*, 1999.

[18] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. on Comput.*, C-32(2):108–120, 1983.

[19] V. Lumelsky and S. Tiwari. An algorithm for maze searching with azimuth input. In *IEEE Int. Conf. Robot. & Autom.*, pages 111–116, 1994.

[20] V. J. Lumelsky and A. A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[21] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 322–333, 1988.

[22] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. R. Lick, editors, *Theory and Application of Graphs*, pages 426–441. Springer-Verlag, Berlin, 1976.

[23] S. Petitjean, D. Kriegman, and J. Ponce. Computing exact aspect graphs of curved objects: algebraic surfaces. *Int. J. Comput. Vis.*, 9:231–255, Dec 1992.

[24] M. Pocchiola and G. Vegter. The visibility complex. *Int. J. Comput. Geom. & Appl.*, 6(3):279–308, 1996.

[25] E. Rimon and J. Canny. Construction of C-space roadmaps using local sensory data – what should the sensors look for? In *IEEE Int. Conf. Robot. & Autom.*, pages 117–124, 1994.

[26] B. Simov, G. Slutzki, and S. M. LaValle. Pursuit-evasion using beam detection. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2000.

[27] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Computing*, 21(5):863–888, October 1992.

[28] R. Talluri and J. K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Trans. Robot. & Autom.*, 12(1):63–77, February 1996.

[29] M. Yamashita, H. Unemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers. Technical Report TR-96-07-01, Dept. of Electrical Engineering and Computer Science, University of Wisconsin - Milwaukee, July 1996.