Topological Path Planning For Crowd Navigation

Chao Cao

CMU-RI-TR-19-18

May, 2019

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Howie Choset, Co-Chair
Matt J. Travers, Co-Chair
John M. Dolan,
Wenhao Luo

Thesis submitted in fulfillment of the requirements for the degree of Master in Robotics

Abstract

Real-time navigation in dense human environments has been a challenging problem in robotics for years. Most existing path planners fail to account for the dynamics of pedestrians because introducing time as an additional dimension in search space is computationally prohibitive. Alternatively, most local motion planners only address imminent collision avoidance and fail to offer long-term optimality. In this work, we present an approach, called Dynamic Channels, to solve this global to local quandary. Our method combines the high-level topological path planning with low-level motion planning into a complete pipeline. By formulating the path planning problem as graph-searching in the triangulation space, our planner is able to explicitly reason about the obstacle dynamics and capture the environmental change efficiently. We evaluate efficiency and performance of our approach on public pedestrian datasets and compare it to a state-of-the-art planning algorithm for dynamic obstacle avoidance.

Contents

List of Figures

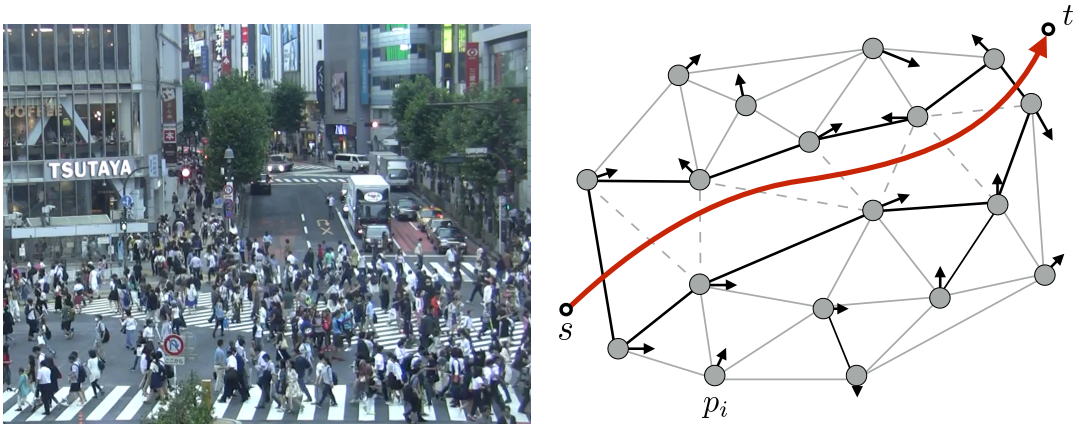
Chapter 1

Introduction

Autonomously navigating through crowded human environments is a cornerstone functionality for many mobile robot applications (e.g., service, safety, and logistic robotics). However, crowd navigation requires safe operation of the robot in close proximity to agile pedestrians: a competent crowd planner needs to efficiently generate feasible, optimal, and even socially compliant trajectories. A typical crowd navigation scenario is shown in Figure 1.1a, where the highly dense human crowd poses a huge challenge for path planning. Most existing approaches focus on either global or local optimality. For instance, sampling-based motion planners often assume a static environment because adding a time dimension exponentially increases computation. Search-based, optimization-based and geometry-based planning algorithms are subject to the same computational shortcoming. Alternatively, local planners attend to agent dynamics by seeking collision-free paths over short time horizons but neglect global optimality.

We introduce *dynamic channels*, a novel crowd navigation architecture combining global optimality and a heuristic for efficiently managing agent dynamics. Our method abstracts the environment using Delaunay Triangulation and then projects obstacle dynamics onto the triangulation. A modified A* algorithm then computes the optimal path, as abstracted in Figure 1.1b, where the optimal path is shown in red. Empirically, performance and efficiency are validated with thousands of real world pedestrian datasets. Our approach outperforms state-of-the-art methods by a significant margin for task completion while remaining real-time computable even for large numbers of dynamic obstacles. It is worth pointing out that our algorithm assumes linear motions of pedestrians during a short period of time, which can be violated when the crowd density exceeds a certain threshold. However, for crowds with moderate density as in the two datasets we evaluate on, our algorithm performs significantly better.

The rest of this report is organized as follows. Chapter 2 gives a detailed discussion on related work with a focus on motion planning in dynamic environments and crowd navigation. Chapter 3 gives a mathematical formulation of the crowd navigation problem and discusses in detail our geometry-based planning framework. In Chapter 4, the experimental evaluation is given to highlight the performance of our method. The completeness proof of our algorithm under such formulation is given in Chapter 5. Chapter 6 concludes our work and presents future work and possible extensions.



(a) (b)

Figure 1.1: (a) A crowded crosswalk in Japan, where the navigation of a robot will be challenging without an efficient planner. (b) The environment abstraction used in our framework. Pedestrians are shown as circles with arrows indicating the moving direction and speed. The "Dynamic Channel" and a safe trajectory inside are shown by bold line segments and the red curve with an arrow.

Chapter 2

Related Work

2.1 Local and Global Planning

Local planning seeks to optimize near-term objective functions; classic approaches include the Dynamic Window Approach [11], Inevitable Collision States [12], and Velocity Obstacles [10]. Variants of Velocity Obstacles [34,37,38,40] solve the multi-agent system problem for reciprocal collision avoidance by assuming that each agent adopts the same navigation strategy (e.g., each agent passes to the right). However, human behavior is probabilistic, and so assuming fixed actions can lead to unsafe robot behaviors. However, even with guaranteed collision-free local actions, local planning is insufficient, since such trajectories may not guarantee, e.g., the shortest traveling distance.

Global planning searches for feasible paths connecting initial and goal states of the robot. Most traditional sampling-based algorithms including Rapidly-exploring Random Tree (RRT) [24] and Probabilistic Roadmap methods (PRM) [19, 20] assume the environment to be static when planning; replanning is employed to account for dynamics. To reduce unnecessary computation [43] proposed a dynamic RRT variant by reusing branches from previous search trees. In [7] a replanning scheme based on RRT* [18] dealt with unknown dynamic obstacles blocking the robot's path.

Many global planners decouple static and dynamic obstacles. An intuitive strategy relies on a local reactive collision avoidance system to deal with local disturbances. Separating static and dynamic obstacles can also be achieved by hierarchical planning and the combination of potential fields: [16,39] decouples high- and low-level planning with a two-level search. Globally, a roadmap is built upon static obstacles while locally a collision-free trajectory is obtained. In [35], a dynamic pedestrian potential field selects the traversal and safety optimal trajectory. The same problem also exists in other planning methods. For Elastic Band [32], Timed Elastic Band [33] and spline-based planners [25], the initial guess determines path partition, which is based on the static assumption. Ultimately, treating static and dynamic obstacles separately has the advantage of being efficient and probabilistically complete. However, sub-optimal paths can result because obstacle dynamics affects global optimality.

An important class of planning methods leverages computational geometry to gain efficiency; Voronoi diagrams and Delaunay Triangulation have long been used for environment abstraction. With Voronoi diagrams, the decomposition of free space based on Voronoi cells efficiently generates homotopically distinct paths [1,6,13,23]. Delaunay Triangulation

has also been used in graphics to solve the virtual character navigation [2, 17]. These data structures have important complexity properties in 2D planar environments: compared to grid-based maps, computational geometric approaches search a much smaller graph while faithfully capturing topological properties like connectivity. In [9] Triangulation A* (TA*) and Triangulation Reduction A* (TRA*) are introduced for pathfinding in the triangulation space. Similarly, [42] presents a path planning approach in Constrained Delaunay Triangulation (CDT) to account for static polygon obstacles. Finally [4] uses dynamic Delaunay triangulation (DDT) to solve the problem of navigation in the absence of a pre-specified goal position.

However, these algorithms do not reason about obstacle dynamics, and naively adding a time dimension results in exponential computation. Our approach introduces a novel approach to incorporate obstacle dynamics in the global planning step and extends the pathfinding algorithm to efficiently analyze environmental evolution.

2.2 Human Aware Navigation

Human-aware navigation [21] focuses on socially acceptable robot behaviors rather than explicitly solving the collision avoidance problem. In [36] cooperative collision avoidance is leveraged to develop a tractable interacting Gaussian Process (IGP) model. A socially-inspired crowd navigation approach is developed in [30] where the robot follows pedestrians moving towards its goal. However, this method might fail in situations where the major flow of pedestrians is not in the robot's goal direction. Other work leverages learning to model and replicate socially compliant behaviors for crowd navigation [5, 14, 22, 28, 29]

In this work, we do not explicitly address socially acceptable navigation. However, our framework can be extended to incorporate social norms. For example, in the graph construction step (described below), the interaction between people can be encoded in the graph if it is obtainable (one person taking pictures for the other, a group of friends walking together), so that edges between pedestrians can be denoted as non-crossable for the robot. Moreover, the path generated by our planner naturally captures some social norms. For example, when passing a pedestrian, our planner will favor passing from behind rather than in front of the pedestrian, because the former leads to a shorter traveling distance.

Chapter 3

Methodology

Let the crowd positions at time τ be denoted as $\mathcal{P}_\tau = \{p_i^\tau \mid p_i^\tau \in \mathbb{R}^2, i = 1, \ldots, n_t\}$, where n_τ is the number of pedestrians. Let $\mathcal{V}_\tau = \{v_i^\tau \mid v_i^\tau \in \mathbb{R}^2, i = 1, \ldots, n_t\}$ be pedestrian velocities at τ. Let the robot starting point be $s \in \mathbb{R}^2$ and goal $t \in \mathbb{R}^2$. We seek the collision-free trajectory that is optimal with respect to travel time and distance.

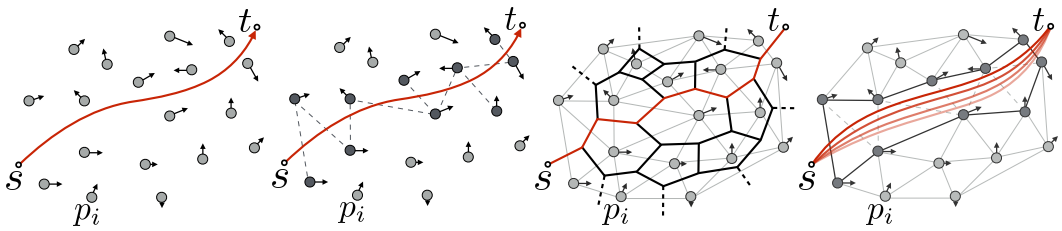


Figure 3.1: (a) A path from s to t planned through the crowd. Pedestrians are shown as grey circles with arrows denoting velocities. (b) The path sequentially passes through pairs of pedestrians shown as darker circles connected by dashed line segments. (c) The dual graph $\mathcal{T}^*$ of the triangulation of $\mathcal{P}$. A path connecting s and t is shown in red. (d) A "channel" (bold line segments) and the corresponding homotopy class of paths (red curves) bounded within.

3.1 Delaunay Triangulation

Our first objective is to plan a path through the obstacles $\mathcal{P}_\tau$ at the topological level. This is achieved by Delaunay triangulation on $\mathcal{P}_\tau$, which produces the graph $\mathcal{T}_\tau$ (light gray graph in Fig. 3.1c; p_i^τ are vertices). The dual graph $\mathcal{T}_\tau^*$ (black graph in Fig. 3.1c) has one vertex for each face of $\mathcal{T}_\tau$ and one edge for face pairs separated by an edge in $\mathcal{T}_\tau$. If we define homotopic equivalence as paths that can be continuously deformed into each other without passing through vertices p_i^τ, then $\mathbb{R}^2$-paths and $\mathcal{T}_\tau^*$-graphs are equivalent (Fig 3.1c, 3.1d):

Theorem 3.1.1. *Any path through $\mathcal{P}_\tau$ in $\mathbb{R}^2$ uniquely determines a path in $\mathcal{T}_\tau^*$.*

Theorem 3.1.2. *A path on $\mathcal{T}_\tau^*$ without repeating vertices corresponds to one homotopy class of paths in $\mathbb{R}^2$*

For brevity, we prove these theorems in Chapter 5. *Thus, instead of optimizing paths in $\mathbb{R}^2$ we can equivalently search the graph $\mathcal{T}_\tau^*$. This graph search is computationally efficient (Section 4.2).* An important advantage of searching $\mathcal{T}_\tau^*$ is that we can reason about pedestrian pair interactions, instead of individually checking collisions. Consider a path passing through a group of people as shown in Fig. 3.1a, and the directional curve showing the robot path. By projecting pedestrian velocities onto triangulation edges, triangulation time evolution is achieved. In this manner, pedestrian dynamics are analyzed as a network.

3.2 Dynamic Channel

A path on $\mathcal{T}_\tau^*$ (Fig. 3.1c) is equivalent to a triangulated simple polygon called a "channel" [17] (Fig. 3.1d). We extend this concept to a "dynamic channel." This dynamic channel is precisely the time evolution of our triangulation.

Pedestrian dynamics cause the dynamic channel to deform constantly. Of most concern is "gate" change (light gray dashed lines in Fig. 3.1d), the distance between pedestrian i and j at time τ:

$$D^{ij}(\tau) = ||p_\tau^i - p_\tau^j|| \tag{3.1}$$

where $||.||$ denotes the Euclidean distance. Let $D_{thresh} = 2(r_{obs} + S_{safe})$ be a threshold for the width of a gate considered feasible for the robot to pass, where r_{obs} is the radius of a pedestrian expanded by the radius of the robot, and S_{safe} is the minimum safe distance between the robot and pedestrians. Solving for

$$D^{ij}(\tau) \geq D_{thresh} \tag{3.2}$$

we obtain the time intervals $T_{feasible} = [\tau_1, \tau_2] \cup \cdots \cup [\tau_{m-1}, \tau_m]$ when the gate is wide enough for the robot to cross. Given the Estimated Time of Arrival (ETA) for the robot to reach the gate τ_{ETA} (detailed in the next section), safe passage is guaranteed when $\tau_{ETA} \in T_{feasible}$. This condition must hold for all gates to guarantee safe passage.

We assume that pedestrian behaviors can be modeled linearly. Thus the trajectory (starting from the current frame $\tau = 0$) of pedestrian i can be described by:

$$p^i(t) = p_0^i + v^i \tau \tag{3.3}$$

where p_0 and v^i are the current position and velocity. $D^{ij}(\tau)$ is then a quadratic function with respect to τ and the graph of $D^{ij}(\tau)$ is a parabola opening upward as in Fig. 3.2b. Solving for Equation 3.2 gives zero, one or two solutions. When there is no solution, the minimum distance $D_{min}^{ij}(\tau)$ between pedestrians i and j will never be too close for the robot to cross. When there is only one solution, a critical moment exists when the robot can be just safe enough to cross. When there are two solutions, denoting t_1 and t_2 with $t_1 < t_2$, it is required that $t_{ETA} \in ([-\infty, t_1] \cup [t_2, +\infty]) \cap [0, +\infty]$ for safe passage as shown in Fig. 3.2b.

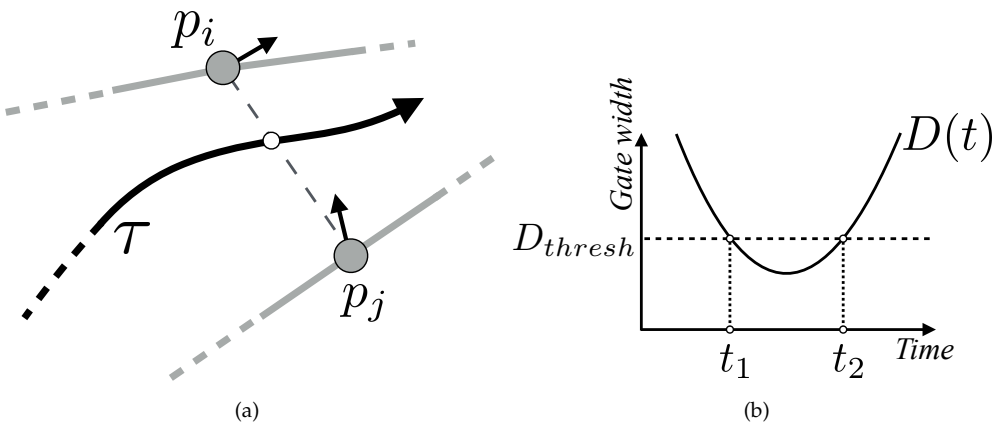
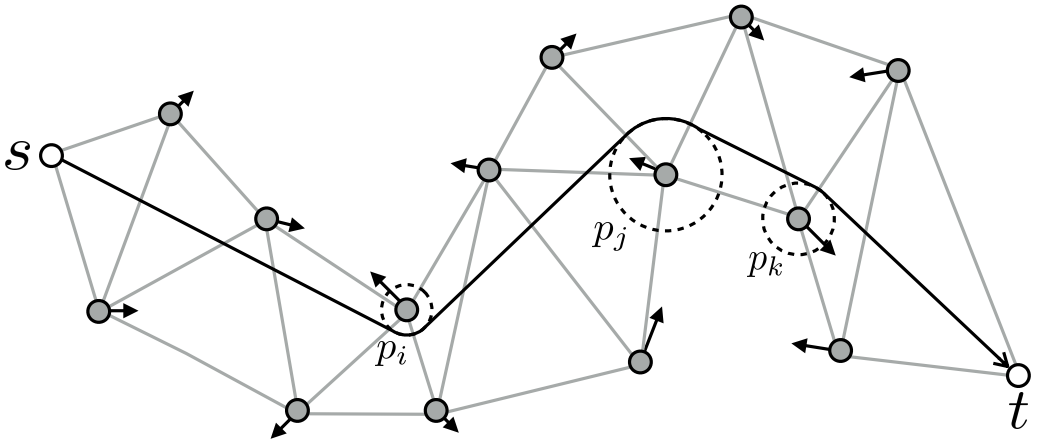


Figure 3.2: (a) A gate formed by pedestrians p_i and p_j. A path τ is planned through the gate. (b) The distance between two pedestrians changes with time as a parabola. t_1 and t_2 show the moments when the gate is wide enough (above the threshold D_{thresh}) for safe passage.

3.3 Timed A* Search

The A* algorithm has been used for searching for shortest paths in the triangulation space, which is referred to as Triangulation A* (TA*) [9, 17]. For TA*, the states in the search are nodes on the dual-graph $\mathcal{T}_\tau$ representing the triangles. Note that during the search, the actual path length through the sequence of triangles depends on the node placement, which can only be determined by methods like the "Funnel Algorithm" (section 3.4) after the goal point is reached. Thus care needs to be taken in computing the cost incurred (the *g*-value) and the heuristic (the *h*-value) used in TA* to search for the shortest path. [42] proposes the "target attractive principle" to determine the placements of nodes when building the dual graph on constrained Delaunay Triangulation. Here we adopt a similar rule-based method as in [42] to determine the graph nodes. Within a triangle, a graph node is placed relatively closer to the goal point and further away from the constrained/non-crossable edges, following the "target attractive principle". The detailed discussion on the node placements can be found in [42]. Note that in our work, the non-crossable constraints are introduced by edges that will lead to potential collision rather than static obstacles.

Here, we use a modified A* algorithm to incorporate the robot and pedestrian dynamics in the search of the shortest path. Within the triangulation framework, we focus on the interaction between connected pedestrian pairs. Thus the dynamics of the robot and pedestrians can be studied by projecting them along the edges in the triangulation. In particular, we compute the g and h-value with a lookahead time period τ. While searching, the velocity of a node representing a triangle can be determined by interpolating the velocities of three endpoints. Assuming the velocity of a node remains constant during τ, we can then compute $g(\tau)$ and $h(\tau)$ using the position of the node after moving for time τ, which give the predicted cost incurred and heuristic. To determine a proper τ for each node, we compute the Estimated Time of Arrival (ETA) τ_{ETA} of the robot arriving at that node to obtain $g(\tau_{ETA})$ and $h(\tau_{ETA})$. Similar to the estimate of g-value in a regular A* algorithm, we track path history while searching. For each node, the current path to the node is time-parameterized by taking into account the kinematics and dynamics of the robot and curvature of the path. Note that due to the approximated node placements at searching



(a)

Figure 3.3: Within a channel the shortest path is determined by our *funnel algorithm*. Clearance radius is determined by whether the pedestrian is "threatening" or "enlarging" the channel. Larger radius is assigned for the former (p_j) while smaller radius is assigned to the latter (p_i, p_k).

time as in [42] and the linear model assumption, τ_{ETA} can only be approximated accurately in a finite time horizon. Due to the time-variant $g(\tau)$ and $h(\tau)$ value, our planner will seek a candidate shortest path in the long term by taking into account the future evolvement of the obstacles, while a normal A* search only considers the shortest path in the current time frame assuming obstacles are static.

3.4 Path Optimization

The output from our Timed A* algorithm is a path on $\mathcal{T}_\tau^*$ consisting of a series of triangles that form a channel. We find the shortest path in the channel that ensures sufficient clearance. The *Funnel algorithm* [3], [26], [15] can be used for computing a shortest path within a simple triangulated polygon connecting the start and goal points. To account for non-zero-radius obstacles, [9] proposed a modified *funnel algorithm* by inserting conceptual circles at each vertex of the channel. The resulting path is composed of alternating straight-line segments and arcs. In our work, we further extend this method to account for the pedestrian dynamics. Given a pedestrian p_i^τ, we project the velocity v_i^τ onto the vector $e_{ji} = p_j - p_i$, where p_j is the other end of the edge in the channel. If the projection $v_i' > 0$, the path will pass by p_i^τ in the front, potentially blocking p_i^τ's moving direction. In this case, we assign a circle with the radius $r = k||v_i'||$ around p_i^τ to create clearance, where k is a tunable parameter. Similarly, if $v_i' < 0$, the path will pass from behind. Figure 3.3a provides an illustration, where pedestrians p_i, p_j and p_k are assigned different-sized circles based on their velocities.

Most of the collision avoidance effort described above is implicitly done by enforcing timely traversal. However, when obstacles move faster than the robot or the linear motion assumption of the pedestrians is violated in crowds with high density, simply following the path cannot guarantee collision-free navigation. In such cases, a low-level robot-specific controller is needed to perform maneuvers for the robot to avoid collision. In a

Delaunay Triangulation, the circumcircle of any triangle will not include other points inside. Thanks to this property, a collision-free circular area can be identified in the robot's vicinity, which can be used for local maneuvers when needed. Furthermore, if no paths are found by the Timed A* algorithm, other behavioral strategies can be employed to utilizes the local collision-free area. For example, the robot can follow the people walking in the goal direction while maintaining a proper clearance to others, as in [30].

Chapter 4

Experimental Evaluation

We demonstrate the advantage of our planner by testing on a large number of navigation tasks in simulated environments based on public human-trajectory datasets. We compare success rate, efficiency of traversal, and time complexity of our algorithm against two other planning methods. Finally, we empirically study the effect of observability on our algorithm.

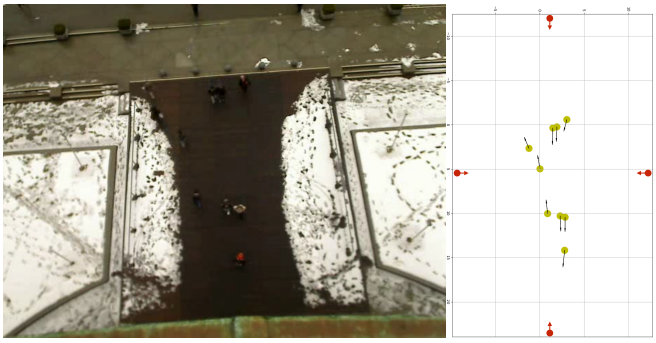
4.1 Experiment Settings

4.1.1 Datasets

For experiments, we implemented our planning framework to control a car-like robot navigating in simulated environments. In particular, we use the pure-pursuit algorithm [8] to track a path output from our planner. For benchmarking, in our simulation we replay the human-trajectories from two public datasets: ETH [31] and UCY [27]. There are five subsets in total recorded in different scenarios, denoted as eth_hotel, eth_univ, ucy_univ, ucy_zara01 and ucy_zara02.

For each dataset, the trajectory data are first interpolated and then re-projected back to the world coordinates using the homography matrices provided. We define a rectangular region as the workspace which is bounded by extreme values of the pedestrian positions of each dataset. For the navigation task, the robot starts at four mid-points of the workspace boundaries and moves to the antipodal goal position, as shown in fig 4.1. In the experiments, only pedestrians are considered to be obstacles and all free space is assumed traversable for the robot. Fig. 4.1b shows an example testing configuration, where pedestrians are shown as yellow circles with black arrows indicating velocity. The starting positions are shown as red circles with arrows. The velocities of pedestrians are approximated by the position differences between two consecutive frames divided by the sampling period.

Trials begin at a sequence of starting times at intervals of $3s$, which results in 1520, 1148, 184, 148, and 120 trials for each dataset respectively. All experiments were run on a 2.6 GHz CPU/15.5 GB memory computer.



(a) (b)

Figure 4.1: Experiment Settings. (a) shows a frame from the ETH dataset. (b) shows the pedestrian positions extracted from the frame. Red circles with arrows show the starting positions of the robot in the experiments.

4.1.2 Generalized Velocity Obstacle Planner

For comparison, we implemented a planning algorithm based on the Generalized Velocity Obstacle (GVO, [41]) to control the same car-like robot. The algorithm first randomly samples a control input u from a feasible set. Then a trajectory is simulated within a time horizon for the input, which takes into account the robot dynamics. The trajectory is then checked for collision with GVO prediction on each moving agent. Finally, among all collision-free control inputs, the one closest to the *preferred* control u^* is chosen. As suggested in the paper, we choose the *preferred* control to be the one driving the robot directly to the goal as if there were no obstacles. However, we do not explicitly solve the optimization problem for t^* that results in minimum distance between the robot trajectory and obstacles. Instead, we discretize the trajectory and check collision at 10Hz. Only in this manner can we run the algorithm efficiently enough for the benchmarking. The other parameters used were $time\_horizon = 3.5s$, $wheelbase = 1m$ and $sampling\_per\_timestep = 40$.

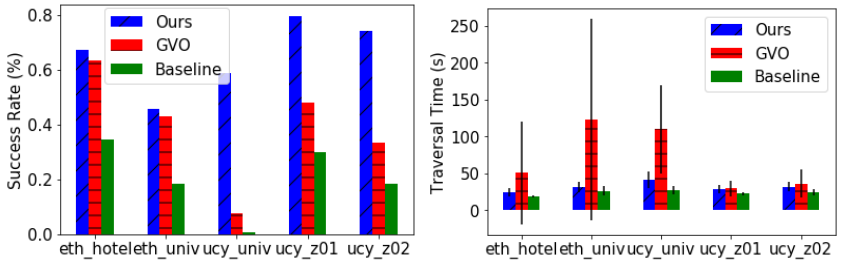
4.1.3 "Wait-and-go" Planner

We designed the simplest possible crowd navigation strategy, "wait-and-go", as a baseline: the robot drives in a straight-line towards the goal. When the robot comes too close to pedestrians or there is a potential collision determined by Velocity Obstacle, the robot stops (wait) and resumes moving (go) when possible.

4.2 Comparing success rate, efficiency of traversal and time complexity

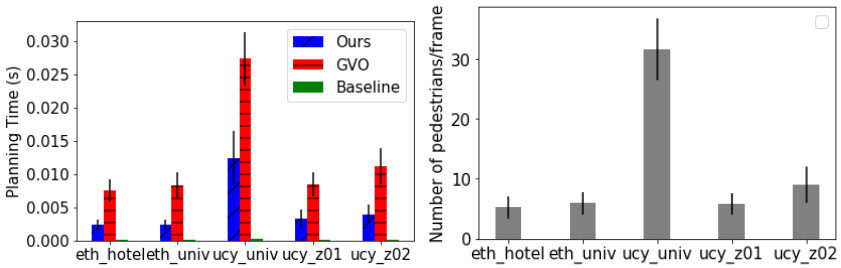
We run each trial until the robot reaches the goal position. In the case when all the frames have run out before the robot reaches the goal position, we replay the sequence from the beginning for a fixed number of times. If the distance between the robot and any pedestrian is less than $1m$ during the run, a collision is reported and the trial fails. Otherwise, if there is no collision reported and the robot reaches the goal position before all the frames run out, the trial is marked successful. We also calculated running time to compare the traversal

efficiency of the three approaches. Note that when a trial fails, the timer will not stop until the trial is finished. The robot has a speed limit of $1.2m/s$, which is slightly slower than the average human walking speed.



(a) Success Rate (b) Traveling Time

Figure 4.2: Experiment Results. Comparison of: (a) the success rate of navigation tasks in each dataset. (b) The average traveling time of the robot in each dataset. Vertical line segments show the variance by a standard deviation.



(a) Time Complexity (b) Number of Pedestrians

Figure 4.3: Experiment Results. Comparison of: (a) the average planning time of a frame for each dataset. (b) The average number of pedestrians in each dataset. Vertical line segments show the variance by a standard deviation.

4.2.1 Success Rate

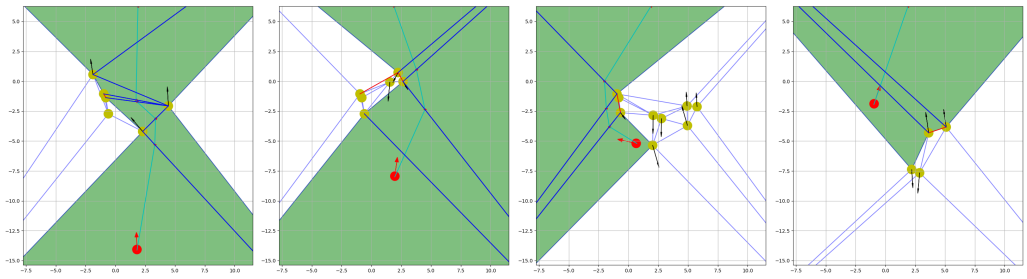
From the result shown in Fig. 4.2a, our planner has the highest success rates in all five datasets, while GVO ranks second and the baseline planner performs last. For the challenging dataset *ucy_univ*, where the crowd density reaches 30 pedestrians per frame, our planner significantly outperforms the baselines. Anecdotally, most of the failure cases of our algorithm are due to sudden pedestrian appearance and nonlinear pedestrian movement, while the causes of failure of GVO and wait-and-go are mostly due to being trapped in sub-optimal situation (e.g. surrounded by a group of pedestrians).

4.2.2 Traveling time

For traversal time, as shown in Fig. 4.2b, our planner is comparable to the "wait-and-go" baseline and is more efficient than GVO, where GVO exhibits much longer times with much larger variance. This is because GVO tends to passively dodge approaching pedestrians (resulting in more evasive strategies), while our planner actively looks for open spaces and a long-term shortest route.

4.2.3 Computational Complexity

For computational complexity (4.3a), both our approach and GVO sees an increase in computation as more pedestrians enter the scene. However, our method is significantly more time-efficient than GVO.



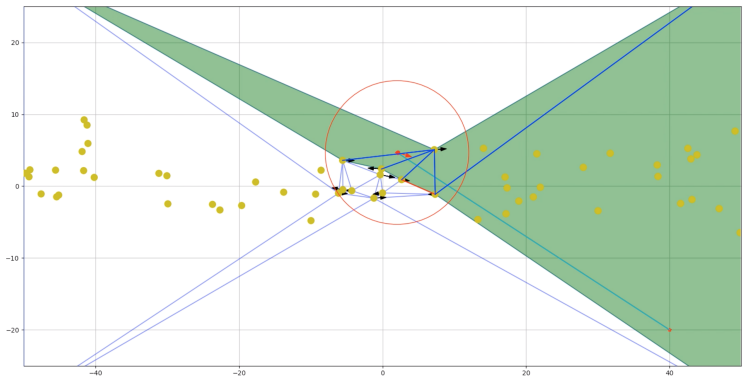
(a) $t = 2s$ (b) $t = 7.25s$ (c) $t = 11.25s$ (d) $t = 15s$

Figure 4.4: The path found by our planner in the *eth_univ* dataset. Green areas show dynamic channel. Cyan line shows trajectory. Blue line segments show valid gates to cross; red ones show unsafe gates. The robot (red filled circle) attempts to navigate from bottom to top. Note that we place four static "virtual pedestrians" at the four corners of the workspace, allowing the robot to find the paths that go around the "actual" crowd if it is necessary.

Qualitatively, Fig. 4.4 shows a sequence of the robot navigating in the eth_hotel dataset. At the beginning, the robot plans to pass by the pedestrians on the right. Between $t = 7.25$ and $t = 11.25$, there suddenly appears a group of people on the right side. Our planner is able to adapt to this and steer the robot to the left for a safe-pass.

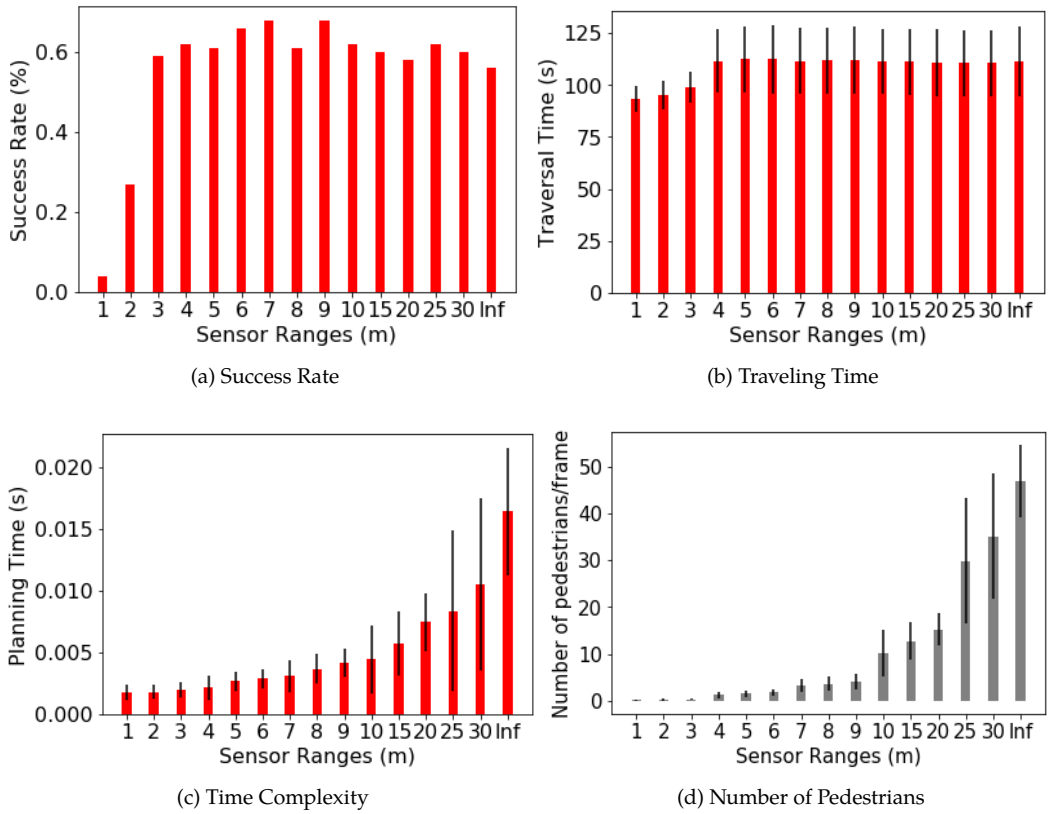
4.3 Observability Effects

We also study observability effects by testing our planner on a larger simulated scene with limited sensor ranges. The simulated pedestrians are generated as follows. For every $5s$, 2 pedestrians are spawned from the two sides of the workspace. Each pedestrian is assigned with a random goal position at the other side and a random linear velocity towards the goal. While traveling to the goal the pedestrians will randomly change their goal and speeds every $3s$. The observation range of the robot is tested from $1m$ to $30m$, with the interval being $1m$ when under $10m$ and $5m$ when above $10m$. We also run trials without sensor range limit for comparison. Beyond the sensing range, only the goal position is known, so the robot has no information about pedestrians outside the field of view. In each trial, the robot navigates from the lower right corner to the upper left corner or reversely. For each sensing range, 100 trials are performed. Fig. 4.5 shows an example scene when the robot navigates in the synthetic crowd with the sensing range limited to $10m$.



(a)

Figure 4.5: Path found by our planner in a synthetic dataset with a limited sensor range, denoted by the red circle. Pedestrians outside the sensor range are not included in the triangulation and thus not observed by the planner.



(a) Success Rate

(b) Traveling Time

(c) Time Complexity

(d) Number of Pedestrians

Figure 4.6: Experiment Results on the effect of observability. Comparison of: (a) average navigation success rates, (b) average traveling time, (c) average planning time and (d) average number of pedestrians seen by the planner with different sensing ranges. Vertical line segments show standard deviation.

Results are presented in Fig. 4.6. In Fig. 4.6a and Fig. 4.6b, we observe that the sensor range does not make a big difference for the success rate and traveling time beyond $3m$. Medium ranges (3 to 9m) are only slightly better than long ranges (10 to without limit). This is because the longer the channel, the less accurate the τ_{ETA} will be, and thus the guarantee for safe-passage for later portions of the channel is less likely to be valid. A practical solution to this problem is to set a time horizon for the planner. Whenever the estimated τ_{ETA} is longer than the horizon, we release the safety check. For the time complexity in Fig. 4.6c, we see that the time needed for a planning cycle increases rapidly with the increasing number of pedestrians per frame, as in Fig. 4.6d. However, our algorithm is still efficient in such crowded environments. It can be seen from the figure that for pedestrian number over 50, our method re-plans at 50Hz and with fewer than 30 pedestrians our method re-plans at 100Hz. Note that with the observability limit, only a subset of the crowd is considered by our algorithm, which no longer guarantees the topological completeness with respect to all the pedestrians. However, our above experiment results show that this does not become a tradeoff between the success rate and computational complexity.

Chapter 5

Proof of Topological Completeness

We present a detailed discussion on the completeness of pathfinding in triangulation. Let the crowd positions at time τ be denoted as a finite point set $\mathcal{P}_\tau = \{p_i^\tau \mid p_i^\tau \in \mathbb{R}^2, i = 1, \ldots, n_t\}$, where n_τ is the number of pedestrians, the robot starting point be $s \in \mathbb{R}^2$, and the robot ending point be $t \in \mathbb{R}^2$ $(t \neq s)$. The task is to plan homotopically distinct paths through the crowd. Without loss of generality, we assume that s and t are outside of the convex hull of $\mathcal{P}_\tau$. In the context of path planning for crowd navigation, which seeks the shortest path, we restrict the following discussion to paths without loops/duplicate vertices only. Here we provide the following important definitions.

Definition 1 (Path). *A path is a continuous mapping* $\mathbf{f} : [0, 1] \to \mathbb{R}^2$ *with* $\mathbf{f}(0) = s$ *and* $\mathbf{f}(1) = t$.

Definition 2 (Path homotopy). *Two paths* $\mathbf{f}_1$ *and* $\mathbf{f}_2$ *are said to be path-homotpic if there exists a continuous map* $F : [0, 1] \times [0, 1] \to \mathbb{R}^2$ *such that* $F(i, 0) = \mathbf{f}_1$ *and* $F(i, 1) = \mathbf{f}_2$, $F(0, j) = s$ *and* $F(1, j) = t, \forall i, j \in [0, 1]$.

Lemma 5.0.1. *A path can be approximated arbitrarily well by a bivariate polynomial parameterized as* $f(x, y) = 0$.

Proof. This directly follows from the Weierstrass approximation theorem. □

In the following discussion, we use $f(x, y) = 0$ to denote the polynomial and the path approximated by the polynomial interchangeably. Informally, in the context of motion planning among obstacles $\mathcal{P}_\tau$, we say $\mathbf{f}_1$ and $\mathbf{f}_2$ are path-homotopic with respect to $\mathcal{P}_\tau$ if one can be continuously deformed into the other without intersecting any points in $\mathcal{P}_\tau$. Note that a path $f(x, y) = 0$ intersects with points from $\mathcal{P}_\tau$ when $f(x_p, y_p) = 0, (x_p, y_p) \in \mathcal{P}_\tau$

Lemma 5.0.2. *A path* $f(x, y) = 0$ *not intersecting points from* $\mathcal{P}_\tau$ *partitions* $\mathcal{P}_\tau$ *into two disjoint sets* $(\mathcal{P}_\tau^+, \mathcal{P}_\tau^-)$, *where* $\mathcal{P}_\tau^+ = \{(x_p, y_p) \in \mathcal{P}_\tau | f(x_p, y_p) > 0\}$ *and* $\mathcal{P}_\tau^- = \{(x_p, y_p) \in \mathcal{P}_\tau | f(x_p, y_p) < 0\}$

Proof. A polynomial parameterized by $f(x, y) = 0$ partitions the $\mathbb{R}^2$ space into three disjoint open sets $U_1 = \{(x, y) \in \mathbb{R}^2 | f(x, y) = 0\}$, $U_2 = \{(x, y) \in \mathbb{R}^2 | f(x, y) > 0\}$ and $U_3 = \{(x, y) \in \mathbb{R}^2 | f(x, y) < 0\}$. With $\mathcal{P}_\tau \cap U_3 = \emptyset$, we have $\mathcal{P}_\tau^+ = \mathcal{P}_\tau \cap U_1$, $\mathcal{P}_\tau^- = \mathcal{P}_\tau^- \cap U_2$ and $\mathcal{P}_\tau = \mathcal{P}_\tau^+ \cup \mathcal{P}_\tau^-$. □

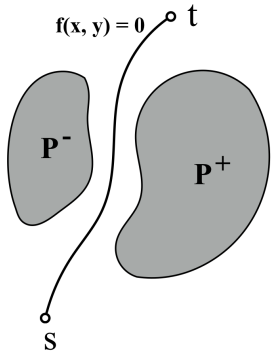


Figure 5.1: Illustration of a partition of $\mathcal{P}_\tau$ by $f(x, y) = 0$.

Proposition 5.0.3. *Given two paths $f_1(x, y) = 0$ and $f_2(x, y) = 0$ partitioning $\mathcal{P}_\tau$ into (B^+, B^-) and (C^+, C^-) respectively, (a) if f_1 and f_2 are homotopic, then $B^+ = C^+$ and $B^- = C^-$. (b) if $B^+ = C^+$ and $B^- = C^-$, then f_1 and f_2 are homotopic.*

Proof. Proof of (a): Prove by contradiction. Assume that $B^+ \neq C^+$ and $B^- \neq C^-$, there exists a point $p = (x_p, y_p)$ such that $p \in B^+$ and $p \notin C^+$, which implies $p \in B^+$ and $p \in C^-$. Therefore, $f_1(x_p, y_p) > 0$ and $f_2(x_p, y_p) < 0$. Since f_1 and f_2 are homotopic, let H be a continuous map from f_1 to f_2, $H : \mathbb{R}^2 \times [0, 1] \to \mathbb{R}$, then

$$\begin{aligned} H(x, y, 0) &= f_1(x, y) \\ H(x, y, 1) &= f_2(x, y) \end{aligned} \tag{5.1}$$

for all x, y in the domain of f_1 and f_2. Then at $x = x_p$ and $y = y_p$,

$$\begin{aligned} H(x_p, y_p, 0) &= f_1(x_p, y_p) > 0 \\ H(x_p, y_p, 1) &= f_2(x_p, y_p) < 0 \end{aligned} \tag{5.2}$$

Because H is continuous, by the intermediate value theorem, there exists some $T \in (0, 1)$, such that $H(x_p, y_p, T) = 0$. Thus contradiction occurs.

Proof of (b): We will prove the contrapositive. Consider two paths $\mathbf{f}_1$ and $\mathbf{f}_2$ that are not homotopic, then one cannot continuously deform into the other without intersecting $\mathcal{P}_\tau$. It follows that for any continuous map H from f_1 to f_2, there exists a point $p = (x_p, y_p) \in \mathcal{P}_\tau$ and $T \in (0, 1)$ such that

$$H(x_p, y_p, T) = 0 \tag{5.3}$$

and

$$H(x_p, y_p, 0)H(x_p, y_p, 1) < 0 \tag{5.4}$$

Otherwise, there exists $H(x, y, t) > 0$ or $H(x, y, t) < 0$, $\forall t \in (0, 1)$, which means f_1 can continuously deform to f_2 without intersecting p. Therefore, $f_1(x_p, y_p)f_2(x_p, y_p) < 0$. Let $f_1(x_p, y_p) > 0$, then $f_2(x_p, y_p) < 0$, which implies $p \in B^+$ and $p \in C^- \Rightarrow p \notin C^+$. Therefore $B^+ \neq C^+$ and $B^- \neq C^-$, which also holds when $f_1(x_p, y_p) < 0$ and $f_2(x_p, y_p) > 0$.

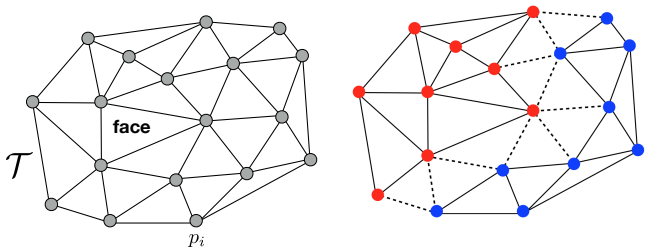
□

Definition 3 (Triangulation $\mathcal{T}$ of $\mathcal{P}_\tau$)**.** *A triangulated graph $\mathcal{T} = (V, E)$, where $V = \mathcal{P}_\tau$ and E gives a maximal set of non-crossing edges between points of $\mathcal{P}_\tau$*

Fig. 5.2a shows an example of the triangulation $\mathcal{T} = (V, E)$ of $\mathcal{P}_\tau$, where V and E are shown as grey filled circles and line segments, respectively.

Definition 4 (A cut/cut-set of $\mathcal{T}$)**.** *A cut of $\mathcal{T} = (V, E)$ is a partition of V into two disjoint subsets. A cut-set is the set of edges that have one endpoint in each subset.*

Fig. 5.2b gives an example of cut of $\mathcal{T}$. Vertices of $\mathcal{T}$ are partitioned into two disjoint subsets distinguished by red and blue circles. Edges in the cut-set are shown by the dashed line segments.



(a) (b)

Proposition 5.0.4. *A cut of $\mathcal{T}$ uniquely determines a homotopy class of a path through $\mathcal{P}_\tau$.*

Proof. Let a cut of $\mathcal{T}$ partition $\mathcal{P}_\tau$ into two disjoint subsets A and B. Let $\mathcal{P}_\tau^+ = A$ and $\mathcal{P}_\tau^- = B$, according to Lemma 5.0.2, we can find a curve parameterized by $f(x, y) = 0$ that gives the partition $\mathcal{P}_\tau = (\mathcal{P}_\tau^+, \mathcal{P}_\tau^-)$. Because of proposition 5.0.3, the partition is equivalent to a homotopy class of paths. Therefore, a cut of $\mathcal{T}$ uniquely determines a homotopy class. □

Definition 5 (Face/Outer face of $\mathcal{T}$)**.** *A face of $\mathcal{T}$ is the region bounded by edges of a triangle in $\mathcal{T}$, as illustrated in Fig. 5.2a. The (outer face) is defined as the "exterior" region outside the convex hull of $\mathcal{P}_\tau$, of which the boundary consists of edges from $\mathcal{T}$ and forms a Jordan Curve.*

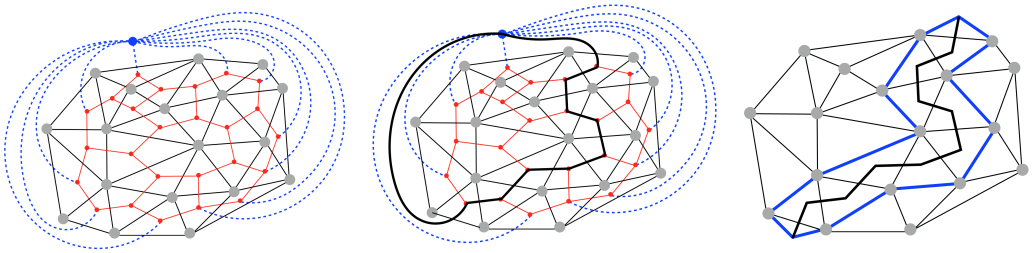
Definition 6 (Dual graph $\mathcal{T}^*$ on $\mathcal{T}$)**.** *The dual graph $\mathcal{T}^* = (V^*, E^*)$ has a vertex for each face of $\mathcal{T}$ and an edge connecting two vertices if the faces in $\mathcal{T}$ are adjacent. Note that there is a vertex v_{out} representing the outer face of $\mathcal{T}$.*

Fig. 5.2a gives an example of a dual graph $\mathcal{T}^*$ on $\mathcal{T}$, where V^* and E^* are shown in red circles and line segments. The blue circle shows the v_{out}. Note s and t are in the outer face represented by v_{out} since they are outside of the convex hull of $\mathcal{P}_\tau$.

Definition 7 (A walk w on $\mathcal{T}^*$)**.** *Given $\mathcal{T}^* = (V^*, E^*)$, a walk w on an $\mathcal{T}^*$ is a finite alternating sequence of vertices and edges.*

Definition 8 (s-t cycle on $\mathcal{T}^*$)**.** *An s-t cycle on $\mathcal{T}^*$ is a walk w on $\mathcal{T}^*$ with the starting and ending vertex v_{out} representing the outer face. Note that there is no repeated vertices on the walk except for the starting and ending vertex.*

Fig. 5.2b gives an example of s-t on $\mathcal{T}^*$ starting and ending at v_{out}.



(a) (b) (c)

Figure 5.2: (a) Dual graph in red. (b) A loopy walk through $\mathcal{T}^*$. (c) A channel (interior of blue lines).

Definition 9 (Channel)**.** *A channel is a series of triangles uniquely determined by a loopless walk on $\mathcal{T}^*$.*

Fig. 5.2c gives an example of a channel determined by a walk on $\mathcal{T}^*$. Note that the starting and ending triangles are both represented by v_{out}.

Proposition 5.0.5. *An s-t cycle $\mathcal{T}^*$ uniquely determines a homotopy class of a path through $\mathcal{P}_\tau$.*

Proof. Because of the duality between cycles and cuts in dual planar graphs, an s-t cycle on $\mathcal{T}^*$ uniquely determines a cut on $\mathcal{T}$. According to Proposition 5.0.4, the cut on $\mathcal{T}$ is in turn equivalent to a homotopy class of paths through $\mathcal{P}_\tau$. Therefore, an s-t cycle is also equivalent to a homotopy class. □

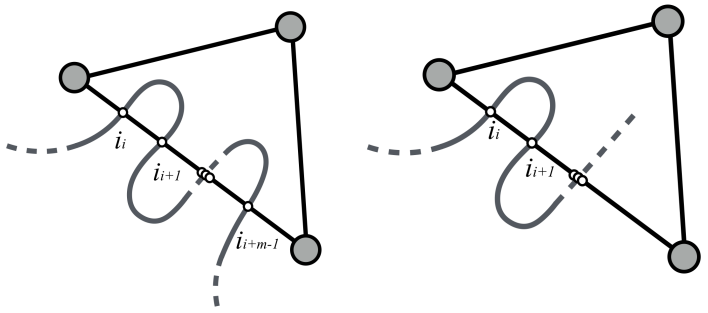
Theorem 5.0.6. *A channel on $\mathcal{T}$ uniquely determines a homotopy class of a path through $\mathcal{P}_\tau$.*

Proof. By Definition 9 and Proposition 5.0.5. □

Theorem 5.0.7. *Every path through $\mathcal{P}_\tau$ uniquely determines a walk on $\mathcal{T}^*$, which is equivalent to a channel by Definition 9.*

Proof. Let $\mathbf{f} : [0, 1] \to \mathbb{R}^2$ be a path intersecting the edges of $\mathcal{T}$ at intersection points $\{i_0, i_1, \ldots, i_n\}$. Let $I_{e_i} = \{i_i, i_{i+1}, \ldots, i_{i+m-1}\}$ be m consecutive intersecting points with the same edge e_i (see Fig. 5.3a). Given $||I_{e_i}|| = m$, I_{e_i} can be reduced to I'_{e_i}, where $||I'_{e_i}|| = m \mod 2$ without changing the homotopy class of $\mathbf{f}$. Namely, if m is odd, then I_{e_i} can be reduced to $I'_{e_i} = \{i_{i+j}\}, j \in [0, m-1]$ and if m is even, I_{e_i} can be reduced to $I'_{e_i} = \emptyset$. This is because that for any two consecutive intersection points i_j and i_{i+1} with the same edge, the line segment $e_{i_{i_j \to i_{j+1}}}$ and path segment $\mathbf{f}_{i_{i_{j+1} \to i_j}}$ forms a loop that contains no vertices from $\mathcal{P}_\tau$ inside, otherwise intersections with other edges will occur between i_j and i_{j+1} because vertices are all connected. Therefore, a path $\mathbf{f}$ with the set of intersection points given by $I_\mathbf{f}$ is homotopically equivalent to $\mathbf{f}'$ with $I_{\mathbf{f}'} = I_\mathbf{f} - \{i_j, i_{j+1}\}$. Thus the reduction of every I_{e_i} described above is legitimate. An example is illustrated by Fig. 5.3. In Fig. 5.3a when m is even, the path is equivalent to never intersecting with edge e_i. In Fig. 5.3b, when m is odd, the path is equivalent to only intersecting with edge e_i once.

As shown above, a path $\mathbf{f}$ can be reduced to $\mathbf{f}'$ such that every two consecutive intersections of $\mathbf{f}'$ are with distinct edges of $\mathcal{T}$. For each intersection points $i_j \in I_\mathbf{f}$, an edge in $\mathcal{T}^*$ can be uniquely determined. For every two consecutive intersection points $i_j, i_{j+1} \in I_\mathbf{f}$, the triangle that both edges belong to (a vertex in $\mathcal{T}^*$) can be uniquely determined. Therefore, consecutive edges of $\mathcal{T}^*$ given by $I_\mathbf{f}$ are connected with each other, which forms unique a walk on $\mathcal{T}^*$. □



(a) m is even. (b) m is odd.

Figure 5.3: Illustration for Theorem 5.0.7

By proving Theorem 5.0.6 and Theorem 5.0.7, we show that there is an one-to-one correspondence between a channel in $\mathcal{T}$ and a homotopy class of paths through $\mathcal{P}_\tau$. In other words, a path on the discrete graph $\mathcal{T}^*$ is related to a class of homotopic paths in the continuous $\mathbb{R}^2$ space, where searching in the former is significantly more efficient than searching in the latter.

Chapter 6

Conclusions and Future Work

We introduced a geometry-based planning framework for crowd navigation. Our primary contribution is efficient global planning while accounting for obstacle dynamics; a proof of completeness and optimality is provided. Empirically, dynamic channels improved success rate by up to a factor of six in the most crowded environments and task completion by up to a factor of three, and decreased computational burden by a factor of two. However, our framework has limitations. First, the method assumes perfect measurements. Second, we assume linear pedestrian motion models; even though our method was evaluated on real world pedestrian datasets with extended periods of nonlinear motion, further experiments are needed to validate robustness in interactive crowds. Future work will integrate dynamic channels with low-level planners to increase this robustness. Additionally, we will extend our architecture for multi-robot navigation, crowd simulation, and the handling of both dynamic and static obstacles with arbitrary shapes and movement uncertainty.

Bibliography

[1] B. Banerjee and B. Chandrasekaran. A framework of Voronoi diagram for planning multiple paths in free space. *Journal of Experimental & Theoretical Artificial Intelligence*, 2013.

[2] P. Brož, M. Zemek, I. Kolingerová, and J. Szkandera. Dynamic Path Planning With Regular Triangulations. 24(3):119–142, 2014.

[3] B. Chazelle. A theorem on polygon cutting with applications. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982.

[4] J. Chen, C. Luo, M. Krishnan, M. Paulik, and Y. Tang. An enhanced dynamic Delaunay triangulation-based path planning algorithm for autonomous mobile robot navigation. 2010.

[5] Y. F. Chen, M. Everett, M. Liu, and J. P. How. Socially Aware Motion Planning with Deep Reinforcement Learning. In *IROS*, 2017.

[6] H. Choset and J. Burdick. Sensor-based exploration: The hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):96–125, 2000.

[7] D. Connell and H. M. La. Dynamic Path Planning and Replanning for Mobile Robots using RRT*. 2017.

[8] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[9] D. Demyen and M. Buro. Efficient triangulation-based pathfinding. *Proceedings of the 21st National Conference on Artificial Intelligence*, 6:942–947, 2007.

[10] P. Fiorini and Z. Shiller. Motion Planning in Dynamic Environments using Velocity Obstacles. *International Journal of Robotics Research*, 1998.

[11] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation Magazine*, 1997.

[12] T. Fraichard and H. Asama. Inevitable Collision States: a Step Towards Safer Robots? In *ICIRS*, 2003.

[13] M. Garber and M. Lin. Constraint-based motion planning using voronoi diagrams. *Algorithmic Foundations of Robotics V*, 2004.

[14] P. Henry, C. Vollmer, B. Ferris, and D. Fox. Learning to Navigate Through Crowded Environments. In *ICRA*, 2010.

[15] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Lecture Notes in Computer Science*, 1991.

[16] Jur P. van den Berg and Mark H. Overmars. Roadmap-Based Motion Planning in Dynamic Environments. *Transactions on Robotics*, 2005.

[17] M. Kallmann. Path planning in triangulations. *International Joint Conference on Artificial Intelligence*, pages 49–54, 2005.

[18] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. In *IJRR*, 2011.

[19] L. Kavraki, L. Kavraki, P. Svestka, P. Svestka, J.-C. Latombe, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *Robotics and Automation*, 1996.

[20] L. E. Kavraki, M. N. Kolountzakis, and J.-c. Latombe. Analysis Of Probabilistic Roadmaps For Path Planning. *IEEE Transactions on Robotics and automation*.

[21] T. Kruse, R. Alami, A. K. Pandey, and A. Kirsch. Human-aware robot navigation: a Survey. In *RAS*, 2013.

[22] M. Kuderer, H. Kretzschmar, and W. Burgard. Teaching Mobile Robots to Cooperatively Navigate in Populated Environments. In *IROS*, 2013.

[23] M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard. Online generation of homotopically distinct navigation paths. *IEEE International Conference on Robotics and Automation*, 2014.

[24] J. Kuffner and S. LaValle. RRT-connect: An efficient approach to single-query path planning. *IEEE International Conference on Robotics and Automation.*, 2000.

[25] B. Lau, C. Sprunk, and W. Burgard. Kinodynamic motion planning for mobile robots using splines. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2427–2433, 2009.

[26] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

[27] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, 2007.

[28] M. Luber, L. Spinello, J. Silva, and K. O. Arras. Socially-Aware Robot Navigation: A Learning Approach. In *IROS*, 2012.

[29] C. I. Mavrogiannis, V. Blukis, and R. A. Knepper. Socially Competent Navigation Planning by Deep Learning of Multi-Agent Path Topologies. In *IROS*, 2017.

[30] J. Müller, C. Stachniss, K. O. Arras, and W. Burgard. Socially Inspired Motion Planning for Mobile Robots in Populated Environments. *Computer*, 2008.

[31] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: modeling social behavior for multi-target tracking. In *International Conference on Computer Vision*, 2009.

[32] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807, 1993.

[33] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. Efficient trajectory optimization using a sparse model. *2013 European Conference on Mobile Robots*.

[34] J. Snape, J. van den Berg, S. Guy, and D. Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 2011.

[35] M. Svenstrup, T. Bak, and J. Andersen. Trajectory Planning for Robots in Dynamic Human Environments. In *IROS*, 2010.

[36] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. *IEEE International Conference on Intelligent Robots and Systems*, 2010.

[37] J. van den Berg, S. Guy, M. Lin, and D. Manocha. Reciprocal n-body Collision Avoidance. In *ISRR*, 2009.

[38] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha. Optimal Reciprocal Collision Avoidance for Multi-Agent Navigation. page 8, 2010.

[39] J. Van Den Berg and M. Overmars. Kinodynamic motion planning on roadmaps in dynamic environments. *IEEE International Conference on Intelligent Robots and Systems*, 2007.

[40] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. *IEEE International Conference on Robotics and Automation*, 2011.

[41] D. Wilkie, J. Van Den Berg, and D. Manocha. Generalized velocity obstacles. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, (June 2014):5573–5578, 2009.

[42] H. Yan, H. Wang, Y. Chen, and G. Dai. Path planning based on Constrained Delaunay Triangulation. *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 5168–5173, 2008.

[43] M. Zucker, J. Kuffner, and M. Branicky. Multipartite RRTs for rapid replanning in dynamic environments. *IEEE International Conference on Robotics and Automation*, 2007.