Deployment of Real Time UAV Aerial Surveillance with Coverage Model

Tong Zhang
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/7412

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
Deployment of Real Time UAV Aerial Surveillance with Coverage Strength model

By
Tong Zhang

A Thesis
Submitted to the Faculty of Graduate Studies through the Department of Electrical and Computer Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario Canada
2017

© 2017 Tong Zhang
Deployment of Real Time UAV Aerial Surveillance with Coverage Model

by

Tong Zhang

Approved By:

____________________________________________________
J. Defoe

Department of Mechanical, Automotive and Materials Engineering

____________________________________________________
K. Tepe

Department of Electrical and Computer Engineering

____________________________________________________
X. Chen, Advisor

Department of Electrical and Computer Engineering

Sep 22nd, 2017
Declaration Of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.
This thesis assesses the feasibility of applying coverage model to the problem of using unmanned aerial vehicles (UAVs) for aerial surveillances. The purpose of aerial surveillance is using sensors to cover the task area to obtain the information regarding to this area, such as environmental study. Comparing to static sensors, business purposes UAVs have higher mobility. Since static sensors have limited sensing range, it is not possible to use them to cover the task area. UAVs with sensors onboard can be used for surveillance and save the data to obtain more detailed information of the task area. The data retrieved by the sensors can be used for developing autonomous control algorithms for navigation of the UAV. However, there are some inevitable factors that shortens the flight time of the UAV. For example, considering the maximum payload of the UAV, the battery of UAV can usually last at most 30 minutes since it cannot be very large. It is very important to improve the efficiency of UAV surveillance by pre-designing the flight path for one UAVs or deployment positions for multiple UAVs.

The coverage strength model converts parameter of sensors into geometrical constraints, which allows using mathematical equations to represent the sensor and using convex optimization to find deployment of multiple sensors to cover the task area with best visual coverage and image quality. The coverage model consists of three different models: sensor model, environmental model, and task model. The convex function is constructed based on the three models and optimized to the minimum to get the best deployment.

It is also possible to use autonomous UAV in this application. Thus, the controlling algorithm is also a part of this thesis. Different from application of indoor UAVs, only GPS can be used for positioning and navigating the movement of the UAV. A black box control only based on GPS is presented.

The contributions of this thesis are in two perspectives. First, it applies the coverage strength model to mobile sensors, UAVs. It improves the efficiency of using UAVs to survey the task area. With the method applied, it may need less hardware requirements and time to achieve good result. Second, it is an autonomous UAV that is built for this application. Therefore, every detail of building this drone is introduced in this application including all hardware and software details.

In the following parts of the thesis, the coverage strength model is introduced. The instruments and the controlling algorithm of the UAV are presented. The experimental procedure and the result of applying the coverage strength model are shown as well.
Acknowledgements

First, I would like to thank Dr. Xiang Chen as my supervisor. He helped and taught me during my studies over the past three years for obtaining my master degree. His superior way of thinking and creative ideas inspired me a lot. His patience and effort showed me how to do research with the most correctly and efficiently. It is a great honor for me to have him as my mentor and work with me. I also want to thank other committee members, Dr. Kemal Tepe and Dr. Jeff Defoe for providing valuable feedback and spending time reviewing my work.

I would like to thank to my fellow students Farsam Fazardpour, who is currently working as a PHD candidate in the lab. The discussion and the work we have done also gave me a lot of ideas in finishing my thesis. He also gave me a lot of suggestions for improving my work.

I would like to thank my family and my friends who supported me during my years of study. My Parents, Xinye Zhang and Yanrong Liu, they raised me with love and patience over the last 26 years. They also mentally and financially supported me during my study in a country other than my homeland. My grandparents, Changxiu Lu and Tianxiang Liu, showed me the way of being a kind and generous person.

Finally, I would like to thank University of Windsor. It gave me the chance to study abroad to for undergraduate and master program.
# Contents

Declaration of Originality ........................................ iii
Abstract ................................................................ iv
Acknowledgements ....................................................... v
List of Tables ................................................................ viii
List of Figures ................................................................ ix
List of Algorithms and Definitions ........................... x
List of Equations .......................................................... xi

1 Aerial Surveillance with UAV .......................................... 1
   1.1 Aerial Surveillance with UAVs ................................ 1
   1.2 Motivations ......................................................... 2
   1.3 Propositions ......................................................... 5
   1.4 Thesis Outline ...................................................... 6

2 Coverage Strength Model and Optimization Method .......... 7
   2.1 Overview ............................................................. 7
   2.2 Coverage Strength Model ...................................... 7
   2.3 Partition and Optimization .................................... 13

3 Hardware and software .................................................. 18
   3.1 UAV selection ...................................................... 18
   3.2 Hardware Structure of UAV .................................. 20
   3.3 Camera Parameters .............................................. 21
   3.4 Other Sensors ...................................................... 23
   3.5 Software ............................................................. 23

4 Experiment ............................................................... 26
   4.1 Indoor Experiment ............................................... 26
   4.2 Outdoor Experiment ............................................. 28
   4.3 Conclusion .......................................................... 32

5 Results and Analysis .................................................... 34
   5.1 Indoor Experiment Results .................................... 34
   5.2 Outdoor Experiment Results ................................. 36
   5.3 Analysis .............................................................. 39

6 Conclusion and Future work .......................................... 40
   6.1 Conclusion .......................................................... 40
   6.2 Future work ......................................................... 40

Appendix A Equipment ................................................... 42
Appendix B Codes .......................................................... 45
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibliography</td>
<td>65</td>
</tr>
<tr>
<td>Vita Auctoris</td>
<td>68</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 UAV Specification</td>
<td>19</td>
</tr>
<tr>
<td>4.1 Channel Override Value</td>
<td>30</td>
</tr>
<tr>
<td>4.2 PID Tuning</td>
<td>31</td>
</tr>
<tr>
<td>5.1 Indoor Deployment Positions</td>
<td>34</td>
</tr>
<tr>
<td>5.2 Coverage</td>
<td>36</td>
</tr>
<tr>
<td>5.3 Outdoor Deployment Positions</td>
<td>38</td>
</tr>
<tr>
<td>5.4 Outdoor Coverage</td>
<td>38</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Difference in Coverage 3

2.1 Resolution 8
2.2 View angle and Field of View 8
2.3 Normal Coverage Model 10
2.4 Coverage Strength Model 11
2.5 Multiple coverage model 12
2.6 Voronoi Partition 14
2.7 Optimized Sensor Position 15
2.8 Path of Sensors and Total Coverage Function 17

3.1 Quad Copter DJI F450 19
3.2 Hardware Structure of Ardrone 21
3.3 Hardware Structure of DJI F450 21
3.4 Camera Parameters 22
3.5 Simulation of the UAV 25

4.1 Indoor Test Area 26
4.2 Image and measurement 27
4.3 Test Field 28
4.4 Remote Controller 29
4.5 Channel Override from Mission Planner 29
4.6 Altitude Simulation 31
4.7 Controller Scheme 32

5.1 Deployment and Percentage of Coverage 35
5.2 Image and deployment at point (4.13, 5.0) 35
5.3 Sensor Mapping 37
5.4 Trajectory of UAV and Image at feature point 37

A.1 Onboard Computer 42
A.2 Flight Controller 42
A.3.1 Logitech web camera 43
A.4 GPS 43
A.5 Parrot Ardrone 44
A.6 General Purposed Computer 44
# List of Algorithms and Definitions

1. Definition (coverage function) 9
2. Definition (sensor characteristics) 11
3. Definition (multiple coverage function) 12
4. Partition algorithms 14
5. Gradient 15
6. Optimization algorithm 16
7. Camera Parameters 22
8. Parameters of the UAV 30
Chapter I

Introduction to Aerial Surveillance

1.1 Aerial Surveillance with UAV

UAVs, which is the abbreviation of Unmanned Aerial Vehicles or also known as drones, are aircrafts with no pilot onboard. There are usually two ways of operating this kind of flying drones, either remotely controlled by ground station with commands or controlled by the onboard computer with pre-defined algorithms autonomously [1]. Due to this easy-manipulation feature, UAVs can be used in many different applications, such as military, entertainment, science and technology [2][3]. UAVs have high mobility and maneuverability, it can go places where humans cannot go. It is often assigned with tasks which people consider dirty or dangerous. UAVs were originally designed for military uses which makes the size quite big, but after many years of evolving, the size of the drones is decreasing. More UAVs equipped with sensors, such as cameras, sonars and lidars, are launched on the civil market. It turns out that drones can perform very well in environmental studies such as retrieving information about some certain area for learning the species. Those drones are considered as the best choice for environmental studies because of low cost, high mobility, real time data transmitting function [4][5].

Aerial surveillance provides a reliable solution to environmental study. The goal is to monitor a certain area to get information about the target area. The task is to cover the area with a minimum number of blind spots. In other words, the problem of aerial surveillance can be reformed as an area coverage problem. In a traditional area coverage problem, static monitoring instruments, for example cameras, are used to cover the whole area. However, for environmental study problems, the scenario is different. The task area is larger in environmental study case so that static sensors may not be able to reach the designated spots. Comparing with static sensors, mobile vehicles with sensors installed, especially aerial vehicles with cameras, are preferred in this application [6].

There are mainly two methods used separately in aerial surveillance with UAVs based on different number of UAVs used in this application. one method is for one or a small group of UAVs to survey the whole task area with pre-designed path [7][8]. In other words, when the number of UAVs is not enough to cover the area, the path method is used for surveillance. The other method is for large number of UAVs to cover
the task area with assigning them to some specific positions. However, the selection of path and deployment determines the aerial surveillance result.

In communication, the area is usually split into small clusters and the station are built at the centre of each cluster to satisfy the needs of users [9]. The same idea can also be applied to this application because the idea of sensor networks is introduced here [10]. Sensor networks consists of multiple sensors and have communication ability between the nodes [11]. Each sensor is responsible for one area which shares the same idea as clustering. The centre of the cluster can be considered as the position of the sensor and the cluster is the area covered by each sensor. This idea can be also applied on this application. Based on the sensor coverage area, finding the best deployment of the UAVs to cover the task area is the strategy.

1.2 Motivations

Choosing UAV as the media for environmental study has many advantages and disadvantages [12]. One of the disadvantages is the flight time problem. The size and power supply method restrict the duration of flight time. Usually, business purposed UAV can fly for approximately at most 35 mins. Therefore, it is essential to find the most efficient way to survey or monitor the task area. Galceran and Carreras [13] mentioned that sweep solution can provide a good coverage with one drone. When the task area is large, the UAV may not be able survey the whole area due to battery issue. Also, since sensors have blind spots, it is possible that some places are not covered during sweeping. Therefore, the problem is to find the most efficient path or deployment that can cover the area with minimum number of blind spots.

To find the most efficient path or deployment, both positions and orientations of the sensors should be considered. The key to solve this problem is to use some optimization method. However, for most sensors, only statistical data is provided which is none helpful to the optimization part. Like other optimization problems, a cost function should be used to evaluate the result of optimization [14]. Most sensors are provided with data such that includes the characteristics of the sensors. Therefore, the key problem is to construct a mathematical model from the parameter data of the sensors. The coverage strength model is introduced in this thesis. The sensor’s coverage area can usually be transformed into a geometric shape which represents the intrinsic characteristics of the sensors [15]. The geometric shape can be presented with mathematical equations, which is called coverage strength functions. A coverage strength function provides a feasible solution for finding the best deployment of the sensors.
For an environmental study purposed UAV, the sensor, which usually is the camera, can be placed either at bottom of the drone or in the front of the drone. The images taken from the camera are from aerial view for bottom camera [16] and from horizontal view front camera. The purposes of putting camera at different positions are manifold. For example, if the camera is put at the bottom, it is more used to retrieve information about the target area, such as size and objects. If the camera is placed at the front, it is used for real time monitoring. M Schwager et al [17] introduced a method of covering the area with multiple UAVs with downward facing cameras for constructing the sensor model. Normally the target area is considered as a 2D flat surface, and it is not necessary to worry about pitch-roll-yaw of the camera if a downward facing camera is used for surveillance because the coverage is the projection of the bottom view of a cone, which is a circle. This reduces the workload of constructing the sensor’s model, which gives a possibility to find the optimized deployment.

![Figure 1.1 Difference in Coverage](image)

Figure 1.1 shows the geometric shape of coverage when the same camera is used on UAV but mounted at different positions. The projection of the coverage onto the 2D task area is the model discussed in this thesis. Even though the camera’s coverage is cone, with different view, the coverage models that are used for optimization are different which could be either a circle or a circular sector. As a result, the deployments with different orientations of the camera are different.

### 1.2.1 Model construction and deployment method

As mentioned above, the model construction is essential in this problem because an accurate model determines whether the optimized result matched the real deployment. The coverage model provides a possible mathematical solution for finding the deployment of the sensors. This coverage model is formed based on the parameters of the sensors. Some parameters, such as the coverage range and the coverage shape,
should be considered [18]. The construction of the coverage model is introduced in later chapters.

The deployment is another issue for the sensors to cover the task area. The most efficient solution is to use the least number of sensors to cover more area. In most researches, the deployment is determined by a very simple method. However, in this thesis, since the coverage model is built and it is a mathematical model, it is possible to get better deployment with multiple UAVs with optimization methods. The sensor’s position and orientation (roll-pitch-yaw) are optimized to get a better performance [19] for general 3D sensor deployment problems.

1.2.2 Multiple drones’ deployment and one drone’s path planning

In aerial surveillance problems with UAVs, there are two different scenarios, either with one UAV or multiple UAVs. For one drone, the area coverage method should be path planning. The drone should fly among a pre-defined path. The task area is fully surveyed when the drone is flying along that path and the information is recorded with the sensor mounted on the UAV. For multiple drones, the UAVs should move to pre-defined points and rotate in the right direction for monitoring, and taking short videos or real-time images in the whole task area. The UAVs can be deployed from a starting position and reach the optimized positions. Theoretically, the UAV should move when the optimization is ongoing because the purpose of applying coverage strength model and optimization is to for autonomous sensor deployment. In other words, with a carrier such as robotic arm or UAV, the sensor should be capable of moving to the right position with no human interaction. This can improve the accuracy of deployment. Thus, the deployment method affects the coverage of the task area.

Although the one drone’s path planning and multiple drones’ deployment are two scenarios, they share the same principle. Maza et al [20] shows that the area coverage of using multiple UAVs can be reformed as using one drone to survey the area. For both cases, the positions that the UAVs should go are decided by several factors, including sensors’ model, initial position, and optimization method. This thesis provides a solution for finding the best deploying points and explains how this could work for both cases.

1.2.3 Autonomous drones

In this thesis, an autonomous drone that is specifically designed for aerial surveillance is used. Autonomous drones are popular in business and scientific topics. They could be used in many different areas. For example, UAVs can be used for business delivery [21]. Both the deployment positions and drone control can be done by computers with pre-defined algorithms, which means during this progress no human actions are needed. With this method, the drone can do aerial surveillance
autonomously. In this thesis, the building of an autonomous drone that is used for the experiment is presented in chapter 3.

To avoid any safety issues, the drone can be always shut down with a kill switch on the remote controller, if the drone flies out of sight or the autonomous algorithm does not work. In autonomous controlling algorithm, there is also an boundary for the UAV’s movement. If the UAV flies out of the boundary, the drone is terminated immediately.

1.2.4 Future vision

The thesis intends to solve a 2D problem of aerial surveillance. However, in real world, the environmental is a 3D situation. For 2D cases, only position and orientation, more specifically, yaw of the UAV needs to be considered. For 3D cases, not only position and yaw angle, but also altitude, pitch and roll are the factors that affect the coverage of the area. On most business purposed UAVs, there is a motor-driving pan-tilt controlling device where the camera is mounted. The orientation or the camera can be adjusted with the device.

Coverage model has already been used for static cameras to cover some object from different positions and angles in 3D environment. Therefore, the future improvement can be expanding this 2D aerial coverage problem into 3D aerial coverage problem for UAVs. The coverage strength model needs to be expanded to 3D cases.

1.3 Proposition

To achieve the best the coverage, a mathematical solution should be introduced here. Since both the sensor’s model and task area are some geometrical shapes, a geometrical matrix should be constructed to represent the sensors [18] [19]. The geometrical matrix enables the calculation for the best positions and orientations, which will be discussed in chapter 2 and 3. This geometrical matrix also needs to satisfy the properties of the sensor. For example, if a sensor’s sensing area is a sphere, the geometrical model cannot be a cube. That is the basis of constructing the geometrical model and matrix.

The optimization method is as important as the sensor’s model in solving the problem of aerial surveillance. It is used for optimizing the best view angles and positions for the UAVs. In optimization problem, a cost function needs to be used to estimate whether the optimized results meet the requirement. In this thesis, the cost function is built based on the geometrical sensor’s model. The optimization can be reformed as a control problem having two inputs, which are positions and orientations, and one output, which is the cost function. The result of the cost function should be as small as possible to maximize the coverage.
The real deployment of the UAVs is another problem. The UAVs should reach the proposed position and orientations to get the best coverage. During this progress, accurate robotic control is required. The sensors of the UAVs, including Global Positioning System sensor (GPS), inertial measuring unit (IMU), camera and other sensors, provide detailed information about the flight status of the UAVs. This gives another challenge for controlling the UAVs. For autonomous drones, no human interaction is taken during the whole surveillance progress. Thus, the control loop should be designed carefully. All the problems in the thesis are analyzed and solved from the perspective of control theory.

1.4 Thesis Outline

This thesis begins with an introduction to the coverage strength model and optimization method in chapter 2. Section 2.1 gives the reason that coverage model can improve the efficiency of aerial surveillance. Section 2.2 introduces coverage model which includes single coverage strength model and multiple coverage strength model. Section 2.3 presents the partition and optimization method that are used for finding the best deployment for UAVs.

Chapter 3 introduces the hardware and software that are used for finishing the thesis. Section 3.1 talks about the two types of UAV that are selected for the experiments. Hardware structure, including detailed plots, are the main components of section 3.2. Section 3.3 presents the camera’s coverage range and parameters, and section 3.4 mentions the other sensors that are related to the project. In section 3.5, the software that are used for simulation and installation is shown.

Chapter 4 is about the experimental setup and details. Section 4.1 presents the indoor UAV deployment experiment and section 4.2 has the information of outdoor UAV deployment experiment. The test area, the deployment method and flight control algorithm are introduced separately for indoor and outdoor experiments.

Chapter 5 shows the real experiment results and simulation results to judge whether the coverage strength model can be applied to UAVs for both indoor and outdoor cases. It also shows the advantage of using coverage strength model to find deploying points for single UAV surveillance other than sweeping methods.

Chapter 6 ends with conclusion and future improvements.
Chapter II

Background

2.1 Overview

This is the literature review part that specifically introduces coverage model and optimization. All relative work is also referred in this part for giving the readers a better opinion of sensor’s model and coverage strength matrix.

The original coverage strength model consists of three different models, which are sensor’s model, environmental model, and task model. Since this is an area coverage problem and there is not specific object to measure, the task model is objective which in this case, is to use UAVs to cover the whole area. In this chapter, the coverage model is introduced. Also, as mentioned above, the optimization method is very necessary to solve this problem. At the very beginning, the UAV might not be deployed at the best positions to cover the area. The optimization can be used to find out the best positions of the drone. It is also introduced in this chapter.

2.2 Coverage Strength Model

2.2.1 Sensor’s model

A well-constructed sensor’s model has been proven to be very useful industrialized process and autonomous robot areas [22] [23]. Especially for autonomous robot, which has attracted many interests from both industry and academy, requires very accurate sensor’s model for simulation, testing and experiments. Tisdale’s et al [24] and Schwager’s [20] work both propose a method to apply sensor’s model to solve the problem of aerial coverage with multiple UAVs. Kannala and Brandt [25] constructed models for cameras with different lenses. The approach converts the sensor’s parameters to geometrical constraints. Fisheye cameras are mostly omni-directional, which means they have wide but single directional FOV. It is also very important to build a sensor’s model for fisheye cameras on robotic applications [26]. There are many parameters for a sensor. However, for different applications, the parameters that need to be included are not the same. For aerial coverage problem, these are the factors that must be evaluated for the camera’s model.

1. Resolution
2. FOV, field of view and View Angle

To get the best sensor’s model for cameras in this application, it is very important to know the fundamental of the cameras.
The resolution is one of the most important features for cameras. It stands for the units of length that are represented by a unit pixel in an image as shown in figure 2.1. Depending on the cameras used, the resolution may not be the same. Normally, this resolution is affected by the focal length of the lens. Nowadays, cameras can easily change the thickness of the lens to get different focal lengths.

![Figure 2.1 Resolution](image)

FOV is another important characteristic of the cameras. It is defined as the area captured on the camera’s imager. For most cases, FOV refers to the angle and the distance that the sensor can cover. It is the other parameter that decides the sensing area of the camera. For different cameras, the FOVs are different. FOV includes horizontal field of view, vertical field of view and diagonal. For different scenarios, the importance of using these FOVs are not the same. Sometimes, maybe one or two of them are negligible.

View angle is a third feature that should be considered. It represents orientation of the camera in local coordinate. For finding the best deployment for UAVs, the view angle of sensor mounted on each UAV decides covered area. If all sensors are in proper positions and orientations, the task area should be covered with almost no blind spots.

Figure 2.2 shows the field of view and view angle. Although these are two factors that are affecting the coverage performance, to achieve the best coverage, these two parameters are combined with one equation.

![Figure 2.2 View angle and Field of View](image)
2.2.2 Environmental model

The environmental model should contain the structure and contents of the environment. Information of the target area, such as:

1. size and shape of the target area
2. static objects (e.g. trees)
3. dynamic objects (e.g. vehicles)

It is important to construct the environmental model because the size and shape of the task area also affects the deployment’s performance. The shape and size of the test area can vary. As a result, the environmental model should also change correspondingly.

Depending on the complexity of the environment, there might be extra factors that need to be added while constructing the environmental model [15]. In this problem, since the experiment is conducted either indoor or at an open space in the park, most of the factors listed above are not necessary. The size and shape of the target environment is much more important.

2.2.3 Task Model

The task model is the purpose of using the coverage strength model. In this case, the task model is using sensors mounted on the UAV to cover a task area.

2.2.4 Coverage strength function

As mentioned above, the coverage strength model consists of sensor’s model and environmental model. The coverage strength model provides a mathematical solution to get the best deployment. In this section, the coverage strength model is described. The coverage model is a model that represents how well a sensor is covering the target area. It means that every single position is assigned with a coverage value. For normal sensor model, a binary value of 0 and 1 is used for representing whether that position is covered or not. In coverage strength model, all strength values are between 0 and 1.

![Image](image.png)

1 Definition (coverage function)

Given a 2D target area, coverage function assigns a strength value to each point in the area. C: [0,1]. C(p), where p means all the positions inside the area.

The strength value used here are factors related. All the three parameters of the sensor, which are resolution, FOV and view angle, form this coverage strength function. In addition, the position of the sensor in the target environment also affects the coverage strength function. The equation below is the coverage strength function. It describes the components of the coverage strength function.

\[
C(P) = e^{C_R(P) \times C^P(P) \times C^F(P)}
\]

(2.1)
1. $C^R(P)$ represents the Euclidean distance from a certain position to the sensor’s location.

2. $C^v(P)$ represents the angular difference between a certain position and the camera’s view angle in camera coordinate.

3. $C^F(P)$ represents the FOV.

An example of coverage model is shown below explaining about how the mechanism works. Figure 2.3 shows the normal coverage model. Normal coverage model mostly shows whether the position in the task area is covered. Binary values are used to represent the coved and uncovered area.

![Figure 2.3 Normal Coverage Model](image)

Different from normal coverage model, coverage strength model contains more information about the task area. Each position in the uncovered area is assigned with a strength value that is different from zero. This strength value is determined by several factors, such as the Euclidean distance between the point and the centre of the sensor, the angular difference and the field of view.
As mentioned above, the strength values are determined by three different factors, the Euclidean distance, the view angle and field of view. Given the characteristics of the sensor, mathematical equations are used for constructing the coverage strength model. Before showing the equation, some definitions need to be made here.

2 Definition (sensor characteristics)

1. \( r \): The maximum coverage distance for a directional sensor
2. \( \Theta \): The view angle. (0,2\(\pi\))
3. \( \beta \): The FOV of the sensor
4. \( a, b \): The position of the sensor

In this thesis, the forward-facing camera is used for surveillance. The altitude of UAV is set to be 1 to 1.5 meter away from ground for taking pictures and videos. If a downward facing camera is used, the altitude of drone also affects the performance of coverage strength model.

\[
C^R(P) = \frac{\text{Max}(r, \sqrt{(X(P)-a)^2+(Y(P)-b)^2})}{r}
\] (2.2)

The equation above determines the \( C^R(P) \), which refers to the Euclidean distance term. Based on the sensor position, this equation gives coverage strength value to all positions in the task area. Based on the maximum coverage distance, \( r \), the covered area is given a strength value of one. The uncovered area is given a value starting from one up to infinity.

\[
C^\theta(P) \times C^\beta(P) = \frac{\text{Max}(\frac{\beta}{\pi}|\Theta-(\tan^{-1}\frac{y(P)-b}{x(P)-a}+\pi)|)}{\frac{\beta}{\pi}}
\] (2.3)
This equation determines the angular difference between a certain position and camera’s view angle in local coordinate. The FOV is also considered into calculation in this equation. If the position is in the camera’s FOV, the coverage strength value is one. Otherwise, the coverage strength value is large than 1.

The combination of these two components gives the coverage strength model. The exponential makes the function decay.

2.2.5 Multiple coverage strength

In the previous section, the coverage strength model for one sensor is introduced. However, in real cases, there are multiple UAVs (Sensors) in the task area. To represent multiple sensors, the multiple coverage strength model is applied here. The multiple coverage strength model is combining each individual sensor.

\[
C_m(P(x)) = \max(C_n(P(x))) \quad \text{where } n = 1, 2, 3, \ldots
\]  

(2.4)

Figure 2.5 Multiple coverage model

In this equation, \(C_m\) is the multiple coverage strength, and \(C_n\) represents the coverage strength for each sensor. This multiple coverage strength model still follows the rule of constructing coverage model for single sensor. However, in multiple coverage model, each point in the task area is assigned with the largest coverage strength value from each single coverage model. In Cartesian coordinate, one point is defined to be closer to the origin if the Euclidean distance is smaller. In this case, a point in the task area (if not covered) is defined to be more possibly covered by a sensor if the coverage value is larger.

3 Definition (multiple coverage function)

\[C_m(P(x)) = 1 \text{ if point } x \text{ is covered by any sensor.}\]

\[C_m(P(x)) \in (0,1) \text{ if point } x \text{ is not covered by any sensor.}\]
$C_k(P(x) > C_i(P(x))$ point x is more possible to be covered by sensor k.

Multiple coverage model provides information for partition and optimization. For sensor networks, most optimization method used is distributed optimization. However, the information of each sensor still needs to be sent to a central processing unit. The central processing unit process the data, optimizes cost function and give updated positions to the sensors. Next section presents the partition and optimization method.

2.3 Partition and Optimization

In this thesis, the optimization method used is decentralized optimization method (or distributed optimization method), which is very commonly used in multi-sensor network problems. Decentralized optimization is used to solve a problem when there are many subsystems and each of them is optimized individually. This optimization method assigns each sensor an area to work on and optimize the sensor’s positions to get best coverage [27] [28]. Therefore, a partition method is needed for separate the task area.

2.3.1 Voronoi Diagram

Voronoi Diagram partitions an area based on the distance into specific amount of sub areas. In this thesis, the coverage strength value gives all points in the task area of the probability covered by a certain sensor. In figure 2.4, the yellow area is the covered by the sensor. In that area, all the points have to probability of one to be covered. The area with darker color has lower probability to be covered. The task area is partitioned based on the probability.

\[ P(x) \in j \mid d_i(P(x) < d_j(P(x)) \] x is a point on surface P \hspace{1cm} (2.5)

As mentioned in previous sections, for a point in the task area, it is more possible to be covered if the coverage strength value in sensor i is bigger than sensor j. Figure 2.6 shows the partitioned task area based on the multiple coverage strength model. For some overlapping sensors, it is hard to separate a clear task area. Some of them might not be continuous area. Therefore, optimization method needs to be applied to change the sensor’s position to maximize the coverage.
Figure 2.6 Voronoi Partition

Figure 2.6 only shows assigned area for each sensor at the beginning, which is based on the starting position of the sensors. The sensors’ positions are optimized. Therefore, the partitioned area is updated based on the optimization result. This process is introduced in next section.

4 Partition algorithms

\[ C_m(P(x)) = 0 \]

i = 1

if \( C_m(P(x)) < C_i(P(x)) \)

P(x) is assigned to be in Sensor i’s coverage region

\[ C_m(P(x)) \text{ gets new value from } C_i(P(x)) \]

i + 1

if i = number of sensor

no: repeat

yes: end

2.3.2 Optimization

The optimization method is used for optimizing the sensors’ positions to get the best coverage. In figure 2.7(a), the sensors are overlapping with each other. As a result, the coverage is not maximized. There are still a lot of uncovered area.

Gradient Descent Optimization method is used in this thesis. This method is using gradient descent, which takes steps proportional to the negative of the gradient of the function to reaches the local minimum of the function.

\[ p_{n+1} = p_n - y\nabla F(p_n) \quad (2.6) \]
Since it is gradient descent, the values at each point of cost function $F$ is decaying. The $Y$ is the step size, which is number between 0 and 1. It is very essential to decide the most appropriate value for step size.

![Figure 2.7 (a) (b) Optimized Sensor Position](image)

Figure 2.7 (a) and figure 2.7(b) show the starting and ending positions, and the movement of the sensors. It is clear to see that after some iterations, the coverage saturates. Even though the sensors are covering the area in most efficient way, the number of sensors restricts the total coverage. Since each sensor has a certain coverage, if the task area is larger than total coverage that all sensors can cover, the coverage percentage saturates.

In the distance equation, there are three parameters, which are $X$, $Y$, and $\theta$. The gradient of coverage strength function determines the sensor’s movement in each direction.

5 Gradient

\[
\begin{align*}
\dot{x} &= \frac{d}{dx} F(x, y, \theta) \\
\dot{y} &= \frac{d}{dy} F(x, y, \theta) \\
\dot{\theta} &= \frac{d}{d\theta} F(x, y, \theta)
\end{align*}
\]

\[
F(x, y, \theta) = \frac{\text{Max} \left( r, \sqrt{(X(p) - a)^2 + (Y(p) - b)^2} \right)}{r} \times \text{Max} \left( \frac{\beta}{2}, \theta - \left( \tan^{-1} \frac{y(p) - b}{x(p) - a} + \pi \right) \right)
\]

For coverage strength model, the most efficient coverage means that no overlapping coverage exists and all sensors are at the best positions to cover the assigned area. In this case, the optimization should be based on the result from Voronoi partition. To avoid overlapping coverage, each sensor should move inside the assigned area. To
achieve the maximum coverage for each assigned area, the summation of the coverage strength value of that area should be minimum.

6 Optimization algorithm

After Voronoi partition

\[
\begin{align*}
\dot{x} & = \sum \mathcal{V} \left[ \frac{d}{dx} F(x, y, \theta) \quad \frac{d}{dy} F(x, y, \theta) \quad \frac{d}{d\theta} F(x, y, \theta) \right] \text{ for all point } (x, y) \in \text{sensor } i \\
\dot{y} & = \sum \mathcal{V} \left[ \frac{d}{dx} F(x, y, \theta) \quad \frac{d}{dy} F(x, y, \theta) \quad \frac{d}{d\theta} F(x, y, \theta) \right] \\
i_{x+1} & = i_x - y \dot{y} \\
i_{y+1} & = i_y - y \dot{y} \\
i_{\theta+1} & = i_{\theta} - y \dot{\theta} \\
i + 1 & \text{ until } i = n \text{ for which } n \text{ is the number of sensors}
\end{align*}
\]

back to Voronoi Partition

The sensors’ positions are updated in each iteration, which means in each iteration, partition and optimization should be applied. After many iterations, the sensors’ positions are optimized and the coverage reach certain percentage. However, most optimization are NP hard problems, which means optimization can always continue. Therefore, there should be a threshold that can evaluate when the optimization can stop.

Figure 2.8 shows movement of the sensors. It is clear to see that after some iterations, the coverage saturates. Even though the sensors are covering the area in most efficient way, the number of sensors restricts the total coverage. Since each sensor has a certain coverage, if the task area is larger than total coverage that all sensors can cover, the coverage percentage saturates.
2.4 Conclusion

This section provides detail information about how coverage strength model can be applied in optimization. Since it is a sensor network, the optimization should be decentralized. For each sensor, the position is optimized inside the assigned area. In all previous mentioned coverage models, they are for general directional sensors. To apply them on the moving UAVs, all the parameters should be from the cameras. In next chapter, the real coverage model based on the camera used in the application and environment is introduced. Also, the information of the UAV is revealed.

Figure 2.8 Trajectory of Sensors’ movement
Chapter III
Hardware and Software

3.1 UAV Selection

There are two different kinds of UAVs which are multi-rotor and fixed wings. The mechanical structures of these two UAVs are different. As a result, the performances and applications that these two UAVs are applied are different.

Fixed wing UAV is similar to modern aeroplanes. It is larger in size and can fly at high altitude with very fast speed. For aerial surveillance application, it needs high resolution camera because the image can only be taken at high places. It moves from point to point very quickly and gathers data very efficiently. However, it cannot be used for indoor applications since it cannot hover during flight mission and is more dangerous. In addition, it does not have spaces for modification. It is hard to add more sensors once it is finished assembly. It is stable and less affected by the environmental conditions because of its structure.

Multi-rotor UAV is equipped with brushless rotors. The size of multi-rotor varies from the scale of centimeters to meters. Different from fixed wing UAV, multi-rotor can fly with slow speed but still maintain stable. It also can hover during flight mission and is safer than fixed wing UAV when being operated. Multi-rotor has more potential to be modified in hardware and software for different applications. L.Wallace et al [29] shows that UAV with Lidar system can be used for applications to forest inventory. A software is developed to plan UAVs stereoscopic flight to monitor the area to predict earthquakes [30]. However, for outdoor surveillances, the multi-rotor is weak to resist conditions changes.

Considering the cost and the scenarios for using UAV, multi-rotor, more specifically, quad-copter is selected to be used in this thesis. For aerial surveillance with coverage model, the most important feature is to hover and take real time images or videos. Quad-copter can hover and remain at a position in both indoor [31] and outdoor scenarios. In this thesis, two quad-copter drones are used for indoor and outdoor experiments, Parrot Ardrone 2.0 and DJI Flamewheel F450.

Parrot Ardrone 2.0 is a business purposed UAV. The size is small comparing to other quad-copters. The speed of Ardrone is not very fast, which gives more space for precise control. It has cameras for taking and recording image, ultrasonic for keep a safe distance from ground and IMU for monitoring the flight status. It has no default GPS which can be used for outdoor tracking. Its flight controller and onboard processing
computer are integrated as one microcomputer to save spaces. Its communication range is short because the drone creates its own Wifi for receiving and transmitting data. It has protective polyfoam for crash and collision. The development kit provides a python package for coding. Due to these characteristics. It is selected for indoor experiment.

DJI Flamewheel F450 is a quad-copter frame. It can be modified based on the requirements of the mission. The payload of the frame is 1.4 Kg, which means the total weight of the accessories that can be mounted on this frame is 1.4 Kg. Since every component of this UAV is available for modification, the performance of the UAV is determined by hardware capability and software practicability.

![Figure 3.1 Quad Copter DJI F450](image)

The table 3.1 shows the two quad-copters used in the experiments.

<table>
<thead>
<tr>
<th>UAV Type</th>
<th>Parrot Ardrone 2.0</th>
<th>DJI Flamewheel F450</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manipulation</td>
<td>Apps on mobile phone or computer</td>
<td>Remote Controller or onboard Computer</td>
</tr>
<tr>
<td>Size(mm)</td>
<td>L450 x W310 x H130</td>
<td>L450 x W340 x H160</td>
</tr>
<tr>
<td>Speed(m/s)</td>
<td>0 - 20</td>
<td>0 – 100</td>
</tr>
<tr>
<td>Default Sensors</td>
<td>Cameras, Ultrasonic, IMU</td>
<td>GPS, IMU</td>
</tr>
<tr>
<td>Onboard Computer</td>
<td>Yes</td>
<td>Available for modification</td>
</tr>
<tr>
<td>Safety</td>
<td>Protective Polyfoam</td>
<td>None</td>
</tr>
</tbody>
</table>
3.2 *Hardware Structure of UAV*

This section presents the hardware structure and layout. In previous section, it mentions that two UAVs are used in the experiment for different environment. Although these two drones are different, the hardware structure should be similar. Based on the application, the sensors mounted on the UAV might be slightly different. For example, in indoor aerial coverage application, GPS sensor is not necessary because GPS signal is not available indoor. Ultrasonic is used to keep a safe distance between the UAV and the ground. It is very necessary for indoor application since the room is small.

The onboard computer is the brain of the UAV in this application. It receives data from onboard sensors, processes the raw data, and sends command to the flight controller to adjust the UAV position and orientation. After receiving command signals from onboard computer, the flight controller sends PWM signals to rotors.

For Parrot Ardrone, the onboard computer and the flight controller are integrated onto one microcomputer. However, this microcomputer is not allowed to be re-coded to be autonomous drone. The development kit solves this problem. The controlling algorithm can be edited in python on a computer. This computer can be considered as the ground station. When running the code, Ardrone’s onboard computer access the ground station via Wifi shell created by itself. It runs the code on the ground computer remotely to be autonomous. If the drone flies away from the maximum communication range, the algorithm does not work anymore.

For DJI flamewheel F450, the onboard computer and flight controller are separate. The flight controller is Pixhawk and the onboard computer is Raspberry Pi 3. They both can be physically installed on the frame of DJI. Different from onboard computer of Ardrone, any codes can be directly stored and loaded on Raspberry Pi, which makes it fully autonomous as an individual. The ground station is only used for monitoring the position and flight status of the UAV. The above figures show the structure of two UAVs used in the experiment. Even though the sensors and hardware structure are slightly different, they both function in the aerial coverage experiment.

Figure 3.2 shows the structures of Parrot Ardrone and figure 3.3 shows the structure of DJI F450. On Parrot Ardrone, each component independent from other components. On DJI F450, IMU is part of the flight controller. Wifi module belongs to the onboard computer.
### 3.3 Camera Parameters

In chapter 2, the importance of constructing the accurate sensor’s model is presented. It directly affects the error in simulation result and real experiment results. To find the most appropriate sensors’ model, the sensors’ parameters are the necessary data. In this thesis, two cameras are applied separately on two UAV frames.

Parrot Ardrone uses a HD camera which has a resolution of 720P and a refresh rate of 30 FPS. It can take and record high resolution images. The format of the picture is JPEG. DJI F450 is equipped with a Logitech web camera C910, which has a resolution of 1080P and refresh rate of 30 FPS. The resolutions above for two cameras are the
maximum resolution. Using OpenCV, which is introduced later, can change the resolution from software aspect.

It is very important to find the coverage range of the camera because it directly affects the accuracy of mapping the sensors. An inaccurate sensor’s model produces huge error between simulation result and real deployment result, and lead to worse coverage. In this thesis, the method that is used to obtain the accurate coverage range is to measure the length based on the image taken by the camera.

Figure 3.3 shows the 3D model of the camera. The horizontal field of view and vertical field of view can be calculated by equation 3.1 and 3.2. The FOV and camera detecting range are used for constructing the coverage strength model.

### 7 Camera Parameters

- **cx, cy:** Real distance between center of the frame and edge of the frame
- **dx:** distance between camera to the setting point in the image
- **HFOV:** horizontal FOV
- **VFOV:** vertical FOV

\[
VFOV = \tan^{-1} \frac{cy}{dx} \quad (3.1)
\]

\[
HFOV = \tan^{-1} \frac{cx}{dx} \quad (3.2)
\]

In this thesis, the focus of the problem is a 2D case, which means only horizontal field of view and horizontal coverage range is considered. In chapter 4 and 5, the real measurement of the camera is presented.
3.4 Other Sensors

In this experiment, there are also others sensor used, such as GPS, IMU and Ultrasonic. Ultrasonic is used for measuring the distance between ground and UAV to avoid crash. IMU, consists Gyro and accelerometer, is used to monitor the flying status, including roll, Pitch, yaw, and acceleration of UAV. GPS is used for positioning the UAV in global coordinate.

Lidar sensor will be added in future works. Most mechanical Lidars rotate with a very fast speed and measures distance to surrounding area. It can reconstruct the environment with point cloud [32]. Means et al shows that Lidar can be mounted on airborne to predict forest standings [33]. [29] [34] present that a Lidar based UAV system detect tree and find a routine to survey.

3.5 Software

This Section presents the software that is used over the whole thesis, including the simulation software and UAV controlling software.

3.5.1 Matlab

Matlab is used as the software for constructing sensors’ model, partition the task area and optimize the positions of the sensors. It has many tool boxes and packages that can be used for simulation and data processing. It is convenient to program with mathematical expressions. Most importantly, it is strong in matrix calculations. In this thesis, the coverage strength model for each sensor is in terms of matrix. The number of sensors, the length and width determines the dimensions of matrix. Mostly, the matrix is three dimensional. The coverage strength matrix requires large amount of calculation. In addition, the partition of the task area and optimization also have heavy computing load. Matlab is used for this part of simulation.

3.5.2 Python

Python is used as the software for all UAV related programs, including the simulation of the flight software and real flight control. Python has many different packages for different purposes, from simplest calculation to model construction. Its coding scheme is in order and convenient. In addition, it has cython module which can compile codes between C++ and python.

The reason to use python is manifold. First, a well-developed Ardrone package is on python. This package has protocols to transmit all sensor’s data. It also allows ground station to remotely control the UAV’s movement. Second, python has socket module which allows users to transmit data. In this thesis, the real-time image and video taken by the onboard camera can be transmitted to ground station via Wifi. Third, it has Dronekit package. Dronekit package is developed for UAVs using Pixhawk as the flight controller. It establishes a communication channel between Pixhawk and the onboard
computer. The onboard computer can send UAV controlling commands to Pixhawk for autonomous purposes. Fourth, python is compatible with image processing software, OpenCV. In this thesis, OpenCV is used for capturing the image, process the raw data and store the image. In future work, OpenCV is used for real-time image processing for applications of autonomous UAVs.

3.5.3 Dronekit

Dronekit consists of two parts, simulation of the movement of the UAV and real control of the UAV. As mentioned before, this package is designed for UAVs that equip Pixhawk as the major flight controller. Mission planner is a software that can track drones with Pixhawk and radio transmitter. In this thesis, Ardrone uses its own onboard computer and flight controller. DJI uses Pixhawk. Therefore, Dronekit is only used for DJI F450.

a) Simulation tool (Dronekit-SITL)

SITL is the abbreviation of simulation tool. The simulation part of Dronekit is mainly used for testing whether the control algorithm is appropriate. The reason for simulating the drone’s movement is that for outdoor UAVs, the speed is very fast. To avoid crashes or any unsafety behavior, it is very important to simulate the drone’s behavior before test flight. This simulation software allows users to load google map at where they want to do the tests. The figure 3.4 shows the environment of the testing area.

The simulation tool loads the parameters of the UAV used in the experiment. For example, the firmware version of the Pixhawk, the power output of the brushless rotors and battery information. By giving the corresponding commands that control the movement of the drone, the UAV in the simulation software follows the commands and move accordingly.

Also, SITL allows users to change the parameter of the environment, such as wind speed and directions. It affects the movement of the UAV in the simulation. Although it might not be realistic, it can be used to test the robustness of the controlling algorithm.
b) Real flight control

Regardless of the simulation part, Dronekit also can access to Pixhawk with onboard computer. The controlling algorithm that is tested with the simulation tool is loaded on the onboard computer and the UAV flies autonomously based on the code. To establish the link between onboard computer and flight controller, a necessary python package Mavlink is required to be installed.

c) Mission Planner

Mission Planner is used to track the UAV’s position and shows the trajectory of the UAV. It reads the GPS readings send by the radio transmitter on Pixhawk real-time and displays the information on the map. It can also be used to monitor other information, such as acceleration, roll-pitch-yaw of the UAV, and safety switch of the UAV. Mission planner can perform some autonomous missions. By selecting points on the map, and loading them to the controller, the UAV can fly to the positions.

There is another important feature of using this software. Since Pixhawk is a common used controller, it can be also used for controlling other types of mobile vehicles, for example, fixed wings, hexa-copter and some ground vehicles. Different firmware needs to be installed when different types of vehicles are used. Mission Planner can be used to download the required firmware to Pixhawk.

3.5.4 OpenCV

The software that is used to access the camera, take images and process image data in this experiment is OpenCV. OpenCV is developed by Intel and provides coding platform for computer and robotic vision. It is compatible with C++, python and Java, and can compile between all three coding programs. In this experiment, OpenCV is used for taking image and videos, and processing raw image data for research purposes.
Chapter IV
Experiment

This chapter talks about the experiment procedure. The details, including the size of the test area, the coverage of the sensor and other information are also presented. This chapter consists of two major sections, which are the two experiments conducted to prove the feasibility of applying coverage strength model on the aerial coverage problem. The two experiments are conducted separately for indoor and outdoor cases.

4.1 Indoor Coverage Experiment

The purpose of indoor experiment is to test feasibility of applying coverage strength model to UAVs by comparing the simulation result of total coverage of the target area to the real experimental result of total coverage using UAV.

4.1.1 Test Area

The test area is a 16.5 m × 8m enclosed room. The UAV is deployed at all necessary positions to cover the test area. Since the deployment method is designed for cover the area, it is a 2D problems. The height of the room is not considered in this thesis. There are two pillars in the room. In this thesis, the pillars are not considered as the obstacles. However, in [28], it shows that the coverage model can be also applied when obstacle exits. Figure 4.1 shows the test area.

![Figure 4.1 Indoor Test Area](image)
4.1.2 Camera Coverage

To obtain the best total coverage, the single sensor coverage is necessary to be known. Figure 4.2 shows the photo taken by the camera. The coverage area is measured based on the image taken by the camera. The distance between camera and the measuring point is 3.9 meters and the horizontal length of the image is 5.2 meters.

Therefore, the FOV and the coverage radius of the camera can be calculated based on equation 3.2. The FOV is 68.2 degrees and the coverage radius of the camera is 4.72 meters. The number of sensors that needs to be used to cover the area can be calculated. H and W are the dimensions of the room.

\[
n = \frac{H \times W}{\frac{FOV \times 2}{360} \times \pi \times r^2}
\]  

(4.1)

In this case, the n that is needed to cover the whole area is 10, which means ten sensors is enough to cover the whole area. However, if ten sensors are used, there are overlapping coverage, which means some area are covered by two or more sensors at the same time. This reduces the efficiency of applying coverage model on UAV. The reason to apply coverage model is to use the minimum number of sensors to cover the most area. Several experiments are conducted for testing when different number of sensors are deployed.

4.1.3 Deployment

The first issue that needs to be solved is to define a local coordinate for the room. As shown in figure, the X and Y axis is along the two side walls. 0 degree is set to be pointing at the positive direction of the Y axis.

Since in this thesis, the indoor UAV tracking is not considered, the UAV’s position is not trackable. Therefore, the indoor experiment for the UAV is not autonomous. The
feature points for UAV deployment are marked. The UAV takes off at the featured positions.

Different number of deploying points are used for experimental purposes in this thesis. The deployment of the UAVs changes when numbers of the UAV are different. Also, the total maximum coverage is different.

4.2 Outdoor Coverage Experiment

The purpose for outdoor experiment is to find the deployment based on the coverage strength model and convert the deployment positions to be a possible surveillance route for one UAV. Comparing to normal sweeping method, finding feature points with coverage strength model reduces the flight time but still reaches a high coverage.

4.2.1 Test area

The test area of outdoor aerial surveillance experiment is shown in figure. It is a 100m * 60m soccer field. The positive direction of x and y are labeled in the figure. 0 degree is also set to be the positive direction of y axis. The x, y coordinate is converted to the global coordinate for deploying the UAV.

![Figure 4.3 Test field](image)

There is one other important problem in this experiment. Because the test field is very close to a highway, there must be a method to immediately shut down the UAV since it is flying autonomously. On most remote controllers, there is a switch to kill the UAV. In this experiment, a method that is based on GPS is used. If the UAV flies out of the boundary for 10%, the UAV shuts down.

4.2.2 Deployment

In outdoor deployment, because most cameras have very far vision, the coverage range of the camera depends on how clear the image should be. In this experiment, the coverage range of the UAV is 20 meters with a FOV of 66.7degrees. 10 deploying
positions are selected for deployment of UAV and these ten points are connected as a path for the UAV to survey. Also, in the experiment, the time that UAV needs to survey the whole area with sweeping method is presented.

4.2.3 Flying Control Algorithm

The flying control algorithm is essential in this experiment. Since the UAV needs to reach target position accurately, the controlling algorithm is GPS based position control. In this thesis, a PID controlling algorithm is applied to control the movement of the UAV in four directions, which are X, Y, altitude and view angle or Yaw. Since the DJI F450 is dependent of Pixhawk flight controller, it is necessary to know how Pixhawk is working.

Pixhawk flight controller can receive command from remote controller. For most remote controllers, there are two joysticks that are controlling the movement of the UAV. While controlling the UAV, the movement of the joysticks determines the movement of the drone.

![Remote Controller](image)

Figure 4.4 Remote Controller

The remote controller is communicating with Pixhawk via radio frequencies. From Mission Planner, it shows that communication band is divided into eight channels, and each of them responds for a button on the remote controller. While moving the joystick, the value of four channels are varying, which are channel 1 to 4. Table shows the relationship between joystick movement, channel override value and movement of the UAV.
The error in this system is the difference between set position and current position of the system. Therefore, the integral term of the system is the accumulative of $E(t)$. The differential term of the system is the difference between errors, which is instant speed of the UAV.

$$E(t) = S - U(t) \quad (4.2)$$

$$\int E(t) = \sum_{t=0}^{t} [S - U(t)] \quad (4.3)$$

$$\nabla E(t) = [S - U(t)] - [S - U(t - 1)] = V(t) \quad (4.4)$$

### 8 Parameters of the UAV System

- $E(t) = \{E_x(t), E_y(t), E_z(t), E_\theta(t)\}$: error of the system in four axis
- $S(x, y, z, \theta)$: Set position or Final Deployment
- $U(t) = \{X(t), Y(t), Z(t), \theta(t)\}$: Current position of UAV
- $V(t) = \{V_x(t), V_y(t), V_z(t), \dot{\theta}(t)\}$: Speed and angular speed
- $P_I, P_d, P_k$: PID parameters of the system
- $C_1, C_2, C_3, C_4$: Override Valus in four Channels

The whole system is position based PID control. The input and output should be GPS coordinates of the UAVs. The position information of the UAV should be converted to the Channled Override Values above to control the movement of the UAV, until the UAV reaches the pre-defined position.

$$C_i = P_k * E(t) + P_d * V(t) + P_i * \int E(t) \quad (4.5)$$

This channel override value is calculated by the onboard computer and sent to Pixhawk Flight Controller to drive the UAV. However, since the mathematical model of the UAV is unknown. The PID parameters of the UAV are adjusted based on the real flight and simulation results. The figure shows one example of the flight result with PID controller on Z-Axis. The target amplitude is set to be at 10m and 20m.
The relationship between input (channel override value) and output (position of UAV) is estimated from statistical data and real flight performance of uav because the mathematical model is unknown. It is hard to obtain an accurate mathematical model with system identification method because many unpredictable factors, such as wind, affect the performance of the UAV. Also, the number of experiments are limited due to the regulations of Transport of Canada and hardware problems (broken parts).

Therefore, the PID tuning is adjusted during each flight. However, some general rules are established during the experiments such as table 4.1. Also, it is known that the speed of the UAV is faster when the channel override value is closer to the extreme value (e.g. 993 or 2016 for channel 3) and slower when value is close to medium (e.g. 1504 for channel 3). The rules are applied for tuning the PID.

![Figure 4.6 Altitude Simulation](image)

In figure 4.6, the X-axis is the number of clock cycles and Y-axis the altitude that the UAV is at with respect to clock cycle time. The two subplots show the difference between position control performances when different PID settings is used.

<table>
<thead>
<tr>
<th>Table 4.2 PID Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure</td>
</tr>
<tr>
<td>$P_l, P_d, P_k$</td>
</tr>
<tr>
<td>Settling Time</td>
</tr>
<tr>
<td>Overshoot</td>
</tr>
</tbody>
</table>

Since the drone is used for outside aerial surveillance, the outdoor environmental condition always changes. For example, the wind speed can affect the flight of the UAV. Therefore, even the channel override values may be the same, the movement of the
UAV could be different. This gives some difficulty in tuning the PID controller. However, after testing, there are some general characteristics that can be used to tune the PID controllers. Table 4.2 shows the performance when different PID values are applied.

1. $P_k$ mainly determines the speed and movement. If $P_k$ is larger, the UAV reaches the set position faster. However, the overshoot is also larger.

2. The value of $P_i$ cannot be very large because integral is mainly used for countering shifting. For example, the drone remains at its position when the wind is blowing the drone.

3. $P_d$ affects the settling time after reaching the position. If $P_d$ is larger, the settling time is smaller.

Figure 4.7 shows the controlling scheme.

![Figure 4.7 Controller Scheme](image)

**4.3 Conclusion**

The experiment consists of two parts, indoor and outdoor. For these two parts, the purposes are not the same. The purpose of indoor aerial surveillance is to compare the total coverages between theoretical deployment using coverage strength model and real deployment based on the simulation result to judge whether coverage model can be applied to moving UAV. The purpose of outdoor aerial surveillance is to find the deployment based on the coverage strength model and design a surveillance path based on the deployment for one UAV to survey the target area. If the result proves the idea of using coverage model is applicable, many other application can be generated as well.

The other important is the autonomous UAV controlling algorithm. It is position based PID control. It is a good selection when mathematical model of the UAV is unknown, especially when the condition is always changing outdoor.

During the experiment, it is found out that the tuning of PID for the flight controller is very important. To maintain a stable controlling of the UAV, many factors have to be
considered. However, because the mathematical model of the UAV is unknown, the PID tuning can only be modified based on the performance.
Chapter V
Results and Analysis

This chapter discusses the result obtained from the experiment. Since the real deployment is based on the simulation result, the comparison between real result and simulation is presented as well. The results are presented separately for indoor and outdoor experiments.

5.1 Indoor Experiment Results

This section shows the results of indoor UAV deployment. The UAV is deployed at different positions and take images. Since indoor tracking of the UAV is not included in this thesis, the deployment of the UAV is remote controlled. The number of deploying positions are set to be 6, 7, 8 and 9. From previous chapters, it is known that the number of sensors determines percentage of total coverage.

The deployment of seven sensors is presented here as the example for analysis. Table shows the seven deploying positions and corresponding angles in the room. Since the UAV is not trackable indoor, the UAV is deployed remotely at these positions. Figure shows the deployment for all positions and figure shows coverage percentage over the whole simulation. From the figure, it is known that using seven sensors can cover almost 70% percent of the target area.

Table 5.1 Indoor Deployment Positions

<table>
<thead>
<tr>
<th>Sensor No</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(Meter)</td>
<td>4.13</td>
<td>7.48</td>
<td>11.67</td>
<td>7.33</td>
<td>4.81</td>
<td>6.13</td>
<td>16.5</td>
</tr>
<tr>
<td>Y(Meter)</td>
<td>5.0</td>
<td>4.22</td>
<td>3.35</td>
<td>4.36</td>
<td>5.99</td>
<td>3.16</td>
<td>8.0</td>
</tr>
<tr>
<td>θ(Degree)</td>
<td>169</td>
<td>290</td>
<td>234</td>
<td>194</td>
<td>90</td>
<td>-22 or 338</td>
<td>131</td>
</tr>
</tbody>
</table>
Figure 5.1 (a) (b) Deployment and Percentage of Coverage

Figure 5.1 (a) and (b) show the UAV that is deployed at point (4.13, 5.0) and the image taken by the onboard camera. The coverage of real experiment is measured and calculated based on the photos taken by the onboard camera.

Table shows the maximum total coverage from theory and average coverage from experiments. Each set of experiments is done ten times. There are two conclusions that can be drawn from this table. First, the total coverage increases when more sensors are used. Second, the difference between theoretical deployment and actual deployment decreases when more sensors are used.
Table 5.2 Coverage

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>theoretical</td>
<td>0.59</td>
<td>0.678</td>
<td>0.714</td>
<td>0.81</td>
</tr>
<tr>
<td>Actual (10 Experiments)</td>
<td>0.54</td>
<td>0.64</td>
<td>0.689</td>
<td>0.789</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.91</td>
<td>0.94</td>
<td>0.96</td>
<td>0.974</td>
</tr>
</tbody>
</table>

However, there is also drawbacks using coverage strength model. When more sensors are used, the total coverage does not increase much. In theory, ten sensors are enough to cover the whole room. However, after simulation, the total coverage with ten sensors can only reach 82%. If more sensors are used, the total coverage does not exceed 82% and the overlapping coverage shows up. This disobeys the rule of using less sensors to cover more area. Therefore, there should be a method which can be used to find the best number of sensors.

The reason that the model does not predict the result could be explained from three perspectives.

First, sensor’s geometric shape of coverage restricts the total coverage. The task area is rectangular and the sensing range is circular sector. There should be some gaps if overlapping coverage is not allowed. To solve this problem, sensors with smaller sensing region can be used to fill in the gaps.

Second, as mentioned in chapter 2, although FOV and view angles are two parameters that are affecting the performance of the coverage, they are integrated with one equation. Some properties of these two parameters are omitted when the model is being constructed.

5.2 Outdoor Experiment Results

The result of outdoor experiment is shown in this section. Figure shows mapping the simulation result to the actual map. Ideally, ten sensors are deployed at these positions. In real, since the number of the UAVs is not enough, the UAV goes to all ten points and takes images.
Figure 5.3 Sensor Mapping

The figure shows the trajectory of the UAV during one flight with monitoring function from Mission Planner and the image is taken when the drone is hovering at one of the designed positions. Same as indoor experiment, the total coverage is measured based on the picture taken by the camera at 10th point.

Figure 5.4 (a) (b) Trajectory of UAV and Image at feature point

For normal path planning method, the UAV keeps moving and taking images all the time. In the experiment, the UAV reaches each position and hover at the position for 2 seconds for taking pictures. Table show the ten positions that is obtained by the simulation result and the corresponding real GPS coordinates. The 10 GPS coordinates are saved as a text file. The onboard computer of UAV reads all positions from the file and stores them as a matrix in python.
Table 5.3 Outdoor Deployment Positions

<table>
<thead>
<tr>
<th>Sensor</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>(49,27)</td>
<td>(94,57)</td>
<td>(77,24)</td>
<td>(70,23)</td>
<td>(17,25)</td>
<td>(65,37)</td>
<td>(70,34)</td>
<td>(21,54)</td>
<td>(24,38)</td>
<td>(84,33)</td>
</tr>
<tr>
<td>Global</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
<td>42.27</td>
</tr>
<tr>
<td></td>
<td>8493,</td>
<td>8003,</td>
<td>8299,</td>
<td>8327,</td>
<td>8337,</td>
<td>8295,</td>
<td>8327,</td>
<td>8672,</td>
<td>8694,</td>
<td>8131,</td>
</tr>
<tr>
<td></td>
<td>9273</td>
<td>9136</td>
<td>9445</td>
<td>9222</td>
<td>9233</td>
<td>9271</td>
<td>8897</td>
<td>9053</td>
<td>83.06</td>
<td>9389</td>
</tr>
<tr>
<td>Angle</td>
<td>130</td>
<td>295</td>
<td>261</td>
<td>175</td>
<td>179</td>
<td>13</td>
<td>102</td>
<td>244</td>
<td>82</td>
<td>339</td>
</tr>
</tbody>
</table>

With ten sensors, theoretical total coverage of the area is 78 percent. The table shows the time that is required for a UAV to survey 78 percent of the target area with two different methods, using coverage strength model and normal sweeping method for five experiments. For same set, the experiments using two methods are conducted within 20 minutes so that the outdoor conditions have the smallest effect on results.

Table 5.4 Outdoor Coverage

<table>
<thead>
<tr>
<th>Method</th>
<th>Coverage Strength Model (10 points)</th>
<th>Coverage Strength model (11 points)</th>
<th>Sweeping method (90%-100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>117s (73%)</td>
<td>131s (76%)</td>
<td>167s</td>
</tr>
<tr>
<td>Experiment 2</td>
<td>124s (72%)</td>
<td>137s (78%)</td>
<td>171s</td>
</tr>
<tr>
<td>Experiment 3</td>
<td>110s (73%)</td>
<td>141s (77%)</td>
<td>170s</td>
</tr>
<tr>
<td>Experiment 4</td>
<td>110s (77%)</td>
<td>124s (75%)</td>
<td>181s</td>
</tr>
<tr>
<td>Experiment 5</td>
<td>127s (74%)</td>
<td>131s (79%)</td>
<td>187s</td>
</tr>
<tr>
<td>Average</td>
<td>117.6s (73.8%)</td>
<td>132.8s (77%)</td>
<td>175.2s</td>
</tr>
</tbody>
</table>

From table 5.4, it can be concluded that using coverage strength model can guarantee a good coverage. However, if the surveillance needs to be very detail, the sweeping method is the choice at this stage because as mentioned in 5.1, more deploying positions does not give more coverage.

In order to achieve better performance with coverage strength model, there are two problems that needs to be solved. First, in this problem, the path is not optimized. The path is generated by connecting the deploying points in order. If the path can be optimized, the flight time could be shorter. Second, the gaps due to the geometric shape
should be solved. In section 5.1, a method of deploying sensor with smaller sensing region is suggested. For single UAV in this case, the onboard camera can change the sensing region by automatically changing the zoom of the camera.

5.3 Analysis

It is known from previous sections that using coverage model can have a good coverage but not full coverage of the area. For example, in indoor deployment experiments, the coverage can only reach a maximum of 82% in ideal situations. For outdoor experiments, the coverage saturates at 77% even though more deploying points are added. The reason is also presented in previous section is that non-overlapping coverage restricts the total coverage since the geometric shape of sensor is not similar to the test area.

Since coverage model generates deploying points, for UAV, it is very clear where to go. Comparing to the normal sweeping method, using coverage model shortens the flight time but sacrifices the coverage performance. The flight path can be optimized so that the UAV can finish the survey with less time and still maintain the same coverage as long as the deploying positions remains the same. To solve the problem, the focus should be on the improvement of coverage strength model.
Chapter VI
Conclusion and Future Work

6.1 Conclusion

The coverage strength model is to convert the geometric shape of the sensor coverage to a mathematical expression. It can be used for any directional sensors up to now. The sensors’ positions are optimized to fulfill the task. This can be used on many autonomous progresses when deployment is not accurate, the robot can adjust its position and orientation based on all kinds of sensors. This thesis intends to prove that coverage strength model and corresponding optimization method can be also used on moving robots equipped with sensors, such as UAVs.

There are two types of experiments conducted. Indoor aerial surveillance is to compare the difference between simulation result to experimental results to show the difference. Outdoor aerial surveillance is to show that the feature points based on coverage model can increase the efficiency for one UAV.

The experimental results show the advantage of applying coverage model. The difference between total coverage in simulation and total coverage in real deployment is very small. If the UAV is indoor trackable, and it can reach the position and orientation autonomously, the difference is even smaller. In addition, comparing to normal path planning method for UAV, using coverage model to find feature points can reduce the surveying time. The total coverage remains very high.

In future work, the coverage strength model is expanded to different sensors. Also, more robots will be used for experiment and more application will be done.

6.2 Improvement

1. Real time UAV deployment

In the experiment, the UAVs are deployed after the optimization is done until the simulation gives the best positions. The original idea was to figure out the UAVs’ positions at the very beginning, and UAVs move when the deployment are being optimized. In other words, the UAV should move as the simulation continuing, not only move at the end. This is the difference between online and offline deployment. The reason that causes this problem is that the optimization and coverage model requires a lot of calculation. The time delay between optimization and real UAV movement is very huge, which means it is very hard to update the optimized position and transmit the information to the UAV at the same time. To solve this problem, there are two methods. First method is to optimize the optimization method and coverage strength model, so
that the processing load of this part reduces. The other method is to distribute the process task to several computers. Although it is called distributed optimization method, the information of all sensors is still known by one computer. The optimization is on one computer. This increases the computing load. If each sensor can optimize its position individually, this problem can be also solved.

2. Precise UAV control algorithm and accuracy

The second part that can be improved is the control algorithm and accuracy. In this thesis, the controlling algorithm is position based PID since the main goal is to control the drone to pre-defined positions. The reason to use PID is that the model of the UAV is unknown. However, during the outdoor experiment, some issue about drone control shows up. For example, the UAV is not stable when the wind is strong. Also, the accuracy of GPS restricts the accuracy of the control. The improvement method is to construct a mathematical model for the UAV. Advanced controllers can be used if the model of the UAV is known. Also, the data from IMU, including the accelerometer and gyroscope should also be considered when controlling the drone.

3. Indoor position tracking

For indoor experiment, the UAV is not autonomous because the position of UAV is not trackable. However, there are several methods that can track the drone indoor. For example, a camera network can be used build a camera coordinate and track the drone. It is also possible to track the drone based on Wifi signal strength and IMU of the drone. Thus, for future work, an indoor tracking algorithm can be used for indoor surveillances.
Appendix A

Equipment

A.1 Onboard Computer

The onboard computer is the one attached on the DJI f450, since Parrot Ardrone has its own onboard computer. It can collect information from sensors, analyze raw data and send movement commands to the flight controller.

A.2 Flight Controller

The flight controller used on DJI F450 is Pixhwak 2.0, which is a mobile vehicle controller. It can generate PWM signal and transmit the signal to the rotor to drive the UAV. It can also read raw data from different sensors and send them to onboard computer. It has IMU which can be used to monitor the drone’s flight status.
A.3 Camera

There are two cameras used in the experiments.

A.3.1 Logitech Web camera

The Logitech web camera C910 is the camera mounted on the DJI F450. It is used to obtain the photos. The onboard computer saves the photos, which are used to estimate the results. The resolution of the camera is 1080P, which means 1920 * 1080.

A.3.2 Parrot Ardrone Camera

Parrot Ardrone Camera is installed on the Ardrone Frame. It can obtain photos or videos. The Onboard Computer of Ardrone directly sends video stream or photo to ground stations via Wifi.

A.4 GPS

The GPS module is necessary in this thesis. It is used to track the UAV’s position.
A.5 DJI Flamewheel F450

DJI Flamewheel F450 is the UAV frame used for outdoor aerial surveillance in this thesis. It is a quad-copter as figure 3.1.

A.6 Parrot Ardrone 2.0

The Parrot Ardrone 2.0 is the UAV used in the indoor coverage experiment.

A.7 General Purposed Computer

The figure shows the general purposed computer used in the experiments.
Appendix B

Codes

1. PID Position Controller for Ardrone

```python
import dronekit as dk
import numpy as np
import os
from matplotlib import pylab as pl
import decimal
#
import cv2
import RPi.GPIO as GPIO

vehicle = dk.connect('/dev/ttyS0', baud=57600, heartbeat_timeout=20)

x1 = 1498
x2 = 1515
x4 = 1499
aalt = []

allposition = []
cap0 = cv2.VideoCapture(0)

GPIO.setup(TRIG,GPIO.OUT)

TRIG = 23
ECHO = 24
print "Distance Measurement In Progress"

GPIO.setup(TRIG,GPIO.OUT)
```
GPIO.setup(ECHO, GPIO.IN)

GPIO.output(TRIG, False)
print "Waiting For Sensor To Settle"
time.sleep(0.1)

# Armed and Takeoff
print "Arming motors"
vehicle.mode = dk.VehicleMode("STABILIZE")
print vehicle.mode
vehicle.armed = True
time.sleep(2)
for h in range(0, 8000):
    print vehicle.armed
time.sleep(3)
print "Taking off"

# Taking off PID Controller

original_altitude = sonarlib.distance(TRIG, ECHO)
last_altitude = sonarlib.distance(TRIG, ECHO)
integral_altitude = 0
altitude = sonarlib.distance(TRIG, ECHO)
print sonarlib.distance(TRIG, ECHO)

set_altitude = 0.5

# Coefficient of PID
Kp = 10
Ki = 0.5
Kd = 0.5
error = set_altitude - altitude

g = 1300
while True:
    vehicle.channels.overrides = {'1': x1, '2': x2, '3': g, '4': x4}
    print vehicle.channels.overrides
    takeoff_alt = sonarlib.distance(TRIG, ECHO)
    print takeoff_alt
    if takeoff_alt > 0.1:
        g = g - 4
        break
    g = g + 3
    print g
    time.sleep(0.1)

# Position PID for TAKING OFF
set_lat = vehicle.location.global_frame.lat
set_lon = vehicle.location.global_frame.lon
set_latm = round((set_lat * 1000 - np.fix(set_lat * 1000)) * 100, 2)
set_lonm = round((set_lon * 1000 - np.fix(set_lon * 1000)) * 100, 2)
now_lat = vehicle.location.global_frame.lat
now_latm = round((now_lat * 100 - np.fix(now_lat * 100)) * 1000, 2)
now_lon = vehicle.location.global_frame.lon
now_lonm = round((now_lon * 100 - np.fix(now_lon * 100)) * 1000, 2)
original_latm = round((now_lat * 100 - np.fix(now_lat * 100)) * 1000, 2)
original_lonm = round((now_lon * 100 - np.fix(now_lon * 100)) * 1000, 2)
lat_dis = set_latm - now_latm
lon_dis = set_lonm - now_lonm
last_lat = 0
last_lon = 0
integral_lat = 0
integral_lon = 0

# Coefficient of Position PID Controller during Taking Off
Kp_take = 1.5
Ki_take = 2
Kd_take = 0

for i in range(0, 200000):
    # Position PID control for TAKING OFF
    now_lat = vehicle.location.global_frame.lat
    now_lon = vehicle.location.global_frame.lon
    now_latm = round((now_lat * 1000 - np.fix(now_lat * 1000)) * 100, 2)
    now_lonm = round((now_lon * 1000 - np.fix(now_lon * 1000)) * 100, 2)
    error_lat = set_latm - now_latm
    error_lon = set_lonm - now_lonm
    delta_lat = error_lat - last_lat
    delta_lon = error_lon - last_lon
    last_lat = error_lat
    last_lon = error_lon
    if error_lat > 15 or error_lon < -15 or error_lon > 15 or error_lon < -15:
        vehicle.mode = dk.VehicleMode("LAND")
        time.sleep(0.5)
        exit()

    if i % 1500 == 0:
        integral_lat = error_lat + integral_lat
        integral_lon = error_lon + integral_lon

    b = 1515 - Kp_take * error_lat - Ki_take * integral_lat - Kd_take * delta_lat # for ch2
if b < 992:
    b = 992
elif b > 2014:
    b = 2014
#

if c < 991:
    c = 991
elif c > 2010:
    c = 2010

##########################################################################
#
  Altitude PID controller
altitude = sonarlib.distance(TRIG, ECHO)
error = set_altitude - altitude
delta_altitude = altitude - last_altitude
last_altitude = altitude
if i % 15000 == 0:
    integral_altitude = error + integral_altitude
elif i % 100000 == 0:
    integral_altitude = 0
a = g + Kp * error + Kd * delta_altitude + Ki * integral_altitude # change Kd to be -
if a < 995:
    a = 995
elif a > 2017:
    a = 2017
vehicle.channels.overrides = {'1': c, '2': b, '3': a, '4': x4} # wind speed is 10, dir is south 1625
print vehicle.channels.overrides
print altitude
if altitude > 4.5:
    vehicle.mode = dk.VehicleMode("LAND")
time.sleep(0.5)
exit()
print vehicle.location.global_frame

aalt = np.append(aalt, altitude)
if i >= 110000:
    if aalt[i] == aalt[i-500]:
        break
time.sleep(0.2)

# Mission Start
print "start go to target location"
with open('positions.txt','r') as f:
    for line in f:
        allposition.append(map(float,line.split(',')))
n = 0
while True:
    if n > 2:
        break
    alatdis = []
alondis = []
    set_position = allposition[n] # set position
    set_lat = set_position[0] # set latitude
    set_latm = round((set_lat * 10 - np.fix(set_lat * 10)) * 10000, 2) # convert latitude in meters
    now_position = (vehicle.location.global_frame.lat, vehicle.location.global_frame.lon)
    now_lat = vehicle.location.global_frame.lat
    now_latm = round((now_lat * 10 - np.fix(now_lat * 10)) * 10000, 2) # convert now latitude in meter
    original_latm = round((now_lat * 10 - np.fix(now_lat * 10)) * 10000, 2)
    lat_dis = set_latm - now_latm
    set_lon = set_position[1] # set longitude
    set_lonm = round((set_lon * 10 - np.fix(set_lon * 10)) * 10000, 2) # convert longitude in meters
now_lon = vehicle.location.global_frame.lon
now_lonm = round((now_lon * 10 - np.fix(now_lon * 10)) * 10000, 2) # convert now longitude in meter
original_lonm = round((now_lon * 10 - np.fix(now_lon * 10)) * 100, 2)
lon_dis = set_lonm - now_lonm

last_lat = now_lat # last latitude for Kd
last_lon = now_lon # last longitude for Kd

integral_lat = 0 # integral latitude for Ki
integral_lon = 0 # integral longitude for Ki

# Keep Longitude when move latitude
keep_lon = vehicle.location.global_frame.lon
keep_lonm = round((keep_lon * 10 - np.fix(keep_lon * 10)) * 10000, 2)

# Kp, Ki and Kd for the Latitude Movement
Kp_lat = 50
Ki_lat = 0.5
Kd_lat = 15
Ki_lat1 = 0

# Kp, Ki, and Kd for keep latitude and longitude
Kp_keep = 5
Ki_keep = 0.5
Kd_keep = 0

alat = []
alon = []

# Latitude PID control
for j in range(300000):
# Keep Altitude PID
altitude = sonarlib.distance(TRIG, ECHO)
error = set_altitude - altitude
delta_altitude = altitude - last_altitude
last_altitude = altitude
if j % 15000 == 0:
    integral_altitude = error + integral_altitude
elif j % 100000 == 0:
    integral_altitude = 0
a = g + Kp * error + Kd * delta_altitude + Ki * integral_altitude
if a < 995:
a = 995
elif a > 2017:
a = 2017

# Keep Longitude PID
now_lon = vehicle.location.global_frame.lon
now_lonm = round((now_lon * 10 - np.fix(now_lon * 10)) * 10000, 2)
error_lon = keep_lon - now_lonm
delta_lon = now_lonm - last_lon
last_lon = now_lonm
if j % 2000 == 0:
    integral_lon = error_lon + integral_lon
    c = round(1498 + Kp_keep * error_lon + Ki_keep * integral_lon + Kd_keep * delta_lon, 1)
    if c < 991:
c = 991
elif c > 2010:
c = 2010

# Latitude Position PID
now_lat = vehicle.location.global_frame.lat
now_latm = round((now_lat * 10 - np.fix(now_lat * 10)) * 10000, 2)
alat = np.append(alat, now_latm)
lat_avg = now_latm
if j >= 4:

error_lat = set_latm - lat_avg
delta_lat = lat_avg - last_lat
last_lat = lat_avg
if j % 1000 == 0:
    integral_lat = error_lat + integral_lat
b = round(1515 - Kp_lat * error_lat - Ki_lat * integral_lat - Kd_lat * delta_lat, 1) # it was -
if b < 992:
    b = 992
elif b > 2010:
    b = 2010
vehicle.channels.overrides = {'1': c, '2': b, '3': a, '4': x4}
lat_dis = set_latm - lat_avg
print "lat_dis is %s" % lat_dis
print vehicle.channels.overrides
alatdis = np.append(alatdis, lat_dis)
if j > 100000:
    if np.abs(alatdis[j]) < 3 and np.abs(alatdis[j] - alatdis[j - 1000]) < 0.5:
        print "arrive set latitude"
break

elif j > 30000 and np.abs(lat_dis) > 50:
    vehicle.mode = dk.VehicleMode("LAND")

if lat_avg > 7959.10 or lat_avg < 7779.60 or now_lonm < -6922.30 or now_lonm > -6833.10:
    while True:
        vehicle.channels.overrides = {'1': x1, '2': x2, '3': 995, '4': x4}
        vehicle.mode = dk.VehicleMode("LAND")

        time.sleep(0.5)
        exit()
        time.sleep(0.2)
        # Keep Latitude
        keep_lat = vehicle.location.global_frame.lat
        keep_latm = round((keep_lat * 10 - np.fix(keep_lat * 10)) * 10000, 2)
        last_lat = keep_latm
        integral_lat = 0

        # Longitude PID control
        for k in range(0, 300000):
            # Keep Altitude PID
            altitude = sonarlib.distance(TRIG, ECHO)
            error = set_altitude - altitude
            delta_altitude = altitude - last_altitude
            last_altitude = altitude

            if k % 15000 == 0:
                integral_altitude = error + integral_altitude
            elif k % 100000 == 0:
                integral_altitude = 0

                a = g + Kp * error + Kd * delta_altitude + Ki * integral_altitude
                if a < 995:
                    a = 995
elif a > 2017:
    a = 2017
    # Keep Latitude PID
    now_lat = vehicle.location.global_frame.lat
    now_latm = round((now_lat * 10 - np.fix(now_lat * 10)) * 10000, 2)
    error_lat = keep_latm - now_latm
    delta_lat = now_latm - last_lat
    last_lat = now_latm
    
    if k % 2000 == 0:
        integral_lat = error_lat + integral_lat
        b = round(1515 - Kp_keep * error_lat - Ki_keep * integral_lat - Kd_keep * delta_lat, 1)
        if b < 992:
            b = 992
        elif b > 2014:
            b = 2014
    # Longitude PID
    now_lon = vehicle.location.global_frame.lon
    now_lonm = round((now_lon * 10 - np.fix(now_lon * 10)) * 10000, 2)
    alon = np.append(alon, now_lonm)
    lon_avg = now_lonm
    if k >= 4:
        lon_avg = (alon[k - 4] + alon[k - 3] + alon[k - 2] + alon[k - 1] + alon[k]) / 5
        # print "lon avg is %s" % lon_avg
    error_lon = set_lonm - lon_avg
    delta_lon = lon_avg - last_lon
    last_lon = lon_avg
    if k % 2000 == 0:
        integral_lon = error_lon + integral_lon
        c = round(1498 + Kp_lat * error_lon + Ki_lat * integral_lon + Kd_lat * delta_lon, 1)  # was +
if c < 991:
c = 991
elif c > 2010:
c = 2010
vehicle.channels.overrides = {'1': c, '2': b, '3': a, '4': x4}
lon_dis = set_lonm - lon_avg
print "lon_dis is %s" % lon_dis
print vehicle.channels.overrides
alondis = np.append(alondis, lon_dis)
if k > 100000:
  if np.abs(alondis[k]) < 4 and np.abs(alondis[k] - alondis[k - 1000]) < 0.5:
    print "arrive set longitude"
cap = cv2.VideoCapture(0)
cap.set(4, 800)
cap.set(3, 600)
for l in range(0,30):
  vehicle.channels.overrides = {'1': x1, '2': x2, '3': g, '4': x4}
  ret, frame = cap.read()
  cv2.imwrite(os.path.join("/home/pi/", 'position%d.png') % n, frame)
  cv2.imshow('frame', frame)
cap.release()
cv2.destroyAllWindows()
break

elif k > 30000 and np.abs(lon_dis) > 50:
  vehicle.mode = dk.VehicleMode("LAND")
if lat_avg > 7959.10 or lat_avg < 7779.60 or now_lonm < -6922.30 or now_lonm > -6833.10:
  while True:
    vehicle.channels.overrides = {'1': x1, '2': x2, '3': 995, '4': x4}
    vehicle.mode = dk.VehicleMode("LAND")
time.sleep(0.5)
exit()
time.sleep(0.2)

n = n + 1
j = j + 1
k = k + 1

print j
print k

vehicle.close()

2. Camera Access for Ardrone

import time, sys
import cv2
import numpy as np
import ps_drone # Import PS-Drone-API

contourArray = []

drone = ps_drone.Drone() # Start using drone
drone.startup() # Connects to drone and starts subprocesses

drone.reset() # Sets drone's status to good
drone.setSpeed(0.05)
while (drone.getBattery()[0]==-1): time.sleep(0.1) # Wait until drone has done its reset
print "Battery: "+str(drone.getBattery()[0])+"% "+str(drone.getBattery()[1]) # Battery-status

drone.useDemoMode(True) # Set 15 basic dataset/sec

##### Mainprogram begin #####
drone.setConfigAllID() # Go to multiconfiguration-mode
drone.sdVideo() # Choose lower resolution (try hdVideo())
drone.frontCam() # Choose front view
CDC = drone.ConfigDataCount
while CDC==drone.ConfigDataCount: time.sleep(0.001) # Wait until it is done (after resync)
drone.startVideo() # Start video-function
#drone.showVideo() # Display the video

##### And action! 
IMC = drone.VideoImageCount # Number of encoded videoframes
drone.trim() # Recalibrate sensors
drone.getSelfRotation(5)

print "Auto-alternation: " + str(drone.selfRotation) + " dec/sec" # Showing value for auto-alteration

drone.takeoff() # Fly, drone, fly!
while drone.NavData["demo"][0][2]: time.sleep(0.1) # Wait until the drone is really flying (not in landed-mode anymore)

##### Mainprogram begin #####
print "Drone is flying now"

# contour filter
def is_contour_bad(c):
    # approximate the contour
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.02 * peri, True)

    # the contour is 'bad' if it is not a circle
    return not len(approx) == 4

while True:
    while drone.VideoImageCount != IMC: time.sleep(0.01) # Wait until the next video-frame
        IMC = drone.VideoImageCount
        img = drone.VideoImage
        frame = cv2.resize(img,(720,480))
        blur = cv2.GaussianBlur(frame, (5,5),0)

        #set marker
cv2.imshow('Drones video',frame) # Show processed video-image
#cv2.imshow('Mask',mask_green)
k = cv2.waitKey(5) & 0xFF
if k == 27:
    drone.stop()
    break

#drone.land()
capture.release()
cv2.destroyAllWindows()

3. Simulation of Coverage Strength Model (Created by Xuebo Zhang and Farsam Fazardpour, edited by Tong Zhang)

global h_map w_map n_sensor theta_sensor p_sensor x_sensor y_sensor alpha_sensor rho
global asx asy bxs bsy beita_as beita_bs;
global alpha_sensor X Y r_sensor

h_map=80;
w_map=165;

%%% Initial sensor properties
n_sensor=9;
r_sensor=ones(1,n_sensor).*47.2;
% R=r_sensor;
alpha_sensor=ones(1,n_sensor).*(0.59*2);
y_sensor=w_map*rand(1,n_sensor);
x_sensor=h_map*rand(1,n_sensor);
theta_sensor=2*pi*rand(1,n_sensor);
%save Iniial_Cond y_sensor x_sensor theta_sensor
load Iniial_Cond

%%% a_s of all sensors
asx=r_sensor.*(cos(alpha_sensor/2)-0.5);
asy=r_sensor.*(sin(alpha_sensor/2));
beita_as=atan2(asy,asx);
bxs=asx;
bys=-asy;
beita_bs=atan2(bsy,bxs);

%%% Generating the discrete AREA
X_base=1:w_map;
Y_base=1:h_map;
[X,Y]=meshgrid(X_base,Y_base);
Gx=zeros(h_map,w_map);
Gy=zeros(h_map,w_map);
% Csmax=zeros(h_map,w_map);
% Csind=zeros(h_map,w_map);
Cs=zeros(h_map,w_map,n_sensor);
dcs_dtheta=zeros(h_map,w_map,n_sensor);
dcs_dp_X=zeros(h_map,w_map,n_sensor);
dcs_dp_Y=zeros(h_map,w_map,n_sensor);

rho=0.4;%.1;
k_c=1;
k_theta = 0.002;16

T=100;
w=5;
total_coverage = ones(1,T);
for t=1:T
    coverage = 0 ;
    if double(get(gcf,'CurrentCharacter'))==27
        break;
    end
    Csmax=[];
    Csind=[];
    Csmax=zeros(h_map,w_map);
    Csind=zeros(h_map,w_map);
    for i=1:n_sensor
        %mapping
        [qsx,qsy]=getTransform(X,Y,i);
        %[dc,dcs_dtheta(:,:,i),dcs_dp_X(:,:,i),dcs_dp_Y(:,:,i)]=getG(qsx,qsy,X,Y,i);
        %coverage distance and ....
        Cs(:,:,i)=exp(-rho.*dc);%
        coverage_distance and ...
        Csind=Csind.*(Cs(:,:,i)<=Csmax)+i.*(Cs(:,:,i)>Csmax);%
        Csmax=max(Csmax,Cs(:,:,i));
    end
    subplot(2,2,1);
    imshow((Csmax/max(max(Csmax)))*2.8)
    hold on
    axis on
    box on
    % contour(Csmax/max(max(Csmax)))
    axis xy
    axis square

    subplot(2,2,3);
    imshow((Csmax/max(max(Csmax)))*2.8)
    hold on
    axis on
    box on
    % contour(Csmax/max(max(Csmax)))
    axis xy
    axis square

    subplot(2,2,2);
    imshow((Csmax/max(max(Csmax)))*2.8)
    hold on
    axis on
    box on
    % contour(Csmax/max(max(Csmax)))
    axis xy
    axis square

    subplot(2,2,4);
    imshow((Csmax/max(max(Csmax)))*2.8)
    hold on
    axis on
    box on
    % contour(Csmax/max(max(Csmax)))
    axis xy
    axis square
hold off;
 imshow(Csind./n_sensor);
 axis xy
 hold on;


 w=5;%%
 sumH=0;
 for i=1:n_sensor

 \%
 Updating the sensors parameters

 x_sensor_old=x_sensor(i);
 x_sensor(i)=x_sensor(i)+dH_dp_X.*k_c;
 x_sensor(i)=min(w_map,x_sensor(i)); % escape avoidance
 x_sensor(i)=max(0,x_sensor(i)); % escape avoidance

 y_sensor_old=y_sensor(i);
y_sensor(i)=y_sensor(i)+dH_dp_Y.*k_c;

y_sensor(i)=min(h_map,y_sensor(i));  % escape avoidance
y_sensor(i)=max(0,y_sensor(i));    % escape avoidance

theta_sensor(i)=theta_sensor(i)+dH_dtheta.*k_theta;

%%% Plotting

arrow_x=[x_sensor_old,x_sensor(i)];
arrow_y=[y_sensor_old,y_sensor(i)];

subplot(2,2,4);
hold on;
plot(arrow_x,arrow_y,'-');
axis square
hold off
xlim([0,w_map])
ylim([0,h_map])

end
for j=1:h_map
    for k=1:w_map
       if Cmax(j,k)==0.6703 || Cmax(j,k)>0.6703
          coverage = coverage + 1;
       end;
    end;
end;
total_coverage(t)=coverage./(h_map*w_map);
pause(0);
hold off;
subplot(2,2,2);
hold on;
axis square
plot(t,sumH,'r.');

sumHRes(t)=sumH;
...

aad=toc;
if aad>1000
    break
end

t=linspace(1,aad,length(sumHRes));
figure;plot(t,sumHRes,'o-');
save GD1
xlabel('Time[s]')
ylabel('Total coverage performance')
$$k=\text{linspace}(1,aad,\text{length}(\text{total}\_\text{coverage}))$$

figure;plot(k,total\_coverage,'o-'')
save GD1
xlabel('Time[s]')
ylabel('Total coverage')

4. **Image and Video Transmission**

**Client**

import socket
import cv2
import numpy

TCP\_IP = '192.168.1.125'
TCP\_PORT = 10000

sock = socket.socket()
sock.connect((TCP\_IP, TCP\_PORT))

while True:

    capture = cv2.VideoCapture(0)
capture.set(1,740)
capture.set(2,320)
ret, frame = capture.read()

    encode_param=[int(cv2.IMWRITE_JPEG\_QUALITY),90]
result, imgencode = cv2.imencode('.jpg', frame, encode_param)
data = numpy.array(imgencode)
stringData = data.tostring()

    sock.send(str(len(stringData)).ljust(16));
respone = sock.recv(1024)
feedback = int(respone)
if feedback == 1:
    #print feedback
    sock.send( stringData );
    decimg=cv2.imdecode(data,1)
    cv2.imshow('CLIENT',decimg)

    #
    #capture.release()
    #
    cv2.destroyWindow('CLIENT')

**Server**

import socket
import cv2
import numpy

TCP\_IP = '192.168.1.125'
TCP\_PORT = 10000

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)
s.bind((TCP\_IP, TCP\_PORT))
s.listen(True)

conn, addr = s.accept()
buff = 1024

while True:

    length = conn.recv(16)
    counter = int(length)
    conn.sendall(str(1))
    img = b''
    newimg = b''
    while counter:
        newimg = conn.recv(buff)
        img += newimg
        counter -= len(newimg)

    data = numpy.fromstring(img, dtype='uint8')
    decimg = cv2.imdecode(data, 1)
    cv2.imshow('SERVER', decimg)
    cv2.waitKey(1)
    # cv2.destroyAllWindows('SERVER')
Bibliography


Vita Auctoris

Tong Zhang was born in 1991 in Nanjing, Jiangsu, China. He received Bachelor of Applied Science in Electrical Engineering from University of Windsor in 2009. He started program in Master of Applied Science in Electrical Engineering after graduation under the supervision of Dr. Xiang Chen. Tong is expected to obtain the Degree of Master of Applied Science in Nov 2017.