Development of Teleoperated and Semi-Autonomous Aerial Vehicles

By Abdullah Al Redwan Newaz

A project paper submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Nak Young Chong

March, 2014

Development of Teleoperated and Semi-Autonomous Aerial Vehicles

By Abdullah Al Redwan Newaz (1210060)

A project paper submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Nak Young Chong

and approved by
Professor Nak Young Chong
Associate Professor Fumihiko Asano
Associate Professor Azman Osman Lim

February, 2014 (Submitted)

Abstract

A quadrotor offers a challenging control problem due to its inherently unstable nature. An effective control methodology is therefore needed for such a unique airborne vehicle. A robot with this capability could be useful for many applications including search and rescue, exploration in hazardous environments, surveillance, etc. However, there are many challenges that engineers must face before developing such a robot, including the strict limitations in sensing technologies, power consumption, platform size and embedded processing. A key direction of this research aims at designing a stable system for controlling teleoperated quadrotor that is equipped with limited range of sensors. It is very difficult to achieve autonomous aerial navigation in GPS denied environment toward a goal position avoiding unpredictable collision. In order to safely maneuver within these environments, it would be beneficial for such a robot to be able to hover. This alone introduces many difficult problems including stability control, altitude control, platform drift, collision avoidance, and platform design, all being important for a successful operation. The system must also be able to sense its environment, prevent collisions and maneuver through the environment safely. First part of this project we have developed semi-autonomous aerial vehicle which is capable to control its altitude in GPS denied environment and it is a future platform for developing fully autonomous agile aerial vehicle. Quadrotor UAVs appear in miniature form in contrast to typical aerial vehicles, whereby the possibility of aerial vehicle swarming becomes a reality. We can use this terrific technology in different application and different purpose. Therefore, in the second part of this project we have discussed couple of path planning algorithms in cluttered environment where reduction of computational expense has given greater attention; one of them is rescue mission after nuclear disaster. However, in this particular problem we want to localize all of radioactive materials after nuclear disaster. The algorithm of informative path planning for mobile UAVs is addressed to reduce the uncertainty in rapid localization of radiation contaminated quantities that is a prime interest of research in the future world. However, without a priori knowledge on the whereabouts of the source of radiation substances leakages, it is very difficult to select the region of interest and appropriate measurement locations which are deemed to contain the most valuable information. Although sequential surrogate modeling provides a global picture of the radiation exposure of an area, particularly for fast emergency response, all the regions are not as informative as to explore. Therefore, to minimize the number of UAVs and the operating time required to explore the whole area, we propose a single UAV path planning algorithm for building an intensity contour map with the budget based greedy algorithm. We have demonstrated the efficiency of the proposed algorithm to create a surrogate model of intensity contour map with the V-REP robot simulator and analyzed the contour map by numerical simulations using MATLAB.

Contents

Chapter 1

Introduction

This project work focuses on the development of teleoperated and semi-autonomous aerial vehicles where Vertical Take-Off and Landing (VTOL) is demonstrated. The proposed structure is a four propeller helicopter called quadrotor. In these last years, a growing interest has been shown in robotics. In fact, several industries (automotive, medical, manufacturing, space, . . .), require robots to replace men in dangerous, boring or onerous situations. A wide area of this research is dedicated to aerial platform. Besides industrial application, this robot has terrific opportunities to contribute in following fields

- **First response:** Looks for intruder, gas leaks etc.
- **Transportation:** Recently amazon has introduced shipment of products and delivery to consumer by UAV
- **Construction**: It is capable to carry construction materials to build a large structure.
- **Film shooting:** Nowadays this kind of UAV is popular for filming due to its stable dynamics.
- **Search and rescue:** We can send it to reactive building to map the radiation level.

Several structures and configurations have been developed to allow 3D movements. For example, there are blimps, fixed-wing planes, single rotor helicopters, bird-like prototypes, quadrotors, . . . Each of them has advantages and drawbacks. The Vertical Take-Off and Landing requirement of this project exclude some of the previous configuration. However, the platforms which show this characteristic have unique ability for vertical, stationary and low speed flight. The quadrotor architecture has been chosen for this research for its low dimension, good maneuverability, simple mechanics and payload capability. Quadrotor is highly agile but fragility comes as a cost. As another main drawback, the high energy consumption can be mentioned. However, the trade-off results very positive. In this paper, teleoperated means UAV is connected to remote computer via wireless device as well as it includes on board processor to communicate with motors and other devices. Basically, this kind of teleoperated UAV has been developed with high speed communication to

base node (computer). There are several advantages to have this kind of system. One of the major benefits is that computationally expensive algorithm can be implemented to high performance computer end without disturbing the basic hardware setup of UAV. However, the disadvantage comes in that the situation when base node does not response to the remote system (UAV). To overcome this problem we have built our UAV with on-board processor and sensor so that it can cope with that by hovering until battery goes to dead. Energetically, a closed-loop teleoperator is a two-port system with the master and slave ports being coupled with the human operators and slave environments, respectively. Therefore, the foremost and primary goal of the control (and communication) design for the teleoperation should be to ensure interaction safety and coupled stability [21] when mechanically coupled with a broad class of slave environments and humans. To ensure such interaction safety and stability, energetic passivity (i.e., mechanical power as the supply rate [90]) of the closed-loop teleoperator has been widely used as the control objective. This is due to the property of the passive systems [86] : a feedback interconnection of any passive systems (with compatible supply rates) is necessarily stable (and also passive). In this project we consider to indoor flight of UAV. The major problem of indoor navigation is that global position is unknown. Since indoor is Global positioning system (GPS) denied environment, it is very hard to localize UAV by itself. We have introduced embedded sensors like gyro, accelerometer, ultrasonic sonar etc. to our system so that it can minimize the error of localization locally, but this error will be accumulated over time and it will cause a large error in global position. Human has versatility in this regard. Sometimes human can work as a sensor that artificial complex sensor does. Our objective is to reduce the burden of human so that without paying much concentration and skilled operation, UAV could be controlled autonomously.

Chapter 2

State of the art

In the last few years, the state of the art in Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicle (UAV) has received several contributes. Moreover, most of the attention has been focused on, the quadrotor structure. Some projects are based on commercially available platforms like Draganflyer, X-UF and MD4-200 [31] . Other researchers prefer instead to build their own structure. A few examples are the mesicopter, the X4-Flyer and the STARMAC. There are articles which present hybrid configuration such as structure with non-symmetric rotation directions or with two directional rotors [6, 93] . A few other works focus instead on modelling derivation [2, 34] and efficient configurations [43] . Multi-agent task is also an interesting field for VTOL UAV [84] . Even though there are a lot of different topics about the qudrotor structure, that one on which most of the publications have focused on is the control algorithm. It can be stated that the 85 percent of the articles propose a control low or compare the performance of few of them. The most important techniques and the respective publications are now presented: The first control is done using Lyapunov Theory [12,18,19,75] . According to this technique, it is possible to ensure, under certain condition, the asymptotical stability of the helicopter. The second control is provided by PD 2 feedback and PID structures [13, 81, 82] . The strength of the PD2 feedback is the exponential convergence property mainly due to the compensation of the Coriolis and gyroscopic terms. On the contrary a PID structure does not require some specific model parameters and the control law is much simpler to implement. The third control uses adaptive techniques [4, 66] . These methods provide good performance with parametric uncertainties and unmodeled dynamics. The fourth control is based on Linear Quadratic Regulator (LQR) [13, 20] . The main advantage of this technique is that the optimal input signal turns out to be obtainable from full state feedback (by solving the Ricatti equation). On the other hand the analytical solution to the Ricatti equation is difficult to compute. The fifth control is done with backstepping control [59–61] . In the respective publications the convergence of the qudrotor internal states is guaranteed, but a lot of computation is required. The sixth control is provided by dynamic feedback [64, 65] . The seventh control is based on visual feedback. The camera used for this purpose can be mounted on-board [41, 63, 83] (fixed on the helicopter) or offboard [3, 33] (fixed on the ground). Other control algorithms are done with fuzzy techniques [23] , neural networks [32] and reinforcement learning [88].

Chapter 3

Mathematical modeling

The quadrotor is an under-actuated system with four actuators controlling its six degrees-of freedom position and orientation. The flight controller is responsible for achieving two challenging goals simultaneously: (i) controlling the quadrotors position, while (ii) stabilizing its attitude, i.e., orientation (roll, pitch and yaw angles). More specifically, given a desired position (px, py, pz) and yaw angle , the goal is to design a controller to force these control states to converge to their respective desired values, while maintaining the pitch and roll angles as close to zero as possible. The quadrotor configuration space C can be expressed by

$$q = \frac{\partial^2}{\partial t^2}(x, y, z, \theta, \phi, \psi) \in C \tag{3.1}$$

Since the quadrotor c-space has 6 dimension, it has 6 degree of freedom.

$$C \subset R^3 \times S^3 \tag{3.2}$$

For quadrotors there are several coordinate systems that are of interest. We will define and describe the following coordinate frames: the inertial frame, body frame and the vehicle frame. Throughout the book we assume a flat, non-rotating earth: a valid assumption for quadrotors

1 **INERTIAL FRAME:**The inertial frame,$F_i = (\overrightarrow{x_i}, \overrightarrow{y_i}, \overrightarrow{z_i})$, is an earth-fixed coordinate system with the origin located on the ground, for example, at the base station. By convention, the x-axis points towards the north, the y-axis points towards the east, and the z-axis points towards the center of the earth.

2 **BODY FRAME:** The body frame,$F_b = (\overrightarrow{x_b}, \overrightarrow{y_b}, \overrightarrow{z_b})$ with its origin located at the center of gravity (COG) of the quadrotor, and its axes aligned with the quadrotor structure such that the x-axis is along the arm with front motor, the y-axis is along the arm with right motor, and the z-axis , where x denotes the cross product.

3 **THE VEHICLE FRAME:** The vehicle frame, $F_v = (\overrightarrow{x_v}, \overrightarrow{y_v}, \overrightarrow{z_v})$, is the inertial frame withthe origin located at the COG of the quadrotor. The vehicle frame has two variations, F_ϕ andF_θ. F_ϕ is the vehicle frame, F_v, rotated about its z-axis $\overrightarrow{z_v}$ by

an angle so that $\overrightarrow{x_v}$ and $\overrightarrow{y_v}$ are aligned with $\overrightarrow{x_b}$ and $\overrightarrow{y_b}$, respectively. F is frame F rotated about its y-axis, $\overrightarrow{y_\theta}$, by a pitching angle, θ, such that and are aligned with $\overrightarrow{x_\phi}$ and $\overrightarrow{z_\phi}$, respectively.

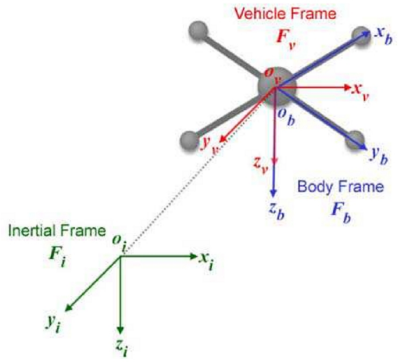


Figure 3.1: Quadrotor's coordinate frame

The moment of inertia is calculated by assumingthe quadrotor as a central sphere of radius r and mass Mo surrounded by four point masses representing the motors. Each motor is supposed to have a mass m and attached to the central sphere through an arm of length l. With the derived kinematic and dynamic model, we will now define the forces and torques acting on the quadrotor. The forces include the aerodynamic lift generated by each rotor, and the gravitational pull acting in counter to the total lift generated. The moments are the torques generated in order to achieve the roll, pitch and yaw movements. The following forces and torques are produced Upward Force (Thrust): The total quadrotor thrust is the sum of the thrust produced by each propeller, as depicted in Figure 8(a):

$$T = T_f + T_r + T_b + T_l \tag{3.3}$$

Rolling Torque: This is the torque produced by increasing the left rotors thrust while decreasing that of the right rotor, or vice versa, as shown in Figure 8(b):

$$T = T_f + T_r + T_b + T_l \tag{3.4}$$

Pitching Torque: The pitching torque in Figure 8(c) is produced by increasing the front rotors thrust while decreasing that of the back rotor, or vice versa:

$$\tau_\theta = l(T_f - T_b) \tag{3.5}$$

Yawing Torque: The yawing torque is the result of all four individual torques generated due to the spinning rotors. The front and back rotors spin in the clockwise direction, while the left and right rotors spin in the counterclockwise direction. As shown in Figure 8(d), an imbalance between these two pairs results in a yawing torque causing the quadrotor to rotate about its z-axis:

$$\tau_\psi = T_f + T_r - T_b - T_l \tag{3.6}$$

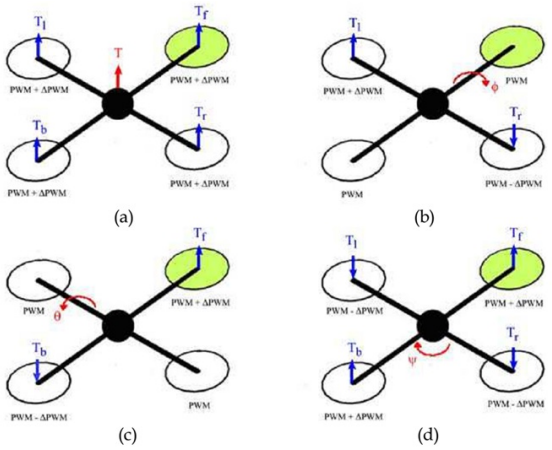


Figure 3.2: **Quadrotor dynamics:**(a) Quadrotor thrust; (b) Rolling torque; (c) Pitching torque; and (d) Yawing torque.

Let $P_F^T = [p_x, p_y, -p_z]$ and $\Omega_F^T = [\theta, \phi, \psi]$ denote the quadrotor's position and orientation within a given frame F. M is the quadrotors total mass. The terms τ_θ , τ_ϕ , and τ_ψ are the roll, pitch and yaw torques respectively. Thus, the translation dynamic model can be written as

$$\begin{bmatrix} \ddot{p}_x \\ \ddot{p}_y \\ \ddot{p}_z \end{bmatrix} = \begin{bmatrix} \dot{\psi}\dot{p}_y - \dot{\theta}\dot{p}_z \\ \dot{\phi}\dot{p}_z - \dot{\psi}\dot{p}_x \\ \dot{\theta}\dot{p}_x - \dot{\phi}\dot{p}_y \end{bmatrix} + \frac{1}{M} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3.7}$$

while the rotational model is

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{jy-jz}{jx}\dot{\psi}\dot{\theta} \\ \frac{jz-jx}{jy}\dot{\phi}\dot{\psi} \\ \frac{jx-jy}{jz}\dot{\theta}\dot{\phi} \end{bmatrix} + \frac{1}{M} \begin{bmatrix} \frac{1}{jx}\tau_\phi \\ \frac{1}{jy}\tau_\theta \\ \frac{1}{jz}\tau_\psi \end{bmatrix} \tag{3.8}$$

where, J is inertia matrix that can be calculated by equation

$$j_z = j_y = j_z = \frac{2M_0 r^2}{5} + 2l^2 m \tag{3.9}$$

Chapter 4

Component selection

The following major components are used to develop our quadrotor :

1) **Frame wheel and landing skid:** We have used DJI 450 frame wheel which is built by carbon fiber material. This frame is extremely durable and provides optimum space for electronic components. Because of carbon fiber body, we can avoid the error of magnetometer caused by metal body. However, our landing skid set also made by carbon fiber and it provides abundant space from ground to vehicle body. It also safety take off as well as landing and unwanted crash.

2) **High speed DC brushless motors and propeller:** DJI 2212 has been used which can rotate at 920 kv speed. It is compatible with DJI 1045 propellers. Furthermore, carbon fiber 10 propeller can produce more torque by consuming little motor power.

3) **Electronic Speed Controller (ESC):** An Electronic Speed Controller, or ESC controls the speed of the motor. ESCs will have a power limit. The more power an ESC can handle, the larger, heavier and more expensive the ESC will be. When choosing an ESC, it needs to match or exceed the motors peak amperage. We have used DJI 30A Opto ESC which is wide range of compatibility to flight controller boards.

4) **Power distribution board:** We have designed a separate power distribution board for our UAV. We have used two different capacity lithium batteries. Our power distribution board offers individual operation of electronic circuit and motors. Therefor, it will provide long time battery life by preventing unnecessary draining by motors and ESCs. Furthermore, the combine switch allow to couple to battery to ensure more flight time during maneuvering of quadrotor.

5) **Autopilot :** Autopilot is a command and control system for UAV. It provides a three-axis body orientation and moving map on the ground station. The unmanned aerial robot has instrumentation for attitude, engine and position. There are branch of autopilot system available in commercial market. We have used Ardru Pilot Mega

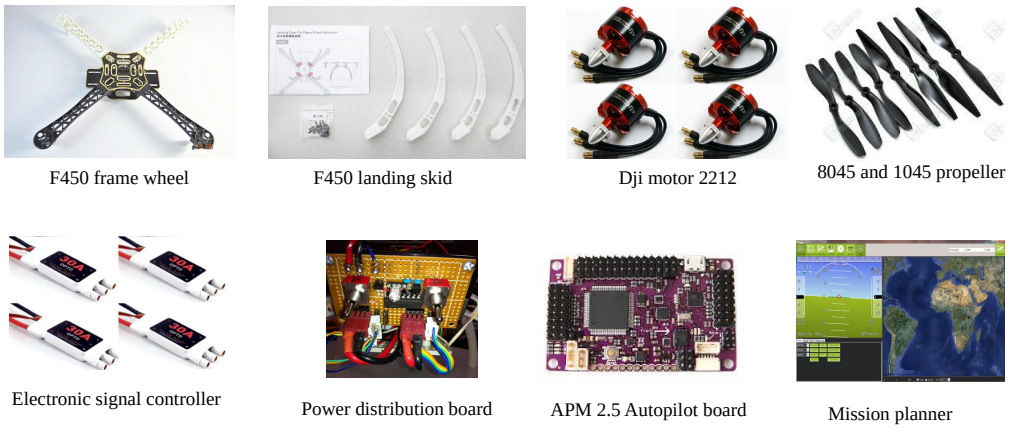


Figure 4.1: Components used in developing quadrotor

(APM 2.5) autopilot board to our system. APM2.5 is an open source auto pilot device that is easier to use compare to other devices.

6) **Mission planner:** APM 2.5 board comes with mission planner which is commercial an environment for communicating to hardware. The APM 2.5 board has 3 PID controllers to control roll, pitch and yaw orientations. However, it allows to change the variable of different parameters that are programmed in hardware for instance PID gains, radio calibration values, damping frequency value etc.

Chapter 5

Hardware development

5.1 Transmitter Design

Our objective is to control the UAV from a remote computer where our main controller has been implemented. Therefore, an interface is required that allows to connect our system to computer. Conventional way is to use 7 to 9 channels radio (R/C) transmitter and a commercial interface called PCTx to communicate with telepresence UAV. However, the data transfer rate of this interface is slow and the communication medium is bit complex[22]. Generally in R/C device uses either Pulse Width Modulation (PWM) or Pulse Position Modulation (PPM) to communicate with autopilot. Some devices that use PWM for control are ESC's (electronic speed controls) and servos. PWM is a technique used to relay data in the form of a varying pulse width. Usually in R/C equipment an entire PWM pulse will last a total of 20ms. The entire pulse is called a frame. A complete frame will include both the time the pulse is high (1-2ms) and the time the pulse is low. The image below represents a typical PWM frame. Although the frame lasts 20ms the important part of the pulse is the time the pulse is on; 1-2ms. Although the time between pulses is not as important it does play an important role. Usually keeping the time between pulses around 20ms is best. If the delay is longer, a servo for example will lose holding power. A pulse can be generated much faster but 20ms is best for the most situations.

On the other hand, PPM basically is several PWM signals lined up back to back. A PPM frame looks like this: Aside from the gaining servo holding power, the reason for the

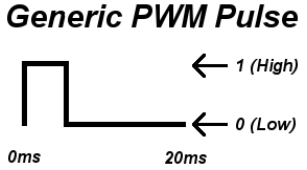


Figure 5.1: PWM pulse generation

20ms frame is just having the ability to line up several PWM signals in the same frame.

Like the time the pulse is on is what is important because we are able to strip out this relevant data from a PPM frame to re-generate a PWM frame. For example, if a radio only sent 1 PWM signal at a time, it would take 20ms per channel. If we use an 8 channel radio each update would take 160ms. The same data can be packed into a PPM frame and only take 20ms per update. Transmitters and receivers are the two most common R/C devices that use PPM.

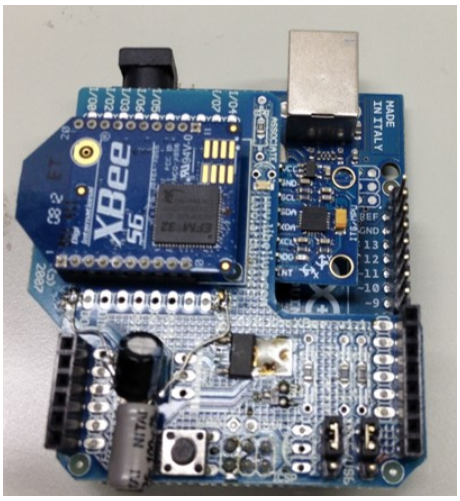


Figure 5.2: Designated transmitter

Ironically, neither PWM nor PPM can communicate to computer, serial communication is used where at a time one bit data sequentially transmit to communication channel or computer bus. The amount of digital signal transmission is expressed as baud rate where higher baud rate means higher pulses per second. We have designed a novel transmitter which offer higher data transmit rate and easier communication medium. To develop this interface following components are used:

1. Arduino UNO microcontroller
2. XBEE wifi with Arduino shield

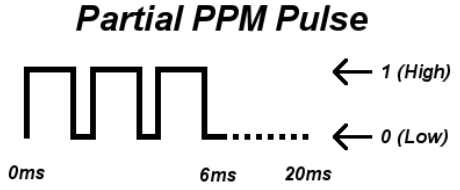
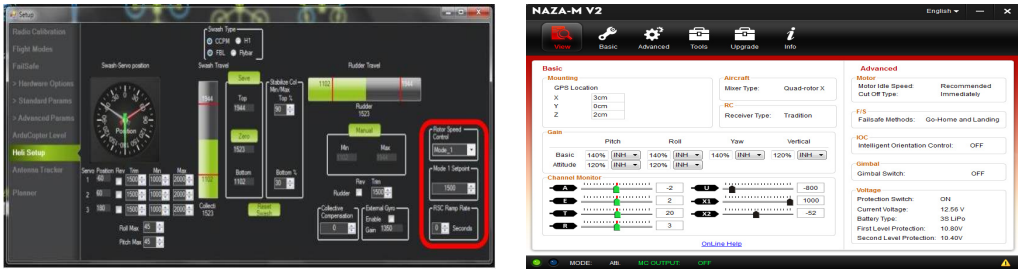


Figure 5.3: PPM pulse generation

The benefit of using XBEE wifi is that we dont have to use another device like R/C transmitter at computer side since our computer is already connected to internet. XBEE wifi offers higher data rate that is at most 3mb/s by using SPI communication. However,

PCTx can communicate at 50Hz PPM refresh rate whereas by using XBEE wifi Universal asynchronous receiver/transmitter (UART) communication media, we can use 115200 baud rate to talk to on board Arduino UNO microcontroller. Arduino UNO translate serial data to PWM signal to communicate with autopilot board. XBEE wifi is mounted on UNO board by Arduino Xbee shield. However, traditional Arduino Xbee shield does not support XBEE series 6 wifi model since it consumes more power than previous model. Therefore, 400uF capacitor is soldered to the shield for supplying extra current during power spike. We have testified our wireless transmitter by using two different commercially available software i.e. 1) mission planner for APM 2.5 autopilot board. 2) NAZA M V2 for DJI autopilot board. By sequentially increasing the value of servo speed, we can determine exact numbers to communicate with autopilot board from radio calibration option.



(a) APM 2.5 radio calibration (b) DJI Naza radio calibration

Figure 5.4: transmitter calibration by using two different environment

5.2 Sensor attachment

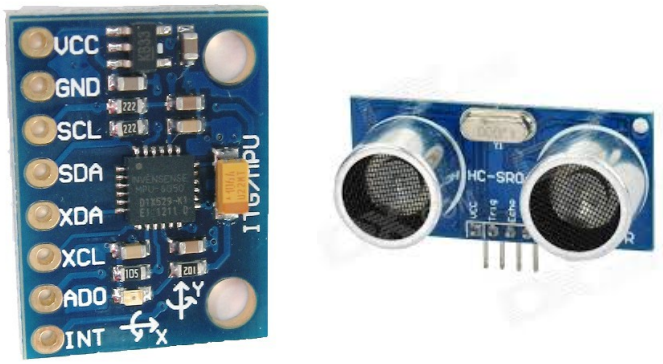
Although our IMU board includes different types of onboard sensors, we dont have any access to that sensors due to commercial environment. Therefore, we have developed our transmitter equipped with cheap but useful on-board sensors *i.e.* Motion sensors and Ultrasonic sensor. Motion sensors include 3Axis gyro and 3 Axis acceleration information and the particular model we have used for our system is called MPU-6050. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore, it captures the x, y, and z channel at the same time. Reading the raw values for the accelerometer and gyro is easy. The sleep mode has to be disabled, and then the registers for the accelerometer and gyro can be read. But the sensor also contains a 1024 byte FIFO buffer. The sensor values can be programmed to be placed in the FIFO buffer. And the buffer can be read by the Arduino. The FIFO buffer is used together with the interrupt signal. If the MPU-6050 places data in the FIFO buffer, it signals the Arduino with the interrupt signal so the Arduino knows that there is data in the FIFO buffer waiting to be read. The MPU-6050 always acts as a slave to the Arduino with the SDA and SCL pins connected to the I2C-bus. But beside the normal I2C-bus, it has its own I2C controller to be a master on a second (sub)-I2C-bus. It uses the pins AUX-DA and

AUX-CL for that second (sub)-I2C-bus. It can control, for example, a magnetometer. The values of the magnetometer can be passed on to the Arduino. MPU-6050 sends the Least Significant Bit (LSB/g) to the Arduino. We can get the real acceleration by using following formula

$$Acceleration = (9.81/LSB) * 1000 \tag{5.1}$$

Our next sensor is an ultrasonic range finder named HC-SR04. It detects the distance of the closest object in front of the sensor (from 2 cm up to 4m). It works by sending out a burst of ultrasound and listening for the echo when it bounces off of an object. The Arduino board sends a short pulse to trigger the detection, then listens for a pulse on the same pin using interrupt function. The duration of this second pulse is equal to the time taken by the ultrasound to travel to the object and back to the sensor. Using the speed of sound, this time can be converted to distance.

$$Distance = ((Durationofhighlevel) * (Sonic : 340m/s))/2 \tag{5.2}$$



(a) Dual motion sensor (b) Sonar sensor

Figure 5.5: Sensors attached to processor

Chapter 6

Controller Design

We have designed our controller by using LabVIEW (short for Laboratory Virtual Instrument Engineering Workbench) environment which is a system-design platform and development environment for a visual programming language from National Instruments. The benefits of using LABVIEW environment are as following:

a. **Interfacing:** A key of LabVIEW over other development environments is the extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments and buses are included or are available for inclusion. These present themselves as graphical nodes. The abstraction layers offer standard software interfaces to communicate with hardware devices. The provided driver interfaces save program development time.

b. **Code compilation:** In terms of performance, LabVIEW includes a compiler that produces native code for the CPU platform. The LabVIEW syntax is strictly enforced during the editing process and compiled into the executable machine code when requested to run or upon saving. The run-time environment makes the code portable across platforms.

c. **Large libraries:** Many libraries with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc., along with numerous graphical interface elements are provided in several LabVIEW package options. The number of advanced mathematic blocks for functions such as integration, filters, and other specialized capabilities usually associated with data capture from hardware sensors is immense.

d. **Code re-use:** The fully modular character of LabVIEW code allows code reuse without modifications: as long as the data types of input and output are consistent, two sub VIs are interchangeable. The LabVIEW Professional Development System allows creating stand-alone executables and the resultant executable can be distributed an unlimited number of times. The run-time engine and its libraries can be provided freely along with the executable.

e. **Parallel programming:** LabVIEW is an inherently concurrent language, so it is very easy to program multiple tasks that are performed in parallel by means of multithreading.

f. **Ecosystem:** Due to the longevity and popularity of the LabVIEW language, and the ability for users to extend the functionality, a large ecosystem of 3rd party add-ons has developed through contributions from the community.

For explanation of designing controller in computer end, we have subdivided this chapter to further three section.

6.1 Network design

After choosing appropriate environment the next thing that ought to do for teleoperated UAV is networking that brings life to the vehicle. We have established our network by Transmission Control Protocol (TCP) which is used to communicate with transmitter interface. The main advantage of TCP over User Datagram Protocol (UDP) is that an acknowledgement system works for prevention of losing packet. TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). IP works by exchanging pieces of information called packets. A packet is a sequence of octets (bytes) and consists of a header followed by a body. In our system, before initiating transmission from our main controller end, we send an acknowledgment signal what tells to on-board controller that we are ready to receive sensor data. However, sensor yields different size of packet that makes difficulty in receiving end. Therefore, before transmitting the packet we use bit stuffing to ensure constant length of each packet. However, in our specific problem we have used 10 bytes of packet and added zero in front of packet if size is less than 10. In serial data communication abrupt high speed communication stuck the system to deadlock situation. We have to sequentially increase the speed of communication to establish safe and durable connection. To get 25 ms sampling time we have reduced the time from hundred to 25 by 5 steps where each steps last about 100 ms. Finally we have got 25ms sampling time where samples taken from motion sensor are 60 and 5 from ultrasonic sensor; 65 samples per package. Different size of sample data from different sensors has been chosen.

6.2 Observer design

We have used accelerometer and Ultrasonic distant meter to measure the altitude of UAV. However, the response time of two sensors is not equal and accelerometer response much faster than sonar sensor. Therefore, between the sensing time interval of sonar and accelerometer, we have calculated the approximate distance based on accelerometer value. Approximate distance is calculated by following equation

$$\begin{aligned} Z^{+} &= a_z \times \partial t \\ Z &= Z + Z^{+} \end{aligned} \tag{6.1}$$

Where, Z is the previous distance achieved from sonar sensor, Z^+ is the intermediate distance calculated by accelerometer value and δt is the sampling time.

6.3 PID controller design

In the industrial area the most used liner regulators are surely the PID. The reasons of this success are mainly three *i.e* simple structure, good performance for several process and does not required specific model to design the controller. In robotics, PID technique represents the basics of control. Even though a lot of different algorithms provide better performance than PID, this last structure is often chosen for the reasons expressed above. The traditional PID structure is composed of the addition of three contributes, as shown in fig.(6.1) and eqn.(6.2). The blocks **"1/s"** and **"s"** represents the integration and

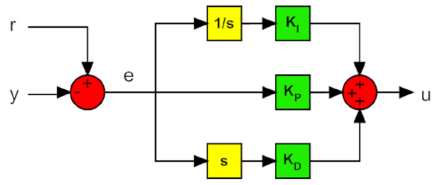


Figure 6.1: Traditional PID structure

derivation operations.

$$u(t) = K_P e(t) + K_I \int_0^t e(t)\,dt + \mathrm{K_Z}\frac{de(t)}{dt} \tag{6.2}$$

Where u is a generic controlled variable, e is the error between the task r and the process output y, K_P is the proportional coefficient, K_I is the integral coefficient and K_D is the derivative coefficient. The first contribute (P) is proportional to the error and define the proportional bandwidth. Inside this interval the output will be proportional to the error while outside the output will be minimum or maximum. The second contribute (I) varies according to the integral of the error. Even though this component increases the overshoot and the settling time, it has a unique propriety: it eliminates the steady state error. The third contribute (D) varies according to the derivate of the error. This component help to decrease the overshoot and the settling time. In the Laplace domain, the traditional PID structure can be rewritten according to eqn. (6.3).

$$u(s) = \left(K_P + \frac{K_I}{s} + sK_D\right) e(s) \tag{6.3}$$

Since this function is improper, it is not physically feasible (because of the derivative term). After a certain frequency, the D contribute must be attenuated to filter the off-band noise. For this reason, in the real derivator a pole is added as shown in eqn. (6.4)

$$u(s) = \left(K_P + \frac{K_I}{s} + \frac{sK_D}{1 + sK_D/k\,K_P}\right) e(s) \tag{6.4}$$

The traditional PID structure presents two main drawbacks:

- The derivate action is calculated from the error. If the task adds a step in the reference, the output of the derivator would present an impulse. This sharp movement can saturate the actuators and push away the system from the linear zone. For this reasons most of the PID architecture presents the derivate action of the process output only.
- The integral action combined with an actuator saturation can provide a non linear effect which can decrease the performance of the control system. When the integral value is large and the error changes sign it is necessary to wait a lot of time before the system restores its linear behavior (after the discharging of the integral action). This phenomenon is called integral wind-up. To avoid it, a saturator is added after the integral to limit its maximum and minimum values.

In our experiment due to lack of indoor global position system, we have demonstrated altitude control with the help of Ultrasonic sonar sensor and accelerometer. Block diagram of Altitude control is as follow: zd[m] represents the desired height, Z_{Acc}[m] is the height

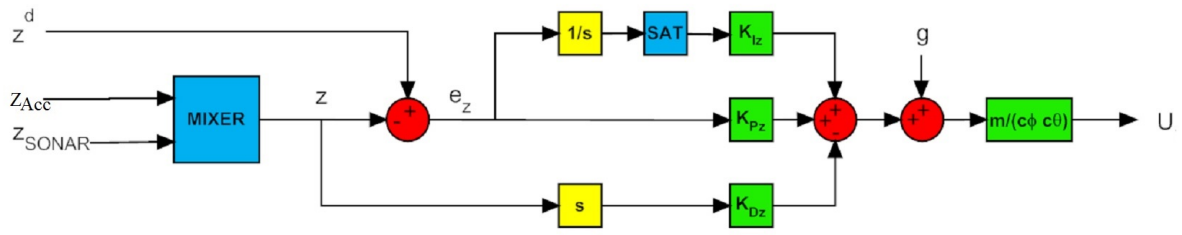


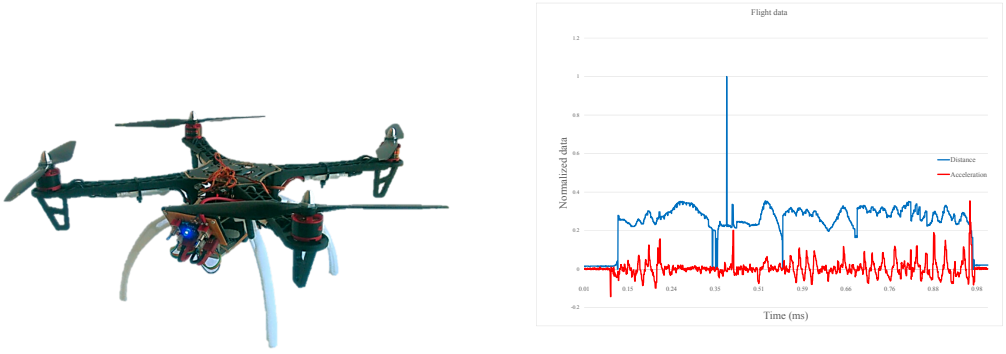
Figure 6.2: Block diagram of height control

measured by the accelerometer module, Z_{SONAR}[m] is the height measured by the SONAR module, z [m] is the height estimated from the sensors, e_z [m] is the height error, the block "SAT" represent the saturator and U[N] is the required thrust. $K_{Pz}[s^{-2}]$, $K_{Iz}[s^{-3}]$ and $K_{Dz}[s^{-1}]$ are the three control parameters. At last $g[ms^{-2}]$ is the acceleration due to gravity, $m[kg]$ is the mass of the quadrotor, c_ϕ is the roll angle cosine and c_θ is the pitch angle cosine. Respect to the enhanced PID architecture, the block diagram of the height control presents two main differences: The MIXER block must be added to process the height data from the accelerometer module and the SONAR. Its purpose is to calculate the height of the quadrotor with the best accuracy it can achieve from the two sensors. Furthermore, MIXER block, the algorithm estimates the real height and avoids ambiguity. According to eqn. (6.4) the height dynamics is more complex than the other three. In fact, it also depends from the roll and pitch angles. Furthermore the acceleration due to gravity must be compensated. The quadrotor mass, m (kg) has the same role as the moments of inertia in the angular case.

Chapter 7

Experimental result

Our developed UAV has 0.450 m length in diagonal. The total mass of this UAV is 1.63kg and the peripheral mass is 0.375kg. Due to homogeneous structure the moment of inertia is calculated by Eqn.3.9 The main controller at computer terminal includes two graph indicators that show acceleration and height information of UAV. We have used gaussian mean tool kit library to filter out noise comes through sensor. There are two independent loops to capture the value of transmitter and receiver respectively. The sampling rate is gradually fixed at 25ms (milliseconds). We have also designed a memory storage function to capture and store the real time flight data. After experimental flight following graph is obtained that includes a clear comparison between the accelerometer and sonar sensor information. However, we have encountered a major problem of sonar sensor during experiment. The sonar sensor responds slowly while moving above soft carpet due to the most of signals emitted to measure the height is disrupted by rough surface of soft carpet. The maximum delayed in responding is about 1.2 seconds. However, the respond time is pretty good during maneuvering over hard surface e.g. floor, plastic board, wooden table etc. There are eventually two individual algorithms *i.e.* Transmitter algorithm and



(a) Photo of assembled quadrotor

(b) Dead reckoning: blue-sonar value, red- accelerometer value

Figure 7.1: Experiment with JAIST Quadrotor

Controller Algorithm work together to fly this robot. However, first of all controller needs to establish network connection by using TCP. UAV will respond to the controller after getting default control signal from it. Moreover, if there is no data received from the transmitter, controller can't yield input command. Hence, it is very important to get the real time data within bounded communication time.

Algorithm 1 Transmitter Algorithm

```
1: Data = Receive_control_signal()
2: if Data then
3:     for i = 1; i < 3; i++ do
4:         S[i] = Get_Acceleration()          ▷ Read accelerometer
5:         S[i + 3] = Get_Gyro()              ▷ Read gyro sensor
6:     end for
7:     for j = 1; j < 11; j++ do
8:         Send_Motion_values(S)
9:     end for
10:    for j = 1; j < 5; j++ do
11:        D = Get_Get_altitude()             ▷ Read Ultrasonic sensor
12:        Send_Ultrasonic_values(D)
13:    end for
14: end if
```

Algorithm 2 Controller Algorithm

```
1: Get_default_values_input
2: Send_to_transmitter
3: Data = Transmitter_response()
4: if Data then
5:     Decode_Control_Signal
6:     Filter_Out_Noise()
7:     Estimate_Altitude()
8:     Operate_PID()
9:     Send_Inputs()
10: end if
```

After receiving signal from transmitter, controller decode the signal to understand specific input and sensor values. The signal contains noise, therefore it is imperative to remove noise from sensor values. Afterwards, Altitude is calculated by combining acceleration and proximate distance values. Now, sensor reading is ready to input to the PID controller as a process variable. However, PID yields values with the respect to distance, as a result we have to scale up the output values to transmitter input domain which is between 1000 to 2000 in number.

Chapter 8

Path planning Algorithm 1

3D Exploration Priority Based Heuristic Approach for Path Planning of Unmanned Aerial Vehicles

This paper presents a 3D online path planning algorithm for Unmanned Aerial Vehicles (UAVs) equipped with limited range sensors and computational resources in unknown cluttered environments. Even though quadrotor UAVs are considered to be a promising technology for surveillance purposes in indoor environments and for close observation in outdoor urban areas, it is very difficult to achieve autonomous aerial navigation toward a goal avoiding unpredicted collisions. Furthermore, greater attention and effort should be aimed at improving the computational efficiency and performance of path planning algorithms. The proposed heuristic algorithm offers on-the-fly path findings with a lesser computational complexity. We demonstrate the efficiency of our algorithm in a real world scenario implemented using the V-REP simulator.

8.1 Introduction

In this paper, we consider the surveillance and recovery mission after nuclear disasters or severe accidents in industrial areas which are inaccessible by humans. We divide the possible use of UAVs in the above-mentioned missions into three different categories: 1) indoor environment without having GPS or motion capture sensors [44], 2) indoor environment with motion capture sensors, 3) outdoor environment for close observation with the help of GPS.

In an indoor environment, one of the most important challenges for UAVs is to localize itself and search for the position to move, since the GPS does not work. Moreover, vision based localization is not always useful especially when after disaster situations are considered, since we cannot guarantee adequate lighting. Furthermore, scanning laser range finders or different types of vision sensors increase the computational burden, as a result it is quite difficult to implement in low-cost on-board processing units. Therefore, in the first scenario, without considering the computationally expensive sensors, the Exploration

Priority Based Heuristic Approach (EPBHA) to online path planning is proposed.

(a) Outdoor GPS coordinates (b) Indoor path planning

Figure 8.1: Initial nominal path with respect to GPS coordinates

Obviously, we must have an offline plan [70] before a surveillance mission that can be made by choosing several waypoints using the satellite maps as shown in Fig. 8.1. The waypoints are Cartesian coordinates representing spatial positions in the horizontal XY plane where a nominal height is assumed in the Z axis. Since every movement of an UAV creates its own coordinates, if we calibrate the UAV coordinate system with respect to outdoor GPS coordinates, the UAV can reach its goal along a set of waypoints. Moreover, we must have an approximate idea for maximum altitude for indoor environments. Specifically, if the previously unknown obstacles are detected, the UAV has to re-plan its nominal path in real time. If the path planner fails to generate a safe path within a bounded time, collisions with obstacles may result. Since the computational time of deterministic and complete algorithm grows exponentially with the dimension of the configuration space, those algorithms do not provide an adequate solution for online UAV path planning in indoor environments. However, as the UAV can not compare its coordinates with GPS or specific land coordinate systems, inertial navigation errors can be accumulated with its exploration. The accuracy of goal findings depends on a proper calibration system. Nevertheless, we assume that the requirements for the localization accuracy are not very strict for surveillance missions particularly within a small arena.

In the next scenario, stationary motion capture sensors are used to identify the coordinate of UAV. We can implement our algorithm in this scenario, since a number of infrared reflective markers are used to identify the object, therefore the environment is unknown unless the markers are attached to obstacles. Finally, our algorithm works for close observation in outdoor environments, where different size and shape of obstacles may appear in the path of a navigating UAV.

8.2 Related Works

Path planning has been one of the most important elements of mission definition and management of vehicles and it became crucial after birth and growth of UAVs. Quadrotors inaugurate the miniature form of UAV and furthermore their kinematics gives hovering capabilities that make it easy to create paths on the fly. Several algorithms were developed

for robotic ground vehicles [11, 48, 74]. Likewise, physics for potential field algorithms [8,89], mathematics for probabilistic approaches [73], or computer science for graph search algorithms [45] were applied to UAV path planning.

Binney *et al.* [9] presented a path planning method for autonomous underwater vehicles (AUVs) to maximize mutual information. Pereira and Sukhatme [72] presented minimum-risk path planning for AUVs operating in coastal regions with high ship traffic. Jung and Tsiotras [52] explained on-line path generation for UAVs using B-spline path templates where they investigated the problem of generating a smooth, planar reference path, given a family of discrete optimal paths. Jun and D'Andrea [50] used the probability map and Bellman-Ford shortest path algorithm in adversarial environments, maximizing the safety of the vehicles. Yang and Sukkarieh [92] discussed 3D path planning for an UAV operating in cluttered natural environments. Hrabar [49] proposed a synthesis of techniques for rotorcraft UAV navigation through obstacle-populated environments. Rohmer and Randall provided the target position programming solution [40], where a low level control of UAV was implemented with the target subdivided into *horizontal control and vertical control*.

Many techniques were developed to tackle the independent components for safe vehicle navigation in unknown environments. We handpick a selection of these that, when combined, offers what we believe is the best solution for the disaster surveillance with quadrotor UAVs. The need for off-line and real-time replanning substantially revises the path planning strategy. Moreover, the computational performances of the control station, where the mission management system is running, can influence the algorithm selection and design. The use of evolutionary algorithms for path optimization is an important solution permitting to apply kinematic constraints to the path. Using splines or random trees to model the trajectory, these algorithms can reallocate the waypoint sequence to generate optimum solutions in complex environments [42, 69]. Being interesting and flexible, the evolutionary algorithms are spreading on different planning problems, but their complexity is paid with a heavy computational effort [17]. The Dijkstra algorithm is one of the first greedy algorithms for graph search and permits to find the minimum path between two nodes of a graph with positive arc costs [30]. An evolution of the Dijkstra algorithm is the Bellman-Ford algorithm [7, 39] that finds the minimum path on oriented graphs with positive and negative costs. Another important method is the Floyd-Warshall algorithm [38,87] that finds the shortest path on a weighted graph with positive and negative weights, but it reduces the number of evaluated nodes compared with the Dijkstra algorithm. The A* algorithm is one of the most important solvers explicitly oriented to robotics. A* improved the logic of graph search with heuristic evaluations inside the loop [9]. Dynamic re-planning with graph search algorithms was introduced. D* (Dynamic A*) represented the evolution of A* for re-planning [78]. Then, research on dynamic re-planning brought to the development of Lifelong Planning A* (LPA*) and D* Lite. They are based on the same principles of D* and D* focused, but they recall the heuristic aspect of A* to improve the speed of the search process [54, 55]. Different approaches were developed to cope with the suboptimal solutions problem, based on post-processing algorithms or on improvements of the graph-search algorithm itself. Very

important examples are Field D* [35] and Theta* [67]. These algorithms refined the graph search obtaining generalized paths with any heading.

Comparing all of these algorithm, De Filippis *et al.* [25] conclude that Theta* is the most promising solution for the path planning of fixed-wing UAVs. However, the shortcoming of Theta* is the computational time. Our proposed algorithm is thus aimed at reducing the computational time.

8.3 Problem Statement

We categorized indoor surveillance missions further into two cases: 1) navigate in the obstacle free space and 2) avoid obstacles and escape from a deadend passageway. The former case means that the UAV finds the minimum distance path toward a goal position, if there is free space to move. The latter, however, must conform to several crucial conditions: how to avoid unexpected obstacles that appear in the path (located in front, or to the left or right, or any possible arrangement of obstacles, except for the upward direction). A complex and unpredictably changing environment makes it difficult to accomplish safe path planning. Moreover, using of vision sensors increases the computational complexity that makes it difficult to accommodate on-board implementation requirements. Therefore, without having any *a priori* knowledge of the environment, this paper proposes a new heuristic approach to allow UAVs to navigate through complex terrains, ensuring near-constant computations.

Now we address the path planning of UAV in unknown environments as follows: *Assuming a surveillance UAV equipped with limited range sensors exploring an arena, where different types of unknown obstacles exist, how to make it go to a goal position avoiding the obstacles with comparatively little computational cost?*

The path planning problem above can be decomposed into two sub-problems:

- **Sub-problem 1** (*free space*) How does it travel a minimum possible distance in an obstacle free area?
- **Sub-problem 2** (*obstacle avoidance*) How does it re-plan its position, while avoiding obstacles in its path?

8.4 Algorithm Description

The idea underlying the proposed algorithm is similar to A* algorithm. However, in A* algorithm for 2D plane, 8 Cartesian coordinates are computed and the coordinate of minimum cost among the cost of all coordinates is required to determine the movement position. Since the UAV does not know *a priori* the location of the obstacle, the cost of each coordinate is calculated based on, for instance, 'Manhattan Distance'. Although the proposed algorithm is a 3D path planning algorithm, to reduce the computational complexity, only one plane is chosen at a time for maneuvering. In practice, 6 movement options (forward and backward, left and right, and up and down) are available for the

proposed path planning, while up and down movements are considered the special cases of obstacle avoidance maneuver. Therefore, normally for maneuvering UAV (U), costs are calculated based on 4 coordinates, which are front (C12), left (C21), back (C32), right (C23), respectively, as shown in Fig. 10.2.

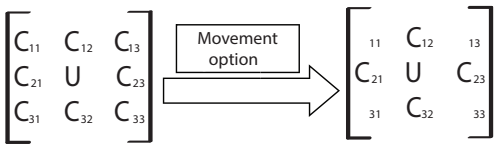


Figure 8.2: Reduced cost assessment

Definition 1 (Input Description) *The Cartesian coordinates of current position and goal position are given and the rest of coordinates are unknown. The UAV thus knows its own position and goal position but does not know* a priori *the obstacle position. The distance between one coordinate and the next coordinate is defined as step length d. The value of d is propositional to the velocity of UAV. For larger value of d, the UAV increases its velocity to cope with the distance that is required to travel within limited time boundary. The goal position is divided into two parts,* i.e.*, the goal in the XY plane and the YZ plane, respectively. After reaching the goal in one plane, the goal is automatically shifted to the other.*

Definition 2 (Cost for coordinate) *A coordinate cost is defined by the difference between the current position (x1, y1) and the next position (x2, y2) given by*

$$Cost := A \times (x_1 - x_2) + B \times (y_1 - y_2),$$

where A and B are arbitrary even constants for emphasizing the straight forward (X-axis) or straight sideward (Y-axis) movements instead of the diagonal movements travel. If $A > B$, then the UAV moves forward or backward, while $A < B$ indicates left or right movements.

8.5 3D Exploration Priority based Heuristic Approach for Obstacle Avoidance

In the proposed algorithm, the UAV searches two 2D planes separately to reduce the complexity of computations. After achieving the goal in the XY plane, it will shift its goal into the YZ plane that is the final goal. The searching algorithms is also divided into the obstacle free area and the obstacle cluttered area. The UAV tries to identify the shape of obstacle using its limited range of sensing which is analogous to the blind cane. It then chooses appropriate predefined maneuvering behaviors to avoid the particular type of

obstacle. The proposed algorithm is basically divided into four parts: 1) grid making, 2) cost calculation, 3) obstacle avoidance, and 4) move to minimum cost point. Furthermore, four subfunctions are used which are the heading axis, sensor value, movement option, and next set position, respectively.

1. Grid making: The incremental distance between the parent coordinates and next coordinates is termed as d as defined in the previous section. For the next set position, one coordinate is chosen among four neighboring coordinates. The value of d could be determined by calibrating in the real world environment. If we compare our result to real world GPS values outdoors, then we have to calibrate it with respect to GPS values. Moreover, the smaller value of d ensures lesser probability of colliding with obstacles.

Algorithm 3 Pseudocode for grid making

1: **for** $i = 1; i < 5; i++$ **do**
2: $grid[i][1] = i$ ▷ indexing
3: $grid[i][2] = x \pm d$ ▷ next x-coordinate
4: $grid[i][3] = y \pm d$ ▷ next y-coordinate
5: **end for**

2. Cost estimation: This part restricts the movement options of UAV: straight or perpendicular movements are more emphasized than diagonal movements. Therefore, costs of diagonal movements are higher than straight or perpendicular movements. This cost estimation (which is defined in Definition 2) is valid when there is no obstacle around the UAV.

3. Obstacle search: When the UAV finds an obstacle, it acquires two or more equal minimum cost coordinates at the same time. Therefore, according to A* or other existing algorithms, the UAV has to search every possible way to reach the goal, which we believe is quite impractical. In this work, the UAV has a preplanned idea about 'how to avoid the obstacles' and 'how to reduce the computational complexity'. Specifically, during the time of avoidance, it does not consider the cost for the goal. To acquire the knowledge of 'how to avoid the obstacles', we define several subfunctions detailed below.

Subfunction 1. (Direction of Heading) Comparing the current position ($x1$, $y1$) and the previous position ($x0$, $y0$), the UAV determines the X-axis or Y-axis along which it should move.

Algorithm 4 Pseudocode for heading direction in XY plane

1: **if** $(x0 - x1) > (y0 - y1)$ **then**
2: `heading_is x`
3: **else**
4: `heading_is y`
5: **end if**

Since the UAV's heading direction is likely to change, the variables of sensor value are also re-oriented accordingly. In Fig. 8.3, s1, s2, s3, and s4 represent the front, right, back, and left sensor value with respect to the UAV heading direction.

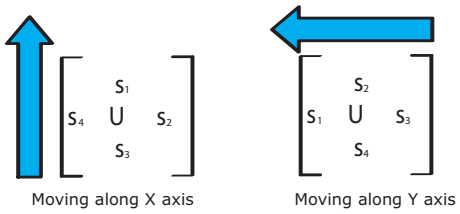


Figure 8.3: Heading and sensor variables

Subfunction 2. (Sensor Value) Obstacle detection is limited by the detection range and precision of sensors, where no detection range, the offset from the starting range, is introduced for sharp angle avoidance. Higher sensor range ensures safety, but decreases the accuracy to reach the goal.

Subfunction 3. (Movement Option) A sensor reports a certain range of numeric values, when it finds an obstacle. The available movement options are determined by counting the number of sensors that do not detect anything.

Algorithm 5 Pseudocode for movement option

```
for i = 1; i < 7; i++ do
    if value_of_sensor[i] > sensor_range then
        count+ = 1                          ▷ number of activated sensors
        movement_option = 6 − count
    end if
end for
```

Subfunction 4. (Next Set Position) The set position at the next moment ($x2$, $y2$) can be computed from the current heading and position ($x1$, $y1$) of the UAV.

Algorithm 6 Pseudocode for next set position in XY plane

```
if heading_axis == X then
    if y1 > y2 then
        next_set_position_is = left
    else
        next_set_position_is = right
    end if
else if heading_axis == Y then
    if x1 > x2 then
        next_set_position_is = left
    else
        next_set_position_is = right
    end if
end if
```

While the UAV moves along an axis and finds an obstacle in front of it, it calculates the set position at the next moment with respect to the current position, which gives priority

to a certain direction (Subfunction 4). This change of heading is due to the UAV's myopia in orienteering. In most cases, the position of obstacle is close to the ground, hence the UAV may find an obstacle-free path at a certain height from the ground. As a result, in this algorithm, passing over is another priority after the heading changing movement for obstacle avoidance. The UAV will determine more than one obstacle from the sensor value and movement option subfunctions.

Moreover, the most interesting feature of the proposed algorithm is to avoid the cave type obstacle. In order to avoid such an obstacle, the UAV detects overhead obstacles and looks for its backward movements with respect to its heading direction (Subfunction 1). To reduce the penalty of backward movements, we have emphasized a special diagonal movement instead of straight backward movements.

4. Moving to minimum cost point: The UAV finds an optimal coordinate for its next set position and relocates its position to this coordinate. When the UAV changes its heading, the sensor indexes are also changed accordingly.

Below is a sketch of the proposed algorithm, incorporating the above-mentioned function modules:

Algorithm 8.5.1: SEARCHING GOAL IN XY PLANE(x, y)

repeat
 GRIDMAKING()
 read sensor value
 if obstacle exist
  then EPBHA()
  else COSTESTIMATION()
 FINDMINIMUMINDEX()
 $compare(UAVPos(x, y), goalPos(x, y))$
 if $goalPos(x, y) - UAVPos(x, y) ==$ desired accuracy
  then xy search is finished
until xy search is not finished

Algorithm 8.5.2: SEARCHING GOAL IN YZ PLANE(y, z)

repeat
 GRIDMAKING()
 read sensor value
 if obstacle exist
  then EPBHA()
  else COSTESTIMATION()
 FINDMINIMUMINDEX()
 $compare(UAVPos(y, z), goalPos(y, z))$
 if $goalPos(y, z) - UAVPos(y, z) ==$ desired accuracy
  then yz search is finished
until yz search is not finished

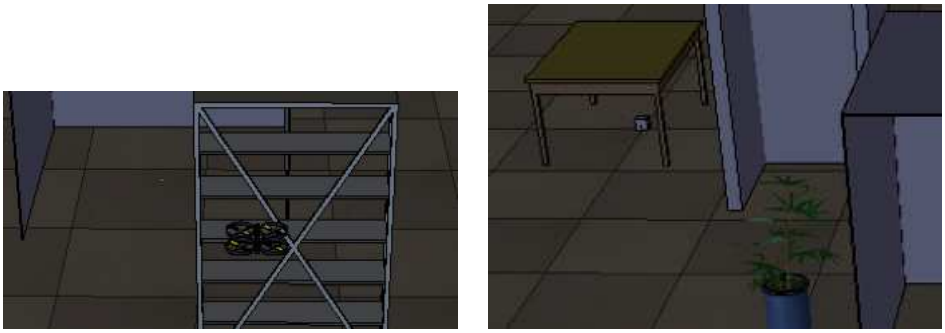
8.6 Simulation Results and Discussion

Six infrared sensors are used as proximity sensors to detect obstacles which are mounted on top, front, right, left, back, and bottom, of the UAV, respectively. The proximity sensors have $0.5m$ range and 45° angle of detection. Moreover, it is imperative to place the sensors 15° to 30° inclined to the surface of body for proper detection and safe avoidance of obstacles. However, as we do not consider the measurement accuracy and signal processing of the sensors, sensor data is assumed to be accurate, noiseless, and achieved instantaneously. Although the proposed path planning is also valid for dynamically changing goals, the goal position is considered as static in this simulation. The initial status of the UAV is the standard hovering position, where we specify the goal position, seen in Fig. 10.3 (a) and (b), respectively, and the rest of the UAV kinematics are adjusted automatically using the dynamic simulation engine. Furthermore, as this paper does not deal with a low level control system, we therefore assume that we can accurately estimate the next movement of UAV without dead reckoning and/or other aerodynamics errors. Note that the flight path varies depending on the situation and environment as shown in Fig. 10.3. Moreover, it is assumed that the sensing range for UAV is limited (*i.e.*, $0.5m$) and there is no initial information such as map or pre-specified path. Therefore, the UAV can not plan a long distance path and does not require to retrieve previously given data, as a result the computation complexity is much lesser. We compare our algorithm with existing A* and D* search algorithms which are commonly used for flight path planning to find the shortest path. The main difference starts while the UAV finds any obstacle along its path. When an obstacle appears in the path of UAV, it gets two or more minimum points for its next move. To find the shortest path, the UAV needs to explore every possible solution and decide which flight path it should choose.

In Fig. 8.5, the blue area indicates the searching area, where the yellow, black, red, and orange indicates the starting position, obstacle, goal position, and shortest path, respectively. From the figure, it is obvious that the proposed method offers less search, while other algorithms ensure the shortest path with higher search. We assumed that there is no initial information or map for the given place, therefore it is redundant for a single UAV to explore every possible way and choose the best one. Instead of searching for the minimum distance path, the 3D exploration capability of UAV allows it to easily avoid the obstacles. The most notable feature of the proposed method is that, obstacles reduce the searching time, while other existing searching algorithms always increase the computational parameters. Fig. 8.6 shows a significant decrease in coordinate cost estimation according to the obstacles position.

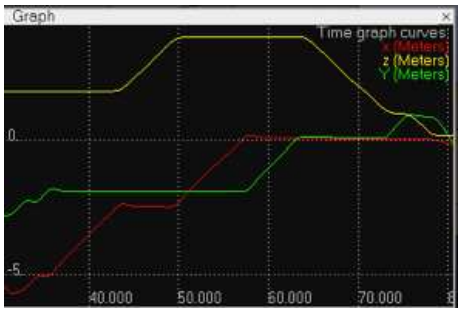
Since this algorithm does not use the global information, it does not ensure the shortest path. It assures one of the feasible paths with lesser computations. For the surveillance mission considered, it is not essential to find the shortest path all the time. It should ensure a close-up view for that place. Likewise, obstacles do not always prevent UAVs from navigating a preplanned path, rather they could be also important items for surveillance purposes.

Fig. 8.7 represents the offline path planning, where all the environment information is initially available and the path obtained is the shortest path. Meanwhile, Fig. 8.8 shows a



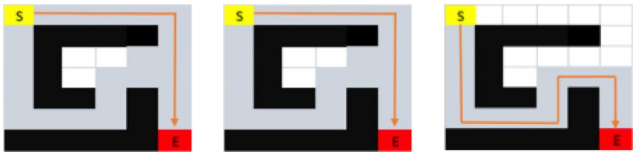
(a) Starting position, in a stable, flying condition

(b) Goal position, the box under the table



(c) Path traversing graph

Figure 8.4: Dynamics simulation setup and result



(a) A* Search (b) D* Search (c) EPBHA

Figure 8.5: Comparison between search algorithms and EPBHA

longer path compared to Fig. 8.7, but it ensures a close view for obstacles. This heuristic algorithm does not always guarantee to find the goal. For instance, it does not give any solution, while the UAV detects an obstacle in the backward direction. However, hovering is proposed for such a deadlock situation. Furthermore, the goal position is very close to the ground, therefore a small amount of error remains in the Y and the Z axis as shown in Fig. 10.3 (c). For better accuracy, the goal position should be located somewhere above the ground level and obstacle free environment. Fig. 8.9 shows that the real search time for the worst case setup is 2 minutes 20.57 seconds calculated by the real time function.

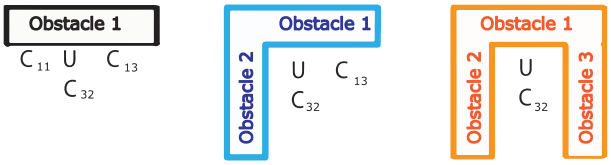


Figure 8.6: Coordinate cost calculation during obstacle avoidance

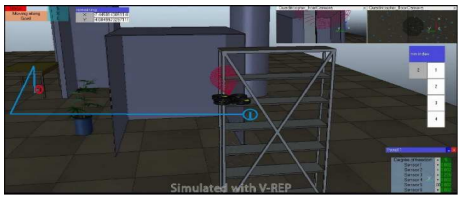


Figure 8.7: Offline path planning for known environment

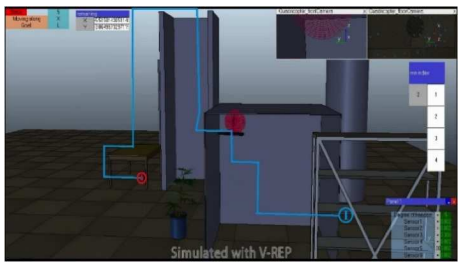


Figure 8.8: EPBHA path planning

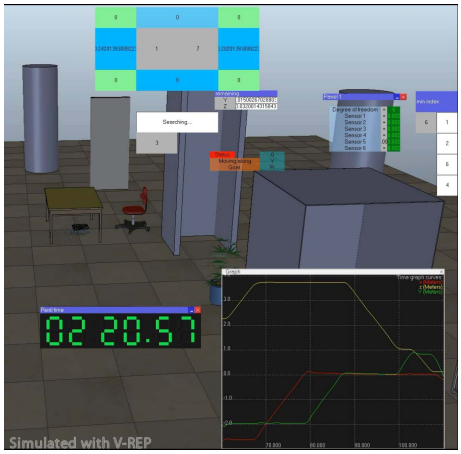


Figure 8.9: Real time of exploration

Chapter 9

Path planning Algorithm 2

3D Exploration Priority Based Flocking of UAVs

This paper presents a 3D flocking algorithm for a team of unmanned aerial vehicles (UAVs), where each member is equipped with limited range sensors and computational resources. A minimal leader-follower communication scheme is proposed for maneuvering huge swarm of UAVs. The proposed triangular formation compacts the overall group size. Even though UAVs are considered for tactical, remote monitoring, and surveillance purposes in both indoor and outdoor environments, it is very difficult to achieve autonomous aerial flocking in unknown cluttered environments. Specifically, greater attention is placed to reduce computational complexity for on-board implementation. We demonstrate the efficiency of our algorithm in a real world scenario with the V-REP simulator employing a group of five UAVs.

9.1 Introduction

In recent decades, unmanned aerial vehicles (UAVs) have attracted much attention due to their wide range of applications and reasonable manufacturing cost. Among different types of UAVs, researchers increase their focus on rotor wing UAVs particularly Quadrotor UAVs, because their kinematics offers low speed maneuvering and hovering. Quadrotor UAVs appear in miniature form in contrast to typical aerial vehicles, whereby the possibility of aerial vehicle swarming becomes a reality. Flocking is one of the basic elements of aerial swarm behavior. Considerable effort has been directed toward understanding how a group of autonomous creatures creates a certain form of clusters. Similar problems have been studied in ecology and theoretical biology, in context of animal aggregation and social cohesion in animal groups [14] [80].

Reynolds [71] proposed the basic model that was later modified in different ways. Delgado-Mata *et al.* [28] introduced the effects of fear by observing the activities of Olfaction to transmit emotion between animals through pheromones modeled as particles in a free expansion gas. Hartman and Benes [46] incorporated a complementary force to the alignment for a leadership change, where the steer defines the chance of the boid

to become a leader and try to escape. Hemerlijk and Hildenbrandt [47] used attraction, alignment and avoidance and extended the algorithm with a number of traits of starlings given by

1) birds fly according to the fixed-wing aerodynamics, while rolling and turning

2) they coordinate with a limited number of neighbors

3) staying above a sleeping site is given priority and when moving outwards the sleeping site, they return to it by turning,

4) fixed relative speed is proposed.

The authors claimed that the specifics of flying behavior as well as large flock size and low number of interaction partners were essential to the creation of the variable shape of flocks of starlings. Related problems have become a major thrust in system and control theory [24, 51, 53, 57, 77]. Vicsek *et al.* [85] proposed the leader following model, in which one agent acted as a group leader and others would just follow the aforementioned cohesion/separation/alignment rules.

Meanwhile, Lee and Chong [58] proposed the equilateral triangle lattice model in establishing selective local interactions among neighboring robots. They claimed that the equilateral triangle can reduce the number of robots in a given location, and improve the network connectivity and hole repair capability [16, 37].

Inspired by the results of [85] and [58], this paper introduces the *communication model* to the existing basic models of flocking behavior, where small intermediate *equilateral triangles* are considered to communicate with neighbors. Each intermediate group consists of three members *i.e., one leader and two followers*. We have previously proposed the exploration priority based heuristic approach (EPBHA) for UAV collision-free path planning with lesser computational complexity in cluttered environments. EPBHA reorients equilateral triangles into arbitrary triangular shapes depending on the type of obstacle and allows flexible path planning to avoid collisions. The communication model increases the overall team efficiency, since every robot is not required to find the position of obstacle that will be discovered by one of the team members.

9.2 Problem Statement

We categorize aerial swarm missions further into three cases: 1) navigate in obstacle-free environments 2) build/maintain a team via internal communications and 3) avoid obstacles and escape from a deadend passageway. Case 1) lies in more general context, which means if there is free space to move, the robot finds the minimum distance path toward a goal position. Case 2) and Case 3), however, must conform to several crucial conditions: how to build a large group, how to avoid unexpected obstacles that appear in the path of navigating robots. A complex and unpredictably changing environment makes it difficult to accomplish safe path planning. Moreover, vision sensors increase the

computational complexity that makes it difficult to accommodate on-board implementation requirements. Therefore, without having any *a priori* knowledge of the environment, this paper proposes a new heuristic approach to allow a group of UAVs consisting of two intermidate small groups to navigate through complex terrains, only based on six infrared sensors, ensuring a near-constant computational complexity.

Now we address the flocking problem of a group of UAVs in unknown environments as follows:

> *Assuming a group of UAVs equipped with limited range sensors exploring an arena, where different types of unknown obstacles exist, how to make it reaches a goal position avoiding the obstacles with comparatively little computational cost?*

This problem can be decomposed into simpler problems:

- **Sub-problem 1** (*path planning for free space*) How do all the members of a group travel a minimum possible distance in an obstacle free environment?
- **Sub-problem 2** (*group formation*) How to build and maintain a group?
- **Sub-problem 3** (*path planning for obstacle avoidance*) How does a group re-plan its next position, while avoiding any obstacles in its path?

9.3 Algorithm Description

Definition 3 (Triangular Configuration) *Given the leader robot r_l and neighbour robots r_{f1} and r_{f2}, a triangular configuration is defined as the set of their distinct positions $\{P_l, P_{f1}, P_{f2}\}$ denoted by*

$$T_i = \{P_l, P_{f1}, P_{f2}\}.$$

One half of the interior angle $\angle P_{f1}P_lP_{f2}$ is denoted by θ.

Definition 4 (Sensing Range) *Each robot is equipped with 6 proximity sensors detecting up to S_d with a 45° angle of coverage*

Definition 5 (Inter-robot Distance) *Given T_i, a safe distance is configured between the leader and follower robots, which must be greater than the sensing range (S_d) of individual robots.*

Definition 6 (Input Description) *The leader robot knows its own position and goal position but does not know* a priori *the obstacle position. The distance between one coordinate and the next coordinate is defined as step length d. The value of d is responsible for smooth motion planning which is propositional to the velocity of robot. The goal position is divided into one goal for the XY plane and another for the YZ plane. After reaching a goal in one plane, the goal is automatically shifted to the other plane.*

Definition 7 (Coordinate Cost) *A coordinate cost is defined by the difference between the current position* (x_1, y_1) *and the next position* (x_2, y_2) *given by*

$$Cost := A \times (x_1 - x_2) + B \times (y_1 - y_2),$$

where A *and* B *are arbitrary even constants for emphasizing the straight forward (X-axis) or straight sideward (Y-axis) movements instead of the diagonal movements travel. If* $A > B$*, then the robot moves forward or backward, while* $A < B$ *indicates left or right movements. Fig. 1 represents the cost reduction assessment where diagonal movement cost calculation is omitted by introducing maneuvering options.*

Now the proposed exploration priority based heuristic aerial swarming algorithm is divided into three major functions:

1 Path planning in obstacle free environments

2 Communication to follower robots

3 Path planning in cluttered environments

9.3.1 Path planning in obstacle free environment

The leader robot moves forward to the goal position based on *Manhattan distance*. By introducing *maneuvering options*, we restrict diagonal movements. Therefore, each robot is capable to move only six directions, *i.e.*, forward, backward, left, right, upward, and downward, respectively. Fig. 2 represents the triangular configuration for group formation, where the red circle represents the leader robot and the blue represents followers, respectively. The leader robot determines the positions for adjacent follower robots with respect to its current position. A safe distance is ensured by maintaining inter-robot distances greater than their sensing range. The leader robot plans for the equilateral triangular configuration for its group. Every robot creates four square grids, while moving towards the goal position. Among the four grids indicating four different coordinates, robots choose the best coordinate for their next movement by calculating the minimum cost.

Grid making The incremental distance between the parent coordinates and next coordinates is termed as *d*. For the next set position of robots, one coordinate is chosen among four neighboring coordinates. The value of d neds to be kept as small as possible to ensure lesser probability of colliding with obstacles.

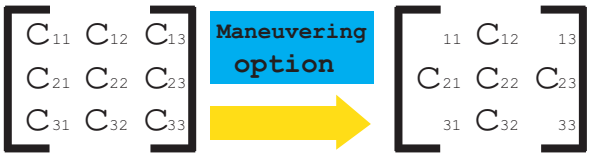


Figure 9.1: Reduced cost assessment

Algorithm 9.3.1: GRID MAKING(x, y, d)

```
for i ← 1 to 4
do { for j ← 1 to 3
     do { Grid[i][1] = i
          comment: indexing
          Grid[i][2] = x ± d
          comment: next x coordinate
          Grid[i][3] = y ± d
          comment: next y coordinate
```

Maneuvering Option A sensor reports a certain range of numeric values, when it finds any obstacle within its sensing range. The available movement options are determined by counting the number of sensors that do not detect anything.

Algorithm 9.3.2: MANEUVERING OPTION(g, h, i)

```
for i ← 1 to 6
do { if value_of_sensor[i] > sensor_range
     then { count+ = 1
            comment: number of activated sensors
            movement_option = 6 − count
```

Cost estimation The degree of freedom of robots is restricted by introducing maneuvering options, whereby straight or perpendicular movements are more emphasized than diagonal movements. Therefore, costs of diagonal movements are higher than straight or perpendicular movements. This cost estimation is valid when there is no obstacle around the robot.

Moving to minimum cost point The robot finds an optimal coordinate for its next set position and relocates its position to this coordinate.

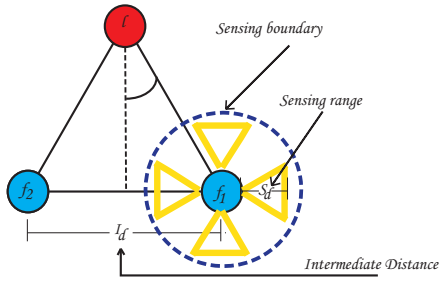


Figure 9.2: Triangular group formation

9.3.2 Communication to follower robots

There are two kinds of goal for flocking, such as the user defined goal for overall team maneuver and the leader defined intermediate goals. The leader robot is assumed to know the user defined goal and it creates new goals for its followers equipped with a wireless transceiver while traveling every new grids. To reduce communication between robots, only the leader sends the position information to its followers and does not take any feedback from them. Similarly, a follower which is the leader of the next triangular group sends its position to its followers. Follower robots estimate a safe distance from the information of leader position.

$$Send = \begin{cases} \texttt{Position} & \textbf{if } obstacle = \textbf{false} \\ \texttt{NULL} & otherwise \end{cases} \tag{9.1}$$

9.3.3 Path planning in cluttered environment

One half the interior angle between the leader and follower robots is denoted by θ whose unit is $degree/100$. Let $P(f_1|l)$ indicate the probability of obstacle existence with respect to the leader robot sensing value, whereas $P(l)$ represent the probability of obstacle existence with respect to the position of leader. The value of $P(l)$ is always 1 since the leader only communicates when it finds an obstacle. Therefore the probability of existing obstacle with respect to the follower position can be given by

$$\begin{aligned} P(f_1) &= P(f_1|l) \times p(l) \\ &= (1-\theta) \times 1 \end{aligned}$$

It is obvious that the path distortion (*i.e.*, probability of unexpected obstacle in the navigating path) depends on the angle θ, since the leader robot does not know the situation perpendicular to follower robots due to the triangular formation. We use the EPBHA algorithm for obstacle avoidance. In swarming purposes, robots do not communicate to others while avoiding any obstacle. This behavior enhances the efficiency of avoiding obstacles within a short period of time but restricts the minimum limit of interior angle. In Fig. 3, S_d and I_d represent the sensing range and inter-robot distance. Moreover, the blue dotted circle is the sensing boundary based on radius S_d. While the leader robot avoids an obstacle by changing its position to the left or right side, it should be longer than $(I_d \div 2)$, since the leader movements also affect the follower path plan. This function is further divided into two functions:

a) **Obstacle definition and avoidance:** Robots identify the type of obstacle from the number of active sensors as detailed below:

   1. **Easy: single-sided obstacles** The number of active sensors is one, *e.g.*, either front, left, or right. Robots avoid this type of obstacle by moving towards the goal direction.

2. **Medium: partition-type obstacles** The number of active sensors is more than one except upward and bottom sensors, *e.g.*, either Π or Γ shape. Robots pass over the obstacle.
3. **Hard: one-side open box shape obstacles** The number of active sensors is more than one, including upward or downward sensors. Robots move backward diagonally.

b) **Waiting for the leader instruction**: Since the leader does not communicate while avoiding obstacles, followers wait after arriving in their goal given by the leader. However, as soon as a new goal is informed, they boost their speed to achieve that goal position.

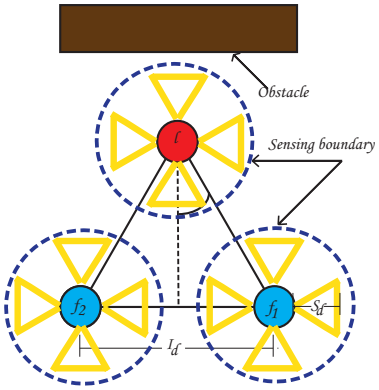


Figure 9.3: Path planning for obstacle avoidance

Below is a sketch of the proposed algorithm, incorporating the above-mentioned function modules: the common goal position is defined in terms of the leader position, therefore the given orientation of group is automatically adjusted for others.

Algorithm 9.3.3: COMMON GOAL PLANNING(x, y)

repeat
 GRIDMAKING()
 read sensor value
 if obstacle exist
  then EPBHA()
  else COSTESTIMATION()
 FINDMINIMUMINDEX() **and** $send(Position)$
 $compare(UAVPos(x, y), goalPos(x, y))$
 if $goalPos(x, y) - UAVPos(x, y) ==$ desired accuracy
  then xy search is finished
until xy search is not finished

Algorithm 9.3.4: INDIVIDUAL GOAL PLANNING(y, z)

repeat
 gridMaking()
 read sensor value
 if obstacle exist
  then EPBHA()
  else costEstimation()
 findMinimumIndex()
 $compare(UAVPos(y, z), goalPos(y, z))$
 if $goalPos(y, z) - UAVPos(y, z) ==$ desired accuracy
  then yz search is finished
until yz search is not finished

Algorithm 9.3.5: FOLLOWER ROBOT PATH PLANNING(x, y)

repeat
 Receive($leaderPosition$)
 Estimate($ownPosition$)
 gridMaking()
 read sensor value
 if obstacle exist
  then EPBHA()
  else costEstimation()
 findMinimumIndex() **and** $send(Position)$
 $compare(UAVPos(y, z), goalPos(y, z))$
 if $goalPos(y, z) - UAVPos(y, z) ==$ desired accuracy
  then yz search is finished
until yz search is not finished

9.4 Simulation Result

Six infrared sensors having $0.5m$ range and 45° angle of detection are mounted on top, front, right, left, back, and bottom of every robot, respectively. The sensor data is assumed to be accurate, noiseless, and achieved instantaneously. As this paper does not deal with a low level control system, dead reckoning and/or other aerodynamics errors are assumed to be negligible. There is no *a priori* information such as map or pre-specified path. The initial position of every robot is in a stable hovering position. Each robot is capable of avoiding collisions and re-planning its path in real time. If the path planner fails to generate a safe path within a bounded time, collisions may result. It is advantageous to form a small group to minimize communication delays between robots and expect fast responses from followers. The proposed algorithm accelerates the overall

team speed, and followers require comparatively less time to find their path and effectively carry out other missions. Thus we can create a binary tree [75] structure to form a large team, where the node will be the leader and leaves will represent the followers. Once a

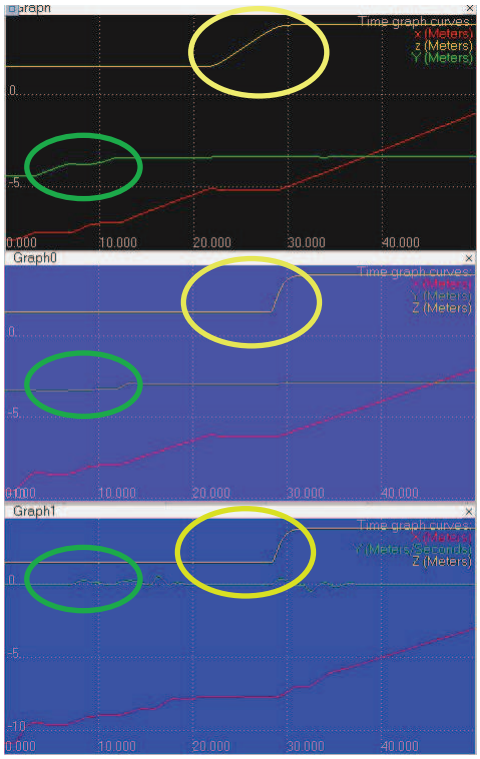


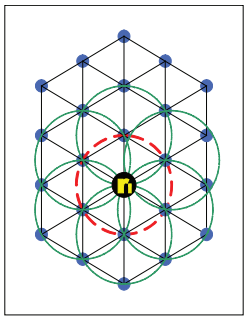
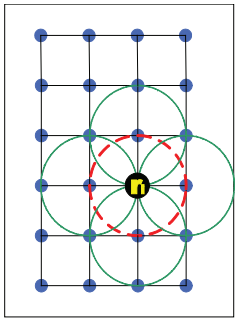
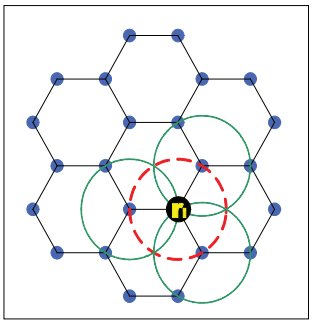
Figure 9.4: Vehicle trajectory analysis. The top graph represents the leader trajectory, and the middle and bottom graphs represent the follower trajectories (robot no. 3 and 5 depicted in Fig. 5(c)), respectively.

huge triangular shape is formed, the members on the boundary have main responsibilities to make decisions for overall team maneuvering. The leader does not communicate to followers while avoiding obstacles. Therefore, followers wait until they receive obstacle free path coordinates from the leader. This will boost the speed of path planning for followers. As shown in Figure 4, followers exhibit faster translation compared to the leader.

In short, every follower decides its path either: 1) self avoidance using its sensors, or 2) prediction of obstacle position using the leader's heads up. Predicting obstacle positions offers fast path planning for followers, while self avoidance ensures safer path planning despite of sensing errors. Open loop communication also increases overall communication speed, since the leader does not need any feedback to be confirmed regarding the exact positions of its followers. We have transformed 3D path planning into two separate XY search and YZ search problems to reduce computational complexity and improve planning efficiency.

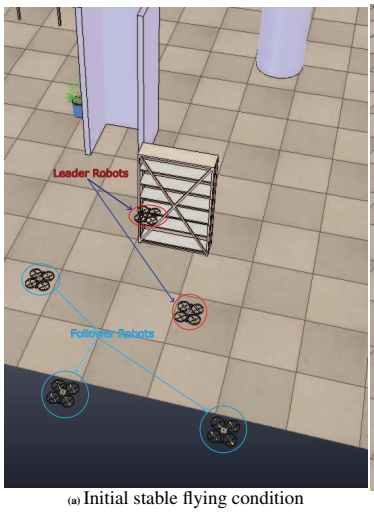
As seen in Table 1 [12] , the triangle geometry offers higher converge density and better connectivity. While following the leader instructions, two neighboring robots may collide

Table 9.1: Comparison of lattice typed network pattern

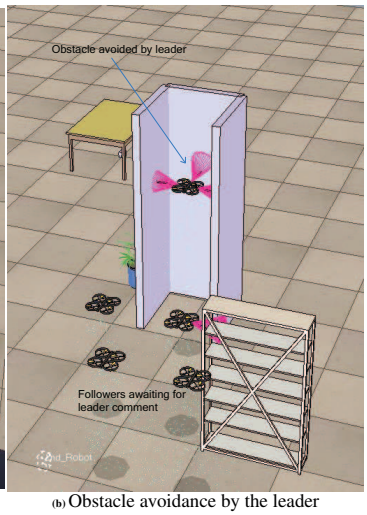
	Triangle	Square	Hexagon
Geometry			
Coverage Area	$\frac{\sqrt{3}}{2} \times n \times (d_u)^2$	$n \times (d_u)^2$	$\frac{3\times\sqrt{3}}{4} \times n \times (d_u)^2$
Coverage density	$\frac{1.2}{(d_u)^2}$	$\frac{1}{(d_u)^2}$	$\frac{0.78}{(d_u)^2}$
connectivity	6	4	3

with each other in the same plane during obstacle avoidance. To cope with this problem, robots create a different layer while passing through a narrow passageway. With this layered formation, robots maintain the pre-defined triangular geometry but do not fly at the same height. This feature increases the volume flow rate of flocking, when passing through a narrow opening. By using [13] illustrated in Fig. 5, we have defined the common goal and individual goals and decided the common goal position based on the first leader robot position (robot no. 1), while followers maintain their geometric shape by maintaining the triangular configuration. After acquiring the common goal position, the leader robot moves toward its individual goal position that is located underneath the table.

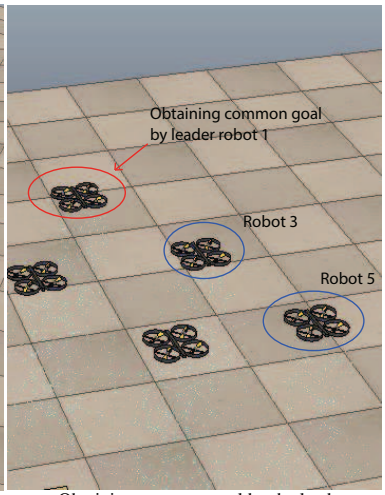
To recapitulate all, we propose a heuristic approach to aerial flocking which ensures *lesser computational complexity, high volume flow rate, and lesser communication delay.*



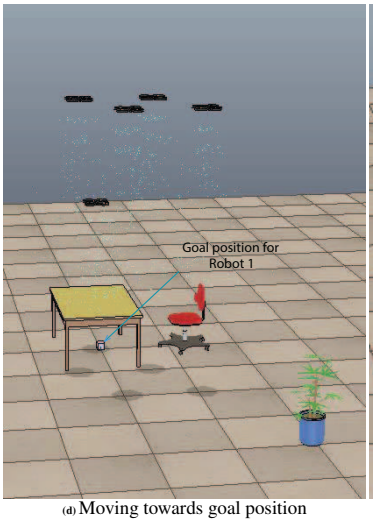
(a) Initial stable flying condition



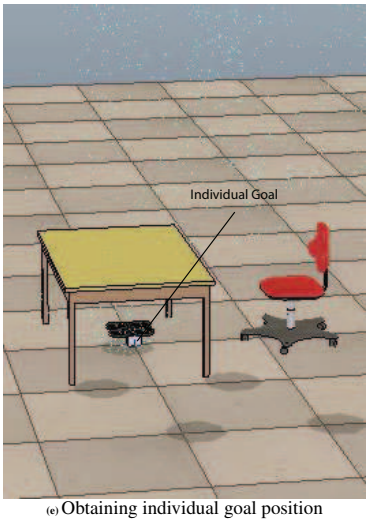
(b) Obstacle avoidance by the leader



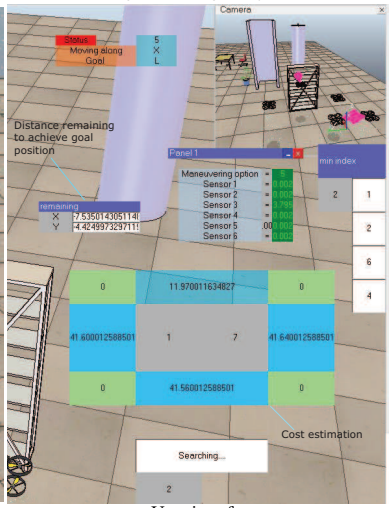
(c) Obtaining common goal by the leader



(d) Moving towards goal position



(e) Obtaining individual goal position



(f) User interface

Chapter 10

Path planning Algorithm 3

A Fast Algorithm for Building Intensity Contour Maps

In this study, a fast algorithm is proposed to create a surrogate model of intensity contour map from a limited number of intensity samples of the radiation field. Unmanned aerial vehicles (UAVs) have been considered as the most promising technology for on-the-spot investigation of accidental radiation releases. However, without *a priori* knowledge on the whereabouts of the source of radiation substances leakages, it is very difficult to select the region of interest and appropriate measurement locations which are deemed to contain the most valuable information. Although sequential surrogate modeling provides a global picture of the radiation exposure of an area, particularly for fast emergency response, all the regions are not as informative as to explore. Therefore, to minimize the number of UAVs and the operating time required to explore the whole area, we propose a single UAV path planning algorithm for building an intensity contour map with the budget based greedy algorithm. Furthermore, we have demonstrated the efficiency of the proposed algorithm with the V-rep robot simulator and analyzed the contour map by numerical simulations using Matlab.

10.1 Introduction

Natural disasters are sometimes inevitable. Thus, effective emergency relief and disaster recovery coordination can be facilitated by immediate on-the-spot investigations. For instance, if radioactivity levels spike in areas around nuclear power plants, it is very important to find contaminated hotspots and leaks to quickly characterize the severity of the situation. Therefore, this paper aims at proposing a robotic algorithm to model, characterize, and localize radiation hotspots in unknown environments, creating a preliminary yet informative radiation map. When considering robots for a variety of rescue missions, unmanned aerial vehicles (UAVs) come to first priority. Among the family of UAVs, quadrotors have achieved more popularity for surveillance and rescue missions because their kinematics and dynamics support hovering even in indoor environments. However, surveillance, patrolling, and rescue must be accomplished within a limited time frame due

to quadrotors' short battery life. Therefore, an efficient navigation algorithm is needed for obtaining informative radiation maps. There are several challenges to characterize and localize radiation hotspots associated with quadrotor path planning in unknown environments. The problems can be summarized as: 1) how to model an unknown area contaminated by radiation, 2) where to sense to model and localize radiation hotspots, and 3) how fast contaminated area could be characterized.

Various types of devices can be used to detect radiation, *e.g.*, Geiger counters, spectroscopy systems, etc. In nuclear search, the strength of the signal relative to noise (SNR) has been considered to measure the intensity of radioactive materials in a certain position, where the contaminated areas behave alike anisotropic field and it is very hard to sense and model the environment by *in-situ* active sensing. In fact, the sensor yields similar values for those areas which are not spatially distant. The problem turns into a more acute form, since cumulative effects of radioactive materials make it ungovernable to identify their individual existence by intensity values. However, to make the decision for exploring among all the areas A_T, where the robot ought to go in an unknown environment to get a significant sensing value difference so that we can avoid the problem of *in-situ* active sensing in anisotropic fields, we want to sense some areas A_S those are spatially distributed so that coverage area A would be minimized. Therefore, we want to develop an intelligent algorithm by which regions of interest could be converged into a minimum coverage area for rapid rescue mission.

The contribution of this paper can be summarized in the following points: 1) modeling of an anisotropic radiation field, 2) reduction of regions of interest to find the radiation hotspots, 3) on-line preliminary contour map building technique, 4) faster exploration algorithm for UAVs to localize radiation hotspots, and 5) a novel approach to the problems of localization, classification, and mapping in anisotropic fields.

10.2 Related Work

The problem of exploring an unknown environment with mobile robots has been studied intensively in the past. Dang [29] reported that learning an unknown environment is an online NP hard problem. However, those algorithms were discussed as offline algorithms in his paper. Later, online versions of those algorithms were proposed *e.g., online TSP, online Chinese postman problem, etc.* Rudlof [36] revived several techniques of graph theory for exploring an unknown environment defined as a fundamental problem of online robotics. He also outlined the category of graph depending on objective function. Monitoring spatial phenomena with sensor networks or mobile robots is often achieved by the concept of multi-robot exploration algorithms, *i.e.*, leader follower approach [27, 68], central coordination mechanism [1], frontiers [91] which are boundaries between explored and unexplored area. Another approach is based on the minimization of the energy consumption by robots to converge the area [62]. Inspiration from biology has also been accounted to solve this problem [26]. Furthermore, several authors proposed mutual information based path planning to monitor the environmental phenomena [10]. Bayesian frameworks have been further employed for rapid source localization and characterization [79]. This

is one of the best time optimal algorithms to monitor dynamic environments [22].

A key problem of robotic environmental sensing and monitoring is that of active sensing. [5] stated the problem of active sensing as a problem of controlling strategies applied to the data acquisition process which will depend on the current state of the data interpretation and the goal or the task of the process. To minimize the uncertainty in modelling and predicting an environmental phenomenon, Dolan *et. al.* [15] proposed Gaussian process-based anisotropic fields and they also reported that it can improve time efficiency while preserving near-optimal active sensing performance.

Recently, Krause*et al.* [56] discussed the problem of autonomous robotic exploration, automatic diagnosis, and activity recognition as artificial intelligence problems. They illustrated the effectiveness of sub-modular functions on problems of monitoring environmental phenomena in contrast to heuristic approaches which cannot provide performance guarantees. Furthermore, they also showed that the multiple robot problem can be reduced to optimizing paths for a single robot [76]. We have considered the problem of conducting rapid rescue mission in an unknown radiation contaminated environment. In our approach, which is basically inspired by the budget based greedy algorithm and Gaussian process-based anisotropic fields, a single robot tries to converge the radiation-contaminated area in two phases: In the learning phase, the robot discovers the highest intensity area to get experience about the intensity gradient level in the environment. Afterward in the execution phase, the robot tries to localize the hotspots of contaminated sources via actively sensing an unknown area.

10.3 System Models

Detection of radio active materials in an unknown environment is similar to create an online map in an unknown environment. Our focus is to provide accurate and energy-efficient localization of radioactive materials by reducing exploration areas from a given area. Based on the intensity value sensing, the coverage area for exploration can be converged to detect the most probable area of radioactive materials being leaked, enabling to create a real-time gradient map from the radiation field. Once the gradient map of radiation contaminated area is created, we can quickly narrow down our region of interest to further accurately localize the radio active materials. The total area (A_T) is sub-divided by several coverage areas ($A_i = \{A_1, A_2, A_3, ..., A_n\}$) that can be expressed using the graph theory $G = (V, E)$, where V, E are the vertex and edge of exploration area A_i. In this paper, we focus on the convergence of coverage area based on the maximum intensity level of radiation

$$A_T > \sum_{i=1}^{n} A_i \tag{10.1}$$

and the efficiency of exploration is defined by

$$E_f = \frac{\sum_{i=1}^{n} A_i | I}{A_T} \tag{10.2}$$

To converge the coverage area we have sub-divided the problems into the following two steps:

1) **Measure the intensity of coverage area:** The intensity of radiation for a specific area($I_j = \{I_1, I_2, I_3, ..., I_n\}$) is inversely proportional to the square of the distance from the source of radioactive material. Therefore, the relation between the intensity and distance motivates bringing the sensor to the source as closely as possible. However, our target is to classify different gradient layers depending on the highest intensity value obtained from the field and which layer is important to explore to find abrupt increments in radiation value (details follow later).

2) **Minimize the traveling distance to cover the area:** All the vertices of $G = (V, E)$ need to be explored to measure the intensity values of coverage area A_i. We have restricted the maximum sampling points denoted by N to be 5 ($N = \{N_1, N_2..., N_5\}$) around the robot position $P_i \in R^2$. If the environment is cluttered, the traveling sales man algorithm is implemented to minimize the exploration distance.

Let, for a given position P_i, the intensity effect be defined by

$$I = \left\{ \sum_{j=1}^{n} I_j | P_i \in R^{2.N} \right\} \tag{10.3}$$

Next, the UAV maintains the same orientation to sense the environment in an anisotropic field given by

$$\psi_{i+1} = \psi_i \tag{10.4}$$

While visiting neighbor nodes, the UAV derives the intensity gradient of each node with the same orientation. Among all of the nodes, the UAV stores the highest intensity node. The gradient of intensity in a specific neighbor point is defined by

$$\nabla I_{N_i} = \left[\frac{\partial I}{\partial x} \hat{i} + \frac{\partial I}{\partial y} \hat{j} \right] \tag{10.5}$$

Now, the gradient matrix of given coverage area can be given by

$$I_{A_i} = \begin{bmatrix} I_N^{X_1} & I_N^{Y_1} \\ I_N^{X_2} & I_N^{Y_2} \\ \vdots & \vdots \\ I_N^{X_5} & I_N^{Y_5} \end{bmatrix} = \nabla {I_{\max}}^i \tag{10.6}$$

However, to detect the contour shape in execution phase, besides the magnitude of intensity, we are rather intend to determine the direction of gradient which is defined by

$$u_i = k.dir\left(\nabla I_{N_i}\right) \tag{10.7}$$

where dir() function is defined as

$$dir\left(v\right) = \begin{cases} v/||v||_2 \\ 0 \end{cases}, \quad \begin{matrix} if||v||_2 \neq 0 \\ otherwise \end{matrix} \tag{10.8}$$

The phenomena could be explained from any arbitrary initial state, the UAV finds the gradient direction with respect to its local coordinate system. Changing sampling orientation is defined as

$$Change\_orientation\left(O_i\right) = \begin{cases} 1 \\ 1 \\ 0 \end{cases}, \quad \begin{matrix} Cost_{p_i} \rightarrow 0 \\ if\ A_i \notin T_A \\ otherwise \end{matrix}, \tag{10.9}$$

implying that the switching conditions depend on the geometry of area and the magnitude of intensity.

Furthermore, we attempt to minimize the coverage area compliant to the reduction of UAV flight time. To calculate the total UAV flight time required, we need to consider both the exploration time t_e and sensing time t_s. Cortex [13] described the procedure of taking radiation measurements in such a way that the robot stays in a cell (an exploration area is discretized by a large number of cells) until the desired variance threshold reaches. However, the sensing time includes the computation time as well. Hence, the total flight time t_f is obtained by

$$t_f = t_e + t_s \tag{10.10}$$

10.4 Algorithm Description

The main algorithm is subdivided by the following two functions: *i.e.*, *learning phase and executing phase.* Initially, the robot does not have *a priori* knowledge about the intensity gradient field and hence in the learning phase, it tries to get experience to radiation gradient level of a given area, A_T.

10.4.1 Learning phase

The objective of learning phase is to converge to the highest radiation region, where the intensity value is maximum. The learning phase has two sub-phases: *i.e.*, move to the maximum intensity region by sensing the environment in a spatially distributed way and classify the different intensity level. After finding the peak intensity of radiation field, the next objective is to classify the environment into different gradient layers. The areas colored differently in 10.3 indicate the different gradient layers. An effective classification of radiation field is a very difficult task without having *a priori* knowledge of radiation field. Therefore, we have a novel budget function to classify the gradient layer. This phase consists of the following sub-functions.

1 **Intensity sampling:** Let P_i be the coordinate of UAV and $\forall N_i$ are located at a uniform θ interval. The angle θ depends on the field of view of UAV given by

$$\theta = \frac{Field\_of\_view}{Sampling\_number} \tag{10.11}$$

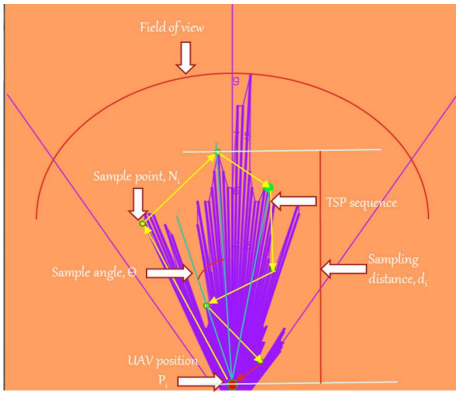


Figure 10.1: Spatially distributed sampling

The sampling point distance d_i is determined by the intensity value. The sampling distance also indicates the coverage area A_i. The sampling distance d_i is not the same for every neighbor point N_i, if there exists obstacles. The linear movement of sampling point will be stopped by the obstacle. We assume that for every sampling point, there exists a path among them. In this state the graph is K_5 which is a non-planar graph. Therefore, to use the Traveling Salesman Problem (TSP) algorithm, we converted this non-planar graph to a planar graph. TSP gives the shortest path to visit all the nodes at most once. If $\exists I_{N_i} > I_{P_i}$, the UAV moves to P_{N_i} for the specific index i.

$$\begin{aligned} & d_i \infty \frac{1}{I_{P_i}} \qquad\qquad where, \\ & \rightarrow d_i = k_1 \frac{1}{I_{P_i}} \qquad\quad k_1 = performance\ gain \end{aligned} \tag{10.12}$$

The value of K_1 indicates the performance speed. However, larger values of K_1 will result poor gradient map acquisition.

2 **Calculation of Coverage area:** It is obvious from Fig. 2 that there are four sequential triangles to make a coverage area A_i. The traveling distance between two consecutive nodes is defined by $t_i = \{t_1, t_2, t_3, t_4\}$

Hence,

$$A_i = \frac{1}{2}\left\{\sum_{i=1}^{4}(t_i \times d_i)\right\} \tag{10.13}$$

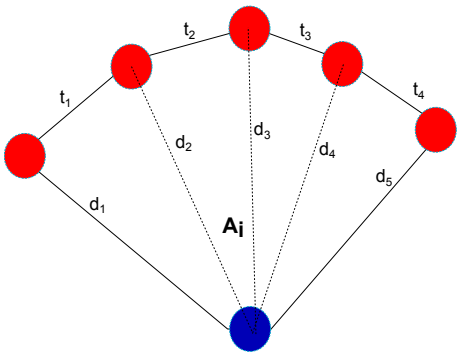


Figure 10.2: Coverage area

3 **Online Path Planning to minimize traveling distance:** TSP is basically an offline NP hard algorithm. Therefore, we developed an online version of the algorithm. If we focus on every single pixel captured by the vision system, it will increase the computational complexity a lot. Instead of looking every pixel, we want to follow the path through our sampling point and adjust d for valid exploration. Furthermore, TSP cannot provide an exact solution for huge numbers of samples. We therefore restricted our traveling points to be 5 sampling points. The main idea underlying for a given graph $G = (V, E)$, we do not know the order of the graph ($n = |V|$), and accordingly the total number of edges for the given graph is also unknown. Recalling Fig. 1, the UAV does not know the total number of nodes and edges before exploration, however for exploration to each A_i, the number of edges ($|E_{A_i}| = 6$) are fixed.

4 **Intensity calculation:** Radioactive materials are characterized by *anisotropic* fields that exhibit a higher spatial correlation along its forward direction than its perpendicular direction. Initially, we assume that the UAV starts to explore the environment, where the intensity of radiation is low, and as it further proceeds, the intensity will increase according to its property. Furthermore, we consider *Gaussian process based anisotropic field* for predicting the phenomena of spatially distributed intensity over the environment. To mitigate the problem of active sensing versus computational complexity trade offs, we only focus on the uncertainty to measure the intensity of spatially distributed sampling points. The covariance structure is defined as

$$\sigma_{p_i} = {\sigma_s}^2 \times E + {\sigma_n}^2 \delta_{p_i} \tag{10.14}$$

where

$$E = \exp\left\{-\frac{1}{2}\left(P_{N_{i+1}} - P_{N_i}\right)^T M^{-2}\left(P_{N_{i+1}} - P_{N_i}\right)\right\}$$

In a transect sampling in neighborhood points, σ_s and σ_n represent the signal and noise variance respectively. M is the diagonal matrix which indicates the correlation or similarity between measurements along the horizontal and vertical directions, and δ_{P_i} is the Kronecker delta of value 1 if $P_i = P_{N_i}$ and 0 otherwise.

Let the sampled location be denoted by s and a column vector z_s of corresponding measurements. The mean and covariance matrix of Gaussian process are defined as

$$I = \mu_{u|s} = \mu_u + \sum_{us} \sum_{ss}^{-1} (z_s - \mu_s) \tag{10.15}$$

$$\sum_{uu|s} = \sum_{uu} - \sum_{us} \sum_{ss}^{-1} \sum_{su}, \tag{10.16}$$

where μ_u or μ_s is a column vector with mean components for every neighbor position P_{N_i}, and, for corresponding location, $\sum_{us}$ or $\sum_{ss}$ represents a covariance matrix with covariance components σ_{P_i} and $\sum_{su}$ is the transpose of $\sum_{us}$.

Eq. 10.15 predicts the sensor values uncertainty of unobserved locations, whereas Eq. 10.16 yields the quantification of its prediction. Finally, from Eq. 10.16, we developed the gradient matrix ∇I based on the criteria that we discussed on system models.

5 **Concave problem solver:** We assume that the intensity of the radiation field will be high at a certain position and afterward it will decrease similar to the concave problem. Therefore, a typical greedy algorithm can be implemented to find the peak intensity value of radiation field but it also calculates upper bounds for further comparison. During environmental mapping, Gaussian normal distribution is assumed to predict the next step which is closed under linear transformation.

The maximum intensity of a coverage area among five sampling positions is recorded as ${I_{\max}}^i$, where i indicates the step number.

If $X_1 = {I_{\max}}^1$ and $X_2 = {I_{\max}}^2$ are the first and second observed values, respectively, then the prediction of next observation value of $\hat{X_3} \epsilon {I_{\max}}^3$ can be obtained by [66] as follows:

$$\hat{X_3} = \frac{aX_1 + bX_2 - (a+b)\,\mu}{\sqrt{a^2 + b^2}} + \mu, \tag{10.17}$$

where μ is the mean of observed values and a, b are arbitrary real numbers. However, we adjust the values of a, b after sensor measurements and remember those for each step.

Theorem 1: *Greedy algorithm gives a single solution of* (a, b) *to express the linear relationship to* $\hat{X_3}$ *with* (X_1, X_2).

Proof: Assume that A_1, A_2, A_3 have the same intensity distribution since they belong to the same gradient layer. Given (X_1, X_2) previous intensity values, where $\left(X_1 = {I_{\max}}^1, X_2 = {I_{\max}}^2\right)$. $\hat{X_3}$ can be measured after reaching at A_3. μ is assumed to be equal to the maximum intensity value recorded in each coverage area ${I_{\max}}^i$. Hence, we can obtain the values of a and b as follows:

We assumed that initially the robot is at a position with low intensity level and as the robot proceed toward the inside of A_T through Y-Axis, the intensity value is supposed to be higher. After covering an area, the maximum intensity value is

Algorithm 7 Greedy algorithm for determining relationship

1: **for** $a = 1; \hat{X}_3 \geq {I_{max}}^3; a++$ **do**
2: **for** $b = 1; b < 10 || \hat{X}_3 \geq {I_{max}}^3; b++$ **do**
3: $\hat{X}_3 = \frac{aX_1 + bX_2 - (a+b)\mu}{\sqrt{a^2+b^2}} + \mu$ ▷ Estimating(a, b)
4: **end for** ▷ ${I_{max}}^3$ is measured value
5: **end for**

recorded and attached to the gradient level. Proceeding through the phase that indicates moving toward the maximum intensity value will engender a frontier where the robot would be trapped by the highest intensity region. However, it would be computationally expensive to identify the deadlock region by comparing all the traversed region. Therefore, immediate consecutive two previous regions are accounted to determine the deadlock region.

$$Deadlock = \begin{cases} 1 & if\ A_i \in \{A_{i-1} \cup A_{i-2}\} \\ 0 & otherwise \end{cases}$$

6 **Identification of gradient layer:**

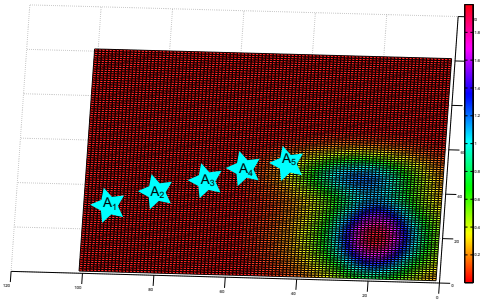


Figure 10.3: Intensity gradient level due to radiation

Recalling the previous section, A is the set of area explored and $I(A_i)$ is a function to calculate the maximum intensity for the intended area explored. However, the greedy algorithm discussed in the concave solver is not only useful to find the peak position, but also helpful to predict and to estimate errors of prediction field. Furthermore, classification can be solved by comparing the intensity distribution of areas rather than comparing the maximum intensity value.

Theorem 2: *Assuming that* $I(A_i) = {I_{\max}}^i$; *where,* $i = (1\ to\ 5)$; *therefore* A_4 *would have same distribution if* $X_4 > I(A_4)$ *can be estimated by a and b which are calculated in previous step.*

Proof: Rewriting Eq. (10.17) as follows

$$\begin{array}{l} \left(\sqrt{a_1^2+b_1^2}\right)\hat{X}_3 = a_1X_1 + b_1X_2 - (a_1+b_1-1)\mu_i \\ C_i\hat{X}_i = a_{i-2}X_{i-2} + b_{i-2}X_{i-1} - D_{i-1}(generalized) \end{array}, \tag{10.18}$$

where C_i means the gain of $\hat{X}_3$ which belongs to real number and $C > 0$; D_i is the stability of distribution. If $\forall X_i \subset$same distribution, incremental $|D_i|$ indicates the higher stability. However, the property of stability determines the distribution which is either comparable or not. If $\hat{X}_4$ has higher stability, we can conclude that the comparison is acceptable to explain the linear relationship of $\hat{X}_4$ to previous related values. The value of C_i indicates the element of budget set $\mathbb{B}$ for exploring an area which has the same intensity distribution.

$$\hat{X}_4 = \frac{a_2X_2 + b_2X_3 - D_3}{C_3} \;\; where,\; i = 4 \tag{10.19}$$

If the explored area has a different intensity distribution (*e.g.* A_5) compared to others, the gradient of budget $F(\mathbb{B})$ will be changed to compensate the different value of $\hat{X}_5$. The gradient of budget is defined as

$$F(\mathbb{B}) = \nabla(\mathbb{B}) = (C_{i+1} - C_i) \tag{10.20}$$

Now we can conclude that the budget gradient will be different whenever any of previous coverage areas exist on different distribution. Classification of distribution decision is taken as follows:

$$Same\_distribution = \begin{cases} 1 & if\ F(\mathbb{B}) > 0 \\ 0 & otherwise \end{cases} \tag{10.21}$$

However, the change of intensity level also indicates the change of gradient level in the continuous domain. Therefore, we have to consider the gradient of budget function to classify different gradient layers. For finding the gradient level of unknown environments, the robot needs to move through the same gradient level around the environment. Therefore, the learning phase assigns an intensity value and starting position of gradient level to the executing phase.

10.4.2 Executing phase

In the learning phase, the robot will

- **[1] Control:** values of (a, b)
- **[2] Calculate:** the gradient of budget $F(\mathbb{B})$
- **[3] Observe:** the stability D_i

However, in the executing phase, our objective is to find the specific contour shape. Therefore, the robot would choose that gradient layer to explore where the intensity drastically increases among the classified layers. The classified gradient layer is considered as local maxima and by taking derivative of local maxima we can easily achieve a global maximum layer. Furthermore, in this phase, the robot does not calculate the real intensity value rather it tries to follow a fixed intensity value. Therefore, the intensity measured function turns out to be the cost function in this phase. After the learning phase, the robot memorizes the starting index of position P_i for different gradient levels, as a result, it is easier in the executing phase to move through the layer which has the same intensity value with a constant variance. To model this environment, we define the cost function in the form of Gaussian normal distribution.

$$cost_{P_i} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu)}{2\sigma^2}\right\} + W_c \times D_i, \tag{10.22}$$

where the mean, μ is the recorded intensity value assigned from the learning phase, and the variable x is the current intensity value. The standard deviation σ is the difference between the assigned layer and immediate previous layer in terms of intensity value. D is the tability of measured cost that can be calculated by Eq. (10.17). W_c is the weighted constant value.

Furthermore, to explore through the assigned gradient level, diagonal movements are preferable in contrast to back and forth movements since during back and forth movements, it may be trapped in deadlock regions due to the limited intensity boundary. Therefore, diagonal samples are weighted by multiplying with some arbitrary constants. The first part of cost function (before weighted constant part) will determine the position which has a similar intensity compared to the assign gradient layer. The second part of cost function (weighted by stability variable) will correct the estimation in terms of sensor accuracy. As the phenomena could be explained among the five candidates positions, the robot will choose the next position not only based on maintaining similar intensity values but also accounting the resolution of gradient fields.

Therefore, we may summarize that the first part of cost function will provide the gradient layer that would be filtered by the second part of cost function.

10.5 Simulation result

We have verified our algorithm using the V-rep simulator, where the open dynamics engine has been used to obtain the simulation field data. The robot vision is considered as 180 degree where 5 samples point are taken at 35 degree angle interval. The sampling distance can be estimated by the total area and expected number of coverage areas to model the environment. However, too small sampling distance cannot yield a significant difference in sensing values, whereas too large sampling step may avoid detecting intermediate gradient layers. Hence, the sampling distance is determined empirically. Since there is no obstacle in this simulation, our online TSP always yields the same sequence to explore. The UAV has to keep the same orientation to sense every sampling vertex in the learning

phase, because the correlation matrix M cannot produce relevant outputs with respect to different orientations of UAV. The environment is modeled by Gaussian distribution, where three radioactive materials exist in three different directions that yield cumulative effects on the environment. Since the detection solely depends on the intensity value, we can draw a contour depending on the gradient layer expected to cover radioactive materials. We set up a $20 \times 20 meter$ floor, where three same effect of radioactive materials exist in different directions. The UAV starts to explore the environment from Y-Axis in the learning phase, and find the contour in X axis in the executing phase. The learning phase is ended by the deadlock situation and Table 1 shows the normalized data sets from the simulation field. Every coverage area remains the same due to absence of obstacles in this simulation. Hence, without calculating the coverage area, we rather remember the index number of coverage area. At least three steps are required to predict forthcoming state, hence the index of coverage area as well as other parameters start after initial three steps. The gradient layer boundaries are determined by the negative value of $\frac{dC}{dA_i}$ which indicates the local maximum budget from the budget set $F(\mathbb{B})$ with respect to the coverage area. Finally, contour exploration execution decision has been made by taking one more derivation among the local maximum budget $i.e. \frac{d^2C}{{dA_i}^2}$. Plotting the data, the following graph can be obtained.

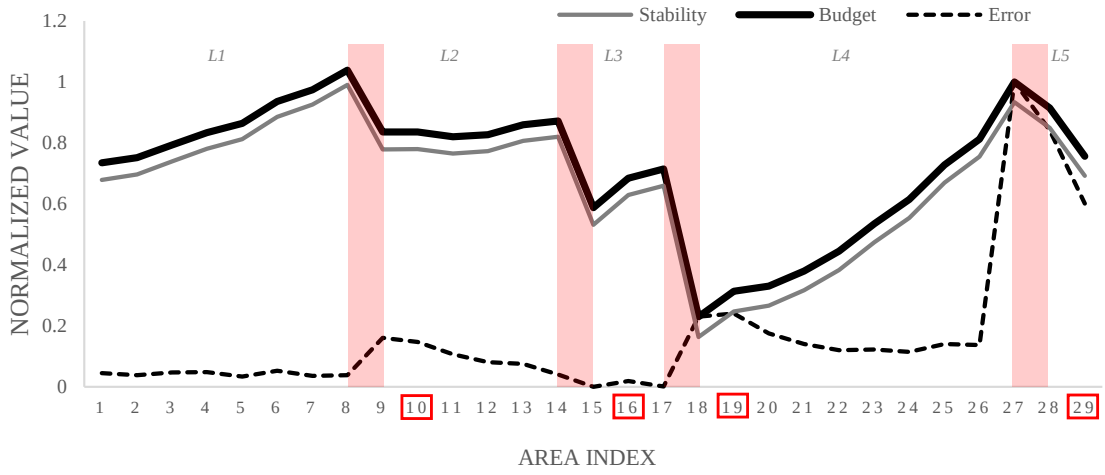


Figure 10.4: **Analysis of radiation field graph:** gray line- stability of measurement, bold black -line budget, dashed black line -error of estimation, red rectangular area- layer boundary, red boxed number -subjected area

The budget is calculated from previous consecutive two observations values. When the budget is not satisfied to predict the field state, *i.e.*, the budget value is negative, the gradient layer is supposed to change at the previous area. Table 1 clarifies the idea of gradient layer detection. After the learning phase, 5 gradient layers have been classified that could be explained by the analysis of radiation field graph and Table 1.

Table 10.1: Boundary layer detection

Area Index (A_i)	**Budget gradient** ($F(\mathbb{B})$)	**Max budget** $\left(\frac{d^2C}{dA_i{}^2}\right)$
9	-0.324	0
15	-0.243	-0.081
18	-0.713	0.470
28	-0.675	-0.038

From the area index 8 to 9, the budget slope decreases by 0.32461 that indicates the boundary region (red rectangular area) of previous gradient layer and after entering into the new gradient layer area 10, the slope increases again. Thus, the rest of the layers follow the same rule. After taking one more derivative of budget from identified gradient layers, we can determine which layer is more important for the executing phase. The layer boundary at area index 18 shows the peak value of budget among others, as a result the gradient layer boundary around the area index 17 and 18 will be further supervised to determine the full shape of contour. After detecting the contour in the execution phase, we have compared our gradient layer detection with the simulation result of Matlab [20].

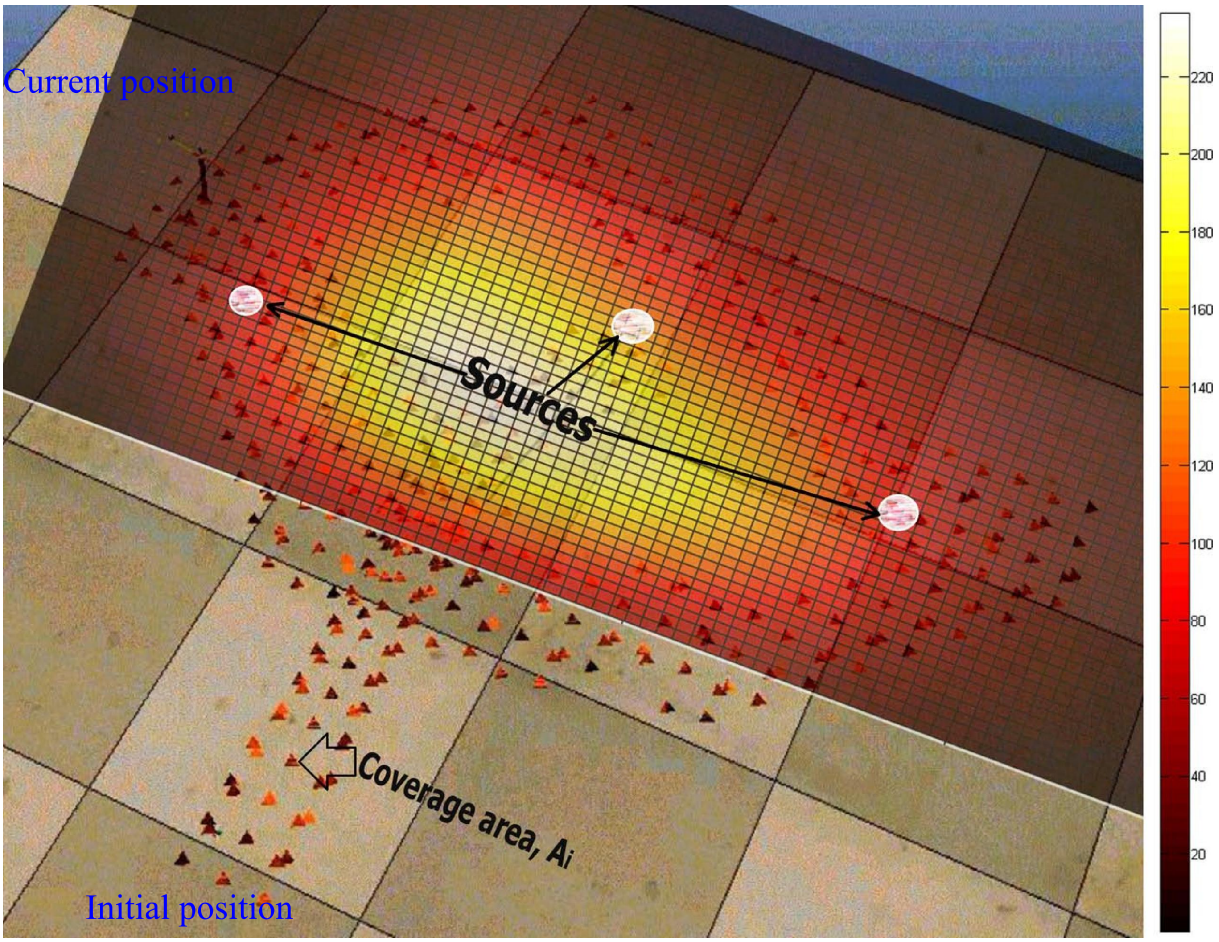


Figure 10.5: Simulation result and intensity contour map

Fig. 10.5 shows the combined pictorial view of our simulation field and Matlab graph. For given setup of radioactive material, Matlab shows five prominent gradient layers, *i.e.* black, red, orange, yellow, and white. The area index between 17-18 exists on the red boundary layer. Therefore, in the executing phase, the gradient layer classified by the red color is subjected to further explore to determine its contour shape. However, after discovering the contour, we have seen that all three radioactive materials have covered by the contour. It is explicitly obvious from Fig. 10.5 that the coverage area A_i marked by separated orange triangular shapes are significantly lesser than the total area A_T.

Conclusion and future direction

This chapter addressed the problem of teleoperated and semi-autonomous flight control of a quadrotor UAV. It was conducted as a research project at the School of Information Science, Japan Advanced Institute of Science and Technology(JAIST). Detailed mathematical modeling of the quadrotors kinematics and dynamics was provided. A modular PID approach was proposed for the semi-autonomous control of quadrotors in general, without the need for a precise mathematical model of their complex and ill-defined dynamics. Although we have received 6 axis motion information, without having an indoor GPS system we can not control the quadrotor's position perfectly. The future work is directed towards achieving fully autonomous flight in indoor environments. Therefore, we would focus on indoor localization system which is a fundamental problem in the field of robotics. To solve this problem vision based localization would be considered where multiple Kinect would be implemented to avail the global position of UAV. Moreover, surveillance in unknown indoor environments is a challenging mission, since substantially more compact spaces and obstacles exist compared to spacious outdoor environments. The proposed first path planning algorithm offers one of the key technologies for low-cost surveillance UAVs in complex, cluttered areas ensuring low computational complexity. In addition, this algorithm envisions a new direction for online path planning, based on the fact that the obstacle does not always hinder us from reaching a goal position, rather sometimes it is helpful to reach a goal position easily. To recapitulate, we may conclude that first work proposed a universal path planning algorithm of quadrotor UAVs equipped with limited range sensors and computational resources, particularly for small area surveillance purposes. However, our second path planning algorithm eventually based on aerial flocking in cluttered environments is challenging due to limited hardware resources and proper swarm behaviors. As sticking to neighbor robots is not efficient in terms of overall team maneuvering, a minimal internal communication scheme was proposed to increase the team efficiency, where the triangular geometry offered better network connectivity and coverage density. Furthermore, to cope with computational intractability for on-board real-time computation, we implemented the exploration priority based approach yielding a new aerial flocking controller for low-cost UAVs. Finally, third path planning work, we have focused on converging our region of interest to detect and identify radioactive materials for rapid rescue missions. The problem has been treated as the gradient layer classification and identification of layer contour shape problem. We have classified the radiation field into four different major gradient layers which were very similar to Matlab numerical classification. Furthermore, we have also identified the most important layer from which the radiation has drastically increased. Our method offers faster convergence of areas of interest by partially mapping the environment, and effectively overcomes the problem of sequential exploration over a whole area to model, map, and characterize the environment.

Bibliography

[1] Mohammad Al Khawaldah, Ibrahim Al-Adwan, and Khaled Eid. New exploration strategy for cooperative mobile robots. In *Proceedings of the 11th WSEAS international conference on Electronics, Hardware, Wireless and Optical Communications, and proceedings of the 11th WSEAS international conference on Signal Processing, Robotics and Automation, and proceedings of the 4th WSEAS international conference on Nanotechnology*, pages 149–154. World Scientific and Engineering Academy and Society (WSEAS), 2012.

[2] Abdel Ilah Alshbatat, Ashar Khamaisa, and Mais Khreisat. Adaptive control system for an autonomous quadrotor unmanned aerial vehicle. *AWERProcedia Information Technology and Computer Science*, 1, 2012.

[3] Erdinc Altug, James P Ostrowski, and Camillo J Taylor. Quadrotor control using dual camera visual feedback. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4294–4299. IEEE, 2003.

[4] Boris Andrievsky, Alexander Fradkov, and Dimitri Peaucelle. Adaptive control experiments for laas" helicopter" benchmark. In *2005 International Conference on Physics and Control, PhysCon 2005*, pages 760–765, 2005.

[5] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[6] Lotfi Beji, Azgal Abichou, and KM Zemalache. Smooth control of an x4 bidirectional rotors flying robot. In *Robot Motion and Control, 2005. RoMoCo'05. Proceedings of the Fifth International Workshop on*, pages 181–186. IEEE, 2005.

[7] Richard Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[8] Luca F Bertuccelli and JP How. Robust uav search for environments with imprecise probability maps. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 5680–5685. IEEE, 2005.

[9] Jonathan Binney, Andreas Krause, and G Sukhatme. Informative path planning for an autonomous underwater vehicle. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4791–4796. IEEE, 2010.

[10] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Informative path planning for an autonomous underwater vehicle. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4791–4796. IEEE, 2010.

[11] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 6. Macmillan London, 1976.

[12] Samir Bouabdallah, Pierpaolo Murrieri, and Roland Siegwart. Design and control of an indoor micro quadrotor. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4393–4398. IEEE, 2004.

[13] Samir Bouabdallah, Andre Noth, and Roland Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2451–2456. IEEE, 2004.

[14] C. M Breder. Equations descriptive of fish schools and other animal aggregations. 1954.

[15] Nannan Cao, Kian Hsiang Low, and John M Dolan. Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 7–14. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[16] Zhiqiang Cao, Min Tan, Shuo Wang, Yong Fan, and Bin Zhang. The optimization research of formation control for multiple mobile robots. In *Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on*, volume 2, pages 1270–1274. IEEE, 2002.

[17] Brian J Capozzi. *Evolution-based path planning and management for autonomous vehicles*. PhD thesis, University of Washington, 2001.

[18] Pedro Castillo, Alejandro Dzul, and Rogelio Lozano. Real-time stabilization and tracking of a four-rotor mini rotorcraft. *Control Systems Technology, IEEE Transactions on*, 12(4):510–516, 2004.

[19] Pedro Castillo, Rogelio Lozano, and Alejandro Dzul. Stabilization of a mini-rotorcraft having four rotors. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2693–2698. IEEE, 2004.

[20] Pedro Castillo, Rogelio Lozano, and Alejandro Dzul. Stabilization of a mini rotorcraft with four rotors. *IEEE Control Systems Magazine*, 25(6):45–55, 2005.

[21] JE Colgate. Coupled stability of multiport systems-theory and experiments. *Journal of dynamic systems, measurement, and control*, 116(3):419–428, 1994.

[22] R Cortez, Xanthi Papageorgiou, H Tanner, A Klimenko, K Borozdin, Ron Lumia, and W Priedhorsky. Smart radiation sensor management. *Robotics & Automation Magazine, IEEE*, 15(3):85–93, 2008.

[23] Cosmin Coza and CJB Macnab. A new robust adaptive-fuzzy control method applied to quadrotor helicopter stabilization. In *Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American*, pages 454–458. IEEE, 2006.

[24] Xiaohui Cui, C. Tim Hardin, Rammohan K. Ragade, and Adel Said Elmaghraby. A swarm approach for emission sources localization. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 424–430. IEEE, 2004.

[25] Luca De Filippis, Giorgio Guglieri, and Fulvia Quagliotti. Path planning strategies for uavs in 3d environments. *Journal of Intelligent & Robotic Systems*, 65(1-4):247–264, 2012.

[26] F De Rango and N Palmieri. Atrc: A swarm-based robot team coordination protocol for mine detection and unknown space discovery. In *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012 International Symposium on*, pages 1–7. IEEE, 2012.

[27] Floriano De Rango and Nunzia Palmieri. A swarm-based robot team coordination protocol for mine detection and unknown space discovery. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pages 703–708. IEEE, 2012.

[28] Carlos Delgado-Mata, Jesus Ibanez Martinez, Simon Bee, Rocio Ruiz-Rodarte, and Ruth Aylett. On the use of virtual animals with artificial fear in virtual environments. *New Generation Computing*, 25(2):145–169, 2007.

[29] Xiaotie Deng, Tiko Kameda, and Christos Papadimitriou. How to learn an unknown environment. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 298–303. IEEE, 1991.

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

[31] Draganflyer. Rctoys, 2013.

[32] J Dunfied, M Tarbouchi, and G Labonte. Neural network based control of a four rotor helicopter. In *Industrial Technology, 2004. IEEE ICIT'04. 2004 IEEE International Conference on*, volume 3, pages 1543–1548. IEEE, 2004.

[33] Matthew G Earl and Raffaello D'Andrea. Real-time attitude estimation techniques applied to a four rotor helicopter. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, pages 3956–3961. IEEE, 2004.

[34] Gary Fay. Derivation of the aerodynamic forces for the mesicopter simulation, 2001.

[35] Dave Ferguson and Anthony Stentz. Using interpolation to improve path planning: The field d* algorithm. *Journal of Field Robotics*, 23(2):79–101, 2006.

[36] Rudolf Fleischer and Gerhard Trippen. Exploring an unknown graph efficiently. In *Algorithms–ESA 2005*, pages 11–22. Springer, 2005.

[37] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Pattern formation by autonomous robots without chirality. In *Proc. 8th Int. Colloquium on Structural Information and Communication Complexity*, pages 147–162. Citeseer, 2001.

[38] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[39] LR Ford and Delbert Ray Fulkerson. *Flows in networks*, volume 3. Princeton Princeton University Press, 1962.

[40] Marc Freese, Surya Singh, Fumio Ozaki, and Nobuto Matsuhira. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 51–62. Springer, 2010.

[41] Nicolas Guenard, Tarek Hamel, and Robert Mahony. A practical visual servo control for an unmanned aerial vehicle. *Robotics, IEEE Transactions on*, 24(2):331–340, 2008.

[42] Giorgio Guglieri. Optimal trajectory tracking for an autonomous uav. *Automatic Control in Aerospace*, 1:1–9, 2008.

[43] Daniel Gurdan, Jan Stumpf, Michael Achtelik, K-M Doth, Gerd Hirzinger, and Daniela Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 361–366. IEEE, 2007.

[44] Robyn Lynn Harmon. *Aerodynamic modeling of a flapping membrane wing using motion tracking experiments*. ProQuest, 2008.

[45] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[46] Christopher Hartman and Bedrich Benes. Autonomous boids. *Computer Animation and Virtual Worlds*, 17(3-4):199–206, 2006.

[47] Charlotte K Hemelrijk and Hanno Hildenbrandt. Some causes of the variable shape of flocks of birds. *PLoS One*, 6(8):e22479, 2011.

[48] Geoffrey Hollinger, Sanjiv Singh, Joseph Djugash, and Athanasios Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201–219, 2009.

[49] Stefan Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft uavs. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 807–814. IEEE, 2008.

[50] Myungsoo Jun and Raffaello DAndrea. Path planning for unmanned aerial vehicles in uncertain and adversarial environments. In *Cooperative Control: Models, Applications and Algorithms*, pages 95–110. Springer, 2003.

[51] Boyoon Jung and Gaurav S Sukhatme. Tracking targets using multiple robots: The effect of environment occlusion. *Autonomous robots*, 13(3):191–205, 2002.

[52] Dongwon Jung and Panagiotis Tsiotras. On-line path generation for small unmanned aerial vehicles using b-spline path templates. In *AIAA Guidance, Navigation and Control Conference, AIAA*, volume 7135, 2008.

[53] Seema Kamath, Eric Meisner, and Volkan Isler. Triangulation based multi target tracking with mobile sensor networks. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3283–3288. IEEE, 2007.

[54] Sven Koenig and Maxim Likhachev. Incremental a*. In *NIPS*, pages 1539–1546, 2001.

[55] Sven Koenig and Maxim Likhachev. D* lite. In *AAAI/IAAI*, pages 476–483, 2002.

[56] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, volume 7, pages 1650–1654, 2007.

[57] KN Krishnanand, P Amruth, MH Guruprasad, Sharschchandra V Bidargaddi, and Debasish Ghose. Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 958–963. IEEE, 2006.

[58] Geunho Lee and Nak Young Chong. A geometric approach to deploying robot swarms. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):257–280, 2008.

[59] T Madani and A Benallegue. Backstepping sliding mode control applied to a miniature quadrotor flying robot. In *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, pages 700–705. IEEE, 2006.

[60] Tarek Madani and Abdelaziz Benallegue. Backstepping control for a quadrotor helicopter. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3255–3260. IEEE, 2006.

[61] Tarek Madani and Abdelaziz Benallegue. Control of a quadrotor mini-helicopter via full state backstepping technique. In *Decision and Control, 2006 45th IEEE Conference on*, pages 1515–1520. IEEE, 2006.

[62] Yongguo Mei, Yung-Hsiang Lu, CS George Lee, and Y Charlie Hu. Energy-efficient mobile robot exploration. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 505–511. IEEE, 2006.

[63] Najib Metni, Tarek Hamel, and François Derkx. Visual tracking control of aerial robotic systems with adaptive depth estimation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 6078–6084. IEEE, 2005.

[64] V Mistler, A Benallegue, and NK M'sirdi. Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 586–593. IEEE, 2001.

[65] Abdellah Mokhtari and Abdelaziz Benallegue. Dynamic feedback controller of euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2359–2366. IEEE, 2004.

[66] Yannick Morel and Alexander Leonessa. Direct adaptive tracking control of quadrotor aerial vehicles. In *Florida Conference on Recent Advances in Robotics*, pages 1–6, 2006.

[67] Alex Nash, Kenny Daniel, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1177. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[68] Hoa G Nguyen, Narek Pezeshkian, Anoop Gupta, and Nathan Farrington. Maintaining communications link for a robot operating in a hazardous environment. Technical report, DTIC Document, 2004.

[69] Ioannis K Nikolos, Kimon P Valavanis, Nikos C Tsourveloudis, and Anargyros N Kostaras. Evolutionary algorithm based offline/online path planner for uav navigation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(6):898–912, 2003.

[70] M. Oborne. arducopter, 2012.

[71] OJ O'Loan and MR Evans. Alternating steady state in one-dimensional flocking. *Journal of Physics A: Mathematical and General*, 32(8):L99, 1999.

[72] Arvind A Pereira and Gaurav S Suhatme. Minimum-risk time-expanded planning for auvs using ocean current predictions. 2012.

[73] Barbara Pfeiffer, Rajan Batta, Kathrin Klamroth, and Rakesh Nagi. Path planning for uavs in the presence of threat zones using probabilistic modeling. *Institute of Applied Mathematics, University of Erlangen, Erlangen*, 2005.

[74] Nicholas Roy, Geoffrey J Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *J. Artif. Intell. Res.(JAIR)*, 23:1–40, 2005.

[75] S Salazar-Cruz, A Palomino, and R Lozano. Trajectory tracking for a four rotor mini-aircraft. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 2505–2510. IEEE, 2005.

[76] Amarjeet Singh, Andreas Krause, Carlos Guestrin, William J Kaiser, and Maxim A Batalin. Efficient planning of informative paths for multiple robots. In *IJCAI*, volume 7, pages 2204–2211, 2007.

[77] John R Spletzer and Camillo J Taylor. Dynamic sensor planning and control for optimally tracking targets. *The International Journal of Robotics Research*, 22(1):7–20, 2003.

[78] Anthony Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical report, DTIC Document, 1993.

[79] Piyush M Tagadea, Byeong-Min Jeonga, and Han-Lim Choi. A gaussian process emulator approach for rapid contaminant characterization with an integrated multizone-cfd model. *arXiv preprint arXiv:1305.2565*, 2013.

[80] H.G. Tanner, A. Jadbabaie, and G.J. Pappas. Stable flocking of mobile agents, part i: fixed topology. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 2, pages 2010–2015 Vol.2, 2003.

[81] A Tayebi and S McGilvray. Attitude stabilization of a four-rotor aerial robot. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1216–1221. IEEE, 2004.

[82] Abdelhamid Tayebi and Stephen McGilvray. Attitude stabilization of a vtol quadrotor aircraft. *Control Systems Technology, IEEE Transactions on*, 14(3):562–571, 2006.

[83] Glenn P Tournier, Mario Valenti, Jonathan P How, and Eric Feron. Estimation and control of a quadrotor vehicle using monocular vision and moire patterns. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 21–24, 2006.

[84] Mario Valenti, Brett Bethke, Jonathan P How, D Pucci de Farias, and John Vian. Embedding health management into mission tasking for uav teams. In *American Control Conference, 2007. ACC'07*, pages 5777–5783. IEEE, 2007.

[85] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229, 1995.

[86] Mathukumalli Vidyasagar. *Nonlinear systems analysis*, volume 42. Siam, 2002.

[87] Stephen Warshall. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12, 1962.

[88] Steven Lake Waslander, Gabriel M Hoffmann, Jung Soon Jang, and Claire J Tomlin. Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3712–3717. IEEE, 2005.

[89] Stephen Waydo and Richard M Murray. Vehicle motion planning using stream functions. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2484–2491. IEEE, 2003.

[90] Jan C Willems. Dissipative dynamical systems part i: General theory. *Archive for rational mechanics and analysis*, 45(5):321–351, 1972.

[91] Kai M Wurm, Cyrill Stachniss, and Wolfram Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1160–1165. IEEE, 2008.

[92] Kwangjin Yang and Salah Sukkarieh. 3d smooth path planning for a uav in cluttered natural environments. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 794–800. IEEE, 2008.

[93] Kadda Meguenni Zemalache, Lotfi Beji, and H Marref. Control of an under-actuated system: application a four rotors rotorcraft. In *Robotics and Biomimetics (ROBIO). 2005 IEEE International Conference on*, pages 404–409. IEEE, 2005.