VISIBILITY-CONSTRAINED PATH PLANNING FOR UNMANNED AERIAL VEHICLES

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Vaibhav Bisht

August 2023

ABSTRACT

Visibility-constrained path planning refers to problems in which a robot must navigate an environment populated by obstacles and occlusions (e.g. opaque objects) whereby avoiding line-of-sight (LOS) detection by one or more sensors. Example applications are unmanned aerial vehicles (UAVs) used to perform tasks like warehouse inventory monitoring where the UAV must avoid detection by the warehouse security system. This thesis presents a visibility-constrained path planning framework for a quadcopter with an onboard camera to navigate in partially known environments, avoiding collisions with obstacles and LOS detection inside sensor visibility regions created using key concepts of visibility theory. A probabilistic roadmap (PRM) is used to find a path on the prior map with sensor visibility regions. Panoptic segmentation is used to obtain a pixel-wise semantic mask to identify obstacles in the RGB frames captured online. The inverse projection method is then used to map these obstacles on an online map of the workspace using ground truth depth maps and collisions with the quadcopter's path are checked. Additionally, findings on the limitations of machine-learning based monocular depth estimation for online mapping are also discussed. A hybrid path replanning algorithm is used that locally replans the path to avoid collisions with obstacles identified online. If local replanning fails to find a new local path, global replanning is performed using an updated PRM. A simulation environment in Unreal Engine is used to test the visibility-constrained path planning framework with the quadcopter imported from AirSim.

BIOGRAPHICAL SKETCH

Vaibhav studied Mechanical Engineering in India where he pursued research on automotive controls, computer vision and Industry 4.0. He joined Laboratory for Intelligent Systems and Control (LISC) after starting at Cornell in the Fall of 2021 and got to work on a diverse set of projects on computer vision, motion planning, active perception, robotics simulation, virtual reality and reinforcement learning.

Apart from academics and research, Vaibhav was awarded the GRASSHOPR fellowship in 2022 and served as the treasurer of Robotics Graduate Student Organization (RGSO) at Cornell. He personally enjoys reading about geopolitics and never misses a chance to go for running, especially on the bright and sunny winter mornings of Ithaca.

This document is dedicated to all Cornell graduate students.

ACKNOWLEDGEMENTS

I would like to thank Dr. Silvia Ferrari for her guidance and mentorship throughout my involvement in the varied research undertakings at LISC. I also thank Dr. Tapomayukh Bhattacharjee for his insightful and critical comments on my work, may it be for research or as part of his course on robot manipulation.

Secondly, I thank all the members of LISC who created an environment of discussion where you could approach anyone anytime, may it be for research, coursework or general robotics discussion. I in particular thank Yucheng Chen, Kyung Rak Jung, Suming Qiu and Sushrut Surve for their roles in work mentioned here including creating the simulation environment and devising the probabilistic roadmap with the sensor visibility regions. I also appreciate members of the robotics fraternity at Cornell University who were the like-minded people I could spend time with.

Lastly, I thank all the members of my family at home in India and here in the US, who ensured that I make the best of my time as a student at Cornell University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Recent advancements in machine learning (ML), computer vision, embedded sensing and communication systems are enabling robots with cognitive, sensing and decision-making skills [1]. Autonomous robots equipped with sensors and computer vision algorithms find promising use in domains of manufacturing [2, 3], medical surgery [4, 5], agriculture [6], healthcare [7, 8] and food packaging [9].

A quadcopter is a rotary-wing aircraft that operates using four rotors known as rotor blades, two rotating in clockwise and the remaining two rotating in an anti-clockwise direction. The autonomy and flexibility of these flying robots enable a wide range of indoor and outdoor sensing as well as navigation tasks that include search-and-rescue operations, disaster response efforts and environmental surveillance [10]. These robots also operate well in swarms where they promise scalability in terms of the number of vehicles engaged and stability for scenarios where some of the vehicles may fail to operate [11].

Visibility-constrained path planning can be referred to planning a path for an autonomous robot avoiding LOS detection by one or more sensors in the presence of occlusions (e.g. opaque objects). According to the concepts of visibility theory discussed in [12], sensor visibility regions are subsets of the sensor field-of-view (FOV) where the LOS visibility condition in presence of occlusions is satisfied. Different approaches have been taken to perform such path plan-

ning, although the study in [13] shows that fairly less work has been done in this domain, especially for UAVs.

The earlier efforts on visibility-constrained path planning use strategies that do not directly focus on robot motion planning. Work in [14] uses insect-scale ground robots to avoid detection. This approach suffers from limited operating range of the robots and use of inadequate sensing hardware like micro-cameras that capture only limited details. Another strategy developed by [15] uses light-intensity sensors to identify dark locations where the robot could hide. This approach lacks applicability as it significantly relies on the lighting conditions of the workspace.

One of the initial studies focusing on planning the path for an aerial vehicle uses the FOV of omnidirectional sensors as the visibility regions [16] but this approach does not consider any obstacles in the workspace. Work in [17] creates a grid-based environment with a visibility-distance cost used to generate a path using their novel search-based Dark Path algorithm that finds the steepest descent path based on cost. Although this method is limited to use in known environments. Work in [18] extends the previous work to partially known environments by an online update of the visibility-distance cost by using a visibility cost function. The Dijkstra's algorithm [19] is identified to be more effective than the Dark Path algorithm for shortest path search in this framework as it iteratively updates the cost for reaching a node if a shorter path is found [20].

Detailed prior maps for many regions cannot be obtained due to their large spatial scales and impracticable maintenance considering the rate at which they change [21]. Obstacle avoidance is a major consideration for visibility-constrained path planning in such environments that are only partially known *a*

priori. This is because the replanned path needs to be constrained to free space, ensuring collision avoidance with other obstacles as well as preventing entry into sensor visibility regions. A hybrid local and global path planning framework can be adopted for such instances [22]. A global path can be created offline using methods like PRM [23] using map of the entire workspace, followed by local replanning to avoid collisions with any newly identified obstacles. If local replanning fails, the roadmap updated on the online map can be queried to find an alternate global path.

1.2 Contribution

This thesis presents a visibility-constrained path planning framework for a quadcopter to navigate in a partially known environment without entering sensor visibility regions. A combined online and offline data processing framework is developed based on the robotics pipeline of perception, planning and control. A prior map with sensor visibility regions is used to generate a path for the robot using PRM. Panoptic segmentation and the inverse projection method are used to map new obstacles in a probabilistic occupancy grid. A path replanning algorithm is developed that performs local path replanning to locally modify the path for avoiding collisions with newly detected obstacles. If local replanning fails to find an alternate path in free space, global replanning is performed where the updated roadmap is queried.

CHAPTER 2

PROBLEM FORMULATION AND ASSUMPTIONS

The visibility-constrained path planning framework considers a camera-equipped drone navigating from defined start to end positions avoiding collision with obstacles and entry into sensor visibility regions. For illustration purposes, the drone is assumed to be operating in a closed, bounded and 2-D workspace, $\mathcal{W} \subset \mathbb{R}^2$.

The drone navigates in the workspace with a rigid geometry $\mathcal{A} \subset \mathcal{W}$. Every point in $\mathcal{A}$ can be represented by the position and orientation of the moving body frame $\mathcal{F}_\mathcal{A}$ relative to an inertial frame $\mathcal{F}_\mathcal{W}$ having origin $O_\mathcal{W} \in \mathcal{W}$. The body frame $\mathcal{F}_\mathcal{A}$ is assumed to be embedded in $\mathcal{A}$ with the origin $O_\mathcal{A}$ being the onboard camera position. The robot configuration $\mathbf{q} = [\mathrm{s}^T, \theta]^T$ comprises of the position, $\mathbf{s} \in \mathcal{W}$, and orientation, $\theta \in \mathbb{S}^1$, of the onboard camera with respect to $\mathcal{F}_\mathcal{W}$. The configuration space is $\mathcal{C} = \mathcal{W} \times \mathbb{S}^1$ and comprises of all possible robot configurations with respect to $\mathcal{F}_\mathcal{W}$.

The motion of the drone is based on models of the robot kinematic and dynamic constraints, governed by the ordinary differential

$$\dot{\mathbf{q}}(t) = \mathbf{f}[\mathbf{q}(t), \mathbf{u}(t), t], \quad \mathbf{q}(t_0) = \mathbf{q}_0 \tag{2.1}$$

where the control vector is given by $\mathbf{u}(t) \in \mathcal{U}$ and $\mathcal{U} \subset \mathbb{R}^m$ is the m-dimensional space of admissible control inputs. The vector function $\mathbf{f}(.)$ is assumed to obey single-integrator dynamics.

The 2-D workspace is assumed to comprise of M opaque and solid objects, $\mathcal{B}_j \subset \mathcal{W}$, indexed by $j \in \mathcal{J}$, $\mathcal{J} = \{1, \ldots, M\}$, that act as obstacles. For obstacle $\mathcal{B}_j$,

C-obstacle is the set of configurations in which the drone with geometry $\mathcal{A}$ can collide with obstacle $\mathcal{B}_j$, that is

$$\mathcal{CB}_j \triangleq \left\{ \mathbf{q} \in C \,|\, \mathcal{B}_j \cap \mathcal{A}(\mathbf{q}) \neq \emptyset \right\} \tag{2.2}$$

Therefore, the free configuration space, $C_{free} \subset \mathcal{W}$, the robot must travel in is

$$C_{free} = C \setminus \bigcup_{j=1}^{M} \mathcal{CB}_j \tag{2.3}$$

The workspace, $\mathcal{W}$, also comprises of N omnidirectional sensors at positions, $v_p \in \mathcal{W}$, indexed by $p \in \mathcal{P}$, $\mathcal{P} = \{1, \ldots, N\}$. For a sensor position v_p, a visibility region is defined as a rigid object that can be described by the closed and bounded subset $\mathcal{V}_p \subset \mathcal{W}$. The sensor visibility region model comprises of free space inside a circle of radius, $r > 0$, centered at v_p, where any point inside the visibility region satisfies the LOS visibility condition.

Given an opaque object, $\mathcal{B}_j \subset \mathcal{W}$, a position, $b \in \mathcal{W}$, can be observed from $v_p \in \mathcal{W}$ by LOS visibility if and only if

$$L(v_p, b) \cap \mathcal{B}_j = \emptyset \tag{2.4}$$

where $L(v_p, b)$ is a line segment connecting v_p and b

$$L(v_p, b) \triangleq \{(1 - \gamma)v_p + \gamma b \,|\, \gamma \in [0, 1]\} \tag{2.5}$$

As the drone navigates in $\mathcal{W}$, it can identify and map new obstacles using the onboard camera sensors. These obstacles, $\mathcal{D} \subset \mathcal{W}$, can be of varying geometries and therefore, a square occupancy grid, G, is used to map the new obstacles in $\mathcal{D}$ on the 2-D workspace.

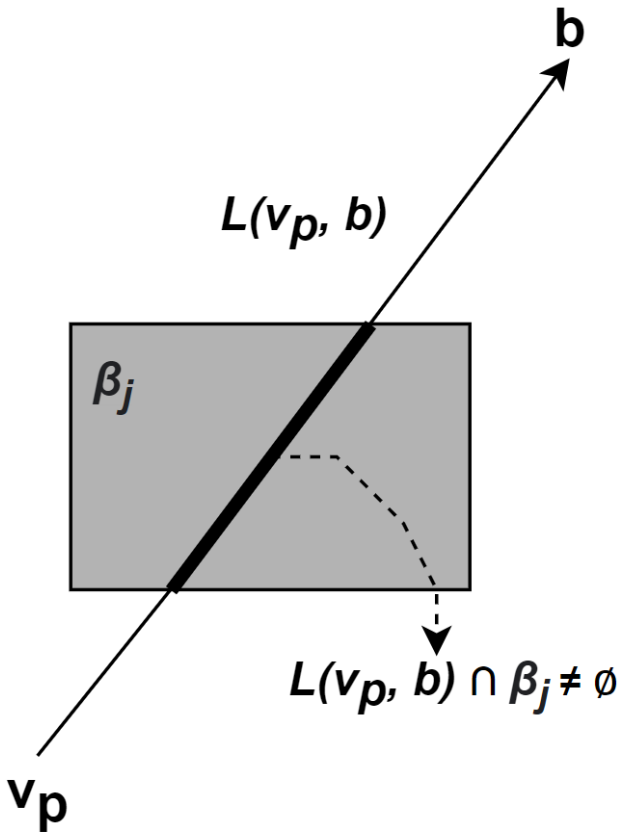


Figure 2.1: Opaque object occludes the position b in workspace

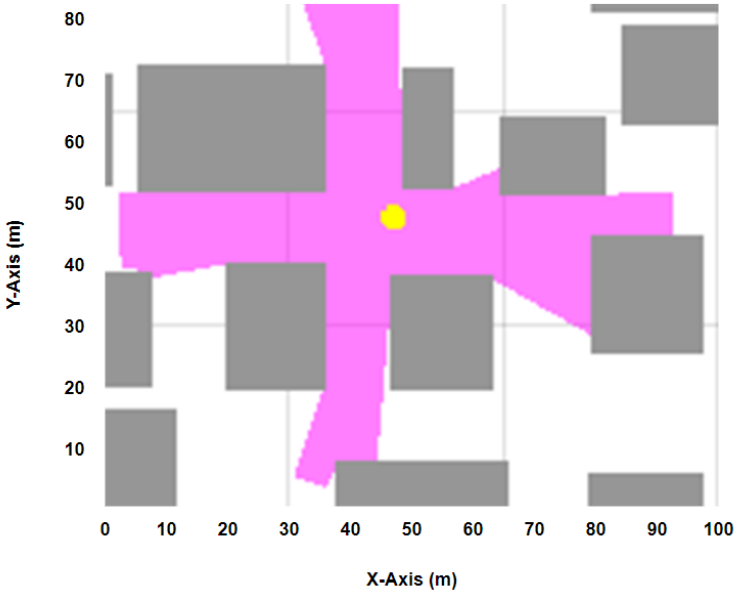


Figure 2.2: Sensor visibility region (pink) in workspace with occlusions (grey)

The drone's start and end positions can be mapped to the configuration space as $\mathbf{q}_0$ and $\mathbf{q}_f$. A path from $\mathbf{q}_0$ and $\mathbf{q}_f$ can be depicted as a continuous map

$$\tau : [0, 1] \rightarrow C_{\text{free}} \tag{2.6}$$

where $\tau(0) = \mathbf{q}_0$ and $\tau(1) = \mathbf{q}_f$.

This path can be expressed as a piecewise-linear path, also called a poylgonal chain [12] having vertices ($\mathbf{q}_0$, $\mathbf{q}_1$, ..., $\mathbf{q}_n$).

Therefore, the aim of visibility-constrained path planning framework is to:

"Plan a path τ for the drone to navigate in $\mathcal{W}$, avoiding collision with obstacles and LOS detection inside sensor visibility regions."

CHAPTER 3

METHODOLOGY

This chapter presents the methodology developed for the visibility-constrained path planning framework. Elements of the robotics pipeline, perception, planning and control [24, 25] are used to implement the offline planning and online update phases.

3.1 Simulation Environment Design

A simulation environment is created in Unreal Engine due to it's capable physics engine, the PhysX, developed by NVIDIA [26] and high quality of rendering graphics [27, 28]. The industrial city environment pack in Unreal Engine 4.27 is chosen. A representation of the simulation environment can be seen in Figure 3.1.

Figure 3.1: Industrial city environment in Unreal Engine

A quadcopter is added to the environment using AirSim [29], an open-source platform based on Unreal Engine offering physically and virtually realistic simulations. AirSim is primarily designed for use in autonomous system development that includes mobile robotics, deep learning, reinforcement learning, computer vision and robot manipulation [30, 31]. It also has a comprehensive physics engine offering suitable rigid body dynamics and functionalities like high-frequency real-time hardware-in-the-loop (HITL) simulations.

Figure 3.2: Quadcopter from AirSim in industrial city environment

Figure 3.2 shows a quadcopter from AirSim in the Unreal Engine industrial city environment. The AirSim plug-in for Unreal Engine is used to create a "Quadcopter" actor for the Unreal Engine environment. Drone control is subsequently accomplished using the AirSim library in Python 3.

Table 3.1: RGB camera intrinsic parameters

Parameter	**Dimension**
Focal Length	4.9152 mm
Sensor Height	8.12 mm
Sensor Width	8.12 mm

An RGB camera is mounted on the front of the quadcopter to capture input frames and their corresponding ground truth depth maps are obtained from the simulation environment. The intrinsic parameters of the RGB camera are shown in Table 3.1. An Alienware Aurora R13 gaming desktop with 12th Gen Intel® Core™ i7-12700F and 32 GB RAM is used to run these simulations.

3.2 Workflow

The offline planning phase creates a prior map of the workspace that represents the position of previously known solid and opaque obstacles, $\mathcal{B}$, and sensor visibility regions, $\mathcal{V}$. This map is used to generate a prior path for the drone.

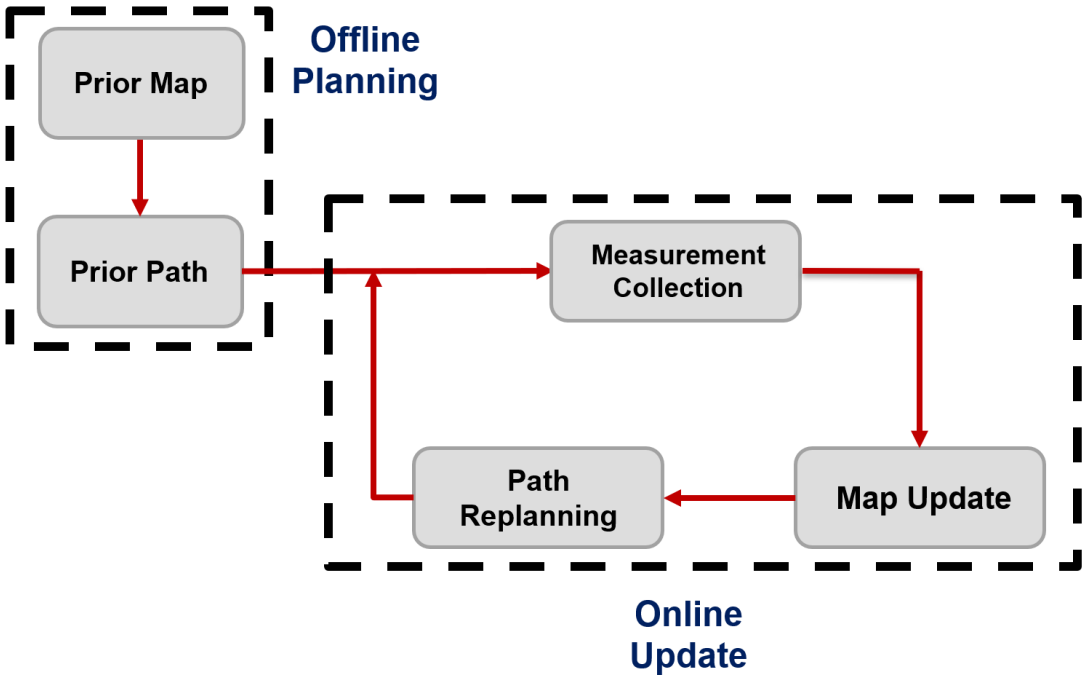


Figure 3.3: Offline planning and online update phases in visibility-constrained path planning

The online update phase is implemented when the drone starts traversing the workspace. At time, t, for a drone configuration, $\mathbf{q}_t$, the sensor measurement from the RGB camera is a 3-channel 8-bit RGB frame, whereas the ground truth depth maps from the Unreal Engine simulation environment are 1-channel 16-bit. The sensor measurements are used to generate 3-D point clouds from which obstacles are mapped to avoid collisions in real-time.

The constituent elements of the offline planning and online update phases can be seen in Figure 3.3.

3.2.1 Offline Planning

The offline planning phase comprises of two steps, prior map determination and prior path generation.

Prior Map

The 2-D map of industrial city in Unreal Engine with convex and polygonal obstacles along with the free configuration space, C_{free}, is shown in Figure 3.4.

The representation of the 2-D workspace is expected to change over the online update phase due to detection of new obstacles that were not mapped offline. Therefore, a 2-D grid, G, is imposed over the workspace where a grid cell, $g \in G$, can have occupancy $\delta_g \in \{0, 1\}$, that is used to represent g being occupied ($\delta_g = 1$) or unoccupied ($\delta_g = 0$).

The log odds value [32, 33] to mark occupancy of each grid cell, $g \in G$, in the prior map is

$$l_g(0) = \log\left(\frac{p(\delta_g = 1)}{(1 - p(\delta_g = 1))}\right) \tag{3.1}$$

where $p(\delta_g = 1) = 0.75$ is the probability of g being occupied.

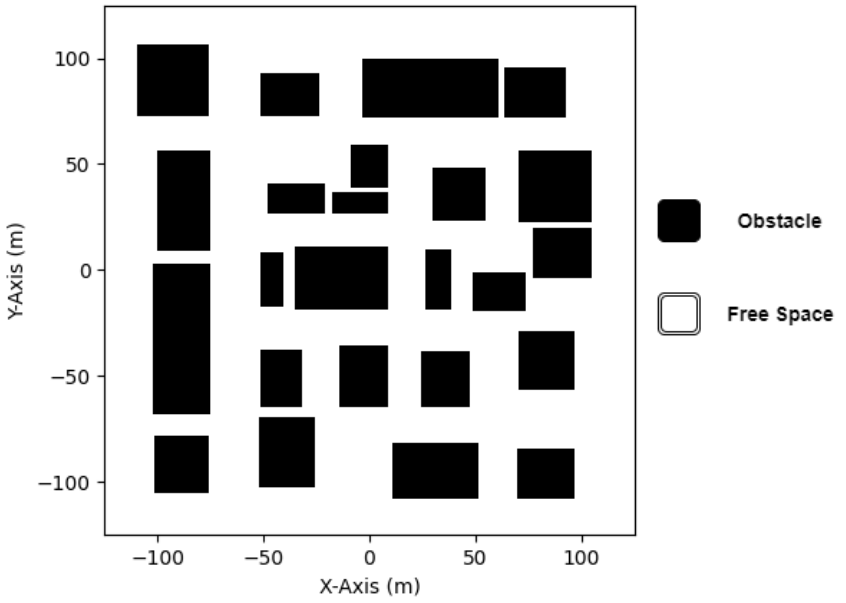


Figure 3.4: 2-D prior map of industrial city

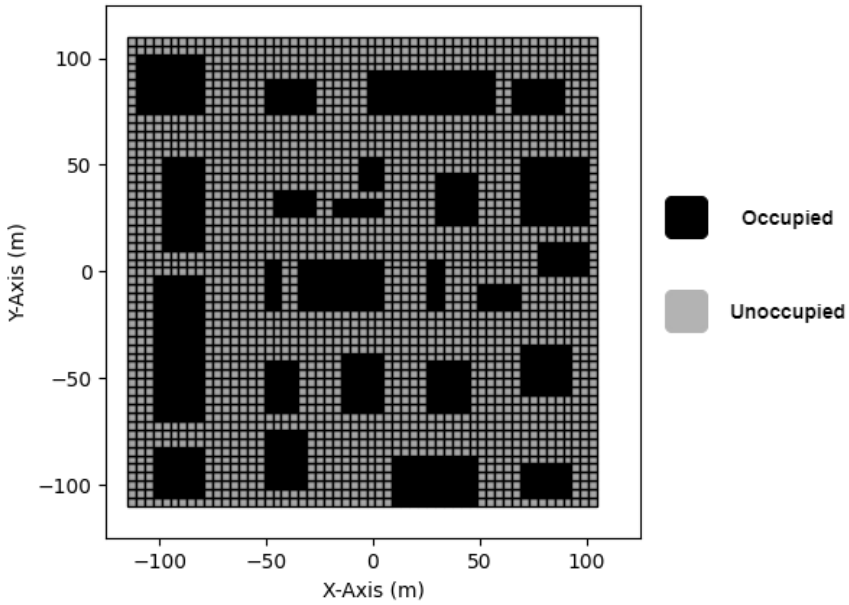


Figure 3.5: 2-D probability occupancy grid over prior map of industrial city

The 2-D prior map layout of the grid world, G, can be seen in Figure 3.5. It comprises of 4x4 grids created over the 2-D workspace of dimension 220x220. Any grid cell that is partially occupied in layout from Figure 3.4 is marked unoccupied, with the online update phase used to update occupancies.

Prior Path

A PRM is used to create the prior path for drone to navigate in the workspace. PRMs are used because of being probabilistically complete [34], that is, the probability of finding a solution for a solvable problem converges to ones as time goes to infinity. PRMs also generate shorter paths than the other sampling-based method of rapidly-exploring random trees (RRT) [35]. RRTs also have an uncertain convergance rate, leading to a suboptimal solution [36]. This is acceptable in high-dimensional spaces where alternate methods may fail but not desirable in the chosen 2-D environment.

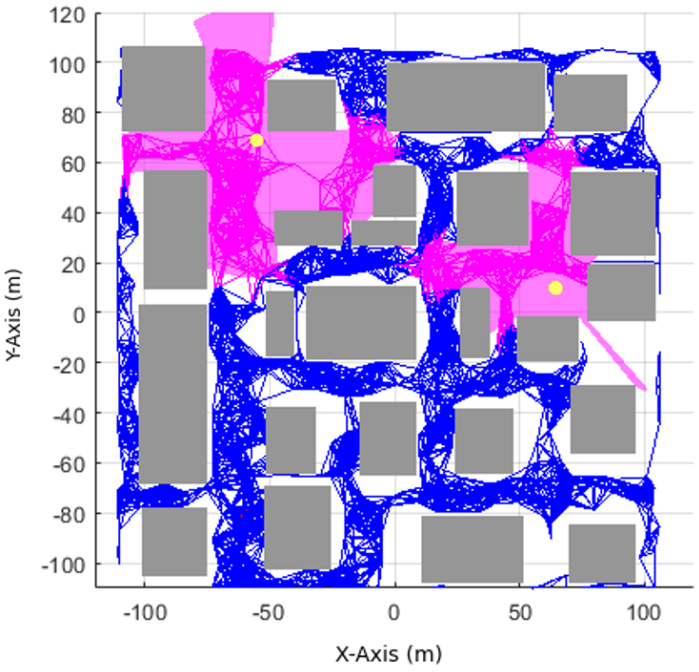


Figure 3.6: Probabilistic roadmap with visibility regions (pink) and arcs (blue)

A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is initialised where $\mathcal{N}$ is the set of drone configurations in free space called milestones and $\mathcal{E}$ is the set of arcs connecting these milestones in the roadmap [10].

Each milestone, $\mathbf{m}_k \in \mathcal{N}$, indexed by $k \in \mathcal{K}$, $\mathcal{K} = \{1, \ldots, K\}$ is an independent and identically distributed (iid) vector drawn from a Uniform PDF, $g_{\mathrm{Y}}(\mathrm{y})$, where sensor configuration vector y is taken as a vector of the random variables denoted by Y [10]. In this thesis, $K = 2000$ milestones are sampled from the PDF. A maximum distance parameter, L = 15 m, is also chosen such that any two milestones $\mathbf{m}_k$ and $\mathbf{m}_l$ that are neighbors in configuration space, that is

$$\|\mathbf{m}_k - \mathbf{m}_l\| \leq L \tag{3.2}$$

are connected by an arc in the roadmap, such that

$$(\mathbf{m}_k, \mathbf{m}_l) \in \mathcal{E} \tag{3.3}$$

if the arc exists in C_{free}.

Arcs are created by connecting each milestone, $\mathbf{m}_k$, to its nearest $k_{max} = 20$ neighbors. According to [10], $k_{max} \approx 20$ allows creating a well-connected graph while avoiding too many arcs that may affect computational complexity. The cost of arc between milestones $\mathbf{m}_k$ and $\mathbf{m}_l$ is given by their Euclidean distance. If the arc lies inside a sensor visibility region, a penalty of 1000 is added to this cost.

The roadmap, $\mathcal{N}$, is queried for milestones closest in distance to the start and final 2-D positions for the drone, $\mathbf{x}_0$ and $\mathbf{x}_f$. Dijkstra's algorithm [19] is used to query a path from the identified milestones on the PRM.

3.2.2 Online Update

The RGB frames and depth maps are used to update the 2-D map and replan path if collisions occur with the obstacles, along with avoiding LOS detection inside sensor visibility regions.

Obstacle Detection and Mapping

Panoptic segmentation [37] is used to obtain pixel-wise labels for different semantic classes [38] in the input RGB frames. Panoptic segmentation is a powerful tool that unifies the capabilities of instance segmentation [39] and semantic segmentation [40] in computer vision. It generates a semantic mask identifying different classes in the input RGB image, like semantic segmentation along with providing unique Instance IDs for *stuff* classes comprising of countable entities like cars and person [41].

Detectron2 [42], an open-source deep learning library was used for performing panoptic segmentation. The library has trained models of different neural network architectures to perform computer vision tasks like instance segmentation [39], semantic segmentation [40] and person keypoint detection [43]. Panoptic segmentation is performed in Detectron2 using a Panoptic Feature Pyramid Network (PFPN) [44] that uses a Convolutional Neural Network (CNN) architecture combining the ResNet-101 [45] backbone with the Feature Pyramid Network (FPN) [46] for feature extraction and object detection tasks.

Selected panoptic segmentation result on RGB frame in Figure 3.7 from the Unreal Engine simulation environment can be seen in Figure 3.8.

Monocular depth estimation using deep learning was tested to assess the im-

Figure 3.7: Select RGB frame from Unreal Engine simulation environment-I

Figure 3.8: Panoptic segmentation result

plementation of the visibility-constrained path planning framework using RGB cameras without ground truth depth maps from the simulation environment. A self-supervised deep learning model, monodepth2 [47], was used to obtain pixel-wise depth maps for the input RGB image frames.

Monodepth2 has a U-Net architecture-based depth network that outputs a depth map given the input RGB frame, as shown in 3.9. The depth network model chosen in this thesis was trained using stereo image pairs from the KITTI Vision Benchmark Suite [48]. The training process uses the depth map and relative camera pose between the stereo images obtained from a pose network

shown in Figure 3.10. The combined loss function used for training the depth network is

$$L_{combined} = \mu \cdot L_p + \lambda \cdot L_s \tag{3.4}$$

where L_p is the per-pixel loss and L_s is the masked photometric loss [47]. $\mu \in \{0, 1\}$ is the per-pixel mask and λ is the smoothness term set to 0.001.

Figure 3.9: Depth map from monodepth2 (relative scale)

The depth map from the Unreal Engine simulation environment is also shown in 3.11.

The pixel-wise depth maps are mapped to 3-D space by the inverse projection method [49] using the RGB camera's intrinsic and extrinsic parameters.

The generated 3-D point clouds for another select RGB frame using depth maps from monodepth2 and the simulation environment can be seen in Figure 3.13(a) and 3.13(b), and Figure 3.14(a) and 3.14(b). As observed, the 3-D reconstruction from depth maps using monodepth2 has qualitative inaccuracies. This behavior is attributed to a lack of domain transferability of the monodepth2 network [50]. Multiple monocular depth estimation networks also suffer from a lack of suitable training datasets. It is desired to have depth maps from sensors like LiDAR for training, however, only a few open datasets offer

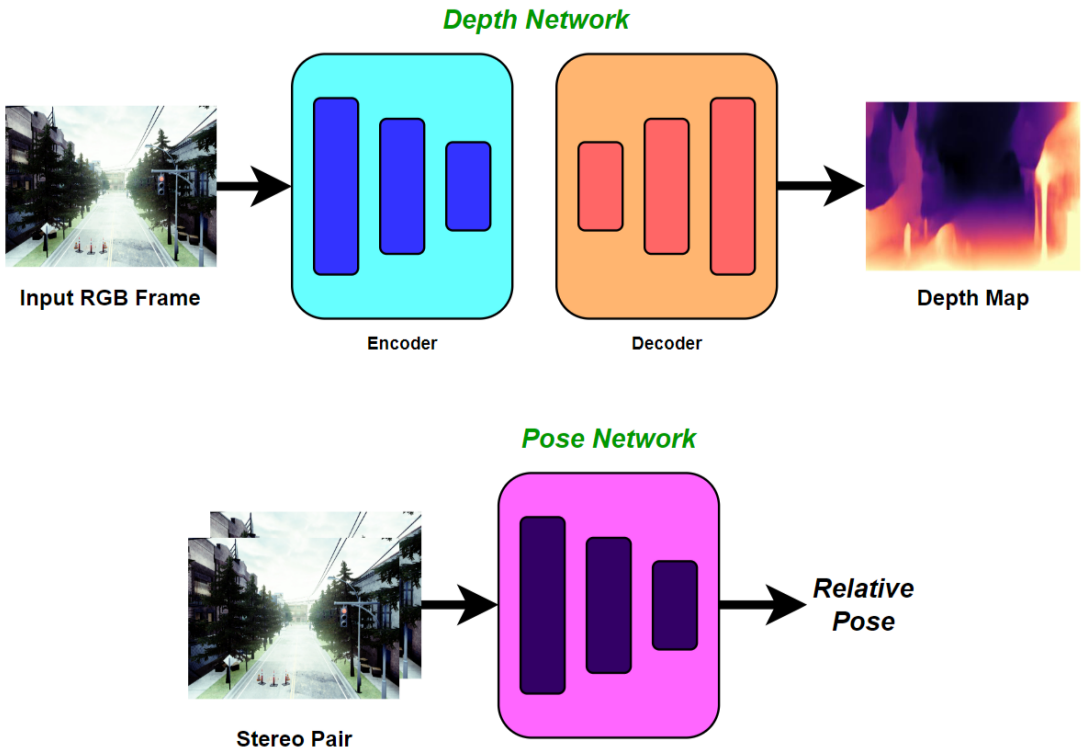


Figure 3.10: Depth network and pose network in monodepth2

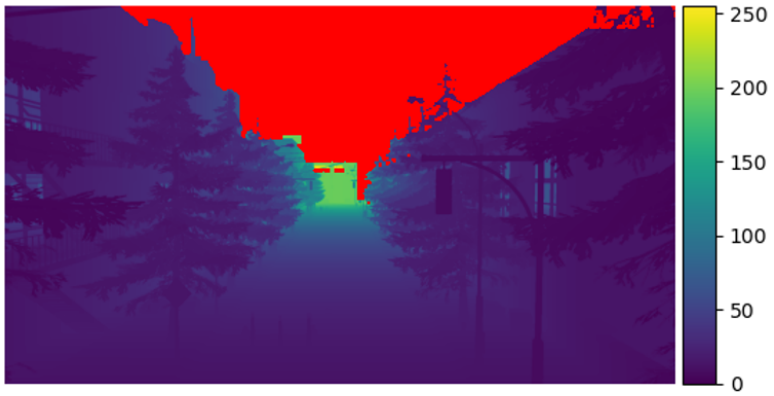
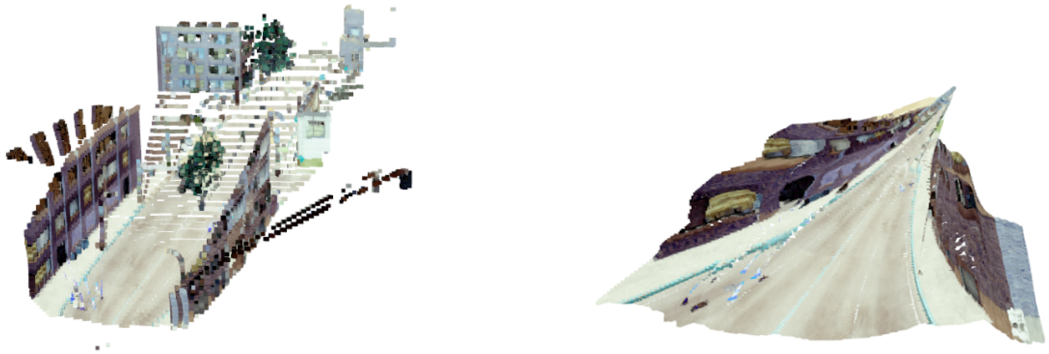


Figure 3.11: Depth map from the Unreal Engine simulation environment

such LiDAR depth data, which is also very sparsely annotated [51]. The depth in monodepth2 also exists on a relative scale and requires a robust calibration for use on the absolute scale, referring to work done by [52]. Therefore, ground truth depth maps from the simulation environment are used in this thesis.

Figure 3.12: Select RGB frame from Unreal Engine simulation environment-II



(a) Using simulation environment's depth map

(b) Using monodepth2's depth map

Figure 3.13: 3-D point clouds (orthographic-view)

(a) Using simulation environment's depth map

(b) Using monodepth2's depth map

Figure 3.14: 3-D point clouds (front-view)

The occupancy grid map, G, was updated by mapping the points in the 3-D point cloud to 2-D space. Assuming the grid cell occupancy probability distributions are independent [53] at time step, t, the Baye's rule can be used to denote belief of grid cell, $g \in G$, being occupied as $P(\delta_g = 1 \mid z_{1:t}, q_{1:t})$ and unoccupied as $P(\delta_g = 0 \mid z_{1:t}, q_{1:t})$. Here, z_t is the sensor measurement at time, t.

The log odds value for grid cell, $g \in G$, at time step, t, is updated by

$$l_g(t) = l_g(t-1) + \log\left(\frac{p(\delta_g = 1 \mid z_{1:t}, q_{1:t})}{1 - p(\delta_g = 1 \mid z_{1:t}, q_{1:t})}\right) - l_g(0) \tag{3.5}$$

Path Replanning

A path replanning algorithm is devised to modify the path of drone if it intersects with a newly identified obstacle. For grid map, G, there exists the set, $D \subset G$, comprising of grid cells that the path of drone passes through. There is the set of points, P, such that for each grid cell, $D_k \in D$, there exists $P_k \in P$, $P \subset \mathcal{W}$, which is the midpoint of the part of the path passing through D_k. An example is shown in Figure 3.15 for a straight line path with the points in P being highlighted.

When an obstacle is identified in the online update phase, the path replanning algorithm finds set $E \subset P$ comprising of points lying inside the occupied grid cells along with n elements before and after the occupied grid cells in the order of occurrence in P. The value of n is usually equal to 1 but may be changed based on the type of environment. An example is shown in 3.16, where the set E has a cardinality of 4, $E = \{E_1, E_2, E_3, E_4\}$, with E_2 and E_3 existing in occupied grid cells. The elements create a polygonal chain called the intersecting path segment (in red) which is a part of the prior path that intersects with the obstacle.

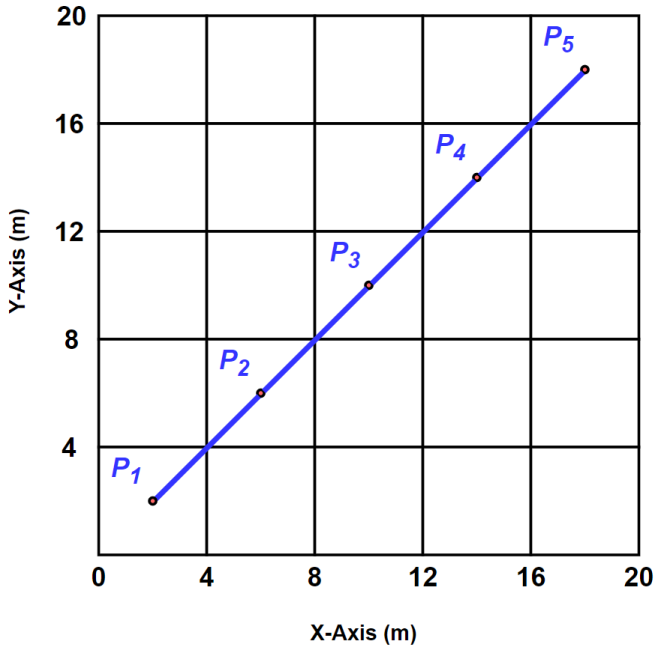


Figure 3.15: Elements of set P in grid environment

Let the intersecting path segment be along the unit vector, $\mathbf{v}$, defined by the direction of line segment connecting E_1 and E_4 directly, $L(E_1, E_4)$. Now, elements of D are displaced in perpendicular directions defined by unit vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ ($\mathbf{v_1}$ = -$\mathbf{v_2}$) by distance ζ. This gives sets E' and E'' where the cth element is

$$E'_c = E_c + \zeta \cdot \mathrm{v}_1 \tag{3.6}$$

$$E''_c = E_c + \zeta \cdot \mathbf{v_2} \tag{3.7}$$

The polygonal chains obtained by connecting elements of E' and E'' with the original path are called candidate path segments (in green). If any of these candidate path segments lies in C_{free}, it is chosen as part of the new local robot path, accomplishing local replanning. When both the candidate path segments are in C_{free}, one is picked at random. If none of the segments lies in free space, global replanning is performed where the updated PRM is queried by Dijkstra's algorithm to obtain a new path to the goal point.

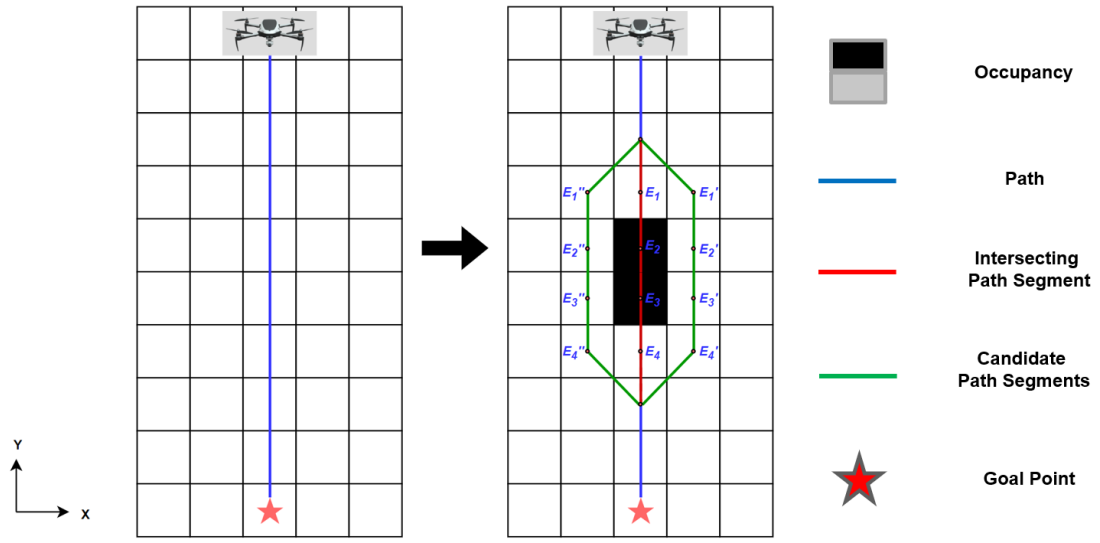


Figure 3.16: Candidate path segment generation

A constraint in the local replanning is that the occupied grid cells need to exist in successive order, or else it is performed multiple times as shown in Figure 3.17. The path is modified locally corresponding to the two different obstacles.

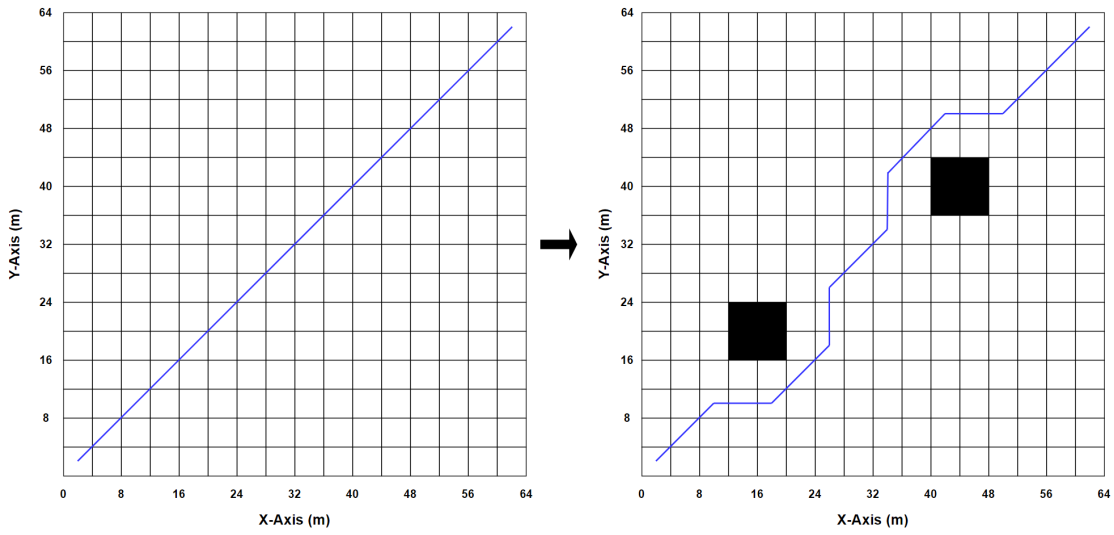


Figure 3.17: Local replanning results

Potential navigation functions [10] are also used for such tasks of avoiding obstacles. These comprise of attractive and repulsive potentials, defined by

$$U(\mathbf{x}) = U_{att}(\mathbf{x}) + U_{rep}(\mathbf{x}) \tag{3.8}$$

where $\mathbf{x}$ represents the current robot configuration in workspace. The robot can then be controlled by virtual forces

$$\mathbf{f}_{pot}(\mathbf{x}) = \mathbf{f}_{att}(\mathbf{x}) + \mathbf{f}_{rep}(\mathbf{x}) \tag{3.9}$$

where

$$\mathbf{f}_{att}(\mathbf{x}) = -\nabla U_{att}(\mathbf{x}) \tag{3.10}$$

$$\mathbf{f}_{rep}(\mathbf{x}) = -\nabla U_{rep}(\mathbf{x}) \tag{3.11}$$

The attractive force pushes the robot towards the goal whereas the repulsive force prompts it to move away from the obstacles. For instance, to reach the goal, $\mathbf{x}_f$, an attractive conic well [10] can be defined in terms of the Euclidean distance between the current and goal configuration

$$d(\mathbf{x}, \mathbf{x}_f) = \|\mathbf{x} - \mathbf{x}_f\| \tag{3.12}$$

as follows

$$U_{att}(\mathbf{x}) = \eta \cdot d(\mathbf{x}, \mathbf{x}_f) \tag{3.13}$$

where η is a scaling factor ($\eta > 0$).

The potential navigation function suffers from local minima where the function gradient at configuration $\mathbf{x}$ is 0, that is

$$\nabla U(\mathbf{x}) = 0 \tag{3.14}$$

The shape of obstacles and number of obstacles/targets primarily guide the robot into a local minima. Harmonic steam functions can be used to ensure that there is no local extrema in the workspace but these include solving the Laplace equation and do not easily extend to account for obstacles and targets that may be detected online [10]. Alternate approaches include filling the well in 2-D and 3-D spaces or using methods like RRT when dimensionality increases [10].

Another case is of multi-target navigation where reaching any target position, a, can be marked by information gain, J_a [10]. The potential in such cases is modified to account for this information gain, being termed the information potential. The information potential lets the robot move out of local minima to move towards a target with greater information gain when the robot may otherwise be stuck in minima using the Euclidean distance based potentials.

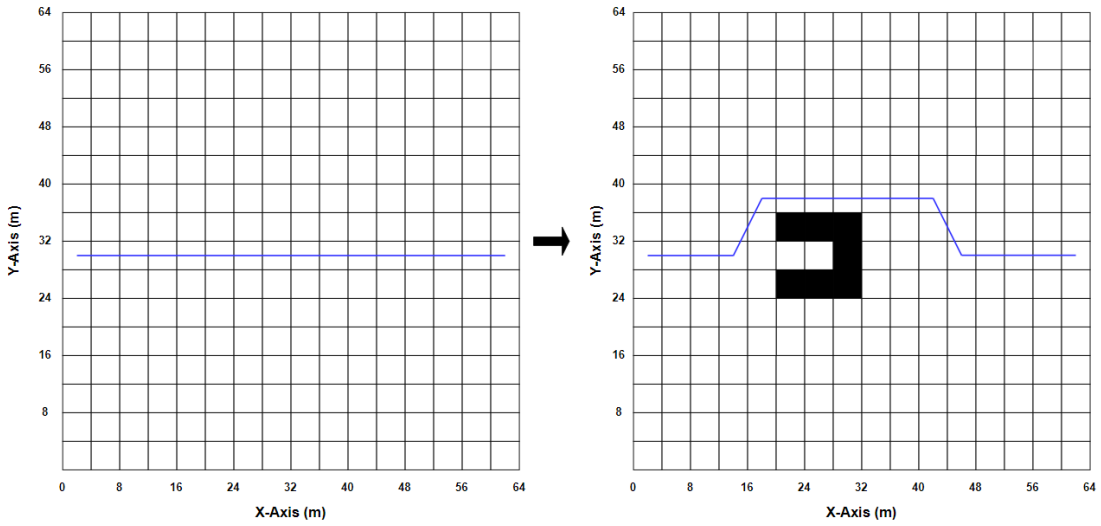


Figure 3.18: Local replanning around concave obstacle

An example of local minima when using potential navigation functions is when the robot goes into a concave obstacle (eg. U-shaped obstacle) as part of the path to reach the goal configuration [54]. In such a case, [10] shows how the robot first moves inside the local minima, samples configurations to move out of it and finally advances towards the target with greater information potential. Using the local replanning framework for this scenario, a candidate path segment can be created to check if an alternate path around this obstacle exists, as seen in Figure 3.18. If such a path exists, the robot can take that directly without running into the local minima. If such an alternate path does not exist, the method shall query the updated PRM.

The local replanning framework is useful for real-time implementation, primarily in less cluttered environments. Methods like D* [55] also modify the path locally but it's search time grows exponentially with grid resolution and size of the graph. RRTs can also be used to sample configurations for moving around the obstacle locally, but the path will be suboptimal [22] and smoothing may affect the real-time applicability. Work in [56] uses the global planner only to create the prior path and then runs the local planner for obstacle avoidance that may fail at times [22]. The discussed path replanning algorithm also updates the PRM using the online map to find a new global path when the local replanning fails.

CHAPTER 4

SIMULATION RESULTS AND DISCUSSION

This chapter presents the visibility-constrained path planning framework run in the Unreal Engine simulation environment.

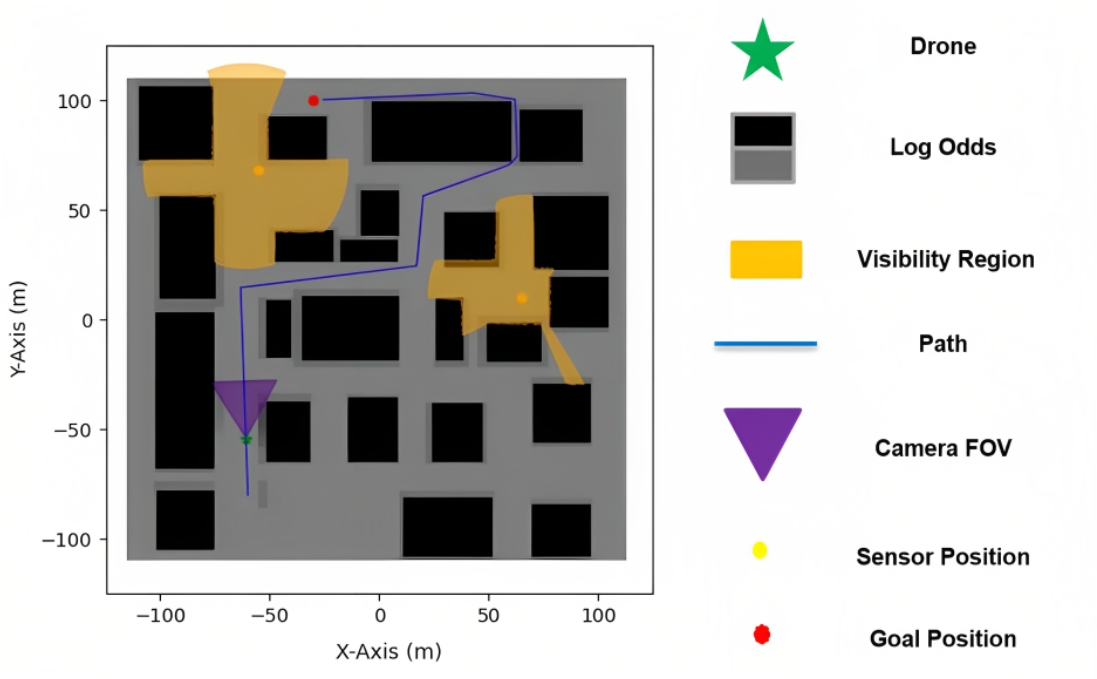


Figure 4.1: Online map of workspace with drone navigating on prior path

Figure 4.1 shows drone navigating in workspace following the prior path generated using the PRM in the offline planning phase.

Figure 4.2 shows an input RGB frame with a previously unknown obstacle in the workspace. Here, local replanning is performed to obtain candidate path segment in C_{free}, as shown in Figure 4.3.

The online update phase identifies an obstacle that requires global replanning due to not having free space to locally replan the path around it, shown in Figure 4.4.

Figure 4.2: Previously unseen obstacle-I

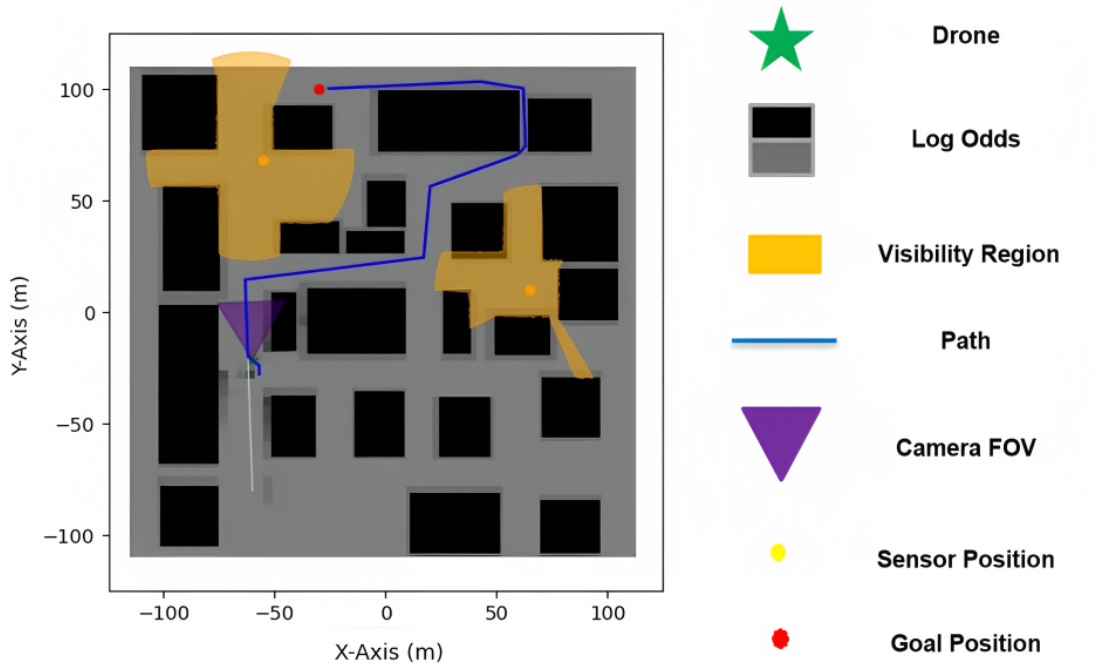


Figure 4.3: Online map of workspace with drone navigating on locally re-planned path

Figure 4.4: Previously unseen obstacle-II

Therefore, the PRM is updated using the online map and queried using the Dijkstra's algorithm to obtain a new global path. The path shown in Figure 4.5 is subsequently created.

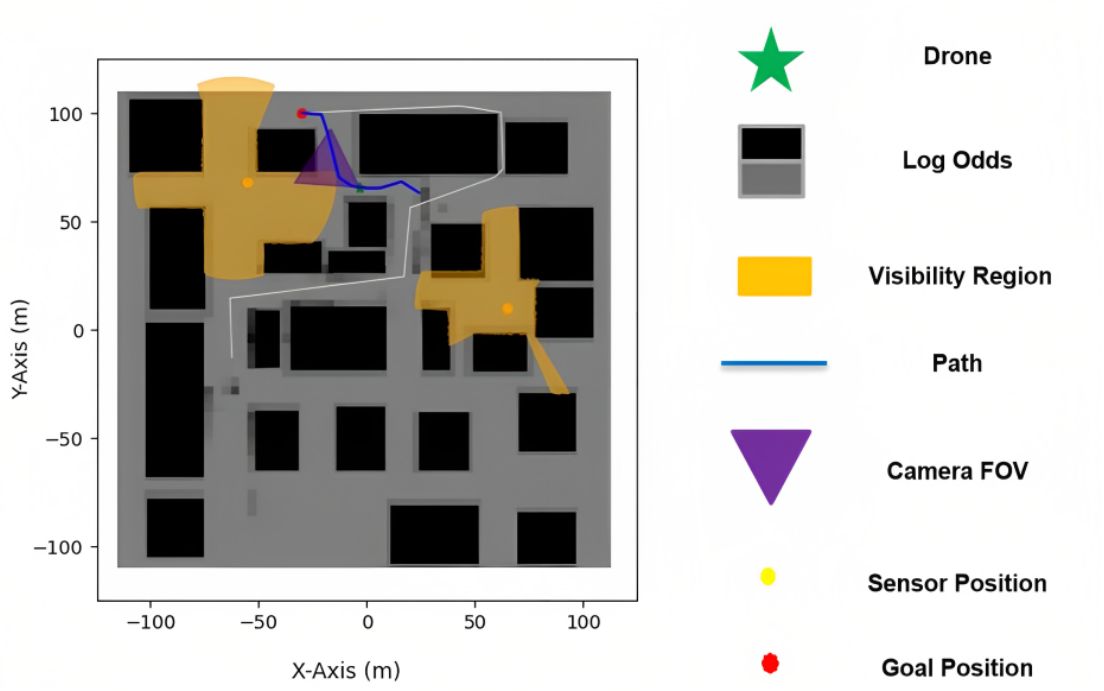


Figure 4.5: Online map of workspace with drone navigating on globally re-planned path

A sampling of 2000 nodes in the PRM has been able to provide a smooth

path unlike methods like RRT. This can be observed in the prior and the globally replanned paths. The PRM queried by the Dijkstra's algorithm for the global re-planning framework can be time intensive. Computation time can be improved by incorporating parallel processing in path computation [57], optimizing node sampling strategies [58] or iterating over roadmap density to reduce the number of nodes that still ensure desired smoothness of path.

CHAPTER 5

CONCLUSION

This thesis presents a visibility-constrained path planning framework for a quadcopter with an onboard camera. The Unreal Engine simulation environment has been critical to this effort due to it's photorealistic simulation and dynamics modeling of the quadcopter's motion.

The key concepts of visibility theory have been used to create sensor visibility regions in presence of the occlusions identified in prior map where the LOS visibility condition is satisfied. A PRM is used to generate a path in the prior map with the visibility regions and convex obstacles. The prior map is also divided into a 2-D probabilistic occupancy grid to update the position of previously unseen obstacles in the workspace during the online update phase.

Panoptic segmentation is used to identify obstacles in the input RGB frames. The inverse projection method is subsequently used to create 3-D point clouds that are mapped to the 2-D probability occupancy grid. A hybrid path replanning algorithm is discussed that performs local replanning for the drone to maneuver around newly identified obstacles. If the local replanning fails to find a local path in free space, the algorithm performs global replanning using the update PRM. The global replanning is expected to find a path as it uses the PRM and will be probabilistically complete. Monocular depth estimation was also used to implement the visibility-constrained path planning framework with only an RGB camera. The limitations of monodepth2 were identified that primarily included inadequate training data and a lack of domain transferability.

Future work in this area includes exploring prior map generation methods

that better approximate the shape of obstacles on the map. The quadtree data structure may also be used to improve the time complexity for updating the online map. The panoptic segmentation pipeline can also be augmented to label classes based on their trafficability, aiding the path modification methodology too. This framework can also be extended to a 3-D space with cubic voxels used in place of the 2-D square grids.

BIBLIOGRAPHY

[1] Zahraa Bassyouni and Imad H. Elhajj. Augmented reality meets artificial intelligence in robotics: A systematic review. *Frontiers in Robotics and AI*, 8, 2021.

[2] María Teresa Ballestar, Ángel Díaz-Chao, Jorge Sainz, and Joan Torrent-Sellens. Impact of robotics on manufacturing: A longitudinal machine learning perspective. *Technological Forecasting and Social Change*, 162:120348, 2021.

[3] Andrea Maria Zanchettin, Nicola Maria Ceriani, Paolo Rocco, Hao Ding, and Björn Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 13(2):882–893, 2016.

[4] Russell H. Taylor, Arianna Menciassi, Gabor Fichtinger, Paolo Fiorini, and Paolo Dario. *Medical Robotics and Computer-Integrated Surgery*, pages 1657–1684. Springer International Publishing, Cham, 2016.

[5] Kerstin Denecke and Claude R. Baudoin. A review of artificial intelligence and robotics in transformed health ecosystems. *Frontiers in Medicine*, 9, 2022.

[6] Chris Lytridis, Vassilis G. Kaburlasos, Theodore Pachidis, Michalis Manios, Eleni Vrochidou, Theofanis Kalampokas, and Stamatis Chatzistamatis. An overview of cooperative robotics in agriculture. *Agronomy*, 11(9), 2021.

[7] Arshia Khan and Yumna Anwar. Robots in healthcare: A survey. In Kohei Arai and Supriya Kapoor, editors, *Advances in Computer Vision*, pages 280–292, Cham, 2020. Springer International Publishing.

[8] Mark Wehde. Healthcare 4.0. *IEEE Engineering Management Review*, 47(3):24–28, 2019.

[9] Peter J. Wallin. Robotics in the food industry: an update. *Trends in Food Science Technology*, 8(6):193–198, 1997.

[10] Silvia Ferrari and Thomas A. Wettergren. *Information-Driven Planning and Control*. Cyber Physical Systems Series. MIT Press, illustrated edition, 2021.

[11] Ying Tan and Zhong yang Zheng. Research advance in swarm robotics. *Defence Technology*, 9(1):18–39, 2013.

[12] Jake Gemerek, Bo Fu, Yucheng Chen, Zeyu Liu, Min Zheng, David van Wijk, and Silvia Ferrari. Directional sensor planning for occlusion avoidance. *IEEE Transactions on Robotics*, 38(6):3713–3733, 2022.

[13] E. Birgersson, A. Howard, and G.S. Sukhatme. Towards stealthy behaviors. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, pages 1703–1708 vol.2, 2003.

[14] Mohamed S. Marzouqi and Ray A. Jarvis. New visibility-based path-planning approach for covert robotic navigation. *Robotica*, 24(6):759–773, 2006.

[15] Paul E. Rybski, Sascha A. Stoeter, Michael D. Erickson, Maria Gini, Dean F. Hougen, and Nikolaos Papanikolopoulos. A team of robotic agents for surveillance. In *Proceedings of the Fourth International Conference on Autonomous Agents*, AGENTS '00, page 9–16, New York, NY, USA, 2000. Association for Computing Machinery.

[16] S.A. Bortoff. Path planning for uavs. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, volume 1, pages 364–368 vol.1, 2000.

[17] Mohamed S. Marzouqi and Ray A. Jarvis. Covert path planning for autonomous robot navigation in known environments. 2003.

[18] Mohamed S. Marzouqi and Ray A. Jarvis. Accommodating uncertainty in covert and overt robot path planning. In *TENCON 2005 - 2005 IEEE Region 10 Conference*, pages 1–5, 2005.

[19] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[20] Mohamed S. Marzouqi and Ray A. Jarvis. New visibility-based path-planning approach for covert robotic navigation. *Robotica*, 24(6):759–773, 2006.

[21] Teddy Ort, Liam Paull, and Daniela Rus. Autonomous vehicle navigation

in rural environments without detailed prior maps. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2040–2047, 2018.

[22] H. Du, B. Hao, J. Zhao, J. Zhang, Q. Wang, and Q. Yuan. A path planning approach for mobile robots using short and safe q-learning. *PLoS ONE*, 17(9):e0275100, 2022.

[23] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[24] Zhongli Wang and Guohui Tian. Hybrid offline and online task planning for service robot using object-level semantic map and probabilistic inference. *Information Sciences*, 593:78–98, 2022.

[25] José Ricardo Sánchez-Ibáñez, Carlos J. Pérez-del Pulgar, and Alfonso García-Cerezo. Path planning for autonomous mobile robots: A review. *Sensors*, 21(23), 2021.

[26] Janelle Resch, Ireneusz, Ocelewski, Judy Ehrentraut, and Michael Barnett-Cowan. Gamified automation in immersive media for education and research, 2018.

[27] Chaitya Vohera, Heet Chheda, Dhruveel Chouhan, Ayush Desai, and Vijal Jain. Game engine architecture and comparative study of different game engines. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2021.

[28] Xiaowei Chen, Meihong Wang, and Qingfeng Wu. Research and development of virtual reality game based on unreal engine 4. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1388–1393, 2017.

[29] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, 2017.

[30] Bruno J. Souza, Lucas C. de Assis, Dominik Rößle, Roberto Z. Freire, Daniel Cremers, Torsten Schön, and Munir Georges. Aimotion challenge results: a framework for airsim autonomous vehicles and motion replication. In *2022 2nd International Conference on Computers and Automation (CompAuto)*, pages 42–47, 2022.

[31] Manav Khambhayata. A comparative analysis of carla and airsim simulators: Investigating implementation challenges in autonomous driving. May 2023. Available at SSRN: https://ssrn.com/abstract=4477130 or http://dx.doi.org/10.2139/ssrn.4477130.

[32] Hongjun Li, Miguel Barão, and Luís Rato. Gaussian random field-based log odds occupancy mapping. In *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–4, 2018.

[33] Jugesh Sundram, Duong Van Nguyen, Gim Song Soh, Roland Bouffanais, and Kristin Wood. Development of a miniature robot for multi-robot occupancy grid mapping. In *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 414–419, 2018.

[34] L. Lulu and A. Elnagar. A comparative study between visibility-based roadmap path planning algorithms. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3263–3268, 2005.

[35] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[36] Bhaavin K. Jogeshwar and K. Lochan. Algorithms for path planning on mobile robots. *IFAC-PapersOnLine*, 55(1):94–100, 2022. 7th International Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2022.

[37] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation, 2019.

[38] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 2020.

[39] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[40] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[41] Edward H. Adelson. On seeing stuff: the perception of materials by humans and machines. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, *Human Vision and Electronic Imaging VI*, volume 4299, pages 1 – 12. International Society for Optics and Photonics, SPIE, 2001.

[42] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[43] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xiangyu Zhang, Xinyu Zhou, Erjin Zhou, and Jian Sun. Learning delicate local representations for multi-person pose estimation, 2020.

[44] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks, 2019.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[46] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[47] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation, 2019.

[48] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[49] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. A comparison of volumetric information gain metrics for active 3d object reconstruction. *Autonomous Robots*, 42(2):197–208, 2018.

[50] Qiyu Sun, Gary G. Yen, Yang Tang, and Chaoqiang Zhao. Learn to adapt for monocular depth estimation, 2022.

[51] Armin Masoumian, Hatem A. Rashwan, Julián Cristiano, M. Salman Asif, and Domenec Puig. Monocular depth estimation using deep learning: A review. *Sensors*, 22(14), 2022.

[52] Nicolas Mellado, Matteo Dellepiane, and Roberto Scopigno. Relative scale estimation and 3d registration of multi-modal geometry using growing least squares. *IEEE Transactions on Visualization and Computer Graphics*, 22(9):2160–2173, 2016.

[53] Tobias Gindele, Sebastian Brechtel, Joachim Schroder, and Rudiger Dillmann. Bayesian occupancy grid filter for dynamic environments using prior map knowledge. In *2009 IEEE Intelligent Vehicles Symposium*, pages 669–676, 2009.

[54] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404 vol.2, 1991.

[55] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317 vol.4, 1994.

[56] Aboelmaged Noureldin, Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera. Global and local path planning study in a ros-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, 2018:6392697, 2018.

[57] Soheil Younesi, Bahman Ahmadi, Oguzhan Ceylan, and Aydogan Ozdemir. Optimum parallel processing schemes to improve the computation speed for renewable energy allocation and sizing problems. *Energies*, 15(24), 2022.

[58] Qiang Li, Yan Xu, Shizhong Bu, and Jian Yang. Smart vehicle path planning based on modified prm algorithm. *Sensors*, 22(17):6581, Aug 2022.