Comparison of Constrained Geometric Approximation Strategies for Planar Information States

Yang Song Jason M. O'Kane

***Abstract*— This paper describes and analyzes a new technique for reasoning about uncertainty called *constrained geometric approximation* (CGA). We build upon recent work that has developed methods to explicitly represent a robot's knowledge as an element, called an information state, in an appropriately defined information space. The intuition of our new approach is to constrain the I-state to remain in a structured subset of the I-space, and to enforce that constraint using appropriate over-approximation methods. The result is a collection of algorithms that enable mobile robots with extreme limitations in both sensing and computation to maintain simple but provably meaningful representations of the incomplete information available to them. We present a simulated implementation of this technique for a sensor-based navigation task, along with experimental results for this task showing that CGA, compared to a high-fidelity representation of the un-approximated I-state, achieves a similar success rate at a small fraction of the computational cost.**

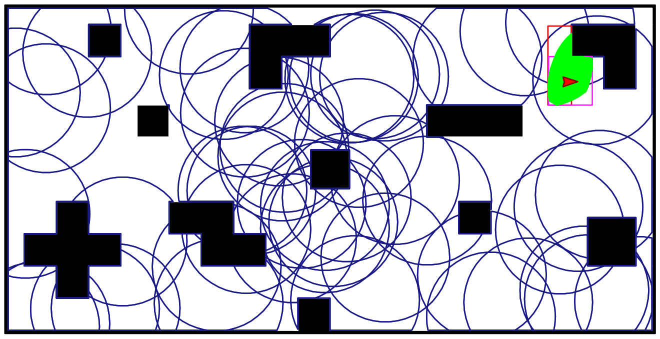


Fig. 1. A mobile robot navigating through its environment by observing landmarks. The robot represents its knowledge about its own position using a double rectangle approximated information state.

I. Introduction

Mobile robots struggle constantly against uncertainty. Whether this uncertainty arises from noisy and incomplete sensing or from motions that are not fully predictable, a robot's success in reasoning about and managing its incomplete information plays a major role in determining its overall effectiveness. At the same time, for applications in which the robot's size, mass, or energy resources are strictly limited (including, for example, micro robots, airborne robots, and mobile sensor networks), computation power remains a scarce resource. The goal of this paper is to describe and evaluate a new technique called *constrained geometric approximation (CGA)* that enables robots to trade precise representation of uncertainty for computational efficiency when dealing with two-dimensional state information.

This approach builds upon the large body of robotics work that uses set-based representations of uncertainty [5], [9], [13]. The intuition of these methods is to maintain a set, called here an *information state (I-state)*, of "possible states" that are consistent with the robot's history of actions and observations. The robot can then use this set directly for decision making. Maintaining such sets requires appropriate geometric algorithms to perform updates when the robot moves, and when it receives sensor data. The intuition of our approach is to accelerate these potentially time-consuming operations by maintaining only an *overapproximation* of the true information state, and constraining this approximation to have a simple, well-behaved geometric form. Note that although the quality of such approximations is well-known to degrade as the dimension of the underlying space increases, we believe that two-dimensional contexts are sufficiently prevalent in mobile robotics applications to motivate special attention to this case.

A crucial building block for this kind of filter is the selection of a *range space* containing the geometric primitives that the robot can use to represent its I-state. We describe a set of operations that the robot must be able to perform on the elements of its range space. In principle, any set of planar figures for which these operations can be computed is a suitable range space. However, intuition suggests a tradeoff between the expressivity of the range space and the computational cost of executing those operations to maintain information states under that range space. As a result, this paper considers several different range spaces with varying levels of expressivity, including a new range space we call the *double rectangle space*. We also compare our results against a high-fidelity polygonal representation to assess the amount of inaccuracy incurred by the overapproximation process.

To evaluate its effectiveness, we simulated this technique on a landmark-based navigation task, in which a robot moves through a series of waypoints with the assistance of a sensor that detects the presence of (but not the direction or distance to) a collection of fixed landmarks along the obstacle boundaries. In our experiments, we found that our approximated I-states achieve task completion rates comparable to high-fidelity I-state representations at a fraction of the computational cost.

Prior work by the authors [12], [14]–[16] has used preliminary versions of the constrained geometric approximation method using specific, fixed range spaces. The new contributions of this paper are (1) a careful formulation of the

Yang Song and Jason M. O'Kane are with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St., Columbia, SC 29208, USA. e-mail: {song24, jokane}@cse.sc.edu.

operations that the range space must support, (2) a collection of geometric algorithms for carrying out those operations for a new range space in which each element is the union of two axis-aligned rectangles, and (3) a series of experiments that measure computation time, approximation quality, and task completion rates for several different range spaces.

The remainder of the paper has the following structure. After reviewing some related research in Section II, we introduce our problem formulation in Section III. A definition and some examples for the Constrained Geometric Approximation approach appear in Section IV, followed in Section V by descriptions of update algorithms for rectangle and double rectangle range spaces. Finally, we present simulation results in Section VI and a preview of future work appears along with some concluding remarks in Section VII.

II. Relationship to prior work

The basic goal of our efforts is to understand how mobile robots can represent and reason about uncertainty in their own states. A common thread of much of the research on this front in the robotics community has been the use of probabilistic methods [7], [17]–[20], [22]. However, such methods are often computationally expensive. For example, the standard Monte Carlo particle filter approach for localization [21] is often implemented with hundreds or thousands of particles, which can be prohibitively time- and memory-consuming on robot platforms with strong limitations on computation power. Moreover, in cases for which the number of particles is kept relatively small, it can sometimes be difficult for the robot to discern whether sufficiently many particles are in use to ensure convergence to the correct localization solution. In contrast to particle filtering, our approach is well-suited to platforms with severely limited computational resources. In particular, because the approximated I-state we maintain is guaranteed to be an overapproximation of the true I-state, the robot can directly determine when the information it represents is not sufficiently detailed to be useful.

Another branch of robotics, which has described itself as a "minimalist" approach, considers algorithms that solve robotic tasks in spite of limitations in the robot's sensing capability [1]–[4], [6], [8], [10], [11], [23]. However, these methods are generally concerned with limitations in sensing, but not in computation. In fact, the I-state computations they propose are often quite complex [24], [26], [27]. The new contribution of this work is to show that such precise I-states are not always necessary for the robot to complete its task, and that reasonable constrained approximations can be computed much more efficiently.

III. Problem statement

This section describes the basic model and notation we use throughout the remainder of the paper.

A. Basic ingredients

We consider robot models with the following elements:

- A division of time into discrete stages, numbered $k = 1, 2, \ldots$.
- A planar state space $X = \mathbb{R}^2$, in which the location of the robot at stage k is denoted x_k. The state space is partitioned into *free space* X_{free} and *obstacle space* X_{obst}.
- A non-empty *action space* U. The robot chooses one action $u_k \in U$ to execute in each stage.
- A set-valued *state transition function* $F : X \times U \to \text{pow}(X)$, in which $\text{pow}(X)$ denotes the power set of X, that describes how the state changes. Because the output of this function is a set of states (rather than a single, fully predictable state), we can model unpredictable state changes resulting from noise or from unknown actions of other agents. Specifically, we define $x_{k+1} \in F(x_k, u_k)$. In this paper, we consider a specific form for the transition function in which the actions are additive, uncertainty arises from additive noise, and the robot cannot pass through the obstacles:
$$F(x_k, u_k) = \{x_k + u_k + \theta_k \mid \theta_k \in \Theta(u_k)\} \cap X_{\text{free}}.$$
Here $\Theta(u_k)$ denotes a bounded set of possible noise values, which may depend on the action the robot selects.
- A non-empty *observation space* Y, so that $y_k \in Y$ models the sensor information collected by the robot at stage k.
- A set-valued *observation function* $h : X \to \text{pow}(Y)$ that describes how the observation y_k is determined by the current state x_k. As with the state transition function, the set-valued nature of this function allows us to model sensing that is not fully predictable given the state. From the observation function, we also define the notion of an *observation preimage*, which denotes the set of states from which a given observation might be obtained:
$$H(y_k) = \{x_k \in X \mid y_k \in h(x_k), y_k \in Y\}. \qquad (1)$$
- An *initial condition* $\eta_0 \subseteq X$, indicating a set of possible starting states. This allows us to model the information available to the robot at the start of its execution.

The key limitation to the robot in this kind of scenario is that the current state cannot be observed directly. Instead, the robot must rely on other cues to draw conclusions about the state. Specifically, the robot must base its decisions only on the initial condition η_0, the history of actions $u_1, \ldots u_{k-1}$ it has executed, and the history of observations $y_1, \ldots, y_k$ it has received.

B. Information states

Based on history described above, at each stage, the robot can compute the set of "possible states". The following two definitions clarify this idea.

Definition 1: A state $x_k \in X$ is *consistent with* a sensor-action history $(y_1, u_1, \ldots, y_{k-1}, u_{k-1}, y_k)$ if there exists some state sequence $x_1, \ldots, x_{k+1} \in X$ such that $x_1 \in \eta_0$, and
$$x_{i+1} \in F(x_i, u_i) \qquad (2)$$

for each $i = 1, \dots, k-1$, and

$$y_i \in h(x_i) \tag{3}$$

for each $i = 1, \dots, k$. □

Definition 2: The *information state* (I-state) η_k at stage k is the set of all states consistent with the robot's sensor-action history. The *information space* (I-space) $\mathcal{I}$ is the powerset of X, which contains all possible I-states. □

The intuition is that η_k contains every state that the robot might possibly be in, given the information available to it. A detailed description of the computations required to compute η_{k+1} given η_k, u_k, and y_k appears in LaValle's book [9]. The basic approach involves an "expansion" to account for any possible changes in the state resulting from action u_k, followed by an intersection with the observation pre-image $H(y_k)$:

$$\eta_{k+1} = \left[\bigcup_{x_k \in \eta_k} F(x_k, u_k) \right] \cap H(y_k). \tag{4}$$

The robot can then execute plans that use feedback on the I-space, of the form $\pi : \mathcal{I} \to U$, so that $u_k = \pi(\eta_k)$.

IV. Constrained Geometric Approximation

This section introduces our CGA approach for maintaining approximated I-states. This method is motivated by the fact that the changes to the robot's I-states in each stage, as described in Equation 4, can be prohibitively expensive to compute directly.

A. Definition

The intuition of our approach is to maintain, instead of η_k itself, only an overapproximation A_k of η_k, so that

$$\eta_k \subseteq A_k. \tag{5}$$

Note in particular that, because $x_k \in \eta_k$, we also have $x_k \in A_k$.

The specific approximation scheme we employ is to select a range space $\mathcal{R} \subseteq \mathcal{I}$ within the I-space, and constrain our approximated I-space to remain always a member of this range space, so that $A_k \in \mathcal{R}$. The advantage of this scheme is that *I-state updates are often more efficient on range space elements than on general I-states.* The specific requirements on the range space are detailed in the definition below.

Definition 3: A range space $\mathcal{R} \subseteq \mathcal{I}$ is a set of I-states, equipped with two functions:

i) An *approximate action update function* $T : \mathcal{R} \times U \to \mathcal{R}$, such that if $\eta_k \subseteq A_k$, then

$$\bigcup_{x_k \in \eta_k} F(x_k, u_k) \subseteq T(A_k, u_k). \tag{6}$$

ii) An *approximate observation update function* $O : \mathcal{R} \times Y \to \mathcal{R}$, such that if $\eta_k \subseteq A_k$, then

$$\eta_k \cap H(y_k) \subseteq O(A_k, u_k). \tag{7}$$

□

In particular, notice that, given a range space with its update functions, if we start with $A_0 = \eta_0$ and in subsequent stages compute A_{k+1} from A_k, u_k, and y_k according to

$$A_{k+1} = O(T(A_k, u_k), y_k), \tag{8}$$

then by induction we have $\eta_k \subseteq A_k$ at each stage k.

B. Examples

First, we consider two "typical" options for the range space.

Example 1: Let $\mathcal{R}_{\text{disk}}$ denote the set of all disks in $\mathbb{R}^2$, each parameterized by its center and radius. For any compact set $S \subset \mathbb{R}^2$, let $\textsc{Sed}(S)$ denote the smallest disk enclosing S. Then we can define

$$T_{\text{disk}}(A_k, u_k) \quad = \quad A_k \oplus \{u_k\} \oplus \textsc{Sed}(\Theta(u_k)), \tag{9}$$

in which $\oplus$ denotes the Minkowski sum operation. Notice that, because both A_k and $\textsc{Sed}(\Theta(u_k))$ are disks, the result is a member of $\mathcal{R}_{\text{disk}}$. Similarly, we define

$$O_{\text{disk}}(A_k, y_k) = \textsc{Sed}(H(y_k) \cap A_k). \tag{10}$$

It is straightforward to see that $\mathcal{R}_{\text{disk}}$ is a range space under those operations. Computing Minkowski sums of disks consists of the addition of their centers and radii, so it is trivial to compute T_{disk}. Algorithms are known to evaluate O_{disk} in time $O(1)$ when the observation preimages are quarterplanes [12] or circles [15]. □

Example 2: Let $\mathcal{R}_{\text{rect}}$ denote the set of all axis-aligned rectangles in $\mathbb{R}^2$, each parameterized by its lower-left corner a and its upper right corner b. For any compact set $S \subset \mathbb{R}^2$, let $\textsc{aabb}(S)$ denote the smallest axis-aligned rectangle enclosing S, that is, its "axis-aligned bounding box." Then we can define

$$\begin{aligned} T_{\text{rect}}(A_k, u_k) = \textsc{aabb}(X_{\text{free}} \cap \\ [A_k \oplus \{u_k\} \oplus \textsc{aabb}(\Theta(u_k))]) \end{aligned} \tag{11}$$

in which $\oplus$ denotes the Minkowski sum operation. For the approximate observation update function, we use

$$O_{\text{rect}}(A_k, y_k) = \textsc{aabb}(H(y_k) \cap A_k). \tag{12}$$

Directly from the definitions, we can show that $\mathcal{R}_{\text{rect}}$ is a range space under T_{rect} and O_{rect}. Details about the algorithms needed to compute T_{rect} and O_{rect} appear in Section V-A. □

The next two examples illustrate two extremes allowed by Definition 3.

Example 3: Let $\mathcal{R}_{\text{ident}} = \mathcal{I}$ denote the I-space itself. Using the update functions

$$T_{\text{ident}}(A_k) = \bigcup_{x_k \in A_k} F(x_k, u_k) \tag{13}$$

and

$$O_{\text{ident}}(A_k, y_k) = A_k \cap H(y_k), \qquad (14)$$

clearly $\mathcal{R}_{\text{ident}}$ is a range space. Note however, that computing A_{k+1} using these functions is just as computationally expensive as computing η_k itself, so nothing is gained from the (vacuous) approximation in this case. □

Example 4: Let $\mathcal{R}_{\text{triv}} = \{X\}$ denote a singleton set whose only element is the entire state space. Define

$$T_{\text{triv}}(A_k, u_k) = O_{\text{triv}}(A_k, y_k) = X \qquad (15)$$

for all u_k and y_k. There functions are trivial to compute, and clearly they make $\mathcal{R}_{\text{triv}}$ is a range space. However, because this range space essentially discards all of the information available to the robot, its approximated I-states are not useful for selecting actions. □

Examples 3 and 4 illustrate an apparent tradeoff between faithfulness to the underlying true I-state η_k (seen in Example 3) and efficient computation (seen in Example 4). In this paper, we consider several different range spaces between these two extremes, including $\mathcal{R}_{\text{disk}}$, $\mathcal{R}_{\text{rect}}$, and a new range space called the double-rectangle space $\mathcal{R}_{\text{dblrect}}$.

To evaluate the quality of the approximation throughout the robot's execution, we can compare the area of the approximated I-state to the area of the I-state itself:

$$Q_k = \frac{1}{k} \sum_{i=1}^{k} \frac{\text{area}(\eta_i)}{\text{area}(A_i)} \qquad (16)$$

Note that higher values for Q demonstrate better approximation quality, up to a maximum value of 1. We present, in Section VI, experiments that measure these approximation ratios, along with the robot's success rate in completing its task using each of these range spaces.

V. Rectangle and double rectangle range space updates

This section presents algorithms for computing the update functions T_{rect} and O_{rect} for the rectangle range space $\mathcal{R}_{\text{rect}}$ (in Section V-A) and introduces the double rectangle range space $\mathcal{R}_{\text{dblrect}}$ and its update functions T_{dblrect} and O_{dblrect} (in Section V-B).

A. Updating rectangle approximated I-states

In the rectangle range space, to compute the approximate transition function T_{rect}, as defined in Equation 11, we proceed in four steps. Figure 2 illustrates these steps.

i) First, we compute $\text{AABB}(\Theta(u_k))$. In the general case, in which $\Theta(u_k)$ is represented by a list of vertices on its boundary, this requires simply selecting the extremal coordinates in each direction from this vertex list.

ii) Second, we compute the Minkowski sum $A_k \oplus \{u_k\} \oplus \text{AABB}(\Theta(u_k))$, using the result of Step 1. Because each of the three operands is either a single point or a rectangle represented by its lower left and upper right corners, the results can be computed by adding the coordinates of the respective corners together.

iii) Third, we intersect the result of Step 3 with X_{free}, using standard geometric algorithms for boolean operations on polygons [25]. The resulting set is a polygonal region that contains η_k, but is not necessarily a rectangle.

iv) Finally, to enforce the constraint that $A_k \in \mathcal{R}_{\text{rect}}$, we compute the axis-aligned bounding box of the result of Step 3. This uses the same algorithm as in Step 1.

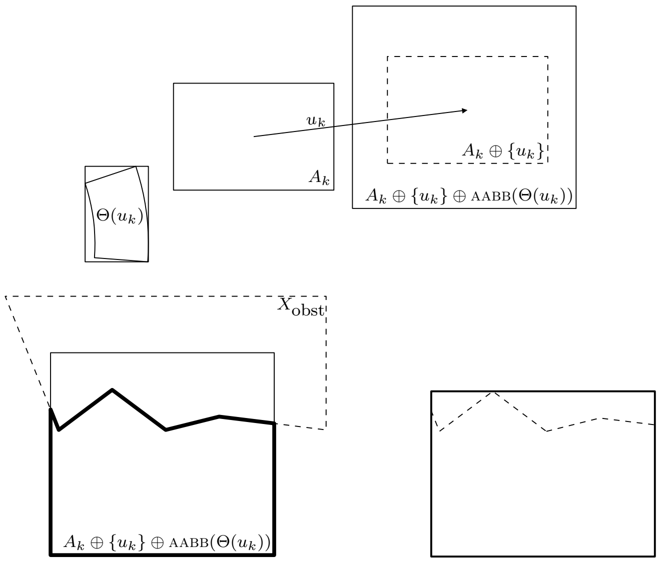

u_k
A_k
$\Theta(u_k)$
$A_k \oplus \{u_k\}$
$A_k \oplus \{u_k\} \oplus \text{AABB}(\Theta(u_k))$
X_{obst}
$A_k \oplus \{u_k\} \oplus \text{AABB}(\Theta(u_k))$

Fig. 2. Four steps to compute T_{rect}. [top] Steps 1 and 2. [bottom left] Step 3. [bottom right] Step 4.

To compute the approximate observation update function O_{rect} for general observation preimage shapes we follow a similar process, but based on Equation 12 instead of Equation 11. Note, however, that if the observation preimages have a specific known shape, the use of a specialized algorithm can accelerate the process. For example, in our experiments, every $H(y_k)$ is a planar disk. In that case we can compute $O_{\text{rect}}(A_k, y_k)$ in constant time using three steps:

i) First, we compute the set of points at which the boundary of the disk $H(y_k)$ intersects the boundary of the rectangle A_k. There are at most 8 such points.

ii) Second, we consider each of the four extremal points of the disk $H(y_k)$: its topmost, bottommost, leftmost, and rightmost points. For each, we test whether it is contained in A_k.

iii) Finally, we find the smallest axis aligned rectangle that contains the 8 or fewer points found in Steps 1 and 2.

See Figure 3.

B. Double rectangle approximated I-states

Both of the "typical" range spaces introduced above, $\mathcal{R}_{\text{disk}}$ and $\mathcal{R}_{\text{rect}}$, have the property that all of the approximated I-states they allow are convex. In cases where the true I-

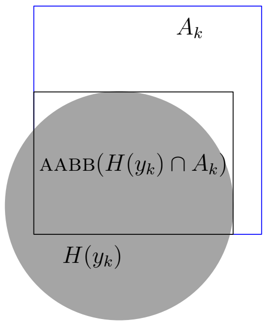


Fig. 3. Computing $O_{\text{rect}}(A_k, y_k)$ to find the rectangle approximated I-state, given an observation y_k.

state η_k is nonconvex, this presents a severe limitation to the ability of A_k to closely approximate η_k.

To overcome this limitation, we propose a more expressive range space that contains non-convex approximated I-states. Specifically, we consider a range space of *double rectangles*:

$$\mathcal{R}_{\text{dblrect}} = \{R_1 \cup R_2 \mid R_1, R_2 \in \mathcal{R}_{\text{rect}}\} \tag{17}$$

The intution is that each $A_k \in \mathcal{R}_{\text{dblrect}}$ is the union of two rectangles, which allows the approximated I-state to correctly represent many typical nonconvexities that occur in η_k, including the case in which η_k "grows" around a reflex vertex of an environment obstacle.

The central problem in using this range space is that there is no obvious analog to the SED function we used to keep A_k in $\mathcal{R}_{\text{disk}}$, nor to the AABB function we used to A_k in $\mathcal{R}_{\text{rect}}$. Instead, we propose a method, called DRAP for "double rectangle around polygon", which accepts a polygonal region of the plane as input, and produces a small double rectangle containing that polygon as output. This algorithm attempts to keep the size of the resulting double rectangle as small as possible, but for efficiency reasons does not guarantee to find the smallest polygon that covers the given polygon.

Algorithm 1 shows pseudocode for this process, and Figure 4 shows an example of the output. The idea is to build the double rectangle from the bottom up, starting with two degenerate rectangles, each of which is a single point at either a vertex of the polygon or at its centroid, as "seeds". Then an iterative process considers each edge of the polygon in turn, and expands one of the two rectangles to contain that edge. At each step, we choose between the two possible expansions by greedily preferring the rectangle to expand that results in the smallest total area. Finally, to ensure that the result does not contain any unnecessary overapproximation, we repeat this process over all pairs of potential seed points, and retain only the smallest overall double rectangle, as measured by its area. Notice that, since the resulting double rectangle contains every boundary edge of P, it is indeed an overapproximation of P. For a polygon with n vertices, the outer loop runs $n(n+1)$ times and the inner loop runs n times, thus, this algorithm takes $O(n^3)$ time.

Based on Algorithm 1, we can define the range space

Algorithm 1 DOUBLERECTANGLEAROUNDPOLYGON(P)

```
$V \leftarrow$ set containing the vertices of $P$ and its centroid
**for** each pair of vertices $p, q \in V, p \neq q$ **do**
    $R_1 \leftarrow \{p\}$; $R_2 \leftarrow \{q\}$
    **for** each edge $e$ of $P$ **do**
        $R'_1 \leftarrow \textsc{aabb}(R_1 \cup e)$
        $R'_2 \leftarrow \textsc{aabb}(R_2 \cup e)$
        **if** $\text{area}(R'_1 \cup R_2) < \text{area}(R_1 \cup R'_2)$ **then**
            $R_1 \leftarrow R'_1$
        **else**
            $R_2 \leftarrow R'_2$
        **end if**
    **end for**
    insert $R_1 \cup R_2$ into $C$
**end for**
**return** $\text{argmin}_{(R_1 \cup R_2) \in C}(\text{area}(R_1 \cup R_2))$
```

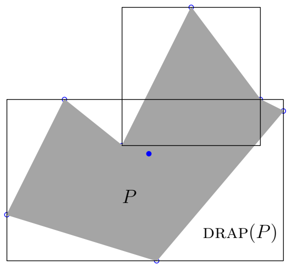


Fig. 4. Algorithm 1 returns a double rectangle $\textsc{drap}(P)$ around the given input polygon P.

operations on $\mathcal{R}_{\text{dblrect}}$ in a manner similar to those for $\mathcal{R}_{\text{rect}}$. For a double rectangle approximated I-state $A_k = R_1 \cup R_2$, we have

$$T_{\text{dblrect}}(A_k, u_k) = \textsc{drap}(X_{\text{free}} \cap [A_k \oplus \{u_k\} \oplus \textsc{drap}(\Theta(u_k))]), \tag{18}$$

and

$$O_{\text{dblrect}}(A_k, y_k) = \textsc{aabb}(H(y_k) \cap R_1) \cup \textsc{aabb}(H(y_k) \cap R_2). \tag{19}$$

Algorithms to evaluate these functions have the same form as the analogous functions for $\mathcal{R}_{\text{rect}}$.

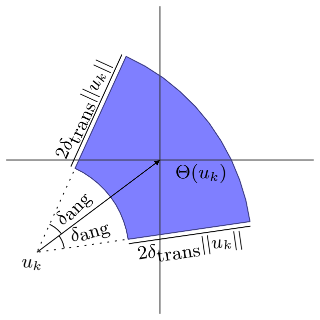
VI. CGA FOR LANDMARK-BASED NAVIGATION

This section describes an implementation in simulation of the CGA approach for a landmark-based mobile robot navigation task.

A. Task description

A point robot moves through a known environment E, which is populated with both obstacles and free space. The robot's action space is $U = B((0,0); v_{\max})$, a closed ball of radius $v_{\max}$, centered at the origin. The robot's actions can be interpreted as commanded displacement vectors.

We assume that noise influences both the direction and magnitude of the robot's motion. The angular noise is



$2\delta_{\text{trans}}||u_k||$
$\Theta(u_k)$
δ_{ang}
δ_{ang}
u_k
$2\delta_{\text{trans}}||u_k||$

Fig. 5. The noise model $\Theta(u_k)$ used in our experiments. It considers angular error bounded by δ_{ang} and translational error bounded by $\delta_{\text{trans}}||u||$.

bounded by a given angle δ_{ang}, and the translation noise is bounded by $\delta_{\text{trans}}||u||$. Therefore, the transition noise set $\Theta(u_k)$ is a slice of an annulus, as shown in Figure 5.

To guide its motions, the robot has a sensor that can detect the presence of any of a collection of known landmarks $l_1, \ldots, l_m$ along the obstacle boundaries, but not the direction or distance to those landmarks. Each landmark l_i has an detection range r_i, and the robot can detect any of the landmarks for which $x_k \in B(l_i; r_i)$. The observation preimages $H(y_k)$ are therefore disks.

The robot is given a sequence of waypoints $w_1, \ldots, w_n \in X_{\text{free}}$, such that each waypoint is visible in X_{free} from both its predecessor and its successor. The goal is to visit each of the waypoints in their given sequence. An execution is considered successful if it achieves this goal within a fixed time limit; an execution is declared a failure if the robot takes too long to reach the final waypoint, or if it collides with an obstacle.

For every range space, the robot aims to follow the waypoints in order guided by the centroid point of the approximated I-state. In other words, we define a plan that uses a feedback on $\mathcal{R}$ as form $\pi : \mathcal{R} \to U$ so that $u_k = \pi(A_k)$. We denote the next waypoint by w and select an action that moves toward w from the centroid of A_k:

$$\pi(A_k) = \frac{w - \text{centroid}(A_k)}{||w - \text{centroid}(A_k)||} v_{\text{max}}. \tag{20}$$

B. Experimental Results

To verify the effectiveness and efficiency of CGA for this navigation task, we have conducted experiments using three distinct environments, and three distinct range spaces $\mathcal{R}_{disk}$, $\mathcal{R}_{rect}$, and $\mathcal{R}_{dblrect}$. The experiments are implemented in C++. All the simulations run on a GNU/Linux PC with Intel Core i7 CPU, 2.8GHz and 8GB memory.

Figures 6, 7, and 8 depict the three environments used in our tests. For each environment, we are interested in:

- the relationship between task completion and number of landmarks,
- the time required to compute approximated I-state compared to the real I-state,
- the approximation ratio (Equation 16), and

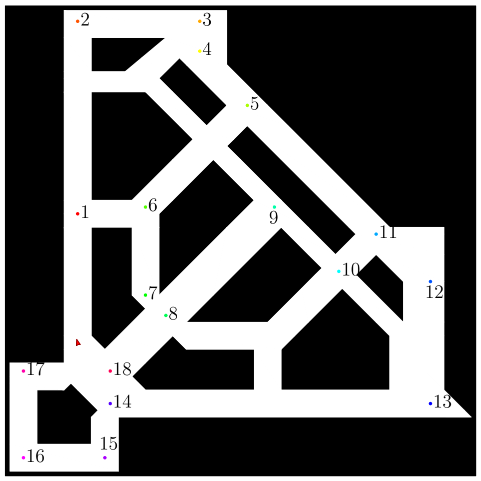


Fig. 6. An office-like environment. The waypoints are numbered.

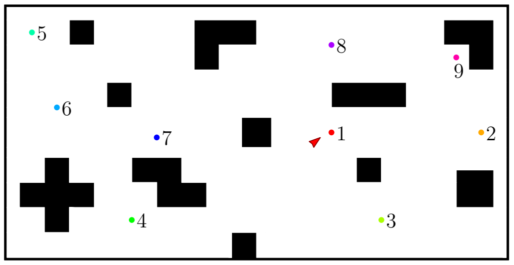


Fig. 7. An environment cluttered with obstacles, along with its waypoints.

- the robustness of the approximation scheme to the rotations of the environment.

For all of the experiments we have some consistent assumptions. The positions of the landmarks are pseudo-randomly generated inside X_{free}. We set the noise model angular error bound to $\delta_{\text{ang}} = 0.4$ and the transition bound to $\delta_{\text{trans}} = 0.2$.

In each of these three environments, we performed a series of three experiments, described below.

i) First, we varied the number of landmarks N between 5 and 250 in increments of 5 and calculated the robot's success rate. For each N, we repeated the experiment 15 times with different landmark distributions by assigning distinct random seeds. The success rate is the number of times that task completes successfully divided by the total number of attempts. Figures 9, 10, and 11 plot the results.
From these results, we conclude that the approximated I-states, especially using $\mathcal{R}_{\text{rect}}$ and $\mathcal{R}_{\text{dblrect}}$, achieve similar performance to the true I-state. We attribute the frequent failures for $\mathcal{R}_{\text{disk}}$ to the fact that T_{disk} does not take the obstacles into account. We omitted this step from T_{disk} because computing SED for complex obstacle shapes is computationally expensive. The results in Figure 9 show weaker performance for CGA than the other environments, which we attribute to the narrow winding corridors in this environment. How-

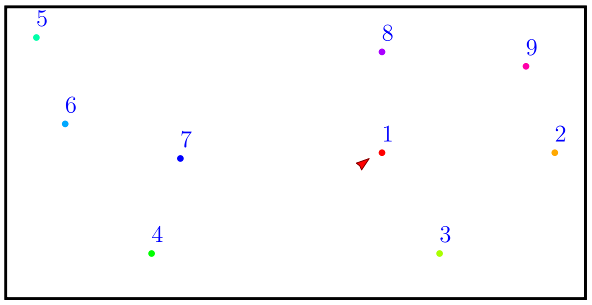


Fig. 8. An obstacle-free environment with 9 waypoints.

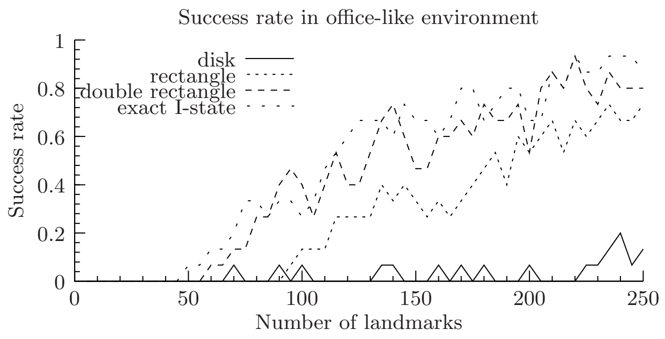


Fig. 9. Success rate vs number of landmarks for the environment in Fig. 6.

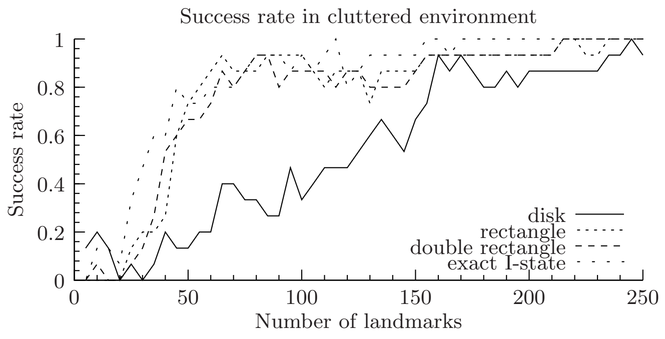


Fig. 10. Success rate vs number of landmarks for the environment in Fig. 7.

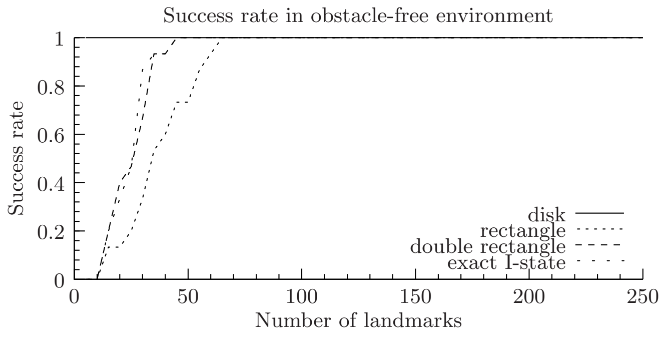


Fig. 11. Success rate vs number of landmarks for the environment in Fig. 8.

ever, the results for that environment do demonstrate the advantage of the double rectangle approximated I-states over the other range spaces.

ii) Our second experiment compares the computation time for our approximated I-states to the coo responding computation for the exact I-states. We used $N = 300$ landmarks and executed 10 trials for each combination of range space and environment. The results are collected in Figures 12 13, and 14 for the results, which show a substantial reduction in computation time resulting from the range space approximations. In these same trials, we also computed the approximation ratio Q as defined in Equation 16. The results show, as expected, the the double rectangle range space provides the highest approximation quality.

iii) Finally, we performed experiments to evaluate the impact of the selection of axes. This consideration is important because both $\mathcal{R}_{\text{rect}}$ and $\mathcal{R}_{\text{dblrect}}$ use only axis-aligned rectangles. We applied rotations between 0 and 2π radians each environment, in increments of $\pi/12$, and conducted 10 trials with 250 landmarks for each case. Across these trials, we computed the success rates, which appear in Figures 15 and 16. We not plot the results for the obstacle-free environment because the robot succeeded in every trial. These results illustrate that there is no substantial degradation of CGA's performance as a function of environment rotation.

VII. Conclusion

This paper presented a method for representing a robot's uncertain information about the current state using constrained geometric approximation and demonstrated the effectiveness of that approach.

However, it remains as future work to consider additional range spaces. In particular, one of the innovations in this paper was to approximate the robot's I-state as the union of two rectangles. Notice, however, that this approach can be generalized to k-fold unions of rectangles, without any substantial changes to the algorithm. In future work, we will investigate the tradeoffs between accuracy and efficiency that are exposed by using larger numbers of rectangles in the approximated I-state. We are also considering range spaces that allow more general polygonal shapes, but maintain efficiency by limiting their number of vertices, and by applying appropriate expansion operations to enforce this limit.

Finally, we note that there may also be some advantage to algorithms that also generate provable *under*-approximates of the I-state. The discrepancy between the over- and under-approximations could then be used by the robot to estimate the quality of its representation.

Acknowledgments

We gratefully acknowledge support for this work from the U. S. National Science Foundation (IIS-0953503).

References

[1] E. U. Acar and H. Choset, "Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of

Range Space	Time (s)	Approximation Ratio
$\mathcal{R}_{disk}$	0.292	0.220
$\mathcal{R}_{rect}$	0.415	0.661
$\mathcal{R}_{dblrect}$	1.491	0.720
$\mathcal{I}$	26.895	1.000

Fig. 12. Average computation time and approximation approximation ratio for the environment in Figure 6.

Range Space	Time (s)	Approximation Ratio
$\mathcal{R}_{disk}$	0.162	0.155
$\mathcal{R}_{rect}$	0.441	0.632
$\mathcal{R}_{dblrect}$	1.122	0.691
$\mathcal{I}$	10.218	1.000

Fig. 13. Average computation time and approximation approximation ratio for the environment in Figure 7.

Range Space	Time (s)	Approximation Ratio
$\mathcal{R}_{disk}$	0.163	0.155
$\mathcal{R}_{rect}$	0.396	0.642
$\mathcal{R}_{dblrect}$	1.021	0.684
$\mathcal{I}$	10.074	1.000

Fig. 14. Average computation time and approximation approximation ratio for the environment in Figure 8.

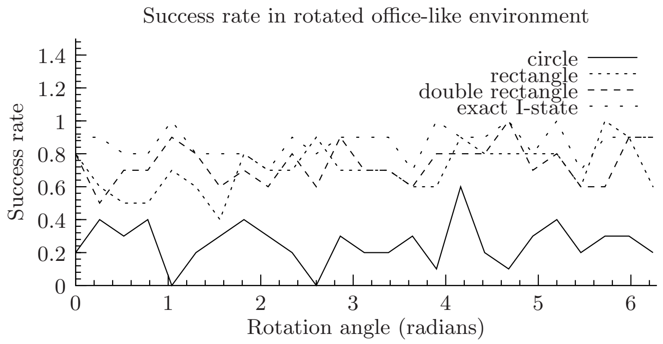


Fig. 15. Success rate vs rotation of the environment for the environment in Figure 6.

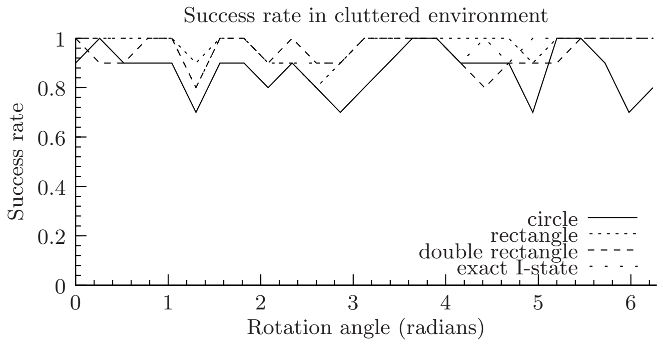


Fig. 16. Success rate vs rotation of the environment for the environment in Figure 7.

critical points and voronoi diagrams," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.
[2] S. Akella and M. Mason, "Posing polygonal objects in the plane by pushing," *International Journal of Robotics Research*, vol. 17, no. 1, pp. 70–88, Jan. 1998.
[3] A. Blum, P. Raghavan, and B. Schieber, "Navigating in unfamiliar geometric terrain," *SIAM Journal on Computing*, vol. 26, no. 1, pp. 110–137, 1997.
[4] H. Choset and J. Burdick, "Sensor based motion planning: Incremental construction of the hierarchical generalized Voronoi graph," *International Journal of Robotics Research*, vol. 19, no. 2, pp. 126–148, 2000.
[5] M. Erdmann and M. T. Mason, "An exploration of sensorless manipulation," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, Aug. 1988.
[6] M. A. Erdmann, "Using backprojections for fine motion planning with uncertainty," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 19–45, 1986.
[7] P. Jensfelt and S. Kristensen, "Active global localisation for a mobile robot using multiple hypothesis tracking," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 748–760, Oct. 2001.
[8] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 6, pp. 814–822, Dec. 1997.
[9] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.
[10] A. Lazanas and J. C. Latombe, "Landmark-based robot navigation," in *Proc. National Conference on Artificial Intelligence (AAAI)*, 1992.
[11] V. J. Lumelsky and S. Tiwari, "An algorithm for maze searching with azimuth input," in *Proc. IEEE International Conference on Robotics and Automation*, 1994, pp. 111–116.
[12] J. M. O'Kane, "On the value of ignorance: Balancing tracking and privacy using a two-bit sensor," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2008.
[13] J. M. O'Kane and S. M. LaValle, "On comparing the power of robots," *International Journal of Robotics Research*, vol. 27, no. 1, pp. 5–23, Jan. 2008.
[14] J. M. O'Kane and W. Xu, "Energy-efficient target tracking with a sensorless robot and a network of unreliable one-bit proximity sensors," in *Proc. IEEE International Conference on Robotics and Automation*, 2009.
[15] ——, "Network-assisted target tracking via smart local routing," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.
[16] J. M. O'Kane, "Decentralized tracking of indistinguishable targets using low-resolution sensors," in *Proc. IEEE International Conference on Robotics and Automation*, 2011.
[17] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," in *Proc. International Joint Conferences on Artificial Intelligence*, 1995, pp. 1080–1087.
[18] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Probabilistic algorithms and the interactive museum tour-guide robot Minerva," *International Journal of Robotics Research*, vol. 19, no. 11, pp. 972–999, 2000. [Online]. Available: http://ijr.sagepub.com/cgi/content/abstract/19/11/972
[19] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning*, pp. 1–25, Apr. 1998.
[20] ——, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.
[21] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.
[22] M. Tomono and S. Yuta, "Mobile robot localization based on an inaccurate map," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 399–404.
[23] B. Tovar, L. Guilamo, and S. M. LaValle, "Gap Navigation Trees: Minimal representation for visibility-based tasks," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2004.
[24] B. Tovar and S. M. LaValle, "Visibility-based pursuit-evasion with bounded speed," *International Journal of Robotics Research*, vol. 27, pp. 1350–1360, Nov. 2008.
[25] B. R. Vatti, "A generic solution to polygon clipping," *Communications of the ACM*, vol. 35, no. 7, pp. 56–63, July 1992.
[26] J. Yu and S. M. LaValle, "Tracking hidden agents through shadow information spaces," in *Proc. IEEE International Conference on Robotics and Automation*, 2008.
[27] ——, "Probabilistic shadow information spaces," in *Proc. IEEE International Conference on Robotics and Automation*, 2010.