SURVEILLANCE MISSION PLANNING FOR UAVS IN GPS-DENIED URBAN ENVIRONMENT

WANG PENGFEI

(*B. Eng., HUST*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2016

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Wang Pengfei

Wang Pengfei

25 November 2016

ACKNOWLEDGEMENT

I would like to express my appreciation and thanks to my supervisor, A/Prof. Zhang Yunfeng, for his encouragement during my PhD study and guidance to me on my way of research.

I would also like to thank National University of Singapore to offer me scholarship to fulfil my PhD research in four years.

I would also like to thank Dr. Gen Ling, Dr. Wang Jingjing, Dr. Cui Jinqiang, Dr. Wang Fei for their advices in my research.

I would also like to thank Ms. Lan Lijun, Mr. Qin Hailong, Mr. Hu Yuchao, Mr. Shan Mo, Mr. Chen Xudong, Dr. Phang Swee King for their kind help in the programming and the experiments.

I would also like to thank Dr. Lin Feng and Dr. Teo Swee Huat, Rodney for their supporting in the experiments and providing the workplace in the Temasek Laboratories @ NUS.

I would like to thank my family for their love and support throughout my whole PhD student life.

Finally, I also want to express my thanks to Dr. Wang Sibao, Mr. Liu Ning, Mr. Ren Kai, and all my other fellow students, for their encouragement to help me to face with difficulties during the four years.

TABLE OF CONTENTS

Abstract

In the past decade, both research community and industry have witnessed the rapid development of unmanned aerial vehicle (UAV) and its wide applications in different kinds of missions. In this thesis, the issues involved in the mission planning of UAVs for city surveillance have been studied. In this thesis, the research includes two major parts.

Firstly, a mission planning system is developed that generates mission plans for a group of fixed-wing UAVs with on-board Gimbaled cameras to provide continuous surveillance over an urban area. Given the map including terrain and buildings in the target area, a two-stage approach is employed to solve the problem. In the first stage, a set of camera locations called the vantage set is generated that provides complete coverage of the target area. In the second stage, one or several UAVs are determined to collectively share the vantage set and their individual paths are generated to carry out the continuous surveillance duty. In both stages, evolutionary algorithms are developed to search for the optimal solution. During the search, realistic constraints such as the flying capabilities of UAVs and collision avoidance are imposed to guarantee the feasibility of the final result.

Secondly, the problem of perching location selection (as part of perch-and-stare surveillance mission) for rotary-wing UAVs in a GPS-denied environment is studied. In this kind of mission, a UAV is dispatched to perch on a roof of a building to keep surveillance on a given target. The tasks are (1) to select a perching location form the existing offline map of the target for the UAV, and (2) to identify the selected perching position from the reconstructed online map (in the form of 3D point cloud) and check its feasibility for landing. Two algorithms, one offline and one online, have been developed to deal with these two tasks, respectively. For offline perching location selection, a set of realistic geometric constraints are considered, including the dimension of the UAV, position of on-board camera, landing area requirement, slope of roof, camera range, and line of sight from candidate perching location to the target. For online perching location identification, the online reconstructed map in the form of 3D point cloud (covering the selected perching location) is assumed available. The roof contours are firstly extracted and constructed. They are then matched with the offline map to locate the selected perching location. Finally, the feasibility of the landing zone (around the landing location) is evaluated.

The proposed algorithms to UAV surveillance mission planning (fixed-wing and rotary-wing) have been implemented and tested. It represents an important step towards achieving autonomous planning in UAV surveillance missions.

LIST OF TABLES

LIST OF FIGURES

LIST OF GLOSSARY

Symbols:

γ_v: angle of view

α: elevation angle in camera model for fixed-wing UAV

β: azimuth angle in camera model for fixed-wing UAV

θ: view orientation in camera model for fixed-wing UAV

θ_o: optimal view orientation in camera model for fixed-wing UAV

R_t: turning radius

$\theta_{C,D}$: climbing/diving angle

R_{min}: minimal distance from camera to surface point

R_{max}: maximal distance from camera to surface point

d_{max}: maximal sensor range of the UAV in the offline perching mission

A_{min}: minimal required landing area for the UAV

$\theta_{s,max}$: the maximum allowable slope angle of the landing zone for the UAV

d_s: safety distance to the edge of the roof for perching

h_{min}: minimal height of preselected roof for landing

h_{max}: maximal height of preselected roof for landing

λ_0, λ_1, λ_2: local normal of the point in a point cloud

λ_p: inclination angle of a plane fitted for the neighbour of a point

σ_p: roughness of a plane fitted for the neighbour of a point

r_{es}: resolution of conversion from point cloud to binary image

row_k, col_k: the pixel of k-th point of a contour

I_c: the image of extracted contours

θ_p: main orientation of an image

I_t: template image in matching

I_r: reference image in matching

γ: correlation matrix

$\boldsymbol{C_b}$: minimal bounding circle

a_k, b_k: the column and row number of the k^{th} effective pixel

r_{pa}: difference between peak and average in correlation matrix

r_{fspd}: difference between peak and second peak in correlation matrix

α_h: parameter in the algorithm of α-hull

Abbreviations:

UAV: unmanned aerial vehicle

3D: three dimension

2D: two dimension

AOV: angle of view

cP: camera position

sP: surface point

vD: view direction

vP: vantage point

BFS: brute-force search

DEM: digital elevation map

FOV: field of view

GA: genetic algorithm

MTSP: multiple travelling salesman problem

PSO: particle swarm optimization

TSP: travelling salesman problem

VRP: vehicle routing problem

LOS: line of sight

POI: point of interest

FOI: face of interest

CPE: candidate perching edge

FPE: feasible perching edge

CPP: candidate perching point

FPP: feasible perching point

LiDAR: light detection and ranging

OSM: open street map

PCA: principal component analysis

CHAPTER 1

INTRODUCTION

An unmanned aerial vehicle (UAV) is a kind of aircraft without a pilot. It can be remotely controlled or fly autonomously (Yanushevsky 2011). It is usually equipped with different kinds of devices performing data capture, navigation, fly control, and all kinds of missions. The most popular UAVs are classified into two categories: fixed-wing UAVs and rotary-wing UAVs, as shown in Figure 1.1.

(a) A fixed-wing UAV

(b) A rotary-wing UAV

Figure 1.1 Commonly used UAVs

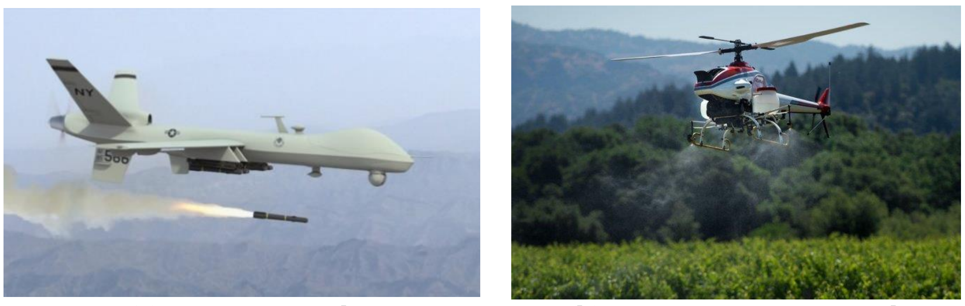
Over the past decade, both research and industry communities have witnessed the rapid development of UAV and its wide application in different kinds of missions, for its advantages of light weight, low cost, high manoeuvrability, and so on. In military applications, UAVs can be utilized for missions such as enemy target attack, ground convoy protection, boundary loitering, etc. In civil applications, UAVs can be used in geological survey, city monitoring, search and rescue, agriculture, even in logistics. One of the important missions, both military and civil, is city surveillance. For example, a UAV can be used to fly over an enemy city to collect military intelligence. A

UAV can also be deployed to monitor a region/target continuously in urban environment to detect crime.

In this chapter, the applications of UAVs are firstly briefly introduced, followed by a more detailed description of their applications in city surveillance. Then, by reviewing the state-of-the-art research of UAVs in city surveillance, the motivation of the thesis is presented. Furthermore, the research objectives and scope are descripted in details. Finally, the organization of the thesis is outlined.

1.1. Unmanned Aerial Vehicle (UAV) and Its Applications

The first UAV was invented in England in 1917. Almost thirty years later, the V-1 UAV was created and used to attack civil targets in World War Ⅱ. In 1995, the Ryan Firebee was produced to carry out reconnaissance task in American army. Since the War in Afghanistan, as the rapid development of electronic information technology and aeronautical engineering, the applications of UAVs have been on a rise in both military and civil missions.



(a) UAV in attack (b) UAV in spraying work

Figure 1.2 Examples of UAV applications

In military scenarios, UAVs can be used in convey protection for ground vehicles (Ding et al. 2010). Some fixed-wing UAVs were designed to load weapons to execute attack mission (Min et al. 2013) as shown in Figure

1.2a. In civil scenarios, existing and potential applications of UAVs include surveillance, structure inspection, aerial photography, exploration, search and rescue, etc. In agriculture, UAVs have been used to do spraying work (Huang et al. 2008) as shown in Figure 1.2b. Compared to the manual spraying, using UAVs can protect farmers from poison in the pesticide. In addition, UAVs can obtain high-resolution image of the farm land, contributing to monitoring the farmland and acquiring the vegetation growth status. Exploration and mining is another important application area of UAVs, as the mineral resources are usually located in areas such as mountains, deserts, deep sea, etc. Some fixed-wing UAVs also serve in meteorological reconnaissance work (Lin and Lee 2008). For example, UAVs can be launched into the eye of the typhoon to collect data for scientific researches. In civil engineering, UAVs with payload can undertake the job of architecture construction (Willmann et al. 2012). Some researchers even proposed the idea to use UAVs in autonomous logistics (Dogo et al. 2011). Recently, Amazon planned to use UAVs to perform express delivery service and some preliminary tests have been carried out.

1.2. Application of UAVs in City Surveillance

One important task among all the applications of UAVs is city surveillance, which has drawn increasing attentions recently. In city surveillance mission, UAVs are widely used to conduct target searching, target tracking, or information gathering. In such a mission, the UAV equipped with different kinds of sensors are dispatched to cover an area, which is called continuous surveillance. For example, in (Flint et al. 2002, Tisdale et al. 2009), the problem was investigated for a team of UAVs to search for a stationary target. UAVs are also used to track moving ground targets in urban area (Kim and Kim 2008). In such kind of missions, the fixed-wing UAVs are more suitable than rotary-wing UAVs due to longer endurance and higher flying speed, etc. However, there are also some challenges to be faced. For example, airport is indispensable for fixed-wing UAVs to take off and land. In addition, the planned flying path of the UAV should meet the criteria of the dynamic constraints such as turning radius, climbing/diving angle, etc. Moreover, hazard cannot be ignored when the UAV is dispatched to do surveillance in an enemy city.

On the other hand, in urban security management, there exist some regions/targets that are difficult to reach but need to be monitored continuously, such as traffic accident spot, and communal facilities to be inspected. In such situations, a UAV can be deployed to reach the target region and land at a feasible spot, preferably on roof of buildings. Its on-board camera can then be used to monitor the target and send information back to the control station. This kind of mission is called perch-and-stare surveillance, during which GPS is usually unavailable. Besides, UAVs are also dispatched to conduct remote inspection, such as industrial plant inspection (Fumagalli et al. 2012), power line inspection (Wang et al. 2010), etc. In such applications, the perching location for the UAV is to be selected, and the object should be within the view of the UAV. Rotary-wing UAVs are more suitable for such missions due to its smaller size and more flexible flying abilities. For example, no substantial landing/take-off area is required for a rotary-wing UAV for the sake of its vertical take-off and landing ability. In addition, its hovering ability and its small size make it well suitable to land on the roof of a building to perform the perch-and-stare mission. Challenges cannot be avoided in such mission, for example, there are lots of buildings or other objects blocking the line of sight. In addition, GPS is often denied in the urban environment, so a map is to be built by the sensors onboard the UAV. However, the available sensors are limited by the payload of the rotary-wing UAV.

1.3. State-of-the-art of Researches on UAV in City Surveillance

1.3.1. Coverage path planning for continuous city surveillance with fixed-wing UAVs

In continuous city surveillance mission, the route of the UAV needs to be planned from one site to another, minimizing the path length or completion time, while satisfying various constraints, such as collision avoidance, time window, visiting sequence and so on. This kind of problem is called UAV path planning, which has drawn much attention.

The UAV path planning problem is similar to the travelling salesman problem (TSP) in vehicle route planning, except that the UAVs are subject to flying constraints. If more than one UAV is used in surveillance mission, the cooperation and task allocation need to be considered. This kind of problem is called multi-UAV path planning, which is similar to the multi-travelling salesman problem (MTSP). Some researchers considered additional constraints, such as time window (Lum and Rysdyk 2008), different tasks (Bertuccelli et al. 2004), and so on. This kind of more complicated problem is similar to the vehicle routing problem (VRP). A method of mixed integer linear program (MILP) was developed to solve the problem of UAV VRP with time window (Weinstein and Schumacher 2007).

In the case of threat existence in the target area, the problem is termed as threat-aware path planning. There are various kinds of threat, such as enemy radars, missiles, or terrain obstacles. Sometimes, UAVs fly close to the terrain of the target area, where there are some aerial obstacles, such as mountains. In such a situation, a collision-free path needs to be generated. Some researchers utilized evolutionary algorithm to optimize the path without terrain collision (Hasircioglu et al. 2008). Others used graph-based method to find feasible paths (Shanmugavel et al. 2010). However, in other situations, the UAV must penetrate into the threat area, and it is impossible to avoid the threat totally. To deal with this kind of problem, the probabilistic model was proposed to evaluate the threat (Pfeiffer et al. 2005). The optimization of the path is a trade-off between threat and flying time. The mathematical model of threat from enemy radar was utilized to guide the planning of stealthy path (May et al. 2010).

In other missions, a UAV is tasked to cover a target area to collect information. In such mission, sensor coverage is a critical problem. There are always buildings with relatively high densities, bringing about occlusion to sensor coverage. Such problem is called “coverage path planning”, which has also drawn much attention. Much reported works focused on the decomposition of target area with pre-defined path pattern. Some researchers proposed the algorithm of Boustrophedon Cellular Decomposition (De Carvalho et al. 1997), where the robot move back and forth to cover the whole area. Others utilized similar approach to cover target areas with complex topology (Oksanen and Visala 2009, Li et al. 2011). Ahmadzadeh et al. (2008) proposed to optimize the UAV path aiming at maximizing the coverage area in critical time interval, where there was a fixed camera on-board the UAV, and the field of view was a quadrilateral changing with the UAV posing angle.

Some researchers considered the 3D structure of the target area, where the line of light from the UAV to some ground points may be occluded by buildings (Jakob et al. 2010). UAVs with on-board Gimbaled camera were also utilized in city surveillance for 3D urban structure coverage (Cheng et al. 2008). In our previous research, a problem of continuous coverage on an urban area with a group of UAVs has been investigated. The problem was decomposed into two stages. In the first stage, a set of camera locations are determined to cover the whole urban area. In the second stage, the optimal path was fitted to the go through all the camera locations (Geng et al. 2013). In their work, the camera is assumed to be facing downward. If the camera with pan-tilt ability is implemented, a more optimized solution will be needed for this problem. In addition, there is potential hazard threat, such as missiles and enemy radars, in city surveillance mission.

1.3.2. Landing location selection for rotary-wing UAVs in perch-and-stare mission

In the perch-and-stare mission, the UAV is dispatched to perch on a spot to conduct enduring reconnaissance on a target region. In the mission, the most important task is to select a location for landing. The selection problem can be categorized into two types, offline selection and online selection. In offline selection, an existing 3D map of the target region (such as google map or

OpenStreetMap) is available, and the major task is to select a perching position for the UAV in the map. The map usually has a large coverage, but the accuracy is not high enough. In addition, the dynamic process of landing is not taken into consideration. On the other hand, in online selection, the UAV is assumed hovering above the landing location (selected in offline selection). The task is to identify this selected landing position from the map constructed in real time based on the data collected by the sensor on-board the UAV. The range of the map is limited, but the map consists of detailed 3D structures of the surrounding environment. As the map is updated dynamically, the online landing location selection process is carried out iteratively during the landing process of the UAV.

- **Offline perching location selection for rotary-wing UAVs**

In the perch-and-stare mission, different kinds of criteria for perching location selection have been proposed, which can be categorized into two groups, geometric criteria and mission criteria. Here are some commonly used geometric criteria. For example, the landing location should be large enough without too much slope and roughness and absence of obstacles (Takahashi et al. 2013). Others calculated static stability for helicopter based on gravity distribution (Scherer et al. 2012). While, mission criteria depend on individual tasks. Park et al. (2015) utilized energy consumption as a criterion in indoor landing site selection for quadrotor. Mejias, Fitzgerald et al. (2009) considered gliding distance and human injuries avoidance in forced landing task for fixed wing UAV.

- **Online landing location selection for rotary-wing UAVs**

There have been much reported works on the problem of online autonomous landing location selection for UAVs. Different kinds of sensors have been used and different types of landing locations were addressed. Based on the types of on-board sensor and landing location, the related works can be classified as follows:

(1) *Landing on a known stationary target in a GPS available environment*. Shakernia et al. (2002) developed a system to guide a rotary-wing UAV to land on a designed target. The landing target is predesigned as a piece of paper with monochrome blocks. The navigation is carried out based on the signal from the GPS and IMU. Threshold and corner detection are implemented to identify the position of the landing target. The major drawback is that the landing target detection algorithm can only be applied for certain targets with grayscale images. Kim et al. (2003) added barometer to estimate the height of the UAV besides the GPS receiver, as GPS signal is sometimes missing, e.g., in the case of occlusions. Data from other sensors, such as IMU are often combined to constrain the drift of GPS estimation (George and Sukkarieh 2005). The accuracy comparison of landing with and without IMU compensation demonstrated the positive effect of IMU. Rackliffe et al. (2011) combined geographic information system (GIS) information with sensor data to construct a probability map, then the feasible landing site can be selected based on the map.

(2) *Landing on a known stationary target with camera in GPS-denied environment*. Some researchers used a camera to guide the helicopter

to land on a helipad marker (Saripalli et al. 2003). The colour image is converted to binary by thresholding. The algorithm is restricted by the constraints that the camera is fixed downward and perpendicular to the ground, and the assumption that the intensity value between the helipad and the surrounding environment is large enough. In most marker based landing algorithms, the intrinsic camera parameters are calibrated and fixed before the mission. In (Santamaria-Navarro and Andrade-Cetto 2013), the authors utilized uncalibrated camera to estimate the position of the UAV and guide the landing of the UAV. The focal length is estimated by building a relation between the camera and the frame of the landing target. Verbandt et al. (2014) developed a new marker detection algorithm for autonomous landing of VTOL UAVs based on edge detection and adapted Hough transform, and a binary classifier is trained based on support vector machine to verify the robustness against varying conditions on occlusion, sharpness, and lighting. The experiments on a quadrotor UAV demonstrated its ability for precision landing on a circular landmark. Other researchers utilized monocular camera to simulate the stereo vision by collecting images along one direction parallel to the ground (Sereewattana et al. 2014). Four different colour circles made up the marker, the positions of which were extracted by Hough transform. The height of the UAV is determined by stereo vision algorithm. However, the assumption of fixed camera direction may not be guaranteed. Different types of markers can be utilized for vision tracking in autonomous landing of UAV, such as a pattern of several LEDs, a Bayer pattern of table tennis balls, or a “H” shape landing pad (Masselli et al. 2014). Flying experiments were conducted to compare the three types of landing pads. All the three methods have the comparable accuracy.

(3) *Landing on an unknown target in GPS allowed environment*. In (Scherer et al. 2012), a system was developed to conduct hazard detection and avoidance during the landing of a UAV. The system consists of a GPS receiver, a monocular camera, and a laser altimeter. Structure from motion algorithm is applied to create the terrain map from the image sequences collected by the single camera. Once a safe site is detected in the terrain map, the UAV is guided to the site by the GPS and laser altimeter. The accuracy of landing is good, however, the laser altimeter is very heavy that it can only be applied on a full scale helicopters.

(4) *Landing on an unknown area with camera in GPS-denied environment*. Garcia-Pardo et al. (2002) used a single colour camera to detect safe landing places, based on the idea that the contrast at the boundary of obstacles is higher than flat areas. The implementation of the algorithm is restricted by its assumption that the terrain is flat. Theodore et al. (2006) introduced a vision based autonomous landing system for rotary-wing UAV. The algorithm was developed for terrain map reconstruction and landing zone detection. Flight trials were conducted on Yamaha RMAX helicopter. The terrain map is constructed by determining the disparities between features in two consecutive images collected by stereo camera. The optimal landing point can be selected from the map and the point is tracked during the descent of the UAV.

Some researchers utilized stereo camera to guide the UAV to land on an unknown area (Park et al. 2015). The flatness map is constructed from the depth image collected by the stereo vision system. Others proposed to use a single downward looking camera and an IMU for landing spot selection for micro UAV (Forster et al. 2015), in which a 2D elevation map is constructed by triangulate images from multiple views using recursive Bayesian estimation.

(5) *Landing on an unknown area with laser range finder in GPS-denied environment*. Whalley et al. (2009) presented a system of low altitude navigation and landing for helicopter UAVs. The algorithm for navigation is based on quasi 3D A* route planning approach. The landing site selection is based on the map constructed by the 3D point cloud collected by the LiDAR on-board the UAV. Fly tests have been carried out on a Yamaha RMAX helicopter. Sevcik et al. (2010) explored the problem of landing site detection and selection in the environments obscured by smoke for UAV with a laser range finder. The terrain map was constructed from the point cloud collected by the laser. Then, the irregularly spaced terrain map was converted to a regularly spaced DEM, which is separated into a slope map and a roughness map in a further step. Finally, the most suitable landing zone is selected based on the two maps. The algorithm was tested offline with the collected data by the UAV. It was found that the existence of smoke has bad effect on the performance of landing zone detection.

For most realistic applications, there are no markers to guide the UAV to land. In addition, GPS signal is also not available in many scenarios. Laser range finder can build a relatively high resolution map than camera, so laser is more suitable for achieving accurate landing. In the perch-and-stare surveillance mission, the UAV is tasked to land on a preselected roof spot to keep a line of sight to a target. So the landing location is not only selected from the online collected data, but also should be matched with the preselected location from offline perching location selection. Based on our knowledge, there has been no reported work on solving such a problem and providing a complete solution for online landing location selection and matching with offline map.

1.4. Research Motivation

Based on the above literature study in the area of UAV surveillance mission planning, the following research gaps have been identified, which become the focus of the research effort presented in this thesis.

For the mission of continuous surveillance with fixed-wing UAVs, the problem of coverage path planning of UAV has been investigated. Some researchers planned an occlusion-aware path for a UAV with a sensor pointing downward to covering an urban target area. Others considered path planning of UAV with gimbaled camera for maximizing coverage in wiled area. However, no researches have combined both the two issues together. That is to say, there’s lack of research on the problem of coverage planning of fixed-wing UAV with gimbaled camera in an urban area with hazard existence.

For the perch-and-stare surveillance mission with rotary-wing UAVs, there are some researches on the problem of offline perching location selection in urban area. In the selection, some geometric criteria and mission criteria are considered. For the mission criteria, none of the researches considered the line of sight condition in the selection of the UAV perching location in urban environment.

Although there have been much reported work regarding online landing location selection for rotary-wing UAVs, there is a lack of a complete solution on the integration of landing location selection and registration with offline map.

1.5. Research Objectives and Scope

1.5.1. Research objectives

The following specific research objectives are proposed in this study:

(1) To develop a solution algorithm of optimal flying path planning for fixed-wing UAVs with an on-board Gimbaled camera to cover any given urban area (in the form of 3D map) with hazard existence.

(2) To develop a solution algorithm for selecting feasible perching locations on building roofs from an offline 3D map for a rotary-wing UAV in the perch-and-stare mission.

(3) To develop a solution algorithm for online landing location identification and feasibility checking for a rotary-wing UAV in the perch-and-stare mission. For landing location identification, registration between online collected point data with offline should be achieved.

1.5.2. Research scope

For fixed-wing UAV flying path planning, the following research scope is covered:

(1) In order to cover the whole urban environment, a set of vantage points are to be generated.
(2) At each vantage point, the optimal view direction is to be calculated.
(3) The sequence of the UAVs visiting the vantage points is to be determined.
(4) Smooth and continuous flying paths are to be fitted.

For rotary-wing UAV offline perching location selection, the following research scope is covered:

(1) A set of realistic perching location selection criteria are proposed.
(2) The algorithm of line of sight check is developed.

For rotary-wing UAV online landing location selection and registration, the following research scope is covered:

(1) The method of filtering online collected point cloud is developed.
(2) The method of extracting roof contour is proposed.
(3) The algorithm of registration between the online collected 3D point cloud with the offline map is developed.
(4) The criteria of online landing location evaluation is proposed and demonstrated.

1.6. Organization of the Thesis

In the remaining chapters of the thesis, each one covers a topic of the research objectives. Here’s a brief overview:

Chapter 2 is focused on the problem of fixed-wing UAV path planning. In the problem, a realistic urban environment with hazard existence is modelled, and the method of path planning for covering the whole area is developed.

Chapter 3 solves the problem of landing location selection for rotary-wing UAV within an offline map to keep a line of sight to a target.

Chapter 4 developed a complete solution of online landing location selection for rotary-wing UAV based on 3D point cloud and 2½D map.

Chapter 5 draws the conclusions based on the works discussed in the thesis and gives suggestions for possible future work.

CHAPTER 2

COVERAGE PATH PLANNING FOR CONTINUOUS CITY SURVEILLANCE WITH FIXED-WING UAVS IN HAZARDOUS ENVIRONMENT

In military missions, collecting enemy intelligence is an important but dangerous task. To reduce casualties, UAVs are often dispatched to conduct surveillance missions instead of scouts. In this kind of mission, the deployed UAVs usually fly over a target area and collect information with on-board sensing equipment, such as camera. The objective is to cover the whole target area and make sure that each point on the surface of the target area is captured. In this mission, factors to be considered include flying duration, the sensing ability of the on-board equipment, the distribution of enemy military facilities, etc. Such surveillance mission is illustrated in Figure 2.1.

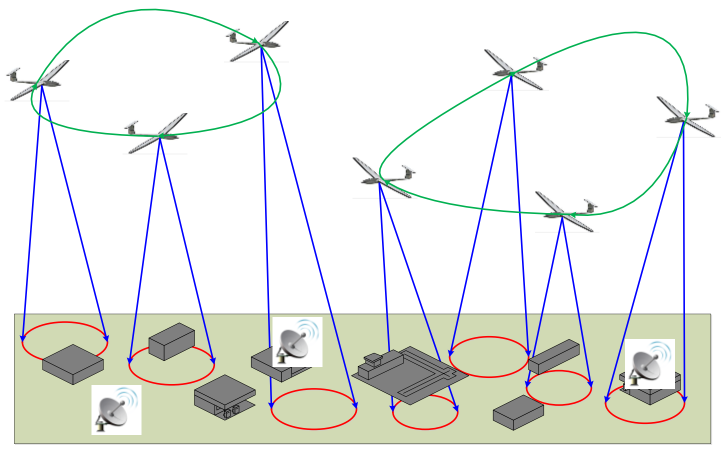


Figure 2.1 UAV surveillance mission

2.1. Background

Nowadays, UAVs are widely used for surveillance tasks. The purpose of surveillance could be tracking of ground targets (Chen and Dawson 2006, Kang and Hedrick 2009), information gathering and classification at certain locations (Quigley et al. 2005, Zhang et al. 2013), or to provide continuous surveillance coverage over a certain area. Coverage of an area with robots in general was first studied by (Zelinsky et al. 1993) over a grid in 2D space with a method based on the distance transform path planning. In (Savla et al. 2007), vehicles following Dubins path, a popular 2D path formation for UAVs, are guided to provide complete coverage of an area with path planning method derived from those intended for the disk covering problem. When UAVs are considered, a certain level of analogy with the 2D model can be built with a series of simplifications, e.g., the target area is flat, the camera faces downward and the UAV flies at a fixed altitude. Some works under these assumptions can be found in (Berger et al. 2009, Jones 2009).

In real-life surveillance tasks, especially those over urban environment, the line of sight between the sensor and a ground point of interest (POI) might be blocked by terrain or building. Because of this occlusion effect, more realistic camera models are provided in (Jakob et al. 2010, Kim and Crassidis 2010). In these works, the field of view (FOV) of a sensor on-board a UAV is modelled as a cone facing downward. Any point inside the cone and with a direct line of sight to the sensor is considered as visible. A more realistic model is to consider the pan-tilt capability of the sensor. In this way, the camera's axis can be optimized for better performance, e.g., maximum coverage.

2.2. Related Works

2.2.1. Coverage path planning for UAV

In (Sujit and Beard 2007), multiple UAVs are deployed in a task to explore an unknown region with obstacles. A dynamical map is created for planning paths in real time, considering the constraints of turning radius, sensor range, and communication range. Maza and Ollero (2007) investigated the problem of covering an area in searching for possible objects by a group of UAVs with Gimbaled camera. The searching areas are partitioned and the paths are predefined to follow a zigzag pattern. Özalp and Sahingoz (2013) planned optimal path for UAV within a simulated 3D environment with threatening radars.

2.2.2. Path smoothing

In the UAV path planning problem, a critical sub-problem is path representation, which can be categorized into two types. The first group is composite path by connecting simple geometric elements, such as straight lines and arcs, tangentially. A typical example is the use of Dubins paths, which are widely used for 2D problems (Nigam and Kroo 2008, Jakob et al. 2010). For this kind of path, the UAV's capability constraints can be incorporated in path planning easily. The major drawback is that the paths are only C^1continuous, which brings about difficulty for control execution. The other type of representation contains spline-based paths, which are inherently C^2 continuous (Nikolos et al. 2003), such as Bezier curve (Macharet et al. 2010), B-spline curve (Berglund et al. 2003), etc. In our problem, C^2 continuous path is also to be used, which is represented by B-spline curves.

2.3. Concepts Defining

2.3.1. The city model

The city environment is modelled in a simplified manner by including only terrain and buildings. The terrain is represented in the form of digital elevation map (DEM), and the buildings are described as simple geometry objectives, such as cuboid, cylinder, dome, and regular prism. The hazard points are either on top of the buildings or on the ground. The whole city surface is discretized into a set of sampled surface points (***sP***s). Figure 2.2 shows an example city model, where the red dots represent the hazard points, and blue points the surface points.

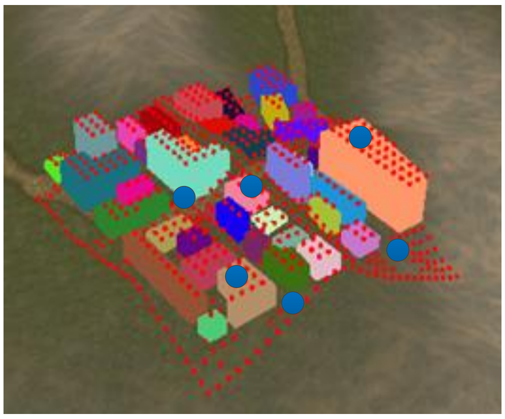


Figure 2.2 City model in coverage path planning

2.3.2. The camera model

The camera on-board a UAV takes pictures at a certain frequency with a certain angle of view (AOV) γ_v, which is the angular extent of a given scene that is imaged by the camera. The range of AOV is called the angle of coverage, as shown in Figure 2.3a. The UAV camera can be rotated to adjust its orientation. For a given surface point ***sP*** (x, y, z), a possible camera position ***cP*** that covers ***sP*** is on a hemi-sphere centred at ***sP***, determined by four parameters (*R, α, β, θ*), as shown in Figure 2.3b and 2.3c, respectively. The procedure for randomly generating a possible camera configuration is as follows:

(1) A hemisphere with radius R ($R_{\min} \leq R \leq R_{max}$) is generated, centred at ***sP***, where R_{max} is the maximum detection range of camera and R_{min} is the minimum safety distance to avoid collision to surface. Secondly, a point ***cP*** on the hemisphere is randomly generated with two angles, *elevation* angle α ($0 \leq \alpha \leq \alpha_{max}$), and *azimuth* angle β ($0 \leq \beta \leq 2\pi$), where α_{max} is the maximum elevation angle.

(2) Feasibility checking. If the line (***cP-sP***) does not interfere with buildings, ***cP*** is considered feasible.

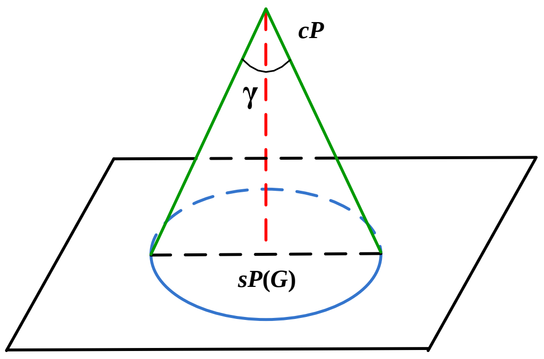
(3) Determination of the optimal view orientation. The optimal view orientation is defined as the one when the coverage area reaches the maximum. The angle θ between view direction (***cP-sP***) and the optimal view orientation (***cP-G***) can be calculated as:

$$\theta = \begin{cases} \alpha, & 0 \leq \alpha \leq \gamma_v / 2 \\ \gamma_v / 2, & \gamma_v / 2 \leq \alpha \leq \pi / 2 \end{cases} \tag{2-1}$$

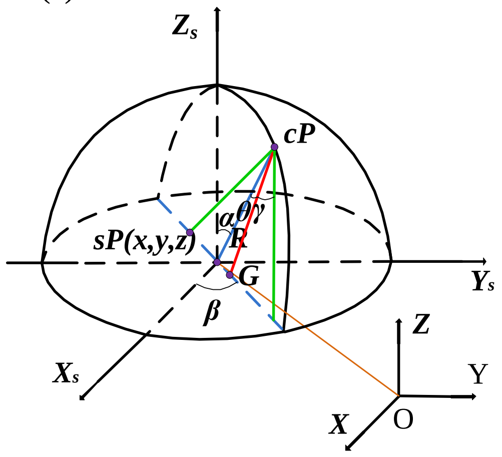
However, when θ=α, as shown in Figure 2.3d, ***sP*** is at the camera's AOV's boundary, which is vulnerable to camera distortion. To avoid this, the optimal value of $\theta_{optimal}$ is set as:

$$\theta_{optimal} = \begin{cases} \alpha, & 0 \leq \alpha \leq \gamma_v / 2 - \theta_0 \\ \gamma_v / 2 - \theta_0, & \gamma_v / 2 \leq \alpha \leq \pi / 2 - \theta_0 \end{cases} \tag{2-2}$$

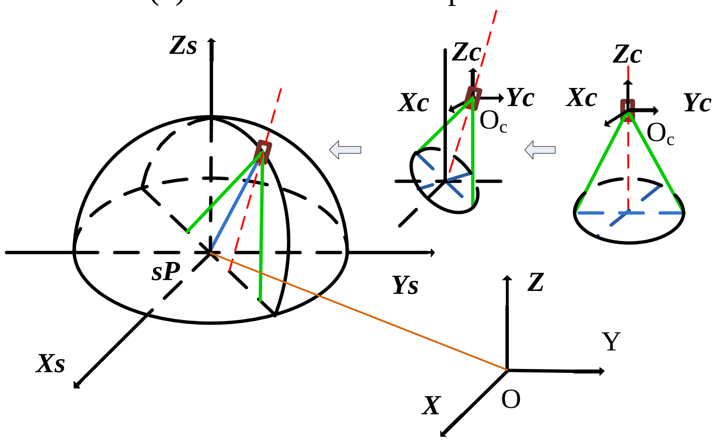
where, θ_0 is set between 1°~2° as a safety factor.



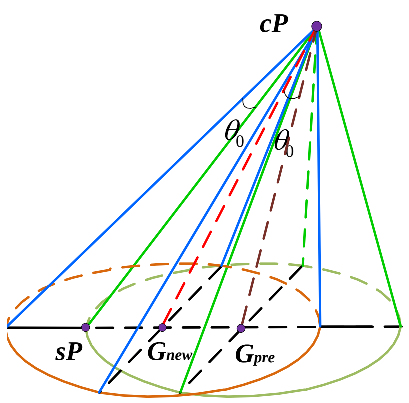
(a) AOV of camera downwards



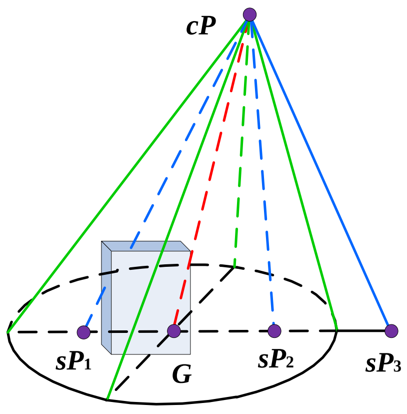
(b) Camera orientation parameters



(c) Camera orientation



(d) Safety factor



(e) Determine camera coverage

Figure 2.3 Gimbaled camera configuration

A camera configuration ***vP*** (x, y, z, α, β, θ) is considered feasible if the following conditions are satisfied:

(1) Line (***cP***-***sP***) is inside the AOV;

(2) Line (***cP***-***sP***) does not interfere with buildings or any other objects;

(3) The distance of (***cP***-***sP***) is within the range [R_{min}, R_{max}].

As shown in Figure 2.3e, ***sP***$_3$ is outside the AOV, ***sP***$_1$ is blocked by the building, only ***sP***$_2$ is feasible.

2.4. The Overall Approach

A feasible UAV flying path is one that the integration of the coverage points at each instance equals to the set of all the sampled ***sP***s. Since it is difficult to find the optimal path directly, the problem is decomposed into two sub-problems: waypoints determination and path planning. As the UAVs are tasked to cover all the ***sP***s, not only the coordinates of waypoints on the path are to be determined, the camera view direction at each waypoint also needs to be determined. A feasible set of waypoints is a set of camera configurations ***vP***s (called the *vantage* set) satisfying the following conditions:

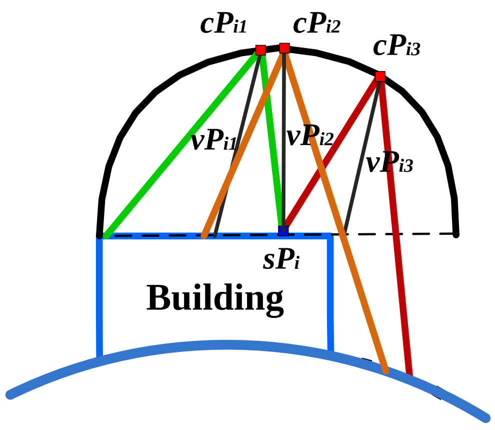
(1) The combination of the coverage at each ***vP*** equals to all the surface points ***sP***s;

(2) Each ***vP*** is unique;

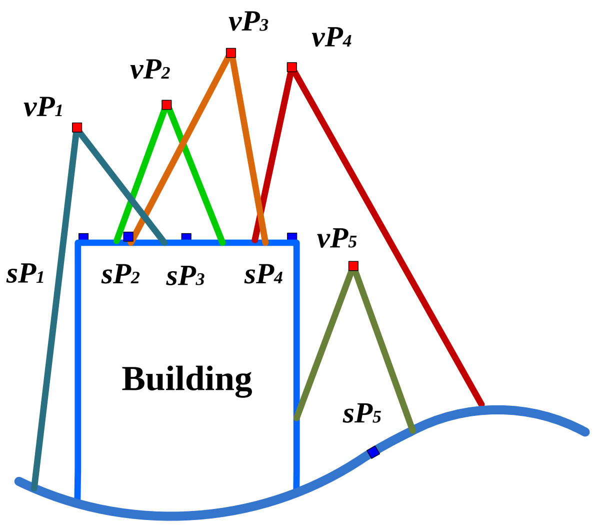
(3) The coverage set of any ***vP*** cannot be fully covered by any other camera configurations.

However, the problem of finding an optimal vantage set is still difficult, so it is decomposed further into two sub-steps:

(1) View direction generation for each surface points: For each ***sP***, the possible camera position (***cP***) covering ***sP*** is assumed on the hemisphere centred at ***sP*** with a radius *R* (*R* is within a feasible range but not fixed), e.g., ***cP***$_1$, ***cP***$_2$, and ***cP***$_3$ in Figure 2.4a. If there is no hazard point in the area, the direction that covers the largest number of surface points (including ***sP***) will be selected as the optimal view direction. If there are hazard points in the area, the ***cP*** that results in the minimum threat will be selected to form the optimal view direction. Therefore, for each ***sP****$_i$*, an optimal view direction (***vD****$_i$*) is identified. The collection of ***vD***s for all the sampled surface points form a ***vD*** set.



(a) Possible camera positions



(b) Coverage selection

Figure 2.4 Possible camera positions and coverage

(2) Generation of vantage set: In this step, an optimal set of camera positions, called the *vantage* set, i.e., ***vP***s, are to be identified based on the given ***vD*** set. As shown in Figure 2.4b, since not all the ***vP***s are needed to cover all the ***sP***s, an optimal subset is to be determined. The basic requirements for the vantage set are: (i) each ***sP*** is covered, (ii) the visiting sequence of ***vP***s is fixed, and (iii) the UAV flying capability constraints are satisfied. As for the optimization objective, there are two scenarios:

   a) If there are no hazard points in the area, the objective is to minimize the total estimated path length;

   b) If there are hazard points in the area, the objective is to minimize the accumulated threat.

Based on the analysis above, the overall approach is summarized as follows:

(1) Given a set of discrete surface points ***sP***s, for each ***sP****ᵢ*, the optimal camera view direction ($\boldsymbol{vD_i}$) to cover it is identified. The collection of ***vD***s for all the sampled surface points form a ***vD*** set.

(2) Given a ***vD*** set, an optimal *vantage* set ***vP***s is to be selected for the ***vD*** set, and the visiting sequence to the vantage points are to be determined. In addition, the 3D flying path is fitted to the sequenced vantage points.

2.5. Determination of the View Direction

As the Gimbaled camera can be rotated to adjust its orientation, there are countless camera configurations covering a given ***sP*** (within a certain distance range). The objective is to find the optimal camera configuration for each ***sP***.

2.5.1. Fitness of vantage set

The criteria to evaluate any camera configuration consist of two aspects:

(1) The total number of surface point covered;

(2) The distance to each hazard point.

If there is no hazard point in the city, the fitness function is simply total number of surface point covered. Otherwise, the fitness function is the sum of distance to each hazard point. Therefore, the fitness function is given as:

$$F = \begin{cases} N, & \textit{no hazard} \\ \sum_i^M d_i, & \textit{hazard exist} \end{cases} \tag{2-3}$$

where N is the total number of surface point covered, M is the total number of hazard points, and d_i is the distance to i-th hazard point.

2.5.2. Method to generate the optimal view direction

For each ***sP***, a corresponding ***cP*** is to be determined. As there are three parameters (R, α, β) to be calculated, the solution space is of high complexity. A simplified method is adopted here to randomly generate a radius, and then the two angles are selected with the best fitness. As the parameters to be optimized are angles of the view direction, it is quite convenient to model the angles as particles of particle swarm optimization (PSO), and the update rule of the particle can be applied elegantly, which makes PSO to be a quite suitable algorithm for this problem. Here, the two angles are selected using a parallel asynchronous PSO method.

A certain number of camera angle configurations are initialized as particles, and their positions are updated to find the global best particle with the maximum fitness. A particular view direction (α, β) is represented as a particle position $\boldsymbol{x_i}$ ($i = 1, 2, \ldots, N$), where N represents the number of particles. The change of view direction is represented as a particle velocity $\boldsymbol{v_i}$. The fitness of a particle is calculated using Equation (2-3). The pseudo code for this method is as follows:

```
Initialize $N$ particles with random positions and velocities, initialize the
previous best position (pbest) of each particle and the global best position
(gbest)
Do
  For each particle
      Calculate particle fitness with Equation (2-3)
      If the current particle fitness is better than the particle's previous best
      fitness, set the current position as the pbest
      Choose the particle with the best fitness value of all the particles as
      the gbest
      Update particle position and velocity with Equation (2-4)
  End For
  If the fitness function changes very little in 10 iterations or reach the
maximum iteration
End Do
Output the global best position as the solution
```

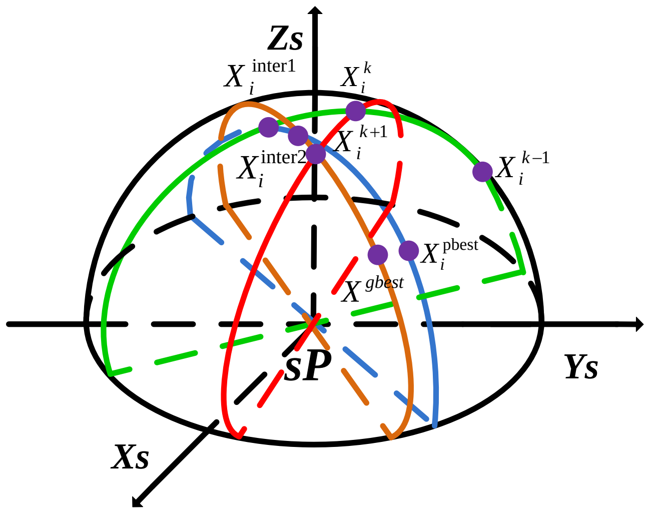
The randomly initialized particles are updated at each iteration k. Denote the position of the i-th particle at iteration k as $\boldsymbol{x_i^k}$, the previous best position of a particle as $\boldsymbol{x_i^{pbest}}$, and the global best position of all the particles as $\boldsymbol{x^{gbest}}$. $\boldsymbol{x_i^k}$ updates its position according to the fitness of $\boldsymbol{x_i^{pbest}}$ and $\boldsymbol{x^{gbest}}$

through moving towards $\boldsymbol{x}_i^{\boldsymbol{pbest}}$ and $\boldsymbol{x}^{\boldsymbol{gbest}}$. As the particles are on a sphere, the shortest path of the particle is to move along the great circles of the sphere. As shown in Figure 2.5a, $\boldsymbol{x}_i^{\boldsymbol{k}}$ moves along the great circle (green circle) determined by $\boldsymbol{x}_i^{\boldsymbol{k-1}}$, $\boldsymbol{x}_i^{\boldsymbol{k}}$ and the centre $\boldsymbol{sP}$. After it reaches an intermediate position $\boldsymbol{x}_i^{\boldsymbol{inter1}}$, it moves towards $\boldsymbol{x}_i^{\boldsymbol{pbest}}$ along the great circle (blue circle) determined by $\boldsymbol{x}_i^{\boldsymbol{k}}$, $\boldsymbol{x}_i^{\boldsymbol{pbest}}$ and the centre $\boldsymbol{sP}$. After it reaches another intermediate position $\boldsymbol{x}_i^{\boldsymbol{inter2}}$, it moves towards $\boldsymbol{x}^{\boldsymbol{gbest}}$ to $\boldsymbol{x}_i^{\boldsymbol{k+1}}$ along the great circle (the orange circle) determined by $\boldsymbol{x}_i^{\boldsymbol{inter2}}$, $\boldsymbol{x}^{\boldsymbol{gbest}}$ and the centre $\boldsymbol{sP}$. The final result is that $\boldsymbol{x}_i^{\boldsymbol{k}}$ moves to a new position $\boldsymbol{x}_i^{\boldsymbol{k+1}}$ along the great circle (the red circle) determined by $\boldsymbol{x}_i^{\boldsymbol{k}}$, $\boldsymbol{x}_i^{\boldsymbol{k+1}}$ and the centre $\boldsymbol{sP}$.

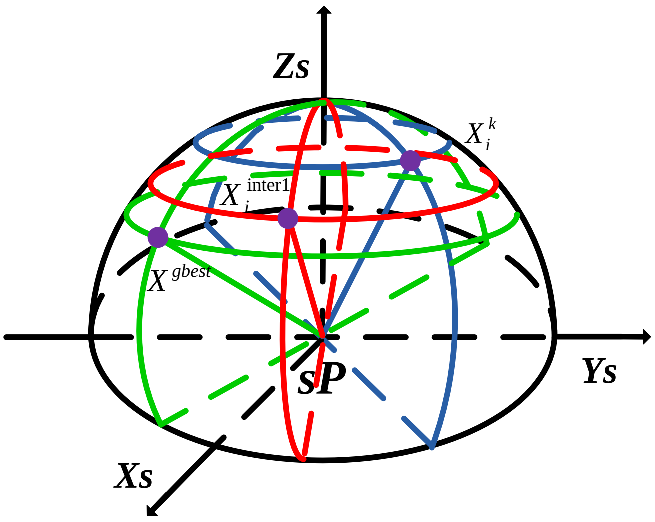
As the calculation is quite complex, a simplified update rule is used, that is, the two parameters (*α, β*) of the particle are updated separately. Particle $\boldsymbol{x}_i^{\boldsymbol{k}}$ updates its position according to $\boldsymbol{x}_i^{\boldsymbol{k-1}}$, $\boldsymbol{x}_i^{\boldsymbol{pbest}}$, and $\boldsymbol{x}^{\boldsymbol{gbest}}$. Figure 2.5b shows how to update $\boldsymbol{x}_i^{\boldsymbol{k}}$ according to $\boldsymbol{x}_i^{\boldsymbol{pbest}}$. Denote $\boldsymbol{x}_i^{\boldsymbol{k}} = (\alpha_k, \beta_k)$, $\boldsymbol{x}_i^{\boldsymbol{pbest}} = (\alpha_{pb}, \beta_{pb})$, and $\boldsymbol{x}_i^{\boldsymbol{inter1}} = (\alpha_1, \beta_1)$, the particle updates α_k to approach α_{gb}, and updates β_k to approach β_{gb}. As a result, $\alpha_1 = r_1 * (\alpha_{pb} - \alpha_k)$, $\beta_1 = r_2 * (\beta_{pb} - \beta_k)$. The update according to $\boldsymbol{x}_i^{\boldsymbol{k-1}}$**,** and $\boldsymbol{x}^{\boldsymbol{gbest}}$ are in a similar way. So, the update rule is:

$$\begin{gathered} \boldsymbol{v}_i = w \times \left(\boldsymbol{x}_i^k - \boldsymbol{x}_i^{k-1}\right) + c_1 \times R_1 \times \left(\boldsymbol{x}_i^{pbest} - \boldsymbol{x}_i\right) + c_2 \times R_2 * \left(\boldsymbol{x}^{gbest} - \boldsymbol{x}_i\right) \\ \boldsymbol{x}_i = \boldsymbol{x}_i + \boldsymbol{v}_i \end{gathered} \tag{2-4}$$

where *R₁* and *R₂* are row vectors, the element of which are random numbers in the range of [0, 1], w, c_1 and c_2 are weights to determine the contribution of $\boldsymbol{x}_i^{\boldsymbol{k-1}}$, $\boldsymbol{x}_i^{\boldsymbol{pbest}}$ and $\boldsymbol{x}^{\boldsymbol{gbest}}$.



(a) Update rule



(b) Simplified update rule

Figure 2.5 PSO update of a particle

2.5.3. Numerical tests

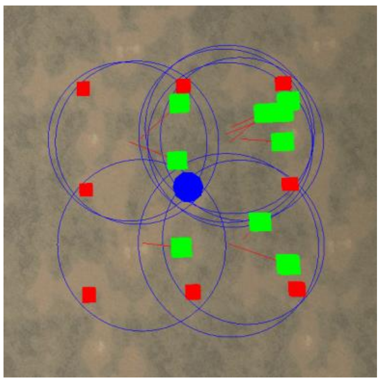
A simple numerical experiment has been carried out to test the method of view direction determination. The size of the city to be reconnoitred is 4m×4m, and surface points are sampled with an interval of 2m. In the scenario with hazard, there is one hazard point located at position with coordinate of (1, 1). Table

2.1 shows the parameters used in the numerical experiment. In this experiment, 50 view directions (α, β) are initialized as particles randomly.

Table 2.1 Parameters in view direction determination

Camera			
γ_v	$R_{\min}$	$R_{\max}$	α_{max}
45°	10m	300m	50°
PSO for vantage point initialization			
Number of particles	Max iteration	θ_0	
50	50	3°	

The results of view direction determination are shown in Figure 2.6. In the scenario without hazard, it is obvious that each camera selects the direction where they can cover the most surface points. As shown in Figure 2.6a, each camera cover 4 surface points, which can also be verified by mathematic calculation analytically. However, in the scenario with hazard, each camera selects the direction where they can avoid the hazard most. As shown in Figure 2.6b, the cameras are located at the farthest position to avoid hazard under the condition that they cover their own surface point respectively.


(a) View directions without hazard

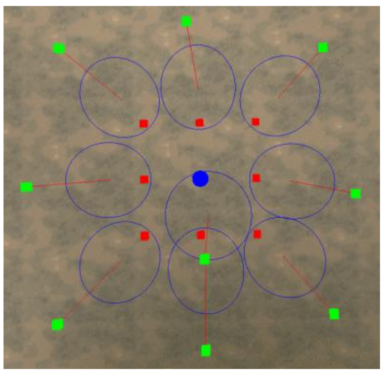

(b) View directions with hazard

Figure 2.6 Results of view direction determination

2.5.4. Comparison of the PSO and the brute-force method

A numerical experiment has been carried out to compare the PSO method with the brute-force search method (BFS), a general method to solve problems systematically enumerating all possible candidates for the solution. The size of the city to be reconnoitred is 20m×20m, and surface points are sampled with an interval of 0.67m. The city is shown in Figure 2.7, with four buildings (colour blocks) and two possible hazard points (red square). Table 2.2 shows the parameters used in the numerical experiment. Here, the BFS is to sample some positions in the solution space uniformly, and calculate the fitness for each position to find the best solution. As the position on the hemi-sphere is determined by two parameters, α ($0< \alpha < \alpha_{max}$) and β ($0 < \beta < 2\pi$), the position sampling is carried out as follows:

(1) Generate n longitude circles (see Figure 2.8, the colour circles) each 1 radian on the hemi-sphere;

(2) Generate n positions (see Figure 2.8, the pink points) each 1 radian on the longitude circles;

The total sample position number is $N = \sum_{i=0}^{i\leq\alpha_{max}} n \times \cos(i/n)$. In the experiment, n=16 and n=32 are chosen to do sampling, as a result, the total sample number are 1182 and 4687, respectively.

Table 2.2 Parameters in comparison tests between PSO and BFS

Camera			
γ_v	$R_{\min}$	$R_{\max}$	α_{max}
45°	10m	300m	50°
PSO method			
Number of particles		Max iteration	
50		50	

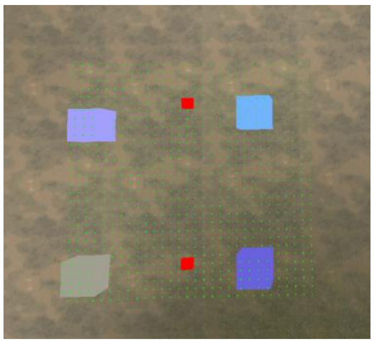


Figure 2.7 City environment for comparison test of PSO and BFS

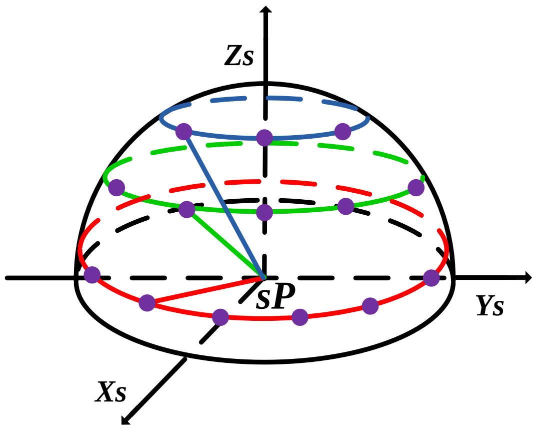


Figure 2.8 Position sampling of BFS

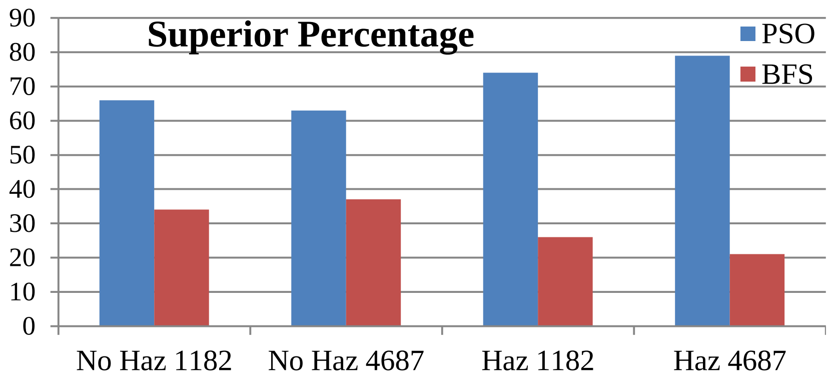


Figure 2.9 Comparison of priority between PSO and BFS

Table 2.3 Comparison of computation time between PSO and BFS

	No hazard	With hazard
PSO	16.7 s	1.6 s
Brute-force search with 1182 sample	768.5 s	21.8 s
Brute-force search with 4687 sample	19598 s	491.8 s

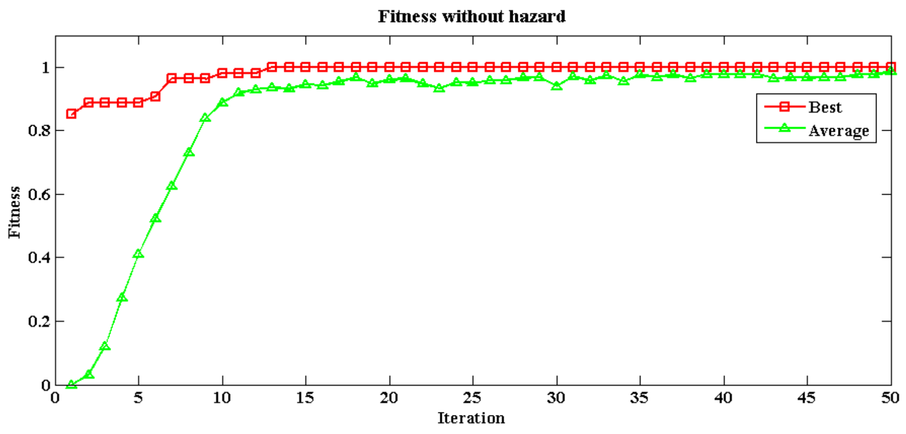
Table 2.4 Comparison of average fitness between PSO and BFS

	No hazard	With hazard
PSO	15.81	76.09
Brute-force search with 1182 sample	15.64	62.31
Brute-force search with 4687 sample	15.65	62.76

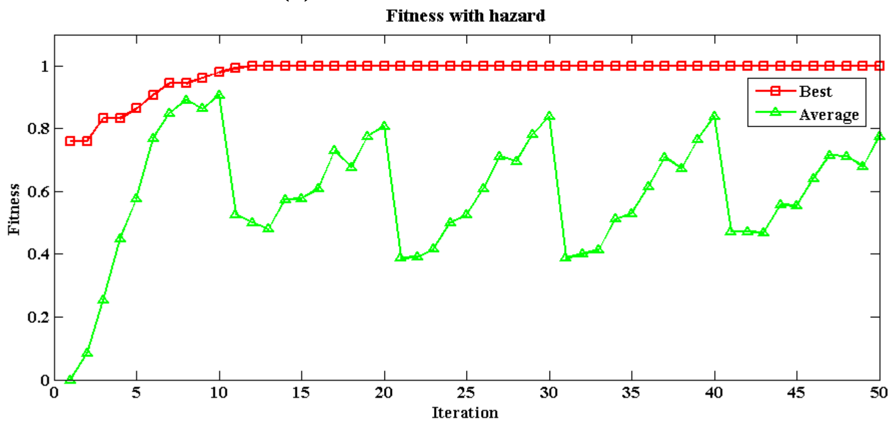
The computation time of the two methods are shown in Table 2.3. And the compare of solution quality is shown in Table 2.4. The average of the fitness of all the surface points is listed for different methods in the scenario with/without hazard. Figure 2.9 shows the superior percentage of the two methods, comparing the computation result of PSO v.s. BFS in the scenario without and with hazard. The PSO method has a higher superior percentage than BFS method. In addition, Table 2.4 shows that the PSO method has higher computation efficiency than the BFS method, especially in the scenario without hazard. As can be seen from Table 2.3, the computation time of the BFS method grows quadratically as the increase of surface points, which is tens, even hundreds times more than that of the PSO method.

2.5.5. Convergence performance test of the PSO method

Furthermore, an experiment was carried out to test the convergence performance of the PSO method. Here, the city and parameters are the same as that in section 2.5.4 (see Figure 2.7). Figure 2.10 shows the fitness record of PSO method with the growth of computation iteration (the fitness has been normalized to the range [0, 1]). In both the scenario without hazard (see Figure 2.10a) and with hazard (see Figure 2.10b), the particle fitness and global fitness grows with the iteration. The global fitness converges eventually, and the particle fitness approaches global fitness gradually.



(a) Fitness without hazard



(b) Fitness with hazard

Figure 2.10 Record of particle fitness and global fitness for PSO

2.6. Determination of Vantage Set and Visiting Sequence

Given a covering ***vD*** set with n ***vD***s, the objective is to find an optimal set of vantage points (***vP***s, called vantage set) that covers all the ***sP***s with the best fitness. In this process, the visiting sequence of the ***vP***s is also determined. For each ***vP***, the camera angle (α, β, θ) is already determined. However, the distance between ***vP*** and ***sP*** is yet to be fixed. As shown in Figure 2.11, the covering ***vP*** set consists of [$\boldsymbol{vP}_1$, $\boldsymbol{vP}_2$, …, $\boldsymbol{vP}_i$, …, $\boldsymbol{vP}_n$], where, the distances between $\boldsymbol{vP}_i$ and $\boldsymbol{sP}_i$ (i = 1, 2, …, n) can be adjusted along the view directions. This problem consists of two parts, vantage set generation and visiting sequence determination. The problem of visiting sequence determination can be converted to a TSP (for single UAV surveillance) or MTSP (for multiple UAV surveillance) problem.

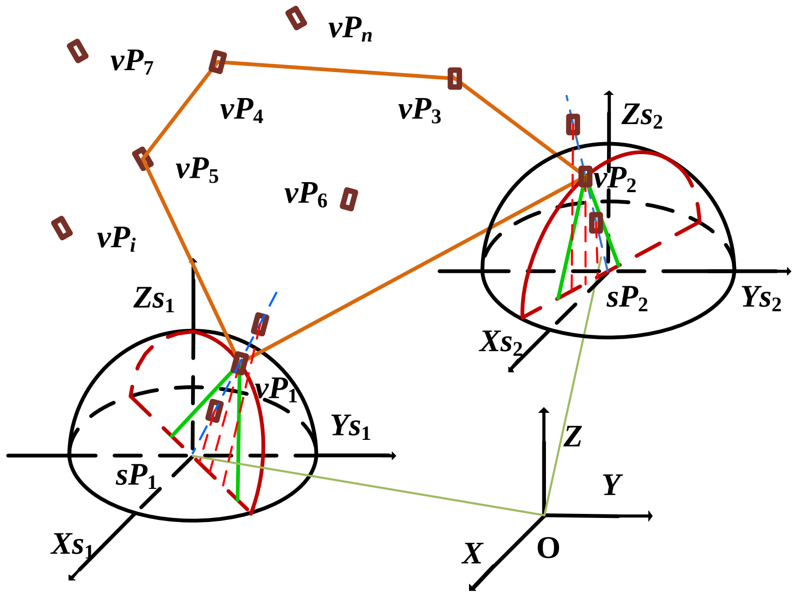


Figure 2.11 Determination of vantage set and visiting sequence

2.6.1. Fitness function for UAV path

(1) Fitness function without the existence of hazard

If there is no hazard point, the criteria to evaluate a solution consist of three factors:

a) The estimated total path length l of the vantage set;
b) The turning radius R_t of the UAV should be no smaller than minimum turning radius $R_{t,min}$;
c) The climbing/diving angle $\theta_{C,D}$ should be no larger than maximal climbing/diving angle $\theta_{C,D,max}$.

Therefore, given a vantage set with n cameras [$\boldsymbol{vP}_1$, $\boldsymbol{vP}_2$, …, $\boldsymbol{vP}_i$, …, $\boldsymbol{vP}_n$], there are n edges to form a feasible path. The fitness function is given as:

$$\text{minimize:} \quad \sum_{i=1}^{n} l_{i,i+1}$$

$$\text{subject to:} \quad \begin{cases} \theta_{i,i+1} \le \theta_{C,D,max} \\ R_t \ge R_{t,min} \end{cases}, \qquad i = 1, 2 \dots n \tag{2-5}$$

where $l_{i,i+1}$ is the edge length between the camera position $\boldsymbol{vP}_i$ and $\boldsymbol{vP}_{i+1}$, $\theta_{i,i+1}$ is the climbing/diving angle of edge $l_{i,i+1}$.

(2) Fitness function with the existence of hazard

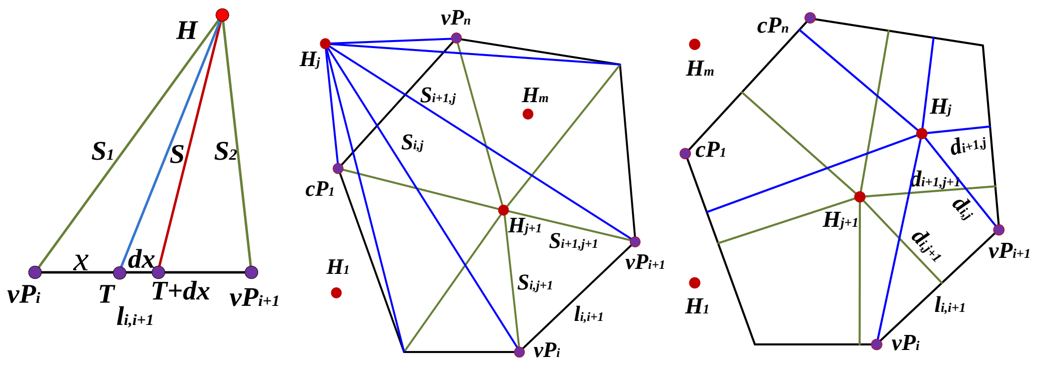
If there are hazard points, the criteria to evaluate a solution consist of three factors:

a) The sum of multiplication of exposure time and distance to each hazard point;
b) The turning radius R_t of the UAV should be no smaller than minimum turning radius $R_{t,min}$;
c) The climbing/diving angle $\theta_{C,D}$ should be no larger than minimum climbing/diving angle $\theta_{C,D,max}$.

Therefore, given a camera configuration set with n camera positions [$\boldsymbol{vP}_1$, $\boldsymbol{vP}_2$, …, $\boldsymbol{vP}_i$, …, $\boldsymbol{vP}_n$], there are n edges to form a feasible path. There are two criteria to evaluate the fitness of a path, the accumulated hazard on each edge or the minimal distance to each edge. As shown in Figure 2.12a and Figure 2.12b, $\boldsymbol{vP}_i$ and $\boldsymbol{vP}_{i+1}$ are two adjacent camera locations of a UAV, $\boldsymbol{H}$ is a hazard point, $\boldsymbol{T}$ is a waypoint on the line $\boldsymbol{vP}_i$ - $\boldsymbol{vP}_{i+1}$, during the time of UAV traveling from $\boldsymbol{vP}_i$ to $\boldsymbol{vP}_{i+1}$, the accumulated hazard is an integration of hazard function $h(x) = k \big/ \left(S(x)\right)^4$ on the line $\boldsymbol{vP}_i$ - $\boldsymbol{vP}_{i+1}$. $S(x)$ is the function of x with parameters (S_1, S_2, $l_{i,i+1}$). Figure 2.12c shows the fitness of minimal distance to each edge, where $d_{i,j}$ is the distance of hazard point $\boldsymbol{H_j}$ to edge $l_{i,i+1}$. The fitness function is as follows:

$$\begin{array}{lll}
\text{minimize:} & \sum_{j=1}^{M}\sum_{i=1}^{n}\int_{0}^{l_{i,i+1}}\frac{k}{\left(S(x)\right)^{4}}dx & \\
\text{or minimize:} & \max_{i,j}\frac{1}{d_{i,j}}, \quad i=1,2\ldots n, \quad j=1,2\ldots n & \text{(2-6)} \\
\text{subject to:} & \begin{cases}\theta_{i,i+1}\le\theta_{C,D,max} \\ R_t\ge R_{t,min}\end{cases}, \quad i=1,2\ldots n &
\end{array}$$

where $l_{i,i+1}$ is the edge length between cameras $\boldsymbol{vP_i}$ and $\boldsymbol{vP_{i+1}}$, θ_i is the climbing/diving angle of edge $l_{i,i+1}$.



(a) Accumulated hazard on an edge (b) Accumulated hazard on a path (c) Minimum distance on a path

Figure 2.12 Fitness for path of UAV in scenarios with hazard

2.6.2. Method to generate vantage set

(1) GA based Vantage Point Initialization

A genetic algorithm (GA) base method is used to generate the vantage set. A feasible solution is a group of camera positions covering all ***sP***s, represented as a chromosome consisting of N genes, and each gene represents a camera position covering some ***sP***s. Suppose there are N ***sP***s, a camera position is initialized right above each ***sP*** (x_i, y_i, z_i) with a random distance between $R_{\min}$ and $R_{\max}$. In the proposed method, a camera position is initialized as a point along the line between ***sP*** and its candidate camera position, with the determined (α, β), as shown in Figure 2.13.

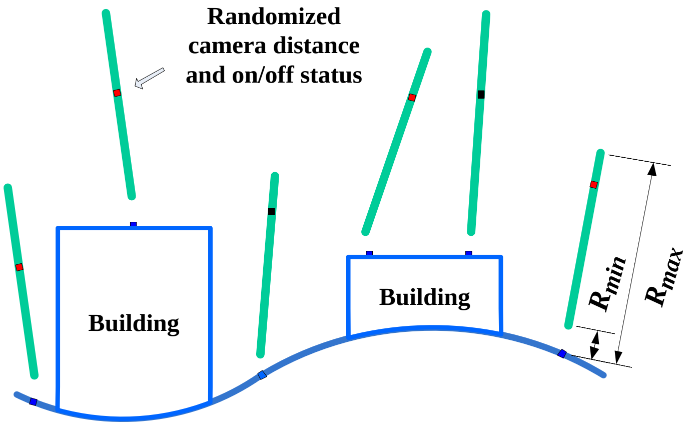
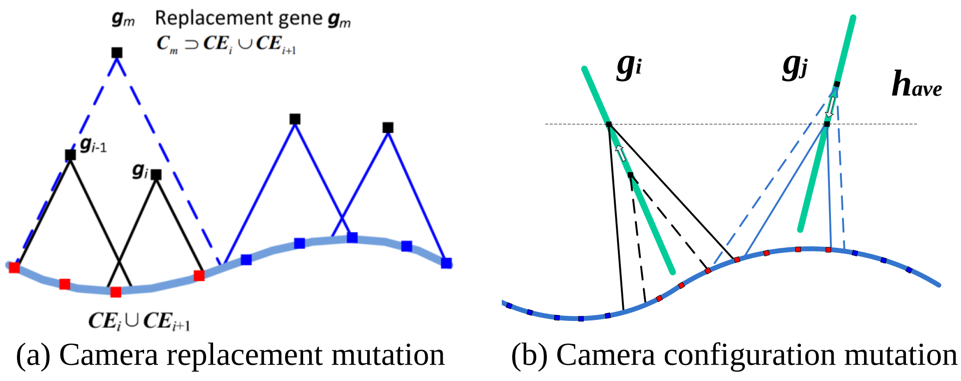


Figure 2.13 Vantage point initialization

(2) Vantage Set Selection

The initial population is randomly split into two groups and conducted reproduction, including cross-over and mutation. The cross-over operator works as follows:

```
Do
    Select randomly one of the parents based on probability proportional to their fitness values;
    Identify the camera position of the selected parent that covers the most uncovered surface points (of the child solution) and place into the child solution;
    If each ground point with the child solution is covered;
End Do
```



(a) Camera replacement mutation (b) Camera configuration mutation

Figure 2.14 GA mutation operator

The mutation operator consists of camera replacement mutation and camera configuration mutation, as shown in Figure 2.14. Camera replacement mutation is to replace two cameras with one, which can cover the union of the effective coverage of the two previous cameras. Camera altitude mutation is to adjust the distance of camera position to reduce the height deviation as long as the effective coverage is not affected. The effective coverage is defined as the ***sP***s covered only by one camera.

2.6.3. Method to determine path sequence

Here, a branch and bound algorithm is utilized to obtain the path sequence of a vantage set. Given a camera configuration set with n cameras [$\boldsymbol{vP}_1$, $\boldsymbol{vP}_2$, …, $\boldsymbol{vP}_i$, …, $\boldsymbol{vP}_n$], there are $n\times(n-1)$ possible path segments. Given a vantage set, a table of edge cost for possible path segments is generated, as shown in

Table 2.5.

Table 2.5 Edge cost for possible path segments

	$\boldsymbol{vP}_1$	$\boldsymbol{vP}_2$	…	$\boldsymbol{vP}_i$	…	$\boldsymbol{vP}_n$
$\boldsymbol{vP}_1$	--	$l_{1,2}$	…	$l_{1,i}$	…	$l_{1,n}$
$\boldsymbol{vP}_2$	$l_{2,1}$	--	…	$l_{2,i}$	…	$l_{2,n}$
…	…	…	--	…	…	…
$\boldsymbol{vP}_i$	$l_{i,1}$	$l_{i,2}$	…	--	…	$l_{i,n}$
…	…	…	…	…	--	…
$\boldsymbol{vP}_n$	$l_{n,1}$	$l_{n,2}$	…	$l_{n,i}$	…	--

Considering the constraints of minimum climbing/diving angle $\theta_{C,D,max}$, the edges with larger $\theta_{C,D}$ will be set with a positive infinite edge cost. After checking for all the edges, the table is updated. In addition, considering the constraints of minimum turning radius $R_{t,min}$, the adjacent edges with smaller turning radius R cannot be selected in the same path sequence. For three adjacent points ($\boldsymbol{vP}_{i-1}$, $\boldsymbol{vP}_i$, $\boldsymbol{vP}_{i+1}$), the turning radius constraint is checked as follows (see Figure 2.15).

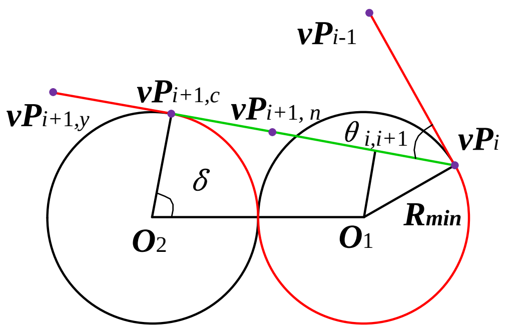
(1) Project the three points onto the horizontal plane, and connect $\boldsymbol{vP}_{i-1}$ - $\boldsymbol{vP}_i$, $\boldsymbol{vP}_i$ - $\boldsymbol{vP}_{i+1}$.

(2) Draw circle $\boldsymbol{O}_1$ with radius $R_{t,min}$, tangential with line $\boldsymbol{vP}_{i-1}$ - $\boldsymbol{vP}_i$ at point $\boldsymbol{vP}_i$.

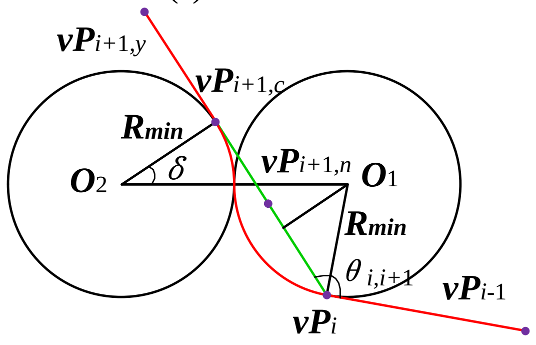
(3) Draw circle $\boldsymbol{O}_2$ with radius $R_{t,min}$, tangential with line $\boldsymbol{vP}_{i-1}$ - $\boldsymbol{vP}_i$ and circle $\boldsymbol{O}_1$, denote the tangent point with line $\boldsymbol{vP}_i$ - $\boldsymbol{vP}_{i+1}$ is $\boldsymbol{vP}_{i+1,c}$, the critical point to determine whether satisfy the turning radius constraints.

(4) If $\boldsymbol{vP}_{i+1,c}$ is located between $\boldsymbol{vP}_i$ and $\boldsymbol{vP}_{i+1}$, turning radius constraint is satisfied.

Denote the length of line segment $\boldsymbol{vP}_i$ - $\boldsymbol{vP}_{i+1}$ as $l_{i,i+1}$, and the angle between $\boldsymbol{vP}_{i-1}$ - $\boldsymbol{vP}_i$ and $\boldsymbol{vP}_i$ - $\boldsymbol{vP}_{i+1}$ is $\theta_{i,i+1}$, the condition of satisfying the turning radius constraint is $l_{i,i+1} \geq R_{t,min} \times (2\times\sin\delta+\sin\theta_{i,i+1})$, where $\delta = \arccos(1-\cos\theta_{i,i+1})$. Another table is generated to save the turning angle check result of all possible path segments. The check result is used during the process of next edge selection. Table 2.6 shows an example of the turning angle check result for the edges containing the node $\boldsymbol{vP}_1$.



(a) Scenario a



(b) Scenario b

Figure 2.15 The check of turning radius constraint

Table 2.6 Example for result of turning angle check

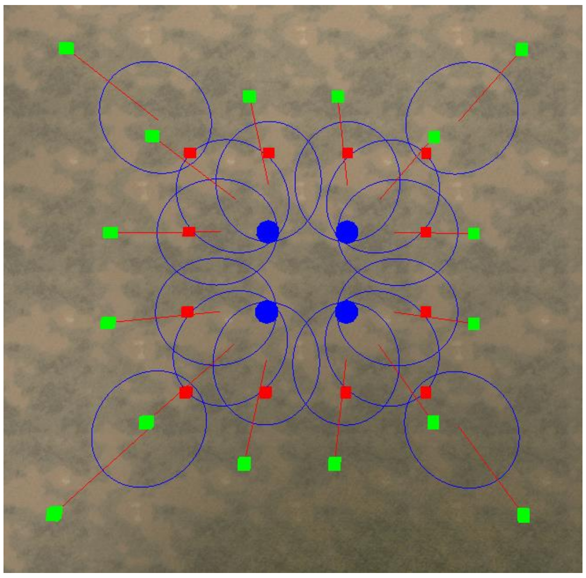
	$l_{1,2}$	$l_{1,3}$	…	$l_{1,i}$	…	$l_{1,n}$
$l_{2,1}$	--	Yes	…	Yes	…	No
$l_{3,1}$	Yes	--	…	Yes	…	No
…	…	…	--	…	…	…
$l_{i,1}$	Yes	Yes	…	--	…	Yes
…	…	…	…	…	--	…
$l_{n,1}$	No	No	…	Yes	…	--

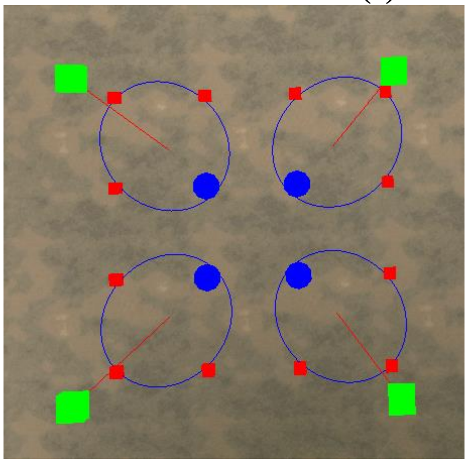
2.6.4. Numerical tests

A numerical experiment has been carried out to test the method of vantage set generation. The size of the city to be reconnoitered is 4m×4m, and surface points are sampled each 2m. Table 2.7 shows the parameters used in the numerical experiment. In the scenario with the existence of hazard, there is one hazard point located at position of (1, 1). Here, the fitness function of minimal distance is used. As shown in Figure 2.16a, the directions of cameras are given. There are four cameras located on the boundary of the terrain, each covering two surface points outside. Another four cameras cover four surface points respectively. The last camera covers two surface points, including the center point.

Table 2.7 Parameters in determination of vantage set and visiting sequence

Camera			
γ_v	R_{min}	R_{max}	α_{max}
45°	10m	300m	50°
UAV flying capability			
Max climbing/diving angle		Min turning angle	
45°		10m	
GA for Vantage Set Generation			
Random mutation probability		No. of max offspring	
10%		600	


(a) Camera directions


(b) Camera position (no hazard)

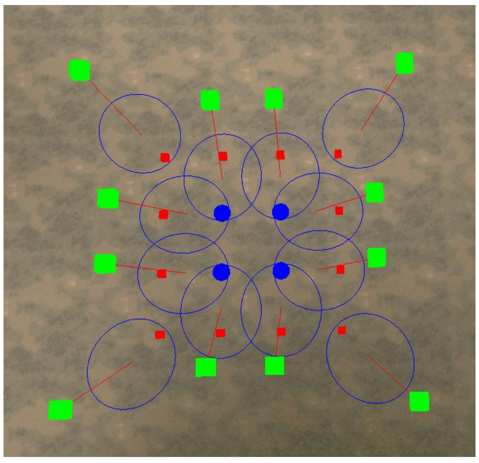

(c) Camera position (with hazard)

Figure 2.16 Numerical result of vantage set generation

The results of vantage set generation without the existence of hazard and with hazard are shown in Figure 2.16b and Figure 2.16c, respectively. As can be seen from Figure 2.16b, in the scenario without hazard, the vantage set with the least path length is selected, that is the four cameras located inside. However, in the scenario with hazard, the other five cameras are selected, for they are located further to the hazard point, resulting a less accumulated hazard.

2.7. Comprehensive Simulation Tests

A numerical experiment has been carried out to test the whole method for coverage path planning of UAVs. The size of the city to be reconnoitred is 20m×20m, and surface points are sampled each 0.67m. Table 2.8 shows the parameters used in the numerical tests. The number of vantage points and total path length are affected by the available number of UAVs and the condition whether there is hazard or not. Here, an experiment is carried out, and the city environment of the experiment is shown in Figure 2.17, with four buildings (colour blocks) and two possible hazard point (red square). The surface points are represented by green points. In the scenario with hazard, the fitness function of minimal distance is used. The results are shown in Table 2.9 and Figure 2.18. As shown in Figure 2.18, all the surface points are covered by vantage points. The surface points covered by vantage points are shown as green dots. The average convergence time of PSO algorithm is 764s. The numerical results show that when there’s hazard in the city, the UAVs will tend to fly far from the city to reduce danger.

Table 2.8 Parameters in comprehensive tests for coverage path planning

Camera			
γ_v	R_{min}	R_{max}	α_{max}
45°	10m	300m	50°
PSO for vantage point initialization			
Number of particles	Max iteration	θ_0	
50	50	3°	
UAV flying capability			
Max climbing/diving angle	Min turning angle		
30°	45°		
GA for vantage set generation			
Random mutation probability	No. of max offsprings		
10%	600		

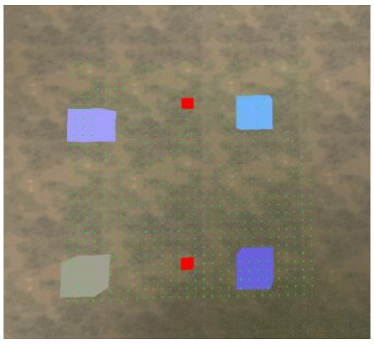
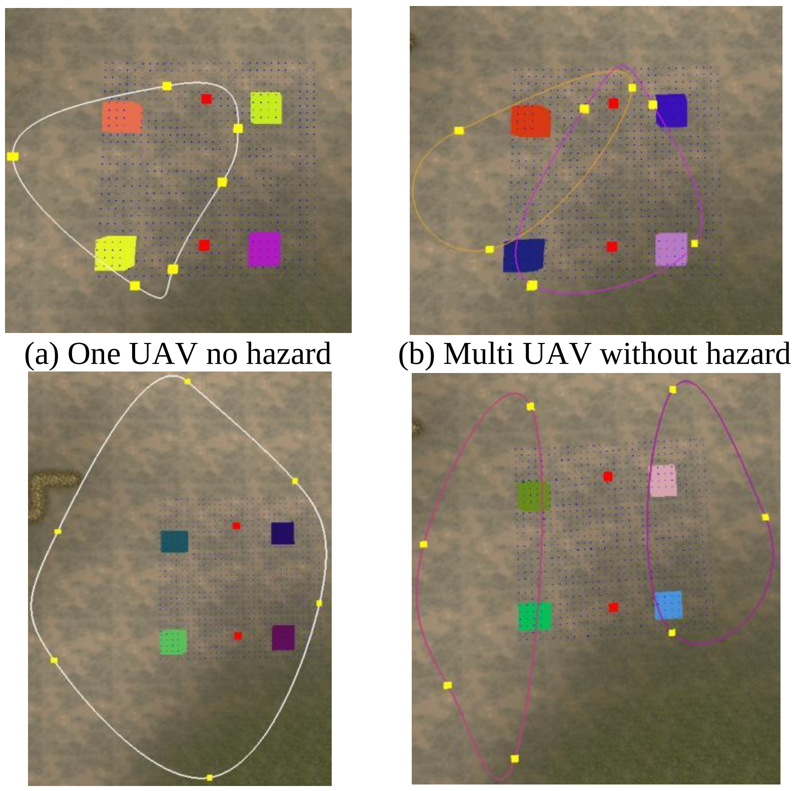


Figure 2.17 City environment of the experiment for coverage path planning

Table 2.9 Numerical experiment results for a simple city

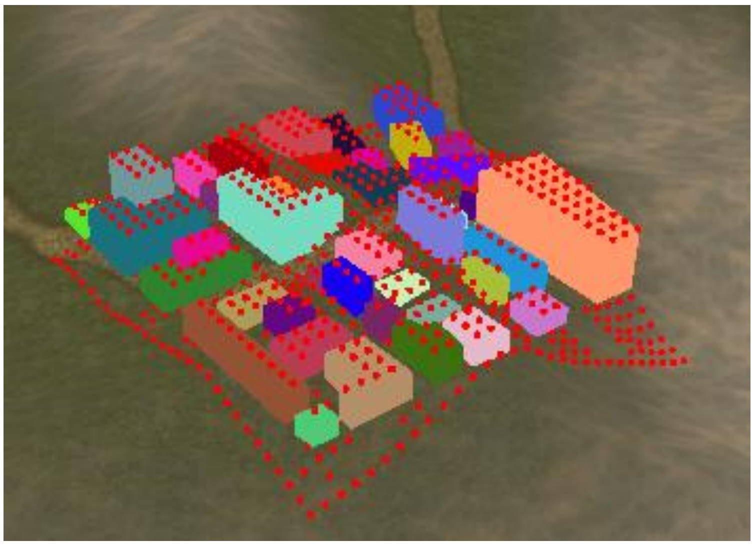
	Without hazard		With hazard	
No. of UAVs	1	2	1	2
No. of vantage points	6	7	6	7
Total path length/m	24	48	54	82


(a) One UAV no hazard (b) Multi UAV without hazard

(c) One UAV with hazard (d) Multi UAV with hazard

Figure 2.18 Computation result of a simple city under different conditions

Another experiment deals with a much more complex case to test the efficiency of the algorithm. As shown in Figure 2.19a, the city model consists of a number of buildings over the uneven terrain, therefore occlusion occurs often. The ***sP***s are also shown in the figure. Figure 2.19b and Figure 2.19c show the obtained ***cP***s in scenarios without and with hazard existence, respectively. Two radar sites are present in Figure 2.19c. Clearly, the obtained ***cP***s are clustered around the periphery of the area, with tilted view directions.


(a) A more complex city environment for coverage path planning

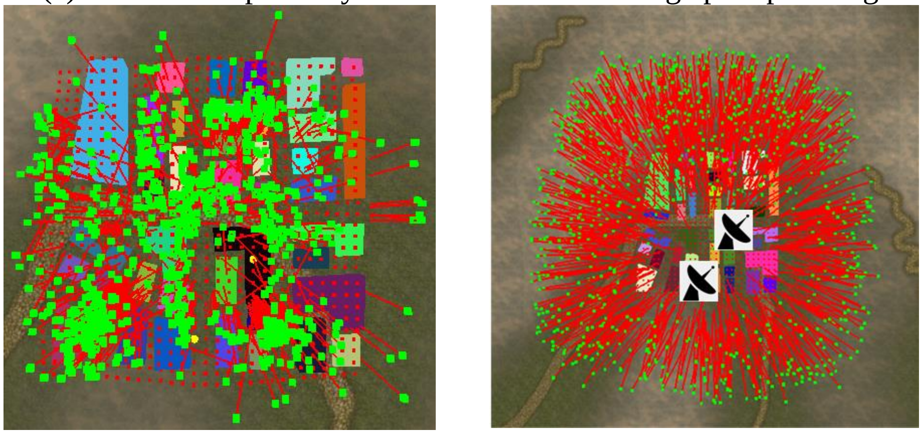
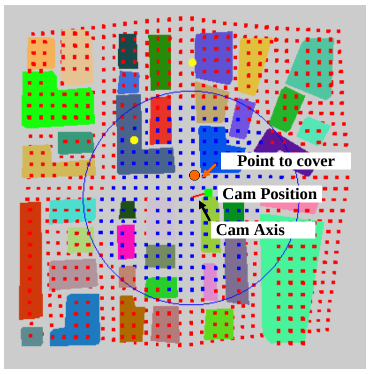
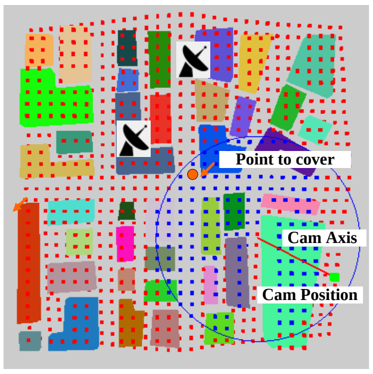

(b) Camera positions (no hazard) (c) Camera positions (with hazard)

Figure 2.19 Determined view direction in a complex city environment

Taking the ***sP*** at the centre of the area for example, the identified ***cP*** without and with hazard are shown in Figure 2.20a and Figure 2.20b, respectively. The ***sP***s inside the circles are covered by the respective ***cP***. The effect of the fitness function is clearly shown: (1) no hazard existence, ***cP*** is close to the centre; (2) with hazard existence, ***cP*** is far away from the centre. The evolvement of the best particle fitness in both the two scenarios are shown in Figure 2.21a and Figure 2.21b, respectively. Clearly, convergence has been achieved in both cases. The mission scenario presented in Figure 2.19 is taken to test the GA-based search algorithm for stage II (vantage set and visiting sequence determination). The UAV flying capability attributes and parameters used in GA are given in Table 2.7.

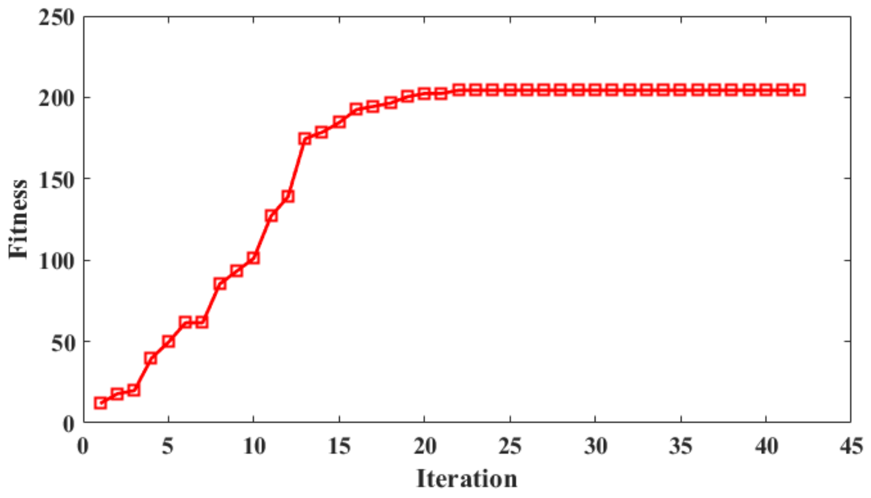


(a) Camera position (no hazard)

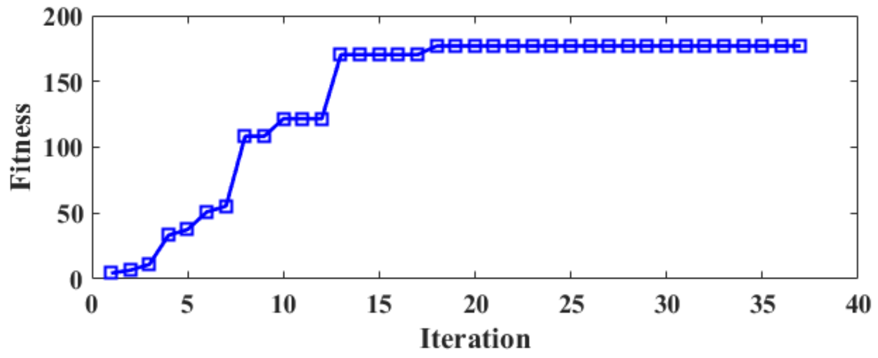


(b) Camera position (with hazard)

Figure 2.20 The obtained ***cP*** for a single ***sP***

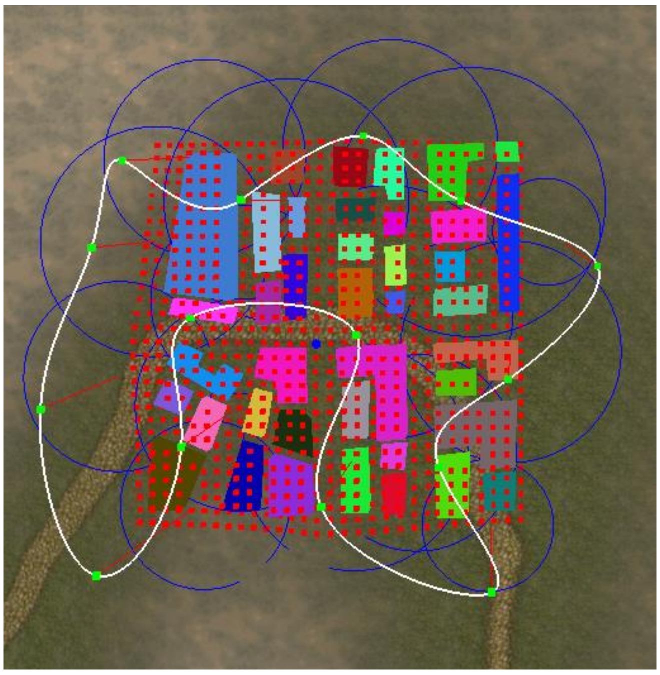


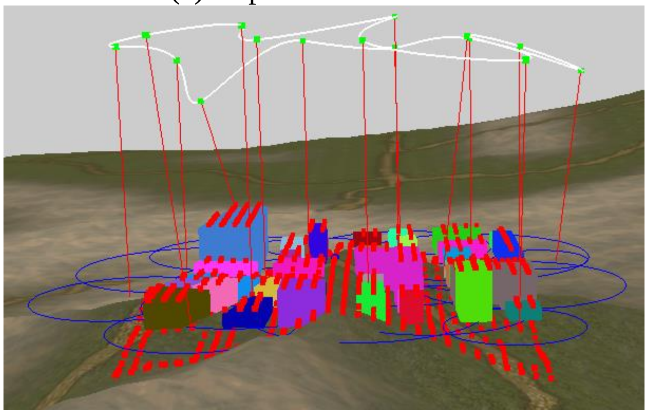
(a) Fitness without hazard existence

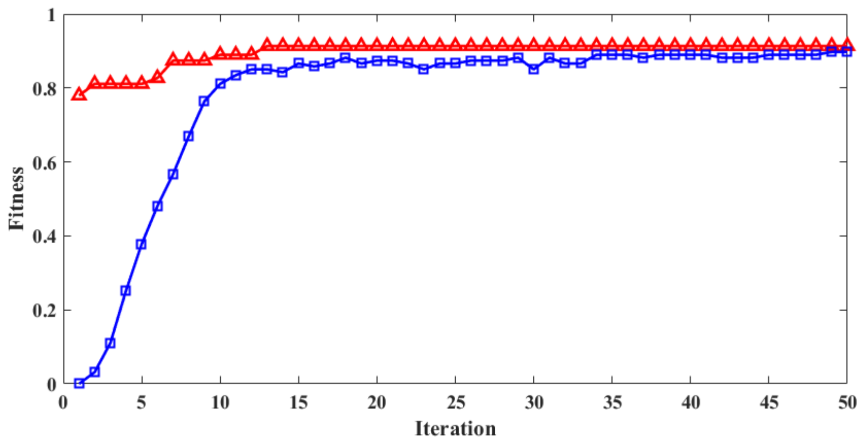


(b) Fitness with hazard existence

Figure 2.21 The fitness evolvement of the global best particle

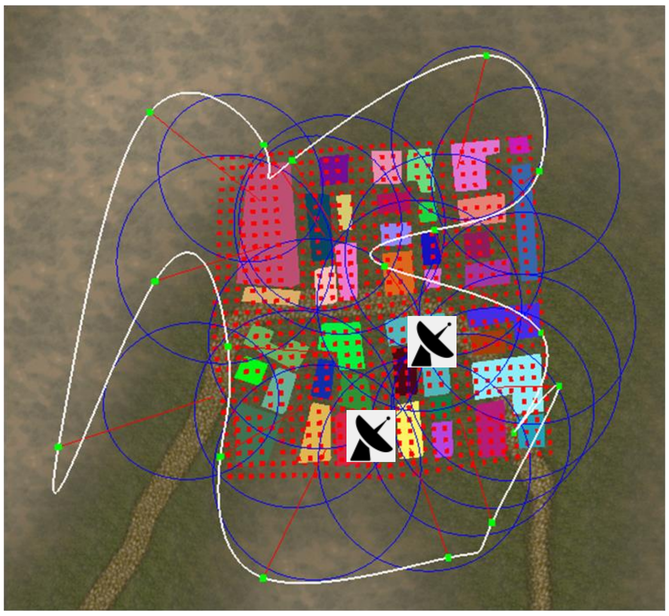

(a) Top down view of the result

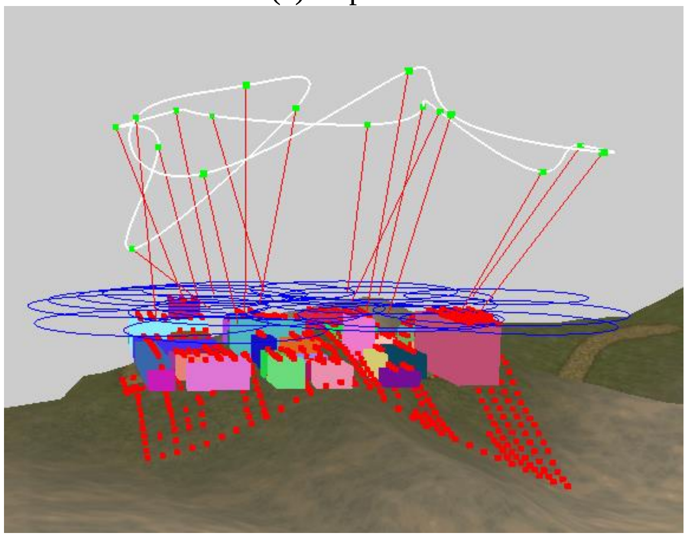

(b) Side view of the result

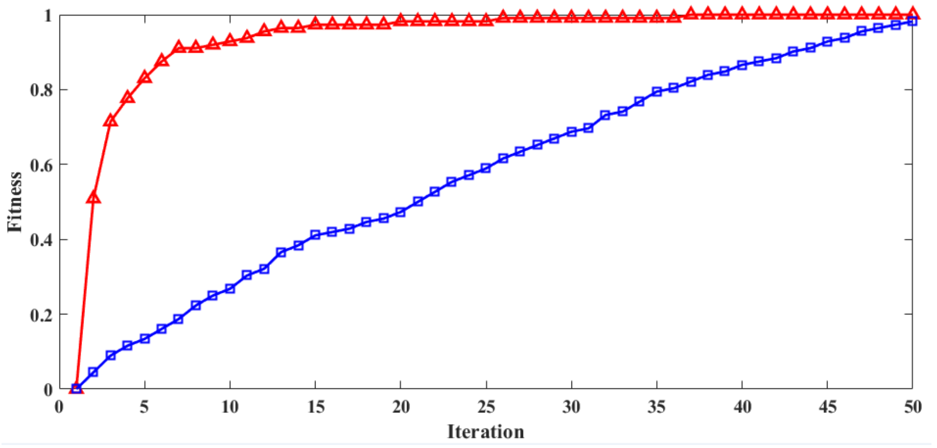


(c) Convergence record without hazard

Figure 2.22 Path planning results in a complex environment without hazard


(a) Top down view


(b) Side view



(c) Convergence record with hazard

Figure 2.23 Path planning results in a complex environment with hazard

Table 2.10 Performance comparison in a complex environment

	Without hazard	With hazard
No. of vantage points	15	16
Total path length/m	62	91
Computation time/s	334	1007

The GA-based algorithm in stage II was tested in two scenarios, without hazard and with hazard. Only one UAV is used for the mission. The ***cP***s obtained without hazard (see Figure 2.19a) and with hazard (see Figure 2.19b) are the input for the stage II. The flying paths, together with the coverage of each ***cP*** in the vantage set, are shown in Figure 2.22 and Figure 2.23, respectively. The convergence records are shown in Figure 2.22c and Figure 2.23c (red: best fitness, green: average fitness). The performance indicators for both the two scenarios are summarized in Table 2.10. It can be clearly seen that the fitness function is reasonable for the optimization. Specifically, in the scenario without hazard, the optimization goes towards the shortest path; while, in the scenario with hazard, the resulted path is kept far from the hazard points.

2.8. Summary

In this chapter, the fixed-wing UAV surveillance mission planning problem has been addressed. The problem is modelled with more realism by considering the presence of hazard points and the pan-tilt capability of the on-board camera. A heuristic is developed to find the optimal view directions regarding to each pair of camera position and surface point. Following this heuristic, the optimal camera configuration is determined for each surface point by a PSO-based algorithm. To construct for the optimal vantage set with GA-based algorithm, the corresponding optimal flying paths for each candidate solution are constructed for accurate fitness evaluation. The effectiveness of the proposed methods has been demonstrated with the case studies.

CHAPTER 3

OFFLINE PERCHING LOCATION SELECTION FOR ROTARY-WING UAV IN URBAN ENVIRONMENT

In this chapter, an offline perching location selection algorithm is developed for rotary-wing UAV to monitor a target, either a point of interest (POI) or a face of interest (FOI). Here, a FOI can be either a horizontal rectangular face or a vertical rectangular face. The selection algorithm is designed in such a way that the task is completed in two steps, preliminary selection and precise selection. In the preliminary selection stage, geometric constraints, including the UAV dimension, camera focus range, roof area, and roof slope, are applied to identify all the feasible roofs for the UAV to land on. In the precise selection stage, line-of-sight check is carried out for each edge of the feasible roofs to determine the feasible landing locations on it. Finally, a set of top ranked landing locations are selected based on the distance to the target to be viewed.

3.1. Background

In the perch-and-stare mission, as shown in Figure 3.1, a rotary-wing UAV flies to the target area and perches on the roof of a building to conduct enduring reconnaissance on a point of interest (POI) or face of interest (FOI). The overall flowchart of this perching mission is shown in Figure 3.2. The perching mission is divided into two steps. In the first step, the mission target is to select a perching location in the offline map. In the second step, after the deployed UAV navigates to the area above the preselected perching location, the LiDAR on-board the UAV collects a 3D point cloud of the surrounding environment (containing the preselected location). Subsequently, the roofs in the point cloud are to be extracted and then matched with the offline map to identify the corresponding zone in the point cloud of preselected perching location. Finally, the identified zone of the landing location and its surrounding area on the roof are checked to determine whether it is feasible for landing.

In the offline selection mission, due to the low resolution of the offline map, the roofs of buildings are assumed to be flat, thus the roughness of the roofs are not to be considered. The requirement for a feasible perching roof is that it has sufficient area and a limited slope. In addition, as the on-board camera has a limited focus range, the perching location must be within a certain distance to the target POI or FOI. Generally speaking, locations close to the edges of roofs are the candidate perching locations. These edges are called *perching edges*.

Given the whole map (2½D or 3D) of the target area, the roof edges that are within a certain distance to the POI or FOI can be considered as candidate perching edges. As shown in Figure 3.3, the position of the on-board camera is usually at the front of the UAV with an elevation of h_c from the base of the UAV. Considering the distance threshold and the line of sight constraints, the effective landing area is near the frontier of the building roof. Therefore, only the roof edges of buildings are considered for perching locations. The candidate roofs can be obtained by shrinking the original building roofs inside by a safety distance d_s and rising by a camera elevation h_c. The scenarios with the target being a POI, horizontal FOI (h-FOI), and vertical FOI (v-FOI) are shown in Figure 3.3a, b and c, respectively.

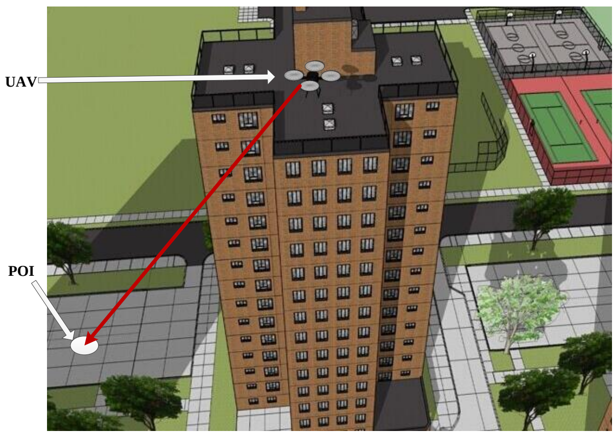


Figure 3.1 Perch-and-stare mission

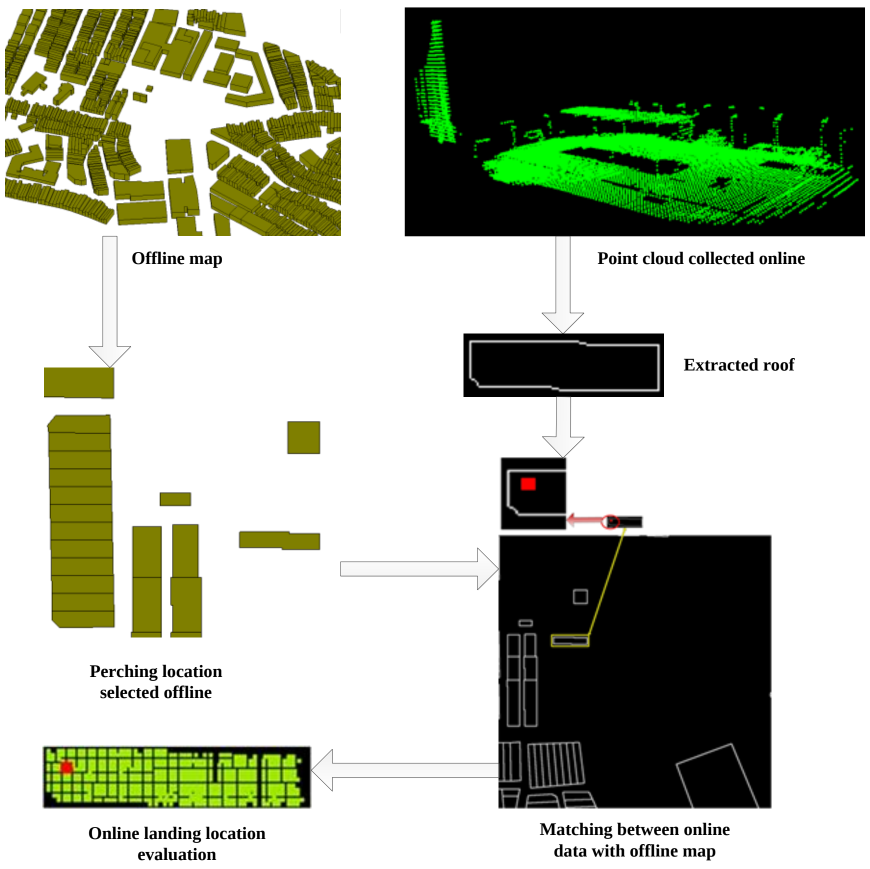
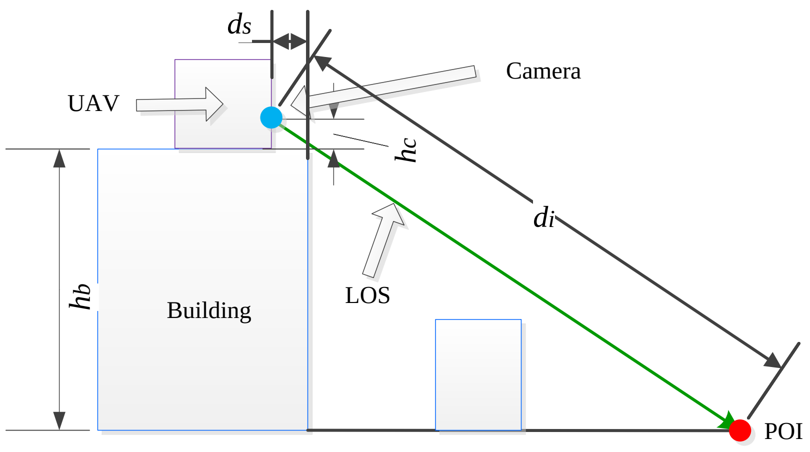
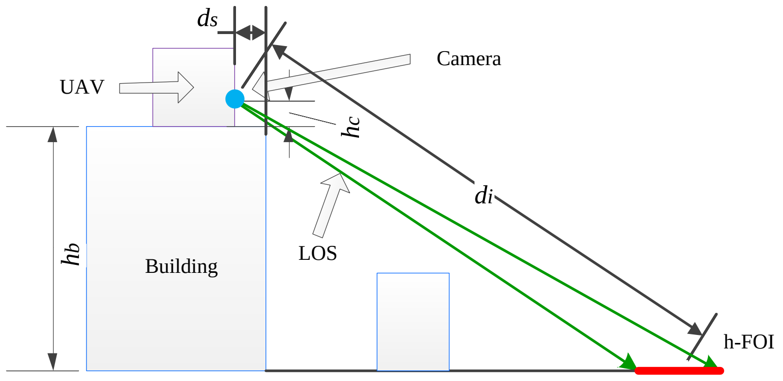

Offline map
Point cloud collected online
Extracted roof
Perching location selected offline
Online landing location evaluation
Matching between online data with offline map

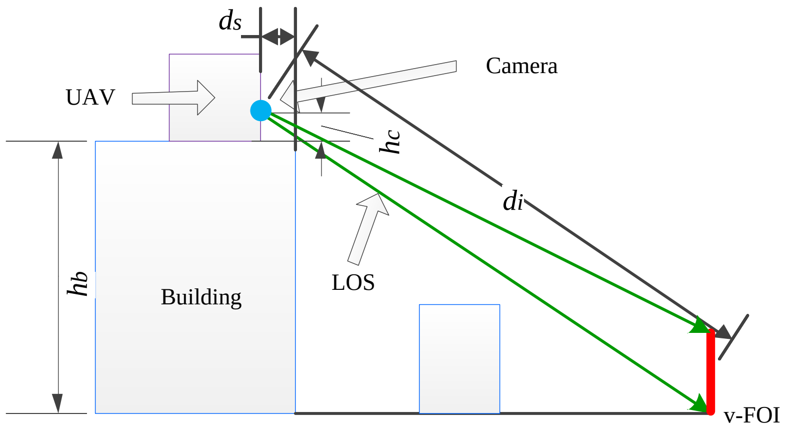
Figure 3.2 Perching mission flowchart



(a) Candidate perching location for POI



(b) Candidate perching location for h-FOI

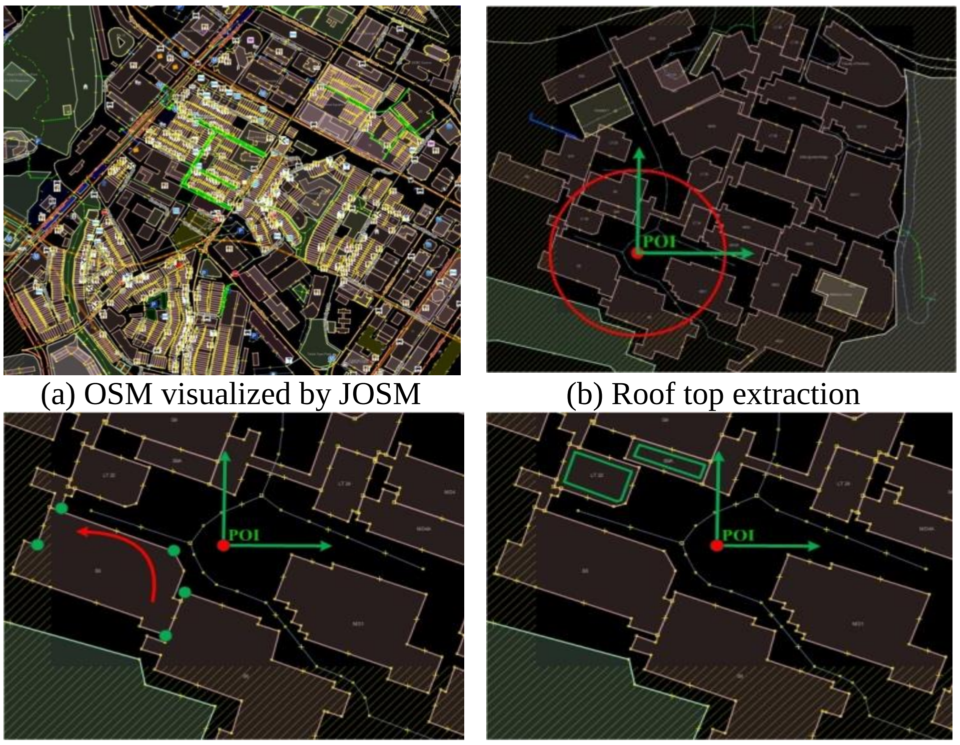


(c) Candidate perching location for v-FOI

Figure 3.3 Candidate perching location

3.2. The Offline City Map

Initially, the city map is given in the OpenStreetMap (OSM) format, which is a kind of 2D map with polygonal contours of objectives, including buildings, roads, and others. An OSM map can be visualized with the software JOSM (see Figure 3.4a). To make use of the OSM map for perching location selection, it is preprocessed and converted to 2½D form that contains buildings only. The roofs are extracted from the OSM map with given heights (see Figure 3.4b). The vertexes of roofs are then stored in counterclockwise sequence (see Figure 3.4c). Finally, the obtained candidate perching edges are stored with the coordinates of edge corners (see Figure 3.4d).



(a) OSM visualized by JOSM (b) Roof top extraction

(c) Building vertices (d) Candidate roofs

Figure 3.4 OSM map pre-processing

3.3. Preliminary Perching Location Selection

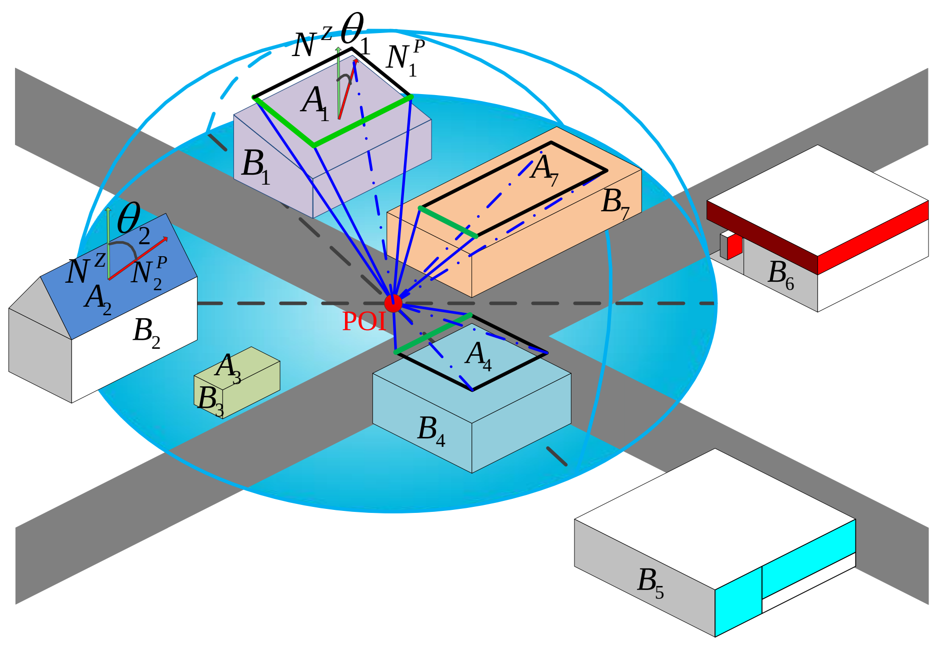
After the OSM map pre-processing is completed, the perching locations are to be selected, which is carried out in two steps: preliminary selection and precise selection. In the preliminary selection, only roofs within the camera range and satisfying the area and slope constraints are selected. The constraints for a feasible perching edge are as follows:

(1) The minimal distance d_i (i = 1, 2, …, n) from the POI to a candidate roof edge should not exceed d_{max}, the maximum sensor range. In the case of FOI, the distance should be measured from each corner point of the FOI. For the example shown in Figure 3.5a, only B_1, B_2, B_3, B_4, and B_7 are within the range of d_{max}, which are thus retained for further checking;

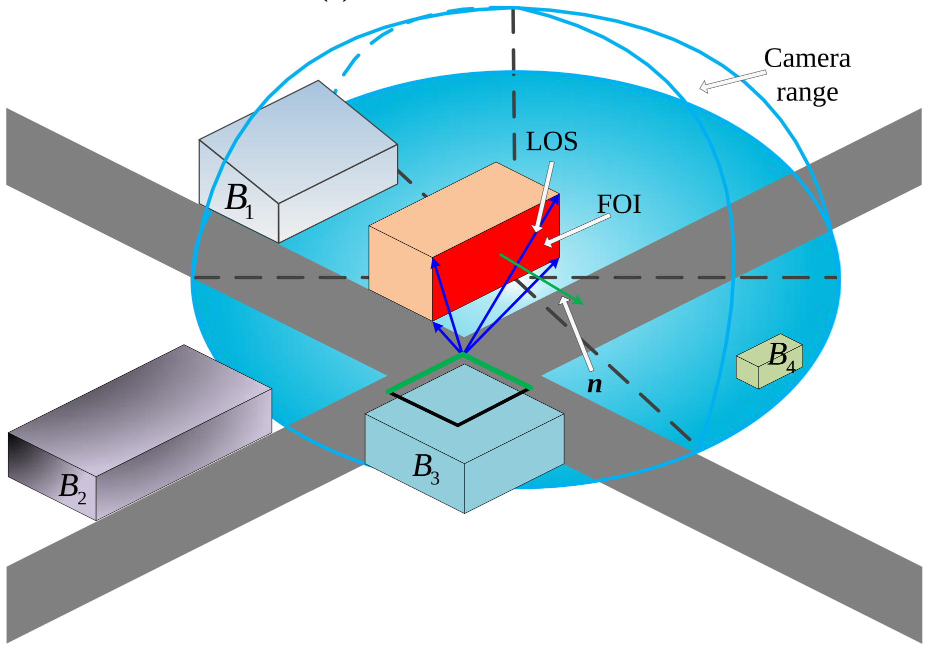
(2) The area of a candidate roof A_i must be larger than the minimal required landing area A_{min} for the UAV, as shown in Figure 3.5;

(3) The slope of a candidate roof, $\theta_{s,i}$, must not exceed, $\theta_{s,max}$, the maximum allowable landing slope angle of the UAV, $\theta_{s,i}$ is defined as the angle between the normal vector of the roof and the z-axis (pointing upwards), as shown in Figure 3.5. For the scenario of FOI, the angle between the normal of FOI and the line of sight (from centre of FOI to a perching point) should not exceed a threshold (less than 90°), such as 85°. As shown in Figure 3.5b, only B_3 and B_4 satisfy this constraint.

On the other hand, the line of sight (LOS) from the perching point to the target should not be occluded (Figure 3.5a). The LOS constraint is not considered here and will be checked in the precise selection stage.



(a) Constraints for POI



(b) Constraints for FOI

Figure 3.5 Slope and LOS condition

3.4. Precise Perching Location Selection

In precise selection, the LOS constraint is to be checked for each candidate perching edge to confirm its feasibility and then the perching locations are selected. Since the implementation procedures for the scenarios of POI and FOI are quite different, they are described separately in the following sections.

3.4.1. Line of sight check for POI

The LOS check process for a POI (e.g., $\boldsymbol{P}$ in Figure 3.6) is as follows: each candidate perching edge (CPE), e.g., $\boldsymbol{AB}$ in Figure 3.6, and $\boldsymbol{P}$ form a view-triangle. Intersection of the view-triangle with any objects in the environment is then checked and the "visible" portion of this CPE is determined, if any. All the identified visible edge portions are considered feasible perching edges (FPEs). For each candidate edge, the intersection check is carried out in three steps:

(1) The whole 3D map is projected onto the X-Y plane, intersection between the projected view-triangle (e.g., $\Delta\boldsymbol{ABP}$ in Figure 3.6) and objects is then checked. If no intersection exists, the whole edge is defined "visible"; otherwise, go to next step.

(2) For each intersected object, intersection is checked between each of its lateral faces (e.g., polygon $\boldsymbol{MNST}$ in Figure 3.6) and $\Delta\boldsymbol{ABP}$. The intersection segment is determined (segment $\boldsymbol{CD}$ in Figure 3.6). Subsequently, the blocked portion of $\boldsymbol{AB}$ can be determined (e.g., segment $\boldsymbol{AE}$ in Figure 3.6).

(3) After all the intersected objects are checked, the complementary set of the union of all the blocked portion of the CPE is determined as the visible portion of it.

For triangle-polygon intersection determination in step (2), a new method is proposed, which is described as follows. As shown in Figure 3.7, $\Delta\boldsymbol{ABP}$ is within plane π_1, and polygon $\boldsymbol{MNST}$ is within plane π_2. The intersection line of the two planes is l. Then, intersection of polygon $\boldsymbol{MNST}$ and l can be determined as $\boldsymbol{CC'}$ (Schneider and Eberly 2002). Similarly, intersection between Δ***ABP*** and *l* can be determined as ***DD´***. Finally, the overlapped portion of ***CC´*** and ***DD´***, i.e., ***CD*** in this case, can be determined as the intersection segment of Δ***ABP*** and polygon ***MNST***.

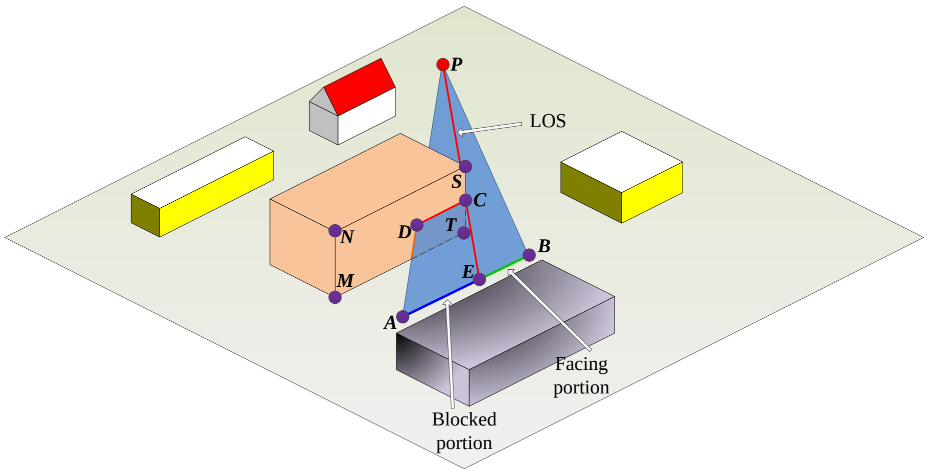


Figure 3.6 LOS check for POI

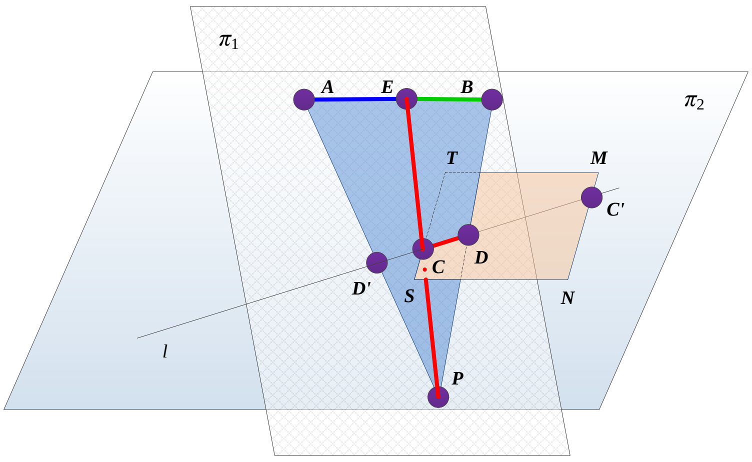


Figure 3.7 Triangle-polygon intersection

3.4.2. Line of sight check for FOI

As for the scenario of FOI, we found that the visible portion of a CPE cannot be determined exactly using analytical methods. Therefore, each CPE is firstly discretized into a set of candidate perching points (CPPs) as shown in Figure 3.8. The line of sight from each CPP to the FOI is then checked to determine whether the CPP is a feasible perching point (FPP). This is done by conducting intersection check between the lateral faces of the view-pyramid (formed by the CPP and the 4 corner points of the FOI, as shown in Figure 3.8) and any objects in the environment. If there is no intersection, this CPP is considered as a FPP. For each CPP, the intersection check is carried out as follows:

(1) Check whether the CPP is facing the positive side of the FOI (e.g., $\boldsymbol{MNST}$ in Figure 3.8) defined by its unit normal vector ($\boldsymbol{n}$). The angle between $\boldsymbol{n}$ and the LOS to the 4 corner points of $\boldsymbol{MNST}$ is calculated. If all the angles are in the range of [95, 180], the CPP is consider a "facing point" (e.g., $\boldsymbol{C}_1$). Otherwise, it is considered as a "blocked point" (e.g., $\boldsymbol{C}_2$) and dropped.

(2) The whole 3D map is projected onto X-Y plane. Then intersection between the projected view-pyramid and objects is checked. If no intersection exists, the CPP is considered a FPP; otherwise, go to next step.

(3) For each intersected object, intersection is checked between each of its lateral faces (e.g., $\boldsymbol{EFGH}$) and each lateral face of the view-pyramid. If any intersection occurs, this CPP is deemed infeasible and dropped. After all the intersected objects are visited and no intersection exists, this CPP is considered a FPP.

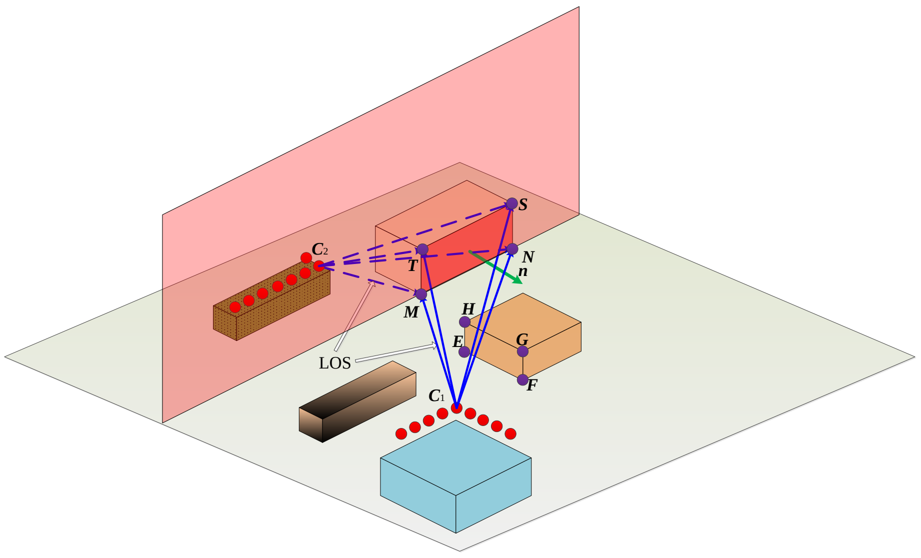


Figure 3.8 Feasibility check for CPPs

For triangle-polygon intersection determination in step (3), a new method is proposed, which is described as follows. As shown in Figure 3.7, $\Delta\boldsymbol{ABP}$ is within plane π_1, and polygon ***MNST*** is within plane π_2. The intersection line of the two planes is *l*. Then, intersection of polygon ***MNST*** and *l* can be determined as ***CC´*** (Schneider and Eberly 2002). Similarly, intersection between $\Delta\boldsymbol{ABP}$ and *l* can be determined as ***DD´***. Finally, the overlapped portion of ***CC´*** and ***DD´***, i.e., ***CD*** in this case, can be determined as the intersection segment of $\Delta\boldsymbol{ABP}$ and polygon ***MNST***.

After the feasible perching edges are determined, the top 3 best perching locations are to be selected. The corners of the perching edge are preferred for the convenience of recognition. The perching locations are ranked by the distance to the target to be viewed, and the top 3 locations with the smallest distance will be selected.

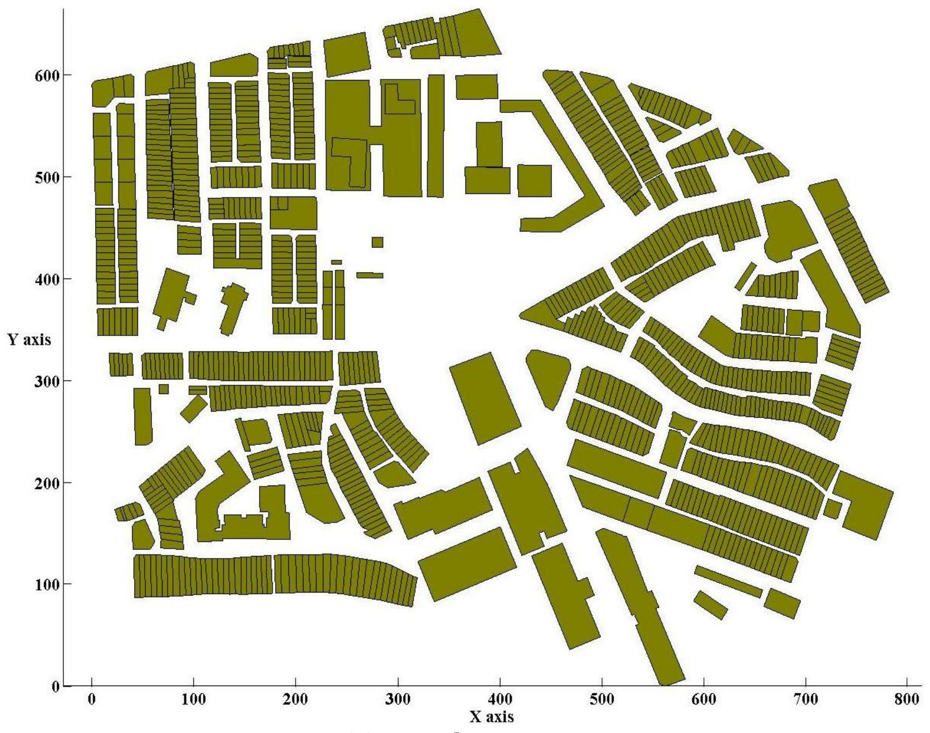
3.5. Numerical Experiments

The offline perching location selection algorithm (for POI and FOI, respectively) has been implemented with JAVA, and the results can be viewed in MATLAB environment. In this section, several testing cases are presented to show the efficacy of the algorithm. In the tests, the parameter settings of the UAV system and target are shown in Table 3.1.

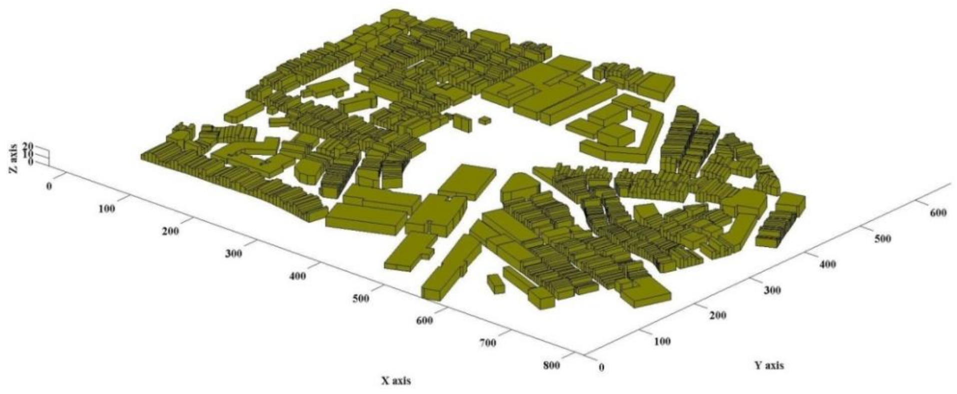
Table 3.1 Parameter Settings in offline perching location selecton

Parameters	**Target type**		
	POI	*h*-FOI	*v*-FOI
UAV dimensions (m)	1× 1× 0.5		
Map size (m)	787× 665		
Camera height (m)	0.5		
Camera range (m)	50		
Safety distance d_s (m)	1		
Target position (m)	(320, 360, 0)	(590, 370, 0) (595, 370, 0) (595, 375, 0) (590, 375, 0)	(275, 430, 0) (285, 430, 0) (285, 430, 5) (275, 430, 5)
Edge discretization interval $d_{e,\,i}$ (m)	N/A	1	1

The input 2½D map as shown in Figure 3.9, consists of around 1000 buildings, which are represented by prisms. Three types of target are used in the tests, i.e., point of interest (POI), horizontal face of interest (*h*-FOI), and vertical face of interest (*v*-FOI). A POI is a point represented by its *x*-*y*-*z* coordinates (the red dot in Figure 3.10a). An *h*-FOI is a rectangular facet facing up on the ground, represented by its 4 corner points forming an anti-clockwise loop (the red rectangle in Figure 3.10b). A *v*-FOI is a rectangular facet on the façade of an object, represented by its 4 corner points forming an anti-clockwise loop (the red rectangle in Figure 3.10c). Its normal vector (pointing out) can be obtained using the right-hand rule. The *x*-*y*-*z* coordinates of the POI, the corners of the *h*-FOI, and *v*-FOI are shown in Table 3.1, respectively.

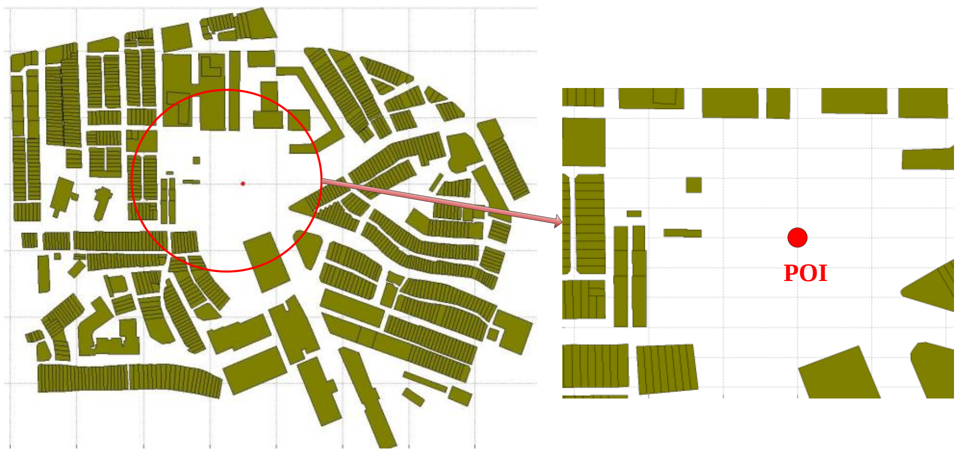


(a) Top down view

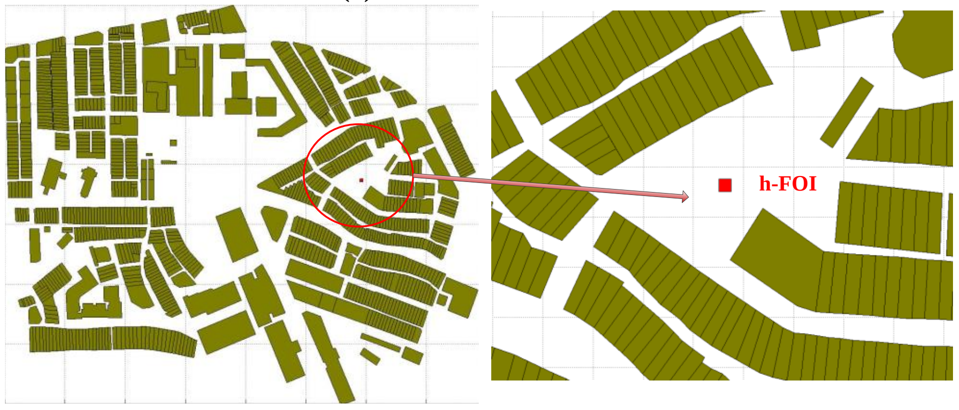


(b) Isometric view

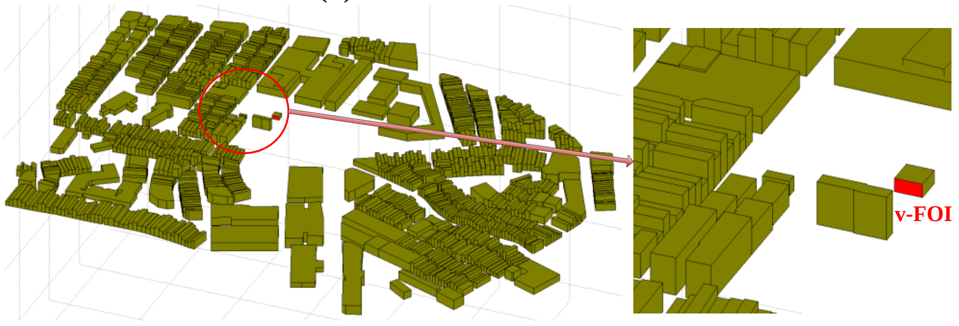
Figure 3.9 The 2½D Map of the environment



(a) Scenario 1 - a POI



(b) Scenario 2 - an h-FOI

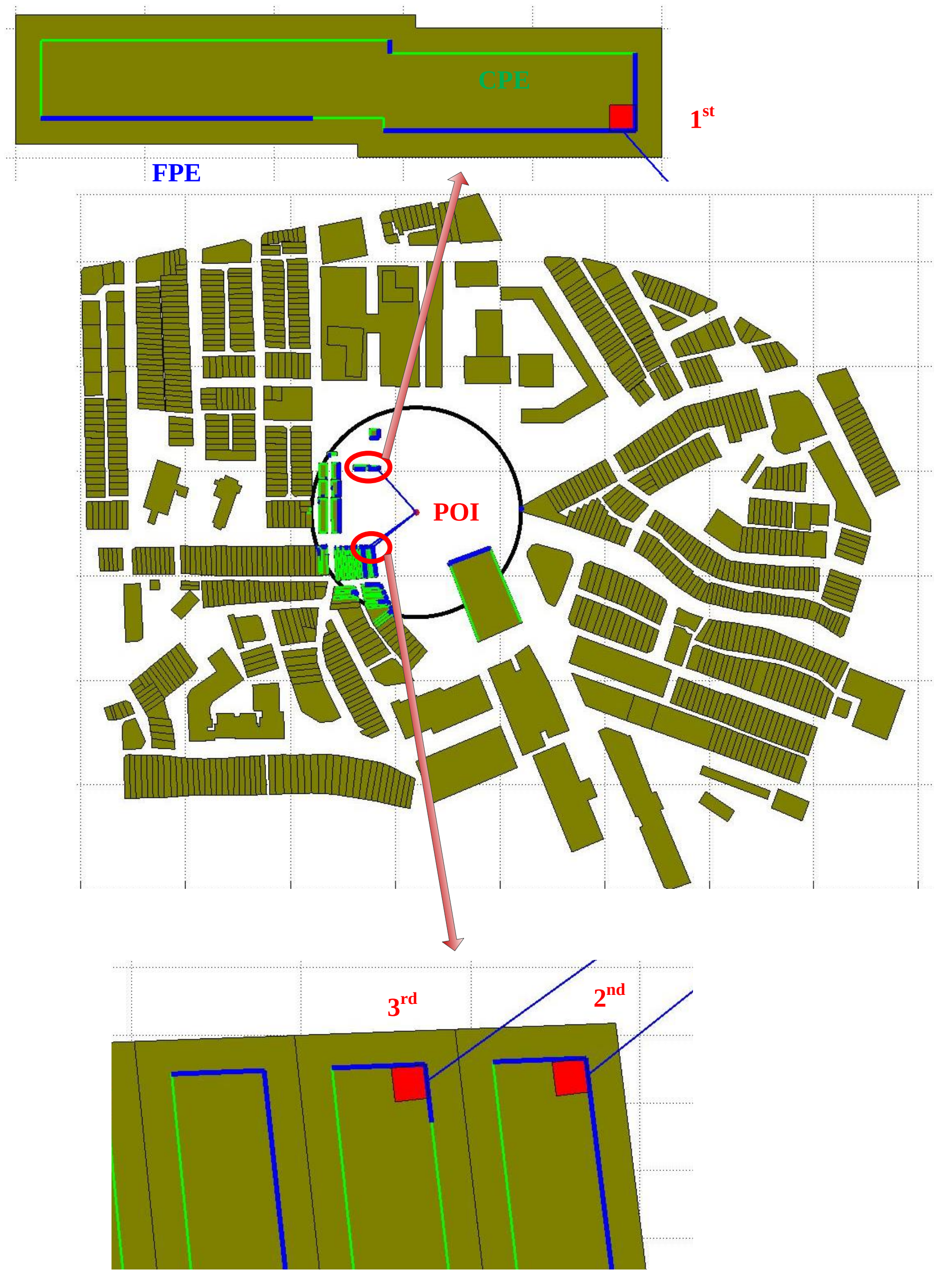


(c) Scenario 3 - a v-FOI

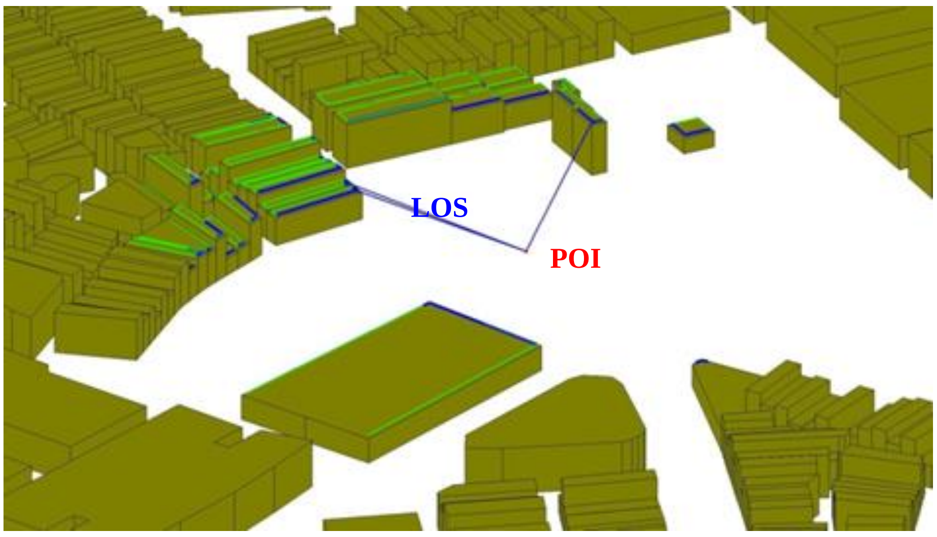
Figure 3.10 Three scenarios in the tests

3.5.1. Scenario 1 - case with a POI

Figure 3.10a shows the scenario with a POI. After running the perching location selection algorithm, the results are shown in Figure 3.11. The black circle in Figure 3.11a is centred at the POI and its radius is the camera range. The roofs within this circle are thus selected as candidates. Subsequently, the roof edges of these candidates are shrunk inwards by the safety distance d_s to finally form the candidate perching edges (the green contours). This is the end of preliminary selection stage.



(a) Top down view



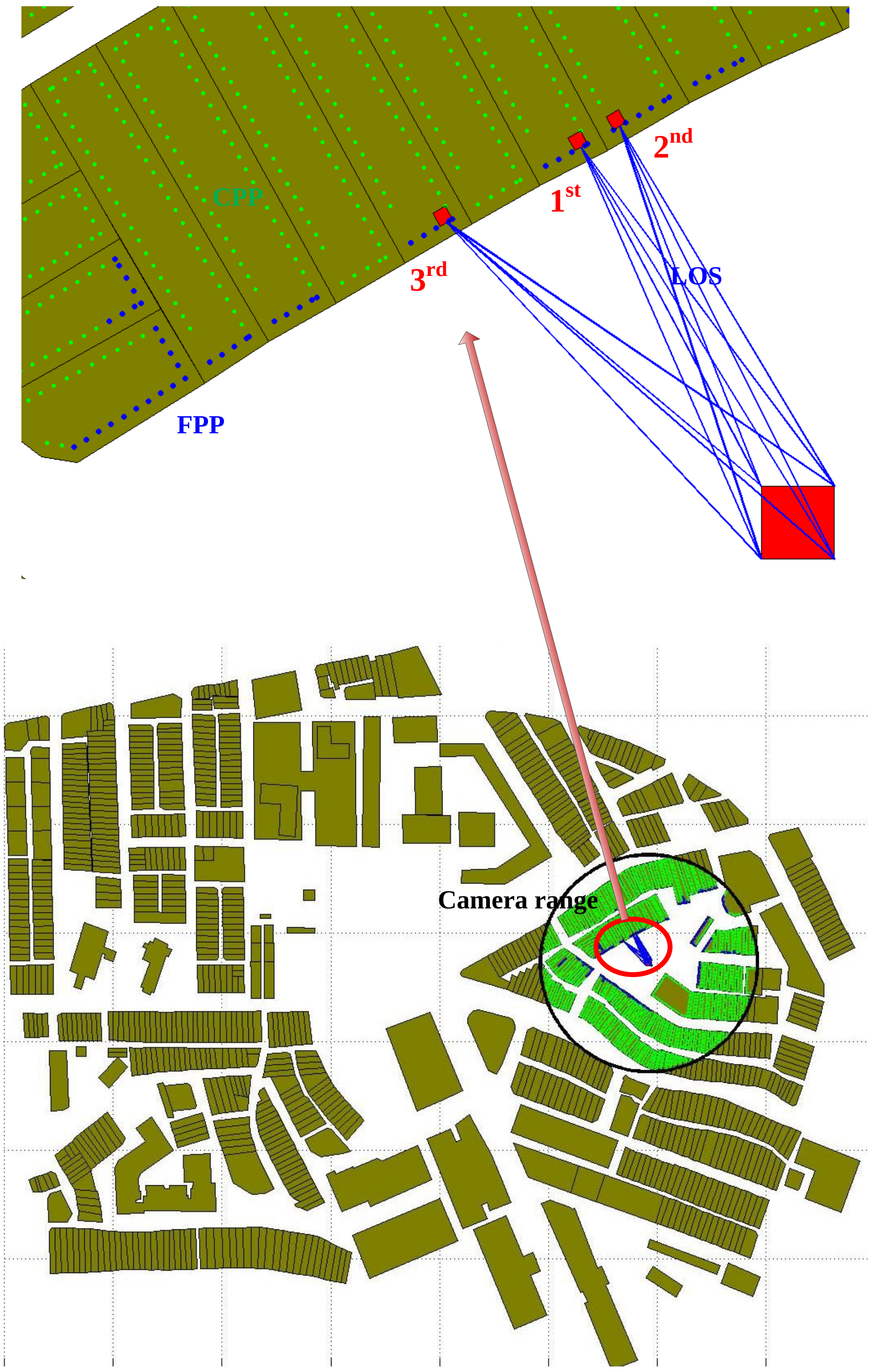
(b) Isometric view

Figure 3.11 Test result for scenario-1

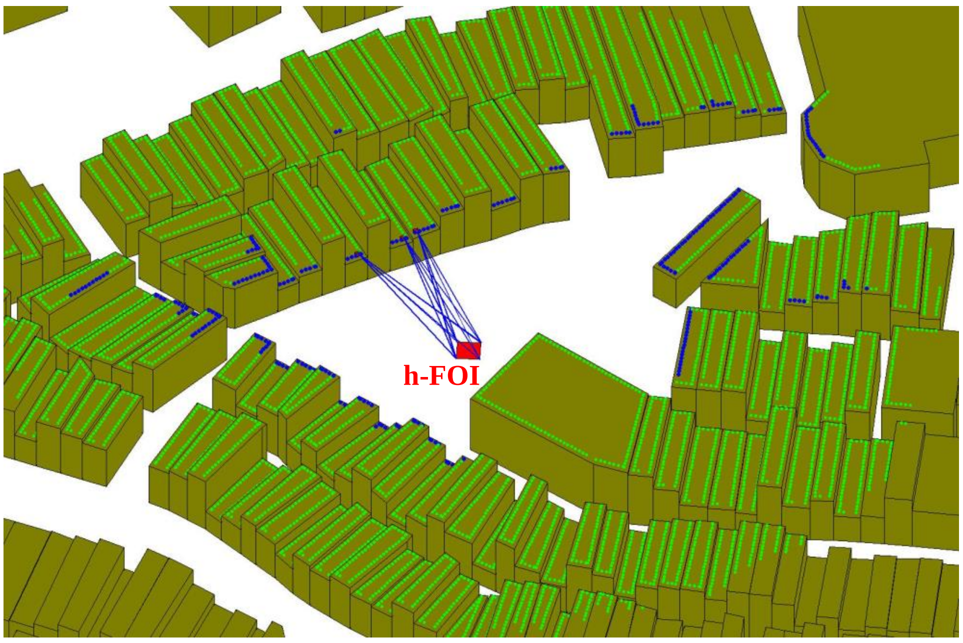
The precise selection is then carried out by checking each candidate perching edge. As a result, the feasible perching edges (FPEs) are determined (see the blue segments in Figure 3.11). Finally, the three top ranked perching points are selected from the FPEs based on the criteria (shown as red boxes in the enlarged views in Figure 3.11a). The total computation time for both the preliminary and precise selection is 116ms (on a workstation with 3.2 GHZ CPU and 16GB RAM).

3.5.2. Scenario 2 - case with an h-FOI

Figure 3.10b shows the scenario with an *h*-FOI in which the *h*-FOI is placed at a different position. After running the offline perching location selection algorithm, the results are shown in Figure 3.12.



(a) Top down view



(b) Isometric view

Figure 3.12 Test result for scenario-2

The black circle in Figure 3.12a shares the centre with the *h*-FOI and its radius is the camera range. The roofs within this circle are thus selected as candidates. Subsequently, the roof edges of these candidates are shrunk inwards by the safety distance d_s to finally form the candidate perching edges (the green contours). The green dots are the result of discretization of the candidate perching edges. After the precise selection is completed, the feasible perching points (FPPs) are determined and shown in blue. Finally, the three top ranked perching points are selected from the FPPs based on the criteria. They are shown in red in the enlarged views in Figure 3.12a. The total computation time for both preliminary and precise selection is 871ms (on a workstation with 3.2 GHZ CPU and 16GB RAM). More experiments have been carried out to test how the computation time increases with the decrease of edge discretization interval $d_{e,\,i}$. The testing result is shown in Table 3.2.

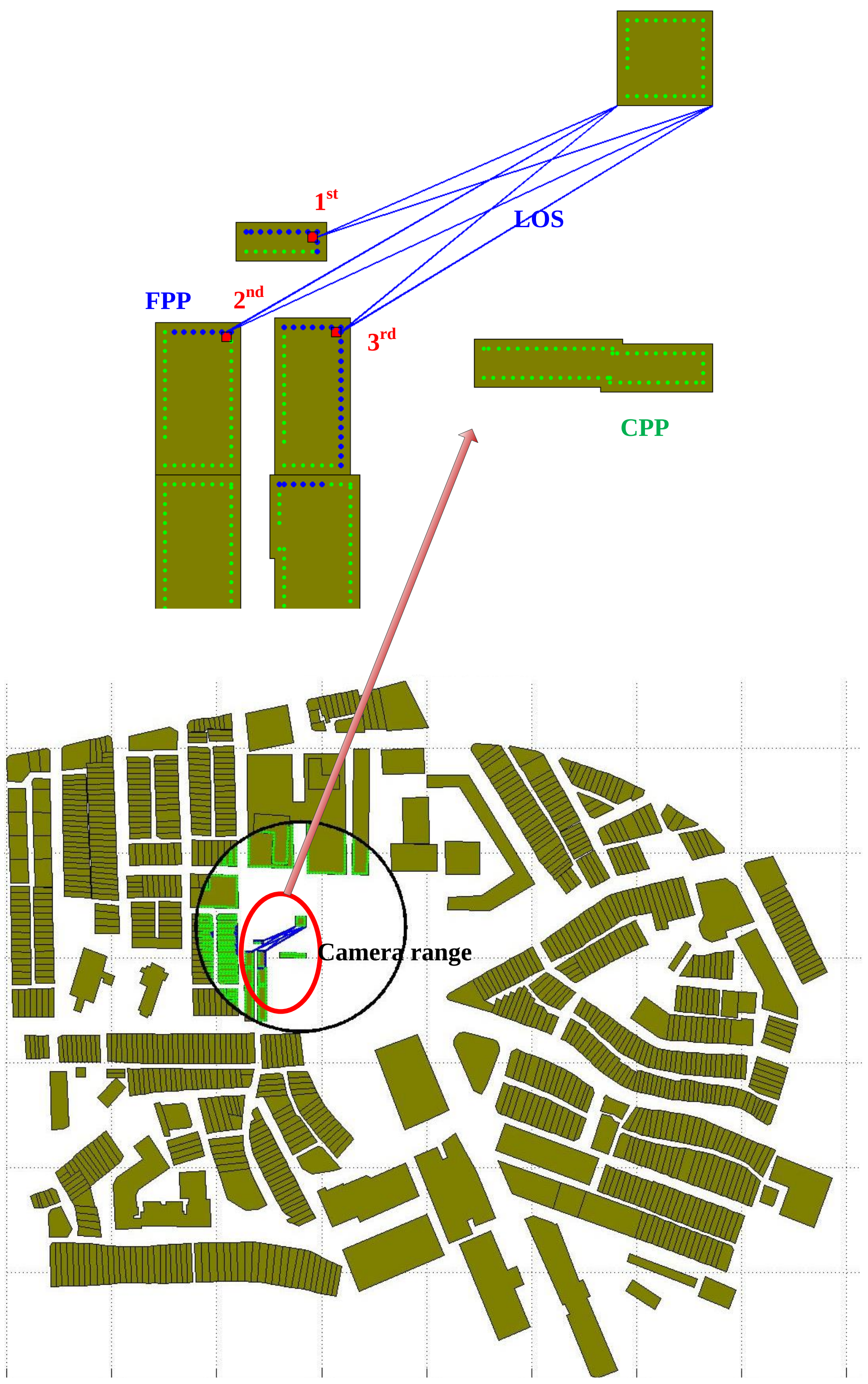
Table 3.2 Computation cost test in the scenario with *h*-FOI

Edge discretization interval $d_{e,i}$ (m)	**Computation time (ms)**
1	871
0.5	1129
0.2	1999
0.1	3334
0.01	20533

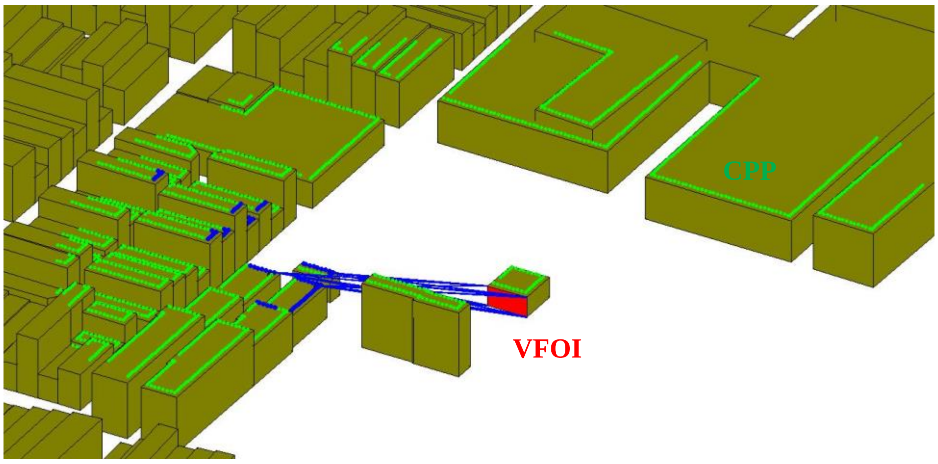
3.5.3. Scenario 3 - case with a v-FOI

Figure 3.10c shows the scenario with a *v*-FOI in which the *v*-FOI is placed at a different position. After running the offline perching location selection algorithm, the results are shown in Figure 3.13.

Again, the black circle in Figure 3.13a shares the centre with the *v*-FOI and its radius is the camera range. The roofs within this circle are thus selected as candidates. Subsequently, the roof edges of these candidates are shrunk inwards by the safety distance d_s to finally form the candidate perching edges (the green contours). The green dots are the result of discretization of the candidate perching edges. After the precise selection is completed, the feasible perching points (FPPs) are determined and shown in blue. Finally, the three top ranked perching points are selected from the FPPs based on the present criteria. They are shown in red in the enlarged views in Figure 3.13a. The total computation time (on a workstation with 3.2 GHZ CPU and 16GB RAM) for both preliminary and precise selection is 521ms. More experiments have been carried out to test how the computation time increases with the decrease of edge discretization interval $d_{e,i}$. The testing result is shown in Table 3.3.



(a) Top down view



(b) Isometric view

Figure 3.13 Test result for scenario-3

Table 3.3 Computation cost test in the scenario with *v*-FOI

Edge discretization interval $d_{e,i}$ (m)	**Computation time (ms)**
1	521
0.5	616
0.2	781
0.1	1109
0.01	3506

3.6. Summary

This chapter addresses the perching location selection problem for rotary-wing UAVs in an off-line mode. Both geometric constraints and mission constraints are considered in the selection procedure, including camera range, roof area, slope, and line of sight (LOS). The output from the selection is either FPEs (for a POI) or FPPs (for FOI). Finally, the top 3 ranked perching locations are identified from the FPEs or FPPs. The algorithm has been implemented and tested with fairly realistic complex problems with satisfactory performance.

CHAPTER 4

ONLINE LANDING LOCATION SELECTION FOR ROTARY-WING UAV IN A GPS-DENIED URBAN ENVIRONMENT

In this chapter, an online landing location selection algorithm is developed for rotoray-wing UAV in a GPS-denied urban environment. Before a UAV carries out a perch-and-stare mission, the landing location (on roof of a building) is already preselected in the 2½D map of the urban environment. The UAV is expected to navigate based on the google map to reach the target region, somewhat above the preslected landing location/roof. The 3D point cloud of the target region is then collected using a downward-facing 2D LiDAR onboard the UAV. The proposed online algorithm should be able to extract the roof contours from the point cloud, and then match them with the offline 2½D map to locate the preselected perching location. Finally, the landing feasibility of the preselected perching location is evaluated. This chapter introduces a new solution to the problem of UAV perch-and-stare mission, which consists of two tasks. The first task is to identify the preselected landing location from the point cloud. And the second task is to check the feasibility of the location for landing. In the first task, the 3D point cloud is firstly converted to an image. And a new *rotation invariant matching* algorithm is proposed to match the two images with not so sufficient amount of features. Finally, experiments are carried out to demonstrate the effectiveness of the algorithms.

In the perch-and-stare mission, the whole online algorithm is carried out onboard the UAV. Limited by the power of the on-board processor and the flight duration of the UAV, the point cloud processing should be completed in real time, for example, in several seconds. Therefore, the efficiency of the algorithm is quite critical. Although quite a lot of algorithms have been developed for the point cloud processing and roof contour extraction, the low efficiency makes them not applicable in the mission.

Template matching (Briechle and Hanebeck 2001) can be used to match the images with not enough features. Nonetheless, this method is not rotation invariant. The prerequisite that two input images have the same orientation cannot be assured in our problem. In addition, some feature matching algorithms, such as SIFT (Lowe 2004) and SURF (Bay et al. 2006), are rotation invariant. However, there are not enough features for matching in the input images of our problem. Therefore, new algorithms have to be developed in order to solve the aforementioned limitations. In this study, the following is to be done:

- Based on preliminary knowledge about the environment, some heuristic constraints are to be implemented to conduct point cloud filtering.
- A rotation invariant template matching algorithm is developed for matching the extracted contour and the offline map.

4.1. Background

In our perch-and-stare mission, a 2½D map of the urban environment (mainly buildings) is available in which a preselected landing location is specified. The

UAV is expected to navigate based on the google map to reach the target region. When the UAV hovers over the target region, an on-board 2D LiDAR is usually employed to scan across the region underneath the UAV. The collected data are then converted to 3D point cloud in the same frame. The proposed online UAV landing location selection algorithm takes over from here. The task is to (1) identify the preselected landing location from the 3D point cloud and (2) check the vicinity of this location to determine its feasibility for landing.

To our knowledge, there is no reported work towards a unified solution to directly address this kind of problem. In this work, we propose the following steps to process the point cloud for accomplishing task (1):

(i) Data filtering

(ii) Roof contour extraction

(iii) Registration of extracted roof contours with the 2½D map

As for task (2), the landing feasibility of the identified location can be determined by checking the area, slope, and surface roughness of its vicinity zone. In the problem, a rotary-wing UAV with a size of around 1m×1m×0.5m is employed. The 2D LiDAR is mounted looking downward from the UAV, and keeps tilting at a frequency of 1Hz between ±40°. Its maximal detected range is 30m. The basic assumptions in our problem are listed as follows:

- The offline map is available and the landing zone has been preselected on the offline map.

- The UAV can navigate to the preselected landing zone, and collect 2D laser scans containing the preselected landing location with the on-board 2D LiDAR.
- The collected 2D laser scans have been fused to a 3D point cloud.
- The parameters to be determined are as follows: the relative position (x, y) of the point cloud within the offline map, and the heading orientation.

The flowchart for the proposed 3D point cloud processing is as shown in Figure 4.1.

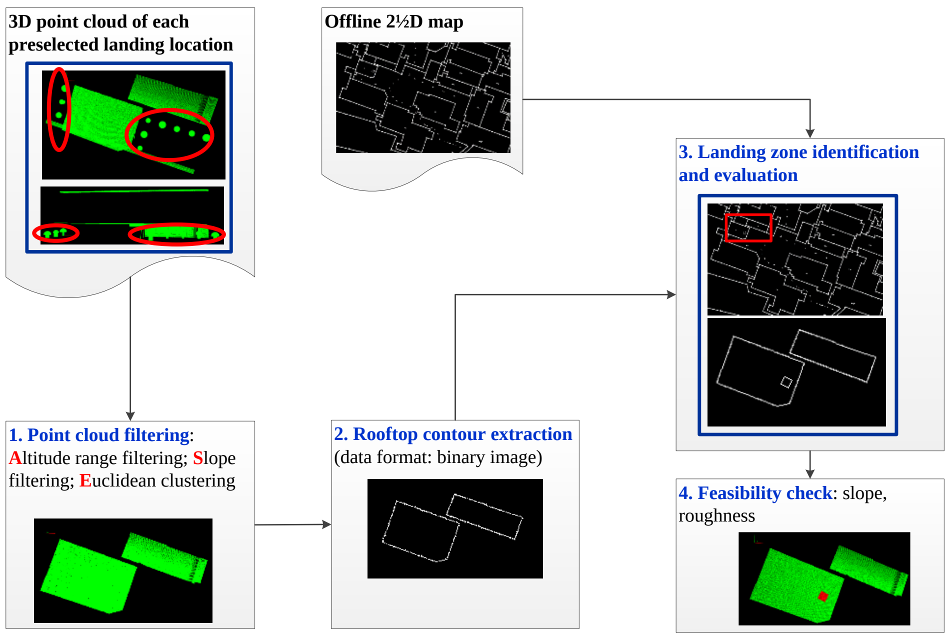

3D point cloud of each preselected landing location
Offline 2½D map
3. Landing zone identification and evaluation
1. Point cloud filtering: Altitude range filtering; Slope filtering; Euclidean clustering
2. Rooftop contour extraction (data format: binary image)
4. Feasibility check: slope, roughness

Figure 4.1 Flowchart for the perching mission

4.2. Related Works

For each step in the mission of online landing location selection, the related works will be reviewed in this section. Based on these reported research works, our proposed methods will be presented.

4.2.1. Point cloud filtering

The data obtained from the 2D LiDAR are a set of unordered, irregular 3D point cloud, which is difficult to analyse. It is necessary to build a topology relationship between the points. Some researchers proposed to regularize the point cloud into a digital elevation map (DEM) (Alharthy and Bethel 2002), where the points can be located by their indices. The DEM can then be regarded as a grey level image, so the methods of image segmentation can always be adopted to process the DEM. However, processing error is inevitable in conversion from point cloud to DEM, which will degenerate the accuracy of roof contour extraction. Some researchers proposed to organize the point cloud with the k-d tree structure (Wald and Havran 2006).

Region growing method was proposed to segment the point cloud, which is based on the idea of connecting the pixels with the homogenous characteristics. The process of the method is to select a small area as the initial seed and then combine the pixels around the seed gradually until it satisfies the pre-defined stopping criteria (Tóvári and Pfeifer 2005). However, the computation efficiency of region growing method is too low for implementation in online point cloud processing.

4.2.2. Roof contour extraction

In online landing location selection, the ideal landing location is roof. Given the 3D point cloud collected by the 2D LiDAR, the roofs are to be detected and the contours are to be extracted. In the literature, Tong et al. (2004) proposed to use the property of neighbourhood to determine boundary points. A point is determined as boundary point if the difference between the two largest eigenvalues of its neighbours exceeds a threshold. However, the algorithm is too sensitive since the distribution of points is not always consistent. Belton and Lichti (2006) proposed a new algorithm that if the position difference between a point with the centroid of neighbour is large enough, it is determined as a boundary point.

4.2.3. Registration between point cloud and image

In our mission, the input data consist of two parts. The offline map is an image, however, the online collected data are 3D point cloud. Registration is therefore to be carried out between the two different types of data. Vasile et al. (2006) proposed an algorithm to align colour imagery to 3D laser data. They proposed to derive pseudo-intensity images from laser data and match them with the colour images. This algorithm requires the knowledge of sun position to derive a 3D sun-ray tracing model. However, such model is quite sensitive to the weather and the luminous intensity, which is the major drawback of this algorithm. Meanwhile, Mastin et al. (2009) proposed to use mutual information as a metric to measure the similarity between the 2D LiDAR data and optical images. They considered the joint entropy among three attributes, LiDAR detection probability, LiDAR elevation, and image illuminance. However, the algorithm requires small direction deviation between the two input data sets.

The prerequisites of the above-mentioned methods of direct registration between point cloud and image cannot be satisfied in our mission. If the point cloud is converted to an image, then the methods of image matching can be utilized.

One of the widely used techniques in image matching is feature matching, which finds the correspondence between two images by detecting and matching features in the two images. Harris and Stephens (1988) proposed the Harris feature to detect corners in images. The intensive computation cost of Harris feature restricts their implementations. Rosten and Drummond (2006) proposed high-speed detector called FAST feature. However, it is not rotation invariant. Lowe (2004) proposed a scale and rotation invariant feature called SIFT by computing a set of sub-octave Difference of Gaussian filters and creating a histogram of all the gradient orientations. Subsequently, Bay et al. (2006) proposed SURF feature to speed up the SIFT.

On the other hand, template matching is also an important technique, which compares portions of the images with another to find similarity between the two images (Briechle and Hanebeck 2001). There's no requirement to extract features for template matching, so it can be implemented in the cases of images without enough features. A basic prerequisite of template matching is that the orientations of the two images should be in accordance. So the major drawback of the algorithm is not rotation invariant.

4.2.4. Landing location evaluation

Given the data describing the terrain in a certain region, the feasibility of the landing location is to be evaluated according to some criteria. Most works used the slope and roughness to determine whether a site is suitable for landing (Johnson et al. 2002, Johnson et al. 2005, Serrano 2006). Johnson et al. proposed to track the image collected by the camera, and extract the features to evaluate landing location. Serrano utilized the method of plane fitting to extract the feature to evaluate the surface. Some researchers also considered landing location size and the distance to the nearest non-feasible point (Takahashi et al. 2013). They firstly converted the LiDAR data into a grid map by interpolation, then used a circle window to search feasible landing locations, and finally used fuzzy logic to rank the potential landing locations. Other researchers even considered the contact condition and stability of aircraft in selecting landing location (Scherer et al. 2012). The evaluation consists of two stages, rough evaluation and fine evaluation. In the rough evaluation stage, the area was divided into regular cells and the cells eligible for slope and roughness criteria were evaluated in the next stage continuously. Then the skid contact area and pose angle on terrain were calculated with discrete landing orientation of the aircraft. However, as the map is divided into cells artificially, some feasible landing location may not be considered.

4.3. Point Cloud Filtering

As the collected 3D point cloud includes not only points of the preselected roof, but also the points belonging to the environment surrounding the roof, in order to extract the contour of the roof, the point cloud is to be filtered first. Point cloud filtering is not a trivial problem, and no general method can deal with all this kind of problems well. In our mission, since there is some preliminary knowledge about the environment, some heuristic constraints can be implemented to conduct point cloud filtering. The basic heuristics proposed in this study are as follows:

- The estimated height of a preselected roof from the ground is known. Considering the measuring noises of laser, the height range of the points representing the roofs can be bounded.
- The slope of the preselected roofs should be under certain threshold.
- The points sufficiently near to each other can be considered to belong to the same object. So the point cloud can be separated into different clusters based on the distance constraint.
- The area of the preselected roof is above certain range, so the isolated points occupying not sufficient area should be excluded from the point cloud.

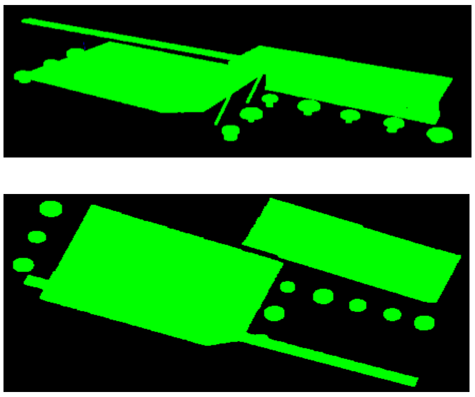
Based on the heuristics above, the point cloud filtering algorithm is proposed as follows (refer to Figure 4.2 for illustration):

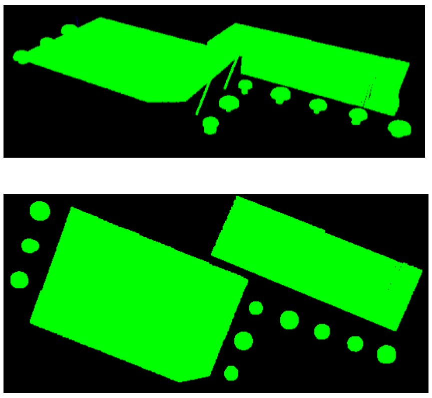
(1) *Altitude range filtering*: Given the point cloud set as shown in Figure 4.2a, assume the height of preselected roof is between h_{min} and h_{max}, the points out of the range is filtered out (see Figure 4.2b).

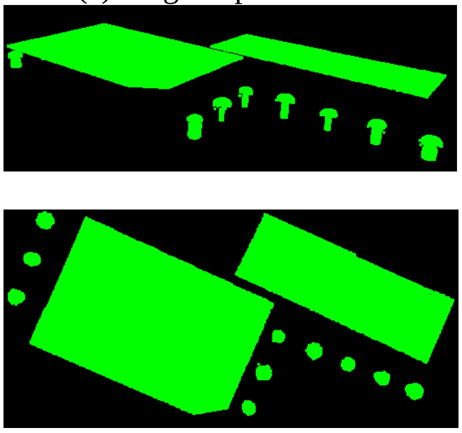
(2) *Slope filtering*: The point cloud is organized with k-d tree structure (Wald and Havran 2006). For each point, the neighbour points within certain radius are determined as nearest neighbours (see Figure 4.3a). And PCA method (Pauly et al. 2002) is utilized to calculate the local normal of the point (λ_0, λ_1, λ_2) and its projection on each axis (a, b, c) (see Figure 4.3b). The inclination angle of a plane is defined as Equation (4-1). The points with inclination angle larger than the threshold 45° is removed (see Figure 4.2c).

(3) *Euclidean clustering*: The points are separated into several clusters based on the Euclidean distances, and the clusters with points less than certain threshold n_{min} are removed (Rusu 2010) (see Figure 4.2d).

$$\lambda_p = \cos^{-1}\frac{|a|}{\sqrt{a^2+b^2+c^2}} < 45° \tag{4-1}$$


(a) Original point cloud


(b) After altitude range filtering


(c) After slope filtering

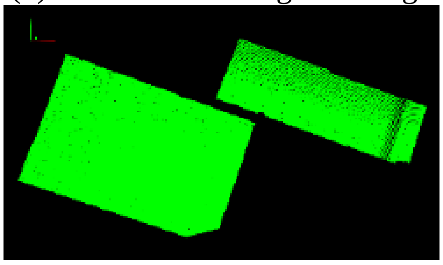
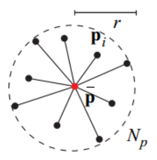
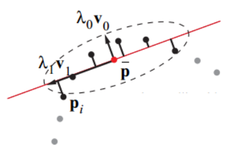

(d) After clustering

Figure 4.2 Point cloud filtering process



(a) Point neighbors



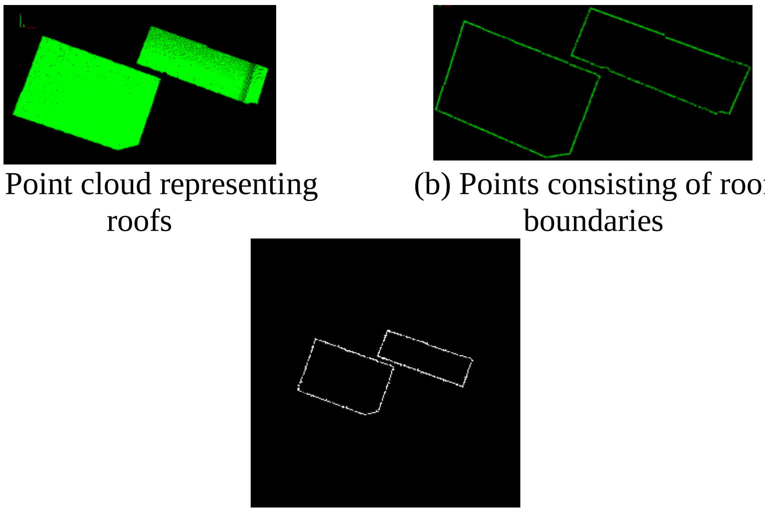
(b) PCA method

Figure 4.3 Point normal calculation

4.4. Roof Contour Extraction

The point cloud obtained after point cloud filtering represents the roofs. Then, the landing conditions of which are to be evaluated. In order to check whether the roofs are the preselected ones, the point cloud is to be registered with the offline map. As the offline map includes contour of roofs, the most straightforward method is to extract the roof contour from the point cloud, and match it with the offline map. The proposed method is described as follows:

1) The 3D point cloud is projected onto the X-Y plane (see Figure 4.4a).
2) The points representing roof boundaries are extracted (see Figure 4.4b). The method of α-hull (Akkiraju et al. 1995) is utilized to extract roof boundary. α-hull is a generalized concept of convex hull, with a parameter α_h to control the radius of searching next boundary point (see Figure 4.5).
3) The point cloud is converted to a binary image (see Figure 4.4c) with certain resolution r_{es}, in which “1” represent the boundary of roof, and “0” others.


(a) Point cloud representing roofs

(b) Points consisting of roof boundaries

(c) Image representing roofs

Figure 4.4 Contour extraction from point cloud

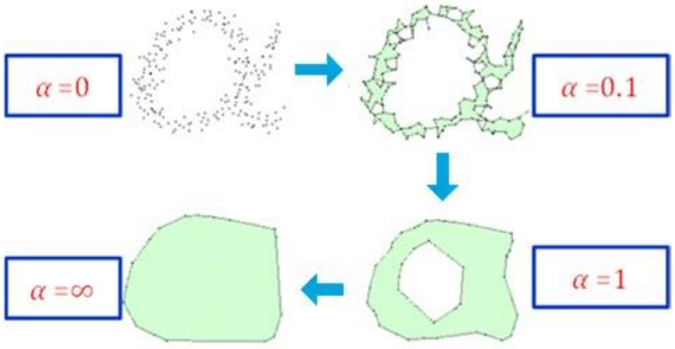


Figure 4.5. Boundary evolution of α-hull

Subsequently, the coordinate boundary of the 3D point cloud is determined as (x_{min}, x_{max}) and (y_{min}, y_{max}). The k-th point of the contour (row_k, col_k) is determined as the following equations:

$$\begin{aligned} m_c &= n_c = \sqrt{(x_{\max}-x_{\min})^2+(y_{\max}-y_{\min})^2} / r_{es} \\ row_k &= \frac{x_k-\dfrac{x_{\max}+x_{\min}}{2}}{r_{es}}+\frac{m_c}{2} \\ col_k &= \frac{y_k-\dfrac{y_{\max}+y_{\min}}{2}}{r_{es}}+\frac{n_c}{2} \end{aligned} \tag{4-2}$$

where m_c and n_c are the width and height of image representing the extracted contours.

4.5. Landing Zone Localization

The image of extracted contours is to be matched with the offline 2D map to locate the landing zone. The input data are binary images with not enough features for utilizing feature-based matching algorithms (e.g., SIFT and SURF). Template matching can be used to match the images with not enough features. Nonetheless, this method is not rotation invariant. The prerequisite that two input images have the same orientation is not available in our problem. So a kind of measurement of the image orientation needs to be defined to align the two input images. Image moments (a kind of weighted average of the image pixels' intensities) can be utilized to determine the orientation of images. The second order central moments represent the intensity distribution of an image (Liao and Pawlak 1996). Then a new algorithm called moment-based template matching is proposed. The image of extracted contours is defined as template I_c, in which the value of the pixels represented building boundaries is 1. The offline map with preselected landing location is defined as reference I_r. The outline of the algorithm is as follows (refer to Figure 4.6 for illustration):

(1) The minimal bounding circle of the image of extracted contours I_c is determined with a centre $\boldsymbol{C}$ and a radius r, and I_c is trimmed with the minimal bounding circle.

(2) The main orientation $\boldsymbol{\theta_c}$ of I_c is determined, and I_c is rotated along the main orientation, resulting a new image called template I_t.

(3) The similarity is measured between the template I_t and the reference I_r by a sliding window. At each window position (i, j), the reference I_r is trimmed by a circle with a radius r, generating an image I_{ij}.

(4) The main orientation $\boldsymbol{\theta_{ij}}$ of the trimmed image I_{ij} is determined.

(5) The correlation between the template I_t and the trimmed image I_{ij} is calculated, the correlation value is determined as γ_{ij}

(6) All the correlation values make up a correlation matrix γ. The trimmed image on the position (p_i, p_j) with the peak value of γ is extracted as the matched part.

The flowchart of the algorithm is shown in Figure 4.6. The details of these processing steps are given in the following sections.

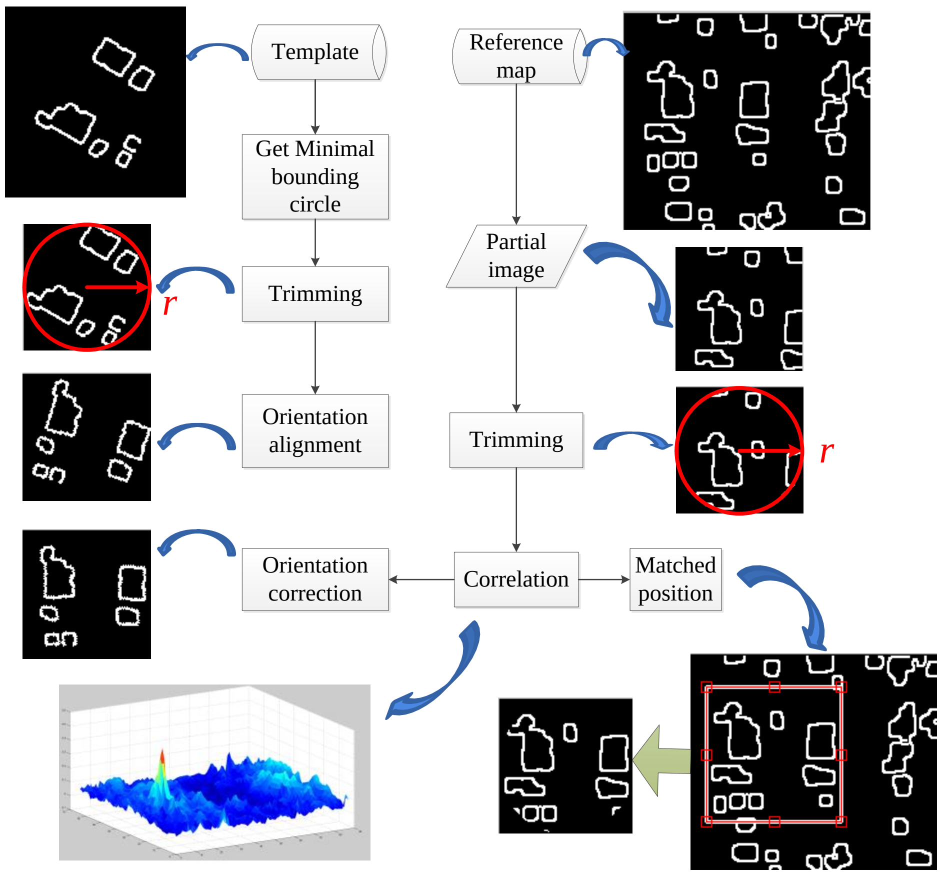


Figure 4.6 Flowchart of moment-based template matching

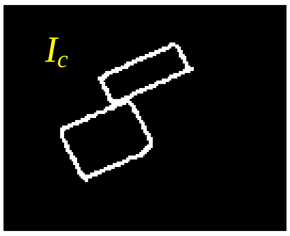
4.5.1. Template trimming

The image of extracted contours I_c is a binary image, in which the value of the pixels representing contours of buildings is 1. I_c is to be trimmed to generate a template I_t. The minimal bounding circle $\boldsymbol{C_b}$ surrounding all the pixels with value of 1 (defined as effective pixel) is to be determined first. This is a typical minimal enclosing circle problem, which is defined as Equation (4-3) as:

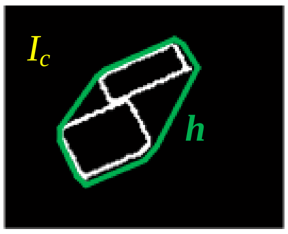
$$\begin{aligned} & \min_{(r,x,y)} r \\ & s.t.[(x-a_k)^2+(y-b_k)^2]^{1/2} \le r (k=1,2,...n) \\ & x,y,r \in \mathbf{Z}^+ \end{aligned} \tag{4-3}$$

In Equation (4-3), point (x, y) is the centre of $\boldsymbol{C_b}$, and r is the radius, n is the number of effective pixels, (a_k, b_k) is the column and row number of the kth effective pixel. The algorithm is as follows:

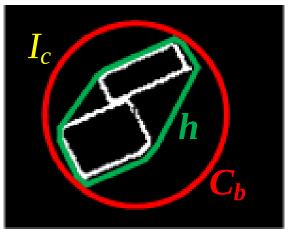
1) The convex hull h (e.g., the green hull in Figure 4.7b) of all the effective pixels is determined by the divide and conquer algorithm (Watt and Watt 2000).
2) The optimal circle is determined as the bounding circle of the convex hull of all the effective pixels (Elzinga and Hearn 1972) (e.g., the red circle in Figure 4.7c).
3) I_c is to be trimmed by the circle $\boldsymbol{C_b}$. The pixels out of the range (x-$r/2 \leq a_k \leq x+r/2$, y-$r/2 \leq b_k \leq y+r/2$) are removed (see Figure 4.7d), generating a template I_t (see Figure 4.7e).



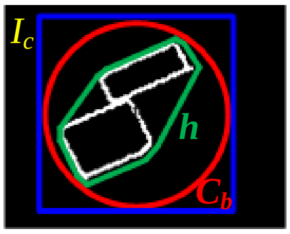
(a) The image of extracted contours



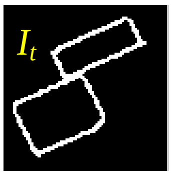
(b) Convex hull of effective pixels



(c) Minimal enclosing circle of effective pixels



(d) Trimming



(**e**) Resulted template

Figure 4.7 Template trimming

4.5.2. Main orientaiton determination

Image moment has been utilized in image analysis over the past few decades, which is defined as:

$$m_{pq} = \int\int x^p y^q f(x,y)dxdy \tag{4-4}$$

where $p, q = 1, 2, \ldots, \infty$.

The second order moment (inertia moment) depicts the mass distribution property of image (Liao and Pawlak 1996). The angle of the principal axis θ_p ($-45° \leq \theta_p \leq 45°$) (see Figure 4.8) nearest to the x axis can be determined as follows:

$$\begin{aligned} a &= m_{20}/m_{00} - m_{10}^2/m_{00}^2 \\ b &= m_{10}/m_{00} - m_{10} \times m_{01}/m_{00}^2 \\ c &= m_{20}/m_{00} - m_{01}^2/m_{00}^2 \\ \theta &= \tan^{-1}\left(2b/(a-c)\right)/2 \end{aligned} \tag{4-5}$$

where m_{00}, m_{01}, m_{10}, m_{11}, are the moments of an image. The main orientation of the template can be determined as $\theta_{p,t}$ (see Figure 4.9).

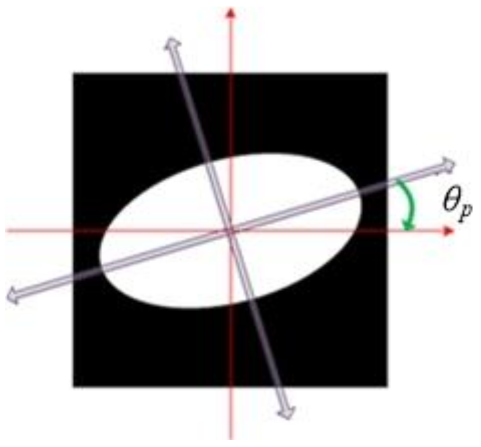
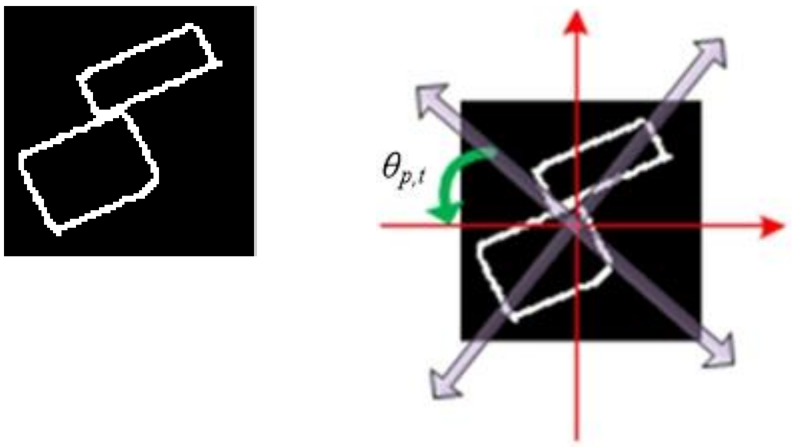


Figure 4.8 The principal axis of an image

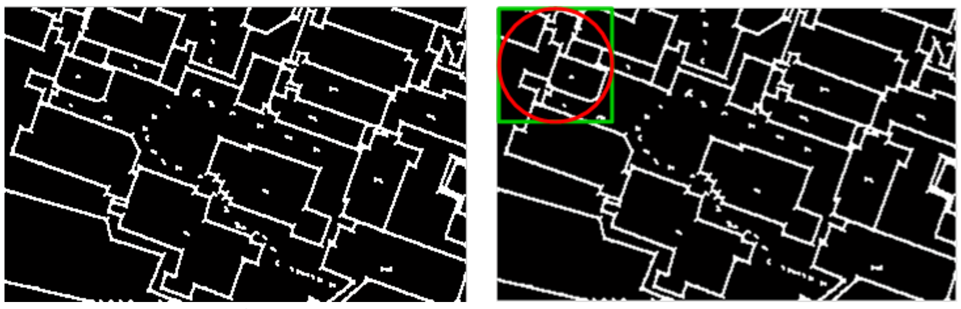


(a) Template (b) Main orientation for template

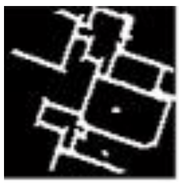
Figure 4.9 Main orientation determination

4.5.3. Reference trimming

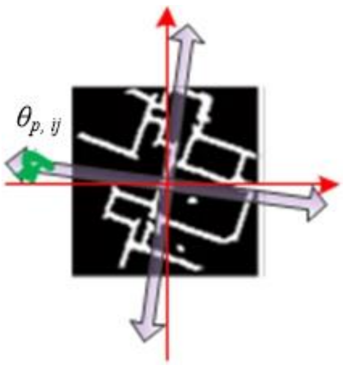
The template I_t (Figure 4.9a) is compared to the reference I_r (see Figure 4.10a) using a sliding window (see Figure 4.10b). At each window position (i, j), the pixels of the reference I_r within the range $(i\text{-}r/2 \leq a_k \leq i\text{+}r/2, j\text{-}r/2 \leq b_k \leq j\text{+}r/2)$ are extracted, generating a new image I_{ij} (e.g., the image within the green box in Figure 4.10b). Then, the pixels in I_{ij} out of the circle with radius r and centre (i, j) (see the red circle in Figure 4.10b) is set as 0. The main orientation of I_{ij} is determined as $\theta_{p,ij}$ (see Figure 4.10d).



(a) Reference (b) Extraction and trimming



(c) Trimmed part



(d) Main orientation determination

Figure 4.10 Main orientation determination and template rotation

4.5.4. Similarity measure

At each window position (i, j), after trimming of the reference, the similarity between I_t (Figure 4.11a) and I_{ij} (Figure 4.11b) is to be measured. There are different techniques to measure the similarities, including Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), Maximum Absolute Difference (MaxAD), and so on. Here, the normalized cross correlation (NCC) of two images γ_I is utilized to measure the similarity (Lewis 1995), which is defined as follows:

$$\begin{aligned}
\overline{T} &= \frac{\sum_{a=1}^{2r+1}\sum_{b=1}^{2r+1} T_{ab}}{(2r+1)^2} \\
\overline{N} &= \frac{\sum_{a=1}^{2r+1}\sum_{b=1}^{2r+1} N_{ab}}{(2r+1)^2} \\
\gamma_I &= \frac{\sum_{a=1}^{2r+1}\sum_{b=1}^{2r+1}\left(T_{ab}-\overline{T}\right)\left(N_{ab}-\overline{N}\right)}{\sqrt{\left(\sum_{a=1}^{2r+1}\sum_{b=1}^{2r+1}\left(T_{ab}-\overline{T}\right)^2\right)\left(\sum_{a=1}^{2r+1}\sum_{b=1}^{2r+1}\left(N_{ab}-\overline{N}\right)^2\right)}}
\end{aligned} \tag{4-6}$$

where T_{ab} and N_{ab} are the value of the pixels at position (a, b) of I_t and I_{ij}, $\overline{T}$ and $\overline{N}$ are the average values of the pixels in I_t and I_{ij}. Finally, the template is rotated to align the two images both at main orientation (see Figure 4.11a and Figure 4.11b). As $\theta_{p,t}$ and $\theta_{p,ij}$ are within the range (-45°, 45°), there are four possible matched orientations between the two images (see Figure 4.11c), i.e., ($\theta_{p,t}$ - $\theta_{p,ij}$), ($\theta_{p,t}$ - $\theta_{p,ij}$ + 90°), ($\theta_{p,t}$ - $\theta_{p,ij}$ + 180°), ($\theta_{p,t}$ - $\theta_{p,ij}$ -90°). The orientation with the largest similarity is selected, and the NCC value γ_{ij} is recorded. At each window position (i, j), one NCC value γ_{ij} is determined, making up a NCC matrix (see Figure 4.12a). Then, the partial image within with peak NCC is extracted as the matched part (see Figure 4.12c). Finally, the template is rotated to align with the extracted part (see Figure 4.12d).

In order to assess the matching result, two criteria of successful matching are proposed:

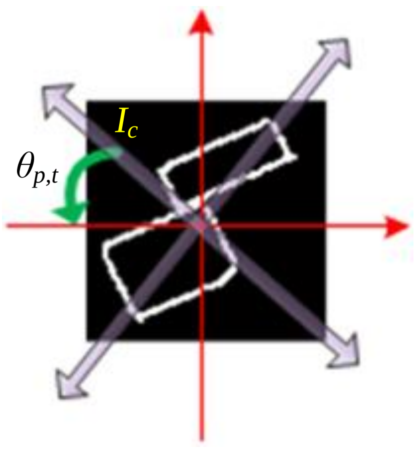
(1) r_{ap}: The difference between peak value and average value in the NCC matrix.

(2) r_{fspd}: The difference between peak value and second peak value.

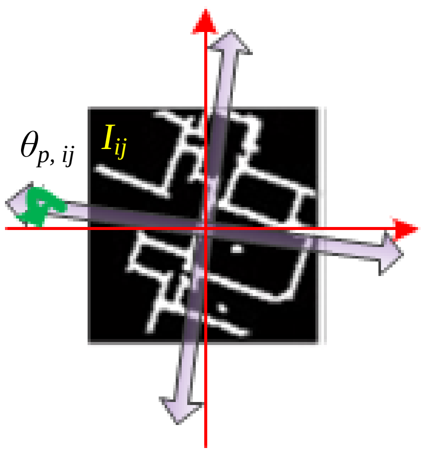
The two criteria are defined as follows:

$$\begin{aligned} r_{pa} &= \frac{\gamma_p - \frac{1}{M \times N}\sum_{i=1,j=1}^{M,N}\gamma_{i,j}}{\gamma_p} \times 100\% \\ r_{fspd} &= \frac{\gamma_p - \max\left\{\gamma_k^l, k=1,2,\ldots,K\right\}}{\gamma_p} \times 100\% \end{aligned} \tag{4-7}$$

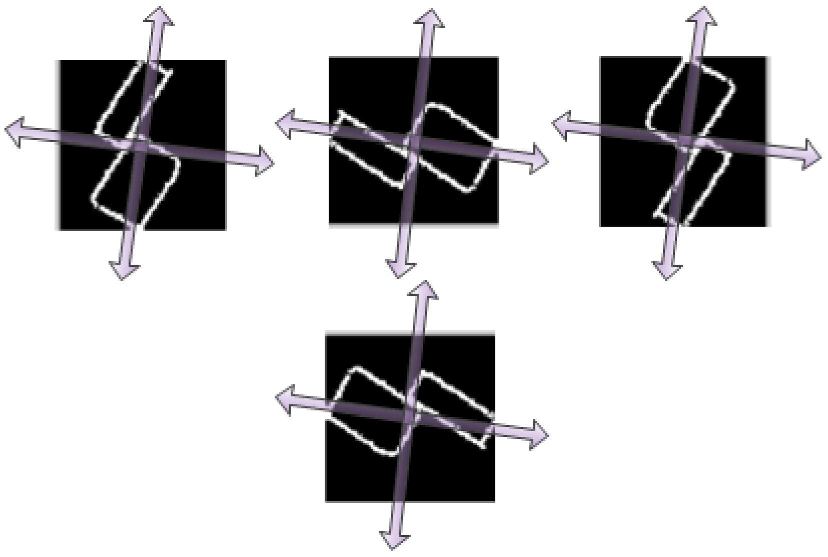
where $\gamma_{i,j}$ is the NCC value at window position (i, j), γ_p is the peak value, and γ_k^l is the k-th local maximum. If the matching result meets all the three criteria, the matching is considered as successful.



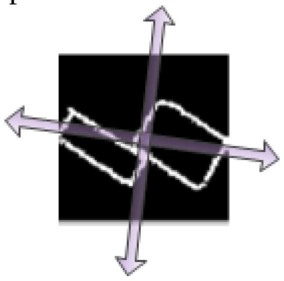
(a) Main orientation of template



(b) Main orientation of trimmed reference

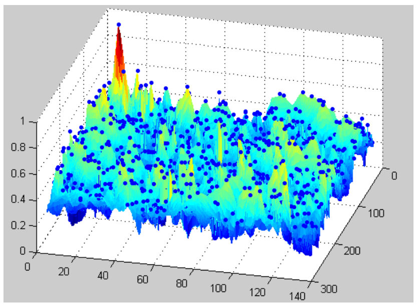


(c) Rotated template with four different orientations

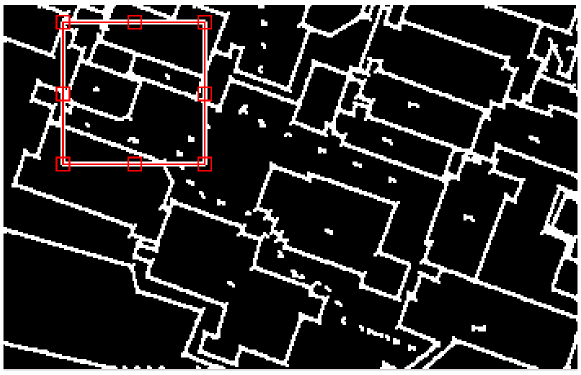


(d) Selected orientation with the largest similarity

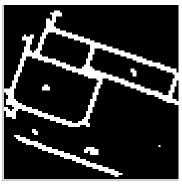
Figure 4.11 Similarity measure and orientation selection



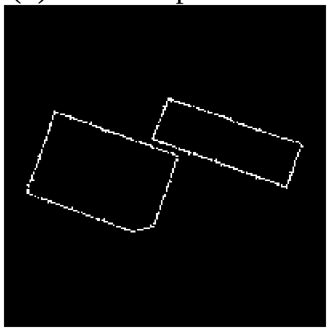
(a) NCC matrix



(b) Matched position



(c) Matched part

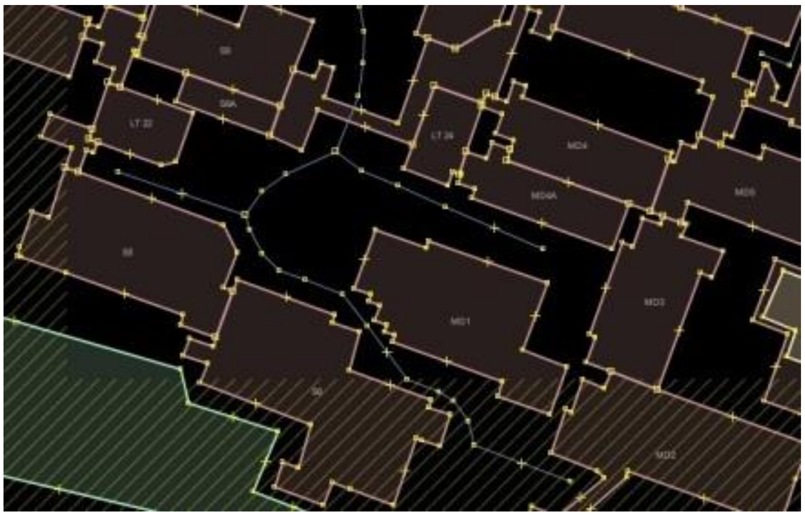


(d) Rotated template according to the map

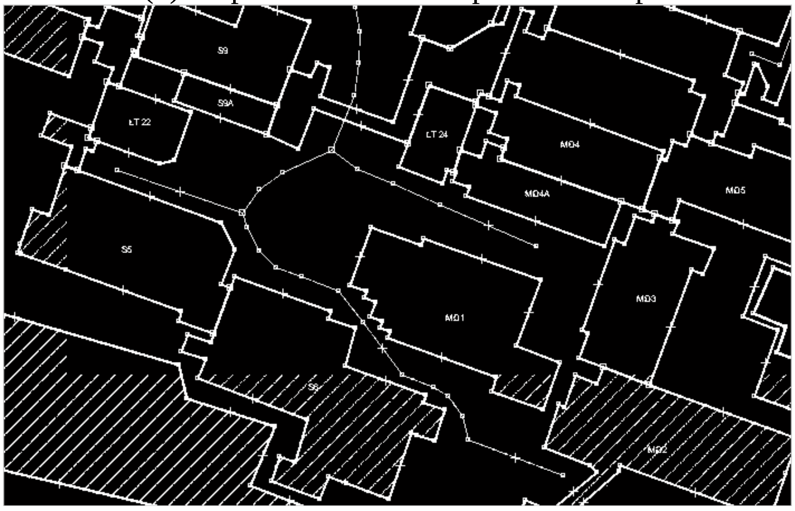
Figure 4.12 Matching result

4.5.5. Testing of the matching method

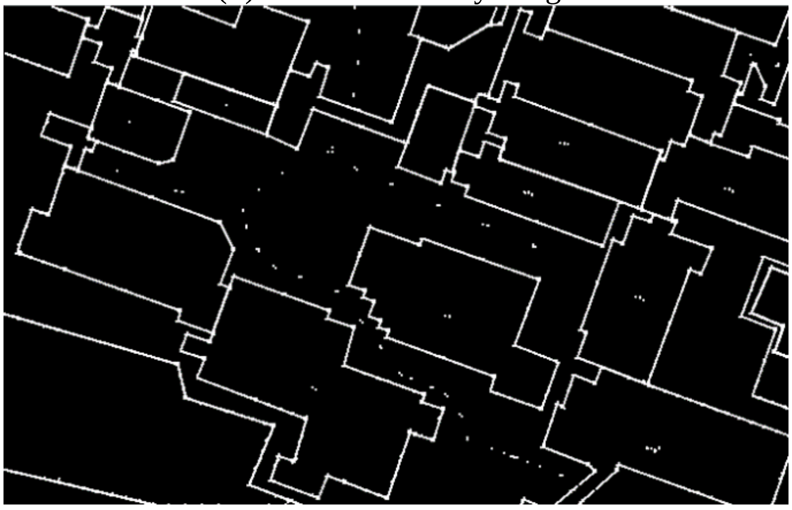
An experiment has been carried out to test the algorithm of moment-based template matching. In the experiment, the offline map is collected from the OpenStreetMap. As shown in Figure 4.13a, the map is part of the campus in National University of Singapore. The map is firstly processed by two steps for matching. The first step is to convert to a binary image with 224×352 pixels (see Figure 4.13b). Then, open morphology is applied to the binary image, generating an offline map (see Figure 4.13c). The contour image to be matched with this map is shown in Figure 4.14 and the desired result for matching is shown in Figure 4.15.



(**a**) Map collected from OpenStreetMap



(**b**) Convert to binary image



(**c**) Offline map generated after open morphology

Figure 4.13 Pre-processing of the offline map

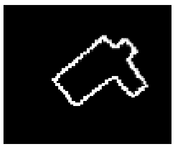


Figure 4.14 Contour image for matching

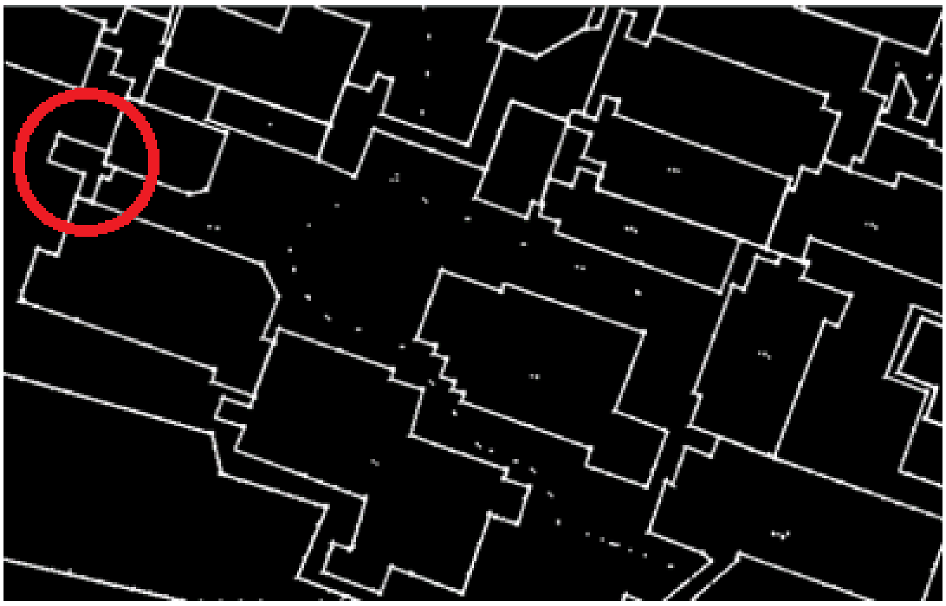
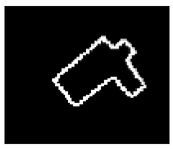
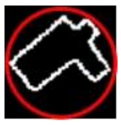


Figure 4.15 Desired result for matching

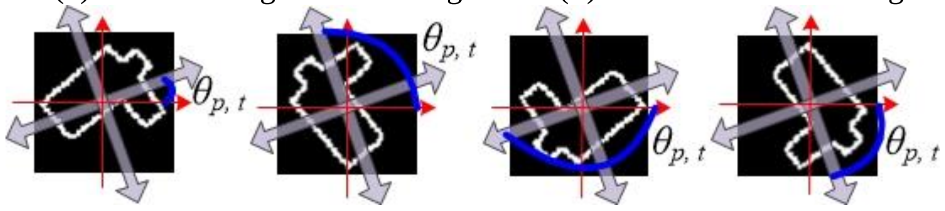
The matching procedure and the intermediate results are shown in Figure 4.16. The input contour image for matching is a binary image with 101×84 pixels (see Figure 4.16a). After trimming, the image has 57×57 pixels (see Figure 4.16b), and its main orientation is determined as $\theta_{p,t}$ =16.9°, 106.9°, -163.1°, -73.1° (see Figure 4.16c). The correlation matrix is shown in Figure 4.16d, and the matched position is located as (15, 47).



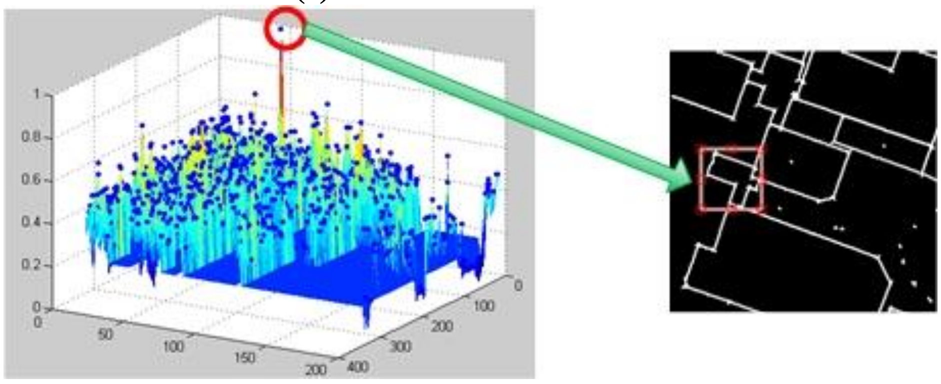
(**a**) Contour image for matching



(**b**) Contour after trimming



(**c**) Four main orientations



(**d**) Correlation matrix and selected matching position

Figure 4.16 Test result for moment based template matching algorithm

4.5.6. Determination of Successful Matching

In order to determine the two proposed criteria of successful matching, a numerical experiment is carried out. The offline map is shown in Figure 4.17. The template to be matched is shown in Figure 4.18b, which is applied scale error. The scale of the template is varied within [0.8, 1.2] at an interval of 0.008, generating 50 samples (see Figure 4.18). The matching algorithm is tested with the 50 samples, and the results for the two criteria are shown in Figure 4.19. From the result, the matching criteria is set conservatively as r_{pa} = 70%, and r_{fspd}= 20%, respectively.

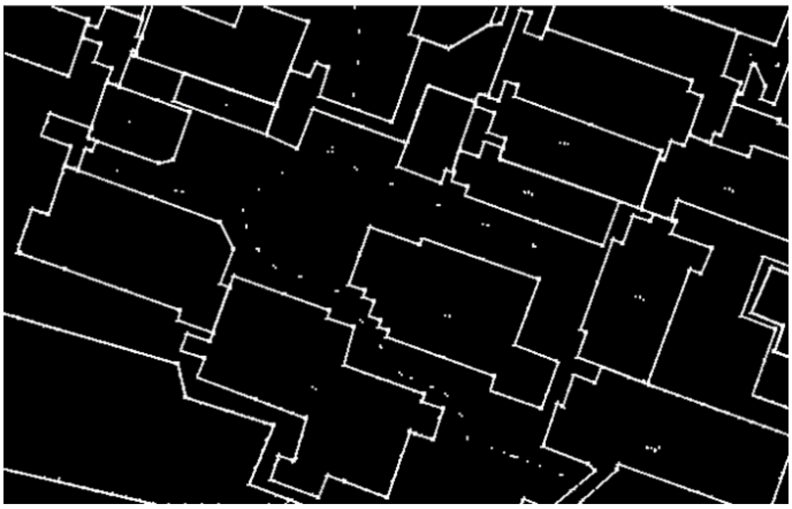


Figure 4.17 Offline map for matching criteria test

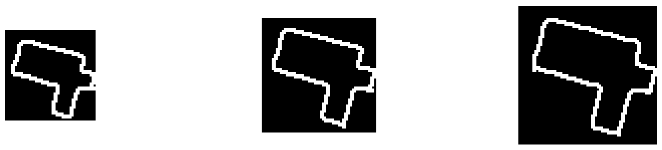

(**a**) Scale=0.8 (**b**) Template (**c**) Scale=1.2

Figure 4.18 The template applied scale error

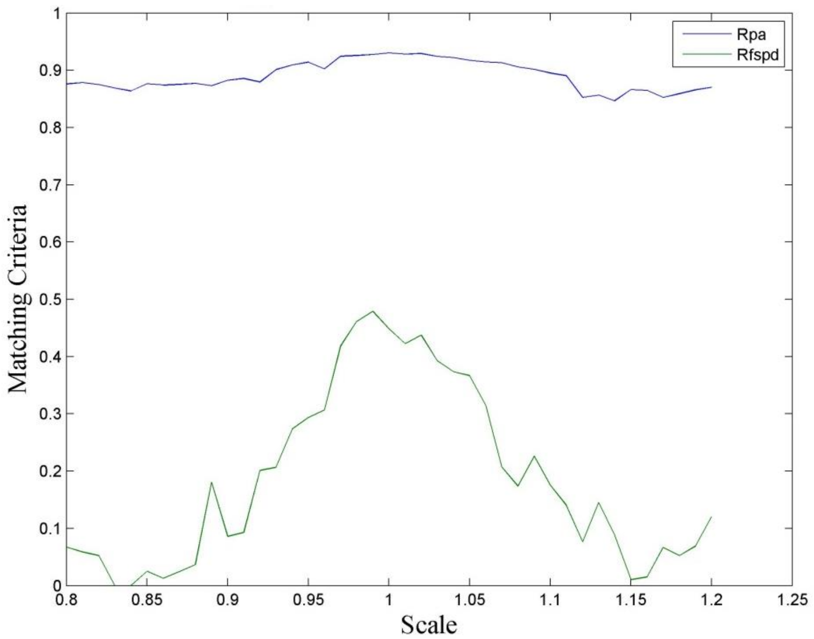


Figure 4.19 The evolution of matching result with scale error

4.6. Landing Zone Evaluation

In the offline map, the landing zone is preselected according to the size of UAV, safety distance to the edge of the roof, etc (see Figure 4.20b). The pixel position of the *v*-th corner of the preselected landing zone in the offline map I_r is ($row_{v,r}$, $col_{v,r}$). After matching between the reference I_r (offline map) and the template I_t (see Figure 4.20a), the matched positon and orientation of the template within the map is determined as (row_m, col_m) and $\theta_{p,m}$.

The next step is to locate the landing zone in the template I_t (see Figure 4.20c). The corresponding pixel positon of the *v*-th corner of the preselected landing zone in the template ($row_{v,t}$, $col_{v,t}$) is determined as follows:

$$\begin{aligned} row_{v,t} &= (row_{v,r} - row_m - \frac{m_c}{2})\cos\theta_m - (col_{v,r} - col_m - \frac{n_c}{2})\sin\theta_m + \frac{m_c}{2} \\ col_{v,t} &= (row_{v,r} - row_m - \frac{m_c}{2})\sin\theta_m + (col_{v,r} - col_m - \frac{n_c}{2})\cos\theta_m + \frac{n_c}{2} \end{aligned} \tag{4-8}$$

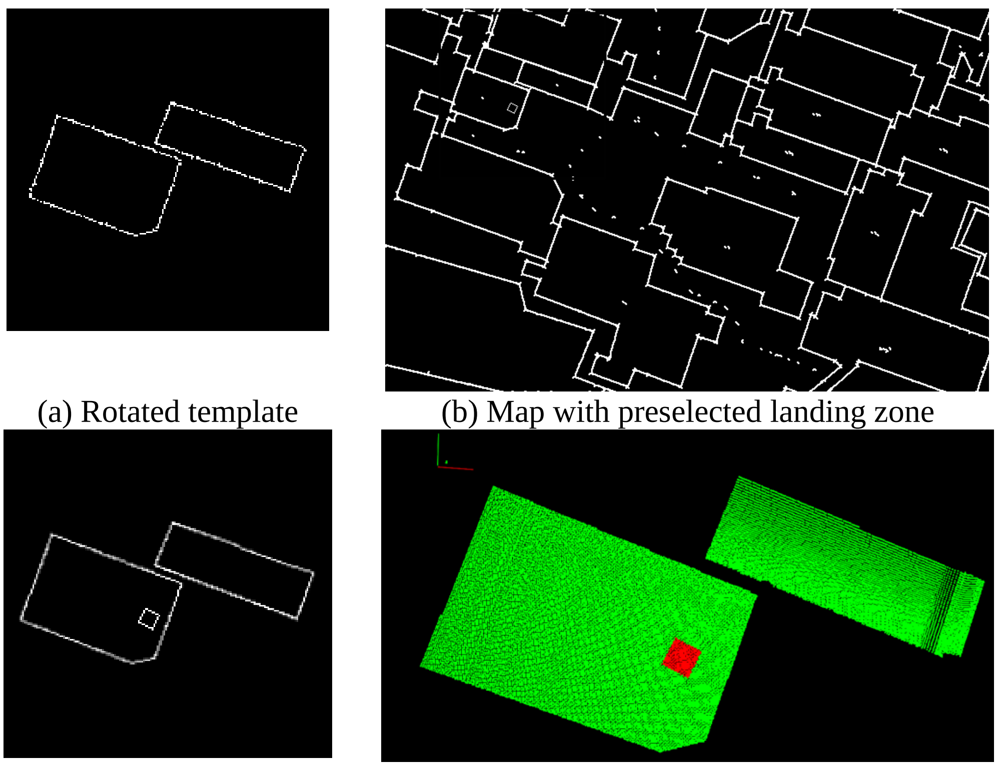
where, (m_c, n_c) is the position of the centre pixel of the image. Then, the *v*-th corner of the landing zone in the point cloud is further identified in the point cloud (see Figure 4.20d) as follows:

$$\begin{aligned} x_{v,t} &= \frac{x_{\max} + x_{\min}}{2} + (row_{v,t} - \frac{m_c}{2})r_{es} \\ y_{v,t} &= \frac{y_{\max} + y_{\min}}{2} - (col_{v,t} - \frac{n_c}{2})r_{es} \end{aligned} \tag{4-9}$$

Finally, the feasibility of the landing zone is evaluated. To evaluate the landing zone, the points within the zone are extracted. Then, a plane ($ax+by+cz+d=0$) is fitted to the points. Whether the landing zone is feasible for landing is evaluated by two criteria: inclination angle of the fitted plane λ_p and roughness σ_p (standard deviation of the distance from the points to the plane) as follows:

$$\lambda_p = \cos^{-1}\frac{|c|}{\sqrt{a^2+b^2+c^2}} < 20°$$
$$\sigma_p = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{|ax_i+by_i+cz_i+d|}{\sqrt{a^2+b^2+c^2}}\right)^2} < 20mm \tag{4-10}$$

Assume the image size of template and map are (m_1, n_1) and (m_2, n_2) respectively, the computation time is proportional to $m_1 \times n_1 \times (m_2 - m_1) \times (n_2 - n_1)$. As the image size of template is generally much smaller than the map, that is, $m_1 << m_2$, and $n_1 << n_2$, so the computation time is approximately proportional to the pixel number of the map. So the complexity of the algorithm is $O(n)$.


(a) Rotated template

(b) Map with preselected landing zone

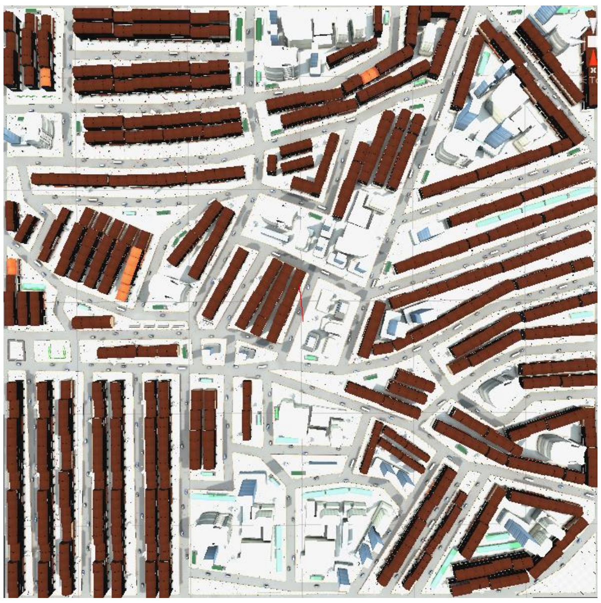
(c) Landing zone in template

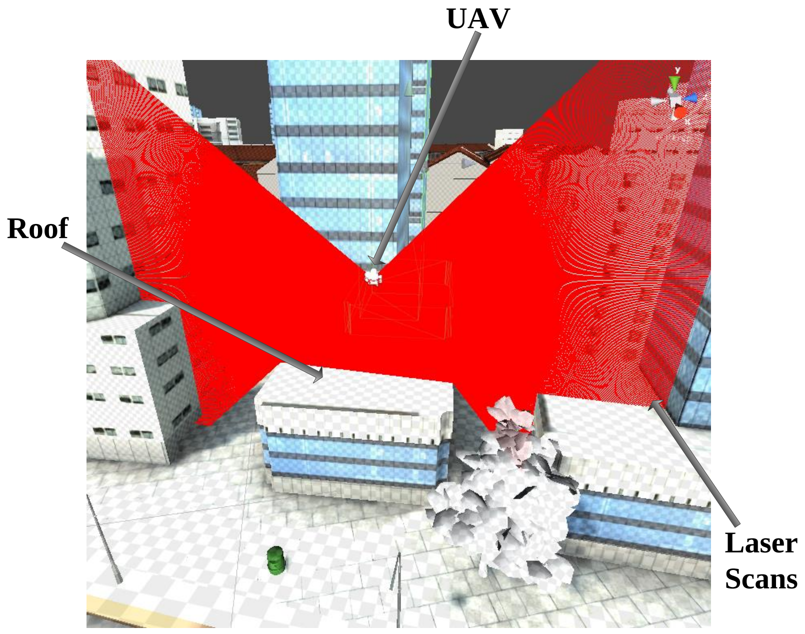
(d) Landing zone in the point cloud

Figure 4.20 Landing zone identification

4.7. Numerical Experiments

Two numerical experiments have been carried out to test the developed algorithm. The point cloud data in both experiments were collected using a simulation environment called Unity3D. In the environment, a city model of over 1000 buildings has been built, as shown in Figure 4.21. In the first experiment, there's one preselected landing location, it is to be determined whether it is feasible for landing by the algorithm. In the second experiment, there are multiple preselected landing locations. It is expected to determine the feasibility of all the landing locations and select the best one.

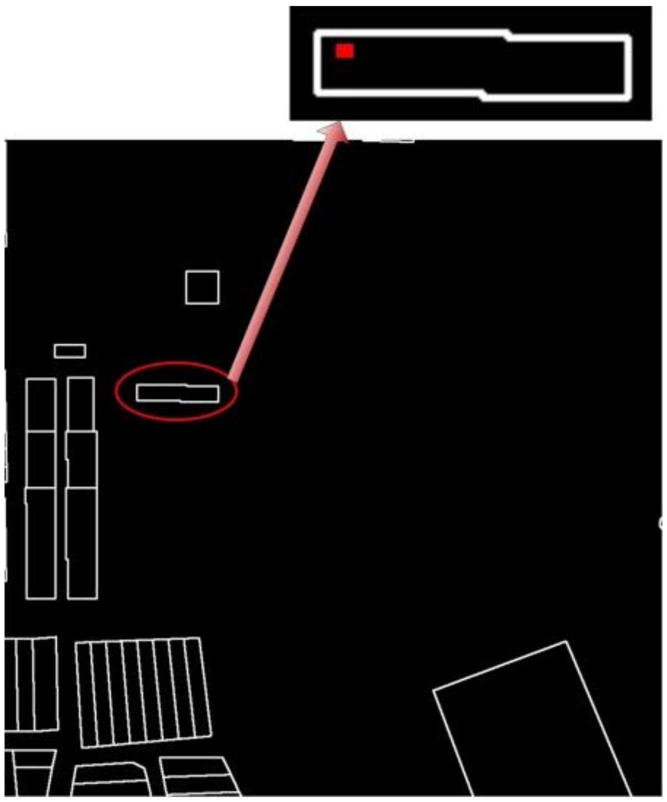

(a) The city model in Unity3D

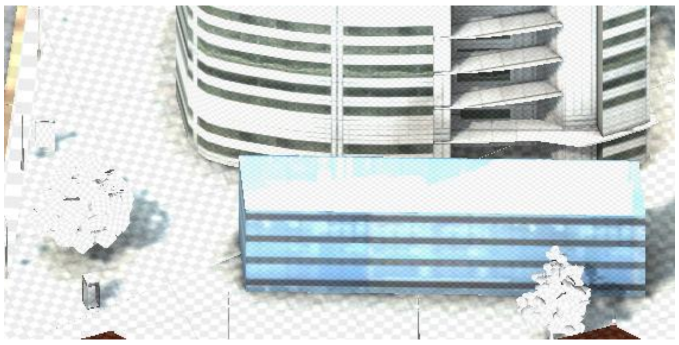


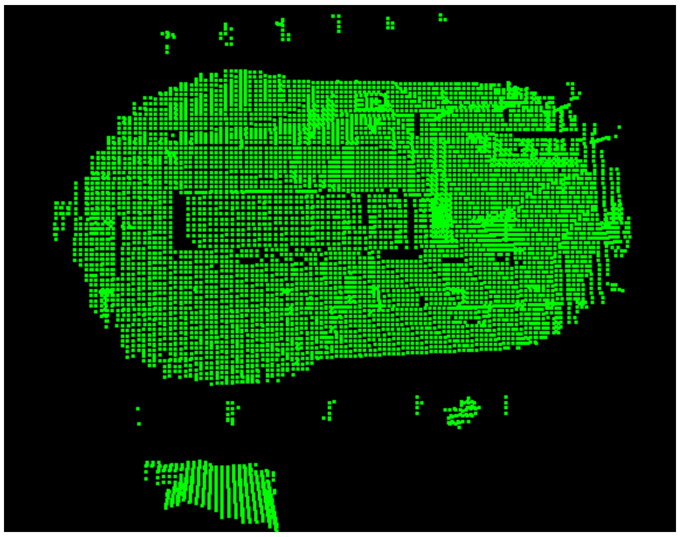
(b) Hovering and data collection

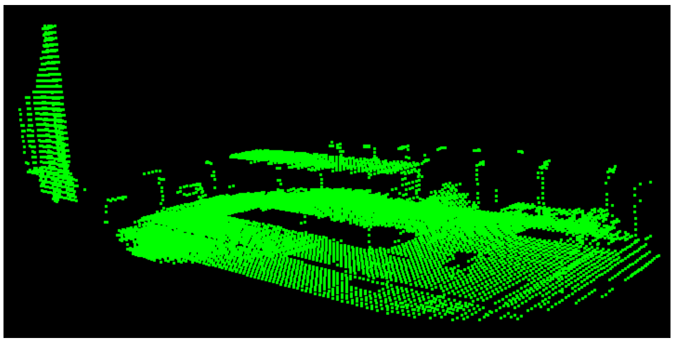
Figure 4.21 City model and data collection

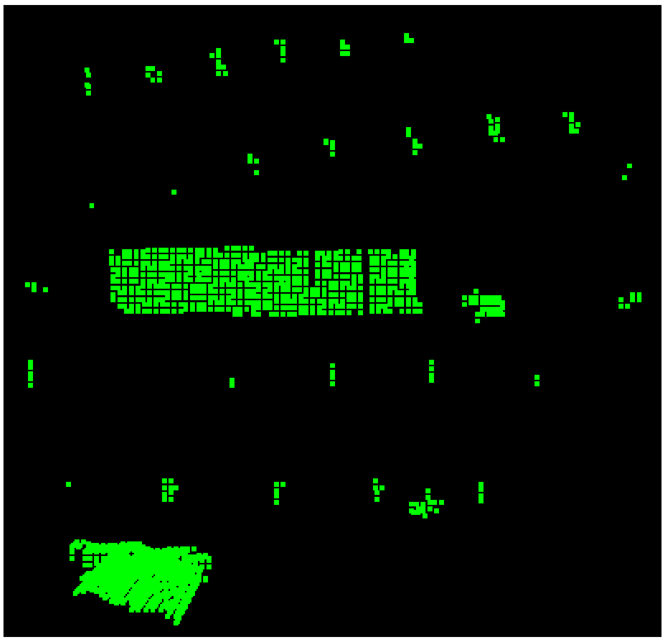
In the first experiment, one landing location is preselected at the roof corner of one building (see Figure 4.22a). The 3D model of the target area is shown in Figure 4.22b. The collected point cloud (see Figure 4.22c) consists of 8,725 points, the average interval of which is 0.55m. The total running time is 0.75s. In altitude filtering step (see Figure 4.22d), the points outside the range (5m, 20m) are filtered. Then, the normal of the remaining points are estimated, in which the neighborhood size of a point is set as 16. The points with slope larger than 45° are filtered out (see Figure 4.22e). After clustering, the clusters with less than 80 points are removed, only one cluster is remaining (see Figure 4.22f). In boundary extraction (see Figure 4.22g), the α_h is set as 2. Then, the contour is converted to an image with a resolution of 0.1m/pixel (see Figure 4.22h). After matching, the matched location of the contour in the offline map is at the pixel (221, 367), and the orientation displacement of the contour is 1° relative to the offline map (see Figure 4.22i). Finally, the feasibility of the landing zone is checked (Figure 4.22j). The inclination angle is 0.9° and the roughness is 3.9mm. So it is considered feasible for landing. The running time of each step is shown in Table 4.2.

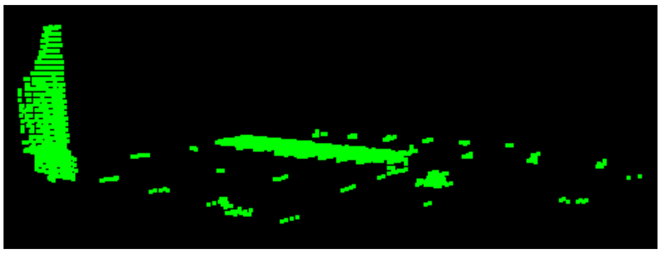

(a) 2D map with preselected landing location

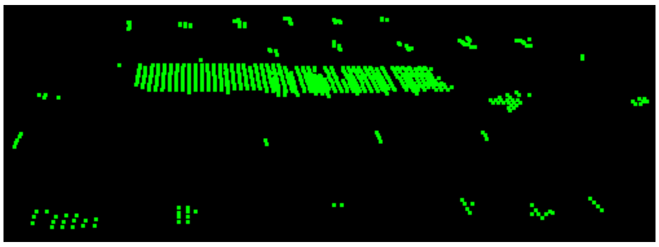

(b) 3D model

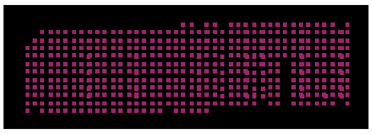


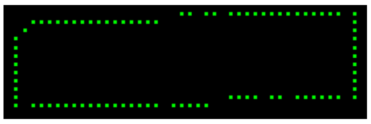

(c) Input point cloud

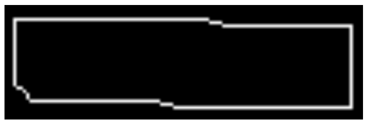


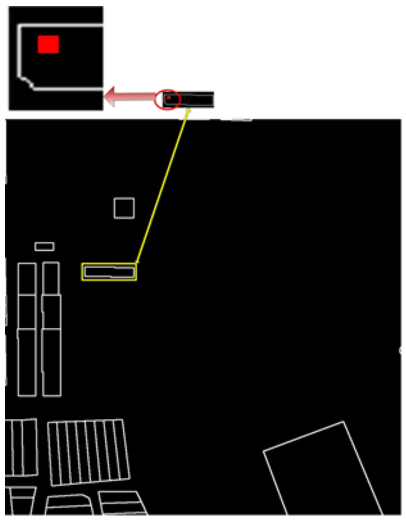

(d) Altitude filtering


(e) Slope filtering


(f) Clustering


(g) Roof boundary


(h) Roof contour


(i) Landing location in roof contour

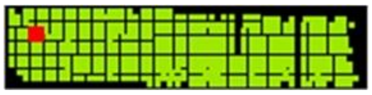

(j) Landing location in point cloud

Figure 4.22 Process of the first numerical experiment

In the second experiment (see Figure 4.23), three landing locations are preselected at the roof corners of three buildings (see Figure 4.23a). The 3D model of the target area is shown in Figure 4.23b. The collected point cloud (see Figure 4.23c) consists of 33,609 points, the average interval of which is 0.48m. The total running time is 1.19s. The parameters are set as the same in the first experiment, such as altitude range, neighbourhood size, and the slope threshold, etc. After matching, the matched location of the contour in the offline map is at the pixel (210, 553), and the orientation displacement of the contour is 0° relative to the offline map (see Figure 4.23i). Finally, the feasibility of all the three landing zones is checked (see Figure 4.23j), and the results are shown in Table 4.1. After comparison, the 1st landing zone is selected as the landing location for the UAV.

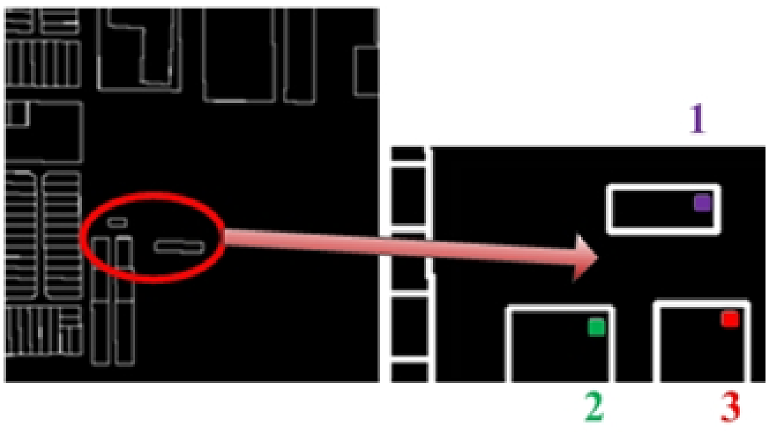
The two experiments are carried out on a computer with a processor of "Inter(R) Xero(R) CPU e3-1225 v3 @ 3.2GHz", memory of 8GB, 64-bit operating system, Ubuntu 12.04, libpcl 1.7, libopencv 2.4.13. The computation time for all the steps of the algorithm is shown in Table 4.2.

Table 4.1 Landing zone feasibility evaluation

Landing criteria	**Landing zone**		
	1*st*	**2*nd***	**3*rd***
Inclination angle λ_p /°	13.1	48.3	52.9
Roughness σ_p / mm	12.5	117	102
Feasibility	Feasible	Infeasible	Infeasible

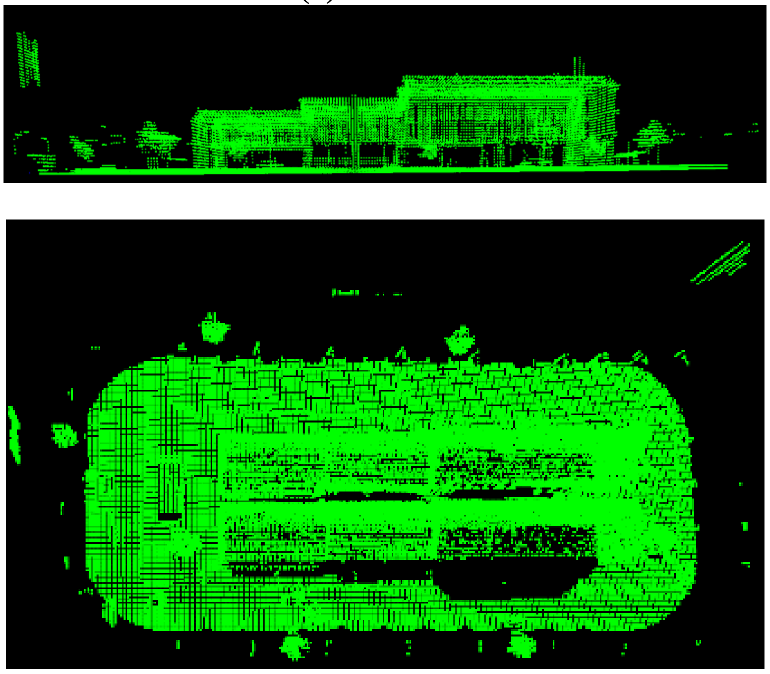
Table 4.2 Computation time for the two experiments

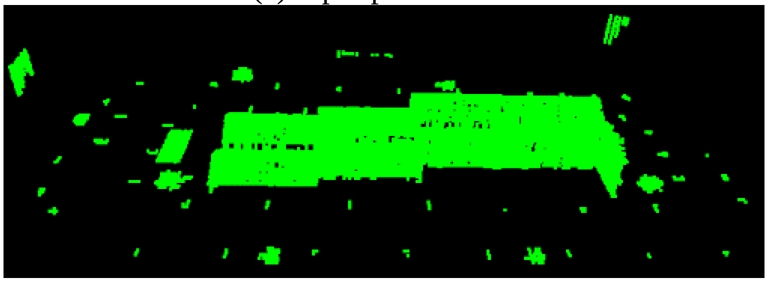
Experiment	1	2
Point cloud size	8725	33609
Read point cloud	0.031683 s	0.116011 s
Altitude filtering	0.003500 s	0.024929 s
Slope filtering	0.004619 s	0.043820 s
Clustering	0.001707 s	0.017379 s
Contour extraction	0.004561 s	0.044232 s
Contour matching	0.695662 s	0.741771 s
Evaluation	0.000116 s	0.000424 s
Total	0.741848 s	0.988566 s

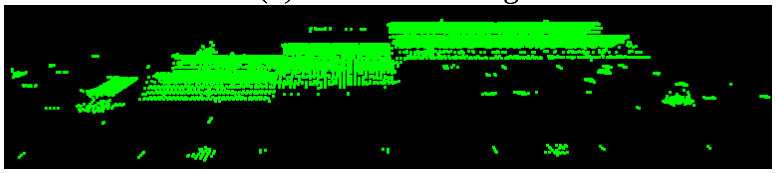


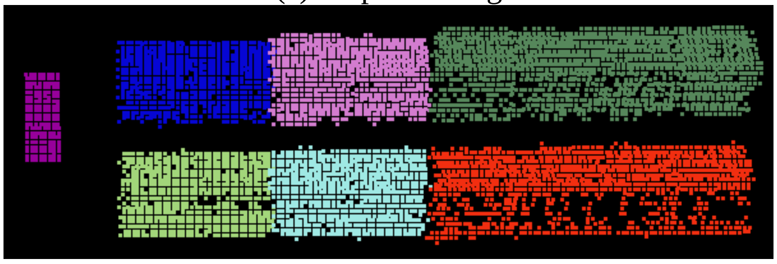
(a) 2D map with preselected landing location

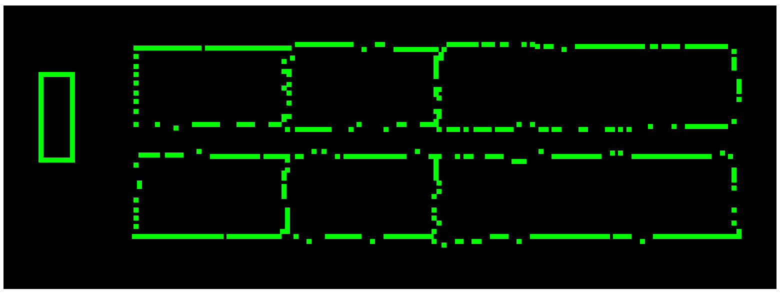
(b) 3D model

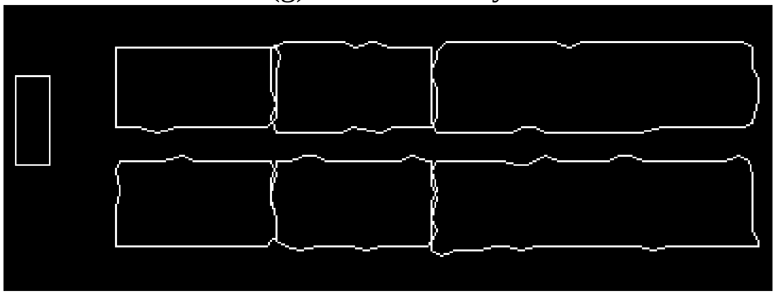

(c) Input point cloud

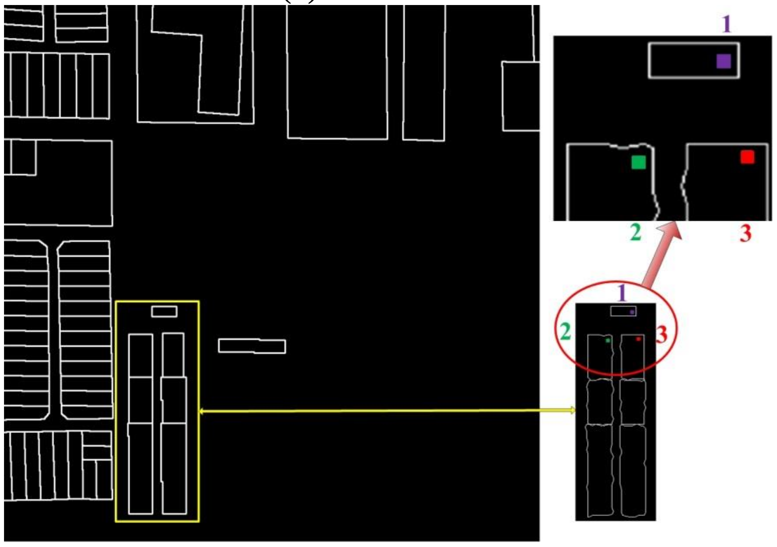

(d) Altitude filtering


(e) Slope filtering

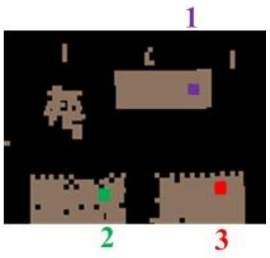

(f) Clustering


(g) Roof boundary


(h) Roof contour



(i) Landing location in roof contour



(j) Landing location in point cloud

Figure 4.23 Process of the second numerical experiment

4.8. Analyzation of the parameters in the experiment

All the parameters in the experiments are tunable. In the step of altitude range filtering, it is assumed that we have some knowledge of the building height range in the certain region. The value (5m, 20m) is set only for this experiment. The larger the range is chosen, the more points are included in the point cloud after the altitude range filtering, resulting more computation time, in addition, points not belonging to roofs may be included, such as ground, cars, and other objects. Too small range may cut off some points belonging to the roofs.

In the step of clustering, the threshold values n_{min} for Euclidean clustering is set as 80, which means the clusters with less than 80 points are not considered as possible roofs. If n_{min} is set too small, points not belonging roofs may be included in point cloud after the clustering. If n_{min} is set too large, possible roofs may be cut off.

In the step of slope filtering, for each point, a plane is fitted to its neighbors. The slope of the plane is checked. If the slope is larger than a certain threshold, the point is considered as not feasible. The threshold angle can be adjusted. In this experiment, 45° is selected because the plane steeper than 45° is considered as walls, the plane slope smaller than 45° is considered as roof. The larger the threshold angle, the more points are included in the point cloud after the slope filter. Too large threshold angle may bringing points belonging walls, while, too small threshold angle may cut off points belonging roofs.

In the step of boundary extraction, α_h controls the shape of the resulted boundary of the roof contour. Larger α_h results in more structured boundary and less number of contours. If α_h goes to infinity, the alpha hull becomes the convex hull, that is, only generate one contour.

In the step of landing zone evaluation, the two criteria α_p and σ_p are evaluated to prevent the UAV to roll over when it lands on the roof. The specific parameters are to be tuned based on different scenarios.

4.9. Summary

This chapter addresses the problem of online landing location identification and feasibility evaluation for rotary-wing UAV. In the selection procedure, the point cloud obtained by laser scanner is filtered, and the contour of the roof is extracted, which is then matched with the offline 2D map to locate the preselected landing point/zone. The main contribution is that the algorithms of point cloud filtering and landing location extraction can be implemented in real time, which makes it feasible for on-board running on UAV. The major difference between our algorithms with other landing location selection algorithms is that we match the online extracted roof contour with an offline map to locate the landing point/zone. The matching algorithm guarantees high accuracy of selected landing location, which is of great importance. The main limitation in the online landing location selection is that the point cloud filtering algorithm is based on some heuristics. For example, the height range, inclination angle and roughness of the feasible landing locations are utilized to do point cloud filtering.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The main objective of this thesis is to further the research in UAV mission planning in city surveillance. In this chapter, the research will be summarized for the thesis and possible recommendations for future work will be presented.

5.1. Conclusions

The specific issues addressed in this thesis are: (1) coverage path planning for fixed-wing UAV city surveillance in hazard environment; (2) offline perching location selection by a rotary-wing UAV for surveillance on a target; (3) online landing location identification and feasibility evaluation by a rotary-wing UAV in a GPS-denied environment.

- **Coverage path planning for continuous city surveillance with fixed-wing UAVs in hazardous environment**

The problem of coverage path planning for continuous city surveillance fixed-wing UAVs in hazardous environment has been addressed. In this problem, a realistic hazardous urban environment model was built, including terrain and buildings. The camera on-board the UAV is modelled with the pan-tilt ability. The flying paths of the UAVs to cover the whole urban area are planned with the proposed algorithms. A two stage approach is applied to achieve a sub-optimal solution. In the first stage, one camera position and attitude is determined for each surface point in the urban area, making up a vantage set. The best camera attitude is determined by a PSO algorithm. In the second stage, the optimal subset is selected from the vantage set to cover the whole surface area and flying paths are fitted for UAVs. A GA-based algorithm is proposed to select the optimal subset. The flying path is fitted with B-spline curve, and realistic constraints are considered in path fitting, such as turning radius, climbing/diving angle. Two kinds of fitness functions are considered for scenarios with and without hazard existence.

- **Offline perching location selection by a rotary-wing UAV for surveillance on a target**

A problem of offline perching location selection by a rotary-wing UAV for surveillance on a target has been solved. Realistic constraints are considered in perching location selection, including both geometric constraints and mission constraints. Geometric constraints includs camera range, roof area, and roof slope. The mission constraints is to keep a line of sight from the perching location to the target to be viewed. The feasible perching locations can be selected by the proposed algorithms.

- **Online landing location selection by a rotary-wing UAV in a GPS-denied environment**

The problem of landing location identification and feasibility evaluation for rotary-wing UAV is addressed. The 3D point cloud collected by a 2D LiDAR on-board the UAV is filtered first, then the roof contours are extracted and matched with the offline map to identify the presected landing zone. Finally, the feasibility for landing is evaluated. The point coud is converted to an image to be registrated to the offline map. A new rotation-invariant template matching method is proposed to match images without enough features. Second order image moment is utilized to determine the main orientation of the image.

5.2. Recommendation for Future Work

Limitations cannot be avoided in the thesis, which may lead to some possible recommendations for future work.

First, for coverage path planning of fixed-wing UAVs in continuous city surveillance mission, the recommendations are as follows: (1) Model more types of hazard, such as missiles, cannon, moving radar vehicle, etc. (2) Directly use GA method to find optimal vantage set, without a pre-selection of view directions. (3) Consider more realistic constraints, such as flying duration, different depots for UAVs, different UAV capabilities.

Second, for offline perching location selection by a rotary-wing UAV for surveillance on a target, the recommendations are: (1) Model more complex urban environment, including trees, uneven terrain, irregular objects, etc. (2) Consider more complex models of the target to be viewed.

Finally, for online landing location selection by a rotary-wing UAV in a GPS-denied environment, the possible directions are: (1) The matching algorithm needs to be improved to be robust to scale of image; (2) Apply pattern recognition to determine the load bearing ability of the landing locations.(3) Improve the algorithms of point cloud filtering to remove the heuristics.

BIBLIOGRAPHY

Ahmadzadeh, A., J. Keller, G. Pappas, A. Jadbabaie and V. Kumar (2008). An Optimization-Based Approach to Time-Critical Cooperative Surveillance and Coverage with UAVs. Experimental Robotics, Springer.

Akkiraju, N., H. Edelsbrunner, M. Facello, P. Fu, E. Mücke and C. Varela (1995). Alpha shapes: definition and software. Proceedings of the 1st International Computational Geometry Software Workshop.

Alharthy, A. and J. Bethel (2002). "Heuristic filtering and 3D feature extraction from LIDAR data." International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences **34**(3/A): 29-34.

Bay, H., T. Tuytelaars and L. Van Gool (2006). Surf: Speeded up robust features. Computer vision–ECCV 2006, Springer**:** 404-417.

Belton, D. and D. D. Lichti (2006). "Classification and segmentation of terrestrial laser scanner point clouds using local variance information." Iaprs, Xxxvi **5**: 44-49.

Berger, J., J. Happe, C. Gagné and M. Lau (2009). Co-evolutionary information gathering for a cooperative unmanned aerial vehicle team. Information Fusion, 2009. FUSION'09. 12th International Conference on, IEEE.

Berglund, T., H. Jonsson and I. Söderkvist (2003). An obstacle-avoiding minimum variation b-spline problem. Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on, IEEE.

Bertuccelli, L., M. Alighanbari and J. How (2004). Robust planning for coupled cooperative UAV missions. Decision and Control, 2004. CDC. 43rd IEEE Conference on, IEEE.

Briechle, K. and U. D. Hanebeck (2001). Template matching using fast normalized cross correlation. Aerospace/Defense Sensing, Simulation, and Controls, International Society for Optics and Photonics.

Chen, J. and D. M. Dawson (2006). Uav tracking with a monocular camera. Decision and Control, 2006 45th IEEE Conference on, IEEE.

Cheng, P., J. Keller and V. Kumar (2008). Time-optimal UAV trajectory planning for 3D urban structure coverage. Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, IEEE.

Cho, A., J. Kim, S. Lee, S. Choi, B. Lee, B. Kim, N. Park, D. Kim and C. Kee (2007). Fully automatic taxiing, takeoff and landing of a UAV using a single-antenna GPS receiver only. Control, Automation and Systems, 2007. ICCAS'07. International Conference on, IEEE.

De Carvalho, R. N., H. Vidal, P. Vieira and M. Ribeiro (1997). Complete coverage path planning and guidance for cleaning robots. Industrial Electronics, 1997. ISIE'97., Proceedings of the IEEE International Symposium on, IEEE.

Ding, X. C., A. R. Rahmani and M. Egerstedt (2010). "Multi-UAV convoy protection: an optimal approach to path planning and coordination." Robotics, IEEE Transactions on **26**(2): 256-268.

Dogo, H., A. Bower, S. Seong, J. E. Peters, C. G. Pernin and A. L. Martin (2011). Unmanned Aircraft Systems for Logistics Applications, Rand Arroyo Center Santamonica Monica CA.

Elzinga, J. and D. W. Hearn (1972). "Geometrical solutions for some minimax location problems." Transportation Science **6**(4): 379-394.

Flint, M., M. Polycarpou and E. Fernandez-Gaucherand (2002). Cooperative control for multiple autonomous UAV's searching for targets. Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, IEEE.

Forster, C., M. Faessler, F. Fontana, M. Werlberger and D. Scaramuzza (2015). Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles. Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE.

Fumagalli, M., R. Naldi, A. Macchelli, R. Carloni, S. Stramigioli and L. Marconi (2012). Modeling and control of a flying robot for contact inspection. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE.

Garcia-Pardo, P. J., G. S. Sukhatme and J. F. Montgomery (2002). "Towards vision-based safe landing for an autonomous helicopter." Robotics and Autonomous Systems **38**(1): 19-29.

Geng, L., Y. Zhang, J. Wang, J. Fuh and S. Teo (2013). Mission planning of autonomous UAVs for urban surveillance with evolutionary algorithms. Control and Automation (ICCA), 2013 10th IEEE International Conference on, IEEE.

George, M. and S. Sukkarieh (2005). Tightly coupled INS/GPS with bias estimation for UAV applications. Proceedings of Australiasian Conference on Robotics and Automation (ACRA).

Harris, C. and M. Stephens (1988). A combined corner and edge detector. Alvey vision conference, Citeseer.

Hasircioglu, I., H. R. Topcuoglu and M. Ermis (2008). 3-D path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms. Proceedings of the 10th annual conference on Genetic and evolutionary computation, ACM.

Huang, Y., W. Hoffmann, Y. Lan, W. Wu and B. Fritz (2008). Development of a spray system for an unmanned aerial vehicle platform, DTIC Document.

Jakob, M., E. Semsch, D. Pavlıcek and M. Pˇechoucek (2010). Occlusion-aware multi-uav surveillance of multiple urban areas. 6th Workshop on Agents in Traffic and Transportation (ATT 2010), Citeseer.

Johnson, A., J. Montgomery and L. Matthies (2005). Vision guided landing of an autonomous helicopter in hazardous terrain. Robotics and automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE international conference on, IEEE.

Johnson, A. E., A. R. Klumpp, J. B. Collier and A. A. Wolf (2002). "Lidar-based hazard avoidance for safe landing on Mars." Journal of guidance, control, and dynamics **25**(6): 1091-1099.

Jones, P. J. (2009). Cooperative area surveillance strategies using multiple unmanned systems, ProQuest.

Kang, Y. and J. K. Hedrick (2009). "Linear tracking for a fixed-wing UAV using nonlinear model predictive control." Control Systems Technology, IEEE Transactions on **17**(5): 1202-1210.

Kim, J.-H., S. Sukkarieh and S. Wishart (2003). Real-time navigation, guidance, and control of a UAV using low-cost sensors. Field and Service Robotics, Springer.

Kim, J. and J. L. Crassidis (2010). UAV path planning for maximum visibility of ground targets in an urban area. Information Fusion (FUSION), 2010 13th Conference on, IEEE.

Kim, J. and Y. Kim (2008). Moving ground target tracking in dense obstacle areas using UAVs. Proceedings of the 17th IFAC World Congress.

Laiacker, M., K. Kondak, M. Schwarzbach and T. Muskardin (2013). Vision aided automatic landing system for fixed wing UAV. Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE.

Lewis, J. (1995). Fast normalized cross-correlation. Vision interface.

Li, Y., H. Chen, M. Joo Er and X. Wang (2011). "Coverage path planning for UAVs based on enhanced exact cellular decomposition method." Mechatronics **21**(5): 876-885.

Liao, S. X. and M. Pawlak (1996). "On image analysis by moments." Pattern analysis and machine intelligence, IEEE Transactions on **18**(3): 254-266.

Lin, P.-H. and C.-S. Lee (2008). "The Eyewall-Penetration Reconnaissance Observation of Typhoon Longwang (2005) with Unmanned Aerial Vehicle, Aerosonde." Journal of Atmospheric & Oceanic Technology **25**(1).

Lowe, D. G. (2004). "Distinctive image features from scale-invariant keypoints." International journal of computer vision **60**(2): 91-110.

Lu, A., W. Ding, J. Wang and H. Li (2012). Autonomous Vision-Based Safe Area Selection Algorithm for UAV Emergency Forced Landing. Information Computing and Applications, Springer**:** 254-261.

Lum, C. and R. Rysdyk (2008). Time constrained randomized path planning using spatial networks. American Control Conference, 2008, IEEE.

Macharet, D. G., A. A. Neto and M. F. M. Campos (2010). Feasible UAV path planning using genetic algorithms and Bézier curves. Advances in Artificial Intelligence–SBIA 2010, Springer**:** 223-232.

Masselli, A., S. Yang, K. E. Wenzel and A. Zell (2014). "A cross-platform comparison of visual marker based approaches for autonomous flight of quadrocopters." Journal of Intelligent & Robotic Systems **73**(1-4): 349-359.

Mastin, A., J. Kepner and J. Fisher (2009). Automatic registration of LIDAR and optical images of urban scenes. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE.

May, K. E., H. J. Sien, Y. S. Ping and S. Z. Hai (2010). Evolutionary approach for trajectory generation of unmanned aerial vehicles (UAVs) over hostile terrain. Computational Problem-Solving (ICCP), 2010 International Conference on, IEEE.

Maza, I. and A. Ollero (2007). Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. Distributed Autonomous Robotic Systems 6, Springer**:** 221-230.

Mejias, L., D. L. Fitzgerald, P. C. Eng and L. Xi (2009). Forced landing technologies for unmanned aerial vehicles: towards safer operations, In-Tech.

Min, R., F. Xiao and S. Lincheng (2013). The threatened environment modeling and evaluation in coordination penetration planning of UAVs. Control and Decision Conference (CCDC), 2013 25th Chinese, IEEE.

Nigam, N. and I. Kroo (2008). Control and design of multiple unmanned air vehicles for a persistent surveillance task. 12th AIAA/ISSMO multidisciplinary analysis and optimization conference. Victoria, British Columbia, AIAA-2008-5913.

Nikolos, I. K., K. P. Valavanis, N. C. Tsourveloudis and A. N. Kostaras (2003). "Evolutionary algorithm based offline/online path planner for UAV navigation." Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **33**(6): 898-912.

Oksanen, T. and A. Visala (2009). "Coverage path planning algorithms for agricultural field machines." Journal of Field Robotics **26**(8): 651-668.

Özalp, N. and O. K. Sahingoz (2013). Optimal UAV path planning in a 3D threat environment by using parallel evolutionary algorithms. Unmanned Aircraft Systems (ICUAS), 2013 International Conference on, IEEE.

Park, J., Y. Kim and S. Kim (2015). "Landing site searching and selection algorithm development using vision system and its application to quadrotor." Control Systems Technology, IEEE Transactions on **23**(2): 488-503.

Pauly, M., M. Gross and L. P. Kobbelt (2002). Efficient simplification of point-sampled surfaces. Proceedings of the conference on Visualization'02, IEEE Computer Society.

Pfeiffer, B., R. Batta, K. Klamroth and R. Nagi (2005). "Path planning for UAVs in the presence of threat zones using probabilistic modeling." Institute of Applied Mathematics, University of Erlangen, Erlangen.

Quigley, M., M. A. Goodrich, S. Griffiths, A. Eldredge and R. W. Beard (2005). Target acquisition, localization, and surveillance using a fixed-wing mini-UAV and gimbaled camera. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, IEEE.

Rackliffe, N., H. A. Yanco and J. Casper (2011). Using geographic information systems (GIS) for UAV landings and UGV navigation. Technologies for Practical Robot Applications (TePRA), 2011 IEEE Conference on, IEEE.

Rosten, E. and T. Drummond (2006). Machine learning for high-speed corner detection. Computer Vision–ECCV 2006, Springer**:** 430-443.

Rusu, R. B. (2010). "Semantic 3D object maps for eachday manipulation in human living environments." KI-Künstliche Intelligenz **24**(4): 345-348.

Santamaria-Navarro, A. and J. Andrade-Cetto (2013). Uncalibrated image-based visual servoing. Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE.

Saripalli, S., J. F. Montgomery and G. S. Sukhatme (2003). "Visually guided landing of an unmanned aerial vehicle." Robotics and Automation, IEEE Transactions on **19**(3): 371-380.

Savla, K., F. Bullo and E. Frazzoli (2007). The coverage problem for loitering Dubins vehicles. Decision and Control, 2007 46th IEEE Conference on, IEEE.

Scherer, S., L. Chamberlain and S. Singh (2012). "Autonomous landing at unprepared sites by a full-scale helicopter." Robotics and Autonomous Systems **60**(12): 1545-1562.

Schneider, P. and D. H. Eberly (2002). Geometric tools for computer graphics, Morgan Kaufmann.

Sereewattana, M., M. Ruchanurucks and S. Siddhichai (2014). Depth Estimation of Markers for UAV Automatic Landing Control Using Stereo Vision with a Single Camera. Int. Conf. Information and Communication Technology for Embedded System.

Serrano, N. (2006). A bayesian framework for landing site selection during autonomous spacecraft descent. Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, IEEE.

Sevcik, K. W., N. Kuntz and P. Y. Oh (2010). "Exploring the effect of obscurants on safe landing zone identification." Journal of Intelligent and Robotic Systems **57**(1-4): 281-295.

Shakernia, C., R. V. Sharp and Y. M. D Shim (2002). A Vision System for Landing an Unmanned Aerial Vehicle. IEEE Int. Conference Robotics and Automation.

Shanmugavel, M., A. Tsourdos and B. A. White (2010). Collision avoidance and path planning of multiple UAVs using flyable paths in 3D. Methods and Models in Automation and Robotics (MMAR), 2010 15th International Conference on, IEEE.

Sujit, P. and R. Beard (2007). Cooperative path planning for multiple UAVs exploring an unknown region. American Control Conference, 2007. ACC'07, IEEE.

Takahashi, M. D., A. Abershitz, R. Rubinets and M. S. Whalley (2013). "Evaluation of safe landing area determination algorithms for autonomous rotorcraft using site benchmarking." Journal of the American Helicopter Society **58**(3): 1-13.

Theodore, C., D. Rowley, A. Ansar, L. Matthies, S. Goldberg, D. Hubbard and M. Whalley (2006). Flight trials of a rotorcraft unmanned aerial vehicle landing autonomously at unprepared sites. Annual Forum Proceedings-American Helicopter Society, AMERICAN HELICOPTER SOCIETY, INC.

Tisdale, J., Z. Kim and J. K. Hedrick (2009). "Autonomous UAV path planning and estimation." Robotics & Automation Magazine, IEEE **16**(2): 35-42.

Tong, W.-S., C.-K. Tang, P. Mordohai and G. Medioni (2004). "First order augmentation to tensor voting for boundary inference and multiscale analysis in 3D." Pattern Analysis and Machine Intelligence, IEEE Transactions on **26**(5): 594-611.

Tóvári, D. and N. Pfeifer (2005). "Segmentation based robust interpolation-a new approach to laser data filtering." International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences **36**(3/19): 79-84.

Vasile, A., F. R. Waugh, D. Greisokh and R. M. Heinrichs (2006). Automatic alignment of color imagery onto 3d laser radar data. Applied Imagery and Pattern Recognition Workshop, 2006. AIPR 2006. 35th IEEE, IEEE.

Verbandt, M., B. Theys and J. De Schutter (2014). Robust marker-tracking system for vision-based autonomous landing of VTOL UAVs. IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, August 12-15, 2014, Delft University of Technology.

Wald, I. and V. Havran (2006). On building fast kd-trees for ray tracing, and on doing that in O (N log N). Interactive Ray Tracing 2006, IEEE Symposium on, IEEE.

Wang, B., X. Chen, Q. Wang, L. Liu, H. Zhang and B. Li (2010). Power line inspection with a flying robot. Applied Robotics for the Power Industry (CARPI), 2010 1st International Conference on, IEEE.

Watt, A. H. and A. Watt (2000). 3D computer graphics, Addison-Wesley Reading.

Weinstein, A. L. and C. Schumacher (2007). UAV scheduling via the vehicle routing problem with time windows. Proc. AIAA Infotech@ Aerospace 2007 Conference and Exhibit. Rohnert Park, California.

Whalley, M., M. Takahashi, P. Tsenkov, G. Schulein and C. Goerzen (2009). Field-testing of a helicopter UAV obstacle field navigation and landing system. 65th Annual Forum of the American Helicopter Society, Grapevine, TX.

Willmann, J., F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio and M. Kohler (2012). "Aerial robotic construction towards a new field of architectural research." International journal of architectural computing **10**(3): 439-460.

Yanushevsky, R. (2011). Guidance of unmanned aerial vehicles, CRC Press.

Zelinsky, A., R. A. Jarvis, J. Byrne and S. i. Yuta (1993). Planning paths of complete coverage of an unstructured environment by a mobile robot. Proceedings of international conference on advanced robotics.

Zhang, W., B. Hyun, P. Kabamba and A. Girard (2013). Improving classification performance through kinematic decisions. Proc. 2013 European Control Conf.