Inferring Obstacles and Path Validity from Visibility-Constrained Demonstrations

Craig Knuth, Glen Chou, Necmiye Ozay, Dmitry Berenson*

University of Michigan, Ann Arbor, MI 48109, USA
{cknuth, gchou, necmiye, dmitryb}@umich.edu

Abstract. Many methods in learning from demonstration assume that the demonstrator has knowledge of the full environment. However, in many scenarios, a demonstrator only sees part of the environment and they continuously replan as they gather information. To plan new paths or to reconstruct the environment, we must consider the visibility constraints and replanning process of the demonstrator, which, to our knowledge, has not been done in previous work. We consider the problem of inferring obstacle configurations in a 2D environment from demonstrated paths for a point robot that is capable of seeing in any direction but not through obstacles. Given a set of *survey points*, which describe where the demonstrator obtains new information, and a candidate path, we construct a Constraint Satisfaction Problem (CSP) on a cell decomposition of the environment. We parameterize a set of obstacles corresponding to an assignment from the CSP and sample from the set to find valid environments. We show that there is a probabilistically-complete, yet not entirely tractable, algorithm that can guarantee novel paths in the space are unsafe or possibly safe. We also present an incomplete, but empirically-successful, heuristic-guided algorithm that we apply in our experiments to 1) planning novel paths and 2) recovering a probabilistic representation of the environment.

1 Introduction

In the paradigm of learning from demonstration, the assumption is often made that the demonstrator is acting (near) optimally. However, in many real world scenarios, a demonstrator only has partial information about the environment and they continuously replan as they gather information. In the context of 2D navigation, such as a robot scouting an underwater wreck or a medical assistance team navigating an unknown environment, demonstrators are able to see obstacles only when they enter the field of view and plan a path to goal with only partial knowledge of the environment, i.e. parts of the environment that have not been occluded. If we cannot access the internal map of the demonstrator, such as an occupancy map of a robot performing SLAM or the memory of a human being, we may want to reconstruct the environment using the demonstration.

Reconstructing the environment when considering visibility constraints can be challenging for a number of reasons. The demonstration may not be globally optimal, or even locally optimal as the robot may make suboptimal decisions

*This research was supported in part by NSF grants IIS-1750489 and ECCS-1553873, ONR grants N00014-17-1-2050 and N00014-18-1-2501, and a National Defense Science and Engineering Graduate (NDSEG) fellowship.

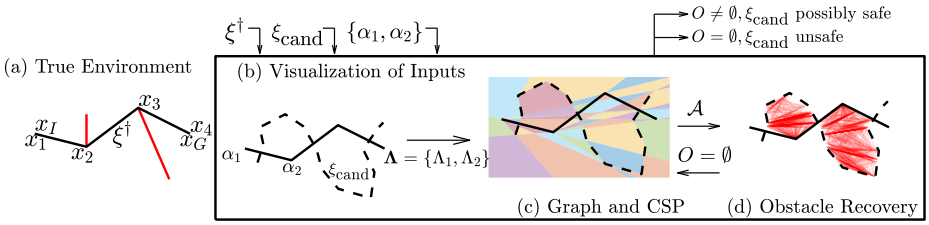

Fig. 1. An overview of the method (boxed in black). Given a demonstration $\xi^\dagger$ and survey points $\mathbb{A} = \{\alpha_1, \alpha_2\}$, we test the validity of a candidate path (dashed black) by (b) decomposing the space into cells $\mathbf{\Lambda}$ encoding visibility constraints, then sample obstacles O from a set of obstacles $\mathcal{A}$ produced by the CSP. If we find a valid obstacle configuration in $\mathcal{A}$, the candidate is possibly safe. If not, then we query for a new $\mathcal{A}$. In (d), we show many sample obstacles found by our method.

due to the lack of global environment knowledge or backtrack upon gaining information. Thus, constraint-learning methods relying on global [1, 2] or local optimality [3] of the demonstrations cannot be applied. The demonstrator is constantly obtaining more information about the environment and therefore plans paths not only based on what they currently see but also what they have already seen. The memory and vision model of the demonstrator means that obstacles in the world are all potentially intertwined; two obstacles may be individually consistent with our assumptions on the demonstration, but together are not. Furthermore, the constraint-learning problem is ill-posed [1] as many environment configurations can be consistent. Thus accounting for visibility constraints introduces unique requirements and challenges for environment reconstruction from demonstration.

We utilize information from a learned environment configuration in two ways. First, in the context of planning, we determine if a novel robot path (for a new start and goal) is certainly unsafe or possibly safe. Second, we probabilistically reconstruct an environment, for instance from the demonstrated path of a scouting robot, by generating many possible environment configurations with an associated probability measure. This method may be useful in scenarios where we do not have a map (such as a private home or disaster environment) but we do have demonstrations from agents such as medical professionals or first-responders.

Our learning approach is composed of the following steps. Given a demonstration or several sequential demonstrations, a set of survey points (points from which the demonstrator sees the environment), and a novel candidate robot path, we perform a Polar Cell Decomposition (PCD) that captures occlusions to a particular survey point. Combining cell decompositions, we construct a graph that captures configurations of occluding obstacles. On this graph, we formulate a CSP such that each assignment denotes a set of possible obstacles. We parameterize and sample from the set of obstacles to find a valid environment configuration (see Fig. 1). In certain cases, we can assert no valid environment exists and conclude the novel candidate robot path is unsafe. To the best of our knowledge, this is the first paper to address the problem of reconstructing an environment from visibility-constrained demonstrations.

The specific contributions of this work are *1)* two algorithms for generating possible 2D environment configurations subject to visibility-constrained demon-

strations; *2)* a theoretical analysis of the method; *3)* an extension to planning novel paths in the demonstrator's environment; *4)* an extension for constructing a probabilistic representation of the environment; and *5)* experimental results of running the algorithm on a scouting robot and in-home example.

2 Related Work

Our method is closely related to the literature in inverse optimal control/inverse reinforcement learning (IOC/IRL) [4–8]. These methods aim to learn a cost function from demonstrations which when optimized, generalize behavior present in the demonstrations. However, by only searching for cost functions, IOC/IRL can be poorly suited for representing richer classes of task specifications, i.e. those requiring satisfaction of hard constraints. In this work, we take steps to bridge this gap by inferring obstacle constraints, assuming that the demonstrator is acting optimally in a visibility-constrained, receding horizon fashion.

Our work is also related to constraint-learning. Much work has focused on learning local trajectory-based constraints [9–14], i.e. task-specific constraints which need not hold globally over all trajectories, while other methods [15, 16] focus on learning geometric constraints. The closest methods in the literature are [1, 2] and [3], which learn safe and unsafe regions of the state space from globally and locally optimal expert demonstrations, respectively, assuming that the demonstrator has full knowledge of the environment *a priori*. This paper removes this assumption, instead focusing on demonstrators that plan optimally while incrementally gathering information about the environment. Removing this assumption fundamentally changes the nature of the problem, requiring a radically different algorithm than previous work.

3 Preliminaries

3.1 Demonstrator's Problem

We consider a demonstrator acting as a point robot with purely kinematic constraints in $\mathbb{R}^2$. They start at position x_I and attempt to find a collision-free path to the goal x_G in the presence of static obstacles (or environment) $O \subset \mathbb{R}^2$ which is incrementally revealed via exploration resulting in a demonstration $\xi^\dagger$.

Given two points p and q, q is visible from p if and only if the open-ended line segment connecting them, denoted as $(\overline{pq} \setminus \{p, q\})$ does not intersect O. Note q may lie in O. On the demonstrated robot path $\xi^\dagger$ at time t, the demonstrator sees a region $V(t) \doteq \{q \,|\, q \text{ is visible from } \xi(t)\}$ (a 360° view with no uncertainty). This sensor model applies to a 360° camera or an omni-directional lidar, which are frequently used on mobile robots. As the demonstrator moves about the environment, they expand the known region $R(t) \doteq \bigcup_\tau V(\tau)$, $0 \leq \tau \leq t$ and the known obstacles $O(t) \doteq R(t) \cap O$.

We assume the demonstrator follows a strategy of planning an optimal path under the known obstacles $O(t)$, following that path a small amount, and then replanning. This strategy is formalized below. Let $\Xi \doteq \{\xi \,|\, \xi : [0, 1] \to \mathbb{R}^2\}$ denote the set of paths from any start to any goal, $c : \Xi \to \mathbb{R}$ be a cost function, and a and b are two points in $\mathbb{R}^2$.

Problem 1 (Demonstrator's Planning Problem [1]).

$$\begin{aligned} \tilde{\xi} \;=\; & \arg\min_{\xi} \quad c(\xi) \\ & \text{subject to} \quad \xi(0) = a,\ \ \xi(1) = b,\ \ \xi(\tau) \notin O(t)\ \ \forall \tau \in [0,1] \end{aligned} \tag{1}$$

We use the symbol $\tilde{}$ (e.g. $\tilde{\xi}$) to denote a path planned with respect to a fixed obstacle configuration to distinguish it from the executed demonstration $\xi^\dagger$. We consider cost to be the total path length. We emphasize that $\tilde{\xi}$ may not coincide with $\xi^\dagger$ as more of the environment is revealed. For example, the path ξ^1_{inf} in Figure 2 intersects with an obstacle that is not visible from x_1 and must be adjusted at x_2. However, from the following problem we know that the demonstration $\xi^\dagger$ and plan $\tilde{\xi}$ will align for at least a small segment which will allow us to verify a demonstration with respect to any obstacle O.

Problem 2 (Demonstrator's Strategy). Find $\xi^\dagger : [0,1] \to \mathbb{R}^2$ such that $\xi^\dagger(0) = x_I$, $\xi^\dagger(1) = x_G$ and is generated with the following strategy. At any time t with the demonstrator at position p, the demonstrator follows $\tilde{\xi}$, the solution to Prob. 1 for start $p = \xi^\dagger(t)$, goal g, and known obstacles $O(t)$, i.e. $\exists \delta > 0, u > 0$ such that $\xi^\dagger(t + \frac{\tau}{u}) = \tilde{\xi}(\tau)\ \forall \tau \in [0, \delta)$.

The final condition enforces that the demonstrator follows $\tilde{\xi}$ at each time t for some nonzero length. We introduce the scaling factor u for time-scaling purposes to ensure $\xi^\dagger$ is defined only on $[0,1]$. We highlight that even though the demonstrator plans optimally with respect to $O(t)$, we do not have access to the entire planned path because the demonstrator's plan may change as $O(t)$ changes. Thus methods like [1, 2] are not applicable.

3.2 Obstacles and Obstacle Vertices

Assumption 1 (Line Segment Obstacles) *We assume obstacles are curves consisting of only straight line segments.*

A consequence of Assumption 1 is that the demonstration can also be decomposed into line segments connecting vertices $\{x_i\}_{i=1}^{n+2} = \{\xi^\dagger(t_i)\}_{i=1}^{n+2}$ where $n+2$ is the number of vertices on the demonstration, and each intermediate vertex $\{x_i\}_{i=2}^{n+1}$ is coincident with an obstacle vertex. We can consider this as an extension to the well-known theorem that the shortest path between any two points in a 2D environment with polygonal obstacles is composed of straight line segments with intermediate vertices at obstacle vertices [17, Visibility Graphs]. There is a subtle distinction here in that the planned path $\tilde{\xi}$ will always have vertices at obstacle vertices, but we show in the following theorem that the executed robot path $\xi^\dagger$ (during which the known obstacles and planned paths may change) will also have vertices close to obstacles.

Theorem 1. *At time t, suppose $\tilde{\xi}$ solves Prob. 1, $\tilde{\xi}$ is not a straight line to goal, and $x^* = \tilde{\xi}(s)$ is the first point on $\tilde{\xi}$ coincident with an obstacle vertex. Then, $\xi^\dagger(t+\tau) = \tilde{\xi}(\tau)$ for all $\tau \in [0, s]$. Therefore, the demonstrator will never deviate from a plan unless at an obstacle vertex.*

[1] A minimum may not exist, but discussion is excluded for brevity. See proof of Thm. 1

For sake of brevity, we present all proofs in the appendix of the extended version [18]. Assumption 1 guarantees our method will produce sound results, but two more assumptions are needed to ensure the completeness of our method.

Assumption 2 (Non-intersecting Environment) *We assume the environment does not contain obstacles which intersect with themselves or one another.*

Assumption 3 (n Obstacle Curves) *We restrict the environment to contain at most n obstacle curves each with an endpoint at some x_i, $i \in \{2, \ldots, n+1\}$. Consequently, there are no "free-standing" obstacles, i.e. obstacles that do not intersect with an intermediate vertex of the demonstration.*

3.3 Verifying Environment Configurations

Thm. 1 informs how we can verify that $\xi^\dagger$ is the solution to Prob. 2 for a given environment configuration O. If all $\tilde{\xi}^i$'s, defined to be the solutions to Prob. 1 with start x_i and goal x_G for $1 \leq i \leq n$, align with the straight line from x_i to x_{i+1} then $\xi^\dagger$ is the solution to Prob. 2 (see Fig. 2). We can verify an environment by only considering plans originating at the first n vertices by Thm. 1.

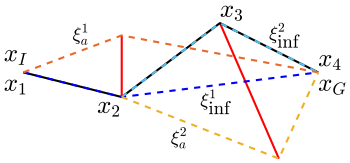

Fig. 2. Inferred ξ_{inf}^i (blue, light blue) and alternative ξ_a^i (orange, yellow) robot paths from vertices x_1 and x_2. Demo is solid black and obstacles are red. Note that the first line segment of each ξ_{inf}^i aligns with the next line segment of the demonstration. Here $\tilde{\xi}^i$ is precisely ξ_{inf}^i.

For each vertex x_i of $\xi^\dagger$ with known obstacles $O(t_i)$, $i \in \{1, \ldots, n\}$, let $\mathbb{H}^i$ be the set of homology classes of robot paths between x_i and x_G, excluding homology classes that wind around obstacles at least once. For further treatment on homology classes in the context of finding robot paths see [19]. For each class $H^i \in \mathbb{H}^i$ we calculate the shortest robot path. At least one of these robot paths (there may be multiple, say N_{inf}^i) will overlap with the next line segment of the demonstration. We define $\tilde{\xi}_{\text{inf}}^i$ as the shortest of these paths, indicating this is the *inferred robot path*. We say this robot path is inferred because it may or may not be the same as the robot path the demonstrator actually planned, depending on the true environment. The rest of the $N_a^i = |H^i| - N_{\text{inf}}^i$ robot paths that do not overlap with the next segment of the demonstration are the *alternative robot paths* $\{\tilde{\xi}_{a_k}^i\}_{k=1}^{N_a^i}$ which leads us to the next definition. Let $\tilde{\xi}_a^i$ be the shortest of these alternatives.

Definition 1 (Consistency). *If for some obstacle O, $c(\tilde{\xi}_{inf}^i) \leq c(\tilde{\xi}_a^i)$ for all $i \in \{1, \ldots, n\}$, we say the obstacle is consistent with the demonstration.*

If O is consistent with a demonstration $\xi^\dagger$, then $\xi^\dagger$ solves Prob. 2 for O. Determining consistency by examining plans only at the first n vertices implies that we can only learn the portion of the environment visible by the nth vertex.

4 Problem Statement

We assume we are given a finite set of survey points $\mathbb{A} \doteq \{\alpha_1, \alpha_2, \ldots, \alpha_{N_\alpha}\} = \{\xi^\dagger(s_1), \ldots, \xi^\dagger(s_{N_\alpha})\}$ such that the known region at each time s_j can be exactly reconstructed from only the visible region at each survey point, i.e. $R(s_j) =$

$\bigcup_{s_k \leq s_j} V(s_k)$. We assume that $\mathbb{A}$ contains $\{x_1, \ldots, x_n\}$ (all points of the demonstration except the last two). Intuitively, these survey points describe the points at which the demonstrator takes in information useful for planning. Note the demonstrator continuously sees more of the environment whereas in reconstruction we have a finite number of survey points which will enable a discrete search.

Given a demonstration $\xi^\dagger$ and survey points $\mathbb{A}$, we evaluate the safety of a given candidate robot path ξ_{cand}. We assume ξ_{cand} is safe and then seek to find a consistent environment O. If we are able to find an environment then we can conclude the candidate is possibly safe otherwise it is certainly unsafe.

Problem 3. Given a demonstration $\xi^\dagger$, survey points $\mathbb{A}$, and a candidate robot path ξ_{cand}, find a consistent environment configuration O such that $O \cap \xi_{\text{cand}} = \emptyset$.

As an extension, we also generate a probabilistic reconstruction of the environment, potentially from multiple sequential demonstrations as in the case of the scouting robot. Let $\{\xi_k^\dagger\}_{k=1}^K$ with $\xi_k^\dagger(1) = \xi_{k+1}^\dagger(0)$ for all $k \in \{1, \ldots, K-1\}$ be a set of *sequential* demonstrations.

Problem 4. Given K sequential demonstrations $\{\xi_k^\dagger\}_{k=1}^K$ and a prior on obstacles, find environment configurations $\mathbb{O}$ and a probability measure $\mathbb{P} : \mathbb{O} \to [0, 1]$ such that $\mathbb{P}$ is consistent with the prior and for each $O \in \mathbb{O}$ with $\mathbb{P}(O) > 0$ and each start $\xi_k^\dagger(0)$ and goal $\xi_k^\dagger(1)$ in $k \in \{1, \ldots, K\}$, the solution to Prob. 2 is $\xi_k^\dagger$.

5 Method

We present an algorithm to find an environment configuration O that solves Prob. 3 under Assumptions 1 - 3. In Sec. 5.1, we present a potentially intractable algorithm that is probabilistically complete which finds an environment configuration that solves Prob. 3 if one exists. In Sec. 5.3 we present an incomplete but tractable variant. First we give a few more definitions. Define the safe set $\mathcal{S} \doteq \{\xi^\dagger(\tau) | \tau \in [0, 1]\} \cup \{\xi_{\text{cand}}(\tau) | \tau \in [0, 1]\}$.

Definition 2 (Full/Partial Occlusion). *A set $X \subset \mathbb{R}^2$ is fully occluded from point p by a set Y if $Y \cap X = \emptyset$ and for all $q \in X$, $\overline{pq} \cap Y_1 \neq \emptyset$. Let $FO_p(X, Y)$ be a proposition that is true in this case and otherwise false. Furthermore, X is partially occluded from p by Y if there exists a point $q \in X$ such that $\overline{pq} \cap Y \neq \emptyset$. Let $PO_p(X, Y)$ be defined similarly.*

5.1 Probabilistically Complete Obstacle Sampling

Our method first determines the polar cell decomposition (PCD) from each survey point, combines the decompositions and survey points in a graph, formulates a constraint satisfaction problem (CSP) on the graph, and then samples obstacles corresponding to an assignment of the CSP.

PCD We perform an exact cell decomposition of the space with respect to the demonstration $\xi^\dagger$ and candidate robot path ξ_{cand} by sweeping rays from each survey point α_j resulting in a *polar cell decomposition* Λ_{α_j}. Each cell $\lambda \subset \mathbb{R}^2$ is closed and shares boundaries with adjacent cells (see Fig. 3 and 4). See [20, pp. 269-270] for a similar method using vertical lines instead of rays. This decomposition captures which cells occlude others from the survey point α_j.

Survey-Cell Graph To construct the survey-cell graph $G \doteq (N, E)$, we first intersect each cell from each Λ_{α_j} to form the full cell decomposition Λ_{full} (Fig. 4). A node $\eta = (\alpha, \lambda) \in N$ of the graph identifies a survey point $\alpha \in \mathbb{A}$ and a cell $\lambda \in \Lambda_{\text{full}}$. Intuitively, a node corresponds to viewing part of the obstacle in cell λ from the survey point α. We use the superscript η^α to denote the survey point and η^λ to denote the cell and similarly for node sequences and paths. An edge $e \in E$ exists between two different nodes (α, λ) and (α', λ') if the intersection $\mathcal{B} \doteq (\lambda \cap \lambda') \setminus \mathcal{S}$ (recall $\mathcal{S}$ is the safe set) is not a single point nor the empty set. Note that any obstacle that lies in $R(t_n)$ corresponds to a path in the survey-cell graph, also denoted a *survey-cell path*.

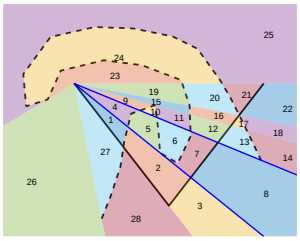

Fig. 3. A PCD with $\xi^\dagger$ (solid black) and candidate robot path ξ_{cand} (dashed black). The cells outlined in blue all share the same bounding rays so nearer cells occlude further ones, e.g. cell 5 occludes cell 7. Cell sequence identification would give $\omega_1 = 2 \to 7 \to 12 \to 16 \to 20 \to 24$.

Cell Sequence Identification Prior to formulating the CSP, we identify sets of cell sequences that may contain a solution O to Prob. 3. We accomplish this by identifying an upper bound on the cost of the shortest alternative robot path ξ_a^i and a lower bound on the cost of the inferred robot path $\tilde{\xi}_{\text{inf}}^i$ from each vertex x_i, $i \in \{1, \ldots, n\}$. See App. B in [18] for an example.

For a sequence of cells ω, we will overload ω to also mean the union of cells in ω. Now consider a tuple of cell sequences $[\omega_2, \ldots, \omega_{n+1}]$, and let $\mathbf{O} = \bigcup_i \omega_i$. Then we find $\bar{c}_a^i \doteq \max_{O \subseteq \mathbf{O}} c(\xi_a^i)$ by planning an alternative (c.f. Sec. 3.3) with the known obstacle set $\mathbf{O}$. Intuitively, an obstacle that completely fills the cells in $[\omega_2, \ldots, \omega_{n+1}]$ induces robot paths as long or longer than an obstacle that only fills part of the cells, so $\bar{c}_a^i$ is an upper bound on the cost of the shortest alternative. Furthermore, a lower bound on the cost of the inferred robot path is the length of the robot path from x_i to x_{i+1} to x_G (since the inferred path must always visit the next vertex), denoted $\underline{c}_{\text{inf}}^i$. Define $\mathbf{\Omega}_{\text{id}}$ such that for all tuples $[\omega_2, \ldots, \omega_{n+1}] \in \mathbf{\Omega}_{\text{id}}$, $\bar{c}_a^i \geq \underline{c}_{\text{inf}}^i$ for all $i \in \{1, \ldots, n\}$ (see Fig. 4). If $\mathbf{\Omega}_{\text{id}} = \emptyset$, then our algorithm guarantees the candidate robot path is unsafe.

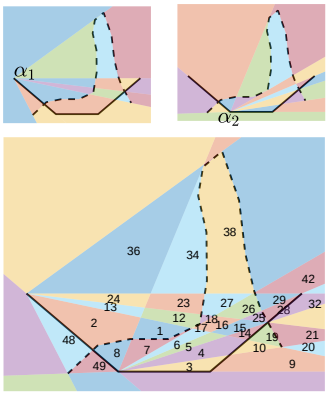

Fig. 4. The full cell decomposition Λ_{full} composed of the intersection of the two polar cell decompositions Λ_{α_1} and Λ_{α_2}. The two individual cell decompositions are shown above.

CSP On the survey-cell graph G, we seek to find an *assignment* $\mathcal{A}$ of n survey-cell paths $(\sigma_2, ..., \sigma_{n+1})$ such that we can construct n corresponding obstacles. By Asm. 3, there can be at most n obstacles in the environment. Fewer than n obstacles manifest in the survey-cell graph as n paths with some that overlap. However, we place some requirements on the paths that

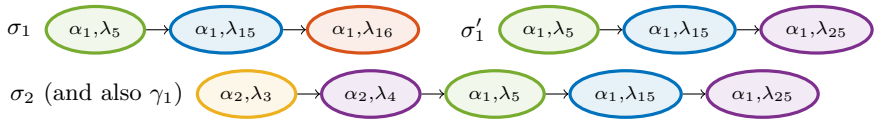

Fig. 5. Three survey-cell graphs representing two obstacles. The colors of the nodes here match the colors of the cells in Fig. 4. Combining σ_1 and σ_2 would not produce a valid path as it would branch from node (α_1, λ_{15}) to both (α_1,λ_{16}) and (α_1,λ_{25}). However, σ_1' and σ_2 do chain to form a single path γ_1 which is the same as σ_2.

overlap in order to ensure that the resulting obstacles do not occlude one another and do not intersect. This is done by virtue of *chaining*; see the definitions below and Fig. 5 for examples of survey-cell paths that can be chained or not.

Definition 3 (Chainable). *Two survey-cell paths σ and σ' are chainable if there exists a survey-cell path γ such that there is exactly one subsequence of nodes that matches the node sequences of σ and σ' (or their reverse) and each node of γ belongs to at least one of these subsequences. Let $\chi(\sigma, \sigma')$ be a proposition that is true in this case and otherwise false.*

Definition 4 (Obstacle Curve Set). *We denote $\mathcal{O}(\omega)$ as the set of obstacle curves belonging to a sequence of cells ω, i.e. $o \in \mathcal{O}(\omega)$ is a curve that satisfies $o \cap \lambda \neq \emptyset$ for all $\lambda \in \omega$ and $o \cap (\lambda \setminus \omega) = \emptyset$ for all $\lambda \notin \omega$.*

Consider two survey-cell paths σ and σ' on G that visit the same cell. Then one of two cases can occur: (i) there may exist two corresponding obstacle curves $o \in \mathcal{O}(\sigma^\lambda), o' \in \mathcal{O}(\sigma'^\lambda)$ such that neither obstacle occludes the another from any corresponding survey point. For instance, if two survey-cell paths terminate at the same node η, then the two obstacles curves could both lie in the same cell with a gap between them (from the perspective of the survey point) or (ii) there may not exist any obstacle curves o, o' such that one does not occlude the other. In this case, we constrain the resulting obstacle curves to be combined into one obstacle curve so that both can be seen from the same survey points. If they must be combined, we enforce that the paths σ and σ' are chainable.

As each obstacle begins at an intermediate vertex of the demonstration, we require that each path begins at a cell that the vertex lies on its boundary. Formally, let $\Lambda_{\text{full}}^{\text{start},i}$ be the set of cells such that for each $\lambda \in \Lambda_{\text{full}}^{\text{start},i}$, $\lambda \in \Lambda_{\text{full}}$ and $x_i \in \partial\lambda$. Let $\sigma^\lambda(1)$ be the cell of the first node on the survey-cell path. Then the CSP is formulated below:

$$\text{Find} \quad \sigma_2, \ldots, \sigma_{n+1} \quad \text{subject to}$$

$$\forall i, \quad \sigma_i^\lambda(1) \in \Lambda_{\text{full}}^{\text{start},i} \tag{2}$$

$$[\sigma_2^\lambda, \ldots, \sigma_{n+1}^\lambda] \in \mathbf{\Omega}_{\text{id}} \tag{3}$$

$$\forall i, i' \ \nexists \eta \in \sigma_{i'} \text{ s.t. } \forall o \in \mathcal{O}(\sigma_i^\lambda), \ \text{FO}_{\eta^\alpha}(\eta^\lambda, o) \tag{4}$$

$$\begin{aligned} &\forall i, i' \text{ if } \exists \eta \in \sigma_i, \sigma_{i'} \text{ then } \exists o_i \in \mathcal{O}(\sigma_i^\lambda), o_{i'} \in \mathcal{O}(\sigma_{i'}^\lambda) \text{ s.t.} \\ &\quad (o_i \cap o_{i'} = \emptyset \wedge \neg\text{PO}_{\eta^\alpha}(o_{i'}, o_i) \wedge \neg\text{PO}_{\eta^\alpha}(o_i, o_{i'})) \vee \chi(\sigma_i, \sigma_{i'}) \end{aligned} \tag{5}$$

Constraint (2) requires that each survey-cell path starts at a cell containing the vertex x_i. Constraint (3) enforces that the resulting survey-cell paths lie in

$\mathbf{\Omega}_{\text{id}}$, where $\mathbf{\Omega}_{\text{id}}$ is the set where there exists a higher cost alternative (c.f. Sec. 5.1). Constraint (4) enforces that no obstacle curve will always be occluded by another obstacle curve from the respective survey point. This is needed in order to ensure that obstacles are revealed at appropriate points in the demonstration. As discussed earlier, constraint (5) enforces that two survey-cell paths that share the same node either chain or contain two obstacles that do not occlude one another in order to ensure that obstacles can be connected if needed. We note (4) and (5) can be checked without sampling obstacle curves o since we know what cell the endpoints of obstacles must lie in and therefore can calculate the occluded space for that particular cell sequence.

The CSP can return failure in finite time in the case that constraint (3) prevents any otherwise possible assignment. In this case, we return that the candidate robot path is unsafe. However, the CSP may also admit infinite length paths, such as cycling between two nodes with different survey points but the same cell, which prevents the CSP from terminating. We address this shortcoming in the tractable version of the CSP in Sec. 5.3.

Obstacle Recovery For sake of brevity, we describe obstacle recovery at a high level here (see a detailed explanation in App. C of [18]). We seek to find $m \leq n$ obstacles in $\mathbb{R}^2$ (curves parameterized by line segments) from the assignment $\mathcal{A}$. We will first chain survey-cell paths if necessary, producing a reduced set of survey-cell paths. For the reduced set $\{\gamma_1, \ldots, \gamma_m\}$, we partition each survey-cell path γ_κ into the minimum number of nonoverlapping survey-cell paths $\nu_{\kappa,1}, \ldots, \nu_{\kappa,K_\kappa}$ such that each path is only seen from one survey point (see Fig. 6).

Corresponding to each of these paths $\nu_{\kappa,l}$, we sample an obstacle segment $\zeta_{\kappa,l}$ by sampling lines that are entirely visible to the survey point $\nu^{\alpha}_{\kappa,l}$ and lie in the cell sequence $\nu_{\kappa,l}$. This produces m obstacles such that $o_\kappa = \bigcup_l \zeta_{\kappa,l}$ for $\kappa \in \{1, \ldots, m\}$. To handle sampling obstacles with an arbitrary number of line segments, we require our obstacle segment sampler to be capable of sampling curves with a specified number of line segments. Define a *segment discretization* as $\Delta \doteq \{k_{1,1}, \ldots, k_{m,K_m}\}$. To sample an obstacle configuration O, we sample each obstacle segment $\zeta_{\kappa,l}$ such that $\zeta_{\kappa,l}$ consists of $k_{\kappa,l}$ line segments and satisfies two constraints: the resulting obstacle curves o_κ are continuous and visit the necessary vertices of the demonstration (also called *point constraints*). In brief, we enforce continuity by restricting adjacent obstacle segments $\zeta_{\kappa,l}$ and $\zeta_{\kappa,l+1}$ to intersect at a *join point* $\beta_{\kappa,l}$.

Probabilistically-Complete Obstacle Sampling is summarized in Alg. 1. The function `AssignShorterThan`(K_1) finds all assignments of the CSP with survey-cell paths no longer than K_1. `DiscLessThan`(K_2) gives all discretizations Δ such that $k \leq K_2$ for all $k \in \Delta$. `SampleObs` follows the procedure described above and `Valid`(O) tests if O is valid according to Def. 1.

5.2 Analysis of Algorithm 1

Suppose there exists O such that $\xi^\dagger$ solves Prob. 2 and O satisfies Asm. 1 - 3. Suppose we can perturb each vertex of O (other than those coincident with a

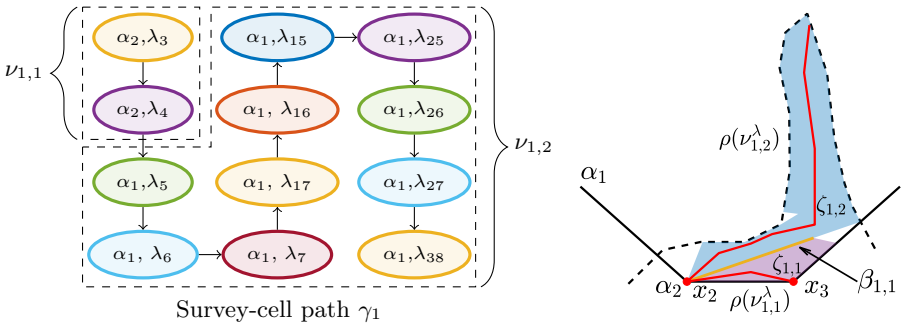

Fig. 6. On the left is shown a survey-cell path γ_1 on the survey-cell graph corresponding to Fig. 4. The path is partitioned into two survey-cell segments, $\nu_{1,1}$ and $\nu_{1,2}$, each with a unique survey point. On the right is shown the candidate (dashed black), demonstration (solid black), and cell sequences corresponding to the survey-cell path segments in blue and purple. The corresponding obstacle segments $\zeta_{1,1}$ and $\zeta_{1,2}$ (in red) must intersect at a join point on the yellow line. This obstacle is also required to intersect with two points constraints in red, x_2 and x_3.

Algorithm 1: ProbComp Obstacle Sampling

Input: $\xi^{\dagger}$, ξ_{cand}, $\mathbb{A}$
Output: possibly safe or unsafe

$\Lambda_{\text{full}} \leftarrow$ `FullCellDecomposition(`$\xi^{\dagger}$, ξ_{cand}, $\mathbb{A}$`)`
$G \leftarrow$ `ConstructGraph(`Λ_{full}, $\mathbb{A}$`)`, $\mathbf{\Omega}_{\text{id}} \leftarrow$ `CellSeqID(`G, $\xi^{\dagger}$`)`
if $\mathbf{\Omega}_{id} = \emptyset$ **then return** *unsafe*
`CSP` $\leftarrow$ `ConstructProbCompCSP(`G, $\mathbf{\Omega}_{id}$`)`, $K_1 \leftarrow 1$, $K_2 \leftarrow 1$
while *True* **do**
  $\mathcal{A}_1, .., \mathcal{A}_{N_1} \leftarrow$ `CSP.AssignShorterThan(`K_1`)`
  for $n_1 \leftarrow 1$ **to** N_1 **do**
    $\Delta_1, .., \Delta_{N_2} \leftarrow$ `DiscLessThan(`K_2`)`
    for $n_2 \leftarrow 1$ **to** N_2 **do**
      $O \leftarrow$ `SampleObs(`$\mathcal{A}_p$, Δ_q`)`
      if `Valid(`O`)` **then return** *possibly safe*
  $K_1 \leftarrow K_1 + 1$, $K_2 \leftarrow K_2 + 1$

vertex of the demonstration) within an ϵ-ball that results in a set of consistent configurations $\mathbb{O}_{\text{soln}}$ that also satisfy our assumptions.

Theorem 2 (Probabilistic Completeness). *Let $\mathbb{O}_{n_1}$ be the set of samples drawn by the n_1^{th} iteration of the outer loop of Alg. 1. Then, under Asm. 1 - 3 and for some $0 < K < \infty$ and $0 < \epsilon_2 < 1$,*

$$\mathbb{P}(\mathbb{O}_{n_1} \cap \mathbb{O}_{soln} \neq \emptyset) \geq 1 - (1 - \epsilon_2)^{n_1 - K} \qquad \forall n_1 \geq K \tag{6}$$

As a consequence to Thm. 2, $n_1 \to \infty$ implies $\mathbb{P}(\mathbb{O}_{n_1} \cap \mathbb{O}_{\text{soln}} \neq \emptyset) = 1$.

Theorem 3 (Candidate Unsafe). *Suppose Assumption 1 and 3 is satisfied and $\mathbf{\Omega}_{id} = \emptyset$. Then the candidate robot path ξ_{cand} is unsafe.*

5.3 Heuristic-guided Obstacle Sampling

Alg. 1 is intractable for a number of reasons, including the infinite length paths already mentioned. As another example, the number of discretizations with increasing K_2 grows factorially. To make the method tractable, we introduce and

modify constraints of the CSP to speed the search for a viable assignment. Furthermore, we present a method for efficiently parameterizing obstacle segments and introduce a gradient ascent procedure. These changes improve tractability but sacrifice completeness of the method.

Tractably Handling Occlusions Though constraint (4) captures exactly what survey-cell paths necessarily contain an obstacle that occludes others, calculating occlusions can be survey-cell path dependent and we empirically found it to be more efficient to consider partial occlusions instead of full occlusions. This change necessitates the addition of a *pseudolayer* (defined below). This pseudolayer captures instances of survey-cell paths where an obstacle partially occludes other cells in the path, but there still exists an obstacle that lies in the obstacle curve set that does not self occlude.

Definition 5 (Pseudolayer). *Suppose a survey-cell path σ_i (or its reverse) has a subsequence of nodes $\mathcal{N}$ such that the first node η_1 partially occludes the last node η_2 from the last node's survey point α. This subsequence is a pseudolayer if there exists a ray r originating from the survey point α such that $r \cap \eta \neq \emptyset$ for all $\eta \in \mathcal{N}$. In this case, we say $\mathcal{P}_i(\eta_1, \eta_2)$ is true and otherwise false.*

For example in Fig. 4, the cells 18, 16, and 15 form a pseudolayer with respect to α_1 but cells 18, 27, 26, 25, and 15 do not. Furthermore, we require a little more notation before defining the tractable CSP. If ω is a sequence, let $\omega(k)$ be the k^{th} entry in the sequence. Recalling $\mathbb{A}$ is the set of survey points, let $\mathbb{A}(x_i)$ be the set of survey points on the demonstration before the point x_i. Then,

$$\text{Find} \quad \sigma_2, \ldots, \sigma_{n+1} \quad \text{subject to}$$

Constraints (2) and (3)

$$\forall i, i', \eta_1 \in \sigma_i, \nexists \eta_2 \in \sigma_{i'} \text{ s.t. } \mathrm{PO}_{\eta_2^\alpha}(\eta_2^\lambda, \eta_1^\lambda) \wedge (i \neq i' \vee \neg \mathcal{P}_i(\eta_1, \eta_2)) \tag{7}$$

$$\forall i, i' \text{ if } \exists \eta \in \sigma_i, \sigma_{i'} \text{ then } \chi(\sigma_i, \sigma_{i'}) \tag{8}$$

$$\forall i, j \;\; \alpha_j \in \sigma_i^\alpha \Rightarrow \alpha_j \in \mathbb{A}(x_{i+1}) \tag{9}$$

$$\forall i \;\; \nexists k_1, k_2, k_1 \neq k_2 \text{ s.t. } \sigma_i^\lambda(k_1) = \sigma_i^\lambda(k_2) \tag{10}$$

$$\forall i, i', \eta_1 \in \sigma_i, \nexists \eta_2 \in \sigma_{i'} \text{ s.t. } \eta_1^\lambda = \eta_2^\lambda, \eta_1^\alpha \neq \eta_2^\alpha \tag{11}$$

We modify constraint (4) into (7) so that if any cell in a survey-cell path occludes a particular cell λ from a survey point α, then the corresponding node $\eta = (\alpha, \lambda)$ cannot lie on any path unless they lie in a pseudolayer. In this case, it still may be possible to construct an obstacle curve that does not occlude itself. We modify the overlapping constraint from (5) into (8) so that all overlapping paths are chained. We include (9) to ensure no obstacle is seen by a future survey point. We include (10) to eliminate paths that visit the same cell multiple times. Finally, (11) is included so that no cell is seen from different survey points.

Specialized Obstacle Recovery We briefly describe the specialized obstacle sampling here and leave a more detailed discussion for App. F in [18]. Essentially, we perform a convex decomposition of each cell sequence of a survey-cell segment

Algorithm 2: Heuristic-Guided Obstacle Sampling

Input: $\xi^\dagger$, ξ_{cand}, $\mathbb{A}$
Output: possibly safe, guaranteed unsafe, undecided

$\Lambda_{\text{full}} \leftarrow$ `FullCellDecomposition(`$\xi^\dagger$, ξ_{cand}, $\mathbb{A}$`)`
$G \leftarrow$ `ConstructGraph(`Λ_{full}, $\mathbb{A}$`)`, $\mathbf{\Omega}_{\text{id}} \leftarrow$ `CellSeqID(`G, $\xi^\dagger$`)`
if $\mathbf{\Omega}_{id} = \emptyset$ **then return** *guaranteed unsafe*
`CSP` $\leftarrow$ `ConstructHeuristicCSP(`G, $\mathbf{\Omega}_{id}$`)`
for $i_{assign} = 1$ **to** I_{assign} **do**
  $\mathcal{A} \leftarrow$ `CSP.NextAssign()`
  for $i_{initial} = 1$ **to** $I_{initial}$ **do**
    $\theta \leftarrow$ `ObsParam(`$\mathcal{A}$`)`
    for $i_{grad} = 1$ **to** I_{grad} **do**
      $\theta \leftarrow \theta + \mu$ `CalculateGrad(`θ`)`
      if `Valid(`$O(\theta)$`)` **then return** *possibly safe*
return *undecided*

$\nu_{\kappa,l}$ by intersecting rays from the corresponding survey point $\nu^{\alpha}_{\kappa,l}$. This, along with join points and point constraints, forms a parameterization θ. We sample θ and optionally perform gradient ascent to arrive at a consistent obstacle.

Computational Complexity We present a detailed derivation of the computational complexity in App. H of [18]. In brief, supposing there are a total of N_v vertices on the demonstration and candidate combined, the running time is dominated by the CSP which considers $O(n(N_\alpha^2)\hat{}(N_v^{N_\alpha}))$ paths. A key takeaway is that the number of survey points dominates the computational complexity.

5.4 Applications in Planning and Scouting

To generate candidate robot paths, we can utilize planners which reason over the safety of entire paths, such as path integral control [21], cross entropy method [22], or hit and run [23]. At a high level, each of these methods rely on sampling control actions and iteratively refining to produce candidate robot paths.

To adjust our method for sequential demonstrations $\{\xi_k^\dagger\}_{k=1}^K$, we note that the inferred and alternative plans made from each intermediate vertex of each demonstration must plan to the respective goal of each demonstration. To solve Prob. 4, instead of terminating our algorithm after finding one feasible environment configuration, we run it for all feasible assignments and sample multiple obstacles for each assignment resulting in environment configurations $\mathbb{O}$. Different probability measures may be employed based on prior knowledge, or simply uniformly, i.e. $\mathbb{P}(O) = |\mathbb{O}|^{-1}$ for all $O \in \mathbb{O}$.

6 Experimental Results

We will apply our method to three experiments that focus on planning novel paths in the environment. In the first, we take a candidate, demonstration, and set of survey points and seek to find a consistent environment. This experiment provides examples of obstacles that can be found with our method and situations in which we reject the candidate as certainly unsafe. In the second

experiment, we generate a set of candidate trajectories using hit-and-run [23] and run our method on each to determine a set of candidates that are possibly safe in some environment. This experiment evaluates the discriminatory ability of our method, i.e. which candidates are identified as guaranteed unsafe, possibly safe, or undecided. We additionally evaluate how many possibly safe candidates are actually safe in the true environment. In the third experiment, we run our method on sequential demonstrations to generate an obstacle "cloud" or a set of consistent environments. We evaluate the quality of this cloud by comparing its similarity (via an appropriate distance metric) to the true obstacle and by evaluating the safety of novel planned paths.

All experiments were implemented in MATLAB and run on an Intel® Core™ i7-6700 CPU @ 3.40GHz x 8 with 32 GB of RAM. We run Alg. 2 setting $I_{\text{assign}} = 10$, $I_{\text{grad}} = 5$, and $I_{\text{initial}} = 4$. In all examples the survey points are all vertices of the demonstration except for the last two.

6.1 Single Candidate

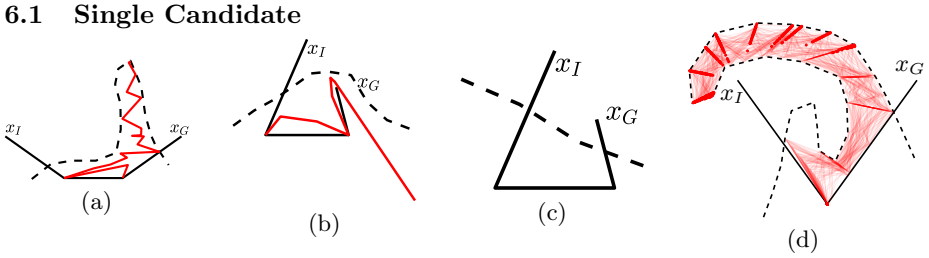

Fig. 7. Example environments with different candidates. Demonstration (solid black), candidate (dashed black), learned obstacle (red). In (d), we show 100 found obstacles with locally supporting vertices shown with a red dot.

In Fig. 7, we show demonstrations with different candidates and resulting environment configurations found by our method. We see that it is capable of finding obstacles that coincide with multiple points of the demonstration (Fig. 7a) rather than finding n separate obstacles. The method can also exploit previously gained knowledge of the demonstrator (Fig. 7b) since the obstacle is only entirely visible from x_I. Furthermore, in Fig. 7c, our method does not return a configuration by constraint (3) which is correct by Thm. 3. In Fig. 7d, we show 100 obstacles found for the given demonstration and candidate with locally supporting vertices for each obstacle shown with a red dot.

6.2 Evaluation of Many Candidates

We consider the problem of planning a novel path to aid an individual inside of a house where access to a map is forbidden for privacy reasons. However, we have access to demonstration provided by a previous medical team. Rather than plan a single path that may be safe, our goal is to plan many possibly safe paths.

For a given demonstration (see Figure 8), we sample 100 candidates using hit-and-run [23] and evaluate each for safety by solving Prob. 3. We run this experiment 50 times. The classification of safe paths results in an average of 13.56 ± 7.31 possibly safe, 22.40 ± 8.84 undecided, and 64.04 ± 13.02 certainly unsafe (see Fig 9). Of the possibly safe paths, an average of 4.68 ± 4.82 are safe with respect to the true environment. With regards to running time, we limit execution

of the CSP to 1 second. Generating trajectories took an average of 0.1405 seconds per trajectory and finding consistent environment took 1.9692 seconds per candidate, totaling 2.1097 seconds.

Since we assume the robot acts with kinematic constraints, we can execute a potentially safe path and backtrack if we discover it is not safe. If there exists at least one possibly safe path which is truly safe, then we will safely reach the goal with this strategy. In Fig. 8, note the portion of the dashed black and orange paths near the demonstration vertex renders the black candidate certainly unsafe which highlights that our algorithm is capable of correctly classifying two similar paths.

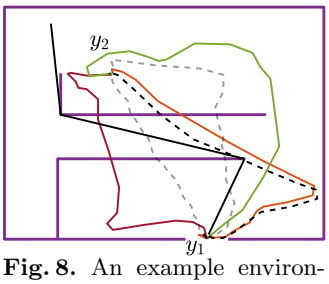

Fig. 8. An example environment of a house (purple), where we seek to plan a path to position y_2. Demonstration is solid black. Our method finds three possibly safe candidate paths (red, orange, green). Also shown are unsafe candidate robot paths (dashed gray and black).

6.3 Sequential Demonstrations and Probabilistic Environment Representation

In Fig. 10a, we show the results of reconstructing the environment from the sequential demonstrations of a scouting robot. We ran the CSP for 10 hours and then used the resulting assignments for recovering obstacles if the assignments produced a configuration O that solved Prob. 4. In this case, the first assignment was found in 20 minutes and the next 3 in 2 hours. We sampled 100 obstacles from the 5 assignments found, taking 2 additional minutes. We only sampled consistent obstacles from 3 of the 5 assignments.

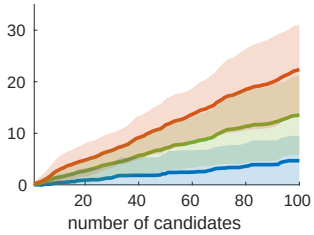

Fig. 9. Number of possibly safe (green) and undecided (orange) determined by the method. The remainder is determined certainly unsafe. In blue is the number of safe candidates with respect to the true environment. Shaded area is one standard deviation.

To evaluate success, we measure the distance of the true obstacles $\mathbb{O}_{\text{true}}$ to the sample cloud $\mathbb{O}$ as samples are taken. Let the distance be $d(\mathbb{O}_{\text{true}}, \mathbb{O}) \doteq \max_{o \in \mathbb{O}_{\text{true}}} \min_{o' \in \mathbb{O}} ||o - o'||_2$, approximated by finely subsampling the cloud and true obstacle (see Fig. 10b). This metric is specific to the given environment, but the maximum distance asymptotically trends towards a lower bound for each assignment and sampling from a new assignment further decreases the distance.

We can use this obstacle cloud to also plan new paths that maximize the likelihood of safety. If we interpret each sample as equally likely, this path planning problem reduces to Minimum Constraint Removal (MCR) [24]. After a batch of 10 samples are taken, we discretize the space and plan a path using (MCR) for 5000 randomly selected start and goal configurations within the bounding box of the environment (see Fig. 10cd). We interpret the fraction of planned paths that do not intersect with the true obstacle as the estimated probability of safety. We see that safety improves with only a few samples and continues to improve with more samples and new assignments. Drops in safety are due to random sampling of the start and goal as well as obstacle samples.

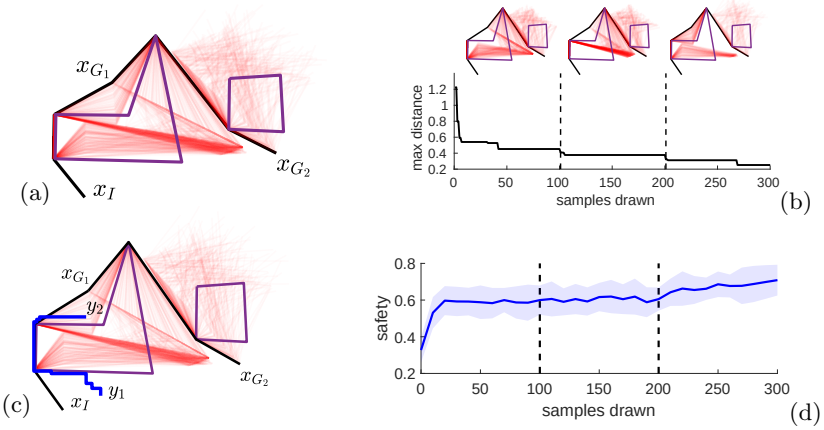

Fig. 10. (a) The true environment (purple) for a scouting robot with intermediate goals x_{G_1} and x_{G_2}. Note the true environment violates the assumptions as the obstacles are enclosed. 300 sample obstacles from 3 assignments are transparently plotted in red. (b) Distance of true obstacle to the cloud. The magnitude of the distance depends on the scale of the obstacle, however, a downward trend in distance is apparent as more samples are drawn. The resulting samples, true obstacles, and demonstration are shown above the corresponding range for each assignment. A dotted line indicates when samples are taken from a new assignment. (c) A safe example plan (blue) from y_1 to y_2 planned with MCR (d) Estimated probability of safety of planning with MCR for random start and goal with 3-σ bound.

This environment is challenging since the demonstrator navigates around one obstacle in both demonstrations. Our method infers the knowledge gained from previous demonstrations in reconstructing obstacles in future demonstrations.

7 Conclusion

In this paper we present two methods for the novel problem of inferring obstacles and path validity from visibility-constrained demonstrations. The first is probabilistically complete in finding obstacle configurations that solve Prob. 3, but is intractable to execute. The second is a tractable method that is capable of finding a subset of the possible true environment configurations and in some cases can assert the candidate path is unsafe. This method can be applied to planning problems or environment reconstruction either online or offline.

This work could be improved upon by extending to obstacles that are enclosed, can intersect or are free-standing (potentially by searching for subgraphs in the CSP) will allow generalization to many environments. Furthermore, extension to multiple demonstrations is possible as long as obstacles can be sampled in a way that satisfies visibility constraints for both demonstrators. Additionally, it may be possible to use the structure of the demonstration (connected line segments) to extend to other vision models such as a forward facing cone. Moreover, assuming the demonstrator optimally plans paths can be restrictive when using human demonstrations; we can address this by modifying verification to partially align inferred plans with the demonstration.

References

1. G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations," *WAFR*, 2018. [Online]. Available: http://arxiv.org/abs/1812.07084
2. G. Chou, N. Ozay, and D. Berenson, "Learning parametric constraints in high dimensions from demonstrations," *Conference on Robot Learning (CoRL)*, 2019. [Online]. Available: http://arxiv.org/abs/1910.03477
3. ——, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robotics Autom. Lett.*, vol. 5, no. 2, pp. 3682–3690, 2020.
4. P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," ser. ICML, 2004.
5. A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *ICML*, San Francisco, CA, USA, 2000, pp. 663–670.
6. "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, 2009.
7. R. E. Kalman, "When is a linear control system optimal?" *Journal of Basic Engineering*, vol. 86, no. 1, pp. 51–60, Mar 1964.
8. N. D. Ratliff, J. A. Bagnell, and M. Zinkevich, "Maximum margin planning," in *ICML*, 2006.
9. C. Li and D. Berenson, "Learning object orientation constraints and guiding constraints for narrow passages from one demonstration," in *ISER*. Springer, 2016.
10. N. Mehr, R. Horowitz, and A. D. Dragan, "Inferring and assisting with constraints in shared autonomy," in *CDC*, Dec 2016, pp. 6689–6696.
11. A. L. Pais, K. Umezawa, Y. Nakamura, and A. Billard, "Learning robot skills through motion segmentation and constraints extraction," HRI, 2013.
12. G. Ye and R. Alterovitz, "Demonstration-guided motion planning," in *ISRR*, 2011.
13. S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *HRI*, 2007, pp. 255–262.
14. ——, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *IROS*, 2008.
15. L. Armesto, J. Bosga, V. Ivan, and S. Vijayakumar, "Efficient learning of constraints and generic null space policies," in *ICRA*, 2017.
16. C. Pérez-D'Arpino and J. Shah, "C-LEARN: learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *ICRA*, 2017.
17. M. de Berg, *Computational geometry: algorithms and applications*. Springer, 2000.
18. C. Knuth, G. Chou, N. Ozay, and D. Berenson, "Inferring obstacles and path validity from visibility-constrained demonstrations," *WAFR*, 2020. [Online]. Available: https://arxiv.org/abs/2005.05421
19. S. Bhattacharya, M. Likhachev, and V. S. A. Kumar, "Search-based path planning with homotopy class constraints in 3d," in *AAAI*, 2010.
20. S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
21. H. J. Kappen, "An introduction to stochastic control theory, path integrals and reinforcement learning," *Cooperative Behavior in Neural Systems*, 02 2007.
22. P. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals OR*, vol. 134, no. 1, pp. 19–67, 2005.
23. Y. Abbasi-Yadkori, P. L. Bartlett, V. Gabillon, and A. Malek, "Hit-and-run for sampling and planning in non-convex spaces," in *AISTATS*, 2017.
24. K. Hauser, "The minimum constraint removal problem with three robotics applications," *IJRR*, vol. 33, no. 1, pp. 5–17, 2014.