Proceedings of the *Second International Workshop on Algorithmic Foundations of Robotics*, Toulouse, France, July 1996

Robust Geometric Algorithms for Sensor Planning*

Amy J. Briggs

Department of Mathematics
and Computer Science
Middlebury College
Middlebury, VT 05753, USA
briggs@mail.middlebury.edu

Bruce R. Donald

Department of Computer Science
Upson Hall
Cornell University
Ithaca, NY 14853, USA
brd@cs.cornell.edu

Abstract

We consider the problem of planning sensor strategies that enable a sensor to be automatically configured for robot tasks. In this paper we present robust and efficient algorithms for computing the regions from which a sensor has unobstructed or partially obstructed views of a target in a goal. We apply these algorithms to the Error Detection and Recovery problem of recognizing whether a goal or failure region has been achieved. Based on these methods and strategies for visually-cued camera control, we have built a robot surveillance system in which one mobile robot navigates to a viewing position from which it has an unobstructed view of a goal region, and then uses visual recognition to detect when a specific target has entered the room.

1 Introduction

This paper introduces a computational framework in which to study the problem of sensor configuration, and develops combinatorially precise algorithms for computing partial and complete visibility maps.

A primary motivation for this work is in the domain of cooperating robots. Suppose robot A is performing a task, and robot B is equipped with sensors and should "watch" robot A. Then our algorithms can be used, for example, to compute the regions that B should not enter, if A is to remain visible. Less generally, A and B could be part of the same robot, for example, a physically distributed but globally controlled system.

*Support for this work was provided in part by the National Science Foundation under grants No. IRI-8802390, IRI-9000532, IRI-9201699, and by a Presidential Young Investigator award to Bruce Donald, and in part by the Air Force Office of Sponsored Research, the Mathematical Sciences Institute, Intel Corporation, and AT&T Bell laboratories. The first author was additionally supported by an AT&T Bell Laboratories Graduate Fellowship sponsored by the AT&T Foundation.

The Error Detection and Recovery Framework [12] provides a natural problem domain in which to apply our strategies. An *Error Detection and Recovery* (EDR) strategy is one that is guaranteed to achieve a specified goal when the goal is recognizably achievable, and signals failure otherwise. Our algorithms can be used in conjunction with an EDR planner to compute where a sensor should be placed in order to recognize success or failure of a motion plan. We explore this problem in Section 5.

Many other applications of automatic sensor configuration arise in the area of robot surveillance. In particular, our algorithms apply to the problems of intruder detection, execution monitoring, and robot reconnaissance. A natural extension of our work can be made to the problem of beacon placement for robot navigation. In Section 6 we discuss a robot surveillance demonstration system that we built using an implementation of some of the algorithms in this paper.

In this paper we restrict our attention to visibility and recognizability problems in the plane. We show that even in the $2D$ case, the geometric computations are nontrivial and significant computational issues arise, making the $2D$ case a natural first consideration. Furthermore, planning motions for a mobile robot often reduces to a computation in $2D$: a mobile robot that maintains contact with the floor usually navigates among obstacles that can be modeled as swept polygons. When the $3D$ obstacles are projected to the floor, their $2D$ footprints yield a map in $2D$. For these reasons, this and much other work in the field of visual agents is in $2D$.

1.1 Error detection and recovery

Much of the early work in robotics focused on developing guaranteed plans for accomplishing tasks specified at a high level. Such task specifications might be of the form "mesh these two gears", or "place part

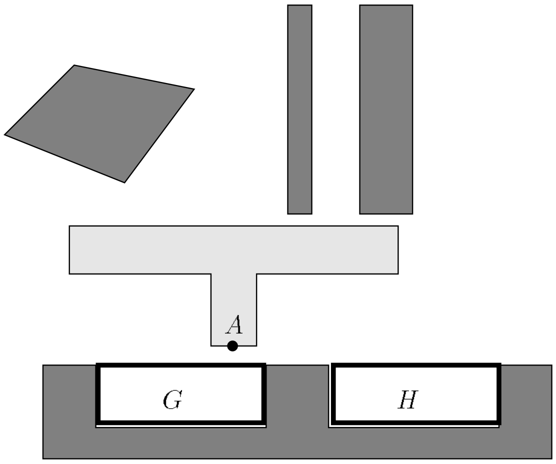

Figure 1: An example setup for the problem of sensor configuration in EDR. A represents the target; its reference point is indicated by the black dot. G is the goal region and H is the failure region. The darkly shaded polygons are obstacles. The problem is to find a sensor placement from which $A \in G$ and $A \in H$ can be distinguished.

A inside region B". It is not always possible, however, especially in the realm of assembly planning, to generate guaranteed plans. For example, errors in tolerancing of the parts might render an assembly infeasible. The Error Detection and Recovery (EDR) framework of Donald was developed to deal with these inadequacies of the guaranteed planning framework. EDR strategies will either achieve a goal if it is recognizably reachable, or signal failure. Given a geometrically-specified goal region G, an EDR strategy involves computing a failure region H and a motion plan that will terminate recognizably either in G or H. The EDR framework guarantees that under generalized damper dynamics[1], the robot will eventually reach G or H. Furthermore, having entered G or H, it will never leave. Given this guarantee of reachability, we wish to strengthen it to a guarantee of recognizability: we want to know which of G and H has been attained. The visibility algorithms presented in Section 3 will be used in showing how a sensor can be configured to distinguish between a target in G and a target in H. Figure 1 gives an example of the problem we would like to solve.

[1]The *generalized damper* is a dynamical model specified by the relationship $F = B(v - v_0)$ between forces and velocities, where F is the vector of forces and torques acting on a moving object, v_0 is the commanded velocity, v is the actual velocity, and B is a damping matrix. For more details, see Donald [12].

1.2 Related Work

The sensor placement problem has previously been addressed by Nelson and Khosla [22] and Kutulakos, Dyer, and Lumelsky [18] for visual tracking and vision-guided exploration. Several researchers have explored the problem of optimal sensor placement. Cameron and Durrant-Whyte [7] and Hager and Mintz [15] present a Bayesian approach to optimal sensor placement.

Hutchinson [16] introduces the concept of a *visual constraint surface* to control motion. The idea is to combine position, force, and visual sensing in order to produce error-tolerant motion strategies. His work builds on that of preimage planners by adding visual feedback to compensate for uncertainty. Details on the implementation of vision-based control are described by Hutchinson and Castaño [9].

Sharma and Hutchinson [26] define a measure of *robot motion observability* based on the relationship between differential changes in the position of the robot to the corresponding differential changes in the observed visual features. Lacroix, Grandjean, and Ghallab [19] describe a method for selecting view points and sensing tasks to confirm an identification hypothesis.

Cowan and Kovesi [10] study the problem of automatic camera placement for vision tasks. They consider the constraints on camera location imposed by resolution and focus requirements, visibility and view angle, and forbidden regions depending on the task. Given values bounding these constraints, they compute the set of camera locations affording complete visibility of a surface in $3D$. Zhang [31] considers the problem of optimally placing multiple sensors.

A different approach from the one taken in this paper to the incorporation of sensor planning in the EDR framework was first presented by Donald [12]. In that approach, an equivalence is established between sensing and motion in configuration space. *Active sensing* for a mobile robot is reduced to motion, by exploiting the similarity between visibility and generalized damper motions. In contrast, we present here a framework that is closer to actual sensors.

Research in the area of *art gallery theory* has introduced and addressed many problems pertaining to polygon visibility. The *art gallery problem* is to determine the minimum number of guards sufficient to guard the interior of a simple polygon (see [23] for more details). Sensor configuration planning addresses the related question of where sensors should be placed in order to monitor a region of interest. In this case we are interested in external visibility of a polygon rather than internal visibility. Furthermore, because we employ real sensors, considerations of uncertainty must be taken into account.

The questions of detecting polygon visibility and constructing visibility regions under a variety of assumptions is a rich area of past and ongoing research in computational geometry. We mention here a few of the papers most closely related to our problem. Suri and O'Rourke [27, 23] give an $\Theta(n^4)$ algorithm for the problem of computing the locus of points weakly visible from a distinguished edge in an environment of line segments. Their lower bound of $\Omega(n^4)$ for explicitly constructing the boundary of the weak visibility region holds as well for our computation of recognizability regions under a weak visibility assumption. Bhattacharya, Kirkpatrick and Toussaint [2] introduce the concept of *sector visibility* of a polygon, and give $\Theta(n)$ and $\Omega(n \log n)$ bounds, depending on the size of the visibility wedge, for determining if a polygon is weakly externally visible. The problem of planar motion planning for a robot with bounded directional uncertainty is considered by de Berg *et al.* [11]. They give algorithms for constructing the regions from which goals may be reached, and show that the complexity of the regions depends on the magnitude of the uncertainty angle.

Teller [29] solves the weak polygon visibility problem for a special case in $3D$. Namely, he computes the antipenumbra (the volume from which some, but not all, of a light source can be seen) of a convex area light source shining through a sequence of convex areal holes in three dimensions. For an environment of total edge complexity n, he gives an $O(n^2)$ time algorithm for computing the piecewise-quadratic boundary of the antipenumbra, which will be nonconvex and disconnected in general.

Tarabanis and Tsai [28] examine the question of complete visibility for general polyhedral environments in $3D$. For a feature polygon of size m and a polyhedral environment of size n, they present an $O(m^3 n^3)$ algorithm for computing the locus of all viewpoints from which the fixed feature polygon can be entirely seen.

Guibas, Motwani and Raghavan consider an abstraction of the robot localization problem [14]. Given a simple polygon P (representing the map of a known environment) and a star-shaped polygon V (representing the portion of the map visible from the robot's position), the problem is to find a point or set of points in P from which the portion of P that is visible is congruent to V (*i.e.*, given V, the robot must determine its position in P). They give a method of preprocessing P so that subsequent queries V can be answered in optimal time in the size of the output.

1.3 Outline of paper

The remainder of the paper is organized as follows. In Section 2 we introduce our approach and define the notions of *recognizability* and *confusability*. Using point-to-point visibility as a model of detectability, we present in Section 3 our algorithms for computing recognizability regions for a target polygon at a known orientation in the plane. The computed regions can be restricted to account for the error characteristics of the sensor, as shown in Section 4. In Section 5 we apply these algorithms to the EDR framework, and show how to compute the set of sensor configurations so that readings that lead to confusion of G and H are avoided. Our experimental results using mobile robots in the Cornell Robotics and Vision Laboratory are presented in Section 6.

2 Preliminaries and definitions

We will start by introducing the notation used throughout this paper, and by formalizing the problems to be solved.

An EDR plan is a motion plan that achieves either G or H, and must be able to report which of G or H has been reached. We develop a method here of determining how a sensor should be configured so that this goal recognizability can be achieved. The basic idea is that the sensor should be positioned in such a way that attainment of the goal G can be recognized, attainment of the failure region H can be recognized, and attainment of G and H cannot be confused.[2] That is, given that we know target A is in $G \cup H$, we can determine which of G or H contains A.

Our target configuration space is denoted C_r, and in this paper we consider two types of planar motion. We have $C_r = \mathbb{R}^2$ when the target has two translational degrees of freedom and a fixed orientation. When the target is allowed to both translate and rotate in the plane, the target configuration space is $\mathbb{R}^2 \times S^1$.

The sensor we employ is an idealized but physically realizable model of a point-and-shoot sensor, such as a laser range finder. A sensor configuration is specified by a *placement* and a viewing direction, or *aim*. When in a particular configuration, the sensor returns a distance and normal reading to the nearest object, which is accurate to within some known bound. Such a ranging device has been developed in our robotics

[2] This is similar to the notion introduced by Buckley [6] of a *confusable set* in the context of motion planning. There, two contact points x and y are said to be *confusable* if they are capable of generating the same position and force measurements.

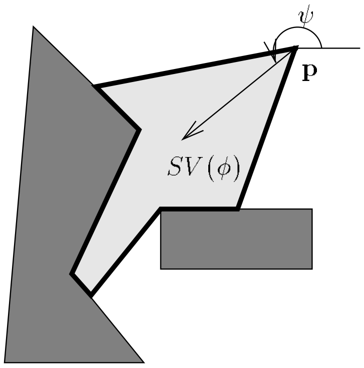

Figure 2: An example sensitive volume for a general sensor at configuration $\phi = (\mathbf{p}, \psi)$. The sensor is placed at position $\mathbf{p}$ and pointed in direction ψ. The sensitive volume is the lightly shaded region $SV(\phi)$. It is partly bounded by obstacles (darkly shaded).

laboratory at Cornell, and has been used for both map-making and robot localization [4, 5].

We denote the space of sensor placements $C_{s_p} = \mathbb{R}^2$ and the space of sensor aims $C_{s_c} = S^1$. Our sensor configuration space is $C_s = C_{s_p} \times C_{s_c} = \mathbb{R}^2 \times S^1$. For a given sensor configuration $(\mathbf{p}, \psi)$, the sensor returns distance and normal readings for a subset of $\mathbb{R}^2$. We call this subset the *sensitive volume* of the sensor, and denote it by $SV(\mathbf{p}, \psi)$. Figure 2 illustrates an example of the sensitive volume for a general sensor at position $\mathbf{p} \in C_{s_p}$ and pointed in direction $\psi \in C_{s_c}$. In what follows, we restrict our attention to questions of visibility within the sensitive volume.

For a region X in the target object's configuration space, let $R(X)$ denote its *recognizability* region, that is, the set of all sensor placements from which the sensor can detect an object A in region X. Let $C(X, Y)$ denote the *confusability* region, that is, the set of all sensor placements from which the sensor cannot tell $A \in X$ and $A \in Y$ apart. To guarantee goal recognizability for an EDR strategy, we wish to find a sensor placement $\mathbf{p} \in C_{s_p}$ such that $\mathbf{p} \in R(G) \cap R(H) - C(G, H)$. Figure 3 illustrates a case in which $A \in G$ and $A \in H$ may be confused.

Before we can solve the problem of sensor planning in the EDR framework, we must develop the necessary strategies for computing visibility maps and make the notion of detectability more concrete. In the next section we will present algorithms for computing visibility maps under two models of visibility, and will use these algorithms in subsequent sections.

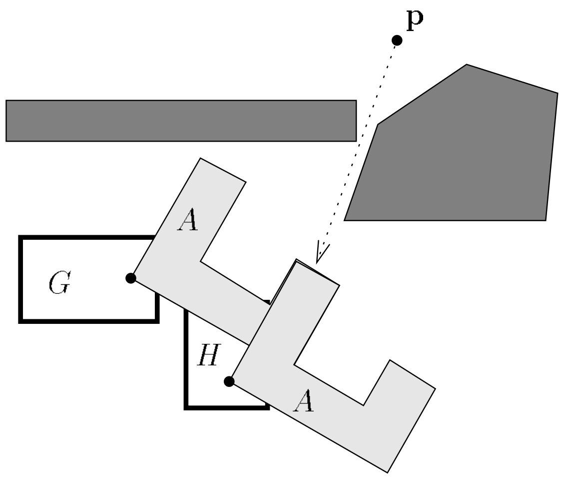

Figure 3: An instance of confusability. A represents the target; its reference point is indicated by the black dot. G is the goal region and H is the failure region. The darkly shaded polygons are obstacles. $A \in G$ and $A \in H$ are confusable from sensor placement $\mathbf{p}$.

3 Visibility algorithms

In Sections 3.1 and 3.2, we give algorithms for computing the recognizability region of a target in a goal under two different notions of visibility. We justify our use of visibility as a model for detectability by noting that the recognizability region of a target given most sensors is a subset of the region from which the target is visible. For an idealized ranging device, the visibility region of the target is equivalent to the recognizability region. The visibility regions can be restricted to account for the error characteristics of the sensor, as shown in Section 4.

Our algorithms explicitly construct the locus of points from which a polygon is visible, and we analyze how these visibility regions change as the distinguished polygon moves.

3.1 Complete visibility

In this section we consider a simplified version of the target detection problem, in which the computed sensor placements are those that allow an unobstructed view of the target. We give algorithms for detecting a stationary target, and for detecting a target at any position and orientation within a goal region.

Our result is the following:

Theorem 1 *The recognizability region of a target translating and rotating through a goal with k vertices can be computed in time $O(n\alpha(n) + nk)$ in an*

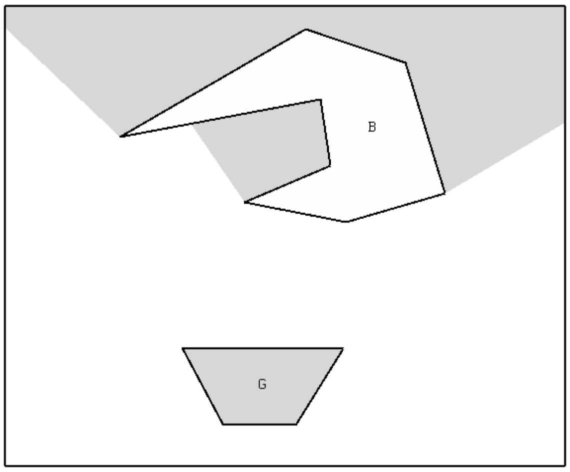

Figure 4: Shadows cast by obstacle B with respect to target G in the complete visibility model. The shadows regions are shown darkly shaded.

environment with n vertices in the complete visibility model.

3.1.1 The complete visibility algorithm for a stationary target

We say that target A at configuration $\mathbf{q} = (x, y, \theta)$ is *completely visible* from sensor placement $\mathbf{p} \in C_{s_p}$ if for no point y on the boundary of the target does the segment $\overline{\mathbf{p}y}$ intersect an obstacle. Note that $\overline{\mathbf{p}y}$ may intersect $A_{\mathbf{q}}$. If $A_{\mathbf{q}}$ is completely visible from $\mathbf{p}$ in the presence of the obstacles then we say that the sensor at $\mathbf{p}$ has an *unobstructed view* of the target at configuration $\mathbf{q}$. Our algorithm assumes that no obstacle lies within the convex hull of $A_{\mathbf{q}}$, *i.e.*,

$$\bigcup_i (CH(A_{\mathbf{q}}) \cap B_i) = \emptyset.$$

The idea is that each obstacle casts *shadows* with respect to the target. Each shadow is a subset of the sensor placement space C_{s_p} from which the target is partially occluded. See Figure 4 for an illustration. To compute the set of placements from which the target at configuration (x, y, θ) is completely visible, we use the following algorithm:

Complete visibility algorithm for a stationary target

1. Construct all local inner tangents between the obstacles and the target. Represent each tangent as a ray anchored on a vertex of the target.
2. Extend each tangent ray starting at the point of tangency with an obstacle until it hits an edge of the environment (an obstacle or the bounding polygon). We call these segments *visibility rays*. See Figure 5 for an illustration.
3. Consider the arrangement of all the polygons in the environment, along with these visibility rays. Compute the single arrangement cell that contains the target polygon. Figure 6 gives an illustration.

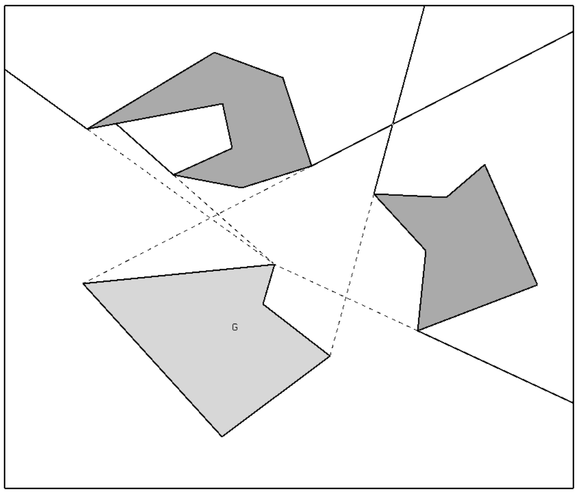

Figure 5: The segments of the tangent rays used in the arrangement computation are shown solid.

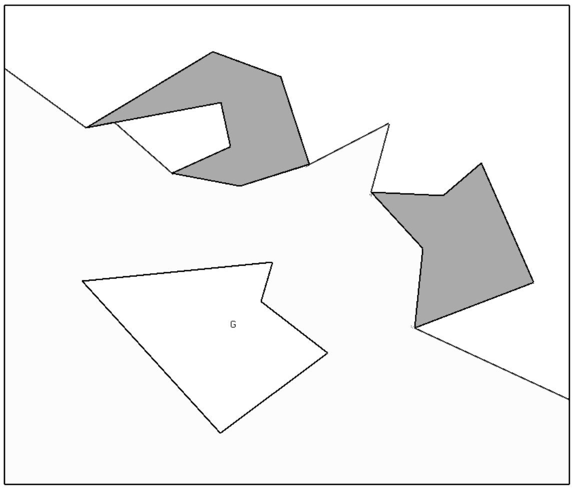

Figure 6: The arrangement cell containing G is shown lightly shaded.

Complexity of the complete visibility algorithm

Let A be a polygon representing the target, and B_i be a polygonal obstacle. If A has m vertices and obstacle B_i has n_i vertices, we can compute the $O(n_i)$ local inner tangents between A and B_i in time $O(mn_i)$. For an environment of obstacles with n vertices overall, we can compute the $O(n)$ local inner tangents in time $O(mn)$. Computing a single cell in an arrangement is equivalent to computing the lower envelope of a set of line segments in the plane, which for a set of size n takes time $O(n\alpha(n))$, where $\alpha(n)$ is the inverse Ackerman function [24]. Thus, the overall time for computing the recognizability region of a stationary target in the complete visibility model is $O(n\alpha(n) + mn)$.

3.1.2 Complete visibility over a region

We now consider sensor placements with an unobstructed view of the target at any position or orientation within a goal region G. We model the target as a connected polygon, with a reference point that lies inside the polygon. Note that the target is said to "lie in the goal" if and only if its reference point lies in the goal. The idea is that a sensor placement is valid if and only if, as its reference point moves within the goal, the entire swept area covered by the target is visible. We present two algorithms below.

Complete visibility algorithm for a translating target

First consider the case where the target has a fixed, known orientation. We further restrict the target's motion to pure translation. Denote the target at orientation θ by A_θ. Consider the Minkowski sum $A_\theta \oplus G$. $A_\theta \oplus G$ is the set of points A_θ can occupy when the reference point lies in G. The complete visibility region for the polygon $A_\theta \oplus G$ is the set of all sensor placements from which A_θ can be completely seen when its reference point lies anywhere in G.

To compute the shadow boundaries, we introduce local inner tangents between each obstacle and the convex hull of $A_\theta \oplus G$, denoted $CH(A_\theta \oplus G)$. Note that this is not an approximation; only the outermost tangents with the distinguished polygon (in this case $A_\theta \oplus G$) generate shadows in the complete visibility model.

We can compute the convex hull of $A_\theta \oplus G$ efficiently by exploiting the fact that for polygons A and B [20]

$$CH(A) \oplus CH(B) = CH(A \oplus B).$$

Note that A and B do not need to be convex. So instead of computing $CH(A_\theta \oplus G)$ explicitly, we simply convolve $CH(A_\theta)$ and $CH(G)$.

If inner tangent e is locally tangent at obstacle vertex v, then we again introduce a visibility ray that extends e away from vertex v. The arrangement of the visibility rays and the environment now partitions C_{s_p} into shadow regions and visibility regions, *i.e.*, regions from which $A_\theta \oplus G$ is partially occluded or entirely visible. But instead of computing the entire arrangement, we again note that it suffices to compute a single cell in the arrangement, namely the cell containing the goal.

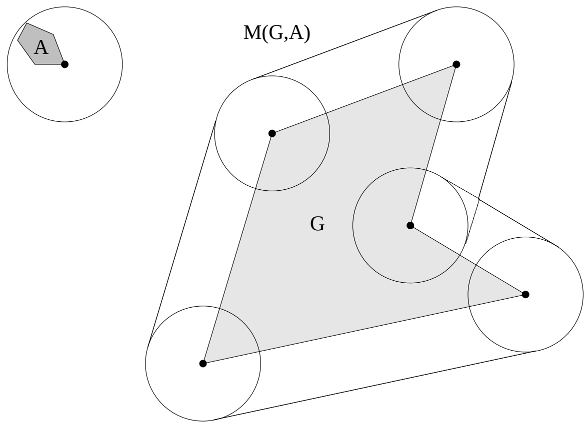

Figure 7: Target A, goal G, and the swept goal region $M(G,A)$.

Complete visibility algorithm for a translating and rotating target

Now consider the case of a target rotating and translating through the goal. We want the set of placements from which no portion of the target's boundary is occluded by obstacles no matter what position or orientation the target has within the goal. We take the longest Euclidean distance from the target's reference point to a vertex of the target. We call this distance the *radius* of the target. Suppose the target has a radius of r and its reference point lies at $\mathbf{w}$. Then the disc of radius r centered at $\mathbf{w}$ is equivalent to the area covered by the target as it rotates around its reference point.

Hence, for target A with radius r, the Minkowski sum of the goal G with the disc of radius r represents the swept area covered by the target as it translates and rotates through the goal. Call this Minkowski sum the *swept goal region* $M(G,A)$. See Figure 7 for an illustration. We now compute the set of sensor positions that have an unobstructed view of $M(G,A)$.

To compute the shadow boundaries, we introduce local inner tangents between each obstacle and the convex hull of $M(G,A)$. This can be accomplished by simply computing all inner tangents between each obstacle and the disc of radius r at each vertex of G, then taking the outermost tangents at each obstacle. Once we have these inner tangents, the rest of the algorithm is the same as above.

Complexity of the complete visibility algorithm over a region

In the translation-only case, we first compute the convex hull of $A_\theta \oplus G$. If A has m vertices and G has k vertices, $CH(A_\theta)$ and $CH(G)$ can be computed in $O(m \log m)$ and $O(k \log k)$ time, respectively [13]. So $CH(A_\theta \oplus G) = CH(A_\theta) \oplus CH(G)$ has complexity $O(m+k)$ and can be computed in time $O(m \log m + k \log k)$.

Computing the $O(n)$ local inner tangents between $CH(A_\theta \oplus G)$ and the environment can be done in time $O(n(m+k))$. The complete visibility region is the arrangement cell containing $CH(A_\theta \oplus G)$. As mentioned above, computing a single cell in an arrangement is equivalent to computing the lower envelope of a set of line segments in the plane. So the overall time to compute the visibility regions for a target translating through the goal is $O(n\alpha(n) + n(m+k))$.

In the case of a target rotating and translating through the goal, the only difference between the algorithm given here and the one given in Section 3.1.1 for a stationary target is that instead of computing tangents between a stationary target and the obstacles, we convolve a disc with the goal and compute tangents between the result of this convolution and the obstacles. In terms of complexity, the algorithms differ only in that the goal complexity rather than the target complexity is relevant. Assuming that we know the target radius r, we can compute $M(G, A)$ for a goal G of size k in time $O(k)$. If obstacle B_i has n_i vertices, we can compute the $O(n_i)$ local inner tangents between B_i and the convex hull of $M(G, A)$ in time $O(kn_i)$. For an environment of obstacles with n vertices overall, we can compute the $O(n)$ local inner tangents in time $O(kn)$. So the overall time for computing the recognizability region for a target rotating and translating through the goal in the complete visibility model is $O(n\alpha(n) + nk)$.

3.2 Partial visibility

We turn now to the question of computing recognizability regions in the *partial visibility* model. First we consider the problem of detecting a stationary target within a polygonal obstacle environment. We then apply these tools to the problem of detecting a translating target as it enters the goal.

We will show the following:

Theorem 2 *In the partial visibility model, the recognizability region of a target translating through a goal of size k can be computed in time $O(kmn^3(n+m))$ for an environment of complexity n and a target of complexity m.*

3.2.1 The partial visibility algorithm for a stationary target

We say that target A is *partially visible* from sensor placement $\mathbf{p} \in C_{s_p}$ if at least one point in the closure of A is visible from $\mathbf{p}$.

For target A at configuration $\mathbf{q} \in C_r$, we construct the partial visibility region using an approach similar to that given by Suri and O'Rourke for computing the region weakly visible from an edge [27]. Our algorithm is as follows:

Partial visibility algorithm for a stationary target

1. Construct the visibility graph for the entire environment, consisting of distinguished polygon A and obstacles B.

2. Extend each edge of the visibility graph maximally until both ends touch an edge of the environment. If neither of the endpoints of the extended visibility edge lie on the polygon A, discard the visibility edge. Otherwise, clip the edge at its intersection with A and call this piece a *visibility ray*.

3. For each vertex v in the environment, perform an angular sweep of the visibility rays incident to v. If A remains visible to v throughout the swept angle between two adjacent visibility rays anchored at v, then the triangular swept region is output as a *visibility triangle.*

The union of these visibility triangles forms the region from which A is partially visible. The complement of the union of triangles and the environment is a collection of holes in the visibility region, which we call *shadows.* Figure 9 shows the shadows for an example environment. This example demonstrates the fact that in the partial visibility model, shadows are not necessarily bounded by tangents between an obstacle and the goal.

Complexity of the partial visibility algorithm

Suppose the obstacles and bounding polygon together have n vertices, and the target has m vertices. The visibility graph for this environment, the basic data structure used in the algorithm, has size

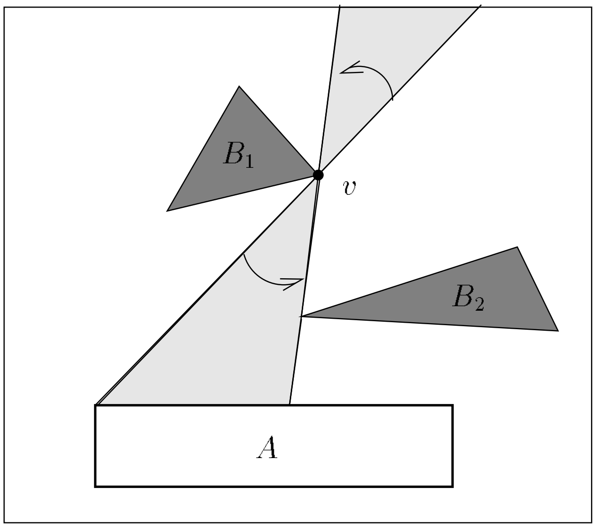

Figure 8: An angular sweep between two visibility rays at vertex v. The lightly shaded regions are visibility triangles.

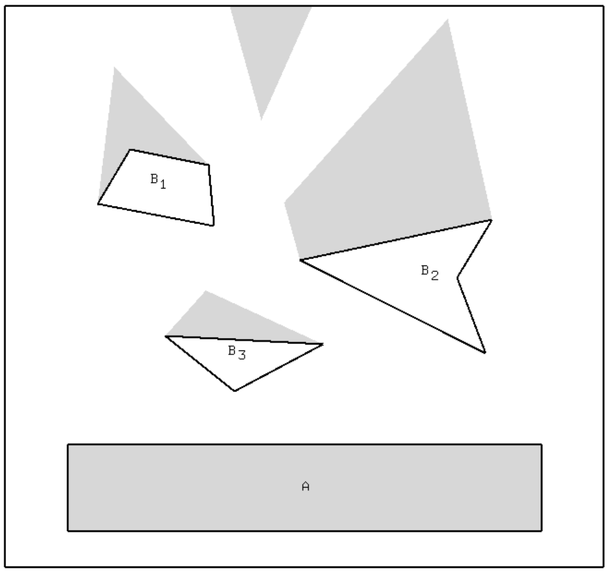

Figure 9: The shadows cast by obstacles B_1, B_2, and B_3 are shown shaded. The complement of the shadows, the obstacles, and the target forms the partial visibility region of target A.

$O(n(n+m))$. Note that we are not interested in visibility edges between the vertices of the target itself. The extended visibility graph will, in practice, have fewer edges than the basic visibility graph, since we only keep the edges whose extensions intersect the target. Its worst-case complexity, however, remains $O(n(n+m))$. Each vertex of the environment has $O(n+m)$ visibility rays incident to it. Therefore each vertex contributes $O(n+m)$ visibility triangles, so we have $O(n(n+m))$ visibility triangles overall. In general, the union of these triangles has complexity $O(n^2(n+m)^2)$. As was mentioned in the paper by Suri and O'Rourke [27], the triangles can be output in constant time per triangle: Asano *et al.* have shown that the visibility edges at a vertex v can be obtained sorted by slope in linear time with Welzl's algorithm for computing the visibility graph [30, 1]. Thus, the overall time for explicitly computing the boundary of the partial visibility region for target A at any fixed configuration $\mathbf{q}$ is $O(n^2(n+m)^2)$. The region can be given as a union of triangles, without computing the boundary, in time $O(n(n+m))$.

3.2.2 Partial visibility over a region

The algorithm above solves the problem of detecting a stationary target in the partial visibility model. We now address the problem of maintaining line-of-sight contact with the target as it moves within the confines of a particular polygon, for example, as the target moves within the goal. How do the visibility triangles and shadows change as the target moves? To answer this question, we need to introduce some additional terminology. Let e be a visibility edge whose associated visibility ray intersects the target at point x. The endpoint of e lying closer to x (possibly x itself) is defined as the *anchor* vertex of e, while the further endpoint is called the *attachment* vertex of e. If a vertex of the shadow (considering the shadow as a polygon) lies in free space, *i.e.*, if it lies inside the bounding polygon and is not on the boundary of an obstacle, then we call it a *free* vertex of the shadow.

As the target translates, free shadow vertices trace out point conics if their generating edges are anchored on the target [3].

3.2.3 Swept shadows in the partial visibility model

We have shown how to compute shadows for any fixed target position, and have discussed how these shadows change as the target translates. In order to detect the target as it enters the goal, we must compute the shadows swept for all positions of the target in the goal. We define a *swept shadow* of the goal in the partial visibility model to be a maximal connected region of C_{s_p} such that for each point $\mathbf{p}$ in the region, there exists a configuration of the target in the goal from which the target is totally occluded.

We compute swept shadows for the target at a fixed orientation anywhere in the goal by translating the target polygon along the edges of the goal polygon. The boundary of a swept shadow is composed of obstacle segments and the curves (lines and conics) traced by free vertices. Discontinuities in the boundary of a swept shadow occur at *critical events*. We characterize the critical events as follows:

1. A moving visibility ray becomes aligned with a fixed edge of the visibility graph.
2. A free vertex of a shadow intersects an obstacle edge or the bounding polygon.
3. Two moving visibility rays bounding a shadow become parallel.

Below we present our algorithm for computing the partial visibility region of a target as it translates through the goal at a known orientation θ. This gives us the set of all sensor placements from which at least one point on the boundary of the target can be seen, no matter where the target is in the goal.

Partial visibility algorithm for a translating target

1. Let e be any edge of goal G. Consider A_θ to be placed on one of the endpoints of e. Call this configuration $\mathbf{q}$. Construct the partial visibility region of target A at configuration $\mathbf{q}$.
2. Translate A_θ along e. As the shadows cast by the obstacles change, call the area swept out by a shadow a *swept shadow*. Between critical events, the vertices of each shadow move along lines or conics. The equations of these curves can be computed algebraically given the positions of the obstacles in the environment and the visibility rays. Update the boundary of the swept shadows at critical events.
3. Translate A_θ along all other edges e_i, $1 \leq i \leq k$, of G, repeating step 2 for each edge.
4. Compute each swept shadow independently as described in the above steps. The complement of the union of all the swept shadows, the target, and the obstacles is the partial visibility region.

The output of the algorithm is the set of swept shadows. Note that the boundary of a swept shadow is piecewise linear and conic.

Complexity of the partial visibility algorithm over a region

The extended visibility edges bounding the shadows are all either external local tangents between an obstacle and the target, or internal local tangents between obstacles. Since the obstacles are fixed, the visibility edges between them remain fixed. As the target moves, the only visibility edges that move are those that are anchored on a vertex of the target.

With n vertices in the environment and m target vertices, there are $O(mn)$ moving visibility edges. As the target translates along an edge of the goal, a visibility edge anchored at target vertex a_i and attached at obstacle vertex b_j could become aligned with each of the $O(n)$ fixed visibility edges at obstacle vertex b_j. This gives $O(mn^2)$ critical events of the first type as the target translates along an edge of the goal. There are $O(m^2n^2)$ free vertices tracing out curves, which may intersect each of the $O(n)$ obstacle segments. This gives $O(m^2n^3)$ critical events of the second type. When the third type of critical event occurs, a free vertex disappears. There are $O(m^2n^2)$ of these events.

At a critical event of the first type, a visibility ray appears or disappears, causing a visibility triangle to appear or disappear. The total cost of handling all updates of this type is $O(mn^3(n+m))$. Only local change is caused by events of the second type and third type.

Between critical events, we simply grow the shadows, either along lines or conics. Note that the shadows never shrink: A point $\mathbf{p} \in C_{s_p}$ is in a shadow with respect to a polygonal goal if there exists some target configuration such that the target is not at all visible from $\mathbf{p}$. The computation of swept shadows is done by translating the target polygon along the edges of the goal, updating the boundary at critical events. The total running time of the algorithm for a goal with k vertices is $O(kmn^3(n+m))$.

4 Uncertainty in sensor placement and aim

A real sensor cannot be configured exactly. Rather, it will be subject to both errors in placement and errors in aim. These errors depend on the sensor platform (*e.g.*, a mobile robot). Therefore we would like to compute sensor strategies that take uncertainty in sensor configuration into consideration. In this section, we sketch how the computation of visibility regions can be extended to handle this type of sensor

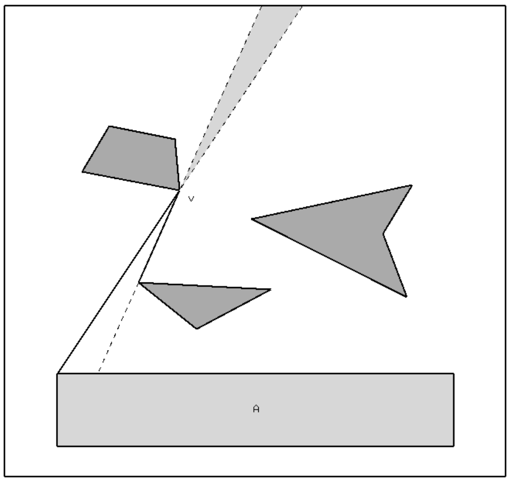

Figure 10: A narrow visibility triangle anchored at vertex v is shown lightly shaded.

error. Our approach does not address the problem of sensor measurement error.

Positional uncertainty characterizes the sensor placement error. Let ϵ_{pos} denote the worst-case positional uncertainty of the sensor. If the commanded sensor placement is $\mathbf{p}$, the actual sensor placement could be any position in the disc of radius ϵ_{pos} centered at $\mathbf{p}$. We handle positional uncertainty by growing the shadows by the uncertainty ball of radius ϵ_{pos}. The complement of the union of these grown shadows and the environment will be the visibility region that accounts for uncertainty in sensor position.

Directional uncertainty characterizes the sensor aim error. Let ϵ denote the maximum angular error of the sensor aim. That is, if the commanded sensing direction is ψ, the actual sensor heading could be any direction in the cone $(\psi - \epsilon, \psi + \epsilon)$. The effect of sensor directional uncertainty is that we must disallow angularly narrow wedges of visibility. This type of uncertainty is most relevant in the case of partial visibility. See Figure 10 for an illustration of a narrow visibility triangle. This triangle does not become part of the visibility region when directional uncertainty is considered.

After we compute the visibility rays as described in Section 3.2.1, we visit each vertex in the environment, and combine adjacent visibility triangles that end on the same polygon. We make the following definitions:

1. The maximal union of adjacent visibility triangles anchored on a single vertex v and ending on the same polygon is called a *visibility polygon*. By construction, visibility polygons are simple.

2. The *core triangle* of a visibility polygon anchored at v is the maximal inscribed triangle whose apex is v.

If the angle at the apex of such a maximal visibility triangle is less than our angular uncertainty bound ϵ, we discard the polygon. Otherwise, we classify the maximal visibility triangle as an *ϵ-fat triangle*. After this processing, we now have $O(n(n+m))$ fat visibility triangles. We can now use a result of Matoušek *et al.* [21] on the union of fat triangles. Their result bounds the number of *holes* in a union of fat triangles. In our case, the "holes" are shadows in a union of visibility triangles. Their theorem states that for any fixed $\delta > 0$, and any family $\mathcal{F}$ of n δ-fat triangles, their union has $O(n/\delta^{O(1)})$ holes. When we restrict our visibility triangles to be at least ϵ-fat, we have at most $O((n(n+m))/\epsilon^{O(1)})$ shadows.

When ϵ is a fixed constant, we have at most $O(n(n+m))$ shadows. In effect, this means that considering directional uncertainty actually lowers the complexity of computing the recognizability region. Note that our construction yields a conservative approximation to the recognizability region under uncertainty.

The next section extends the sensor placement algorithms presented here to the domain of Error Detection and Recovery by avoiding placements that could give ambiguous readings.

5 Avoiding confusable placements

The set $C(G, H)$ is the set of all sensor placements that could lead to confusion of G and H. A placement $\mathbf{p}$ is in the confusable region if the only visible portion of the target polygon could be due to an edge of A in G or an edge of A in H.

Note that a sensor reading that confuses a target $A_{\mathbf{q}}$ in G with a target $A_{\mathbf{q}'}$ in H is due to an edge of $A_{\mathbf{q}}$ being colinear with an edge of $A_{\mathbf{q}'}$. See Figure 11 for an example.

For each pair of edges (e_i, e_j) having the same orientation, we compute the *overlap* region $O(e_i, e_j) = (e_i \oplus G) \cap (e_j \oplus H)$. We define O_α to be the union of all $O(e_i, e_j)$ for all pairs of edges (e_i, e_j) having orientation α. See Figure 12.

The confusable region is defined as

$$C(G, H) = \{\mathbf{p} \mid \exists \mathbf{q} \in G \cup H, \forall \psi : (SV(\mathbf{p}, \psi) \cap A_{\mathbf{q}}) \subseteq O(e_i, e_j) \text{ for some } (e_i, e_j).\}.$$

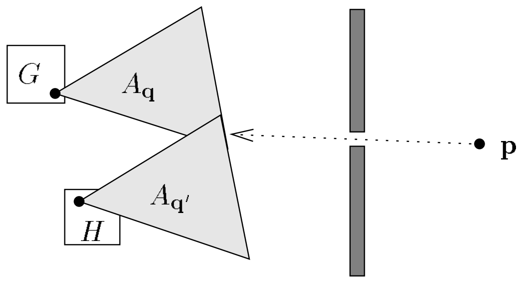

Figure 11: A sensor reading that confuses $A_{\mathbf{q}} \in G$ and $A_{\mathbf{q}'} \in H$ is due to an edge of $A_{\mathbf{q}}$ being colinear with an edge of $A_{\mathbf{q}'}$. The darkly shaded rectangles are obstacles.

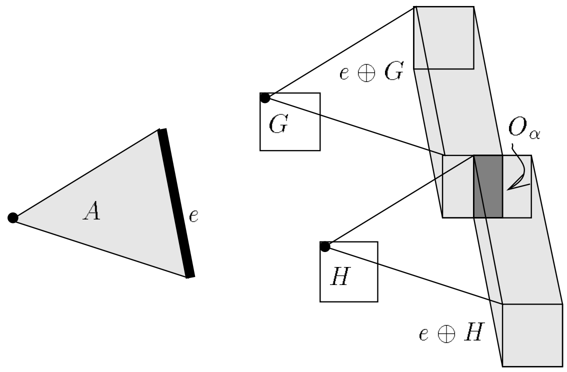

Figure 12: Edge e at orientation α of target A is convolved with G and H. The darkly shaded region is the overlap O_α. Sensor readings in O_α can lead to confusion of G and H.

5.1 Discrete goal and failure regions

Before turning to the problem of handling polygonal goal and failure regions, we first consider the case in which the goal and failure regions are discrete points. Our technique for computing the set of good sensor placements is to first compute the set of overlap regions, and then compute the recognizability regions for the non-overlap portion of A in G and the non-overlap portion of A in H. The algorithm is as follows:

1. Compute all overlap regions $O(e_i, e_j)$ for all pairs of edges (e_i, e_j) having the same orientation. Note that in the case of point-sized goal and failure regions, the overlap regions consist of edge segments.

2. Perform the following steps for A in G and A in H:

   (a) Construct a new target A' by deleting the overlap segments from A. Figure 13 illustrates the overlap for a point-sized goal and failure region. The new target consists of a set of edge segments, where each edge segment has an associated outward-facing normal, so it is visible only from one side.

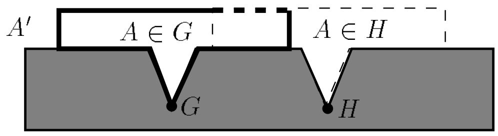

Figure 13: The set of thick solid edges comprises A' for $A \in G$. The dashed line outlines the target polygon in the failure region H. The thick dashed line is the overlap region.

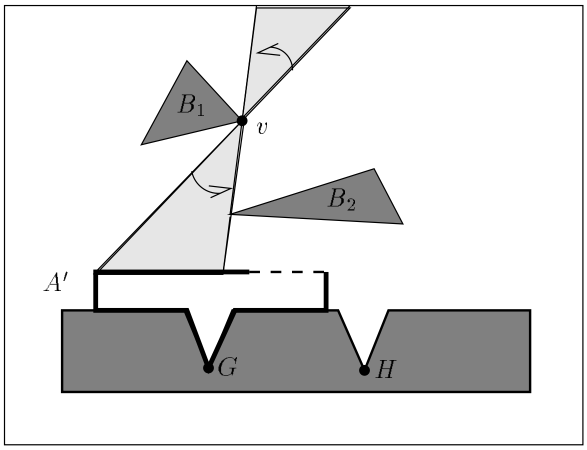

Figure 14: An angular sweep between two visibility rays at vertex v. The lightly shaded regions are visibility triangles. The thick solid edges comprise A', and the dashed line is the overlap region.

   (b) Compute the set of visibility triangles for target A' using the *partial visibility algorithm for a stationary target* as described in Section 3.2. Figure 14 gives an illustration of some visibility triangles.

   (c) Compute the union of the visibility triangles formed above. This is the partial visibility region for the non-overlap portion of A at this configuration.

3. Compute the intersection of the two visibility regions computed for A in G and A in H in steps 2(a)–2(c) above. This gives the set of all sensor placements from which both A in G and A in H can be recognized, but not confused.

5.2 Polygonal goal and failure regions

In the case of polygonal goal and failure regions, the computation of $R(G) \cap R(H) - C(G, H)$ is an incremental one. Recall that each overlap region is due to an edge of A in G being colinear with an edge of A in H. In this case, the overlap region $(e_i \oplus G) \cap (e_j \oplus H)$ for parallel edges e_i and e_j is formed by a line sweeping through a region determined by G and H.

To determine the set of placements from which G and H can be distinguished but not confused, we do the following:

1. Compute the set of overlap regions O_α for all orientations α of the edges of A.
2. Place A at a vertex of G. Let $A'(\mathbf{q}) = A_\mathbf{q} - (A_\mathbf{q} \cap (\cup O_\alpha)$ be the set of edge segments of A at configuration $\mathbf{q}$ not intersecting any O_α.
3. Compute the partial visibility region of $A'(\mathbf{q})$ as it sweeps through G, as described in Section 3. Note that the endpoints of the edges of $A'(\mathbf{q})$ are not fixed, but vary during the sweep.
4. Repeat steps 2 and 3 for A sweeping through H.
5. Take the intersection of the regions computed for A sweeping through G and H, respectively.

The resulting region is the set of all placements from which A at any position in $G \cup H$ can be detected, but $A \in G$ and $A \in H$ can not be confused.

6 Experimental results

The algorithms for computing the complete and partial visibility regions of a polygon have both been implemented and used in conjunction with existing packages for graphics, geometric modeling, and plane sweep.

We used the implementation of the complete visibility algorithm to build a demonstration of robot surveillance using two of the mobile robots in the Cornell Robotics and Vision Laboratory. The autonomous mobile robots are called TOMMY and LILY. The task was for TOMMY to detect when LILY entered a particular doorway of the robotics lab. Initially TOMMY is at a position from which this doorway cannot be seen. Below we describe the various components of the system.

6.1 The visibility component

We constructed by hand a map of our lab, and used that map as the input environment to the complete visibility system. The map and the computed visibility region of the doorway are shown in Figure 15.

TOMMY's task was to monitor the doorway, which is marked in the Figure with "G". The dark gray regions are obstacles representing real objects in the room — chairs, desks, couches, bookshelves, *etc.* Given that most of the objects are regularly shaped and resting on the floor, the idea of using polygons as "footprints" of $3D$ objects turned out to give a good approximation of the $3D$ geometry. Given this map, our algorithms give us the exact placements from where the doorway can be monitored. The lightly shaded region in Figure 15 is the complete visibility region for this environment — the exact set of placements from where the doorway can be entirely seen with a sensing device such as a CCD camera.

6.2 Choosing a new placement

Based on the visibility region and the initial configuration of TOMMY, a new configuration is computed inside the visibility region. A motion plan to reach that new configuration is generated along with the distance from there to the goal.

In particular, we do the following to choose such a placement. We first shrink the visibility region to account for model and sensor uncertainty. The procedure to perform this shrinking returns a list of edges making up the *shrunk visibility region*. We now want to choose a new point inside this shrunk visibility region, one that is closest to the current position of the robot. We use the following heuristic to find such a point: we discretize the edges of the shrunk visibility region, obtaining a list of candidate points. We then sort this list of points by distance from the current position of the robot. Then test each of the points, searching for one that is reachable from the current position in a one-step motion. The first such point found is returned as the new configuration. If no such point is found, this is signaled. This could be due to two reasons: a point reachable in a one-step motion was missed due to the discretization being too coarse, or no one-step motion plan exists (*i.e.*, the robot would have to move around corners, or can not reach the visibility region at all). While the former case could easily be fixed by iteratively refining the discretization, the latter case requires the use of a full-fledged motion planner.

Figure 16 shows the shrunk visibility region and one of the starting points we used, as well as the new placement which was computed using the method described above.

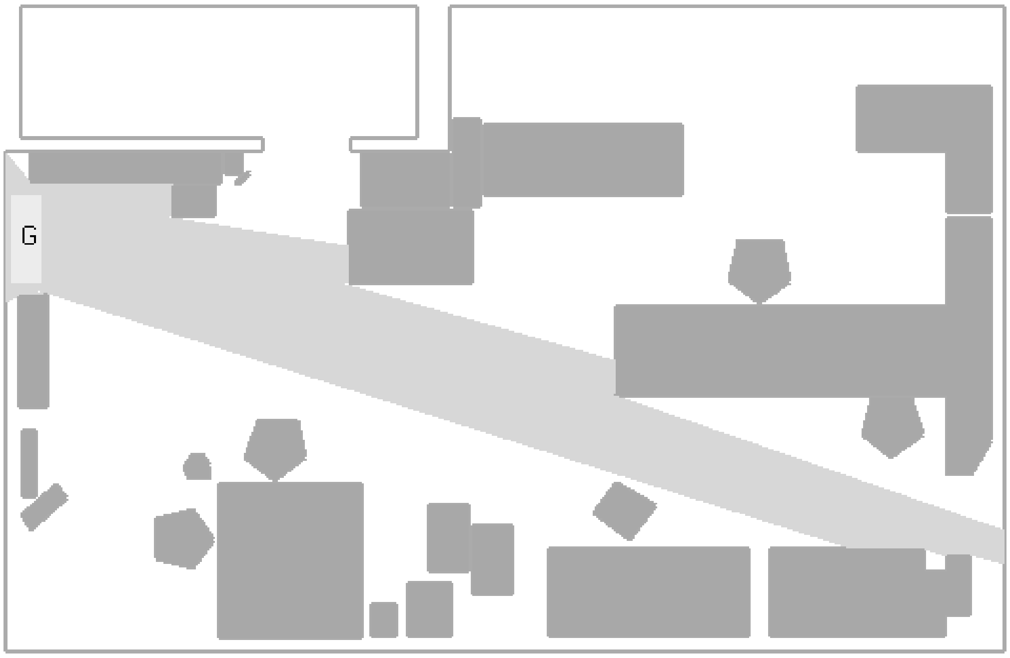

Figure 15: The map and complete visibility region for the robotics lab.

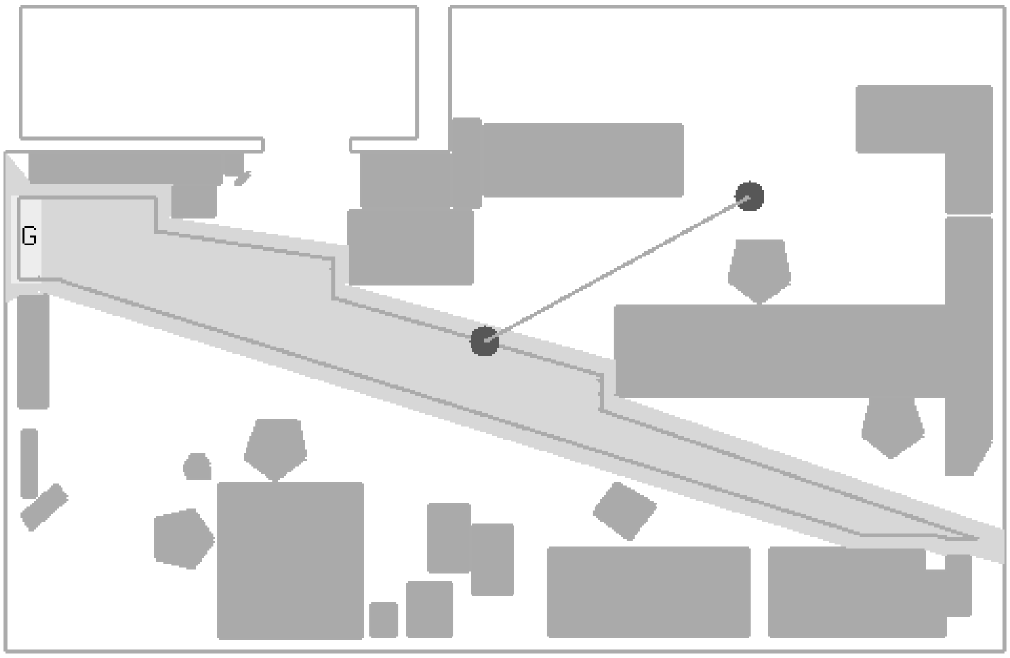

Figure 16: The shrunk visibility region, the computed new configuration and the one-step motion.

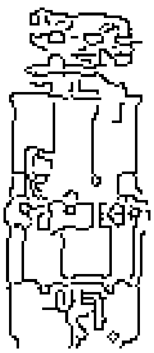

Figure 17: Our model of LILY.

6.3 Computing the viewing direction

The planner computes a viewing direction depending on the new placement and information obtained from the map. We fix a coordinate frame for the lab with the origin in one corner of the room. Then we compute the vector between the new computed placement and the centroid of the goal. The θ component of the new configuration is simply the angle of this vector. The final output from the planner is a vector containing the x-, y-, and θ-components of the new configuration, along with the distance in world coordinates from this new configuration to the centroid of the goal.

6.4 Locating LILY

LILY's task is to move into the doorway and wait for TOMMY. LILY is run without a tether. She is programmed to translate a fixed distance and stop (in the center of the doorway). She then waits until her bump sensors are activated. When a bumper is pressed, LILY translates a fixed distance in reverse, rotates by 180 degrees, and then translates forward a fixed distance in order to leave the room.

Here is how the surveillance and recognition parts of the system work.

We first built a calibrated visual model of LILY. We used the Panasonic CCD camera mounted on TOMMY to take a picture of LILY from a known fixed distance (4 m). We then computed the intensity edges for that image using an implementation of Canny's edge detection algorithm [8]. The actual model of LILY that we created and used is shown in Figure 17. We did not alter the intensity edges that Canny's algorithm output, and experimentation demonstrated that our results are relatively insensitive to the particular image taken.

Based on the distance information from TOMMY's new configuration, the model edges are scaled to the expected size of LILY's image as seen from this configuration, using the fact that the image size is inversely proportional to the distance.

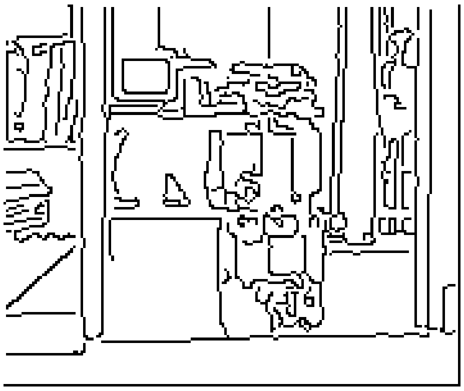

Figure 18: Intensity edges for the cropped image.

The video camera on TOMMY is used to repeatedly grab image frames, which along with the scaled model are input to a matcher that operates on edge images. The following loop is performed until a match is found:

1. Grab a frame.
2. Crop it, keeping only the portion of the image where LILY is expected to be.
3. Compute intensity edges for the cropped image.
4. Run the matcher to find an instance of the scaled model in the cropped image.

Figure 18 shows the intensity edges for a crop of one of the images that was grabbed with TOMMY's video-camera once TOMMY had moved into the computed configuration.

The matcher used in the experiment is based on the Hausdorff distance between sets of points and was written by William Rucklidge [25] and has been used extensively in the Cornell Robotics and Vision Laboratory for image comparison, motion tracking, and visually-guided navigation [17].

The particular matcher used here is a translation-only matcher that uses a fractional measure of the Hausdorff distance. Matches are found by searching the $2D$ space of translations of the model, and computing the Hausdorff distance between the image and the translated model. A match occurs when the Hausdorff distance of a certain fraction of the points is below some specified threshold. All translations of the model that fit the image are returned.

The dark gray outline in Figure 19 shows all matches that were found between the scaled model of LILY and the image in Figure 18.

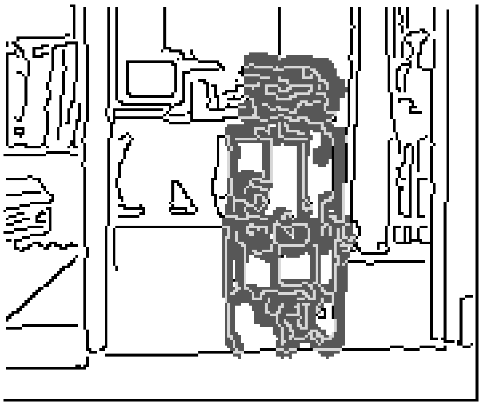

Figure 19: Matches found between the image and the scaled model.

Based on where LILY is found in the image, TOMMY first performs a rotational correction so that LILY is centered in the image. An estimated value for the focal length of the camera was used to perform a rotation to correct for errors in dead reckoning. TOMMY then moves across the room to where LILY is using a simple guarded move.

6.5 Analysis

We videotaped several runs of the system. For these runs, we used two different starting positions for TOMMY, on different sides of the room, both from where the goal doorway could not be seen. We also demonstrated the robustness of the system by having people enter and leave through the doorway while TOMMY was monitoring it. The system performed consistently well. TOMMY never reported a false match — neither when the doorway was empty, nor when other people stood in the doorway. Once LILY was in position, the recognition component (on a SPARC 20 running Solaris) typically took 2–4 seconds to locate LILY. Disk access time accounted for some of this time (saving and loading image files) and could be eliminated by using a different file access strategy.

7 Conclusion

In this paper we explored the problem of automatic sensor placement and control. We presented methods for computing the placements from which a sensor has an unobstructed or partially obstructed view of a target region, enabling the sensor to observe the activity in that region. In particular, we have presented algorithms for computing the set of sensor placements affording complete or partial visibility of a stationary target, complete visibility of a target at any position or orientation within a goal, and partial visibility of a target translating through a goal at a known orientation. The algorithms account for uncertainty in sensor placement and aim.

The *Error Detection and Recovery (EDR)* system of Donald [12] provides a framework for constructing manipulation strategies when guaranteed plans cannot be found or do not exist. An EDR strategy attains the goal when the goal is recognizably reachable, and signals failure otherwise. Our results extend the guarantee of reachability to a guarantee of recognizability for the case of a polygon translating in the plane. In future work we plan to address the problem of planning sensing strategies when the target polygon may translate and rotate, resulting in unknown orientations of the target in G and H.

The implementation of the complete visibility algorithm was used as the planning component in a robot surveillance system employing both task-directed and visually-cued strategies. The system plans and executes sensing strategies that enable a mobile robot equipped with a CCD camera to monitor a particular region in a room, and then react when a specific visually-cued event occurs. Our experimental results demonstrate both the robustness and applicability of the visibility algorithms we have developed. They show that complete visibility of a goal region can be computed efficiently, and provides a good model of detectability in an uncluttered environment. We believe that this successful effort has validated our principled approach to planning robot sensing and control strategies.

Acknowledgements

Many of the ideas in this paper arose in discussions with Mike Erdmann, Tomás Lozano-Pérez, and Matt Mason. The robots and experimental devices were built in the Cornell Robotics and Vision Laboratory by Jim Jennings, Russell Brown, Jonathan Rees, Craig Becker, Mark Battisti, and Kevin Newman. William Rucklidge wrote the vision recognition system, and Mike Leventon and Daniel Scharstein wrote some of the vision code that was loaded onto the robots. Thanks especially to Daniel Scharstein for many discussions and insightful comments on this work.

References

[1] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[2] B. Bhattacharya, D. G. Kirkpatrick, and G. T. Toussaint. Determining sector visibility of a polygon. In *Proc. ACM Symp. on Comp. Geom.*, pages 247–253, June 1989.

[3] A. J. Briggs. *Efficient Geometric Algorithms for Robot Sensing and Control.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, NY, Jan. 1995.

[4] R. G. Brown. *Algorithms for Mobile Robot Localization and Building Flexible, Robust, Easy to Use Mobile Robots.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, NY, May 1995.

[5] R. G. Brown, L. P. Chew, and B. R. Donald. Localization and map-making algorithms for mobile robots. In *Proc. IASTED Int. Conf. on Robotics and Manufacturing*, pages 185–190, Sept. 1993.

[6] S. J. Buckley. *Planning and Teaching Compliant Motion Strategies.* PhD thesis, MIT Artificial Intelligence Laboratory, 1987.

[7] A. J. Cameron and H. Durrant-Whyte. A Bayesian approach to optimal sensor placement. *IJRR*, 9(5):70–88, 1990.

[8] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):34–43, 1986.

[9] A. Castaño and S. Hutchinson. Hybrid vision/position servo control of a robotic manipulator. In *Proc. IEEE ICRA*, pages 1264–1269, May 1992.

[10] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416, May 1988.

[11] M. de Berg, L. Guibas, D. Halperin, M. Overmars, O. Schwarzkopf, M. Sharir, and M. Teillaud. Reaching a goal with directional uncertainty. In *Proc. Int. Symp. on Algorithms and Computation*, Dec. 1993.

[12] B. R. Donald. *Error detection and recovery in Robotics*, volume 336 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1989.

[13] R. L. Graham and F. F. Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[14] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In *SODA*, 1992.

[15] G. Hager and M. Mintz. Computational methods for task-directed sensor data fusion and sensor planning. *IJRR*, 10(4):285–313, Aug. 1991.

[16] S. Hutchinson. Exploiting visual constraints in robot motion planning. In *Proc. IEEE ICRA*, pages 1722–1727, Apr. 1991.

[17] D. P. Huttenlocher, M. E. Leventon, and W. J. Rucklidge. Visually-guided navigation by comparing two-dimensional edge images. In *CVPR*, pages 842–847, 1994.

[18] K. N. Kutulakos, C. R. Dyer, and V. J. Lumelsky. Provable strategies for vision-guided exploration in three dimensions. In *Proc. IEEE ICRA*, pages 1365–1372, May 1994.

[19] S. Lacroix, P. Grandjean, and M. Ghallab. Perception planning for a multi-sensory interpretation machine. In *Proc. IEEE ICRA*, pages 1818–1824, May 1992.

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

[21] J. Matoušek, N. Miller, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. In *Proc. IEEE FOCS*, pages 49–58, 1991.

[22] B. Nelson and P. K. Khosla. Integrating sensor placement and visual tracking strategies. In *Proc. IEEE ICRA*, pages 1351–1356, May 1994.

[23] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, 1987.

[24] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. Technical Report 215, Department of Computer Science, New York University, Courant Institute of Mathematical Sciences, Apr. 1986.

[25] W. J. Rucklidge. *Efficient Computation of the Minimum Hausdorff Distance for Visual Recognition.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, NY, Jan. 1995.

[26] R. Sharma and S. Hutchinson. On the observability of robot motion under active camera control. In *Proc. IEEE ICRA*, pages 162–167, May 1994.

[27] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proc. ACM Symp. on Comp. Geom.*, pages 14–23, 1986.

[28] K. Tarabanis and R. Y. Tsai. Computing occlusion-free viewpoints. In *CVPR*, pages 802–807, 1992.

[29] S. J. Teller. Computing the antipenumbra of an area light source. *Computer Graphics (Proceedings SIGGRAPH '92)*, 26(2):139–148, July 1992.

[30] E. Welzl. Constructing the visibility graph for n line segments in $O(n^2)$ time. *Information Processing Letters*, 20:167–171, 1985.

[31] H. Zhang. Optimal sensor placement. In *Proc. IEEE ICRA*, pages 1825–1830, May 1992.