2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Homotopy-Driven Exploration of Human-made Spaces Using Signs

Claire Liang[1] and Hadas Kress-Gazit[2]

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561862

Abstract— **Robots deployed in airports, malls, and stadiums today require expert oversight, pre-provided maps, and infrastructure. These systems are engineered to their specific deployment spaces and rely heavily on geometric maps for motion planning. In this work we consider a robot with only local-sensing and present a navigation strategy that uses signs and homotopy classes to fuel planning. We prove that in the worst-case, this strategy still maintains the probabilistic completeness that sampling based motion planners provide. Furthermore, we experimentally show that this exploration strategy results in 100% goal completion in real airport floor plans, and demonstrate how the robot's sensing capabilities affects efficiency. We also show the effect of the environment variables, such as total number of obstacles, and density of obstacles, on navigation.**

I. Introduction

Mobile robots deployed in human-made spaces such as malls, airports, and stadiums, use perfect, pre-provided maps to localize and plan or need to create their own [2], [20]. Furthermore, they often need human expert oversight, ready to take over at points of failure[9]. However, human navigators require the same level of knowledge or constant guidance from an expert to plan. As a consequence, for robots whose primary role is to interact and navigate with humans, this discrepancy in assumed knowledge can result in awkward or failed interactions [2], [8]. Humans are regularly able to plan in these spaces using no map and only their local knowledge, but to have a robot navigate with the same input while maintaining global completeness guarantees is a difficult task.

If a robot is able to plan using only local sensing and no provided map, we gain five main advantages: (i) We can remove the need for environment specific engineering or a costly, detailed, pre-provided metric map; (ii) If there are changes in the space, such as temporary construction, the robot can adapt on its own without human intervention; (iii) By using only local-sensing, the robot doesn't need to localize and re-plan trajectories in a global map, and instead plans on the fly; (iv) Since the robot's sensing aligns with human's knowledge of their space, if navigation is paired with an interaction task–such as a guide robot– the robot is able to communicate about space in human terms. (v) The robot is particularly suited to service robotics applications where the robot has never previously seen the space and is likely to be repeatedly thrown into new environments.

[1]Claire Liang is with The Department of Computer Science, Cornell University, Ithaca, NY, USA cliang@cs.cornell.edu

[2]Hadas Kress-Gazit is with The Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, USA hadaskg@cornell.edu

Fig. 1. An example of a real sign from EIN airport.

In this work we present an algorithm for a robot with no map and a fixed radius of local sensing to navigate to a goal using signs that already exist in the space. Previous work [13] relies on a minimum sensing radius assumption to ensure global completeness properties for navigation. The technique in this work gets rid of the necessity for a minimum sensing radius and creates a way for a robot to systematically explore their environment using a lightweight, topological representation of the space. We prove probabilistic completeness and we show the impact of sensing radius on navigation in real human-made spaces as well as artificially generated environments.

II. Related work

Airport robots deployed today have failed human interactions due to awkward behavior caused by replanning [8]. Prior work [13] presents a local-sensing only method for navigation that is able to achieve global guarantees by generating a globally consistent skeletonization within its local sensing representation. The robot then uses real life signage to find its goal. The robot plans in a manner much closer to its human counterparts than the real deployed robots in work such as [8], [10], [20]. However [13] requires the sensing radius to be twice as wide as the diameter of the largest inscribed circle in the environment. If not, when the robot enters a region larger than its sensing sphere it is lost. In this paper, we fill in this gap, presenting an exploration strategy for these 'open-spaces' that builds on the theme of human strategies by using signs and maximizing topological diversity when exploring.

Using only local sensing is a form of minimalist sensing representation. Many minimal sensing approaches consider multi-robot scenarios, such as swarm coordination [3], [6], [15] rather than a single agent in an environment with non-robot agents. Other classic minimal sensing approaches use combinatorial space representations [18] such as visibility-based planning [7] or ray-crossing strategies [19] and capture topological information rather than geometric detail. The

benefits of these strategies are that they assume little about the robot's capabilities and attempt to abstract away unnecessary details so the robot is able to plan with only the amount of information necessary. The work in this paper uses this philosophy to have the robot plan on the fly instead of having to take burdensome replanning steps during execution.

Classic exploration methods for unknown environments ([11], [12])— especially strategies used in search and rescue or field missions ([4], [5])— tend to assume much sparser information than is available in human-made spaces. These methods do not use context, such as signage, to bootstrap the exploration and navigation process. These strategies are also largely defined in metric terms while human counterparts often reason about space topologically (in terms of its general shape). Strategies that do use homotopy for sampling based planning ([1], [17]) can be used in conjunction with our strategy, but on their own do not use the global structure of the space to guide planning.

III. Definitions

We use the following definitions and notation:

Homotopy Class: Two trajectories are considered to be "homotopic" if one can be continuously deformed to the other. Homotopy as an equivalence relation can be used to establish "homotopy classes" of trajectories. [16]

Time: Time is measured in discrete steps $t \in \mathbb{N}$.

Environment: The environment $E \subset \mathbb{R}^2$ is polygonal with static polygonal obstacles $o_i \in O$.

Sign: A sign $s = (p, d, a)$ in a human-made space provides high-level directions. The sign is defined by a position where the sign is located (p), a destination (d), and an arrow (a) that represents a general heading angle to navigate to destination d. We denote S as all signs in E and S_t as all signs that the robot has seen up to time t.

Robot: Robot state at time t is position $\rho(t) \in \mathbb{R}^2$ and the sensing sphere with radius r, ($B_r(t)$), centered at $\rho(t)$

Trajectory: The trajectory the robot has up to time t is τ_t, where τ_1 is the path from $\rho(0)$ to $\rho(1)$ and $\tau_t = [\tau_{t-1},$ path from $\rho(t-1)$ to $\rho(t)]$.

IV. Problem Formulation

Given a robot with initial position $\rho(0)$, sensing sphere $B_r(t)$, environment E with obstacles O, goal $g \in E$ and signs S, find a controller such that $\rho(t) = g$.

A. Assumptions

We assume an omnidirectional robot with a history of its trajectory and the ability to identify and distinguish distinct obstacles. We assume signs to be visible from any angle so the robot is able to read and interpret signs when within r of the sign's location.

V. Approach

The strategy for the robot to find the goal is to use signs to guide the robot's direction forward, and to use a homotopy guided strategy when exploring the space.

When $B_r(t)$ contains ≥ 2 disjoint segments of the boundary of E (such as in a corridor), the robot uses the algorithm from [13] to follow the corridor as explained in section V-A; If not, meaning the robot is in an open space, the robot follows the exploration strategy presented in section V-D.

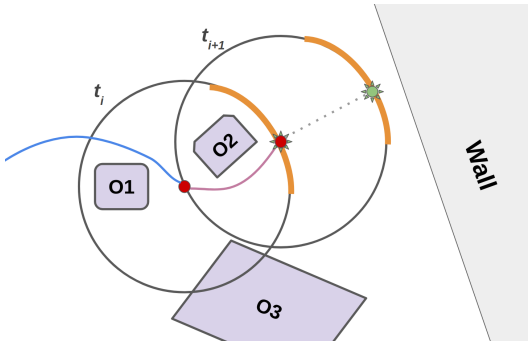


Fig. 2. Example of robot (positioned at red dots), sensing representation (circles), and obstacles (light purple) in an environment at two time steps, t_i and t_{i+1}. The robot's trajectory prior to t_i is in blue, the trajectory planned during t_i and taken between t_i and t_{i+1} is in pink, the trajectory planned during t_{i+1} is dotted. Waypoints at each timestep are green suns ($\rho(t_{i+1})$ is centered on the waypoint $gs(t_i)$) and the sign direction range is represented as the orange arc on each sensing circle.

A. Prior Work: Medial Axis-based Planning

A medial axis is the skeletonization defined by the centroids of maximally inscribed spheres in a space (the lines in fig. 3 are a construction of the medial axis). Work from [13] uses the medial axis, signs, and only local sensing to navigate to a goal; however, it assumes that the sensing radius $r \geq 2d$ where d is the widest corridor in the space (the diameter of the largest inscribed circle in E). When $r \geq 2d$, "medial axis" constructed with $B_r(t)$ provably aligns with the global medial axis within the $B_{\frac{r}{2}}(t)$ ball as shown in fig. 3. [13] uses the medial axis as a graph structure and converts sign directions to edge directions. However in spaces where the global medial axis cannot be reliably reconstructed in $B_r(t)$ (which would happen if the robot cannot meet the $r \geq 2d$ requirement), this algorithm does not apply.

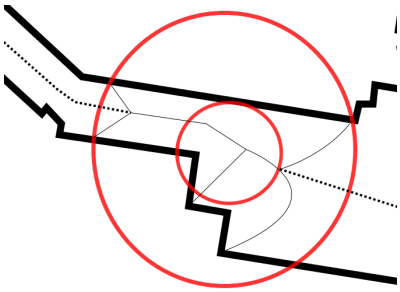


Fig. 3. An example of the global medial axis constructed in the $B_{\frac{r}{2}}(t)$ given local sensing in $B_r(t)$. The two concentric red circles are $B_{\frac{r}{2}}(t)$ and $B_r(t)$. The dashed lines are the global medial axis, the solid lines are what is constructed given only local sensing. The subset of the solid lines in $B_{\frac{r}{2}}(t)$ align with the global medial axis.

However, signs still provide directions even if there isn't a medial axis to root them to. Therefore, the algorithm proposed in this paper is meant to be used in spaces of the environment where a medial axis cannot be reliably

constructed, to fill in the gaps that [13] left and give the robot a navigation strategy invariant of sensing radius.

B. Word representation

We keep track of the robot's trajectory homotopy class in a word W=[W_p][W_c], where W_p captures the obstacles the robot already passed, and W_c contains the obstacles the robot is currently passing. Both W_p and W_c are composed of "letters" which are tuples of the form (o_i, $side$, dir, $flag$):

- $o_i \in O$ is an obstacle that intersects with $\bigcup_{t=0}^{t_i} B_r(t)$
- $side \in$ {`left, right`}: a trajectory τ_t passes an obstacle o_i on the left if it moves counter-clockwise with respect to o_i and right if it moves clockwise
- $dir \in$ {`fwd, bwd`}: A passing is labeled as "forward" (abbreviated fwd) if the obstacle was added into W as the robot was executing the forward algorithm (algorithm 1) and "backward" (abbreviated bwd) if it was during the backward algorithm (algorithm 2)
- $flag \in$ {True, False}: This flag records whether τ_t has explored both the "left" and "right" sides of obstacle o_i in the forward direction.

W_p and W_c: starting and finishing obstacle "passing"

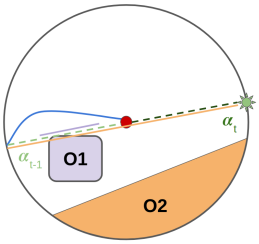


Fig. 4. An example of an obstacle that has been passed (O1, purple) and an obstacle that is currently being passed (O2, orange). The trajectory taken up to t is blue. The axis a_{t-1} is the light green dashed segment and a_t is the darker green green dashed line. The projection for O_1 is the purple segment and the projection for O_2 is orange. O_1 belongs to W_p and O_2 belongs to W_c at the end of this timestep.

Let $gs(t)$ be a waypoint the robot is travelling to from time t to time $t+1$ (waypoint placement is explained in section V-D). The robot's starting position is at $\rho(t)$ and next location $\rho(t+1)$ is located at $gs(t)$. Let *axis* a_t be the line segment from $\rho(t)$ and $gs(t)$. For obstacle o_i let $proj_{a_t}(o_i, B_r(t))$ be $o_i \bigcap B_r(t)$ projected onto a_t. From time t to $t+1$ a robot is *currently passing* object o_i if $proj_{a_t}(o_i, B_r(t)) \neq \emptyset$ and $gs(t) \in proj_{a_t}(o_i, B_r(t))$. Objects currently being passed are captured in W_c. Once this condition is no longer satisfied, the object is removed from W_c.

If $proj_{a_t}(o_i, B_r(t)) \neq \emptyset$ and $gs(t) \notin proj_{a_t}(o_i, B_r(t))$, o_i has been "passed" by τ_{t+1} and is added to W_p. An example of $O_1 \in W_p$ and $O_2 \in W_c$ at time t is shown in fig. 4

Obstacles either appear directly in W_p or move to W_p from W_c. Not all obstacles in W_c necessarily move to W_p. The size of W_p grows monotonically, but W_c may grow or shrink from timestep to timestep.

In fig. 2, the robot starting at time t_i has W: $[(O1, right, fwd, False)]$,$[(O2, right, fwd, False)$, $(O3, left, fwd, False)]$ and starting at time t_{i+1} has the word representation of $[(O1, right, fwd, False)$, $(O2, right, fwd, False)]$, $[(O3, left, fwd, False)]$

C. BACKTRACK_SIGNAL

A robot switches to *backtrack* when the robot hits a dead end and needs to revisit a previously traveled subsegment of its path. Formally, the robot backtracks when $\forall$ possible trajectories projected onto $a_{t+1}\exists$ a nonempty intersection with a_t. For example, the robot at time t_{i+1} in fig. 2 will fulfill this condition in the next time step.

D. Forward movement and backtracking

In **algorithm 1** the robot navigates using signs to dictate a general direction. If the robot hits a backtrack signal, it then backtracks in **algorithm 2** by using the word representation of the current trajectory's homotopy class and traveling a trajectory that minimizes distance traveled while guaranteeing exploration of a previously unexplored homotopy class. We elaborate on the specific steps:

GET_WAYPOINT (choosing the next $gs(t)$***):*** A robot at $\rho(t)$ navigates by first choosing a waypoint $gs(t)$ that lies on the boundary of $B_r(t)$. In the forward algorithms we choose $gs(t)$ based on the signs the robot has seen, S_t, while in the backtracking algorithm, we choose based on the word W and the homotopy class:

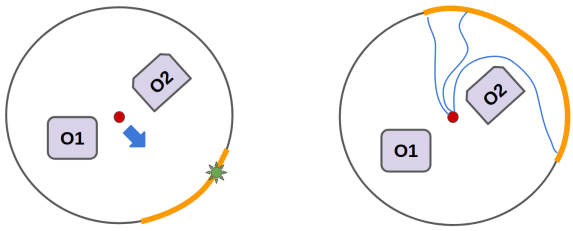


Fig. 5. An example waypoint arc ϕ in the (1) sign following scenario on the left, where the sign-provided direction is the blue arrow and (2) desired homotopy class scenario on the right, where several acceptable trajectories are shown as blue curves. The the arcs in both images are shown in orange and a potential viable $gs(t)$ for the left is shown as a green sun.

1) If $gs(t)$ is chosen based on S_t, there exists a most recent sign $s \in S_t$ that provides a heading angle a that should point in the general direction of the goal. The robot defines an arc $\phi(t)$ on $B_r(t)$ of the angle range $(a-\theta, a+\theta)$ where θ is an adjustable parameter (in our implementation we use $\theta = 15°$).

2) If $gs(t)$ is chosen based on a desired homotopy class, from the set of all endpoints with valid trajectories in the desired homotopy class, the arc $\phi(t)$ is defined by the endpoints of the trajectories that minimize the additional obstacles added into W.

$gs(t)$ is then randomly placed on this arc $\phi(t)$.

PICK_TRAJECTORY: The robot needs to generate a trajectory from $\rho(t)$ to $gs(t)$. If (1) following signage, the robot takes the shortest path from $\rho(t)$ to $gs(t)$. If (2) matching a desired homotopy class, the robot also takes as input the relevant letter in $w \in W$. It picks a shortest path trajectory in the appropriate homotopy class.

Algorithm 1: Forward algorithm

Input: ρ, g, r, $B_r(t), S_t, \tau_t$

while $g \notin B_r(t)$ **do**
  // update signs
  $S_t = \texttt{UPDATE}(S_t)$
  if *not BACKTRACK_SIGNAL* **then**
    // set waypoint gs based on S_t
    $gs = \texttt{GET\_WAYPOINT}(S_t)$
    $\tau_t' = \texttt{PICK\_TRAJECTORY}(\rho, gs, mode = \text{“}sign''\text{)}$ $\tau_t = [\tau_t, \tau_t']$ $\rho = \texttt{UPDATE}(\rho)$, $W = \texttt{UPDATE}(W)$
  else
    BACKTRACKING_ALG(ρ, g, r, S_t, W, τ_t)
  $t = t + 1$

Algorithm 2: Backtracking algorithm

Input : ρ, g, r, S_t, W, τ_t

while $g \notin B_r(t)$ **do**
  if $\exists w \in W_p$ *or* W_c *s.t.* $w.flag = False$ **then**
    W'=$[W_c(end),...,W_c(1), W_p(end),...,W_p(1)]$
    for w *in W'* **do**
      if $w.flag = False$ **then**
        // set waypoint gs based on w
        $gs = \texttt{GET\_WAYPOINT}(w)$, $\tau_t' = \texttt{PICK\_TRAJECTORY}(\rho, gs, mode = \text{“}homotopy''\text{)}, w)$, $\tau_t = [\tau_t, \tau_t']$, $\rho = \texttt{UPDATE}(\rho)$
        $W' = \texttt{UPDATE}(W)$
        if $w \in W'$ **then**
          BACKTRACK_ALG(ρ, g, r, S_t, W', τ_t)
        else
          $W = W'$
  else
    $\rho = \texttt{SAMPLING\_BASED\_METHOD}(\rho)$
    $S_t = \texttt{UPDATE}(S_t)$
    $W = \texttt{UPDATE}(W)$
  $t = t + 1$

As the sensing radius of the robot approaches 0, or the number of obstacles in the space approaches 0 this strategy reduces to a probabilistic road map (PRM) strategy.

E. Guarantees: Probabilistic completeness invariant of r

The robot's navigation policy prioritizes exploration of homotopy classes, but the method is still probabilistically complete (the probability that the planner fails to find a path, if one exists, asymptotically approaches zero).

Lemma 5.1: Assume an unreachable goal, let ρ be a robot with sensing radii $r > 0$. All homotopy classes defined by objects $\{o_1, o_2, ...o_n\}$ that appear in W_p will be explored in finite time.

Proof: Every object is definitionally included in W. W can be represented as a binary tree with each edge as a passing side of an object and each object as a node. ρ using BACKTRACK_ALG ensures that each side of the obstacle is explored at most once. BACKTRACK_ALG also ensures obstacles are explored from the bottom up of the binary tree. The tree will be fully explored because BACKTRACK_ALG runs as long as there exists an object with at least one unexplored passing side. Therefore BACKTRACK_ALG ensures every homotopy class gets explored. Each iteration of BACKTRACK_ALG will be done in finite time. As there are a finite number of obstacles, there are a finite number of homotopy classes, each of which can be explored in finite time; therefore the exploration of all homotopy classes can be completed finite time. ■

Lemma 5.2: Assume an unreachable goal, let ρ be a robot with sensing radius $r > 0$. All objects will eventually be added to W_p by ρ with probability 1 as time approaches ∞.

Proof: Let o_i be the last unpassed obstacle. Due to theorem 5.1 all obstacles in W_p (which is all other obstacles in this case) homotopy classes will be explored in finite time. If o_i is not discovered and passed during this process, when all homotopy classes represented in W_p have been explored, the algorithm uses a probabilistically complete sampling based planner and will be able to find a trajectory that passes obstacle o_i as $t \to \infty$. This extends inductively to any finite set of unpassed $O' \subseteq O$. ■

Lemma 5.3: Assume there exists a point $e \in E$, let ρ be a robot with sensing radius $r > 0$. e will have been included in $\bigcup_{i=1}^{i=n} B_r(t)$ as $n \to \infty$

Proof: e will be seen in as $n \to \infty$ for the same reason that all obstacles will eventually be seen in Lemma 5.2. ■

VI. Evaluation

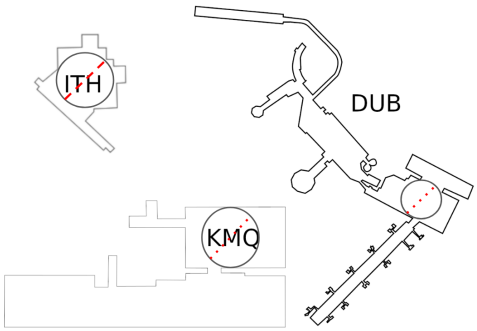


Fig. 6. Three examples of real airport environments with increasing complexity: Ithaca (ITH: Simple), Komatsu (KMQ: Medium), and Dublin airport (DUB: Complex) as used in Experiment 1. d in each airport is shown as a red dashed segment in its respective maximal inscribed circle.

In section V we prove theoretical guarantees of probabilistic completeness, but to show our approach is practical, we experimentally validate the efficiency of the robot's planning. In section VI-A we demonstrate that the strategy from section V-D can be used with the technique from section V-A in real airport floor plans and show the effect of the sensing radius on the trajectory length. To illustrate how variations in the environment can impact the robot's performance across

sensing radii, in fig. 9 we show the effects of sensing radius size on trajectory length in environments that vary in obstacle area density, holding the total number of homotopy classes constant. Then, in fig. 10 we show the effects of the sensing radius size on the trajectory length in environments that vary in total number of homotopy classes, holding the obstacle volume density constant. For each of the experiments, we normalize using the shortest path length in each environment. This allows us account for airports' varying sizes without normalizing over an online sampling based planner's trajectory, which may not be optimal.[1]

A. Experiment 1:

Our first experiment demonstrates that algorithm 1 and algorithm 2 in conjunction with the medial axis and sign following strategy from [13]— explained in section V-A— works in real airports. The experiment also shows the effects of varying sensor radius on navigation efficiency.

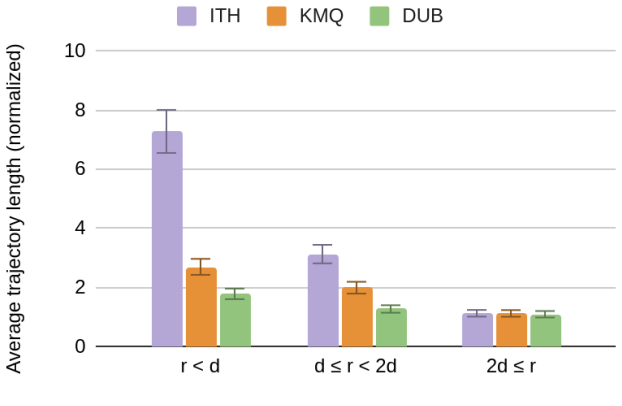
We use 20 airport maps available via Google Maps and annotated with signs from Google Streetview (with the exceptions of ITH and ORD, which were hand-gathered and hand-labeled). The labeling process was done by volunteers who found signs using Google Streetview and noted the directions they interpreted from the signs. We also include polygonal obstacles in the space that represent a subset of the obstacles in seen in the real airports.

For each airport we have one fixed goal. Let d be defined as the diameter of the largest disc that can be inscribed in E, as used in [13]. We define three categories of sensing radius: (1) $r < d$, (2) $d \leq r < 2d$, and (3) $r \geq 2d$. For each category we initialize 100 random starting points in E and a random sensing radius r that satisfies the category's assumption.

The airports can be ranked in complexity by the cardinality of the edge sets of their automatically generated medial axes from [13]. In the bar chart in fig. 7 we show the average trajectory lengths, normalized by the shortest path in the environment, of each radius category in three of the 20 airports we ran on: a simple airport (ITH) , a medium complexity airport (KMQ), and a complex airport (DUB) shown in fig. 6 from left to right. For clarity of presentation, we show only these three as representatives from the pool of 20 airports. In all 20 maps for all trials, the robot found the goal. The average normalized trajectory lengths for each group are shown in the table in fig. 7.

Observations: Under the widest sensing radius setting all airports are strictly following the strategy from [13]. In the other two sensing radius settings, the robot switches between open-space exploration and medial axis following depending on the space's geometry. The majority of simple airports are one large room, but amongst medium and complex airports there is a broad variety of geometric features. We observe that in airports with narrow passageways connecting large

[1]In all three experiments, when $r \geq 2d$, the robot is always able to construct a medial axis and thus navigates using the strategy from [13]. Since the approach is near optimal, when normalized with the shortest path as shown in the bar charts (fig. 7,fig. 9, fig. 10), they perform about the same.



All 20 airports - Normalized average trajectory			
Complexity	$r < d$	$d \leq r < 2d$	$r \geq 2d$
Simple (5)	8.09	4.26	1.01
Medium (7)	9.1	2.79	1.04
Complex (8)	8.47	3.31	1.11

Fig. 7. Bar chart shows average trajectory length normalized by shortest path compared over: ($r < d$, $d \leq r < 2d$, $r \geq 2d$). Simple airport Ithaca (ITH) is in purple, medium airport Komatsu (KMQ) in orange, and complex airport Dublin (DUB) in green, in order from left to right. The table shows average normalized trajectory lengths for all 20 airports categorized into complex, medium, and simple groups (with the number of airports in the left column).

rooms, the robot spends the bulk of its time exploring the rooms (such as KMQ in fig. 6); for airports that are long corridors with occasional offshoots two one or two large rooms (such as DUB in fig. 6), the robot spends the bulk of its time following the medial axis. This experiment shows that the robot can reach its goal in real airports and that a smaller sensing radius decreases navigation efficiency, but the impact is not linear. There is a tradeoff between the sensing radius that real sensors can pragmatically achieve and the efficiency of the robot's navigation. The numbers suggest the sweetspot for environments, invariant of their complexity, lies in the $d \leq r$ range; this insight can influence the design choices of sensors for robots operating in such spaces.

B. Experiment 2:

Our second experiment varies sensing radius and density of obstacles in the space while holding the number of total homotopy classes constant.

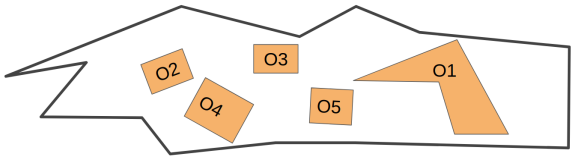


Fig. 8. Example of an artificially generated 2d polygonal environment with polygonal obstacles as used in Experiment 2 and 3.

We use 30 artificially generated polygonal environments with a variety (in shape and size) of polygonal obstacles such as the one shown in fig. 8, 10 under each of three classifications: the area of obstacles at 10% of the environment

area, 25%, and 50%. We hold the number of obstacles in the environment constant.

For each environment, the robot is given a random starting point and an unreachable goal. We have robots with sensing radii in the same ranges as experiment 1: $r < d, d \leq r < 2d, r \geq 2d$. We then have the robot navigate until it explores all homotopy classes.

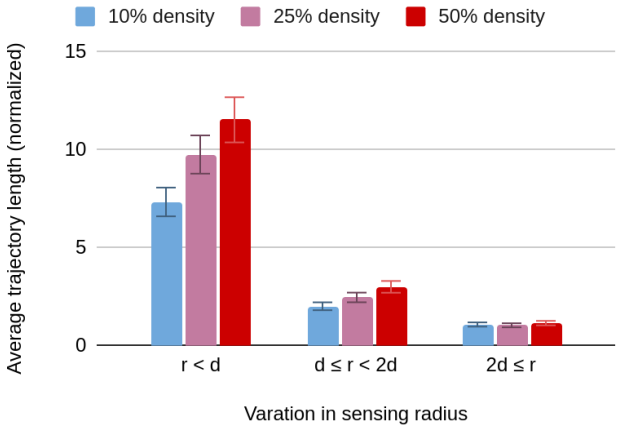


Fig. 9. This bar chart shows the average trajectory length taken to explore all homotopy classes for each classification of obstacle density in the environment and robot sensing radius. 10% density is in blue, 25% density is in mauve, and 50% density is in red, ordered from left to right.

Observations: All robots explore all homotopy classes successfully. The density of obstacles does not have significant correlation with the average trajectory length. However, the sensing radius does. When looking across environments, robots with smaller sensing radii spend more time randomly exploring, searching for obstacles to ground their homotopy class search in. However, when looking exclusively at the trajectory length of the robots during their time spent in homotopy exploration mode, normalized trajectory lengths are not significantly different across different sensing radii. This observation suggests that the efficiency of algorithm 1 and algorithm 2 is invariant of the size of the obstacles.

C. Experiment 3:

Our third experiment varies sensing radius and total number of homotopy classes in the space while holding the density of obstacle volume in the space constant.

We use 30 artificially generated polygonal environments with polygonal obstacles such as the one shown in fig. 8, 10 under each of three classifications: 5, 10, and 20 obstacles. We hold the obstacle area density constant at 25%.

For each environment, the robot is given a random starting point and an unreachable goal. We have robots with sensing radii in the same ranges as experiment 1 and 2: $r < d, d \leq r < 2d, r \geq 2d$. The robot navigates until it explores all homotopy classes.

Observations: All robots explore all homotopy classes successfully. The total number of homotopy classes and sensing radii both impact the robot's efficiency. We find that in the environments with more obstacles, the robot spends more time in homotopy exploration mode than in random exploration mode. Needing to explore more homotopy classes means the overall trajectory length is longer; the busier the space, the more options the robot needs to explore. When using algorithm 1 and algorithm 2 in practice, one should be careful about which static obstacles to consider in the word representation. For example, a real airport robot may consider a baggage claim carousel to be a legitimate obstacle, but leave a ficus plant up to real-time collision avoidance. It likely does not matter if the ficus is explored on both the left and the right. This type of information, given the correct sensing formats, may be available at run-time and could boost efficiency further.

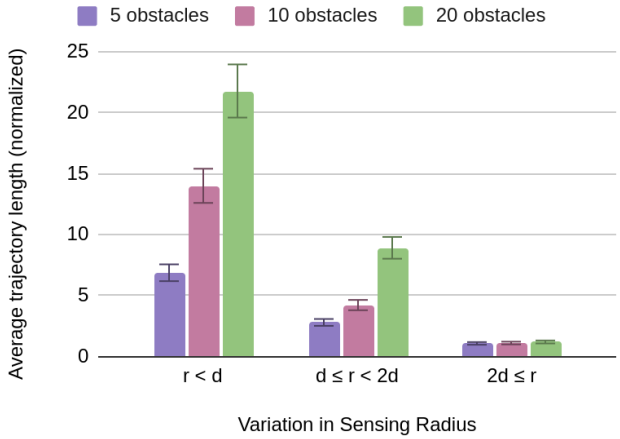


Fig. 10. This bar chart shows the average trajectory length taken to explore all homotopy classes for each classification of total number of obstacles in the environment and robot sensing radius. 5 obstacles is in purple, 10 obstacles is in mauve, 20 obstacles is in green, ordered from left to right.

VII. Future Work

This work does not explicitly handle sensing and sign interpretation. To use this technique, sensor data needs to be interpreted to create (1) A medial-axis like skeletonization (2) an interpreted sign representation. For (1) the robot needs to be able to uniquely identify static obstacles and omit dynamic obstacles when generating a medial axis representation. For (2) the robot needs to perform sign detection, homography estimation to convert the sign's arrows to an executable robot heading angle, and the grouping task to associate arrows with the appropriate destinations listed on the sign (some initial work is in [14]).

Conclusion: In this work we present a sensing-radius invariant approach for navigation of human-made spaces that uses only local sensing and human signage. While performance is best when the robot's sensing radius is greater than twice the width of the largest corridor, even when that condition is not met, we guarantee probabilistic completeness and demonstrate success in real-world spaces 100% of the time. Our experiments illustrate the effects from the varying geometry of real airports and the impact of sensing-radius on efficiency of homotopy class exploration. In this work we make progress towards the goal of a robot that can be placed in any human-made space, without costly initial setup, and be able to navigate alongside real human pedestrians.

REFERENCES

[1] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev. Search-based path planning with homotopy class constraints. In *Conference on Artificial Intelligence (AAAI)*, 2010.

[2] Yingfeng Chen, Feng Wu, Wei Shuai, and Xiaoping Chen. Robots serve humans in public places—kejia robot as a shopping assistant. *International Journal of Advanced Robotic Systems (IJARS)*, 14(3), 2017.

[3] MC Chinnaiah, Sanjay Dubey, L Vineela, K Bindu, and E Bharath Babu. An unveiling path planning algorithm with minimal sensing using embedded based robots. In *IEEE International Conference on Advances in Human Machine Interaction (HMI)*, 2016.

[4] David M Cole and Paul M Newman. Using laser range data for 3d slam in outdoor environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[5] Swarnava Dey and Arijit Mukherjee. Robotic slam: a review from fog computing and mobile edge computing perspective. In *International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services (MobiQuitous)*, 2016.

[6] Jakob Fredslund and Maja J Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE transactions on robotics and automation*, 18(5):837–846, 2002.

[7] Leonidas J Guibas, Jean-Claude Latombe, Steven M LaValle, David Lin, and Rajeev Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry & Applications*, 9:471–493, 1999.

[8] Michiel Joosse and Vanessa Evers. A guide robot at the airport: First impressions. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2017.

[9] Michiel Joosse, Manja Lohse, and Vanessa Evers. How a guide robot should behave at an airport insights based on observing passengers. *CTIT Technical Report Series*, (TR-CTIT-15–01), 2015.

[10] Takayuki Kanda, Masahiro Shiomi, Zenta Miyashita, Hiroshi Ishiguro, and Norihiro Hagita. A communication robot in a shopping mall. *IEEE Transactions on Robotics*, 26(5):897–913, 2010.

[11] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[12] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[13] Claire Liang, Ross A Knepper, and Florian T Pokorny. No map, no problem: A local sensing approach for navigation in human-made spaces using signs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[14] Claire Liang, Cheng Perng Phoo, Laasya Renganathan, Yingying Yu, Bharath Hariharan, and Hadas Kress-Gazit. Perceiving signs for navigation guidance in spaces designed for humans. In *Workshop on Closing the Academia to Real-World Gap in Service Robotics at Robotics Science and Systems (RSS)*, 2020.

[15] KN McGuire, Christophe De Wagter, Karl Tuyls, HJ Kappen, and GCHE de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35), 2019.

[16] James R Munkres. Topology, 2000.

[17] Florian T Pokorny, Danica Kragic, Lydia E Kavraki, and Ken Goldberg. High-dimensional winding-augmented motion planning with 2d topological task projections and persistent homology. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[18] Subhash Suri, Elias Vicari, and Peter Widmayer. Simple robots with minimal sensing: From local visibility to global geometry. *The International Journal of Robotics Research (IJRR)*, 27(9):1055–1067, 2008.

[19] Benjamín Tovar, Luis Guilamo, and Steven M LaValle. Gap navigation trees: Minimal representation for visibility-based tasks. In *Algorithmic Foundations of Robotics VI*, pages 425–440. Springer, 2004.

[20] Rudolph Triebel, Kai Arras, Rachid Alami, Lucas Beyer, Stefan Breuers, Raja Chatila, Mohamed Chetouani, Daniel Cremers, Vanessa Evers, Michelangelo Fiore, et al. Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and service robotics*, pages 607–622. Springer, 2016.