A CELL DECOMPOSITION APPROACH TO AUTONOMOUS PATH PLANNING FOR DIRECTIONAL MOBILE SENSORS

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

Zeyu Liu

May 2018

ABSTRACT

A methodology based on integer programming and cell decomposition is developed for planning the path of UGVs equipped with directional sensors used to classify multiple targets in an obstacle-populated environment. While it is desirable to solve this problem in minimum time, the non-completeness of the connectivity graph and the classification objectives do not allow for a Traveling Salesman Problem (TSP) solution. Moreover, the TSP is known to be NP hard. Therefore, this thesis presents an approach for decomposing the UGV workspace based on the directional sensor FOV, line-of-sight visibility and obstacle map. By this approach, a connectivity graph with observation cells can be obtained and an optimal path can be computed via integer programming. Simulations conducted in Webots, a professional robot simulator that supports accurate simulation of rigid body dynamics and sensors with computer vision capability, demonstrate the effectiveness of this approach compared to the ”nearest neighbor” methods and classical TSP formulations.

BIOGRAPHICAL SKETCH

Zeyu Liu is an M.S. student in the Laboratory for Intelligent Systems and Controls (LISC) at Cornell University. He received the B.S. degree in Mechanical Engineering from Tongji University and Politecnico di Milano (magna cum laude). His research interests include probabilistic reasoning, optimal control, computer vision and machine learning, with a focus in unmanned ground vehicles.

ACKNOWLEDGEMENTS

I would like to sincerely thank Prof. Ferrari who has been providing me with great guidance and advice in my research at Cornell University. It is her kindness and assistance that impacted my life the most. I also want to thank Prof. Knepper for his suggestion in my defense. There are many others I would like to acknowledge. Prof. Kleinberg and Prof. Schalekamp discussed with me about the Traveling Salesman Problem (TSP). Dr. Zhu and Dr. Fu both provided me with much feedback in my discussion with them. Dr. Liu gave me much support in modifying the writing of my thesis. Last but not least, I would like to thank all other members in LISC, Jake, Taylor, Julian, Min, Yucheng, Shi, Jane, Hengye and Quanxing, for their feedback in the lab meetings and my practice presentation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND BACKGROUND

Sensor planning is the problem of determining a strategy to support a sensing objective. With the proliferation of sensors installed on mobile robotic platforms, such as autonomous vehicles, one of the challenging problems is determining the best path for obtaining sensor measurements [19, 1, 29]. In sensor planning, the robot motion is planned to best support the sensing objective, rather than simply using sensor measurements to support robot motion [9]. This thesis addresses the problem of path planning for an Unmanned Ground Vehicle (UGV) equipped with a directional sensor used to classify targets in an obstacle-populated environment in minimum time. This problem is a variant of the treasure hunt problem, in which the shortest path and measurement location of a robotic sensor is planned in order to successfully classify the most valuable targets in an obstacle-populated workspace. Potential application of the minimum time problem addressed in this thesis are robotic mine hunting, cleaning and monitoring of urban environments [5] and search and rescue [51, 35, 34]. The minimum time problem is also an excellent benchmark problem for satisficing decision making. Satisficing refers to fast heuristic decision making that prioritizes some information while ignoring others under uncertainty and time pressure [55, 56, 57, 16]. An example of satisficing task involves human or animal subjects tasked with the problem of classifying targets in a complex workspace while under time pressure [41, 42].

Directional sensor path planning belongs to the class of geometric sensor planning problems that consider the geometry and position of the targets and that of the sensor's field of view (FOV) [17]. Then, using methods such as cell

decomposition, sensor path planning can take into account the motion and geometry of the sensor's platform and FOV, as well as the geometry of all the targets and obstacles. Subsequently, a connectivity graph can be obtained by decomposing the free configuration space into void and observation cells. Void cells represent the configurations in which the sensor cannot obtain any measurements. Observation cells represent configurations in which the sensor can obtain measurements from one or more targets [6]. The resulting connectivity is typically a non-complete graph in which only a small portion of node pairs are connected by an edge. Therefore, the number of edges is relatively small compared to that in a complete graph with same number of nodes. A path planning algorithm that can take advantage of this property is much more efficient and has a shorter running time.

One approach to guaranteeing that a robotic sensor visits all targets is to solve a coverage path planning problem [10]. Popular approaches include lawn mower with optimal line-sweep [21], complete coverage [1, 9], random [1], grid [33] and genetic algorithm [23]. A major drawback of coverage path planning is that the distance traveled is not minimized, resulting in very timely and costly operations. Therefore, an information potential method (IPM) was developed in [36], which defines a potential function from conditional mutual information and generates paths of maximum information value. In IPM, the use of target information value and geometry greatly improves the sensor performance, but target coverage is not guaranteed and the directionality of sensor is not considered. Information roadmap method (IRM) has also been proposed for considering the targets' expected information value in order to generate a roadmap with a high density of high-information-value milestones while capturing the connectivity of the workspace [64]. However, existing methods cannot provide

minimum time paths for covering all of the targets.

Planning the optimal sensor path is intrinsically hard, because the optimization problem is generally non-convex. However, the problem can be transformed into an integer linear programming problem involving discrete decisions about edges under linear constraints [3, 61]. Integer programming has been applied to robot path planning, including aircraft collision avoidance [49] and autonomous underwater vehicle for ocean measurements [63]. The availability of a linear programming routine where constraints can be iteratively added makes the method tractable for problems with large number of constraints [37]. Although integer programming technique can be used for solving traveling salesman problem (TSP), the minimum time sensor path planning problem is by nature very different from TSP, as will be shown in Chapter 5.

In this thesis, the benchmark problem of minimum time sensor path planning is solved via integer programming by transforming the sensing objective into linear constraints based on the connectivity graph constructed by cell decomposition. A closed-form definition of directional C-Target is also proposed together with an algorithm that calculates all C-Targets with linear computational complexity.

In Chapter 3, computer vision algorithms used for classification based on feature extraction from sensor image is discussed. The definition of directional C-Target and an approximate cell decomposition approach are developed in Chapter 4. The proposed approach developed in Chapter 5 incorporates the special properties of the sensing objective and the connectivity graph presented in Chapter 4. As shown in Chapter 6, the proposed approach fulfills the sensing objective requiring less time to classify all targets when compared to the near-

est neighbor method and classical TSP solutions. The proposed method can be applied to different robot parameters of translation and rotation, and guarantee obstacle avoidance as well as non-overpass constraints [53] of platforms that must avoid driving over targets.

CHAPTER 2

PROBLEM FORMULATION AND ASSUMPTIONS

This thesis considers the problem of planning the path of a directional sensor onboard unmanned ground vehicle (UGV) deployed to classify a set of targets in an obstacle-populated workspace in minimum time. The workspace denoted by $\mathcal{W} \subset \mathbb{R}^2$ is assumed here to be a compact subset of a Euclidean space, populated with r fixed targets denoted by $\mathcal{T}_1, ..., \mathcal{T}_r$. The probabilistic model of sensor measurements and classification is learned from data and expert knowledge using a Bayesian Network (BN). BN represents a joint probability mass function (PMF) by a directed acyclic graph (DAG), $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. The node set $\mathcal{N}$ is composed of M discrete feature random variables $X_1, ..., X_M$ and a categorical random variable Y. Each feature random variable $X_j \in \mathcal{N}$ is associated with a finite set of mutually exclusive states $\mathcal{X}_j = \{x_1, ..., x_j\}$, and the states of the categorial random variable Y is $\mathcal{Y} = \{y_1, y_2\}$. Each node in $\mathcal{N}$ is associated with a conditional probability table (CPT) of BN parameters. The set of edges $\mathcal{E}$ represents conditional dependencies between nodes in $\mathcal{N}$, and expresses the joint PMF as

$$p(\mathcal{N}) = p(X_1, ..., X_M, Y) = p(Y|pa(Y)) \prod_{j=1}^{M} p(X_j|pa(X_j)) \tag{2.1}$$

where $pa(X_j)$ is the parent set of X_j , such that $\forall X_i \in pa(X_j)$ there exists a directed arc $(i, j) \in \mathcal{E}$ and similarly for $pa(Y)$.

$\mathcal{W}$ is also populated with n fixed obstacles $\mathcal{B}_1, ..., \mathcal{B}_n$ whose geometry and positions are known a priori. The UGV's geometry is described by a rigid object $\mathcal{A} \subset \mathbb{R}^2$ that is a compact subset of $\mathcal{W}$. The UGV is equipped with a directional

sensor (e.g. a camera) with a field of view (FOV) denoted by $\mathcal{S}$. A configuration vector $\mathbf{q} = [x, y, \theta]^T \in \mathcal{C}$ specifies the position of the UGV's geometrical center and orientation of the UGV with respect to a fixed Cartesian frame $\mathcal{F}_W$ embedded in $\mathcal{W}$ with origin O_W, where configuration space $\mathcal{C} \subset \mathcal{W} \times (0, 2\pi]$ denotes the space of all possible values of the configuration vector $\mathbf{q}$. $\mathbf{q}$ also specifies a moving Cartesian frame $\mathcal{F}_A$ embedded in $\mathcal{A}$ with origin O_A defined at the position of UGV. The UGV is assumed to obey the a unicycle robot kinematics. The control input is $\mathbf{u} = [v\ \omega]^T \in \mathcal{U} = \{v, \omega \mid 0 \leq v \leq v_m, 0 \leq \omega \leq \omega_m\}$, where $v_m, \omega_m \in \mathbb{R}^+$ are the maximum permissible velocity and angular velocity respectively. The unicycle model is

$$\dot{\mathbf{q}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} cos\theta(t) & 0 \\ sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \mathbf{g}[\mathbf{q}(t)]\mathbf{u}(t) \tag{2.2}$$

The sensor is fixed on the UGV, and therefore every point in $\mathcal{S}$ is fixed with respect to $\mathcal{F}_A$. Under this assumption $\mathbf{q}$ also specifies both $\mathcal{A}$ and $\mathcal{S}$. A C-obstacle with respect to obstacle $\mathcal{B}_i$ is defined as a subset of $\mathcal{C}$ that causes collisions with $\mathcal{B}_i$, i.e., $\mathcal{CB}_i = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{A}(\mathbf{q}) \cap \mathcal{B}_i \neq \emptyset\}$, where $\mathcal{A}(\mathbf{q})$ denotes the subset of $\mathcal{W}$ occupied by $\mathcal{A}$ when the UGV is at configuration $\mathbf{q}$. The C-obstacle region $\mathcal{CB}$ is defined as $\cup_i \mathcal{CB}_i$, and the free configuration $\mathcal{C}_{free}$ is defined as $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{CB}$. Planning a collision-free path for the UGV is equivalent to find a trajectory in $\mathcal{C}_{free}$.

A directional sensor is influenced by occlusions caused by obstacles in its

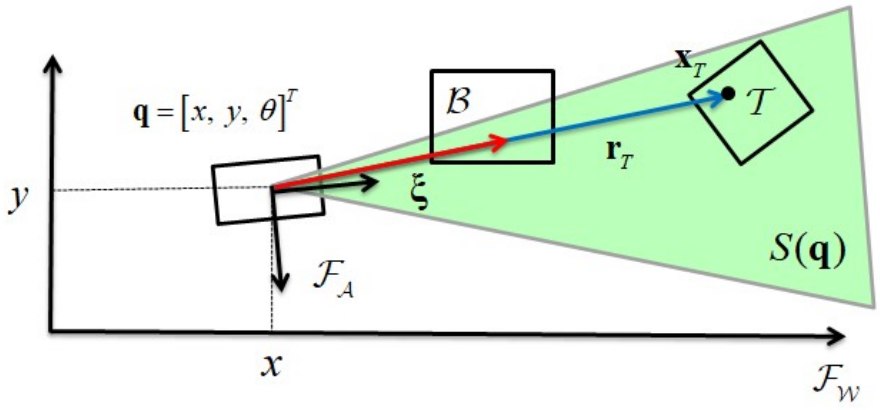

Figure 2.1: Line of sight visibility. Sensor FOV $\mathcal{S}$ is shown by a green triangle. In this figure, the point of interest $\mathbf{x}_T$ is occluded because there exists a $\boldsymbol{\xi}$ that fails the line-of-sight test.

line-of-sight (LoS), in front of the target of interest $\mathcal{T}_i$. The FOV of a sensor with dynamic state $\mathbf{q}$ can be modeled as a compact subset $\mathcal{S}(\mathbf{q})$ of $\mathcal{W}$ occupied by $\mathcal{S}$ when the UGV is at configuration $\mathbf{q}$. Let $\mathbf{x}_T$ be the position of an interest point in target $\mathcal{T}_i$. The coordinate of the point of interest in $\mathcal{F}_A$ is $\mathbf{r}_T = \mathbf{x}_T - \mathbf{q}$. $\mathbf{x}_T$ is in the LoS of the sensor at $\mathbf{q}$ if there are no points in the obstacle region $\mathcal{B}$ that are co-directional with $\mathbf{r}_T$ and closer to $\mathbf{q}$ than $\mathbf{x}_T$, or

$$\nexists \boldsymbol{\xi} \in \mathcal{B} \text{ s.t. } \boldsymbol{\xi} \cdot \mathbf{r}_T = \|\boldsymbol{\xi}\|\|\mathbf{r}_T\| \text{ and } \|\boldsymbol{\xi}\| < \|\mathbf{r}_T\| \tag{2.3}$$

where $\boldsymbol{\xi}$ is defined with respect to $\mathcal{F}_A$, and it is assumed that $\mathcal{T}_i \cap \mathcal{B} = \emptyset$. Target $\mathcal{T}_i$ is visible to the sensor if $\mathcal{S}(\mathbf{q}) \cap \mathcal{T}_i \neq \emptyset\}$ and $\mathcal{T}_i$ is in the LoS of the sensor. C-target of target $\mathcal{T}_i$ is defined as a subset of $\mathcal{C}$ such that $\mathcal{CT}_i = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{S}(\mathbf{q}) \cap \mathcal{T}_i \neq \emptyset\}$ and the LoS test is satisfied for all obstacles. Therefore, the path planning problem for visiting all targets in minimum time can be redefined as:

"Find a trajectory in $\mathcal{C}_{free}$ such that for every C-target $\mathcal{CT}_i$, there exist at least one configuration along the trajectory that belongs to $\mathcal{CT}_i$ for which $\mathcal{T}_i$ is in the LoS of the robot, and the total time is minimized subject to UGV's constraints".

The minimum time sensor path planning problem is solved via integer programming by transforming the sensing objective into linear constraints based on a connectivity graph constructed by cell decomposition. The cell decomposition approach utilizes the closed form definition of directional C-target $\mathcal{CT}$ that will be discussed in Chapter 4.

2.1 Minimum-time optimal path planning

The minimum time path planning problem is simplified by means of the following key assumptions are made. The UGV (2.2) is assumed to always move and rotate at maximum admissible velocity v_m and angular velocity ω_m, which indicates that acceleration and deceleration are ignored. The location of the targets and obstacles are known from prior knowledge, which allows the UGV to plan a path to classify all targets offline. Also, the initial configuration $\mathbf{q}_0$ of the UGV is assumed to be fixed and known.

Because the on-board sensor is directional (e.g. camera), C-Target has a more complicated shape compared to the case with omnidirectional sensor. The aspect angle θ is taken into consideration in sensor path planning which is carried out in 3-dimensional configuration space.

Denote the start and final time as t_0 and t_f respectively. Denote t_m as the time required to go from $\mathbf{q}_0$ along the path to the finish point, which is not fixed and up to the algorithm to optimize. The total time can therefore be decomposed

into two terms: observation time t_o and motion time t_m.

$$t_f - t_0 = t_o + t_m = \sum_{i=1}^{r} t_i + t_m \tag{2.4}$$

where r is the total number of targets, t_i is the time required to classify the i-th target. Under the assumption that the UGV always moves and rotates at fixed velocity v_m, ω_m, the moving time t_m is transformed into a combination of translation distance and rotation angle along the path segments.

The target classification problem assumes that target features and classification can be modeled by a Bayesian Network joint probability mass function (PMF),

$$P(Y|X_1, ..., X_l) = \sum_{X_1} ... \sum_{X_l} P(Y|X_M) \prod_{j=l}^{M-1} P(X_{j+1}|X_j) \tag{2.5}$$

where $X_1, ..., X_l$ are discrete feature random variables and Y is a categorical random variable. The BN model used in this thesis is shown in Figure 2.2. The discrete feature random variables X_1, X_2 and X_3 indicates "shape", "color" and "texture" respectively. Y is a categorical random variable that indicates whether the object is a treasure or not. Each random variable is associated with a finite set of mutually exclusive states,

$$
\begin{aligned}
\mathcal{X}_1 &= \{\text{"Sphere"}, \text{"Box"}\} \\
\mathcal{X}_2 &= \{\text{"Green"}, \text{"Orange"}, \text{"Brown"}, \text{"Black"}\} \\
\mathcal{X}_3 &= \{\text{"Apple"}, \text{"Watermelon"}, \text{"Orange"}, \text{"Basketball"}, \\
&\qquad \text{"Cardboard Box"}, \text{"Wooden Box"}, \text{"Computer"}, \text{"Book"}\} \\
\mathcal{Y} &= \{\text{"Treasure"}, \text{"Non-Treasure"}\}
\end{aligned} \tag{2.6}
$$

The features of the object are revealed in sequential order. Each time the UGV reveals the value of a new feature, the posterior distribution is updated and so is the confidence level.

The confidence level of target classification reflects how much confidence we have when classifying a target. It is defined as an function of Y given the value of l ($l \leq M$) features:

$$
CL(Y, X_1, ..., X_l) = \max P(y|X_1, ..., X_l) \tag{2.7}
$$

To successfully classify a target, the confidence level must reach a threshold $\tau \in (0, 1]$. For this problem the threshold is set to a fixed value for all targets. After reviewing certain amount of features, the posterior distribution become much less uniform than the prior distribution, and the confidence level is therefore increased. Denote the minimum number of features required to reach the threshold for classifying target $\mathcal{T}_i$ as ν_i, i.e., $CL(Y, X_1, ..., X_{\nu_i}) > \tau$. Therefore, the time t_i required to classify $\mathcal{T}_i$ is defined as:

$$
t_i = \sum_{q=1}^{\nu_i} t_{iq} \tag{2.8}
$$

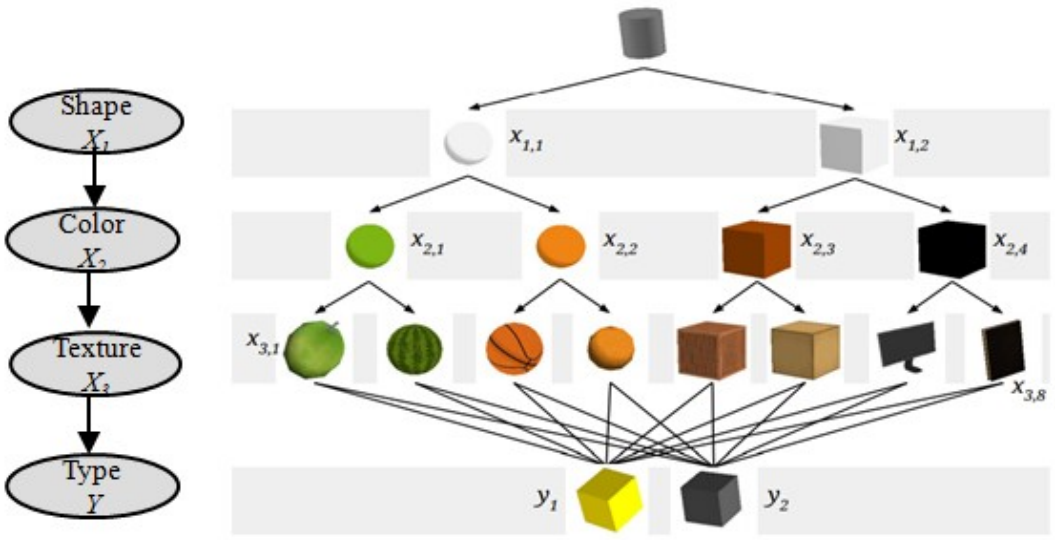

Figure 2.2: Bayesian Network of feature discrete random variables X_1 (shape), X_2 (color), X_3 (texture) and Y (type). The values of X_1, X_2 and X_3 are displayed on the right. The value of the categorical random variable Y is $\{\text{Treasure}, \text{Non-Treasure}\}$.

where t_{iq} is the time required to reveal the q-th feature of target i.

Given the target initial feature level and the CPT under Bayesian Network model for targets, and assuming that target cue level does not change unless the UGV chooses to, t_i will be constant for any target. Note that the objective is to classify all targets, so the sum of time of revealing cues over all targets $t_o = \sum_{i=1}^{r} t_i$ is a constant. Therefore, minimizing the total time T is equivalent to minimizing t_m. The objective is transformed to finding the shortest path in the configuration space that visits all targets while avoiding all obstacles. The problem is therefore restated as: Given the start configuration $\mathbf{q}_0$ of the UGV, determine $\mathbf{u}(t)$ $t \in T = [t_0, t_f]$ to minimize: $J(t_f) = t_f - t_0$, such that

$$\forall \mathcal{T}_i,\ \exists t_i \in T \text{ s.t. } \mathbf{q}(t_i) \in \mathcal{CT}_i \tag{2.9}$$

CHAPTER 3

FEATURE EXTRACTION FROM IMAGE

In the setting of the Satisficing task, human subjects use a camera onboard UGV to obtain measurements and classify targets. Each target is associated with three features: shape, color and texture, revealed in sequential order. In this chapter, three different approaches are discussed for the three features mentioned above.

3.1 Shape Detection using Circular Hough Transform (CHT)

The circular Hough transform (CHT) enables the binary classification of the sphere and box based on shape. The CHT has been proven to be a robust method for circle detection even under noisy circumstances [50]. The CHT was first proposed by Duda et al. [13] after the related 1962 patent of Paul Hough [20]. The CHT transforms points from pixel space to Hough space. The procedure of CHT has three steps [39]: first the edges in the image are found using Canny algorithm [7]; second, for each edge point a circle is drawn with radius r, and increments all coordinates that the perimeter passes through in the accumulator [28, 50]; Third, A circle with radius is found if a point in parameter space is a local maximum that exceeds a threshold. If no circle exists the object is assumed to be a box.

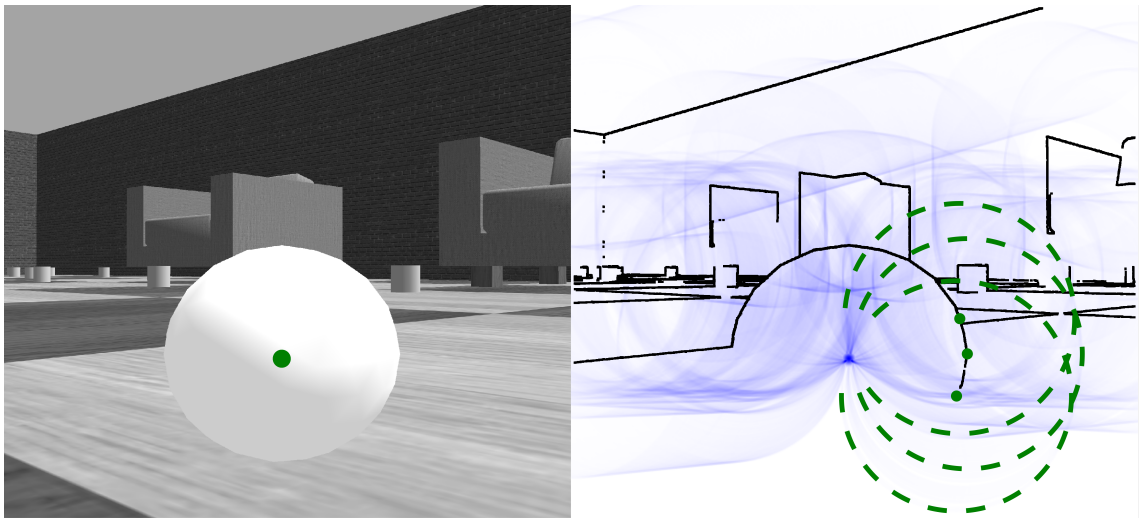

Figure 3.1: Demonstration of shape detection using circular Hough transform. The highest value in the accumulator shown in blue marks the center of a potential sphere.

3.2 Color Detection using HSV

Individual color appearance parameters provide input to image processing algorithms. A color's hue (denote by H) corresponds to the color of the rainbow it's closest to (e.g. pink and brown have red and orange hues respectively). The mean HSV (hue, saturation and value) are calculated for each image, and hue is used for color detection as it is not influenced by lighting conditions as RGB. RGB values are converted to hue and intensity values as [26]

$$H = \begin{cases} \arccos \frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}} & B \leq G \\ 2\pi - \arccos \frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}} & B > G \end{cases} \tag{3.1}$$

$$S = \frac{max(R,G,B)-min(R,G,B)}{max(R,G,B)} \tag{3.2}$$

$$V = \frac{max(R,G,B)}{255} \tag{3.3}$$

where R, G and B are the values of three channels of RGB color model. Hue information alone enables the classification of cue X_2. Shape and texture are the distinguishing features of cues X_1 and X_3 respectively.

3.3 Texture Detection using LBP

A linear support vector machine (SVM)[12, 8] trained with linear binary patterns (LBP) features differentiates target texture. The LBP operator [43] has been proven to be an effective and computational efficiency texture descriptor because of its invariance to monotonic gray level changes [60, 44]. LBP features exhibit invariance to monotonic gray level changes [60, 44]. Let p represent one of P points evenly spaced around a circle centered at c (i.e. p are the vertices of a regular polygon centered at c). Let g_p and g_c denote the grayscale value of points p and point c respectively. A LBP feature is a P bit binary number whose p^{th} bit is a 0 if g_p is less than g_c, and a 1 otherwise. There exist 2^P possible LBP features. Uniform pattern LBP reduces this total feature number by assigning the same value to every binary string that is not a uniform pattern. A string is a uniform pattern if it contains at most two bitwise transitions. E.g. 101011 is not a uniform pattern because it contains 4 transitions, while 100001 is a uniform pattern because it contains 2 transitions. The rotation invariant uniform pattern LBP simplifies LBP further by lumping uniform patterns with the same sum together. Uniform LBP is calculated as [45]

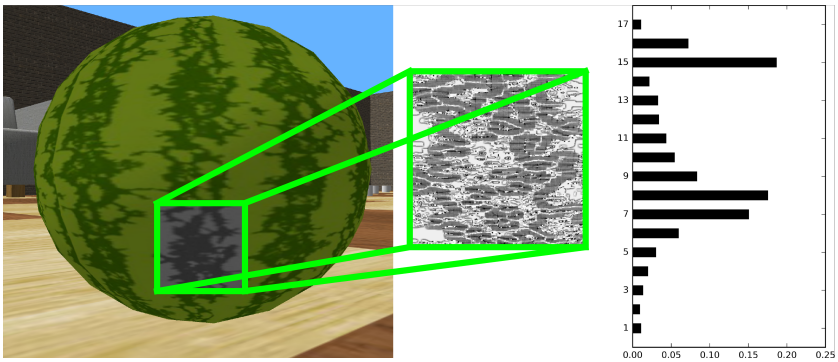

Figure 3.2: Histogram of LBP of a watermelon. The vector of histogram is used to train and test the SVM for classification.

$$LBP^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & \text{if } (LBP_{P,R}) \leq 2 \\ P+1 & \text{otherwise} \end{cases} \tag{3.4}$$

$$U(LBP_{P,R}) = \begin{array}{l} |s(g_{P-1} - g_c) - s(g_0 - g_c)| \\ +\sum_{p=1}^{P-1} |s(g_p - g_c) - s(g_{p-1} - g_c)| \end{array} . \tag{3.5}$$

where superscript ^{riu2} reflects the use of rotation invariant uniform patterns that have U value of at most 2, g_c is the gray value of the central pixel , g_p is the value of its neighbors, P is the total number of involved neighbors, and R is the radius of the neighborhood. By using the uniform pattern, the LBP histogram has a separate bin for every uniform pattern and all nonuniform patterns are assigned to a single bin [2]. A normalized histogram of feature vectors from a textured region comprises a single input to an SVM classifier. The training set is generated by taking images from different angles ranging from 0 to 360 degrees with a step of 5 degrees.

CHAPTER 4

PATH PLANNING METHODOLOGY

4.1 Directional C-Targets

One of the most important objective for sensor path planning is measuring the targets. Along the path of the sensor platform, there exist configurations $\mathbf{q} = [x, y, \theta]^T$ such that the targets fall into the sensor's FOV $\mathcal{S}(\mathbf{q})$. The subsets of $\mathcal{C}$ where the sensor can collect measurements of targets are denoted as C-Targets, defined as $\mathcal{CT}_i = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{S}(\mathbf{q}) \cap \mathcal{T}_i \neq \emptyset\}$ for target $\mathcal{T}_i$. Therefore, for any target $\mathcal{T}_i$, there exists at lease one configuration in the path that belongs to the corresponding C-Target $\mathcal{CT}_i$.

Directional sensor is a type of sensor that obtains measurements only for a bounded subset of all possible aspect angles. The following assumptions are made regarding the FOV of the sensor. First, the sensor is assumed to obey the LoS visibility model [46] [52] [58]. It only measures a point if the straight line segment connecting the sensor to this point does not intersect with any obstacle. In other words, obstacles occlude area from the sensor's FOV. second, the FOV is assumed to be an isosceles triangle with leg length l_S and vertex angle ϕ. The lower bound of the sensor's range is assumed to be zero. Third, the sensor is assumed to be fixed with respect to the UGV. Under these assumptions, the FOV $\mathcal{S}(\mathbf{q})$ of the sensor is characterized by two vectors $\mathbf{v}_1$, $\mathbf{v}_2$ defined as

$$\mathbf{v}_1 = [l_S cos(\theta - \frac{\phi}{2}), l_S sin(\theta - \frac{\phi}{2})] \quad \mathbf{v}_2 = [l_S cos(\theta + \frac{\phi}{2}), l_S sin(\theta + \frac{\phi}{2})] \tag{4.1}$$

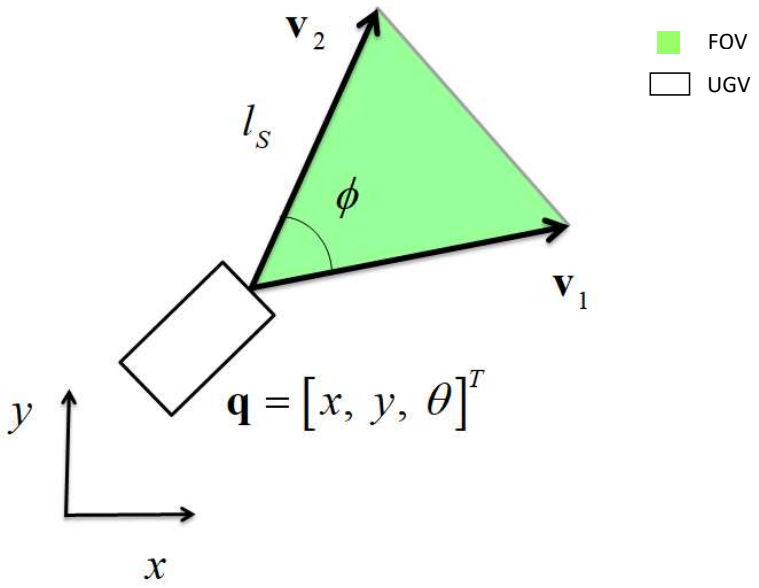

Figure 4.1: Example of $\mathbf{v}_1$ and $\mathbf{v}_2$, given a robot configuration $\mathbf{q} = [x, y, \theta]^T$. The FOV is characterised by $\mathbf{v}_1$ and $\mathbf{v}_2$.

The vectors $\mathbf{v}_1$, $\mathbf{v}_2$ are illustrated by Figure 4.1.

The C-Target of $\mathcal{T}_i$ at robot's orientation θ is denoted as $\mathcal{CT}_i^\theta$. A closed-form definition of $\mathcal{CT}_i^\theta$ can be derived from $\mathbf{v}_1$ and $\mathbf{v}_2$, which can simplify the calculation and reduce computational complexity. Target $\mathcal{T}_i$ is assumed to be a convex polygon. Denote the set of vertices of $\mathcal{T}_i$ as $\mathcal{V}_i$. Define $(\mathcal{V}_i)_{-\mathbf{v}_1}$ as the a set of vertices composed of every element of $\mathcal{V}_i$ translated by vector $-\mathbf{v}_1$. The set of points $S = \mathcal{V}_i \bigcup (\mathcal{V}_i)_{-\mathbf{v}_1} \bigcup (\mathcal{V}_i)_{-\mathbf{v}_2}$ forms the boundary configuration of $\mathcal{CT}_i^\theta$, where the $\mathcal{S}(\mathbf{q}) \cap \mathcal{T}_i$ is a vertex of $\mathcal{T}_i$. Figure 4.2 shows the elements of $\mathcal{S}$ with $\theta = \frac{\pi}{2}$. C-Target $\mathcal{CT}_i^\theta$ is defined as

$$\mathcal{CT}_i^\theta = Conv(S) \backslash \mathcal{T}_i \tag{4.2}$$

where $Conv(S)$ is the *convex hull* of S:

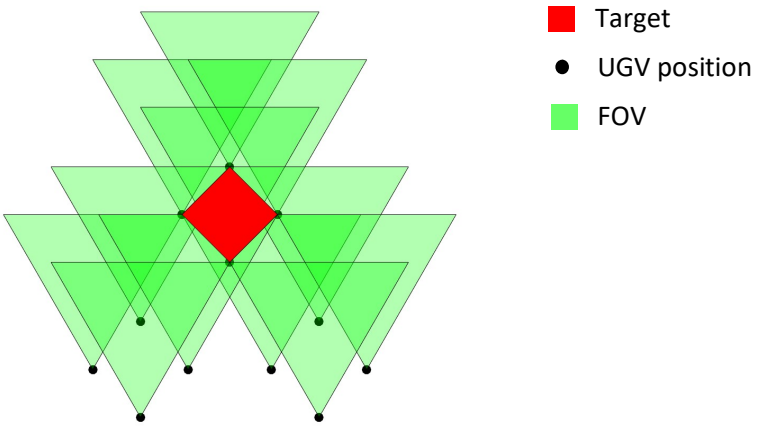

Figure 4.2: An example of $S = \mathcal{V}_i \bigcup (\mathcal{V}_i)_{-\mathbf{v}_1} \bigcup (\mathcal{V}_i)_{-\mathbf{v}_2}$ with $\theta = \frac{\pi}{2}$. Elements of S are shown in thick black dot. FOVs are shown with green isosceles triangles and target is a red polygon. Elements of S indicate the configurations of UGV where $\mathcal{S}(\mathbf{q}) \cap \mathcal{T}_i$ is a vertex of $\mathcal{T}_i$.

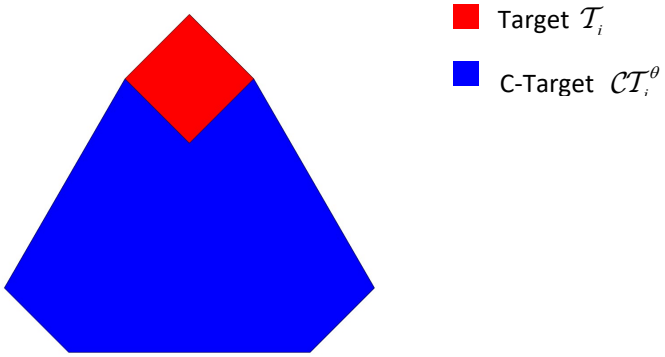

Figure 4.3: C-Target $\mathcal{CT}_i^{\theta}$ is shown in blue. $\mathcal{T}_i$ in red is excluded, as the sensor should not collide with the target.

$$Conv(S) = \{\sum_{z=1}^{|S|} w_z x_z | w_z \geq 0, \sum_{z=1}^{|S|} w_z = 1\} \tag{4.3}$$

4.1.1 C-Target with the presence of Obstacles

In an obstacle-populated workspace, additional constraint must be considered for the definition of directional C-Target. As the sensor is assumed to be LoS sensor, obstacles in the line of sight may cause occlusions to the FOV. $\mathcal{CT}_i^\theta$ should not include the regions in which the sensor cannot obtain measurement from $\mathcal{T}_i$ due to occlusion. In this section, a definition of C-Target with the presence of obstacles is formulated by excluding the occluded area from previously defined C-Target.

Assuming all targets and obstacles are convex polygons. Let the vertices of target $\mathcal{T}_i$ and obstacle $\mathcal{B}_j$ be denoted by $\mathcal{V}_i$ and $\mathcal{V}_j$ respectively. Let $P \in \mathcal{V}_i$ be an arbitrary vertex of $\mathcal{T}_i$. V_l and V_m are chosen as the pair of vertices of $\mathcal{B}_j$ such that the angle $\alpha =< \overline{PV_l}, \overline{PV_m} >$ is maximized. The boundary vectors $\hat{\mathbf{k}}_1$ and $\hat{\mathbf{k}}_2$ are defined as

$$\begin{aligned} \hat{\mathbf{k}}_1 &= \frac{\overline{PV_l}}{\|\overline{PV_l}\|} \\ \hat{\mathbf{k}}_2 &= \frac{\overline{PV_m}}{\|\overline{PV_m}\|} \end{aligned} \tag{4.4}$$

The illustration of the boundary vectors is shown in Figure 4.4 (a). The coverage cone of point P with respect to obstacle $\mathcal{B}_j$ is defined as

$$\mathcal{K}_P(\mathcal{B}_j) = cone(\hat{\mathbf{k}}_1, \hat{\mathbf{k}}_2) = \{\, \mathbf{x} \mid \mathbf{x} = \mathbf{x}_P + \hat{\mathbf{k}}_1 + c_2\hat{\mathbf{k}}_2, c_1, c_2 \geq 0\} \tag{4.5}$$

where $\mathbf{x}_p$ is the coordinate of point P. The shadow region of point P with respect to obstacle $\mathcal{B}_j$ can therefore be defined as

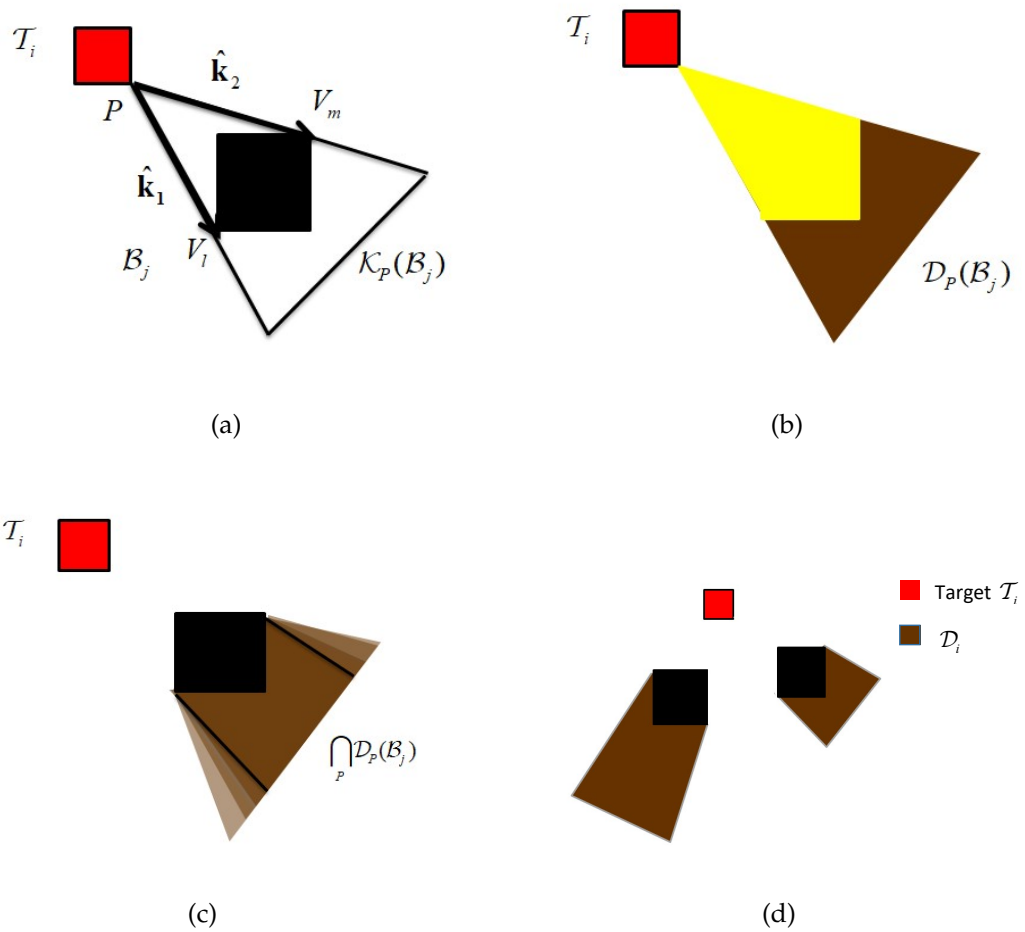

Figure 4.4: Examples of the shadow region. Figure (a) illustrates the coverage cone $\mathcal{K}_P(\mathcal{B}_j)$ defined by the boundary vectors $\hat{\mathbf{k}}_1$ and $\hat{\mathbf{k}}_2$. Figure (b) shows the shadow region$\mathcal{D}_P(\mathcal{B}_j)$ of point P with respect to obstacle $\mathcal{B}_j$, which is shown in dark brown. Figure (c) is the intersect of all the shadow regions defined from the vertices of $\mathcal{T}_i$, $\bigcap_P \mathcal{D}_P(\mathcal{B}_j)\ \forall P \in \mathcal{V}_i$. In such a region, the sensor cannot measure any point of $\mathcal{T}_i$ as it is blocked by $\mathcal{B}_j$. Figure (d) demonstrates the shadow region with respect to all the obstacles in the workspace $\mathcal{W}$.

$$\mathcal{D}_P(\mathcal{B}_j) = \mathcal{K}_P(\mathcal{B}_j) \backslash Conv(\{P\} \cup \mathcal{V}_j) \tag{4.6}$$

The shadow region $\mathcal{D}_P(\mathcal{B}_j)$ represents the set of positions where the UGV cannot see the point P as it is blocked by obstacle $\mathcal{B}_j$. The shadow region of target $\mathcal{T}_i$ with respect to $\mathcal{B}_j$ is the intersect of all the shadow region of its vertices. In

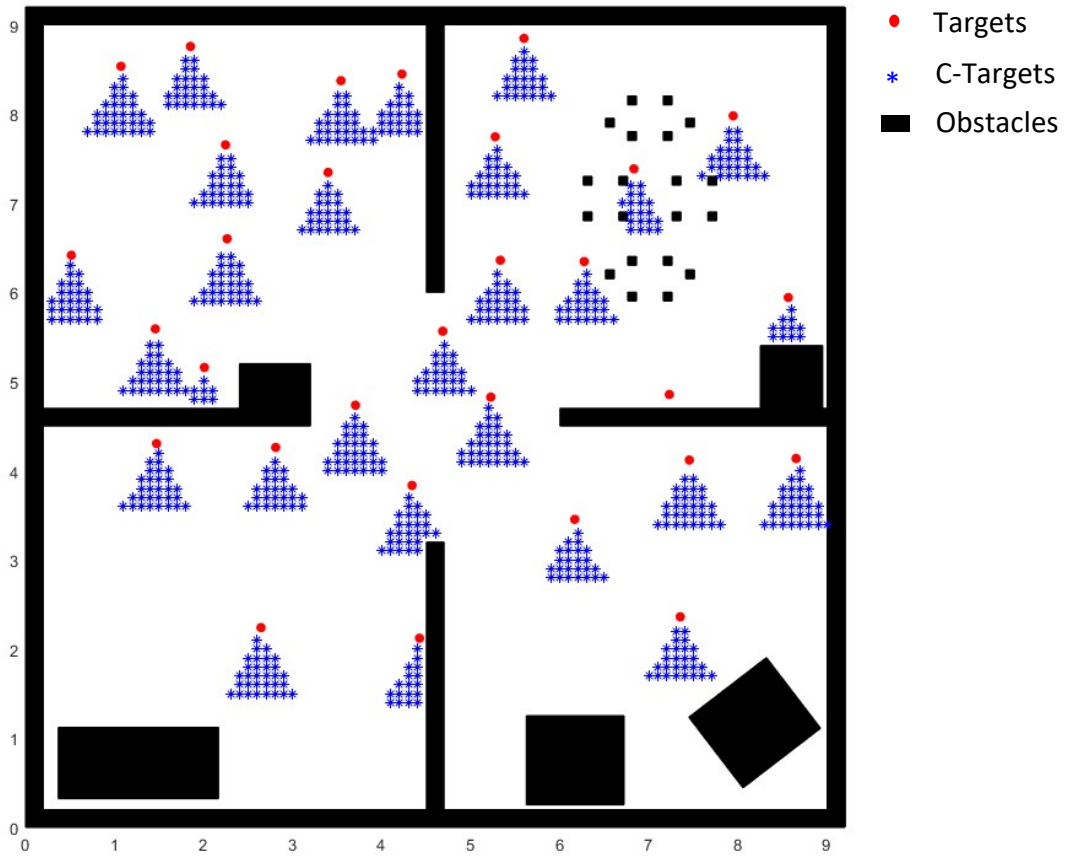

Figure 4.5: C-Target (shown in blue) in the Satisficing test workspace. In the upper right region, there exist 20 small obstacles placed near each other, and a target in the middle of them. The C-Target successfully captures the visibility of this obstacle-populated region, which enables the UGV to navigate in this region and obtain measurement.

Figure 4.4 (c), each $P \in \mathcal{V}_i$ forms a shadow region $\mathcal{D}_P(\mathcal{B}_j)$, and the intersect of all those regions represents the set of positions in which no point of the target $\mathcal{T}_i$ can be measured due to obstacle $\mathcal{B}_j$.

For a region populated with n obstacles $\mathcal{B}_1, ..., \mathcal{B}_n$, the set of all shadow regions of a target with respect to all the obstacles is defined as

$$\mathcal{D}(\mathcal{T}_i) = \bigcup_{j=1}^{n} (\bigcap_{P} \mathcal{D}_P(\mathcal{B}_j) \ \forall P \in \mathcal{V}_i) \tag{4.7}$$

Finally, the directional C-Target of target $\mathcal{T}_i$ at UGV orientation θ in the presence of obstacles is defined as

$$\mathcal{CT}_i^{\theta} = Conv(S) \backslash \{\mathcal{T}_i \cup \mathcal{D}(\mathcal{T}_i)\} \tag{4.8}$$

An example of directional C-Target in the Satisficing human test workspace is shown in Figure 4.5. The workspace is composed of 30 targets and 25 obstacles. In the upper right of the workspace, 20 small obstacles are placed relatively near to each other, and there is a target in the middle of them. As shown in the figure, the directional C-Target successfully captures the visibility characteristics and enable the UGV to navigate through this region to obtain measurement.

4.1.2 Algorithm and computational complexity

This section analyzes the computational complexity of calculating C-Targets for a workspace $\mathcal{W} = [x_0, x_1] \times [y_0, y_1]$ populated with r targets and n obstacles. The objective is to propose an algorithm that calculates C-Targets with acceptable complexity and the running time can be controlled by some of its parameters. The configuration space $\mathcal{C} = [x_0, x_1] \times [y_0, y_1] \times [\theta_0, \theta_1]$ is discertized for the x, y and θ axis with corresponding resolution r_x, r_y and r_θ. In this section and all the following chapters, it is assumed that $\theta_0 = 0$ and $\theta_1 = 2\pi$, as the UGV is free to rotate to any direction. The number of discrete values along each dimension is therefore $n_x = \frac{x_1 - x_0}{r_x} + 1$, $n_y = \frac{y_1 - y_0}{r_y} + 1$ and $n_\theta = \frac{\theta_1 - \theta_0}{r_\theta} + 1$. Denote the set of all

points obtained from discretizing $\mathcal{C}$ as $\mathcal{PS}$. The total number of points in $\mathcal{PS}$ is $N = n_x n_y n_\theta$.

Algorithm 1 Generating C-Target

Let $\mathcal{PS}$ be the set of points obtained from discretizing $\mathcal{C}$, $\mathcal{T}$ be the set of targets, $\mathcal{B}$ be the set of obstacles
Let $\mathcal{CT}$ be an $r \times 1$ array of sets that store C-Target points
for Point $p \in$ Point Set $\mathcal{PS}$ **do**
    Initialize flagT =$False$, flagOB =$False$
    targetID = empty array
    for i = 1 : $|\mathcal{T}|$ **do**
        if $p \in Conv((\mathcal{V}_i)_{-\mathbf{v}_1} \bigcup (\mathcal{V}_i)_{-\mathbf{v}_2} \bigcup (\mathcal{V}_i)) \backslash \mathcal{T}_i$ **then**
            flagT = $True$
            add t to targetID
        end if
        for Obstacle $ob \in$ Obstacle Set $\mathcal{B}$ **do**
            if line segment [p, t] intersects with ob **then**
                flagOB = $True$
            end if
        end for
    end for
    if flagT == $True$ and flagOB == $False$ **then**
        for i in targetID **do**
            Add p to $\mathcal{CT}[i]$
        end for
    end if
end for

Algorithm 1 loops through all the nodes in the configuration space, and checks if each node belongs to any $\mathcal{CT}_i^\theta$. Checking whether a node is in C-Target is implemented by Matlab "inpolygon" function, which takes constant time. For a target $\mathcal{T}_i$, checking the target shadow region $\mathcal{D}_i$ is not implemented directly. Instead, for a given point p, the Matlab "polyxpoly" function is applied to check if the line segment between p and $\mathcal{T}_i$ intersects with any obstacle $\mathcal{B}_j$. Checking each obstacle takes constant time. Therefore, the overall complexity of the algorithm is $\mathcal{O}(Nnr)$, which is linear with respect to the number of targets r and that of obstacles n. The complexity is also linear with respect to n_x

and n_y, which indicates that the running time can be controlled by changing the resolution on each axis.

4.2 Approximate cell decomposition with C-targets and C-obstacles

In the sensor path planning with obstacles, cell decomposition is a well-known method that decomposes the free configuration space $\mathcal{C}_{free}$ into non-overlapping polygons (cells) in which the path is planned for the sensor platform. Despite its computational cost, cell decomposition has the advantage of being resolution complete compared to other methods such as probabilistic roadmap method (PRM) [6]. The approximate rectangloid decomposition method, referred to as the approximate-and-decompose method [65], utilizes a predefined rectangloid shape to decompose the free configuration space $\mathcal{C}_{free}$ with the approximation of C-Obstacles $\mathcal{CB}$ and C-Targets $\mathcal{CT}$ [4, 15, 27, 31, 22, 11, 54, 62]. The approximate cell decomposition of $\mathcal{C}_{free}$ obtained from this method are composed of cells free of C-Obstacles and therefore can guarantee collision avoidance in the path planning step.

The cells are classified into two types. A $void\,cell$ is a convex polygon $\mathcal{K}_{void}$ in $\mathcal{C}_{free}$ such that no target can be measured in any configuration $\mathbf{q} \in \mathcal{K}_{void}$. An $observation\,cell$ is a convex polygon $\mathcal{K}_z$ such that in every configuration $\mathbf{q} \in \mathcal{K}_z$ the UGV can measure at least one target [6]. In the process of approximate cell decomposition, the C-Target index for each cell is maintained. To be more specific, for each C-Target $\mathcal{CT}_i$ ($i = 1, ..., r$), a cell index set $\mathcal{O}_i$ is maintained to store the indices of all the observation cell corresponding to C-Target $\mathcal{CT}_i$. In

the path planning step, visiting one of the observation cells in $\mathcal{O}_i$ is enough to obtain measurement from $\mathcal{T}_i$, and therefore it is necessary to spedify which set of cells need to be visited. Such constraints will be specified in the formulation of integer programming in the next chapter. In the case where a cell belongs to multiple C-Targets, it is also guaranteed that visiting this cell would enable the UGV to obtain measurement for all of them.

The configuration space is decomposed by the following steps [6].

1. For every $u = 1, ..., n_\theta$, compute the C-Obstacle $\mathcal{CB}_j^u$ for each obstacle $j = 1, ..., n$, and the directional C-Target $\mathcal{CT}_i$ for each target $i = 1, ..., r$.
2. For every $u = 1, ..., n_\theta$, generate bounding rectangloid approximation $\mathcal{RB}_j^u$ of $\mathcal{CB}_j$ for each obstacle $j = 1, ..., n$, and a abounded approximation $\mathcal{RT}_i$ for each $\mathcal{CT}_i$.
3. For every $u = 1, ..., n_\theta$, generate a rectangloid decomposition $\mathcal{K}_{void}^u$ of void configuration space

$$\mathcal{C}_{void}^u \equiv \mathcal{C}^u \backslash \{\bigcup_{j=1}^{n} \mathcal{RB}_j^u \cup \bigcup_{i=1}^{r} \mathcal{RT}_i^u\} \tag{4.9}$$

   where $\mathcal{C}^u = [0, x_m] \times [0, y_m] \times \theta^u$

$$\mathcal{K}_{void} = \bigcup_{u=1}^{n_\theta} \mathcal{K}_{void}^u \tag{4.10}$$

4. For every $u = 1, ..., n_\theta$, and $i = 1, ..., r$, generate a rectangloid decomposition $\mathcal{K}_{z,i}^u$ of $\mathcal{C}_{z,i}^u \backslash \bigcup_{l \neq i} \mathcal{C}_{z,l}^u$ where

$$\mathcal{C}_{z,i}^u = \mathcal{RT}_i^u \backslash \bigcup_{j=1}^{n} \mathcal{RB}_j^u \tag{4.11}$$

   and add the indices of cells in $\cup_{u=1}^{n_\theta} \mathcal{K}_{z,i}^u$ to $\mathcal{O}_i$.

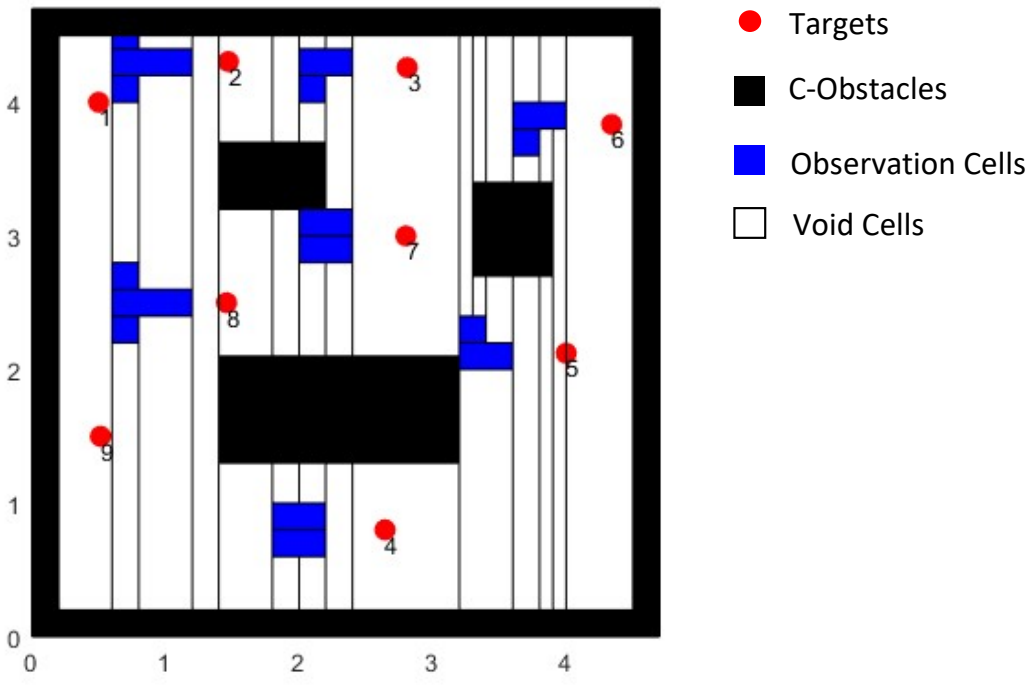

Figure 4.6: Result of approximate cell decomposition at $u = 2$ $(\theta = \pi)$. The observation cells are shown in blue, and void cells are shown in white. It can be observed that for each target there are more than one corresponding observation cells.

5. For each $u = 1, ..., n_\theta$, generate a rectangloid decomposition $\mathcal{K}^u_{z,l}$ of $\bigcup_{i=1}^{r}\{\mathcal{C}^u_{z,i} \cap \bigcup_{l \neq i} \mathcal{C}^u_{z,l}\}$. Then let $\mathcal{K}_z = \bigcup_{u=1}^{n_\theta}\{\{\cup_{i=1}^{r}\mathcal{K}^u_{z,i}\} \cup \mathcal{K}^u_{z,l}\}$. For each cell $c \in \mathcal{K}^u_{z,l}$, if $c \subset \{\mathcal{C}^u_{z,i} \cap \bigcup_{l \neq i} \mathcal{C}^u_{z,l}\}$, then add the index of c to $\mathcal{O}_i$.

The cell size has a large impact on path planning and therefore a parameter c_m is set to control the maximum permissible cell size in the decomposition step. For each cell in $\mathcal{K}^u_z$ $(u = 1, ..., n_\theta)$, if the size exceeds c_m it is decomposed into smaller cells by Algorithm 2.

Figure 4.6 illustrates the result of approximate cell decomposition for a workspace populated with 3 obstacles and 9 targets. The maximum permis-

Algorithm 2 Division of large cells

Let $\mathcal{K}_{void}$ be the set of void cells
Let $\mathcal{K}_z$ be the set of observation cells
Let c_m be the maximum permissible cell size
for cell c in $\mathcal{K}_{void}$ **do**
    if size of c larger than c_m **then**
        decompose c into a set of cells S_c with size $\leq c_m$.
        remove c from $\mathcal{K}_{void}$
        add all cells in S_c to $\mathcal{K}_{void}$
    end if
end for
for cell c in $\mathcal{K}_z$ **do**
    Let I be the index set of all the targets $\mathcal{T}_i$ such that $c \in \mathcal{O}_i$
    if size of c larger than c_m **then**
        decompose c into a set of cells S_c with size $\leq c_m$.
        remove c from $\mathcal{K}_z$
        add all cells in S_c to $\mathcal{K}_z$
        update $\mathcal{O}_i$, $i \in I$
    end if
end for

sible cell size c_m is set to be 5, and θ^u is set as π. It can be observed that for each target there are more than one corresponding observation cells. Visiting one of these observation cells is sufficient to obtain measurement of the target, and visiting extra observation cells for the same target would yield higher cost.

4.3 Connectivity graph

In this section, the connectivity graph used for path planning is described and special attention is paid to the cost on the edges. The UGV sensor is represented as a point in $\mathcal{C}$ by the connectivity graph. A connectivity graph is a directed graph where nodes represents either an observation cell or a void cell in $\mathcal{C}$ [6]. A directed edge e_{ij} that connects node v_i and v_j exists if and only if the cor-

responding cells are adjacent and the rotation required for the UGV to move from v_i to v_j is less than or equal to π. Compared to the definition of connectivity graph as an undirected graph, the directed connectivity graph enables the consideration of rotating cost in the edge costs. In the construction of the connectivity graph, the $observation\,cells$ become $observation\,noded$, and $void\,cells$ become $void\,nodes$. The index of node v uses the same index of its corresponding cell c. As a result, the index set $\mathcal{O}_i(i = 1, ..., r)$ does not need to be modified and now represents the observation node indices. The x, y coordinate of v is set to be the centroid of c, and the θ coordinate of v is the same as c.

The following rules of defining edges are set in accordance with the properties of directed edge e_{ij}. In this section and all the following chapters, n_θ is set to be 4 and θ is therefore discretized into four values: $\frac{\pi}{2}, \pi, \frac{3\pi}{2}$ and 2π.

1. Nodes with same orientation

   If v_i and v_j have the same orientation $\theta_i = \theta_j$, as θ is discretized into four values, the following rule holds: Edge e_{ij} exists if and only if c_i and c_j are adjacent and

$$\begin{cases} max(y(c_i)) \leq min(y(c_j)) & \theta_i = \frac{\pi}{2} \\ max(x(c_j)) \leq min(x(c_i)) & \theta_i = \pi \\ max(y(c_j)) \leq min(y(c_i)) & \theta_i = \frac{3\pi}{2} \\ max(x(c_i)) \leq min(x(c_j)) & \theta_i = 2\pi \end{cases} \tag{4.12}$$

   where $x(c_i)$ and $y(c_i)$ denotes the x and y coordinates of all the vertices of cell c_i respectively. Cells are adjacent if their edges overlap but the intersected area is zero. It is worth noting that the orientation is considered approximately: cells are only required to have overlapping edges, even

though the centroids of each cell might have different x or y. For example, in the case of $\theta = \pi$, ideally the UGV is moving in the horizontal position, which indicates that c_i and c_j should have the same y coordinate. However, the connectivity characteristics is considered approximately, so that as long as c_i and c_j has overlapping edges they are considered connected. This approximation allows the use of larger cell sizes so that the number of nodes are reduced and so is the running time. The influence of the cell size on the running time and performance of path planning will be shown in later chapters.

2. Nodes with different orientation

   In the case where v_i and v_j have different orientation, e_{ij} indicates that the UGV has to conduct both translation and rotation to go from c_i to c_j. As the orientation is discretized with interval of $\frac{\pi}{2}$, the following rule is set: Edge e_{ij} exists if and only if $|\theta_i - \theta_j| = \frac{\pi}{2}$ and the projection of the two rectangloids on the $x - y$ plane intersect with each other.

For each edge in the connectivity graph, it is associated with a cost of positive value. As mentioned before, the UGV has to both translate and rotate to move from one node to another. Therefore, the edge cost is defined as a weighted combination of translation cost and rotation cost. For an arbitrary node v_i, denote $\mathbf{x}_i = [x_i, y_i]^T$ as the $x - y$ coordinate of v_i, and θ_i as the θ corrdinate of v_i. For edge e_{ij} connecting node v_i and v_j, the cost ϵ_{ij} is defined as:

$$\begin{aligned} \epsilon_{ij} &= w_t \times \|\mathbf{x}_i - \mathbf{x}_j\|_2 + (1 - w_t) \times d_\theta \\ d_\theta &= \begin{cases} \frac{\pi}{2} & \{\theta_i = 2\pi, \theta_j = \frac{\pi}{2}\} \text{ or } \{\theta_i = \frac{\pi}{2}, \theta_j = 2\pi\} \\ |\theta_i - \theta_j| & \text{otherwise} \end{cases} \end{aligned} \tag{4.13}$$

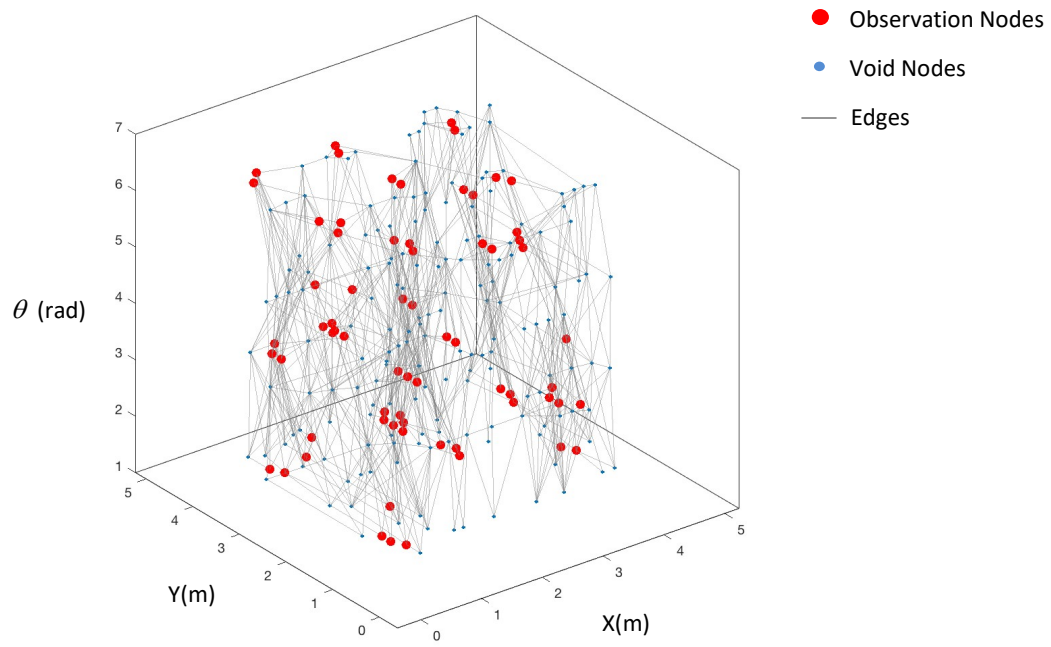

Figure 4.7: Example of the connectivity graph. The observation nodes are shown in red, and void nodes are shown in blue. It can be observed that the graph is not complete: most nodes are only connected to a small number of other nodes.

where w_t is the weight of translation and $1 - w_t$ is therefore the weight assigned for rotation. The translation weight w_t can be chosen as $\frac{\omega_m}{v_m+\omega_m}$. If v_m is far larger than ω_m, w_t is close to zero and the rotation cost dominates and so is the opposite case. In later chapter, we will run simulation under different w_t and observe the difference in the results obtained.

CHAPTER 5

PATH PLANNING VIA INTEGER PROGRAMMING

In this chapter, a specially designed method, integer programming, is derived for solving the path planning problem in the connectivity graph obtained from the previous chapters. The availability of adding constraints and reoptimize makes it possible to adopt an integer programming approach to the traveling-salesman problem (TSP)[37]. The major advantage is the geniality of the method and the modest solution time, which will be demonstrated in the examples in the next chapter. The directional sensor path-planning problem, however, is by nature very different from the TSP and in section 5.1 the important properties of this problem are examined. The following sections will focus on transforming the directional sensor path planning problem into an optimization problem with constraints such that the integer programming can be applied.

5.1 Important properties of the problem

The directional sensor path planning problem has some important properties that make it hard to solve. Some of the barriers come from the objective itself, and some of them are imposed by the properties of the connectivity graph. These properties mark the significant differences from the classical TSP and can explain why many classical solutions would fail in this case.

First, the number of nodes in the path is not fixed. Recall that the nodes are classified into observation nodes and observation nodes. As the objective is to measure and classify all the targets, there is no requirement that all void nodes

need to be visited. In fact, void nodes serve as "transfer stations" that connects all the observation nodes. Visiting unnecessary void cells increases overall time and degrades the performance.

Second, there is even no need to visit all the observation nodes. For each target, there are more than one observation nodes corresponding to different sensor orientation θ, and visiting one of them is sufficient to obtain measurement for this target. For example, in Figure 5.1, there are 10 observation cells for target 7 in which the UGV can take mansurement, and it is sufficient to visit at least one of them. This property distinguishes the problem from the TSP problems: There is no specific set of nodes that must be visited, but only a constraint on a set of nodes that at least one of them needs to be visited. Any algorithm that depend on a set of compulsory nodes cannot be adopted to this problem.

Third, the connectivity graph may not be complete. A complete graph is a graph in which each pair of graph nodes is connected by an edge. As illustrated in Figure 4.7, each cell is only connected to a small number of other cells. In the case of non-complete graph, it can either be modified into a fully connected one, or can be solved directly with existing edges. The consideration of the pros and cons will be discussed in the following section.

Fourth, it is possible that some observation nodes need to be visited more than once in the optimal solution. For example, for the connectivity graph in Figure 5.2, the start node is the void node, and the objective is to visit all the other nodes. The observation node 2 must be visited twice in any optimal solution. In this case, constraining the visit time to one would yield to suboptimal solution or even no solution at all. This property also distinguish the problem from classical TSP, as in the latter case no duplicate visit is allowed.

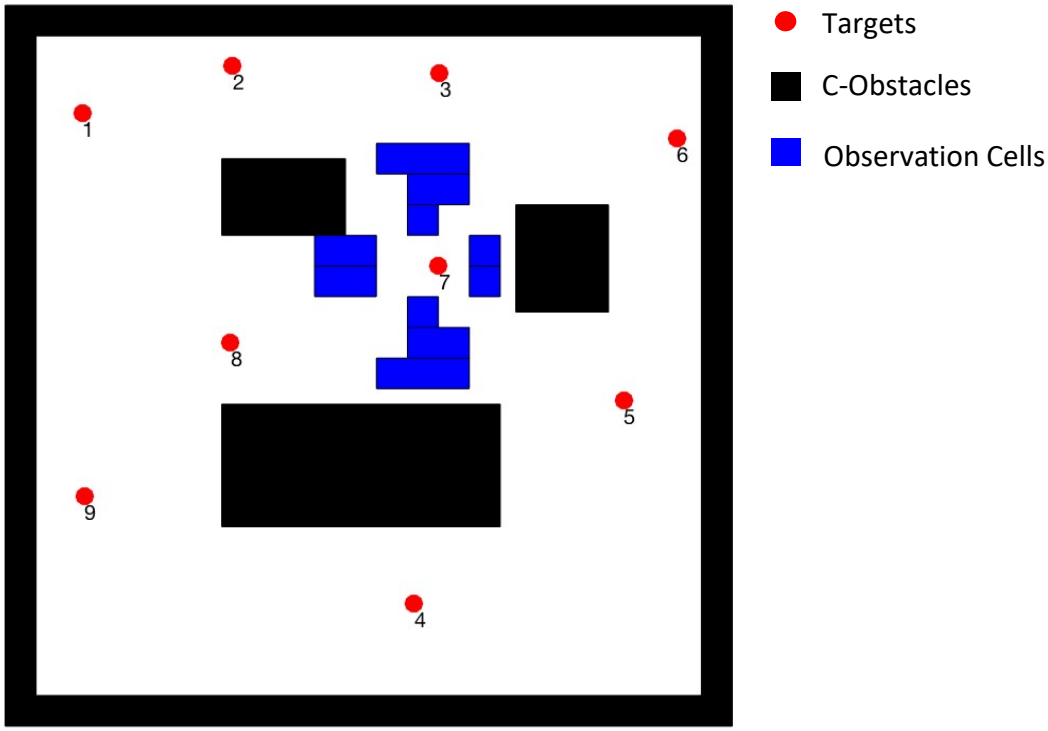

Figure 5.1: An example of many observation cells correspondign to the same target. For target 7, there are 10 observation cells in which the UGV can measure it. Visiting one of them is enough to obtain measurement of Target 7.

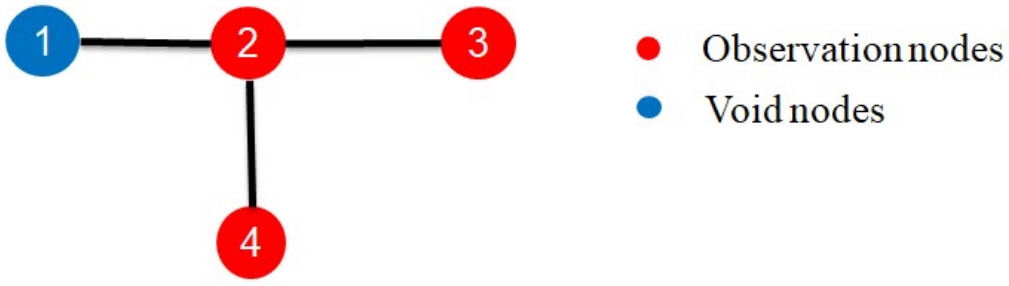

Figure 5.2: An example of the necessity to visit observation node more than once. The start node is void node, and the objective is to visit all the other nodes. Observation node 2 must be visited twice in any optimal solution.

Fifth, in this problem it is not required that the UGV return to the starting position. In other words, the UGV's path does not need to form a loop. As will be discussed in the next section, this "no return" problem can be transformed to a return problem. However, special attention should be paid to the start node, as in this problem the start node of the robot is fixed. It is easy to incorporate this constraint with integer programming.

Looking back at these properties, they make the problem very different from the classical TSP. Property 1 and 2 defines a unique objective that is hard to transform to that of TSP. Property 3 and 4 make the graph much harder to solve, and property 5 causes a tricky problem with the starting node. The integer programming problem is formulated incorporating these properties.

5.2 Non-completeness and no-returning

5.2.1 Non-completeness

In the classical TSP, the graph is restricted to be complete, which indicates that for an arbitrary pair of nodes in the graph there exists an edge connecting them. Many TSP algorithms need this property to obtain an optimal or approximate solution. However, as mentioned before, the case is quite opposite: each node is only connected to a small number of other nodes. Take the workspace of Figure 4.7 as an example. There are 262 nodes in the graph, indicating 68382 possible directed edges. However, the number of existing edges is actually only 2634, or 3.85% of all the possibilities. The sparsity has big influence on the consideration of the transformation to be discussed below.

A first intuitive approach would be to assign a large cost to the edges that should not exist, and then solve the TSP problem as if the graph is complete. The cost should be large enough to eliminate the possible existence of it in the optimal path. This method, however, seraches in an unnecessarily large search space, and would therefore suffer from slow convergence or even inability to solve numerically. In the example above, there should be only 2634 variables to be considered, but this approach searches the optimal solution with 68383 variables. In even larger workspaces, such as the satisficing test workspace, the number of possible directed edges is significantly larger, and the situation worsens. The unnecessarily large search space would affect the performance dramatically and therefore make the approach untractable.

A second method also seeks to make the graph complete, but in a more sophisticated way. As the graph has no isolated node, for any node v_i and v_j that there exists no edge between them, we can find a shortest path from v_i to v_j by Dijkstra's algorithm, and the corresponding cost is denoted as ϵ'_{ij}. An addition edge connecting v_i and v_j with cost ϵ'_{ij}. After iteratively adding edges between all such pairs of nodes, the graph becomes complete. An illustration of this approach is shown in Figure 5.3. For node 3 and node 4, there is no edge connecting them. An additional edge is added between node 3 and node 4, with the cost of the shortest path from node 3 to node 4, which is the cost of the path "3-2-1-4".

The drawback of this approach is clear. First, it does not resolve the issue of extra search space and therefore face the same challenges as the first one. Second, this approach actually loses the optimality of the solution. Recall that the requriement is to visit some of the observation cells. When adding an edge be-

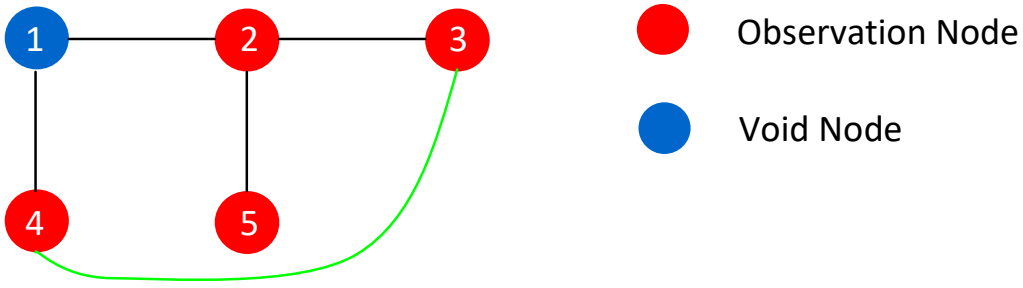

Figure 5.3: An example of the method of making the graph complete. For node 3 and node 4, there is no edge connecting them. An additional edge is added between node 3 and node 4, with the cost of the shortest path from node 3 to node 4, which is the cost of the path "3-2-1-4".

tween v_i and v_j, chances are that the shortest path between them passes through some observation cells, one of which is denoted by v_o. Therefore, when the UGV moves from v_i to v_j in the graph, it already visits v_o and there's no need to visit it again in the next steps. But the algorithm does not recognize this and would pay extra cost visiting v_o. In the example in Figure 5.3, going from node 3 to node 4 indicates adopting the path "3-2-1-4", which already visits obsesrvation node 2. However, the algorithm would still try to visit node 2 in later steps. Therefore, this approach is not desirable.

In order to restrict the search space to be small enough,the variables should only represent the existing edges in the graph. This can be easily implemented in the integer programming approach, the detail of which will be discussed in the next section.

5.2.2 Transforming no-returning to returning problem

The UGV is not required to return to the starting configuration. In the connectivity graph, it indicates that the path does not include an edge that returns to the starting node. If the last node in the path is specified, it can be constrainted that the last node has only incoming edges, but this is not the case in the problem at hand. In fact, it is much easier to transform it to a returning problem by adding a dummy node, with special attention paid to the starting node.

Given a graph $\mathcal{G}$ composed of nodes $\mathcal{V}$ and edges $\mathcal{E}$. A dummy node v_d is added to $\mathcal{V}$, and the new set of nodes is denoted by $\mathcal{V}'$. Edges from v_d to all nodes in $\mathcal{V}$ with zero cost and from all nodes to v_d are added to $\mathcal{E}$, and the new set of edges is denoted as $\mathcal{E}'$. The dummy node is constrained to be visited only once. The problem now can be solved assuming returning to the starting node. The optimal path solved on $G' = \{\mathcal{V}', \mathcal{E}'\}$, denoted as P', consists of two edges from and to v_d with zero cost. Eliminating such two edges would yield the optimal path P that is a solution to $\mathcal{G}$, as the total cost of P and P' would be the same.

In the transformation above, the starting node is not taken into consideration. Adding a dummy node first and eliminating it afterwards might give us arbitrary starting node in G, but in our problem the starting node v_s is fixed as the initial configuration of the UGV is fixed. Therefore, it is necesary to add another constraint specifying that the edge $v_d \rightarrow v_s$ in E' is always selected. This would fix the starting node in P. This will form one of the constraints in integer programming to be described in the next section.

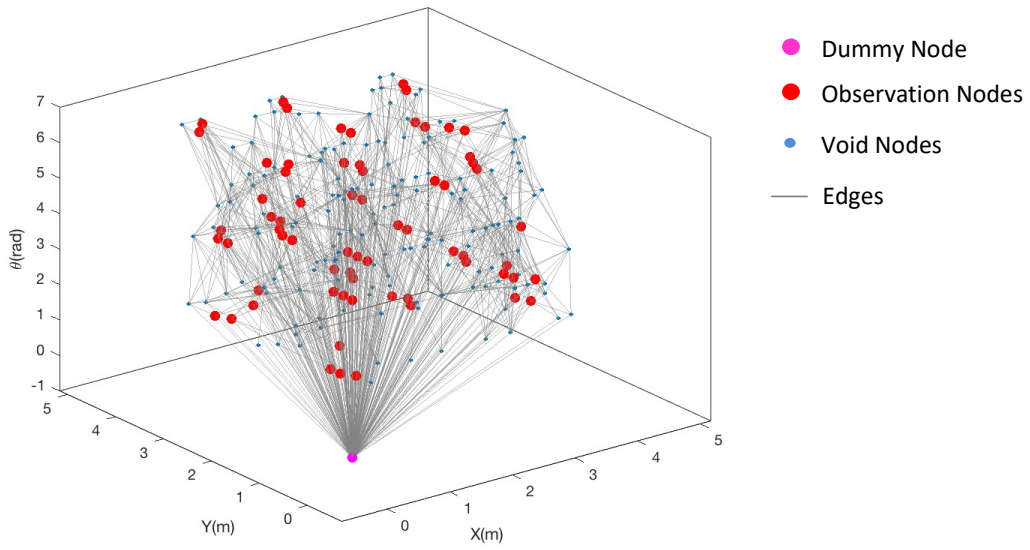

Figure 5.4: The connectivity graph with a dummy node added. The edge between the dummy node and any other node is associated with zero cost. The visiting time of the dummy node must be set to 1, otherwize the algorithm would repeatedly visit the dummy node as it has lowest cost.

5.3 Formulation of Integer Programming

Integer programming method has been applied to many graph problems such as Traveling Salesman. The basic idea is to assign a binary variable denoting the choice of arc in the optimal solution, and minimize the total cost under certain constraints. The optimization problem is derived as follows.

Given a directed graph $G' = \{\mathcal{V}', \mathcal{E}'\}$, obtained by adding a dummy node as described in the previous section, for edge $e_{ij} \in E'$ from node v_i and to node v_j, a corresponding binary variable is defined as

$$x_{ij} = \begin{cases} 1 & \text{if edge } e_{ij} \text{ is in the optimal path} \\ 0 & \text{otherwise} \end{cases}$$

The objective is to minimize the total edge cost.

$$\min \sum_{i,j} \epsilon_{ij} x_{ij} \tag{5.1}$$

The constraints are defined as follows.

1. "In == Out"

   As the problem has been transformed into a returning problem, the times of UGV going to a node equals the times of UGV leaving a node.

   $$\sum_{j} x_{jl} = \sum_{j} x_{lj} \quad \forall l \tag{5.2}$$

2. Visit dummy node only once

   By limiting the time of visiting the dummy node to be one, we ensure that the path obtained is a single path instead of several segments of path. Denote the index of the dummy node as i_d.

   $$\sum_{j} x_{ji_d} = \sum_{i_d} x_{i_d j} = 1 \quad \forall j \tag{5.3}$$

3. Start node

   Denote the index of the start node as i_s.

   $$x_{i_d i_s} = 1 \tag{5.4}$$

4. C-Target visited at least once

   Recall that for target $\mathcal{T}_i$ ($i = 1, ..., r$), $\mathcal{O}_i$ as the set of indices of all observation nodes corresponding to C-Target $\mathcal{CT}_i$. To ensure that all targets are classified, we need to ensure that for any target $\mathcal{T}_i$, nodes in $\mathcal{O}_i$ are visited at least once.

$$\sum_{l} x_{lj} \geq 1 \quad \forall j \in \mathcal{O}_i, \forall i \tag{5.5}$$

5.4 Sub-tour Elimination

When applying integer programming to solve the path planning problem, one major challenge is eliminating subtours. A subtour is a directed cycle that satisfies all the previous constraints. If the optimal math consists of more than one subtour, the nodes in any subtour does not guarantee measuring all the targets. The optimal solution should contain only one subtour that defines the optimal path. However, the constraints above does not guarantee that there exists only one loop, and the optimal solution of integer programming may consist of more than one loops. An example of this case is shown in Figure 5.5 .

To eliminate subtours, there exist several classical formulation. The benchmark is the subtour formulation. Other weaker ones uses extra variables, one of which will be discussed in more detail. A comparison is given in [25, 30, 47]. A first way of eliminating sub-tours is called ”sub-tour formulation”, without assigning additional variables [32].

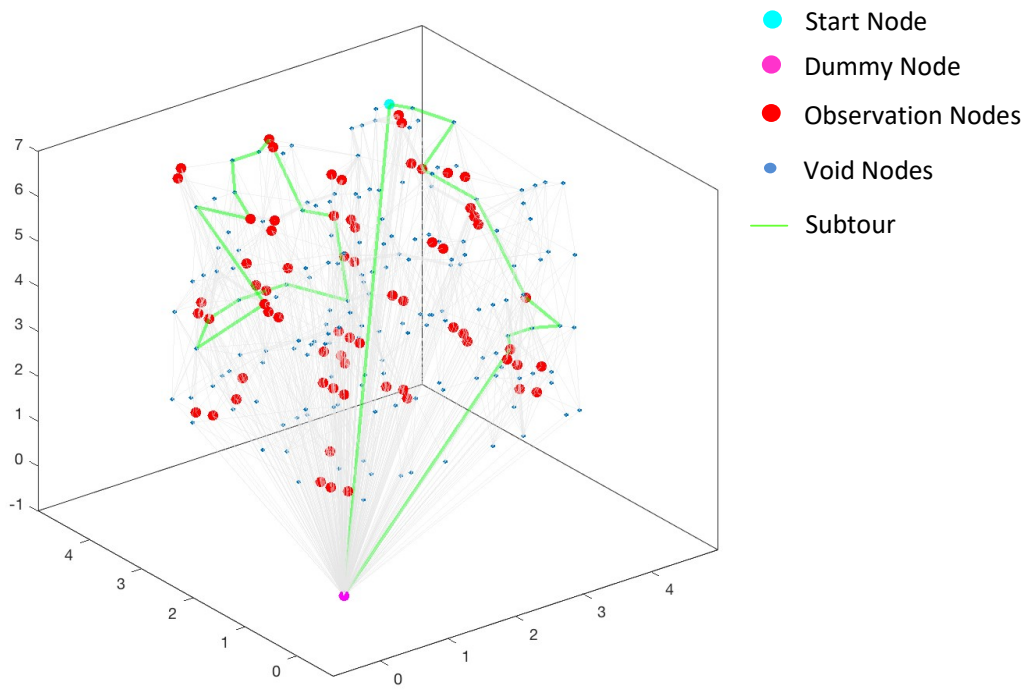

Figure 5.5: An example of the solution of integer programming with 2 subtours. For each subtour, it visits some of the obsevation nodes, but it is not connected to other subtours and is thus isolated. Subtours can be eliminated by iteratively adding constraints that eliminate the solution with subtours obtained at each optimization.

$$\sum_{l \in S, j \in S, l \neq j} x_{lj} \leq |S| - 1 \quad \forall S \subsetneq \mathcal{V}', |S| > 1 \tag{5.6}$$

where $|\cdot|$ denote the cardinality of a set. The number of constraints equals the total number of all possible S. The sub-tour formulation would introduce $2^{|\mathcal{V}'|} - 1$ constraints. The number of constraints in too large to be handled by the numerical solver and therefore this formulation should not be implemented directly.

The second way, called Miller-Tucker-Zemlin (MTZ) formuation [38], uses extra variables u_j $(j = 1, ..., |\mathcal{V}'|)$ and the constraints

$$
\begin{aligned}
u_1 &= 1 \\
2 \leq u_j &\leq |\mathcal{V}'|, \forall j \neq 1 \\
u_j - u_l + 1 &\leq (|\mathcal{V}'| - 1)(1 - x_{jl}) \forall j \neq 1, \forall l \neq 1
\end{aligned}
\tag{5.7}
$$

The major advantage of the MTZ formulation is its relatively small size: there are roughly $\frac{|\mathcal{V}'|^2}{2}$ extra constraints.

However, given the size of connectivity graph at hand, even the MTZ formulation introduces too many constraints for the solver to work effectively. A relatively better approach would be an iterative one. First ignore the subtour constraints and obtain an optimization result, then identify all the subtours in the optimal path. For each subtour, the set of all nodes in the path forms $S \subsetneq V'$ in Equation 5.6, and then add the constraint as defined in Equation 5.6 and run the optimization again. The iteration continues until we obtain a solution with no subtour, which indicates that the result is satisfactory.

Algorithm 3 Subtour iteration

Let $opts$ be the optimization problem of integer programming,
Let $cons$ be the initial constraints
$tour$ = optimal solution of $opts$ with respect to $cons$
nt = number of subtours in $tour$
while $nt > 1$ **do**
    for z = 1: nt **do**
        Let V_z be the set of node indices in the z-th subtour
        Add new constraint $\sum_{i \in V_z, j \in V_z} x_{ij} \leq |V_z| - 1$ to $cons$
    end for
    $tour$ = optimal solution of $opts$ with respect to $cons$
    nt = number of subtours in $tour$
end while

CHAPTER 6

SIMULATION AND RESULTS

The implementation of the path planning methodology described in the previous chapters is tested on a variety of workspace with different connectivity characteristics such as obstacle and target densities and narrow passages. In the next section, the influence of translation weight w_t and maximum cell size c_m on path planning are examined. The results presented in section 6.2 demonstrate that the proposed method outperform the nearest neighbor and 2-opt heuristics in distance-optimal path planning, and the major advantage is that it also takes rotation time into consideration.

6.1 Influence of translation weight w_t

In this section, the optimal sensor path obtained from cell decomposition and integer programming is plotted along with sensor configurations on the workspace. The FOV of the sensor is plotted by a green triangle when it reaches an observation cell and obtains measurement from a target. Figure 6.1 and Figure 6.2 shows the optimal path with different w_t. It is observed that the value of translation weight w_t must be accounted for in sensor path planning. Suppose $\mathcal{W}$ contains 9 targets indicated by red thick dots and 9 obstacles indicated by black solid rectangles as shown in Figure 6.5. Different translation weight w_t would produce different cost on the edges of the connectivity graph. As the objective is to find a path with the lowest cost, a large w_t indicates large cost associated with translation, and encourages the UGV to rotate more to save translation distance. In Figure 6.1, the optimal path is planned with a relatively small value of $w_t = 0.3$. The sensor first measures target 8, then moves forward

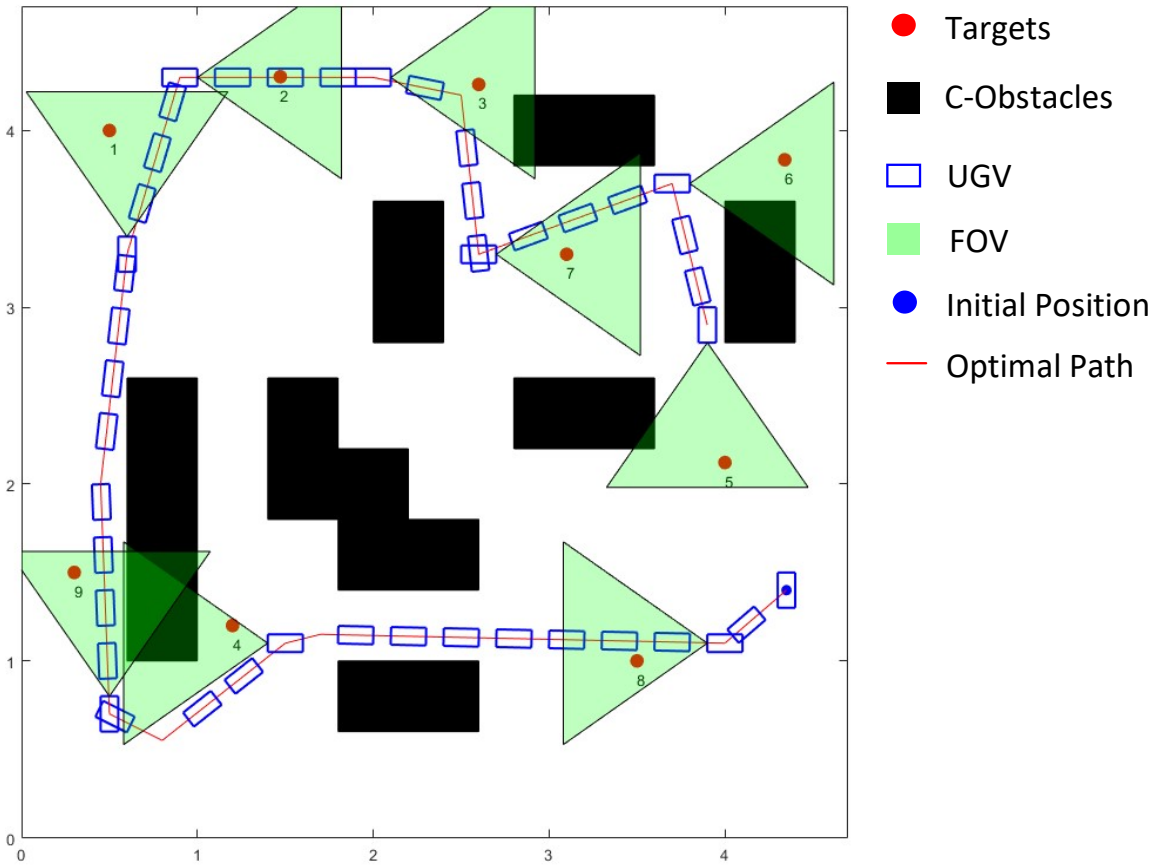

Figure 6.1: Optimal path with $w_t = 0.3$, $c_m = 4$. A low w_t indicates higher rotation cost and discourages rotation. The sensor first measures target 8, then moves forward for a long distance to obtain measurements of target 4 because it does not need to perform rotation in this process.

for a long distance to obtain measurements of target 4 because it does not need to perform rotation in this process. However, in Figure 6.2 where translation cost dominates, the sensor rotates toward a much nearer target 5 after obtaining measurement of target 8. Moveover, the sensor shows a "shake head" behavior near target 5 and 6. It chooses a path that has targets along both sides, and rotates left and right (each time by approximately $\frac{\pi}{2}$) to obtain measurements. In this way the sensor does not need to go to each target and the total translation cost is minimized.

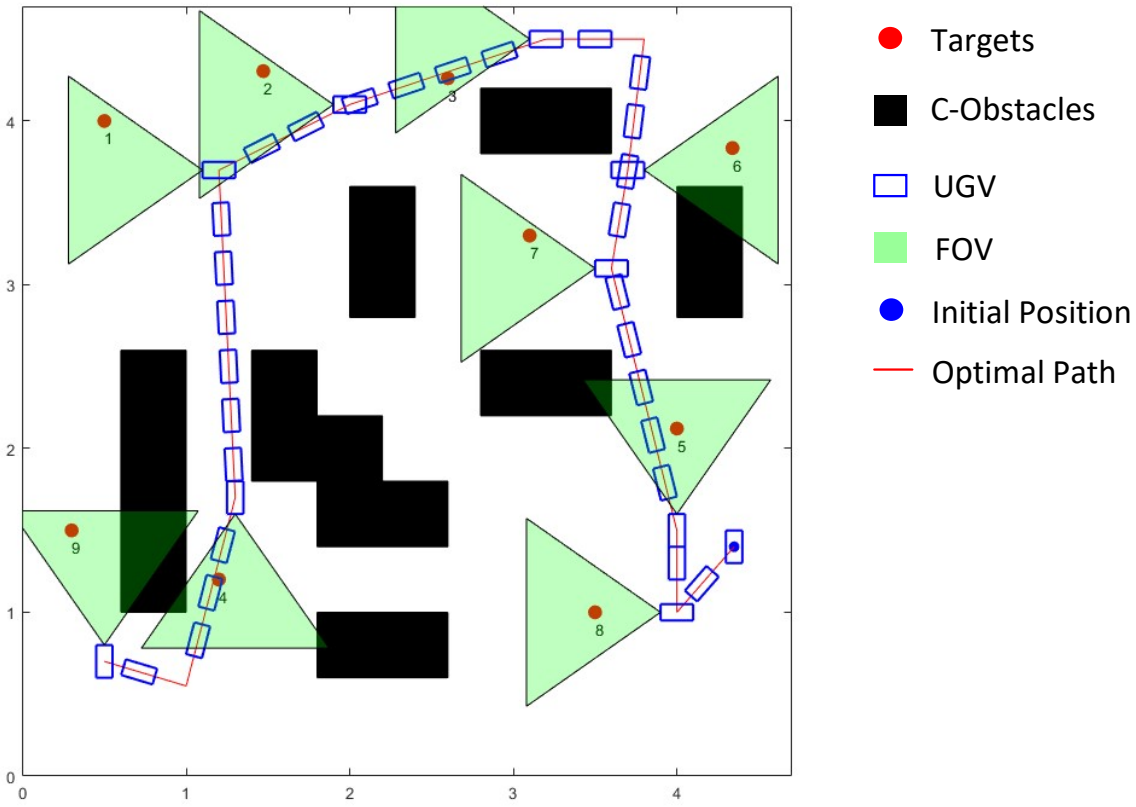

Figure 6.2: Optimal path with $w_t = 0.7$, $c_m = 4$. A high w_t encourages the UGV to rotate more to save the total distance traveled. The sensor rotates toward a much nearer target 5 after obtaining measurement of target 8 that is nearer.

Recall that w_t is chosen as $\omega_m/(v_m + \omega_m)$. For different pairs of v_m and ω_m, different values of w_t are chosen and the corresponding optimal path is planned. In fact, w_t is closely related to the robot kinematics. For example, it is much easier for a unicycle to turn around than a unmanned underwater vehicle (UUV), and the relative cost of translation and rotation are very different in both cases. The freedom of choosing w_t enables the method to adapt to robots with different characteristics.

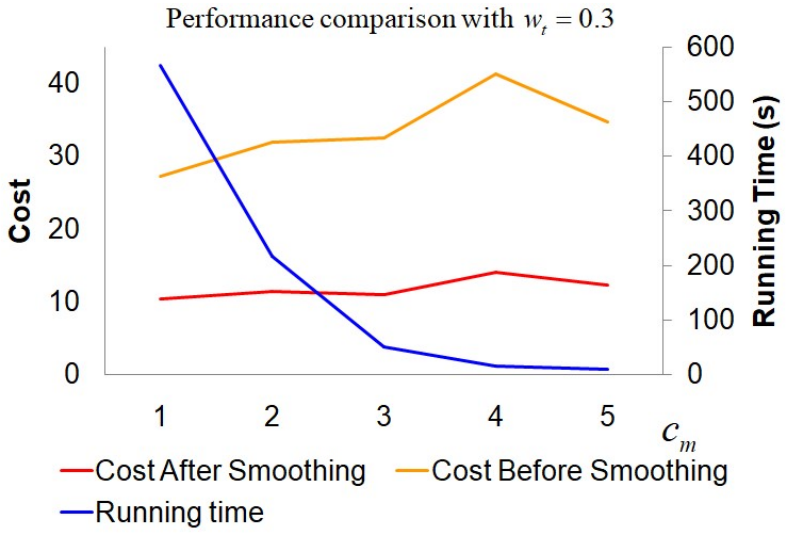

Figure 6.3: Comparison of optimal cost and running time for all c_m. Smoothing significantly reduces the cost. In general, reducing c_m results in lower optimal cost but longer run time.

6.2 Influence of maximum permissible cell size c_m

The cell decomposition approach decomposes $\mathcal{C}_{free}$ into cells with a maximum permissible cell size c_m such that the length and height of each cell is smaller than c_m. In this section, simulation conducted in Map 1 shown in Figure 6.4 demonstrates the influence of c_m on path planning results. Figure 6.3 compares the cost and running time with different cell sizes. Smoothing of the path obtained from cell decomposition would achieve path continuity and help eliminate broken lines, frequently turning points and large cumulative turning angle [14, 59]. Three important observations can be made from Figure 6.3.

First, the running time increases significantly as the cell size decreases. The number of cells would increases quadratically with respect to cell size, and so is the number of nodes in the connectivity graph. Even though the integer pro-

gramming approach avoids the exponential complexity with respect to node number associated with TSP, the running time still increases at a relatively high speed.

Second, a smaller cell size would improve the performance of the path planning approach. The general trend of the cost associated with the optimal path is descending as cell size decreases. Reducing cell size improves the performance for two reasons. On is that as the UGV is unable to maneuver inside a cell, if the cell size is too large, some maneuvers that helps to reduce the cost might be missed. The other is that as the sensor moves between centroid of cells, large cell size would result in broken lines and frequently turning points that can be improved by reducing cell size.

Third, after smoothing the benefit of reducing cell size becomes small. Smoothing reduces the cost associated with broken lines and frequently turning points. After the cell size is sufficiently small, the benefit of maneuver diminishes. This observation suggests the use of a moderate cell size combined with smoothing in order to make a good tradeoff between time and performance.

6.3 Performance comparison with benchmark methods

In this section, the sensor path planning method developed in this thesis is tested on three workspace with various connectivity characteristics. The first workspace, denoted as Map1, consists of 3 obstacles and 9 targets. Only a small portion of straight line between target pairs are blocked by obstacles. Map 2 is populated with 9 obstacles that block most target pairs. Map 3 is the $10m \times 10m$ with 30 targets and 25 obstacles shown in Figure 6.6. Out of the 25 obstacles, 20

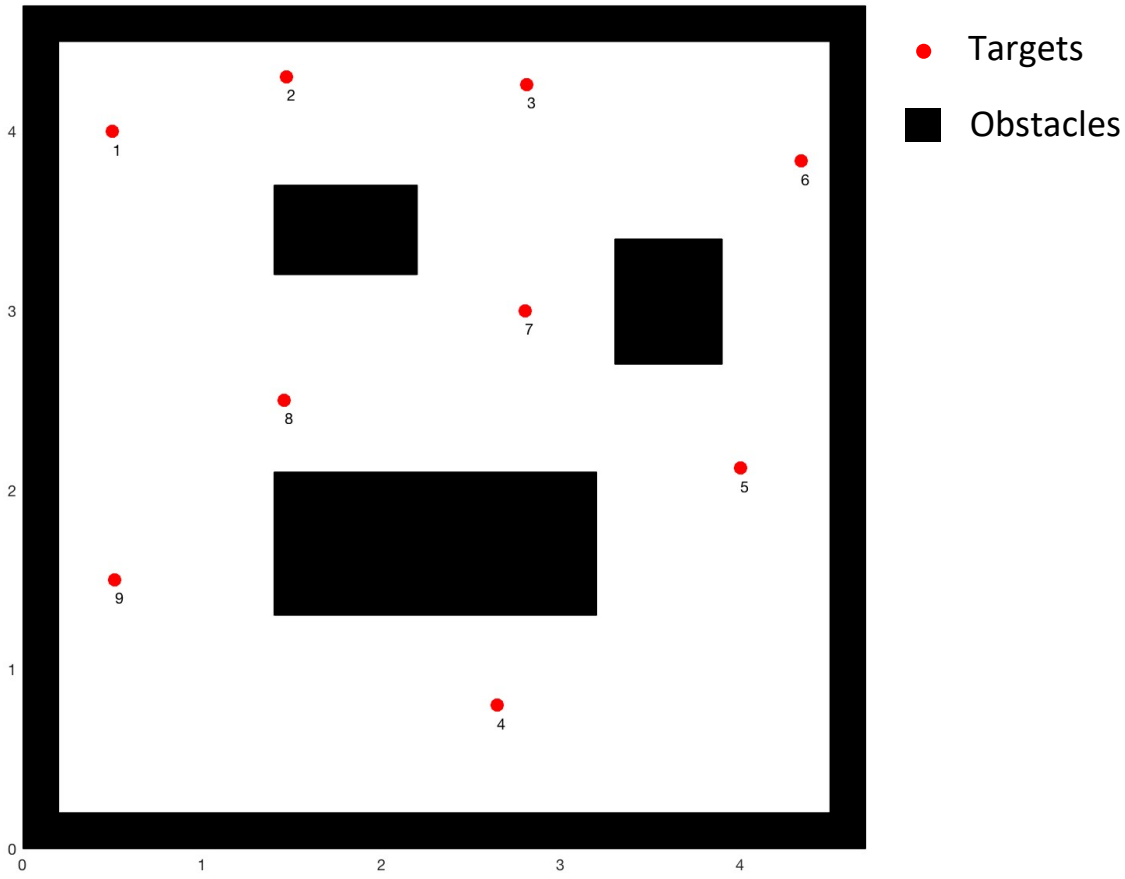

Figure 6.4: Map1 is composed of 3 obstacles and 7 targets. The number of obstacles is relatively small. As a result, a large portion of target pairs are connected with a collision-free straight line segment.

of them are small polygons populated in the upper right room with relatively high density. The 30 targets are spread across the entire workspace. The initial position of the UGV is in the lower right room marked by the thick blue dot.

In the following figures, $\mathcal{S}$ is triangle plotted in green whenever the sensor obtain measurements, and UGV is plotted as a rectangle in blue along the optimal path. The index i of each target that is measured by the sensor is shown next to $\mathcal{T}_i$ in all figures.

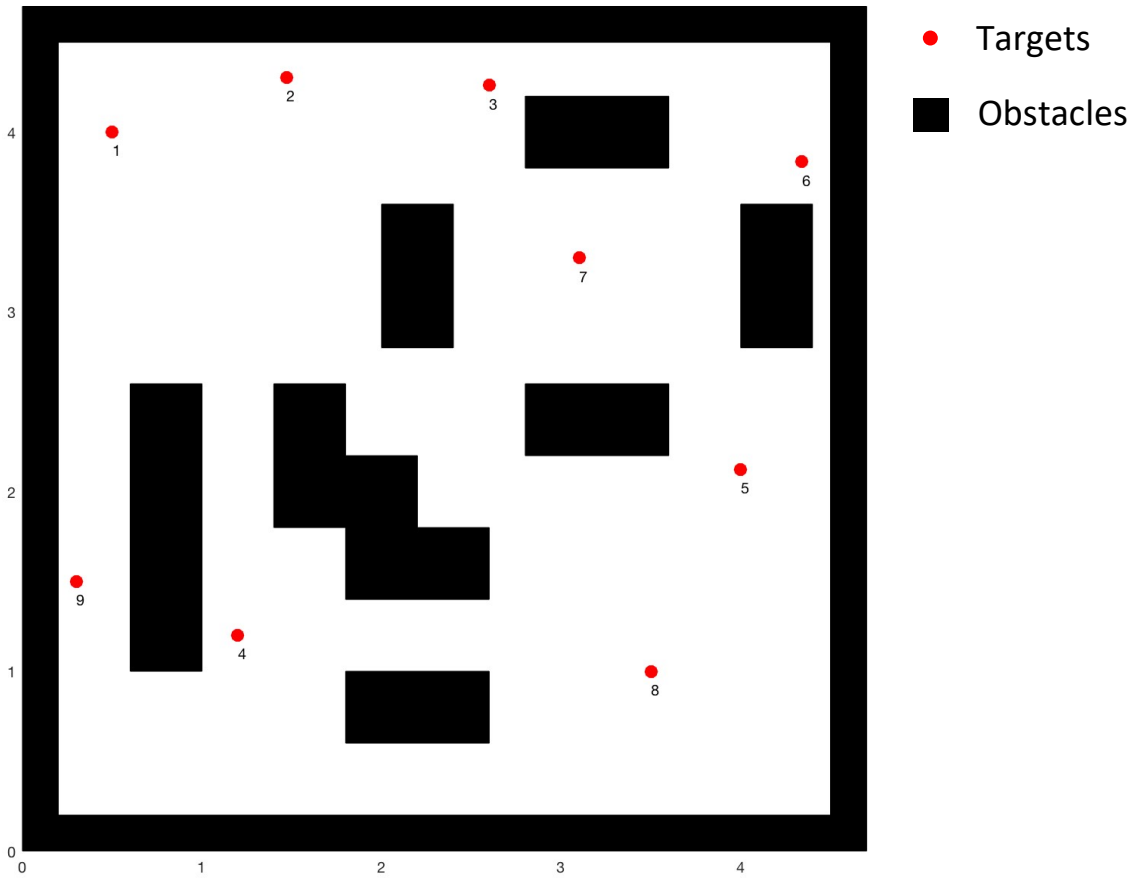

Figure 6.5: Map 2 is composed of 9 obstacles and 7 targets. More obstacles in the workspace blocks more straight paths between target pairs.

For comparison, two benchmark methods, the nearest neighbor (NN) [24] and 2-opt heuristics [24, 18], are implemented for the directional sensor. Since these two methods do not take non-complete graph and obstacles into consideration, they are modified to adapt to the obstacle populated workspace.

The nearest neighbor method is a simple and straightforward approach developed to generate path for TSP [40]. Every time the UGV decides on the next move, the collision-free path from the current configuration to every other target is calculated via methods such as Dijkstra or A* alrogithm. The UGV chooses

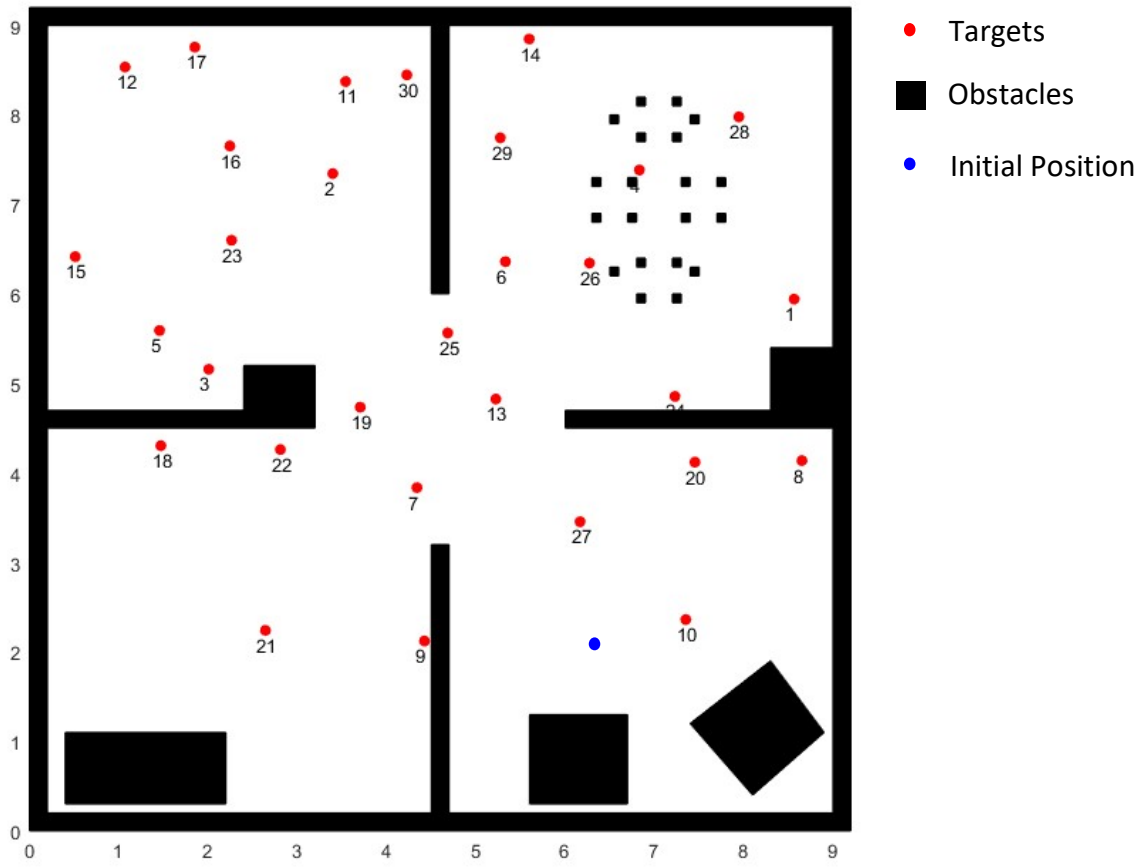

Figure 6.6: Map 3 is composed of 25 obstacles and 30 targets. Out of the 25 obstacles, 20 of them are small polygons populated in the upper right room with relatively high density. The 30 targets are spread across the entire workspace.

the nearest one as the next point and iterates until all the targets have been visited. In the presence of obstacles, the connectivity graph is not complete, and the nearest neighbor approach might fail to find the next unvisited target. Here, the connectivity graph is modified as a complete graph by adding extra edges. For any pair of unconnected nodes, an extra edge is added with a cost corresponding to the shortest collision free path between them. The connectivity graph is therefore complete and the sensor path can be planned using "nearest neighbor" approach.

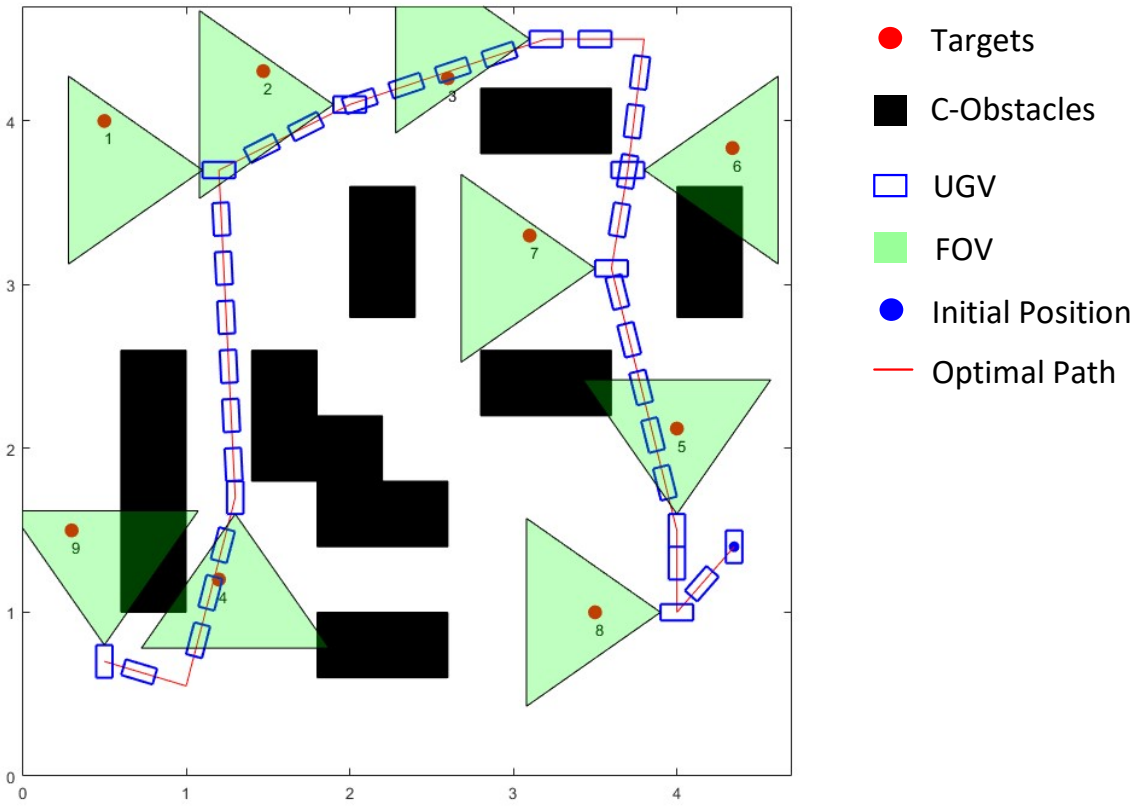

Figure 6.7: The optimal path obtained from cell decomposition and integer programming with $w_t = 0.9$ for Map 2. The sensor demonstrates the "shake head" behavior near target 6 and 7, which allows the sensor to rotate more to reduce translation cost. The total translation is 11.1m and total rotation is 14.7 rad.

A popular approach to reduce path cost is a heuristics named as 2-opt algorithm [40]. The 2-opt algorithm consists of removing two arbitrary edges in the path, reconnecting the path using two new edges, and then maintaining this change if it reduces the total cost. It has been proven to produce a solution no worse than the average cost of a tour in polynomial time [48]. In the sensor path planning problem with obstacles, the 2-opt algorithm is implemented on the path obtained from the nearest neighbor approach.

All three methods are applied to three different workspace with different targets and obstacle conditions. For the cell decomposition method, two ex-

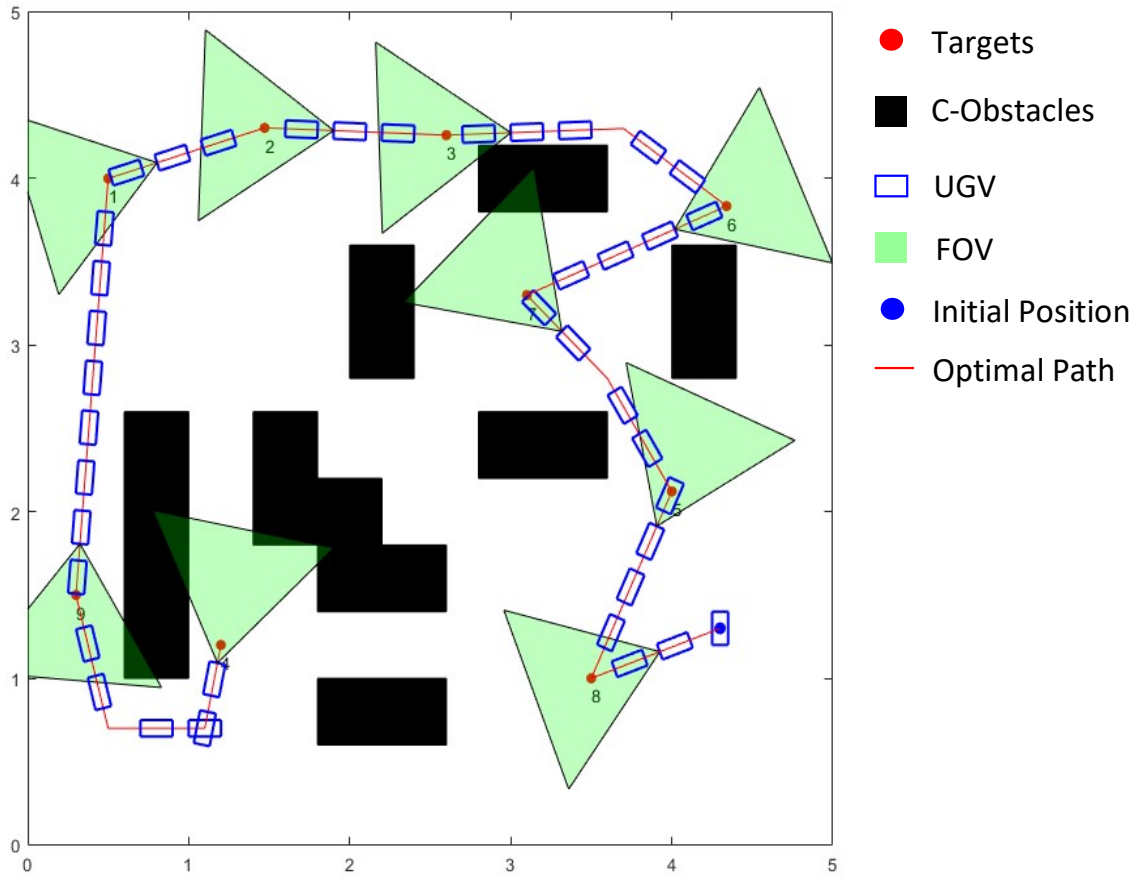

Figure 6.8: The optimal path obtained from 2-opt heuristics for Map 2. The sensor need to go to each target to obtain measurement and therefore the performance is affected. The total translation is 14.0m and total rotation is 12.7 rad.

treme cases of $w_t = 0.1$ and $w_t = 0.9$, are simulated to analyze the performance when either rotation or translation dominates. All three methods are evaluated in terms of rotation time and translation time. The results are shown in Table 6.1.

Table 6.1: Performance Comparison in Map 1, 2 and 3

Map1	Cell ($w_t = 0.1$)	Cell ($w_t = 0.9$)	NN	Heuristics
Translation (m)	15.1	11.2	13.1	13.2
Rotation (rad)	9.3	12.4	16.0	10.0

Map2	Cell ($w_t = 0.1$)	Cell ($w_t = 0.9$)	NN	Heuristics
Translation (m)	14.0	11.1	16.7	14.0
Rotation (rad)	11.8	14.7	14.0	12.7

Map	Cell ($w_t = 0.1$)	Cell ($w_t = 0.9$)	NN	Heuristics
Translation (m)	48.6	39.2	60.8	45.2
Rotation (rad)	38.0	58.5	45.1	42.2

Compared with two benchmark methods that considers only translation, the proposed method obtains a path with smaller cost with a large w_t due to its ability to encourage the UGV to rotate more to save translation distance. Consider the path between target 6 and 7 in Figure 6.7 and Figure 6.8. The "shake head" behavior of the sensor discussed in section 6.1 reduces the translation distance compared with the heuristics. Overall, the proposed method achieves 13% to 16% improvement in performance. The method with the worst performance is the "nearest neighbor" approach, because the sensor greedily moves to the nearest target and may travel back and forth to visit all the targets. Moreover, the proposed approach is able to deal with the case where rotation cost dominates ($w_t = 0.1$), while the benchmark methods do not consider rotation and

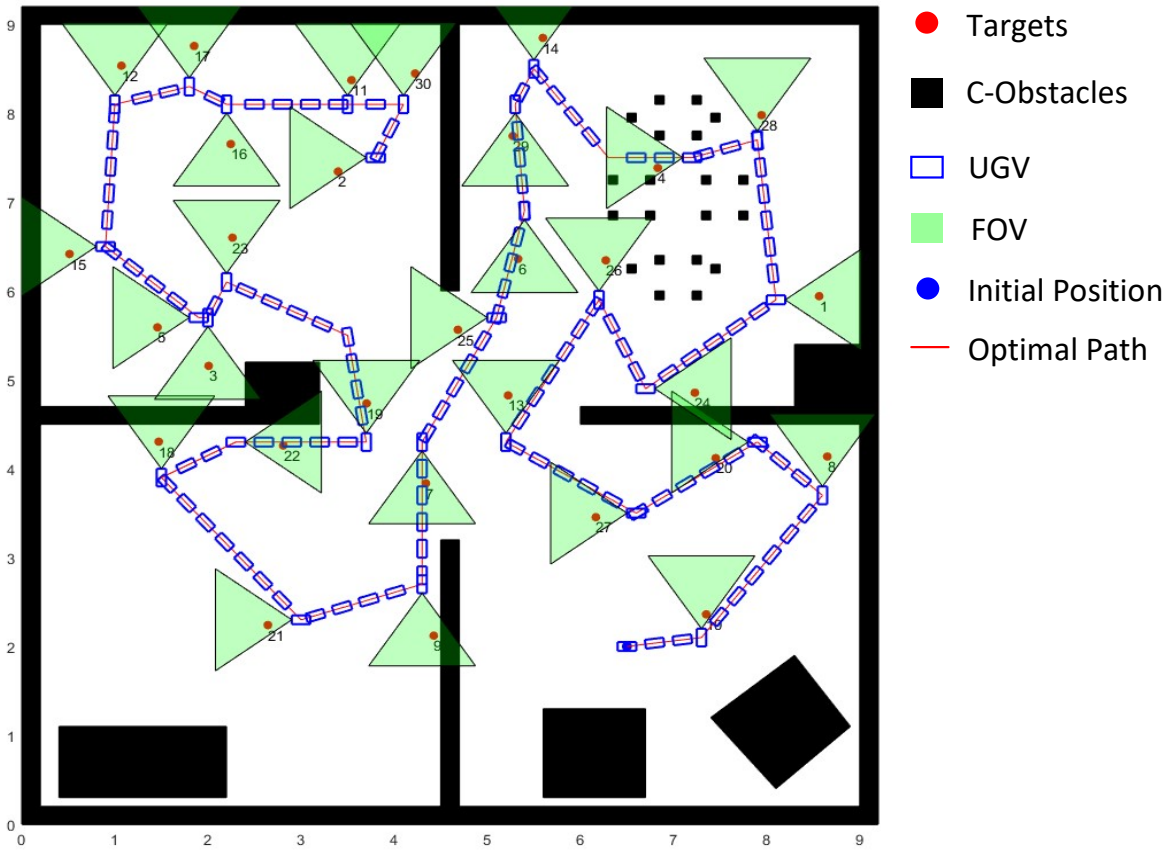

Figure 6.9: The optimal path obtained from cell decomposition and integer programming with $w_t = 0.9$ for Map 3. The total translation is 39.2m and total rotation is 58.5 rad.

the performance is therefore affected. Another important observation is the influence of obstacles on path planning. In Map 1, the number of obstacles is relatively small, and most target pairs are placed relatively close to each other. The benchmark methods can still obtain a path with relatively low translation cost. In Map 2 where more obstacles are present, the performance of the heuristics is affected and so is the nearest neighbor method. The cell decomposition approach applied in this thesis is able to path a path with low cost under all circumstances.

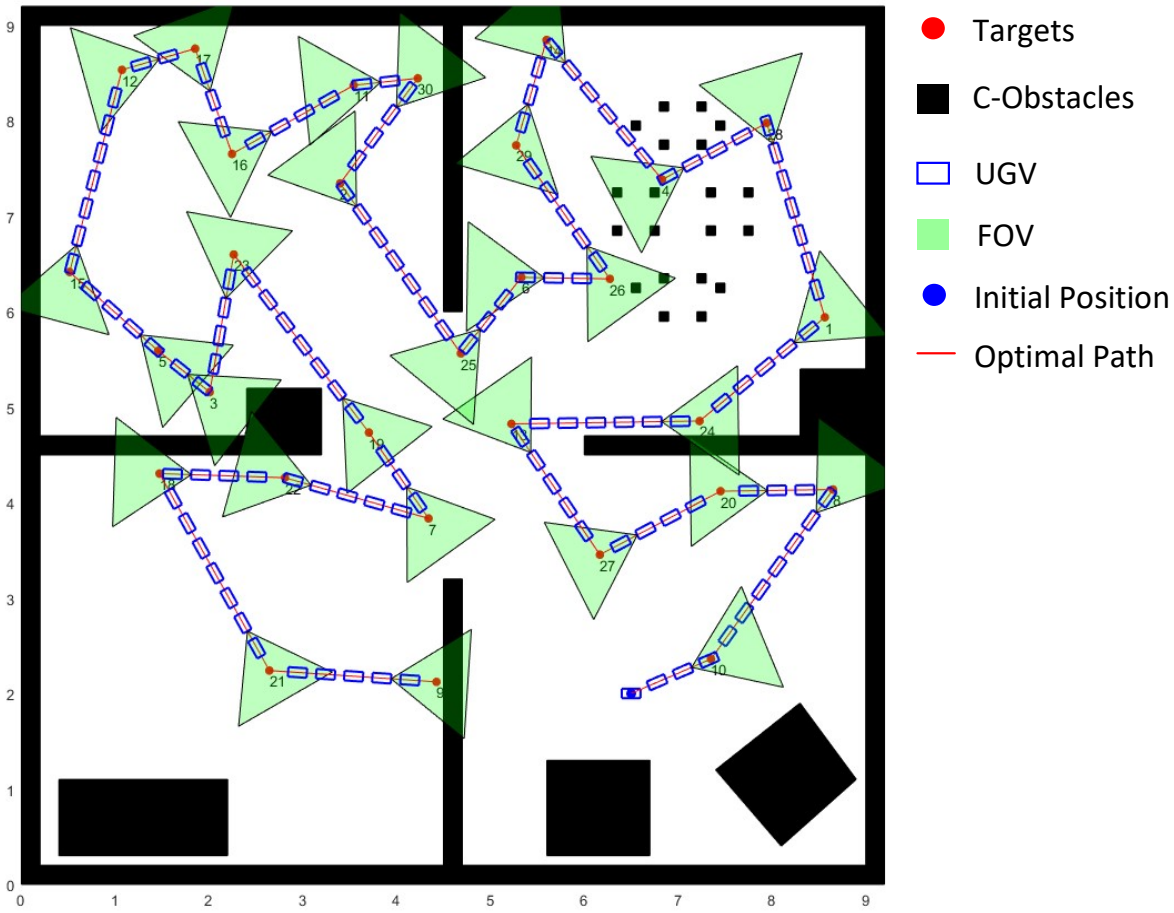

Figure 6.10: The optimal path obtained from ”2-opt” heuristics for Map 3. The sensor need to go to each target to obtain measurement and therefore the performance is affected. The total translation is 45.2m and total rotation is 42.2 rad.

CHAPTER 7

CONCLUSION AND FUTURE STEPS

In this thesis, a methodology based on cell decomposition and integer programming is developed for planning the path for directional sensor platforms to classify targets in the workspace with minimum time. The objective of measuring targets is transformed into visiting the corresponding observation cells obtained from cell decomposition. A closed-form definition of directional C-Target is proposed and can be implemented with linear complexity. An integer programming approach is developed to solve the path planning problem in the connectivity graph. The methodology is flexible to incorporate different weights on translation and rotation and the running time can be controlled by the maximum permissible cell size. Simulations in different workspace with different connectivity characteristics has shown the effectiveness of this method and the relatively better performance compared to the nearest neighbor method and classical TSP formulation.

The running time of the path planning methodology increases with the number of cells. One of the future direction is making the methodology scalable to larger workspace. The integer programming approach can also be combined with other path planning techniques. For example, the probabilistic roadmap method (PRM) picks a collection of random configurations in $\mathcal{C}_{free}$ as milestones, and the connectivity graph is constructed with milestones as nodes. Compared to cell decomposition, PRM is not resolution complete, but is more scalable to larger workspace. The connectivity graph obtained from PRM can also be solved via integer programming to classify all the targets. Analog to the notion of observation cells, the milestones where the sensor can measure tar-

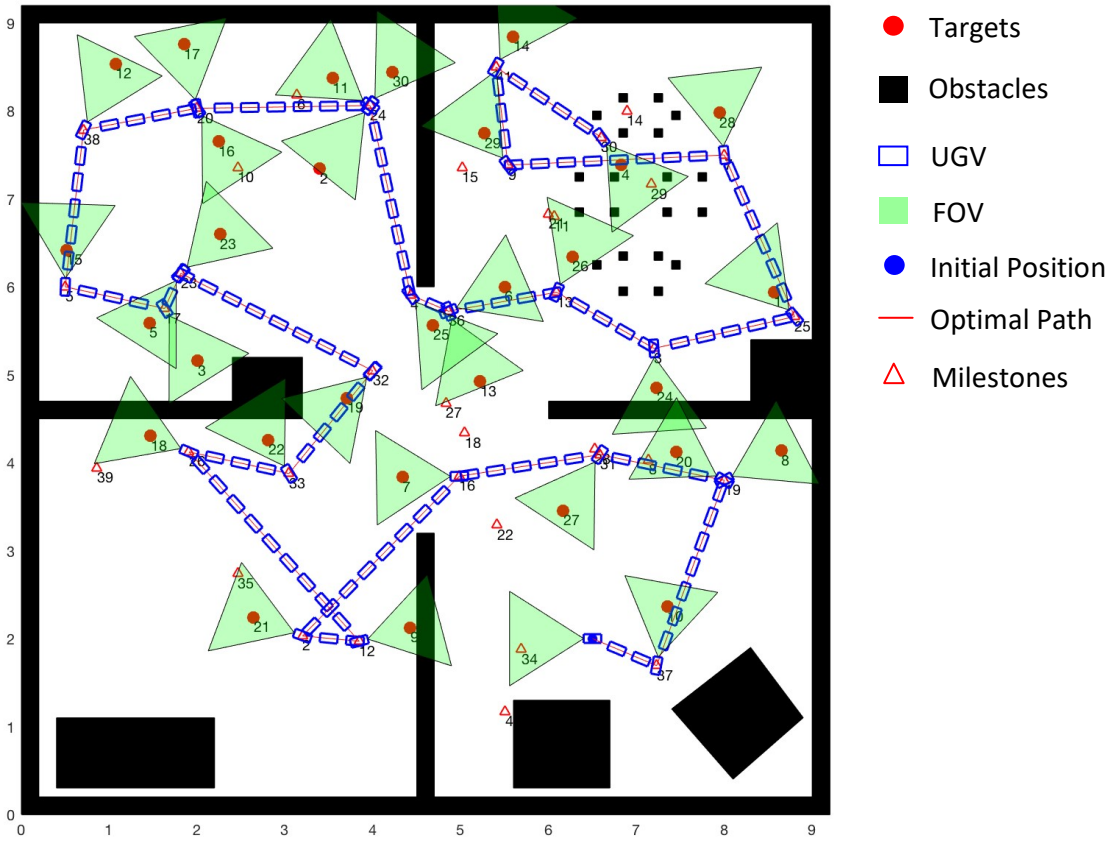

Figure 7.1: The optimal path obtained for Map 3 with PRM and integer programming approach. The milestones are shown by triangles. The path planning algorithm generates the connectivity graph from the milestones, and produce a path that satisfies the sensing objective with minimum time. In this example, only translation cost is taken into consideration.

gets are found out and the index set $\mathcal{O}_i$ is generated in a similar pattern. The path planning algorithm generates the connectivity graph from the milestones, and produce a path that satisfies the sensing objective with minimum time. In this example, only translation cost is taken into consideration. The optimal path is shown in Figure 7.1. The combination of integer programming and PRM demonstrates the flexibility of the integer programming technique, and future work of applying it to sensor path planning would have a broad perspective.

BIBLIOGRAPHY

[1] Ercan U Acar, Howie Choset, Yangang Zhang, and Mark Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International journal of robotics research*, 22(7-8):441–466, 2003.

[2] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[3] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[4] Rodney A Brooks and Tomas Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):224–233, 1985.

[5] Chenghui Cai and Silvia Ferrari. A q-learning approach to developing an automated neural computer player for the board game of clue®. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2346–2352. IEEE, 2008.

[6] Chenghui Cai and Silvia Ferrari. Information-driven sensor path planning by approximate cell decomposition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3):672–689, 2009.

[7] Chester F Carlson. Lecture 10: Hough circle transform. *Rochester Institute of Technology: Lecture Notes*, 2005.

[8] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.

[9] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.

[10] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209. Springer, 1998.

[11] Gianpaolo Conte and Romolo Zulli. Hierarchical path planning in a multi-robot environment with a simple navigation function. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4):651–654, 1995.

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[13] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[14] Mohamed Elbanhawi, Milan Simic, and Reza N Jazar. Continuous path smoothing for car-like robots using b-spline curves. *Journal of Intelligent & Robotic Systems*, 80(1):23–56, 2015.

[15] Bernard Faverjon. Object level programming of industrial robots. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1406–1412. IEEE, 1986.

[16] Gerd Gigerenzer and Wolfgang Gaissmaier. Heuristic decision making. *Annual review of psychology*, 62:451–482, 2011.

[17] Greg Hager and Max Mintz. Computational methods for task-directed sensor data fusion and sensor planning. *The International Journal of Robotics Research*, 10(4):285–313, 1991.

[18] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[19] Christian Hofner and Günther Schmidt. Path planning and guidance techniques for an autonomous mobile cleaning robot. *Robotics and autonomous systems*, 14(2-3):199–212, 1995.

[20] Paul VC Hough. Machine analysis of bubble chamber pictures. In *International conference on high energy accelerators and instrumentation*, volume 73, page 2, 1959.

[21] Wesley H Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 27–32. IEEE, 2001.

[22] Yong K Hwang and Pang C Chen. A heuristic and complete planner for the classical mover's problem. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 1, pages 729–736. IEEE, 1995.

[23] Paulo A Jimenez, Bijan Shirinzadeh, Ann Nicholson, and Gursel Alici. Optimal area covering using genetic algorithms. In *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pages 1–5. IEEE, 2007.

[24] David S Johnson and Lyle A McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1:215–310, 1997.

[25] Michael Jünger and Denis Naddef. *Computational combinatorial optimization: optimal or provably near-optimal solutions*, volume 2241. Springer Science & Business Media, 2001.

[26] Chen Junhua and Lei Jing. Research on color image classification based on hsv color space. In *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, pages 944–947. IEEE, 2012.

[27] Subbarao Kambhampati and Larry Davis. Multiresolution path planning for mobile robots. *IEEE Journal on Robotics and Automation*, 2(3):135–145, 1986.

[28] Carolyn Kimme, Dana Ballard, and Jack Sklansky. Finding circles by an array of accumulators. *Communications of the ACM*, 18(2):120–122, 1975.

[29] Chris M Kreucher, Keith D Kastella, and Alfred O Hero. Multi-platform information-based sensor management. In *Defense Transformation and Network-Centric Systems*, volume 5820, pages 141–152. International Society for Optics and Photonics, 2005.

[30] André Langevin, François Soumis, and Jacques Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127–132, 1990.

[31] C Laugier. An adaptive collision-free trajectory planner. In *Proc. Int. Conf. on Advanced Robotics*, 1985.

[32] Eugene L Lawler, Jan Karel Lenstra, and Alexander HG Rinnooy Kan. The traveling salesman problem. 1985.

[33] Xuejun Liao and Lawrence Carin. Application of the theory of optimal experiments to adaptive electromagnetic-induction sensing of buried targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):961–972, 2004.

[34] Lanny Lin and Michael A Goodrich. Uav intelligent path planning for wilderness search and rescue. In *Intelligent robots and systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 709–714. IEEE, 2009.

[35] Yugang Liu and Goldie Nejat. Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165, 2013.

[36] Wenjie Lu, Guoxian Zhang, and Silvia Ferrari. An information potential approach to integrated sensor path planning and control. *IEEE Transactions on Robotics*, 30(4):919–934, 2014.

[37] P Miliotis. Integer programming approaches to the travelling salesman problem. *Mathematical Programming*, 10(1):367–378, 1976.

[38] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[39] Bryan S Morse. Lecture 15: Segmentation (edge based, hough transform). *Brigham Young University: Lecture Notes*, 2000.

[40] Christian Nilsson. Heuristics for the traveling salesman problem. *Linkoping University*, pages 1–6, 2003.

[41] Hanna Oh, Jeffrey M Beck, Pingping Zhu, Marc A Sommer, Silvia Ferrari, and Tobias Egner. Satisficing in split-second decision making is characterized by strategic cue discounting. *Journal of experimental psychology: learning, memory, and cognition*, 42(12):1937, 2016.

[42] Hanna Oh-Descher, Jeffrey M Beck, Silvia Ferrari, Marc A Sommer, and Tobias Egner. Probabilistic inference under time pressure leads to a cortical-to-subcortical shift in decision evidence integration. *NeuroImage*, 162:138–150, 2017.

[43] Timo Ojala, Matti Pietikäinen, and David Harwood. Performance evaluation of textures measures with classification based on kullback discrimina-

tion of distributions. In *Proceedings of the International Conference on Pattern Recognition (ICPR94)*, pages 582–585, 1994.

[44] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[45] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[46] Joseph O'rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

[47] Manfred Padberg and Ting-Yi Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52(1-3):315–357, 1991.

[48] Abraham Punnen, FrancoiS Margot, and SantoSh Kabadi. Tsp heuristics: domination analysis and complexity. *Algorithmica*, 35(2):111–127, 2003.

[49] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936–1941. IEEE, 2002.

[50] Mohamed Rizon, Yazid Haniza, Saad Puteh, Ali Yeon, Md Shakaff, Saad Abdul Rahman, Masanori Sugisaka, Yaacob Sazali, Mamat M Rozailan, and M Karthigayan. Object detection using circular hough transform. 2005.

[51] N Ruangpayoongsak, H Roth, and J Chudoba. Mobile robots for search and rescue. In *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*, pages 212–217. IEEE, 2005.

[52] Thomas C Shermer. Recent results in art galleries (geometry). *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[53] Rob Siegel. Land mine detection. *IEEE instrumentation & measurement magazine*, 5(4):22–28, 2002.

[54] Thierry Siméon, Stéphane Leroy, and J-P Lauumond. Path coordination

for multiple mobile robots: A resolution-complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42–49, 2002.

[55] Herbert A Simon. A behavioral model of rational choice. *The quarterly journal of economics*, 69(1):99–118, 1955.

[56] Herbert A Simon. Rational choice and the structure of the environment. *Psychological review*, 63(2):129, 1956.

[57] Herbert A Simon. Invariants of human behavior. *Annual review of psychology*, 41(1):1–20, 1990.

[58] Jorge Urrutia. Art gallery and illumination problems. In *Handbook of computational geometry*, pages 973–1027. Elsevier, 2000.

[59] Hongwei Wang, Yong Ma, Yong Xie, and Min Guo. Mobile robot optimal path planning based on smoothing a* algorithm. *Journal of Tongji University (natural science)*, 38(11):1647–1650, 2010.

[60] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39. IEEE, 2009.

[61] H Paul Williams and Sally C Brailsford. Computational logic and integer programming. *Advances in Linear and Integer Programming*, 4:249–281, 1996.

[62] Simon X Yang and Chaomin Luo. A neural network approach to complete coverage path planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):718–724, 2004.

[63] Namik Kemal Yilmaz, Constantinos Evangelinos, Pierre FJ Lermusiaux, and Nicholas M Patrikalakis. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering*, 33(4):522–537, 2008.

[64] Guoxian Zhang, Silvia Ferrari, and M Qian. An information roadmap method for robotic sensor path planning. *Journal of Intelligent and Robotic Systems*, 56(1-2):69–98, 2009.

[65] DJ Zhu and J-C Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7(1):9–20, 1991.