Sensor Beams, Obstacles, and Possible Paths

Benjamin Tovar[1], Fred Cohen[2], and Steven M. LaValle[1]

[1] Department of Computer Science, University of Illinois. Urbana, IL 61801 USA. {btovar,lavalle}@uiuc.edu
[2] Department of Mathematics, University of Rochester. Rochester, NY 14627 USA. fcohen@rochester.edu

WAFR Submission: 15 July 2008

Summary. This paper introduces a problem in which an agent (robot, human, or animal) travels among obstacles and binary detection beams. The task is to determine the possible agent path based only on the binary sensor data. This is a basic filtering problem encountered in many settings, which may arise from physical sensor beams or virtual beams that are derived from other sensing modalities. Methods are given for three alternative representations: 1) the possible sequences of regions visited, 2) path descriptions up to homotopy class, and 3) numbers of times winding around obstacles. The solutions are adapted to the minimal sensing setting; therefore, precise estimation, distances, and coordinates are replaced by topological expressions. Applications include sensor-based forensics, assisted living, security, and environmental monitoring.

1 Introduction

Imagine installing a bunch of cheap, infrared eye beams throughout a complicated warehouse, office, or shopping center; see Figure 1. Just like the safety beam on a motorized garage door, a single bit of information is provided: Is the beam currently obstructed? Now suppose that there are one or more moving bodies, which could be people, robots, animals, and so on. If the beams are distinguishable and we know the order in which beams were crossed, what can we infer about the paths taken by the moving bodies? This may be considered as a filtering problem, but with minimal, combinatorial information, in contrast to popular Kalman filters and particle filters.

This paper proposes the study of inference problems that arise from bodies crossing beams among obstacles. It turns out that the subject is much more general than the particular scenario just described. In addition to binary detection beams or regions placed around an environment, the mathematical model arises in other contexts. For example, if a robot carries a camera and certain image features critically change, then the event may be equivalent to crossing a "virtual" beam in the environment (see Section 3).

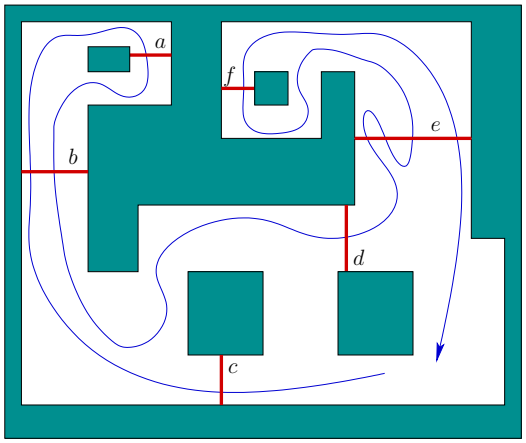

Fig. 1. What can be determined about the path using only the word $cbabdeeefe$, which indicates the sequence of sensor beams crossed?

Our questions are inspired by many problems that society currently faces. There is widespread interest in developing *assisted living* systems that use sensors to monitor the movements of people in their homes or hospitals. How much can be accomplished with simple detection beams, which are affordable, robust, and respect privacy? Alternatively, imagine the field of *sensor-based forensics*, in which police investigators or lawyers would like to corroborate or refute a testimony about how people moved at a crime scene. A simple verification test might based on the sequence of beam crossings might establish that someone is lying. Other problems include tracking wildlife movement for conservation purposes, landmark-based navigation with outdoor vehicles, sensor-assisted safe child care, and security.

Suppose there is one moving body, called an *agent*, and we have the information that a sequence of beams was crossed. We focus on three kinds of questions: 1) Supposing regions are delineated by the arrangement of beams, what possible sequences of regions did the agent visit? 2) What path did the agent take up to homotopy? 3) How many times did the agent wind around each obstacle? These questions form the basis of Sections 4 to 6. Multiple agents are briefly considered in Section 7. The last two questions are familiar problems in topology and group theory, and are motivated by homotopy and homology, respectively. In particular, the topic is close to *word problems* in group theory, in which it must be determined whether two words (e.g., $abac^{-1}b$ and cab^{-1}) are the same group element. In the general group-theoretic setting, such questions go back to 1910 with Dehn's fundamental problems [4]; decidability and complexity results can be reviewed in [9].

The most closely related works are algorithms to decide whether two paths in a punctured plane are homotopic [1, 8]. These algorithms are based on extending vertical lines from each of the punctures. The vertical lines serve two purposes: First, any given path is represented with the sequence of vertical rays it intersects. Second, they connect the different fibers of a covering space of the punctured plane. In this context, two paths are homotopic if and only if they have the same endpoints when they are lifted to the universal covering space. The novelty in our work is that we start with *sensor* words and must first convert them into path descriptions. This represents an *inverse problem* that is constrained by the geometry and topology of the sensors and obstacles.

Following a minimal sensing perspective in the context of sensor networks, works such as [18, 25, 26] detect and count targets using binary proximity sensors. The binary proximity sensors can be considered as *overlapping beams* that go off when an agent is in range. Tracking targets is done with a particle filter, in which each particle is a candidate trajectory of a target. In robotics, the use of particle filters has been very successful in solving tasks such as simultaneous localization and mapping (SLAM) [2, 3, 5, 22, 24, 28]. Traditionally, the focus of SLAM approaches is the production of an environment representation based on metric information. From a sensing perspective, algorithms such as the ones used in SLAM, are concerned with the problem of sensor fusion, in which several sensors are added to increase the accuracy of a solution. In contrast, others have studied the *minimal sensing requirements* to solve a particular task [10, 12, 23, 30]. This typically involves a characterization and simplification of the information space associated with the task [20], which considers the whole histories of commands given to the actuators and sensing observations. From an information space perspective, in [31] the location of moving agents is inferred from combinatorial changes in sensing observations. Such combinatorial changes may correspond to *visual events* [7], which can be abstracted in our work as sensor beams. An example of this is presented in this paper in Section 3, in which the combinatorial changes correspond to crossing of landmarks [29]. The careful consideration of visual events is the basis for solutions to problems such as localization [6, 14] and visibility-based pursuit-evasion [11, 13, 17, 21, 27].

2 Problem Formulation

Let $W \subseteq \mathbb{R}^2$ be the closure of a contractible open set. A common case is $W = \mathbb{R}^2$. Let $\mathcal{O}$ be a set of n pairwise-disjoint *obstacles*, which are each the closure of a contractible open set. Let X be the *free space*, which is the open subset of W that has all $o \in \mathcal{O}$ removed. Let $\mathcal{B}$ be a set of m *beams*, each of which is an open linear subset of X. If W is bounded, then every beam is a line segment with both endpoints on the boundary of X. (Note that beams may connect an obstacle boundary to itself, another obstacle's boundary, or the boundary of W; also, a beam may connect the boundary of W to itself.) If W is unbounded, then some beams may be open rays that emanate from the boundary of an obstacle or even lines that are contained in the interior of W.

Regions

The collection of obstacles and beams induces a decomposition of X into connected *cells*. If the beams in $\mathcal{B}$ are pairwise disjoint, then each $B \in \mathcal{B}$ is a 1-cell and the 2-cells are maximal regions bounded by 1-cells and portions of the boundary of X. If beams intersect, then the 1-cells are maximal segments between any beam intersection points or boundary elements of X; the 2-cells follow accordingly. Every 2-cell will be called a *region*.

Agent path

Suppose that an *agent* moves along a *state trajectory* (or path) $\tilde{x} : [0, 1] \rightarrow X$, in which $[0, 1]$ is imagined as a time interval. (Alternatively, $[0, t_f]$ could be allowed for any $t_f > 0$; however, this flexibility is unnecessary because speed and time scaling are irrelevant to our questions.)

Sensor model

If the agent crosses a beam, what exactly is observed? Assume that the set of possible $\tilde{x}$ is restricted so that: 1) every beam crossing is transverse (the agent cannot "touch" a beam without crossing it and the agent cannot move along a beam) and 2) the agent never crosses an intersection point between two or more beams (if any such intersections exist).

Let L be a finite set of *labels*. Suppose each beam is assigned a unique label by some bijection $\alpha : \mathcal{B} \rightarrow L$. The sensor model depicted in Figure 1 can be obtained by a sensor mapping $h : X \rightarrow Y$, in which $Y = L \cup \{\varepsilon\}$ is the *observation set*. If $\tilde{x}(t) \in B$ for some $B \in \mathcal{B}$, then $h(\tilde{x}(t)) = \alpha(B)$; otherwise, $h(\tilde{x}(t)) = \varepsilon$, which is a special symbol to denote "no beam". This is referred to as the *undirected beam* model because it indicates that the beam was crossed, but we do not know the direction.

To obtain a *directed beam* model, let $D = \{-1, 1\}$ be a set of *directions*. In this case, the observation set is $Y = (L \times D) \cup \{\varepsilon\}$, and the sensor mapping yields the orientation of each beam crossing (note that in addition to $\tilde{x}(t)$, the sensor mapping must now know what side of the beam the agent was on at time t^-).

So far, the beams have been *fully distinguishable* because α is a bijection. It is possible to make $|L| < m$ (the number of beams) and obtain some *indistinguishable* beams, in which case $\alpha : \mathcal{B} \rightarrow L$ is not bijective.

If a collection of beams is disjoint, distinguishable, and directed, the case will be referred to as *ddd-beams*, which is the most ideal situation.

Sensor words

What observations are accumulated after $\tilde{x}$ is traversed? We assume that all ε observations are ignored, resulting in a sequence $\tilde{y}$, called the *sensor word*, of the remaining observations ($\tilde{y}$ is a kind of *observation history* [20]). For the example in Figure 1, suppose $L = \{a, b, c, d, e, f\}$. In the case of a undirected beams, the sensor word is $cbabdeeefe$. If the beams were directed so that left-to-right and bottom-to-top are the "forward" direction, then the sensor word could be encoded as $c^{-1}ba^{-1}b^{-1}dee^{-1}efe^{-1}$. For each $l \in L$, l denotes the forward direction and l^{-1} denotes the backward direction.

Inference

Let $\tilde{Y}$ be the set of all possible sensor words and let $\tilde{X}$ be the set of all possible state trajectories. Let $\phi : \tilde{X} \rightarrow \tilde{Y}$ denote the mapping that produces the sensor word $\tilde{y} = \phi(\tilde{x})$.

Suppose that $\tilde{y}$ has been obtained with no additional information. What can be inferred about $\tilde{x}$? Let $\phi^{-1}(\tilde{y})$ denote the *preimage* of $\tilde{y}$:

$$\phi^{-1}(\tilde{y}) = \{\tilde{x} \in \tilde{X} \mid \tilde{y} = \phi(\tilde{x})\}. \tag{1}$$

The inference problem amounts to: Under what conditions can we compute a useful description of the preimages?

The following are three ways to partially characterize these preimages, forming the basis of Sections 4, 5, and 6, respectively:

1. Using $\tilde{y}$, compute the set of possible sequences of regions visited by $\tilde{x}$. This characterizes $\phi^{-1}(\tilde{y})$ in a stage-by-stage manner.
2. If there are n obstacles, the *fundamental group* $\pi_1(X)$ [15] is the free group F_n on n letters. Using some combinations of initial conditions, and assuming fixed starting and stopping paths, compute a representation of the possible paths as an element of F_n. This clearly throws away some information by considering only the homotopy equivalence class of paths. Hence, it is an over-approximation of $\phi^{-1}(\tilde{y})$.

3. Compute the signed number of windings around each obstacle, in which the order that windings are made is dropped. This throws away even more information than the homotopy equivalence class; therefore, it yields an even larger over-approximation of $\phi^{-1}(\tilde{y})$.

Using these representations, we can answer questions such as: Do two sensor words correspond to homotopic paths? Do they correspond to homologous paths? Is it possible for them to visit the same regions in the same order?

Combinatorial filters

Whenever possible, we try to design a *filter*, which computes statistics incrementally as new data are obtained. The most common example is the Kalman filter, which computes the next mean and covariance based on the new sensor reading and the previous mean and covariance [16, 19]. In this paper, we design *combinatorial filters*, which are minimalist non-probabilistic analogs to Bayesian filters. For a sensor word $\tilde{y}_k$ of length k, let $\kappa(\tilde{y}_k)$ denote a statistic, which could, for example, be the set of possible current regions. A combinatorial filter efficiently computes $\kappa(\tilde{y}_{k+1})$ using only $\kappa(\tilde{y}_k)$ and y_{k+1}, in which y_{k+1} is the last (most recent) letter in $\tilde{y}_{k+1}$. This implies that $\tilde{y}_k$ does not need to be stored in memory; only $\kappa(\tilde{y}_k)$ is needed.

3 Concrete Scenarios

This section motivates the general formulation of Section 2 to illustrate the wide range of settings to which it applies.

Physical sensor beams

The most common set of examples corresponds to actually engineering a physical environment with the placement of cheap beam sensors among with nonconvex obstacles. Recall Figure 1. In this case, virtually any model from Section 2 can be realized in practice. In the remainder of this section, we obtain beams *virtually* via other sensing modalities.

Crossings of landmarks

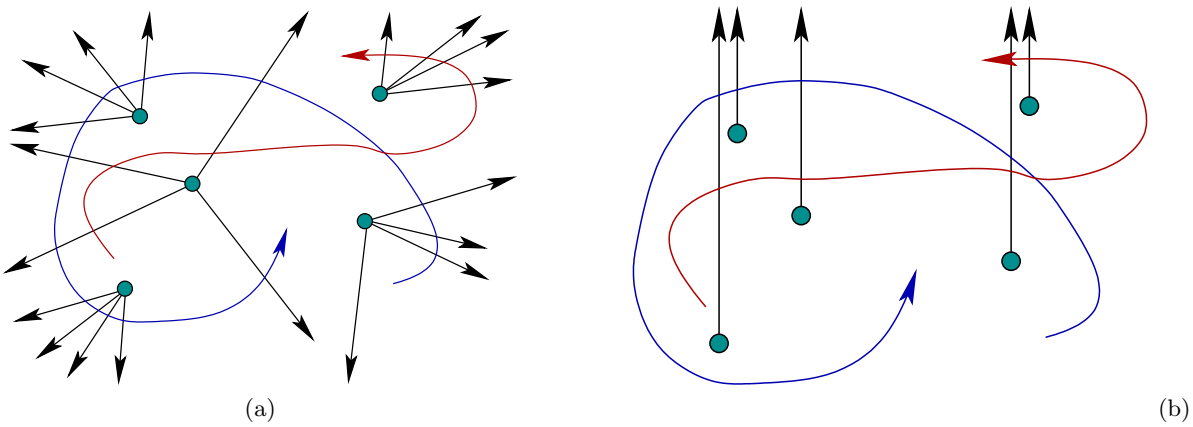

Fig. 2. a) Virtual beams based on pairwise crossings in an image, b) virtual beams based on passing directly north of an obstacle.

Imagine that a robot moves in a large field, in which several landmarks (e.g., radio towers) are visible using an omni-directional camera. This can be modeled by $W = \mathbb{R}^2$ and $\mathcal{O}$ as a set of point obstacles. Suppose that the landmarks are fully distinguishable and some simple vision software indicates when a pair of landmarks are "on top of each other" in the image. In other words, the robot and two landmarks are collinear, with one of the two landmarks in the middle. The result is mathematically equivalent to placing $n(n-1)$ beams as shown in Figure 2.a, in which rays extend outward along lines passing through each pair of landmarks.

Several interesting variations are possible based on precisely what is detected in the image. If the only information is that o_i and o_j crossed each other in the image, then all beams are undirected and the two beams associated with o_i and o_j are indistinguishable. If we know whether o_i passes in front of or behind o_j, then the beams become fully distinguishable. If we know whether o_i passes to the left or right of o_j in the image, then the beams even become directed.

Passing over a windshield mark

Keeping the landmark-based example, consider changing the sensing so that instead of detecting pairwise landmark crossings, the robot simply knows when some landmark is directly south. This could be achieved by using a compass to align the vehicle and noting when a landmark crosses a fixed spot on the image plane or windshield. Figure 2.b shows virtual beams that are obtained in this way. Directed and undirected beam models are possible, based on whether the sensor indicates the left-right direction that the landmark moves as it crosses the fixed spot. An important property of this model is that the beams do not intersect (assuming the points are not collinear). Section 5 utilizes this property to reconstruct the path up to homotopy equivalence.

4 Region Filters

In this section, we present a simple method to keep track of the possible regions in which the agent might be after obtaining the sensor word. Suppose that n obstacles $\mathcal{O}$ and m beams $\mathcal{B}$ are given, which leads to the cell decomposition of X, discussed in Section 2. The beams may intersect and may or may not be directed.

Let R_0 denote the set of possible regions that initially contain the agent, before any sensor data is observed. Let R_k denote the set of possible regions after a sensor word $\tilde{y}_k$, of length k, has been obtained. The task is to design a filter that computes R_{k+1}, given R_k and y_{k+1}, which is the most recent observation in $\tilde{y}_{k+1}$.

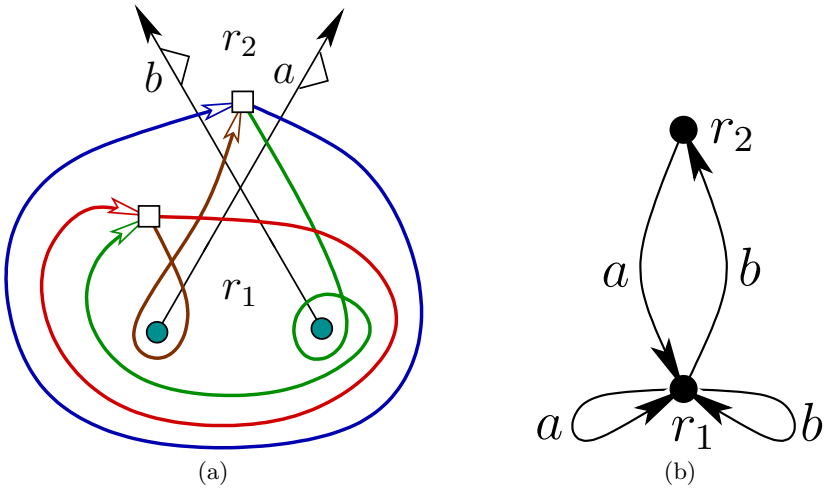

Fig. 3. a) Two intersecting directed beams and four region sequences (and possible paths) that can be inferred from the sensor word ab; b) the corresponding graph G.

Let G denote a directed (multi)graph that possibly contains self-loops. Each vertex of G is a region, and a directed edge is made from region r_1 to region r_2 if either: 1) they share an interval of an undirected beam along their boundary, or 2) they share an interval of a directed beam that is directed from r_1 to r_2. The edge is labeled with the beam label. A self-loop in G is made if it is possible to cross a beam and remain in the same region, which is illustrated in Figure 3.a; a or b may be crossed while remaining in r_1 the whole time. Also, note that if the initial region is unknown and $\tilde{y} = ab$ is the sensor word, there are four possible interpretations in terms of the possible regions traversed. The corresponding graph G is shown in Figure 3.b.

Simulation as a Nondeterministic Machine

The region filter is implemented on G in a way similar to the simulated operation of a nondeterministic finite automaton. The method keeps track possible states by *marking* the corresponding vertices of G. Initially, mark every vertex in R_0. The filter proceeds inductively. At stage k, the marked vertices are precisely those corresponding R_k. Suppose that y_{k+1} is observed, which extends the sensor word by one observation. For each marked vertex, look for any outgoing edge labeled with y_{k+1}. In each case, the destination vertex is marked. If $y_{k+1} = l^{-1}$ for some $l \in L$ and an ingoing edge is labeled with l, then the edge's source vertex is marked. Any vertex that was marked at stage k but did not get marked in stage $k+1$ becomes *cleared*. Note that the total number of marked vertices may increase because from a single vertex there may be multiple edges that match y_{k+1}. Also, this approach works for the case of partially distinguishable beams because the match is based on the observation y_{k+1}, rather than the particular beam. The set of marked vertices yields R_{k+1}.

Suppose that after computing R_k, we would like to know the possible *sequence* of regions traversed by the agent. The graph G can be used for this computation as well by taking each region in R_k and working backwards using the sensor word to construct possible regions at earlier stages. Note that once R_k is given, the set of possible regions R_i, which was computed at some earlier stage $i < k$, might contain regions that are known at stage k to have been impossible. In other words, information gained at later stages can refine our belief about what might have occurred several stages earlier.

Algorithm complexity

The region filter based on marking vertices in G runs in time $O(|V| + |E|)$ for each update from k to $k+1$, in which $|V|$ and $|E|$ are the numbers of vertices and edges in G, respectively. Note that the number of computations grows with the number of marked vertices, which reflects the amount of uncertainty about the current region. In some cases the number of marked vertices cannot increase, as in the case of ddd-beams. Using exponential space (undesirable), each update can be performed in constant time, similar to the classical NFA to DFA transformation.

Computed example

To illustrate the region filters, we present simulation for the concrete scenario of beams coming from crossings of landmarks. We computed two cases. In the first one, all beams are distinguishable and directed (see Figure 4). In the second, the only information available at a crossing is the corresponding pair of landmarks. Therefore, the beams are not directed, and each beam label can be reported by exactly two beams (see Figure 5). In these examples, a beam is identified with the pair of landmarks that produce it.

5 Reconstruction Up to Homotopy

In this section, the task is to use the sensor word $\tilde{y}$ to reconstruct a description of possible paths $\tilde{x} \in \tilde{X}$ up an equivalence class of homotopic paths. Assume that all paths start and stop at some fixed *basepoint* $x_0 \in X$ which lies in the interior of some region; this assumption is lifted at the end of the section. Paths can be described up to homotopy using the fundamental group $\pi(X)$, which is known to be F_n for a planar region with n holes (caused by the obstacles). Each element of F_n is an equivalence class of homotopic loop paths with endpoints fixed at x_0. The set F_n is called the *free group on n letters*, and its elements can be considered as the set of all finite strings that can be formed using any $l \in L$ and their inverse forms l^{-1}. Since F_n is a group, it also contains an identity element ε. Furthermore, the relations $\varepsilon l = l\varepsilon = l$ and $l\ l^{-1} = \varepsilon$ must be applied to shorten strings by applying cancellations and deleting identity elements. The result is referred to in group theory as a *reduced word*.

An important issue with F_n is its choice of basis. Recall from linear algebra that there are many ways to define and transform bases for a vector space. A similar but more complicated situation exists for F_n. For any fixed ordered basis (called letters above), $a_1, \ldots, a_n$, every other possible ordered basis is given by $g(a_1), \ldots, g(a_n)$. The elements g run over *every* element in the automorphism group of F_n, which is denoted by $Aut(F_n)$. In other words, there is a natural one-to-one correspondence between every ordered basis of F_n and $Aut(F_n)$. This structure is currently under active investigation in pure mathematics. In this paper, a basis of F_n will be chosen in the most straightforward way and other bases will be directly transformed to it. The full structure of $Aut(F_n)$ will thus be avoided.

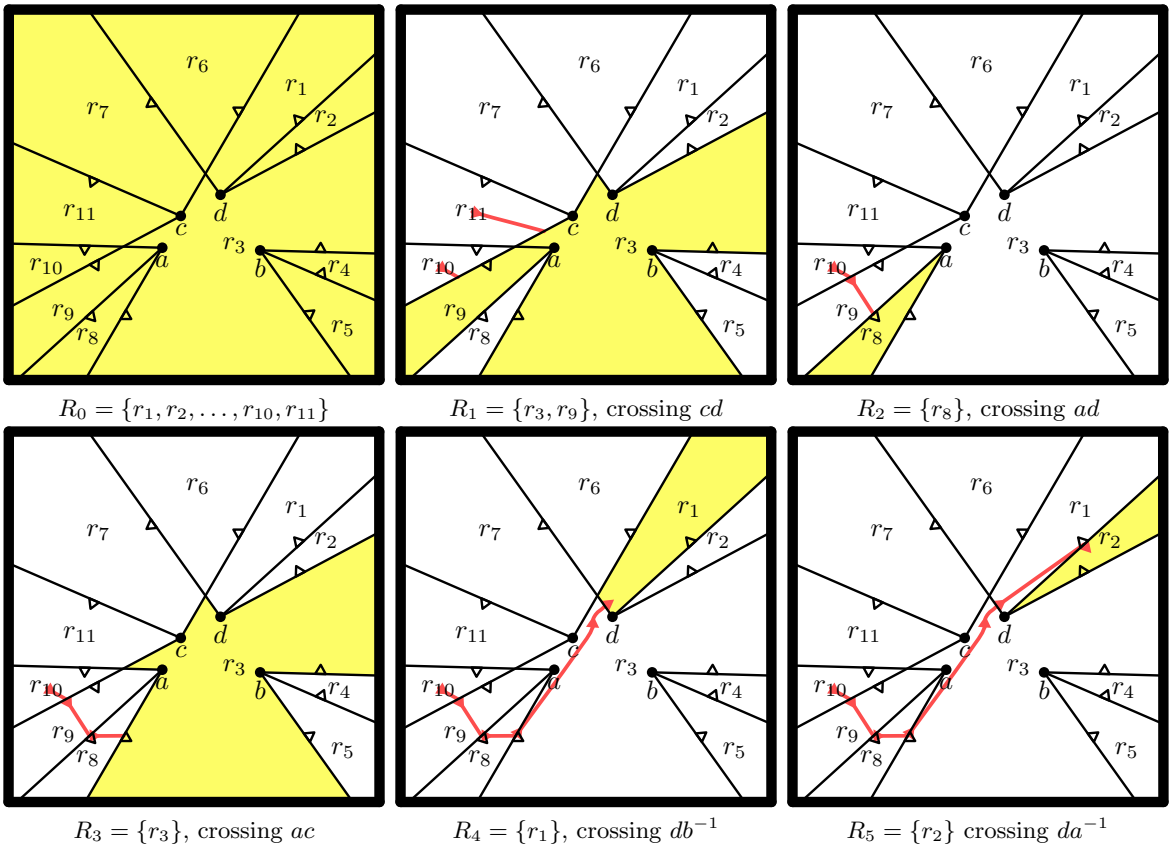

Fig. 4. Region filter simulation. Crossings of landmarks for directed and distinguishable beams.

5.1 Perfect beams

Let a beam be called *outer* if it is either an infinite ray (possible only if X is unbounded) or it is a finite segment that connects an obstacle to the boundary of W. For a set of n obstacles, let a *perfect* collection of beams mean that there are exactly $m = n$ ddd-beams (recall that this means *disjoint, distinguishable, and directed beams*), with exactly one outer beam attached to each obstacle. For convenience, further assume that all beams in a perfect collection are oriented so that a counterclockwise traversal corresponds to the "forward" direction, as shown in Figure 6.

Proposition 1. *For a perfect collection of beams, the sensor word $\tilde{y}$ maps directly to an element of F_n by simply renaming each letter.*

Proof: While preserving homotopy equivalences, the obstacles and basepoint can be moved into a canonical form as shown in Figure 7. This corresponds to choosing a particular basis of F_n in which each generator a_i is exactly a counterclockwise loop around one obstacle. Each b_i corresponds to a beam and letter in L. Using the given sensor word $\tilde{y}$, a word $f(\tilde{y}) \in F_n$ is formed by mapping each a_i in $\tilde{y}$ to b_i in $f(\tilde{y})$. Each beam crossing then corresponds directly to a generator of F_n under the chosen basis. This converts every $\tilde{y}$ into an element of F_n that represents the loop path that was traversed using basepoint x_0. □

Note that reductions of the form $\varepsilon l = l\varepsilon = l$ and $l\ l^{-1} = \varepsilon$ may be performed in $\tilde{y}$ or $f(\tilde{y})$, either before or after the mapping f is applied.

5.2 Sufficient ddd-beams

What if the collection of beams is not perfect? For some arrangements of obstacles, it might not even be possible to design a perfect collection (unless beams are allowed to be nonlinear). Suppose that a collection $\mathcal{B}$ of $m \geq n$ ddd-beams is given. It is called *sufficient* if all of the resulting regions are simply connected; see Figure 8.a. Note that any sufficient collection must contain at least one outer beam. Also, any perfect collection is also sufficient.

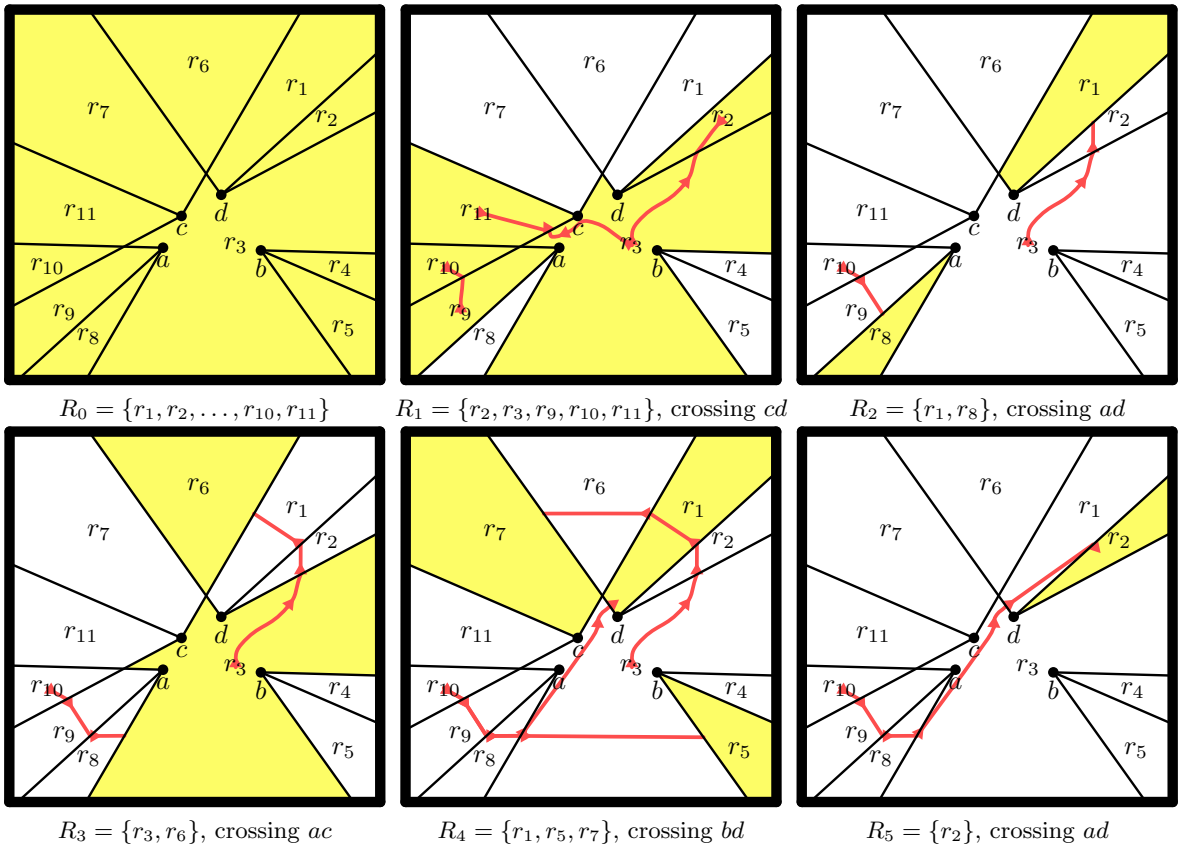

Fig. 5. Region filter simulation. Crossings of landmarks for undirected beams and partially distinguishable beams. Each beam label can be reported by exactly two beams.

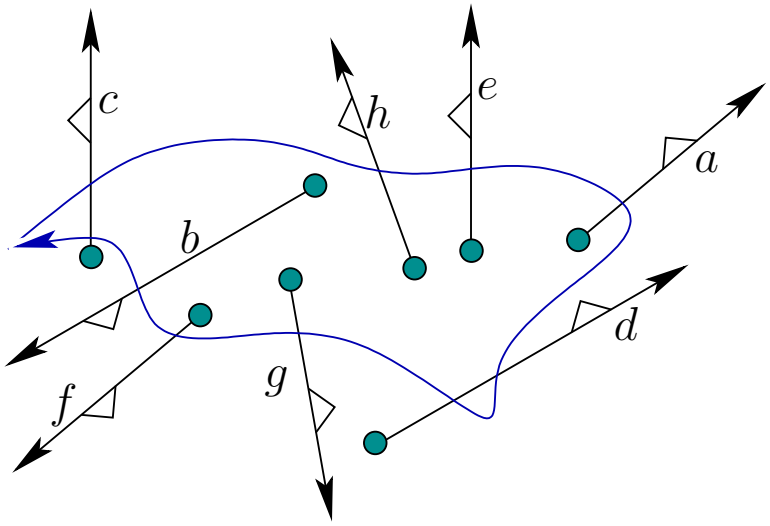

Fig. 6. A perfect collection of beams.

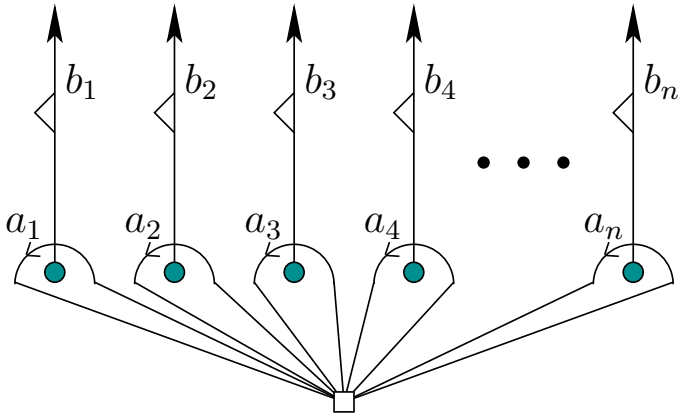

Fig. 7. A construction that converts the sensor word $\tilde{y}$ into an element of the fundamental group F_n.

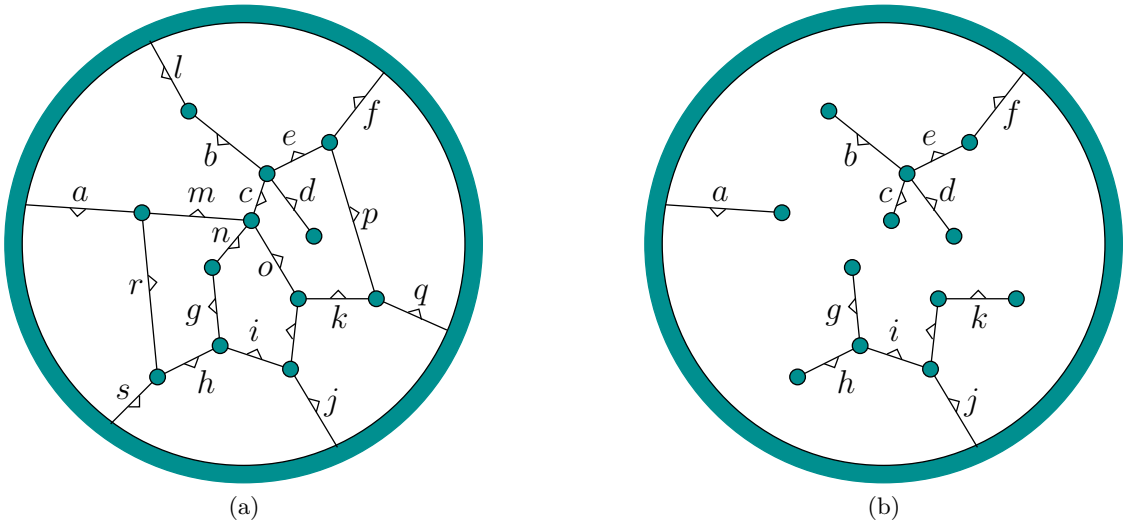

Fig. 8. a) A sufficient collection of ddd-beams; b) a minimally sufficient collection of ddd-beams forms trees that each contain one outer beam. This example is obtained by removing beams from the the figure on the left.

Suppose that a sensor word $\tilde{y}$ is obtained for a sufficient collection $\mathcal{B}$ of beams. The first step in describing the path as an element of F_n is to disregard redundant beams. To achieve this, let $\mathcal{B}' \subseteq \mathcal{B}$ be a minimal subset of $\mathcal{B}$ that is still sufficient. Such a collection is called *minimally sufficient* and can be computed using spanning tree algorithms such as depth-first or breadth-first search. An example is shown in Figure 8.b.

Proposition 2. *For a minimally sufficient collection $\mathcal{B}$ of ddd-beams, the sensor word $\tilde{y}$ maps directly to an element of F_n by simply renaming each letter; however, a different basis is obtained for F_n in comparison to Proposition 1.*

Proof: A basis for the fundamental group is defined as follows. For each tree of beams in $\mathcal{B}$, a collection of loop paths can be formed as shown in Figure 9. Each loop must cross transversely the interior of exactly one beam and enclose a unique nonempty set of obstacles. Such loops always exist and can be constructed inductively by first enclosing the leaves of the tree and then progressing through parents until a loop is obtained that traverses the outer beam. Since there is only one region, it is possible to inductively construct such a collection of loops for every tree of beams in $\mathcal{B}$. The total collection of loops forms a basis of F_n, which can be related to the basis in Proposition 1 via classical Tietze transformations. The mapping f from $\tilde{y}$ to $f(\tilde{y}) \in F_n$ is once again obtained by mapping each letter in $\tilde{y}$ to its corresponding unique loop that traverses the beam. □

Using Proposition 2, a simple algorithm is obtained. Suppose that any sufficient collection $\mathcal{B}$ of ddd-beams is given and a sensor word $\tilde{y}$ is obtained. A spanning tree $\mathcal{B}' \subseteq \mathcal{B}$ of beams is computed, which is minimally sufficient. Let $L' \subseteq L$ denote the corresponding set of beam labels. To compute the element of F_n, the first step is to delete from $\tilde{y}$ any letters in $L \setminus L'$. This yields a reduced word $\tilde{y}'$ for which each letter

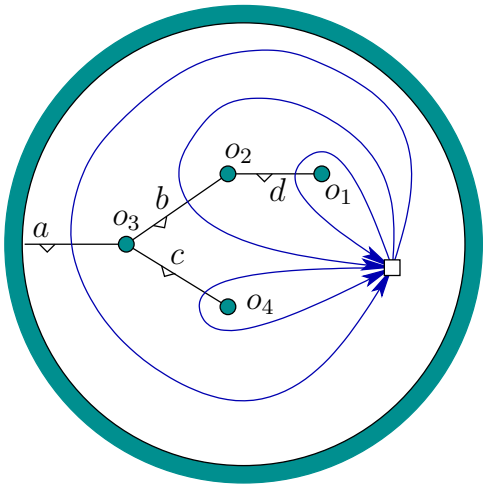

Fig. 9. Forming a basis using a sufficient tree of beams.

can be mapped directly to a loop using Proposition 2 to obtain a representation of the corresponding path in F_n. Once again, reductions based on the identity and inverses in F_n can be performed before or after the mapping is applied.

5.3 The general case

Consider a collection $\mathcal{B}$ of beams in which some may intersect, some may be undirected, and some may even be indistinguishable. The collection is nevertheless assumed to be *sufficient*, which means that all of the corresponding regions are simply connected. Rather than worry about making a minimal subset of $\mathcal{B}$, the method for the general case works by inventing a collection of *imaginary* beams that happens to be minimally sufficient. Since the ambiguity may be high enough to yield a set of possible paths, the region filter from Section 4 is used.

Before any sensor words are processed, the following preprocessing steps are performed based on W, $\mathcal{O}$, $\mathcal{B}$, L, and α:

1. Compute the arrangement of regions and multigraph G from Section 4.
2. For each vertex in G choose a *sample point* in its corresponding region.
3. For each directed edge e in G, compute a piecewise-linear *sample path* that: i) starts at the sample point of the source vertex of e, ii) ends at the sample point of the destination vertex of e, and iii) crosses the beam associated with e in a manner consistent with its label. The sample path cannot intersect any other beams.
4. Construct any minimally sufficient collection $\mathcal{B}_I$ of *imaginary* ddd-beams. A convenient choice is to make all imaginary beams vertical.
5. For all computed sample paths from Step 3, compute their intersections with the imaginary beams of Step 4 and record the order in which they occur.

Now suppose that a sensor word $\tilde{y}$ is given. The region filter of Section 4 is used to determine the set of possible region sequences. For each region sequence, a path $\tilde{x}'$ is obtained from the corresponding sample points and paths in G. An imaginary sensor word $\tilde{y}'$ is obtained by the sequence of beams in $\mathcal{B}_I$ that are crossed by $\tilde{x}'$. Relying on Proposition 2, $\tilde{y}'$ can be mapped directly to an element of F_n. As usual, reductions can be applied to $\tilde{y}'$ or its image in F_n. Once elements of F_n are computed and reduced for each possible region sequence, duplicates are removed to obtain the complete set of possible homotopically distinct paths based on the sensor word $\tilde{y}$. Note that $\mathcal{B}_I$ essentially allows the user to define whatever basis of F_n is desired to express the result.

Now remove the assumption that only loop paths are executed using a basepoint x_0. If all paths start at some x_0 and terminate at some x_1, then a fixed path segment that connects x_1 back to x_0 can be chosen. This path may intersect some beams, which is false information; however, actual possible paths executed by the agent can at least be compared up to homotopy. If either of the endpoints is not fixed, then all paths become trivially homotopic by continuously shrinking each path to the other basepoint. One possibility is to assume that there are several possible fixed points based on the starting and final regions produced in each

sequence from the region filter. In this way, possible paths can at least be compared if their starting and terminating regions match.

6 Path Winding Numbers

Rather than characterizing paths up to homotopy, this section considers the number of times the path "winds" around each obstacle. Suppose that the agent travels along a loop path. There is an integer *winding number* $v_i \in \mathbb{Z}$ for each $o_i \in \mathcal{O}$, in which m_i is defined as the number of times the path wraps counterclockwise around o_i after deleting all other obstacles and pulling the path tight around o_i using homotopy. If there are n obstacles, then a vector of n winding numbers is obtained. Two paths are called *homologous* if and only if their vector of winding numbers are identical.

6.1 Perfect beams

In the case of perfect beams, the winding numbers are obtained by directly "abelianizing" the sensor word $\tilde{y}$. Due to Proposition 1, the sensor word maps directly to the free group element $f(\tilde{y}) \in F_n$. The winding numbers are then obtained by applying the commutativity relation to $f(\tilde{y})$ (or conveniently, directly to $\tilde{y}$) and sorting the terms. For example, $\tilde{y} = aba^{-1}bbab^{-1}b^{-1}ab^{-1}b^{-1}b^{-1}$ is abelianized to:

$$\begin{aligned} \tilde{y} =& aba^{-1}bab^{-1}b^{-1}ab^{-1}b^{-1}b^{-1} \\ =& aaaa^{-1}bbb^{-1}b^{-1}b^{-1}b^{-1}b^{-1} \\ =& a^2b^{-3}. \end{aligned} \tag{2}$$

The first step sorts the terms, and the second step performs cancellations. This result is expressed as a monomial in which l^k is a sequence of k l's and l^{-k} is a sequence of k l^{-1}'s for each $l \in L$. The winding numbers are simply the exponents; for (2) we obtain $w = (2, -3)$. Note that the winding numbers can computed in time $O(|\tilde{y}|)$ without actually sorting by simply maintaining n counters, one for each letter in $l \in L$. Scan across $\tilde{y}$ and increment or decrement each counter, based on whether l or l^{-1}, is encountered, respectively. Note that this makes a constant-time combinatorial filter, as defined in Section 2, by computing the winding numbers w_{k+1} at step $k+1$ from the winding number vector w_k and the last observation y_{k+1}, which is the last letter of $\tilde{y}_{k+1}$.

6.2 Sufficient ddd-beams

Once a minimally sufficient collection is determined, the computation for this case then proceeds in the same way as for perfect beams. This yields winding information, but it needs to be transformed to obtain the correct result. Recall the basis from Figure 9 and suppose that for a path, the abelianized word obtained is $a^5b^{-3}c^4d$. Each exponent may contain information about multiple winding numbers. For example, a^5 implies that the agent wrapped 5 times around o_3, but it also wrapped 5 times around o_1, o_2, and o_4. Likewise, b^{-3} wraps -3 times around o_1 and o_2. A counter is made for each obstacle and each computed exponent raises or lowers some counters. After being performed for each exponent, the result is obtained. For the example $a^5b^{-3}c^4d$ based on Figure 9, the winding numbers are $(3, 2, 5, 9)$. Note that if the positive direction of a beam is in the clockwise direction, then the computed winding number needs to be multipled by -1.

6.3 The general case

Now suppose that a sufficient collection of general beams has been given, which is the model used in Section 5.3. A straightforward approach is to first run the algorithm of Section 5.3. After the sufficient collection of imaginary beams has been placed and the free group elements have been computed, they can be abelianized to obtain the exponents in the method just described. This yields a set of vectors of winding numbers.

This approach, however, computes more information than is needed to simply obtain the winding numbers. Suppose that a sufficient collection of beams is given that is not necessarily disjoint, but all beams are directed and distinguishable. Since the winding number essentially ignores all other beams, an approach can be developed by picking a minimally sufficient collection of beams that is not necessarily disjoint. The beam intersections do not interfere with the calculation of winding numbers. For a given sensor word, any letters

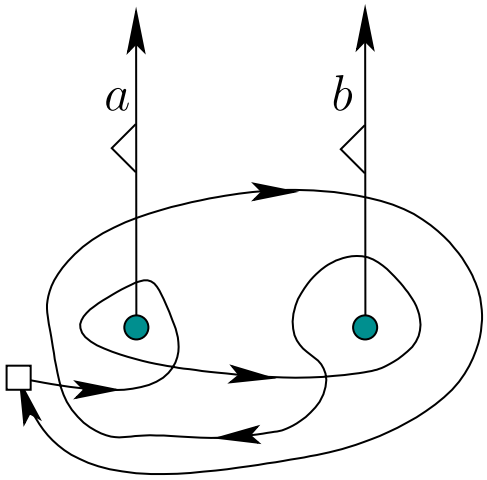

Fig. 10. A simple commutator example that yields sensor word $aba^{-1}b^{-1}$ and winding numbers $(0, 0)$, but corresponds to a non-trivial path.

that do not appear in the minimally sufficient collection can simply be deleted. The method then proceeds as in the case of ddd-beams.

In the most general setting of a sufficient collection of beams, the region filter of Section 4 can be applied to yield possible region sequences. For each computed region sequence, the particular beam and its direction crossed can be inferred. Based on this information, the method described for distinguishable, directed beams can be applied.

6.4 Higher-order winding numbers

Winding numbers give some crude data concerning free groups and an agent path. These give a measure of how many times an agent circles a given obstacle without counting how the agent weaves in between obstacles. However, winding numbers are insensitive to paths such as a commutator around obstacles, as shown in Figure 10.

There are "higher order winding numbers" which keep track of how a agent does in fact interweave through different obstacles. These "higher order winding numbers" in the case above arise in two classical ways reflecting the interplay between geometry and a free group. These are the Lie algebras which arise from either (i) the descending central series of a free group, or (ii) principal congruence subgroups of level p^r in the group of $SL(2, \mathbb{Z})$. These Lie algebras provide measures of complexity in addition to "higher order winding numbers" and it remains an open problem to develop computation methods that characterize them for a given sensor word.

7 Multiple Agents

The formulation in Section 2 can be naturally extended by allowing more than one agent to move in W. In this case, suppose that the sensor beams cannot distinguish between agents. They simply indicate the beam label whenever crossed. For simplicity, assume that agents never cross beams simultaneously. The task is to reconstruct as much information as possible about what path they might have taken. We could proceed as in Sections 4 - 6 and determine region sequences, path homotopy class, and winding numbers. The high level of ambiguity, however, may require further simplifications.

Figure 11.a shows a simple example of this, in which there is one obstacle, two agents, and three undirected beams. Question: If the agents start together in a room, are they together some room after some sensor word was observed? Consider designing the simplest algorithm that answers this question. Figure 11.b shows a surprisingly simple four-state automaton that answers the question for any sensor word. The T state means they are together in some room. Each D_x state means they are in different rooms, with beam x separating them. With only two bits of memory, arbitrarily long sensor words can be digested to produce the answer to the question.

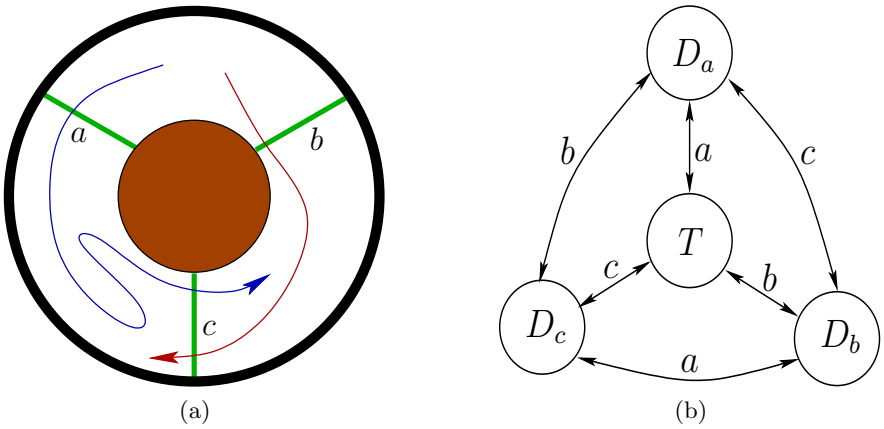

Fig. 11. a) A three-region problem with two agents; b) a tiny automaton (combinatorial filter) that determines whether the agents are together in a room.

Many open questions remain, especially for substantially more complicated environments, such as the one in Figure 2.a with several agents. For which questions can small automata be designed? For a given question, what is the complexity in terms of numbers of agents, obstacles, and beams? What other ways exist for reconstructing and describing and possible paths taken by multiple agents?

8 Conclusions and Open Questions

In this paper we identified a basic inference problem based on agents moving among obstacles and detection beams. Recall from Section 3 that the beams may directly model physical sensors or they may arise virtually from a variety of other sensing models. Therefore, the region filter, homotopic reconstruction, and winding-number computations provide basic information that arises in numerous settings such as robotics, security, forensics, environmental monitoring, and assisted living.

The results presented here represent a first step in understanding this broad class of problems. Many open issues remain for future research, several of which are suggested here: 1) It is assumed that the geometric arrangement of obstacles and beams is known. What happens when this is uncertain? For example, we might not even know which beams intersect. The sensor words can be used to make simultaneous inferences about the agent path and the beam arrangement. 2) Without the assumption of transverse beam crossings and crossings are intersection points, significantly more ambiguity arises. How do these affect the computations? 3) What are the limits of path reconstruction when there are two or more agents? How efficient can filters be made for such problems when there are many obstacles and beams? 4) What other specific path statistics can be computed efficiently from beam data? Can Lie algebra constructions be applied to efficiently compute higher-order winding numbers (based on commutators) for the paths? Can the sensor data be used to compare paths as elements of the braid group? 5) Since the methods so far provide only inference, how can their output be used to design motion plans? In other words, how can the output be used as a filter that provides feedback for controlling how the agents move to achieve some task?

Acknowledgments

The authors are supported in part by the DARPA SToMP program (DSO HR0011-07-1-0002).

References

1. S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink. Testing homotopy for paths in the plane. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 160–169, New York, NY, USA, 2002. ACM.
2. J. Castellanos, J. Montiel, J. Neira, and J. Tardós. The SPmap: A probabilistic framework for simultaneous localization and mapping. *IEEE Transactions on Robotics & Automation*, 15(5):948–953, 1999.
3. H. Choset and K. Nagatani. Topological simultaneous localization and mapping (T-SLAM). *IEEE Transactions on Robotics & Automation*, 17(2):125–137, 2001.

4. M. Dehn. *Papers on Group Theory and Topology*. Springer-Verlag, Berlin, 1987.
5. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions on Robotics & Automation*, 17(3):229–241, 2001.
6. G. Dudek, K. Romanik, and S. Whitesides. Global localization: Localizing a robot with minimal travel. *SIAM Journal on Computing*, 27(2):583–604, April 1998.
7. F. Durand. *3D Visibility: Analytical study and applications*. PhD thesis, Université Grenoble I – Joseph Fourier Sciences et Géographe, July 1999.
8. A. Efrat, S. Kobourov, and A. Lubiw. Computing homotopic shortest paths efficiently. *Computational Geometry*, 35(3):162–172, October 2006.
9. D. B. A. Epstein, M. S. Paterson, G. W. Camon, D. F. Holt, S. V. Levy, and W. P. Thurston. *Word Processing in Groups*. A. K. Peters, Natick, MA, 1992.
10. M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics & Automation*, 4(4):369–379, August 1988.
11. B. Gerkey, S. Thrun, and G. Gordon. Clear the building: Pursuit-evasion with teams of robots. In *Proceedings AAAI National Conference on Artificial Intelligence*, 2004.
12. K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.
13. L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *International Journal of Computational Geometry and Applications*, 9(5):471–494, 1999.
14. L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.
15. A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, U.K., 2002.
16. R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, 1960.
17. T. Kameda, M. Yamashita, and I. Suzuki. Online polygon search by a seven-state boundary 1-searcher. *IEEE Transactions on Robotics*, 22(3):446–460, June 2006.
18. W. Kim, K. Mechitov, J. Choi, and S. Ham. On target tracking with binary proximity sensors. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 40, Piscataway, NJ, USA, 2005. IEEE Press.
19. P. R. Kumar and P. Varaiya. *Stochastic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1986.
20. S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.
21. J.-H. Lee, S. Y. Shin, and K.-Y. Chwa. Visibility-based pursuit-evasions in a polygonal room with a door. In *Proceedings ACM Symposium on Computational Geometry*, 1999.
22. M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI National Conference On Artificial Intelligence*, 2002.
23. J. M. O'Kane and S. M. LaValle. Sloppy motors, flaky sensors, and virtual dirt: Comparing imperfect, ill-informed robots. In *Proceedings IEEE International Conference on Robotics and Automation*, 2007.
24. R. Parr and A. Eliazar. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings International Joint Conference on Artificial Intelligence*, 2003.
25. N. Shrivastava, R. Mudumbai U. Madhow, and S. Suri. Target tracking with binary proximity sensors: fundamental limits, minimal descriptions, and algorithms. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 251–264, New York, NY, USA, 2006. ACM.
26. J. Singh, U. Madhow, R. Kumar, S. Suri, and R. Cagley. Tracking multiple targets using binary proximity sensors. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 529–538, New York, NY, USA, 2007. ACM.
27. I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, October 1992.
28. S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.
29. B. Tovar, L. Freda, and S. M. LaValle. Mapping and navigation from permutations of landmarks. In *AMS Contemporary Mathematics Proc*, volume 438, pages 33–45, 2007.
30. B. Tovar, R Murrieta, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, June 2007.
31. J. Yu and S. M. LaValle. Tracking hidden agents through shadow information spaces. In *Proceedings IEEE International Conference on Robotics and Automation*, 2008. Under review.