INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Obstacle Avoidance Methods in UAVs

ZUBAIR AHMED KHAN
junho de 2019

POLITÉCNICO
DO PORTO

Master's Thesis on Autonomous Systems

Obstacle Avoidance Methods in UAV's

by

Zubair Khan

Dissertation submitted to obtain the
degree of Masters in Autonomous Systems

Department of Electrical and Computer Engineering

June 2019

Acknowledgements

Any achievement big or small should have a catalyst and constant encouragement and advice of valuable and noble minds. The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. I would like to express our deep sense of gratitude to our President Maria Joo Viamonte for her continuous effort in creating a competitive environment in our university and encouraging throughout this course. I would like to thank sincerely to my beloved Course Director professor Cecilia Reis, for her invaluable guidance, support and endurance. I would like to convey heartful thanks to my guide professor Andre Dias, for giving me the opportunity to embrace upon this topic and for his continuous encouragement throughout the preparation.

I also wish to thank all the staff members of the course Autonomous Systems who have helped us directly or indirectly for the completion of our project successfully. Finally, I am thankful to my parents, friends and loved ones for their continued moral and material support throughout the course and in helping me to finalize the project.

Abstract

We contributed a method for avoiding obstacles using monocular vision as the only sensor in UAV (Unmaned Aerial vehicle). The vision based ROS (Robotic operating system) node detects the known obstacles in front of the UAV. Unknown obstacles can be taken care of by adding he information of all the obstacles seen in the scene to a map. The distance to obstacle in this research is calculated by just increasing size of the obstacle in front of the UAV. The image processing libraries were used from OpenCV to do thresholding, noise removal and contours detection. This research also tests and evaluate the path planning of UAV using MoveIt architecture, and evaluates the different results obtained.Hence we show the effectiveness of the monocular vision and size as a constraint algorithm in UAVs to detect and avoid frontal obstacles.

Keywords : ROS ,UAV, OpenCV, MoveIt, Thresholding, Contours, monocular vision.

Declaration of Authorship

I Zubair Ahmed Khan declares, under a commitment of honor, that this work is original and that all non-original contributions were duly referenced, with identification of the source.

June 2019

Signature

Contents

List of Figures

Abbreviations

CPU - Central Processing Unit

GPS - Global Positioning System

IMU - Inertial Measurement Unit

LIDAR - Light Detection and Ranging

MOPS - Multi Scale Oriented Patches

OSRF - Open Source Robotic Foundation

RGB - Red Green Blue

ROI - Region Of Interest

ROS - Robot Operating System

SDF - Spatial Data Format File

SIFT - Scale Invariant Feature Transform

SLAM - Simultaneous Localization and Mapping

SONAR - Sound Navigation and Ranging

SURF - Speed-up Roburst Features

UAV Unmanned Arial Vehicle

URDF - Unified Robot Description Format

YAML - Yet Another Markup Language

Nomenclature

ρ - Length scale for a matrix

D - Distance between obstacle and UAV (m)

D' - Calculated distance between obstacle and UAV (m)

F - Focal length of the camera (mm)

P - Pixel size of for the given width of image (pixel)

W - Width of the obstacle (inch)

B - 5x5 Gaussian Filter

$centroid\_x$ - X co-ordinate of centroid for Contours

$centroid\_y$ - Y co-ordinate of centroid for Contours

G - Edge Gradient

Gx - First derivative for Canny edge detection in horizontal direction

Gy - Second derivative for Canny edge detection in vertical direction

H_{ij} - The Gaussian kernel Formula

M_10 - Image Moment for image pixel x

θ - Direction of edge

Chapter 1

Introduction

The ability of birds, flies to detect and avoid obstacles in very outdoor environment like forests is fascinating and have been eventually one of the motivation for deep study in obstacle detection and avoidance. However for UAV's this includes many applications like search and rescue, information gathering from an area. To accomplish this many active like LIDAR. Few interesting application of drones are in search and rescue operations, inspection and all these application require a very challenging feature which is called Obstacle avoidance. Having said that, its also important to know the environment to add this feature. For example Depth camera will not necessarily work in complex environment like forest and also where the obstacle is in direct exposure to the light.

To solve this problem since all UAV's are equipped with camera and inertial measurement unit, we can use these to detect and avoid obstacles in front of the drone. Which would result in no extra payload and cost effective solution to the problem of obstacle avoidance. In this research we will be focusing mainly on the simplest approach to avoid Obstacle, as we all are inspired by our nature, I would like to remind you how most of the flies and also we humans deals with obstacles.

It is very common to talk about distance when it comes to obstacle avoidance and related research and which seems to be very logical due to the availability of depth cameras and even mimic or find depth using two cameras. Although its a fact that we humans do not calculate distance while moving towards obstacles

or vice verse. What we do is imagine the size of the obstacle in front rather, to determine how far or near the obstacle is. Which make the solution even more realistic and smarter in terms of cost effectiveness and weight on the UAV as we will just require a single camera now to achieve this. Having said that, we will in our research study about various techniques for obstacle avoidance and then implement our approach as a solution to this problem.
The objective of this thesis is to study the available methods for obstacle avoidance which can be applied in UAV's. The objective also includes proposing a new method related to obstacle avoidance and to compare advantage of the proposed method with the obstacle avoidance state of art techniques.

1.1 What is Obstacle Detection and Avoidance Technology

For any vehicle whether it is UAV, robot or even an autonomous car, to detect objects and then perform avoidance task like to go around, stop or even go above needs complex technologies working together to form an integrated system [6]. For this to happen many complex approaches are requires like mathematical modeling, machine learning and aspects of SLAM technology.
Various obstacle avoidance sensors includes Stereo vision, Ultrasonic (SONAR), Time of Flight, LIDAR, Infrared and Monocular vision [6].

1.2 Why Obstacle Avoidance

Obstacle avoidance in a drones can prevent drones from hitting an object in the way or even can warn new pilots from destroying an investment. Obstacle avoidance is very much in need when flying in densely complex environments like outdoor environment and any other structural environmental object which can come in the way.

Many researches on obstacle avoidance techniques are going on which we will cover briefly in upcoming chapters. Now a days drones from DJI, Walkera, Yuneec have can detect obstacles not only from front but also from side, back and bottom [6]. Which makes the drone more safe to fly in dense clustered environments.
The application of obstacle avoiding sensors are not only just to avoid obstacles but the algorithm running on those sensors can also be used to track, follow, or even recognize and object in an environment and hence the main importance to obstacle avoidance technique lies in the field of search and rescue operations. These algorithms can track vehicles, people, animals and many more. Advancement of the application also includes cinematic photography shot done by drone, which is made possible only if the drone is able to recognize the object to be tracked and hence avoid other obstacle in its way.

1.3 How does Obstacle detection Sensors Works

To understand this topic lets go through each collision avoidance sensor we have around us today.

1.3.1 Stereo-Vision sensor for Obstacle Avoidance

Stereo Vision is a technique of extracting depth or distance information from an image. These sensors using image from left and right camera view of the sensor to provide the distance information of the obstacle in front. The more we identify the corresponsing pixels in an image the more are the 3D points which can be determined from a single set of images. As stereo images can be to find the depth this information is quite enough to estimate the approaching obstacle distance and hense with suitable obstacle avoidance techniques , we can also avoid them but creating a disparity map[7].

1.3.2 Ultrasonic Sensors for Detecting Obstacles

The ultrasonic sensor sends out High frequency pulses and then how long does it take for an echo of the sound to reflect back and this time is calculated to estimate how far the obstacle is from the current robot position[6]. These senors have two openings 1 to transmit high frequency pulse and other to receive the echo. The speed of sound in air is 341 meter per second and ultrasonic sensor uses this information to determine the time it takes to listen to its own transmitted echo.

1.3.3 LIDAR For Obstacle avoidance

LIDAR stands for Light Detection and Ranging , and the only difference between a LIDAR and SONAR is that the SONAR uses sound to detect obstacle and LIDAR uses LIGHT. The same way as SONAR LIDAR uses its speed of light as a constraint to determine the distance of obstacle from the robot by transmitting a laser pulse from a source and calculate the time it takes to travel from sensor to the obstacle and back by knowing the speed of light. Effective range sensors like Velodyne LIDAR sensor used in google self diving car combines multiple detector pair of upto 64 into one sensor and each can pulse at 20KHz [6]. This will allow upto 1.3 million data points per second which is very good precision for detection.

1.3.4 Monocular vision Obstacle avoidance

Monocular vision is a technique of processing images from a single lens or camera. Its upon us how and which kind of image processing techniques to apply so as to achieve the best results for obstacle avoidance. Many researchers uses Depth estimation from monocular vision to detect and avoid obstacle. Relative size of an obstacle serves an important monocular cue for depth perception, which is indeed a part of this thesis research and we will know more about it as we go further into the upcomming chapters.

1.4 Objective of this Research

The main objective of this research is to study various obstacle avoidance techniques to be applied in UAV's. After understanding the related work in the field of obstacle avoidance we propose to introduce a new method for obstacle avoidance in UAVs and discuss various techniques which will be used to implement the solution and to understand more about the field of study. Furthermore, the objective of this thesis also includes the evaluation of the proposed method.

1.5 Thesis structure

This thesis is divided in total of 9 chapters, which are intended to describe each and every step involved in the development of the research.
The first chapter deals with the introduction to this research where it is explained in brief what is obstacle avoidance, why it is being used, and the various applications of obstacle avoidance using LIDAR, Stereo-vision, ultrasonic sensor and monocular vision. This chapter also deals with a section which describe the objective of this research.
The second chapters describes the state of art technology used in the field of obstacle avoidance in UAV's. This chapter is intended to give in brief explanation of 3 very important research papers among many others. The explanation is carried out with what is the concept behind approach and the limitations, Advantages of every state of art mentioned.
The third chapter is what is called fundamentals of this research which basically deals with what are the important technologies and methods used behind this research so as to gain fundamental concepts used in this thesis.
The forth chapter is the proposed method where it is explained the method proposed in this research to achieve the respective results
The fifth chapter deals with Implementation of this research which involves the software and its configuration implementation also with the architecture model of the proposed method.

The sixth chapter talks about the results achieved after implementing the concept and proposed method to this research. This also includes the results achieved with raw phone camera and the data extracted from the distance to object and its size. The seventh chapter is conclusion which concludes the thesis with the achieved results and the work involved. The environment in which the research have been tested and validated.

The eighth chapter discusses about the future work that can be done over this research to achieve better results

The final chapter is the references in which the key papers and the source of information to this research are mentioned with the author name and publication date.

Chapter 2

State of the art

Obstacle avoidance have been widely used over the years and a subject of interest for decades for many researchers and several algorithms have been proposed to tackle such problem. This thesis is mainly focused on vision based obstacle dectection and avoidance as the proposed method is also in the direction of vision based obstacle avoidance. The intension of the following state of art technology is to find a close match between the proposed method and the other developed methods.

Obstacle avoidance can also be achieved using LIDAR data to generate a two dimensional Cartesian map [2]. The approach is used to generate the map of the environment prior and then the path will be planned based on the map to allow the drone or robot to move in the desired obstacle free area [2]. In these types of techniques it is assumed to have less moving obstacles as this approach is not build to avoid moving obstacle in the path of the robot. The disadvantage of these approaches are that these approached need prior information of the environment it will be working in.

Obstacle avoidance in drones can also be made working using SONAR in which SONAR sensors are used to know if the obstacle is approaching the drone or vice verse. These sensors on the other hand are having very less working range approximate 2m. Vision related obstacle avoidance gives us long range of view and also most accurate information about the environment.

Obstacle avoidance is also achieved using feature estimation algorithm for terrain mapping [5]. In this algorithm feature measurements from camera images are used to update the extended kalman filter [5]. In this algorithm the frames were captures at 28 fps [5]. Although this method uses GPS and IMU (Inertial measurement units) for its working and hence this is not suitable for indoor environment where the GPS might be inactive.

This chapter further will deal with few very well known techniques of obstacle avoidance techniques this thesis will be using reference from and also we will deeply analyze the algorithm used, the environment in which the algorithm was tested and how was the effectiveness of the solution proposed in these few papers. Further more we will also understand the various techniques and software used to achieve the solution in the field of obstacle avoidance . Below are the techniques which will be referred to in my approach of obstacle avoidance. These listed paper or researches are those which matches more closely my proposed method for obstacle avoidance in drones.

2.1 Obstacle Avoidance using Monocular Vision

The application of vision for obstacle avoidance paper basically deals with the size detector that will be particularly useful to detect and avoid frontal obstacles. One of the methods is based in feature descriptor, which is used to detect relative size change in features and is executed in real time to avoid obstacles in front. With addition to this the algorithm also implements guidance that will permit flights through clustered environment like forests. Since other methods like depth are not applicable in environment like forest and also methods like optical flow deals with more textured obstacles, and fails most of the time in natural environment with tree etc. on the other hand various other implementation also would require known object shape and size o detect and avoid them.

The approach is based on bio-mimetic concept, and a size expansion cue can alert the vehicle about the approaching obstacles, which also tells us looming can stimulate the sense of object approach in a very realistic manner. The algorithm implements a relative size detector and avoidance algorithm which returns obstacle size as a function of time to detect and avoid them from a distance of 2 to 3m. The apparent size becomes 1.5 times larger since the frame rate is known, which can eventually provide us time to collision.

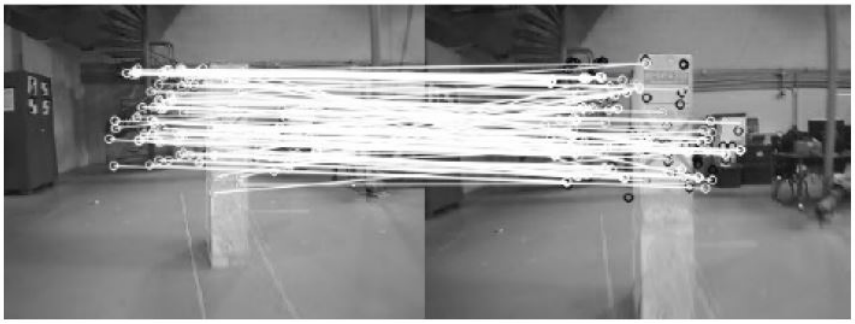


FIGURE 2.1: SURF feature points matching

The approach uses SURF [1] to to find the key points and consecutive image and compare the size around key point to recognize the object approaching camera. After calculating SURF key points , all those key points are discarded whose size did not get bigger while approaching towards it. Template matching [1] is then

used to match those key points in all the frames and if the object is really approaching it will then warn the vehicle about it and can be used to avoid those obstacles. Scale ratio is calculated and compared with previous and current frame and if the current scale is same between those, the key point is discarded. We can see the result of this algorithm in the figures below.

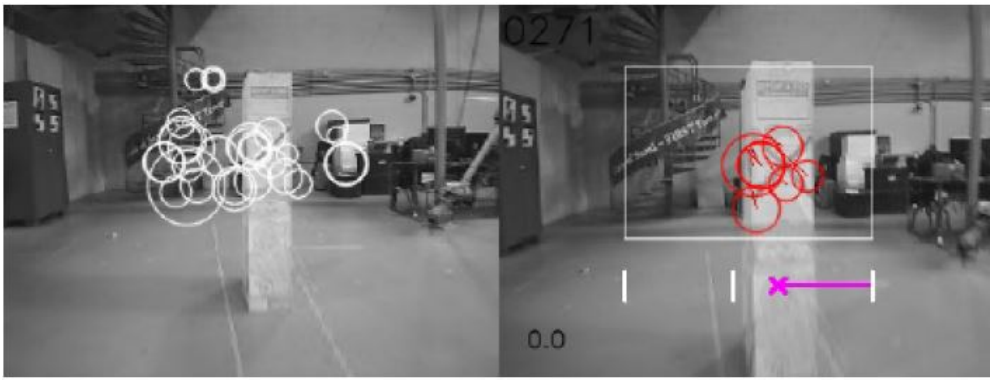


FIGURE 2.2: Previous and Current frame

2.1.1 Limitations

The limitations of this project are that first of all using SURF feature is only for textured objects, object need to be textured to get the SURF features detection from it and also this algorithm uses too much of computational power. Also this work is done using a very low powered camera and using which the over all computation time was not up to the mark, the research now aims to use a high quality camera which will eventually have more burden on the algorithm point of view and the concept might get too much complex to understand.

2.1.2 Advantages

The good point regarding this research is that this can be very effective in real time environment and SURF can be calculated even with few detection from the camera. The overall result of the research was good with 20 out of 23 trials success.

The approach using SURF is quite fascinating because of its usage with monocular vision approach which provides low payload on the vehicle.

2.2 3D Information using MOPS and SIFT

The application of MOPS and SIFT was to prove that 3 dimensional information can be constructed to evaluate the obstacle detection and avoidance. This algorithm basically deals with two main concepts namely MOPS (Multi-Scale-Oriented-Patches) which is nothing but minimalist design for local invariant features. They consist of a simple bias-gain normalized patch, sampled at a coarse scale relative to the interest point detection and the other concept is SIFT feature detection using which the internal outline of the obstacle is created. Both these information is then combined to get the 3 dimensional information of an object.

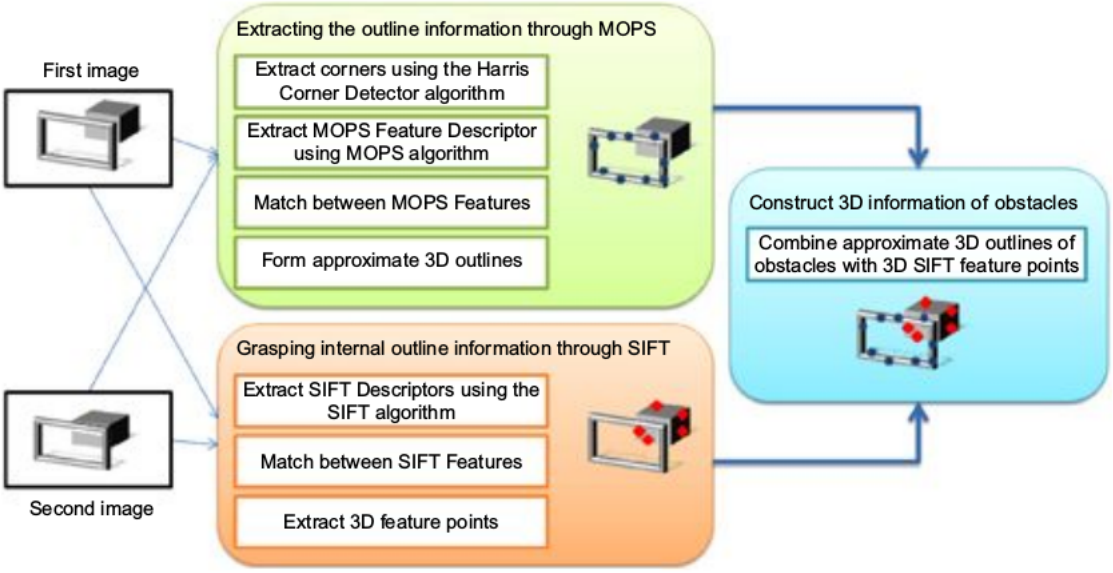


FIGURE 2.3: Algorithm to construct three-dimensional information of obstacles using MOPS and SIFT [2]

The images were captured using streaming image data and the experimental results were carried out using MATLAB. To calculate MOPS for outline information the time required was 0.577 seconds whereas to obtain SIFT feature points for internal outline requires 0.997 sec. As we can see that extracting features from streaming

images requires a lot of computation power and which will eventually affect the reaction time from the vehicle.

2.3 Obstacle Avoidance Using Disparity

Another approach is based on the stereoscopic vision using two cameras and evaluate a depth map. The algorithm is basically divided in two different modules or parts. In the first part, stereo cameras are used to capture images and process those images and find the disparity map, from which the depth can be analyzed which can eventually warn the vehicle about the obstacle knowing the depth or distance from camera to the object. The second part deals with algorithm which uses this information and implements the avoidance method to find the safe path which is obstacle free.

FIGURE 2.4: Color image(left) and Depth map(right) [3]

The approach is made to divide each frame in 3 parts namely left, center and right, and then to analyze which part of the frame gives more depth. The part which return least depth will be chosen to proceed and the command is given to move the vehicle in that particular direction which contains least depth i.e. the obstacle is comparatively far away or even no depth which states no obstacle.

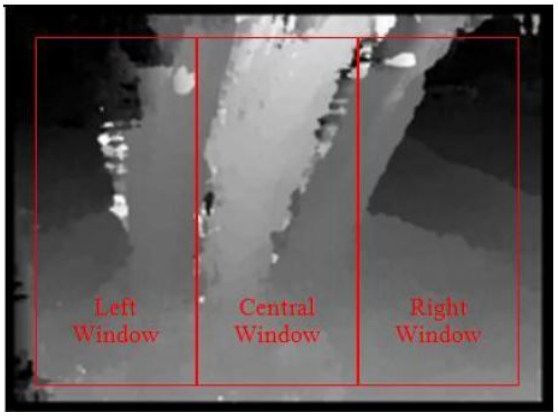


FIGURE 2.5: Disparity to check least depth [3]

2.3.1 Limitations

Disparity is a good approach which uses depth camera's or manually find depth using stereo vision, but on whole depth approach have quite a lot of limitations on its own. The depth from the camera can be too blurry when the object is directly in exposure to the light source and it eventually turns black in from of the drone and hence it can not be detected. Using advanced depth camera like INTEL REALSENSE new series can avoid this limitation too but that is quite expensive approach for a research. Hence depth can be good for limited light conditions, but not outside real time environment.

2.3.2 Advantages

The research concept is very easy to understand with very low computation power which gives huge good impact on the over all research. Although the ROI concept in the proposed method was inspired by this research paper due to its informative way of understanding the approximate obstacle location . However this algorithm is not the only one using ROI approach to avoid obstacle. Hence this approach is quite easy to understand and fits good in land robots and indoor environments.

Chapter 3

Fundamentals

After briefly studying the above state of art as related work in the field of obstacle avoidance we can analyze that the research area under obstacle avoidance is linearly growing due to the high advancement in the technology and methods. Furthermore in this research work I would like to propose my solution in the field of obstacle avoidance in UAV's.

We have seen various techniques which talks about monocular vision and most of them work over feature detection and matching, which will eventually lead the research orientation towards two approaching namely comparing the current image with database image and secondly process feature detection and matching on each frame taking previous image as reference image. Although, analyzing each of these one by one we can come to a conclusion that, In first approach of keeping reference images as database image, the method need a lot of database of images to avoid different kind of obstacle in the environment. For example a tree in the database will probably not match in the tree of real world scenario which will intern make the system more unrealistic of having infinite number of database images to make the system realistic. In the second method where the feature detection and matching is done over previous and current frame, the method requires a lot of computation power and advanced CPU to process that information so as to have a quick reaction from the vehicle. Calculating features on each and every

frame requires a lot of computation.

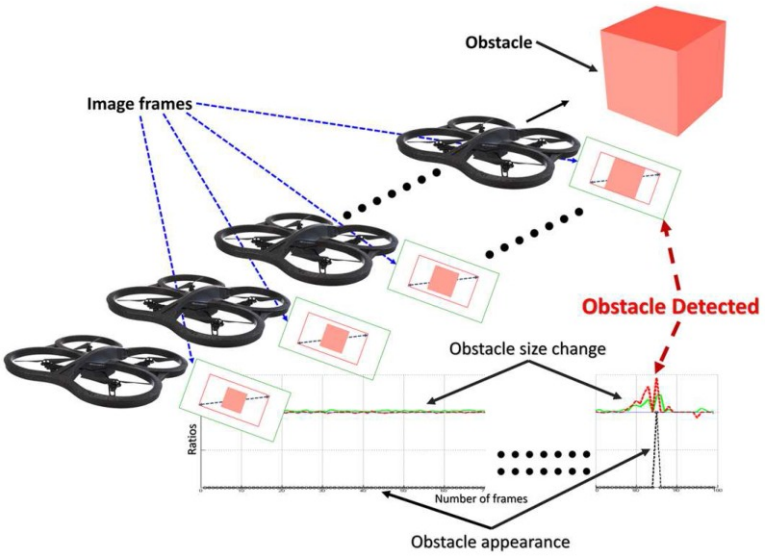


FIGURE 3.1: Concept of approach

3.1 What are contours ?

Contours are simply curve joining all the continuous points having same colour and intensity and these contours are used also in the field of object detection and recognition. Contours are good and more accurate if applied after image thresholding (we will study about it in next section) or canny edge detection.

3.2 What is Image Thresholding ?

Image thresholding is nothing but partitioning an image into foreground and background. Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images [8]. Some algorithm of image thresholding are as follows :

1. Adaptive Thresholding is used when different threshold is used for different regions in a image.

2. Image Segmentation divides the image into its constituent region or objects.

3. Segmentation algorithms are based on one of two basic properties of intensity values discontinuity and similarity.

4. Histogram are constructed by splitting the range of the data into equal-sized bins (called classes). Then for each bin, the numbers of points from the data set that fall into each bin are counted.

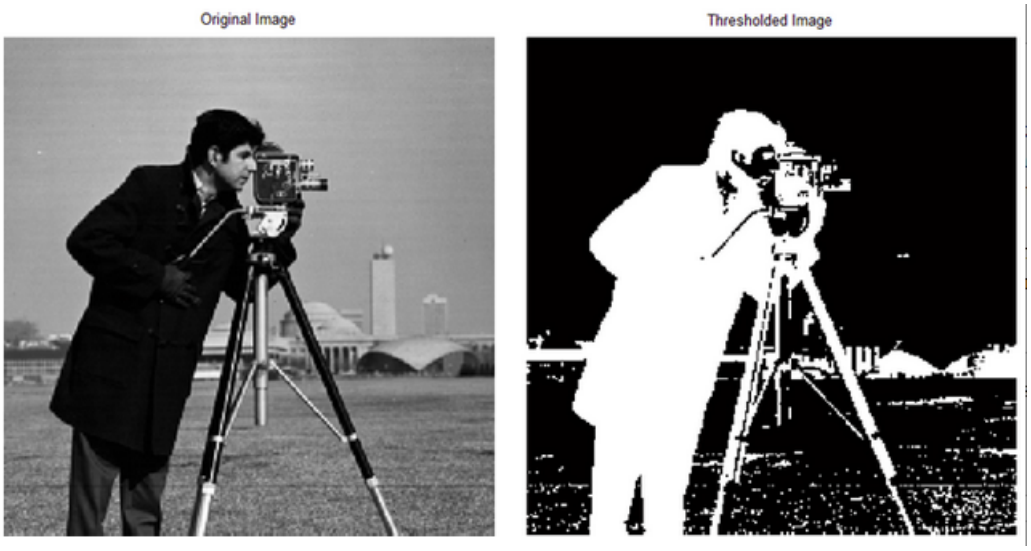


FIGURE 3.2: simple thresholding based segmentation [10]

3.3 Background Subtraction in image processing

Background subtraction is a technique in image processing where an image is classified into background and foreground. This is done to achieve different application which require segmentation of concerned objects from the scene [9]. Model of the background is created first using background subtraction which then segments out everything that does not belong to background based on spatial and temporal setting and once foreground objects are retrieved from the foreground mask, it is easy to extract as well as localize the objects in the scene.
By focusing only on parts of a scene with a significant degree of movement, we can eliminate a large number of unnecessary points to be tracked in later stages [11].

In most cases filtering is done to remove excess information from the image after the background subtract and then the features are detected on relevant regions so as to get most accurate information from features of the image.
The background is usually modelled using an Adaptive Gaussian Mixture Model where each pixel in the scene will be modelled by K Gaussian distributions according to Grimson and Stauffer [37] and [38].

3.4 Clustering

After the feature detection being applied to the image many times there are many isolated and overlapping features which does not give proper information about the object and hence cluttering helps us to remove those unwanted features from a given image. This steps helps eliminating those isolated features and reduces number of points too close to each other.

After analyzing these two main related work, and understanding basic concepts of image processing this thesis deals with a completely different approach to detect obstacle and uses the information from the related work to process the avoidance algorithm. This paper have got motivation from human being and animal's understanding of detecting the approaching obstacle and avoid them. We human beings get to know about object getting closer to us by increasing size of object as we move towards it, the important point to be noted here is we logically not at all calculating any distance to the object but assuming that because the object is getting bigger by certain limit, it might be certain meters away. The whole research in this paper will try and understand this behaviour of animals and to predict or assume the distance to the object by not actually find the distance but to calculate the size change in each frame. The paper deals with the approach to have an ROI divided in 3 parts, and if the size of obstacle is crossing a certain limit when compared to the ROI size, the vehicle will be warned and the region of ROI containing no or less size of obstacle (contour size in threshold image) will

be chosen as the obstacle free area. This method required the concept and understanding of few major concepts namely contours, time to collision, thresholding, background subtraction and concept of re gaining the original path of the vehicle after avoiding the obstacle.

3.5 Robot Operating System

Robot operating system (ROS) is an open source collaborative product developed by Open Source Robotic Foundation (OSRF), and according to OSRF "ROS was build from the ground up to encourage collaborative robotics software development" and also "its a collection of tools, libraries that's basically aims to simplify a task of working out with complex and robust behaviour across all robotic platforms" [12]. The architecture of ROS allows us to build more complex and advanced robotic systems to be performed on multiple platforms combined with the network. ROS network consists of a master and one or more ROS nodes and topics. The advantage of using ROS is that every other node in ROS runs totally independent of any other node inside the same network, which means that START/STOP of one node will not affect the working of other node. The ROS systems uses publisher and subscriber architecture in which if the node needs to provide information to the network, it does it with the help of messages on the pre-defined topic for that particular information and when that information is to be retrieved from the network, the subscriber job is to subscribe to the topic from which the information needs to be retrieved and pull the messages that are being published on that topic. In order to accomplish and establish this behaviour ROS master is used. ROS master is started before all the nodes in the network and when the nodes are started the ROS node gets registered with the master node running on the same or a different machine. As the result of this each node provides to the master all the subscription and publication information. The master also uses this information to keep track of all the nodes, topics, messages and services so that when a new node is registered with the network and which requires the interface with the existing part of the network, master can easily update the nodes with a

new connection without any side affects.

A simple ROS network consisting of a ROS network is described in Figure 4.1. As we can see like the architecture described above this example netwrok also consist of a ROS master which can also be called as ROS CORE in other words, in the figure we have a sensor i.e camera which is giving its data to a node called "Camera Node" and the work here of a camera node is to may be convert the image format or to convert the ROS image to Mat image (for opencv usage). The camera node then publish the desired form of camera data to a topic called "imageDataMessage" as shown in figure. The node-2 which is image processing node is to process the image from the sensor and perform various image processing techniques to achieve the desired results. This node-2 is subscribing to the image topic being published by the publisher i.e node-1, camera node. All these nodes and topic are registered in the ROS master node. There is another node-3 called image display node and its job is to display the the sensor information by subscribing to the same camera topic, this node can be a visualization environment too which we will discuss in later section of this chapter. Although various other aspects of ROS network is not show in this example but this covers at least the major concepts of ROS architecture. This whole architecture combine to form a single ROS network.

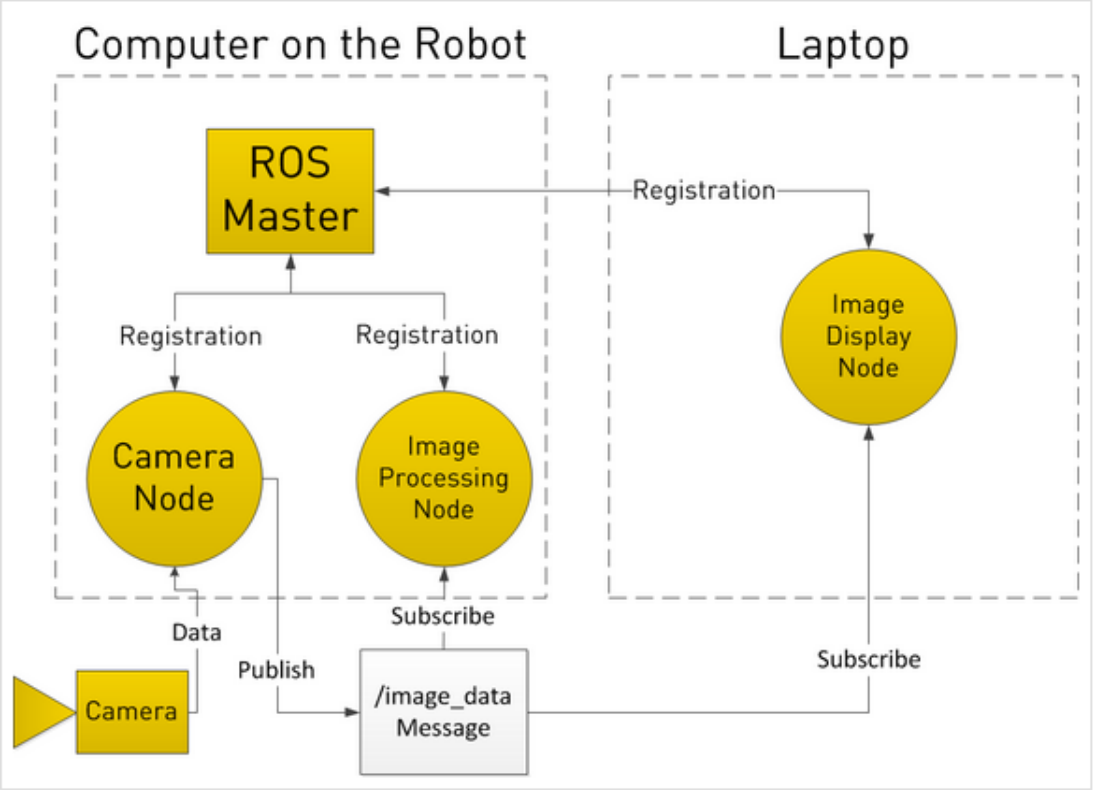


FIGURE 3.3: ROS example network

In this implementation Ardrone is being used to test and validate the results of the algorithm and also the model of the drone is also added with a mono camera which is used to provide us a camera feed on a given topic. The camera topic was called /ardrone/camera/image_raw and this is set in gazebo which we will discuss in detail in our later section of this chapter. The ROS node "contour" was written as a subscriber to this image topic and the work of this node is basically first to subscribe to the camera topic, convert the ROS image to an opencv compatible image using "cv_bridge" and the ROS image is converted into MAT image using this bridge.

The other important node will be an offboard node which will subscribe to the state of the drone through ardrone config and save its current state. The node also publish the local state to the topic called /ardrone/setpoint_position/local at a rate of 10 Hz. Through this node we can arm, takeoff and give the position values to the drone. /ardrone/state is a topic that holds the current state of the vehicle, which includes arming state, local position and other important

data. Through the /ardrone/state topic from the offboard node then publish command to arm the drone. For take off offboard node also uses /ardrone/state and /ardrone/set_position/local to give the take off altitude and position values to the ARDrone quadcopter.

Chapter 4

Proposed method

After studying various researches in the field of obstacle avoidance in UAV's, this research is intended to focus on the method proposed and the results obtained from it. The method is intended to focus on the size of obstacle in front of UAV and estimate the approximate distance of the object from UAV. To achieve the same we have developed the ROS node responsible for image processing so to get the image from the RGB camera installed on the drone and understand the approaching obstacle based on the size of the contours. Hence this will also eliminates the need of more heavy sensors like lasers, LIDAR, radars for performing obstacle avoidance.

The flow chart of the current implementation of the concept is shown below :

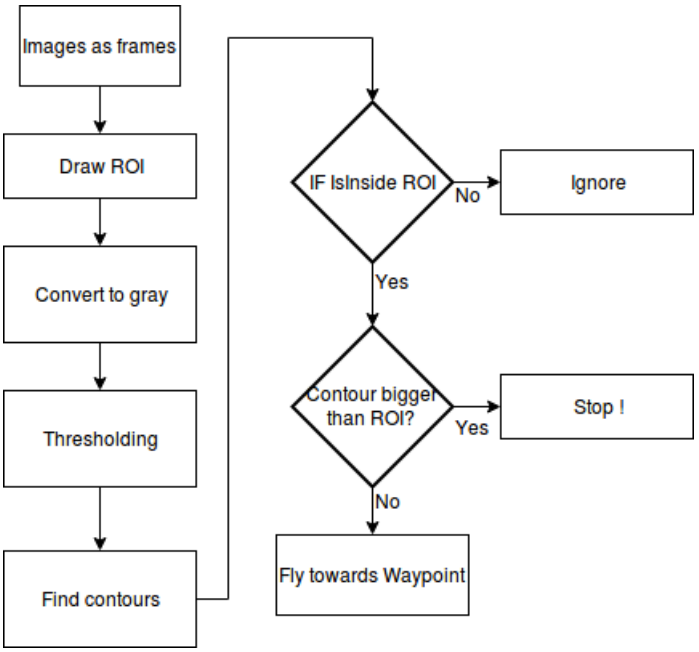


FIGURE 4.1: Flow chart of the Detection implementation

The images are captures as frames from the drone's camera and then it undergoes the convertion from RGB to mat image so as to make it compatible for the processing over it. Ones the we have our frames in MAT format , we do canny edge detection over the frames. Canny detection undergoes the following 5 steps during the complete process :

1. Gaussian filter to smoothen the image frames.
2. Intensity gradient.
3. Non-maximum suppression.
4. Double threshold to determine fine edges

4.0.1 Gaussian filter to smoothen the image frames

We know that the edges information is easily affected by noise and hence it is required to clean or filter out noise from the image. To do so the Gaussian filter is applied. The equation of the Gaussian kernel of size (2K+1)x(2K+1) is :

$$H_{ij} = \frac{1}{2\pi\rho^2} exp(-\frac{(i-(k+1)^2)+(j-(k+1)^2)}{2\rho^2}) \tag{4.1}$$

Hence for 5X5 Gaussian filter , to create adjacent image with ρ=1.4 :

$$B = \begin{pmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{pmatrix} * A \tag{4.2}$$

The selection of the size of the kernel will affect the performance of the canny edge detector. Larger the size of Gaussian kernel, more lower the sensitivity to noise. We have also chosen 5X5 Gaussian kernel because its standard for most of the application to get a fine smooth result.

4.0.2 Intensity gradient

The edge may point to many direction in an image for to correct that intensity gradient is used to detect horizontal, vertical and diagonal edges in the blurred image. Edge detection operation returns a value for the first derivative in the horizontal direction (Gx) and the vertical direction (Gy). From this the edge gradient and direction can be determined :

$$G = \sqrt{G_x^2 + G_y^2} \tag{4.3}$$

$$\theta = a\tan 2(G_y, G_x) \tag{4.4}$$

Where G can be computed with hypot function atan2 is the arc tangent function with two arguments.

4.1 Non-maximum suppression

This is also called Edge thinning technique, after the gradient detection the edge intensity is still blurred there should only be one accurate response to the edge. Thus non-maximum suppression can help to suppress all the gradient values (by setting them to 0) except the local maxima, which also indicates the location with the sharpest change of intensity.
If the Gradient value is 0 degrees , the point is considered to be on edge if the gradient magnitude is magnitudes at pixels in the east and west directions.

4.2 Double threshold to determine fine edges

After Non maximum suppression the edges now provide real good value and to still ignore bad edges due to noise and color variation and keep the edges with high gradient value while ignoring low gradient edges, this is done by selecting high and ow threshold value which in short is also called Image Threholding. Hence the edge gradient magnitude lesser than high threshold value is considered week and the magnitude larger than lower threshold value is marked as week edges and which eventually be suppressed to provide more better edges after the process.

After the image is thresholded and applied canny detection , we find the contours of each objects in front. Then the centroid of the each contour is found so as to get the x and y position of each of the contour. This information is taken using cv2.moments() where centroid can be defined as :

$$centroid\_x = M\_10/M\_00 \tag{4.5}$$

$$centroid\_y = M_0 1/M\_00 \tag{4.6}$$

Chapter 5

Implementation

With all this information to be the basics for understanding the implementation, software suites were selected that will be the major role in achieving the thesis research goals. The purpose of this suite will be to provide the integration of camera with the drone, which will eventually solve the main requirement of the research to be accomplished, use of these types of software helps us to also modify the environment to test for the research.

The Robot Operating System was chosen as the underlying command and control network for the entire system because of its versatility, inter interoperability, and modularity. The modularity of ROS helps integrate multiple programming languages and tools to be be used on a development system, which inturns supports vast development and continuous modification of the research.

Opencv libraries were chosen so as to integrate the software coding part of the project on top of ROS. This can be easily interfaced with ROS as we can include Opencv in our CmakeList.txt file of the ROS package. With the help of Opencv we are also viewing continuous image feeds from the ROS camera which is integrated in our simulated drone. In order to safely implement and test this research a simulation environment was needed which can provide us a good working simulated environment including drone with a mono camera and of course possibility to add and remove object from the scene.

Due to this type of requirement GAZEBO was chosen for simulation because of

its easy integration with ROS and also because of its robust feature set. This also includes built in ROS visualization tool RVIZ, which is helpful in viewing the camera topic more easily and also this provides ease of use and build in support with ROS and gazebo.
The whole tests and ROS network was performed on Ubuntu 16.04 LTS (Long term support) operating system. As the ROS architecture is only designed to be used on Linux, specifically Ubuntu Linux, this Opearing system was chosen. Also the machine on which the whole system i.e ROS, GAZEBO and the main algorithm development was performed was dual booted machine with Intel Core i5-7200U CPU, 2.50GHz x 4 processors and 64 bit system support.

5.1 Gazebo simulation environment

The requirements for the simulation environment in this research included the ability to integrate with the ROS network, have support for the sensors like camera and provide the ability to add or remove objects while the simulation was in progress in order to simulate dynamic obstacles and test the effectiveness of the system. According to the overview of Gazebo, Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments, and offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs [13]. This simulator includes a good variety of quadcopter model which was the basic requirement for this research including the camera plugin, since its developed along side ROS, this provide ROS package called "gazebo_ros_pkgs" to facilitate the communication between our ROS network and the gazebo environment [14].

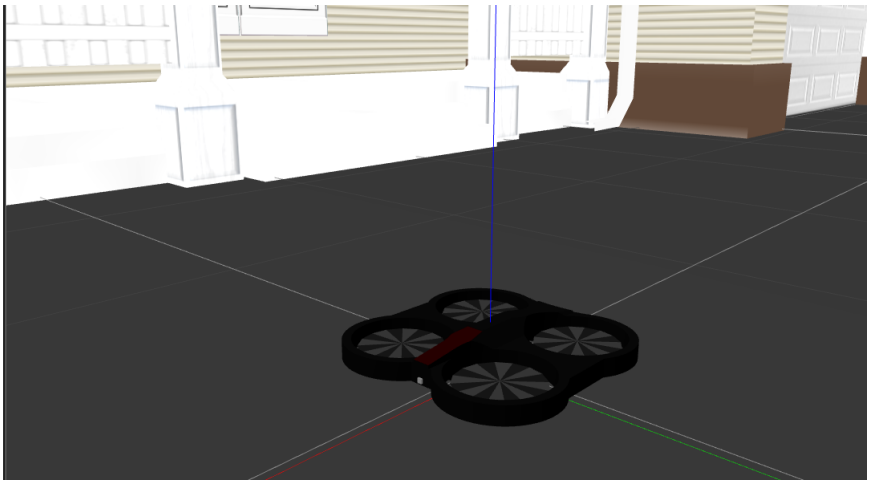


FIGURE 5.1: Gazebo model for AR DRONE with camera link

The Ardrone drone is used with the package tum simulator which contain all the necessary model for the drone and the environment itself. The tum simulator package was easily integrable with MoveIt configuration too and hence no time was invested in making the model for the UAV and its environment.

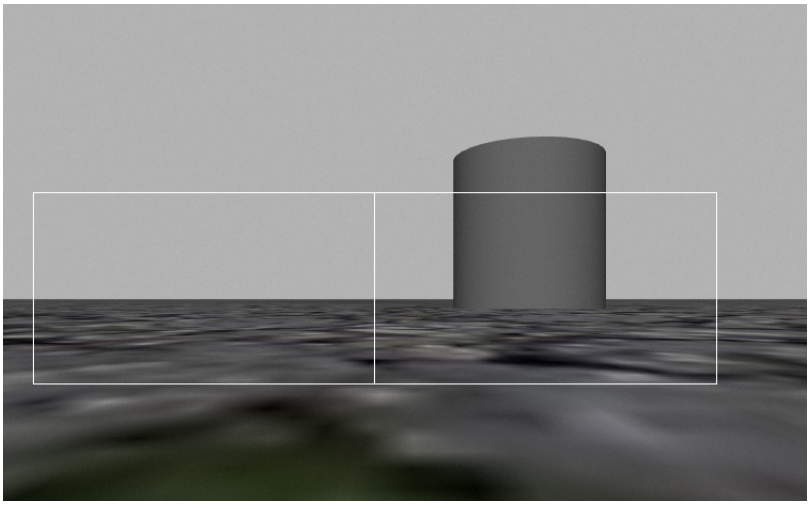


FIGURE 5.2: Image feed with ROI from Ardrone camera

Gazebo plugins give your URDF models greater functionality and can tie in ROS messages and service calls for sensor output. Plugins can be added to any of the main elements of a URDF - a robot,link, or joint depending on what the scope and purpose of the plugin is. To accomplish adding a plugin to a particular element in your URDF, you must wrap your plugin tag within a gazebo element. Going

little deeper in the conversion process, the URDF is converted to a SDF before being parsed by Gazebo. Any further elements inside the gazebo tags which are not in the element table on URDFs are directly inserted into the model tag of the generated SDF. Below figure shows an example of the camera link inside an SDF file.

```
<link name='cam_link'>
  <pose frame=''>-0.118317 0.002195 0.004205 -0 -0 -3.12703</pose>
  <inertial>
    <mass>0.1</mass>
    <inertia>
      <ixx>1</ixx>
      <ixy>0</ixy>
      <ixz>0</ixz>
      <iyy>1</iyy>
      <iyz>0</iyz>
      <izz>1</izz>
    </inertia>
    <pose frame=''>0 0 0 0 -0 0</pose>
  </inertial>
  <self_collide>0</self_collide>
  <kinematic>0</kinematic>
  <sensor type="camera" name="camera1">
  <update_rate>30.0</update_rate>
  <camera name="head">
    <horizontal_fov>1.3962634</horizontal_fov>
    <image>
      <width>800</width>
      <height>800</height>
      <format>R8G8B8</format>
    </image>
    <clip>
      <near>0.02</near>
      <far>300</far>
    </clip>
    <noise>
      <type>gaussian</type>
      <mean>0.0</mean>
      <stddev>0.007</stddev>
    </noise>
  </camera>
```

FIGURE 5.3: .SDF file for camera link

The SDF file was modified to get the camera link in the ARdrone quad copter, the camera plugin name used was "camera_controller" for the camera and for using this "libgazebo_ros_camera.so" with camera image width set as 800 and height as 800 (pixels) too. Although to add a plugin to the model we need to use another tag in xml which is "plugin". The camera image topic on which image will be published must be described here in this plugin tag. Here we are just using two topic namely "camera_info" to get the available camera info parameters and the other is "image_raw" which contains the main image feed from camera, this can be seen in figure 4.7.

```
<plugin name="camera_controller" filename="libgazebo_ros_camera.so">
  <alwaysOn>true</alwaysOn>
  <updateRate>0.0</updateRate>
  <cameraName>iris/camera</cameraName>
  <imageTopicName>image_raw</imageTopicName>
  <cameraInfoTopicName>camera_info</cameraInfoTopicName>
  <frameName>camera_link</frameName>
  <hackBaseline>0.07</hackBaseline>
  <distortionK1>0.0</distortionK1>
  <distortionK2>0.0</distortionK2>
  <distortionK3>0.0</distortionK3>
  <distortionT1>0.0</distortionT1>
  <distortionT2>0.0</distortionT2>
</plugin>
```

FIGURE 5.4: camera plugin and image topic description

With this we now have set all the requirements for our simulation environment to be used for the research. To summarize, with the help of Gazebo, we achieved to get our vehicle and the camera with the running camera topic. With all this the research approach is now to go much deep into the system architecture and algorithmic approach of the obstacle detection technique, and from now we will discuss about our algorithm used and the implementation of the system architecture in later section of this chapter.

As shown in figure 4.9, the we use the ROI over the image to know if the contour is exceeding to a certain limit and hence we made an ROI diving in 2 parts of each frame called left and right as shown in figure 4.10. As the size of the obstacle is equal to or more than the size of ROI, the drone gets a warning to stop as it is approach the obstacle in front. Further this can be added to Moveit path planning algorithm and it will take care of it on its own. Below is the result of the contour implementation from AR drone camera.

FIGURE 5.5: Left : image from ARDrone camera, Right : Contours of obstcles.

5.2 Ardrone Test with Moveit - Configuration

MoveIt provides great user friendly graphical interface for integrating any kind of robot in it. The MoveIt setup assitance takes care of generating all the files based on information provides during the setup. In order to launch setup assistance the following command needs to be executed :

roslaunch moveit_setup_assistance setup_assistance.launch

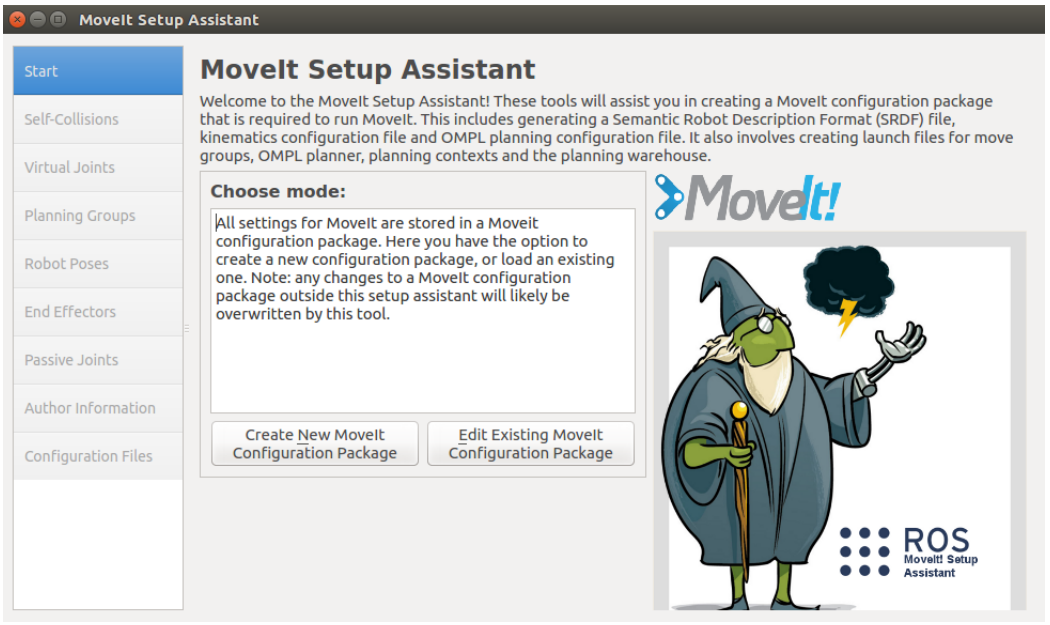


FIGURE 5.6: Front screen of MoveIt setup assistance

The second tab in the screen as shown in figure 4.6 is after loading the URDF of the ardrone to get the configuration of our drone to moveit.

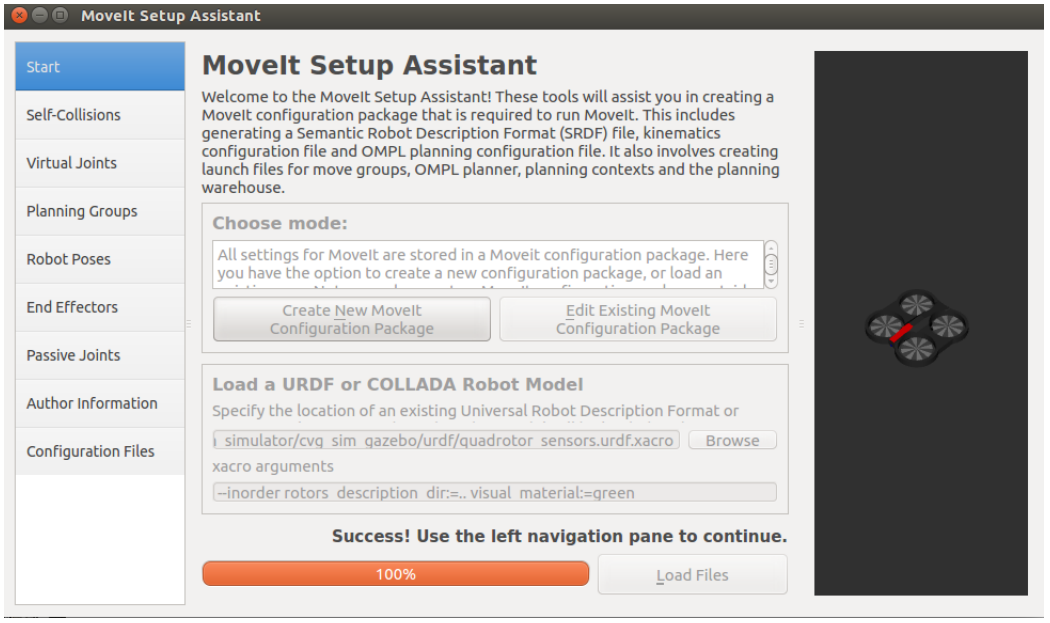


FIGURE 5.7: After loading the ardrone URDF.xacro

The next window is for self-collision check , which mean when the drone is moving around, this check if the collision between the self link is possible. Hence thi

generates a self collsion matrix which tells the MoveIt not to collide with self joints.

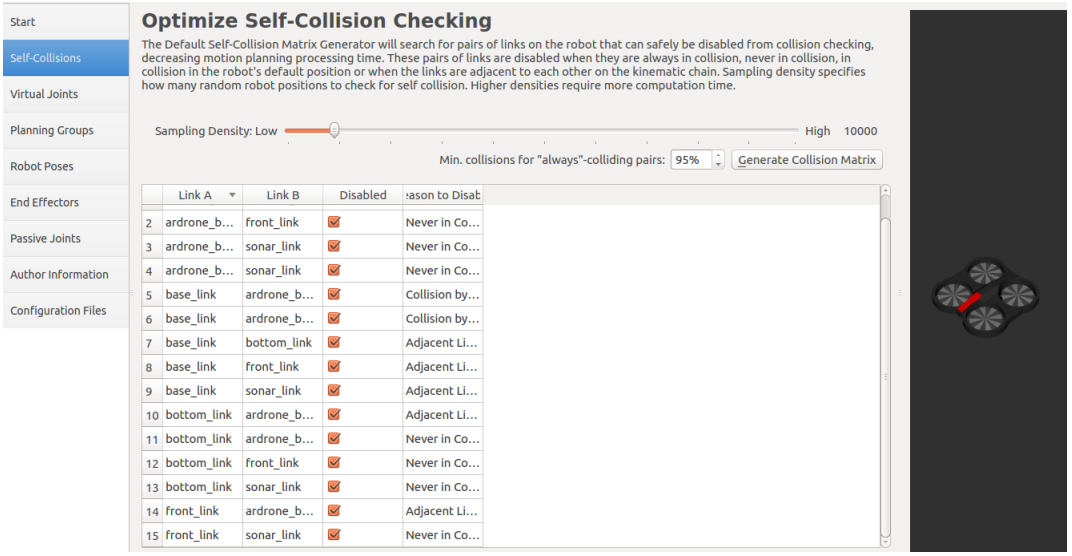


FIGURE 5.8: Setting up self collision for ARdrone

The next step include the virtual joint creation with joint type as floating , configuring child link as base link and parent frame as world. The next step is the planning group in which we need to add a group with name as ARdrone_group, also we will add only the virtual joints as other joints are just the constant transform of the virtual joint and also while adding a link to the planning group we will add base_link and save the configuration. At last we need to add the configuration files to a ROS workspace to summarize the setup.

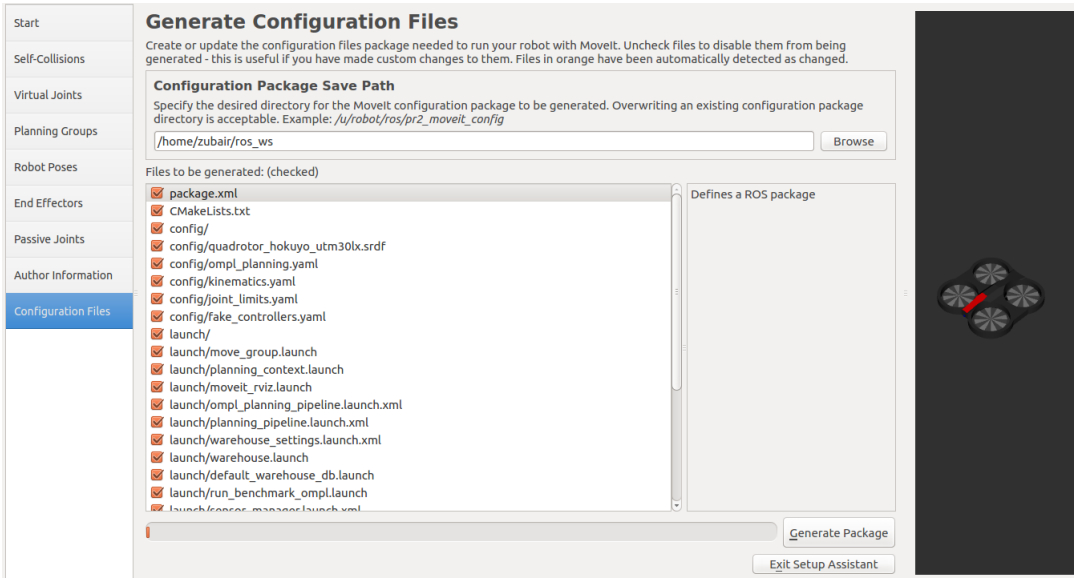


FIGURE 5.9: Finishing setup by giving location of ROS workspace for config files

5.3 Path Planning MoveIt Architecture

Obstacle avoidance is a fundamental problem for any autonomous system as it attempts to reach its destination. The goals of this thesis research require the robot to be able to detect obstacle, the main aim of this research is to mainly do the process for static objects, although this can also be done on dynamic objects with a little more added approach. With this in mind, the approach of this research is to focus on this factor of working with static obstacles, and hence the only sensor used was a mono camera. The sensor data is carried out on ROS topic as discussed in previous sections and hence the image processing algorithm comes into picture. The data from the camera is subscribed in an image processing node to get the contour position and comparing it with the ROI size. The algorithm and the architecture in details will be discussed in this chapter ahead. Also in this section we will combine and understand the big picture of the system.

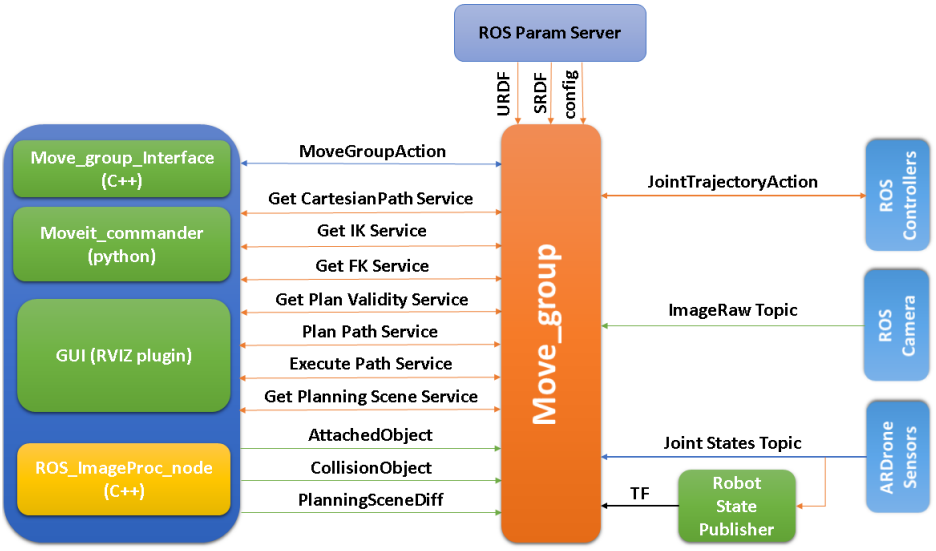


FIGURE 5.10: MoveIt Architecture [14]

Lets discuss about each block briefly so as to understand the functionality of each block in the system.

5.3.1 Move Group

The primary interface to the ROBOT to MoveIt is though class called MoveGroup. This provides easy to use and understandable functionality the we want to carry out, this includes in our case, moving the drone around, setting joint or pose goals, in creating motion Plans for the drone, adding or removing obstacles from the environment. Hence this contains all the joints, sensors and state information of the ARDrone and provides the desired functionality knowing all the listed information about the drone or ROBOT. MoveGroup hold all the possible information about the vehicle and we can access each of them from MoveGroup Class. The example to add an obstacle to the MoveIt environment is shown below from the implemented code.

```
object_in_map_pub_  = nh.advertise<moveit_msgs::CollisionObject>("collision_object", 10);
ros::Duration(2.0).sleep();

moveit_msgs::CollisionObject cylinder_object;

cylinder_object.id = "pole"; /*Id of the BOX is used to identify it*/
cylinder_object.operation = cylinder_object.ADD; /*To add the obstacle*/
cylinder_object.header.frame_id = "base_link";
cylinder_object.header.stamp = ros::Time::now();
shape_msgs::SolidPrimitive object;
object.type = object.CYLINDER;
object.dimensions.resize(2); /*Dimentions of the Obstacles*/
object.dimensions[0] = .9;
object.dimensions[1] = .2;
geometry_msgs::Pose pose;
pose.position.x = .6; /*Positions of obstacle in the scene*/
pose.position.y = -.6;
pose.position.z = .375;
pose.orientation.x = 0;
pose.orientation.y = 0;
pose.orientation.z = 0;
pose.orientation.w = 1;

cylinder_object.primitives.push_back(object);
cylinder_object.primitive_poses.push_back(pose);

cylinder_object.id = "pole";
object_in_map_pub_.publish(cylinder_object);

std::vector<moveit_msgs::CollisionObject> collision_objects;
collision_objects.push_back(cylinder_object); /*Pushing the obstacle in the vector*/
```

FIGURE 5.11: Adding obstacle to Moveit Planning Scene

This implementation successfully added the obstacle in the planning scene. MoveGroup class also talks with other implmentaions in existing on the network such as ROS controllers, ROS camera for ARDrone, Param server and other image processing node. This gather the information about the state of the drone, the planning scene, action callbacks, and just perform requested callback action when in need. The drone information was loaded in the ROS param server by using the Moveit Setup Assistance and hence it contains the URDF, SDF and other configurations regarding the ARDrone.

5.3.2 ROS Param Server

The MoveGroup class is also configured with the ROS tools and definition languages such as YAML, SDF, URDF. This defines the group with the joints that will the part of the group with their joint limits. The drone must also expose JointTrajectoryAction so as to get the output of motion planning to be executed on hardware or even in the software like in our case. In order to executes the planning for the obstacle free path /joints_states is also needed by the means of drone's state publisher, all of these are together provided by ROS control and

senosor drivers. All this is configured by Moveit Setup Assistance, this also builds the MoveIt work space for our drone. Once the param server is loaded to MoveIt, ARdrone configuration is known to Moveit which includes sensors and joints information.

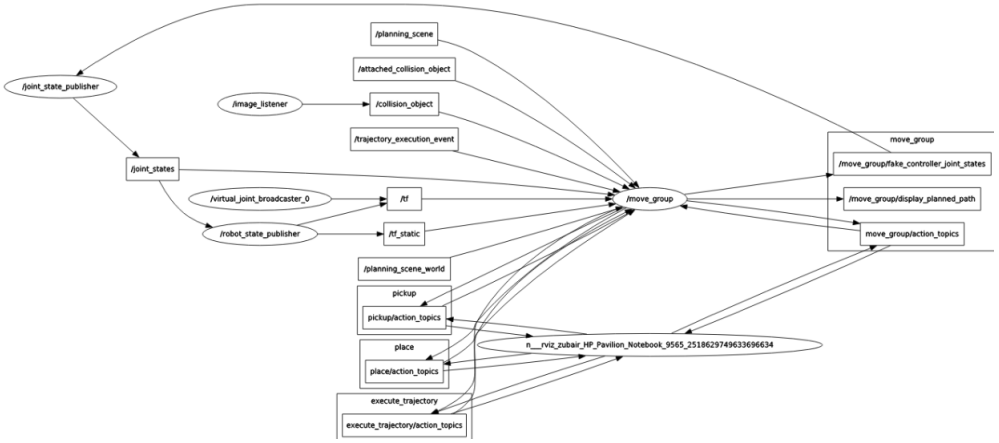


FIGURE 5.12: Rqt graph of the interaction with whole system

In the above graph we can cleary see the whole system intraction with MoveGroup and the exchange of information in the current implementation. Each block in the grapth has its own importance in providing moveit the necessary information it need to do path planning, collision avoidance and load other trajectory information.

5.4 Path Planning Algorithm

Path planning in MoveIt deals with defining an obstacle free path for our drone allowing us to reach the desired goal, in other words keeping the drone away from obstacles perceived from the camera and also taking care of joint limits violations. In our project we are using Moveit motion planning library called OMPL using ROS actions and services

While execution a motion planning request is sent to MoveGroup which will take care of obstacle collision avoidance (including self colision), and finds an optimum trajectory for all the joints in the group, so as to reached the desired goal requested.

The result of motion planning is a trajectory that moves the vehice to the target location. This also satisfies velocity and acceleration constraints at joint level. MoveGroup class have the motion planning pipeline made of planners request adapters. This is used for pre and post-processing of the motion plan request. Pre-processing is use full to check of the initial state of the vehicle is inside the joint limits, while post processing is used to convert paths into time parameterized trajectories. Default Motion planning adapters provided by Move-It are as listed below :

- FixStartStateBonds : This fixes the initial and start state to be inside the joint limits which were configured in the URDF, without this the joints are outside the joint limits , the planner would not be able to find any plan since the joint rule is already violated. This will only come into picture if the joint state is outside the limits exceeding a certain amount.
- FixWorkSpaceBonds : This defines a default work space to be plan, by default the work space is set to 2x2x2 m cube but for our case since we need a bigger work space this was set to 10x10x10 m cube as shown in figure 4.11.

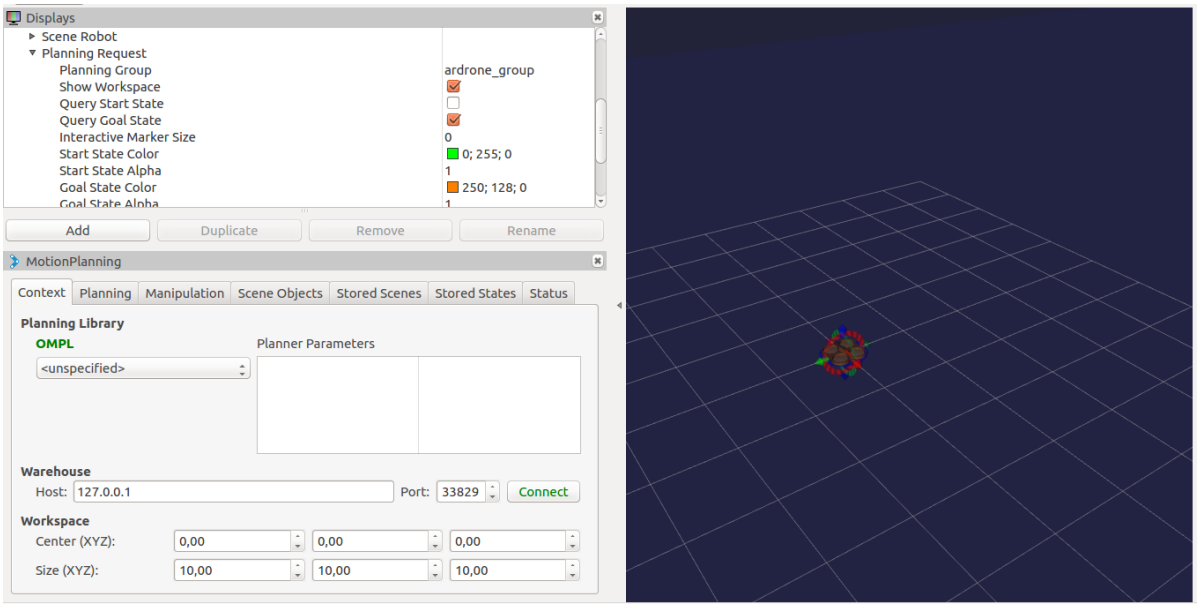


FIGURE 5.13: Model inside a Moveit Workspace

- FixStartStateCollision : This attempts to sample a collision free configuration near a given configuration in collision. It does it by distributing joint states by a very small amount. The world geometry monitor uses occupancy map to monitor and to build a 3D representation of the environment around the drone, and also augments it with the planning scene topic information, such as objects. An octomap is used to register all this information.

5.5 Proposed Architecture Model

In this section we will see and understand the architecture for the proposed method, which involves the obstacle avoidance using size as a constraint. In this method there are some few assumptions we need to make, firstly the obstacle in front of the UAV is previously known, i.e. the dimension and approximate. distance from the camera (initially).
To achieve this we have used a very interesting methodology which is called "Triangle similarity", which goes something like this, Lets say we have a object with a known width W, we will then place this object at some distance D from camera. We will then take a picture of the obstacle from that distance D and measure the apparent width in pixel P. This will then give the perceived focal length of the camera :

$$F = (PxD)/W \tag{5.1}$$

In our case, the obstacle is at a distance of almost 6 meters from the UAV which is equivalent to 236.22 inch (D), with known obstacle width of 39.37 inch (W) and the obstacle apparent width from image processing is found out to be 119 pixels (P) and if we apply all these values in the above formula we get the approximate. focal length of the camera i.e. 714.
Next, the goal of this is to find the distance of this obstacle from our UAV. So,

to achieve that, as we move our camera closer or farther away from the known obstacle, we can then apply triangle similarity to find the distance of the obstacle to the camera/UAV, hence :

$$D' = (WxF)/P \tag{5.2}$$

Now as we go close and far from this obstacle, the pixel (P) will change according to the size of the obstacle, and hence we get the effective distance between the obstacle and the UAV. The architectural view of the proposed system is shown below :

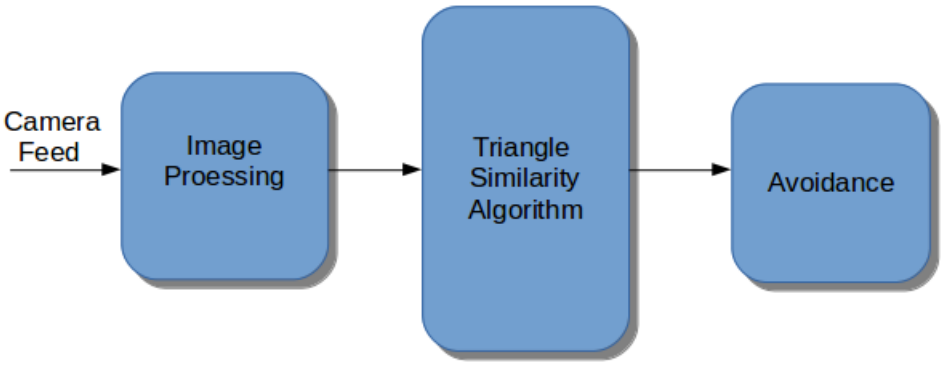


FIGURE 5.14: Proposed System architecture

The system architecture is quite simple and easy to understand. As explained in previous sections the image processing is done over sequence of images from the UAV camera and then the triangular similarity is applied to get the approximate distance from the UAV to the obstacle. The avoidance is done by stopping the UAV in front of obstacle at a distance of 3 meters and then fly up until the contour id is out of the ROI (region of interest). In the picture below we can see that the UAV have fly upward after detecting the obstacle at 3m of distance. Of course

this is just to show that the UAV is able to react according to the result fed depth.

Chapter 6

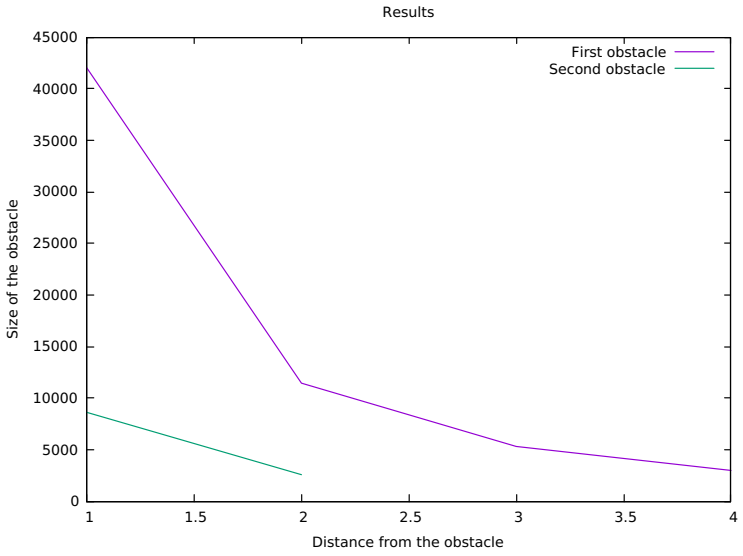
Result

As discussed in above chapters we have successfully tested the detection algorithm in few situation and were able to get the pretty good result out of it. In this research we have successfully tested and implemented following things :

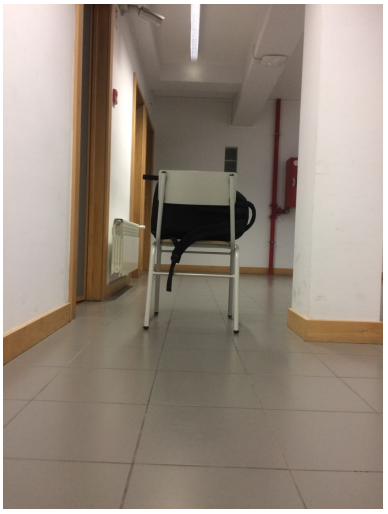
- We have implemented a new proof of concept for the obstacle detection using the image size and contours and results were tested with few conditions and also we have implemented the Obstacle avoidance with one of the Widely used sensor, LIDAR in corporation with MoveIt path planning capabilities. In this section we will study and understand the results we have got from both the implemented solutions i.e LIDAR as sensor and updating the known obstacle info to the Cost Map and on the other side using size as a constraint and getting the approximate. distance from the obstacle. Lets analyze the results produced by each method one by one.

6.1 Results while testing the proposed method (Manually)

We have first tested the changing size of the detected contour after placing few objects in front of the camera and we have tried to determine the changing size of the obstacle(s) as we move 1 meter/reading near to it. In the figure below we can see the graph with 2 obstacles in front of the drone and also analysis of change in size as the obstacle was approaching the vehicle. In the graph the x-axis represents distance from the obstacle in meters where as the y-axis represents the size of the obstacle in pixels.



(a) Figure A

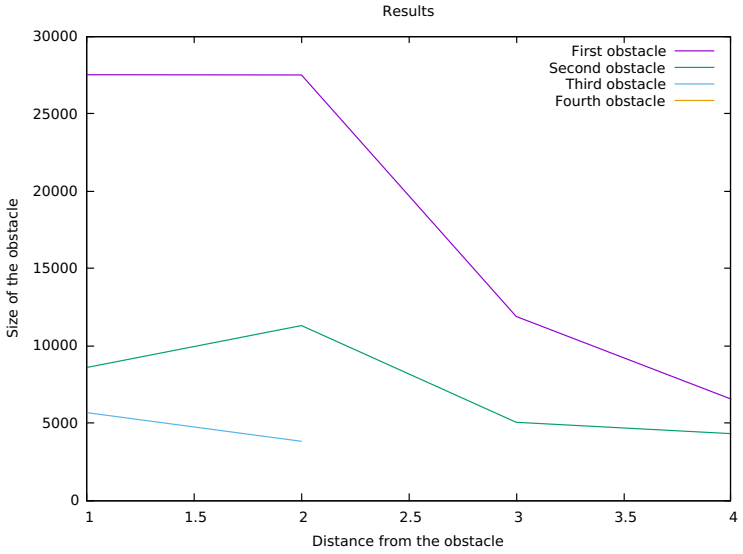


(b) Figure B

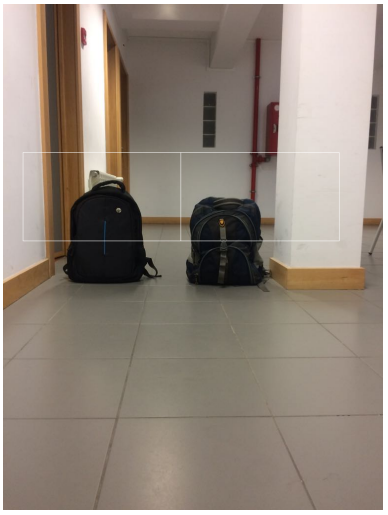
FIGURE 6.1: Obstacle detection (total 2 obstacles in front). Left : Graph of changing size; Right : Image of the scene

We can clearly see that the second obstacle was small enough that it wasn't visible from 4 meters although started to be visible when the vehicle reached 2 meters, and also the first obstacle grown its size drastically at a distance of 2m from the vehicle.

- The algorithm was also tested with 4 obstacles in front and the results seems to be quite good. As usual the x axis here represents Distance in meters and y axis represents the size of the obstacle in pixels.



(a) Figure A



(b) Figure B

FIGURE 6.2: Obstacle detection (total 4 obstacles in front). Left : Graph of changing size; Right : Image of the scene

This result is quite entertaining because as we can see in the graph obstacle 1 i.e pink color and obstacle 2 i.e. blue color are both visible at a distance of 4 meters that means they are big enough to be detected at the range of 4 meters and also

the size of first obstacle grown drastically at the distance of 3 meters for around 28000 pixels and hence in this case the vehicle will get the warning as the obstacle size have already exceeded it limit and hence it can be avoided at the distance of 3 meters. Also the last obstacle was very small to be visible only at a distance of 2 meters.

6.2 Path planning result with MoveIt

- The collision world object of the planning scene is used to configure collision checking , and this is done using a special library called "Flexible Collision Library" package. The type of collision object supported in this library are meshes, primitive shapes, planes and an octomap. For collision check a matrix called Allowed collision matrix is used to encode a Boolean value, which indicates whether the collision is possible or not between two pair of bodies , where value 1 states that the collision is not needed.

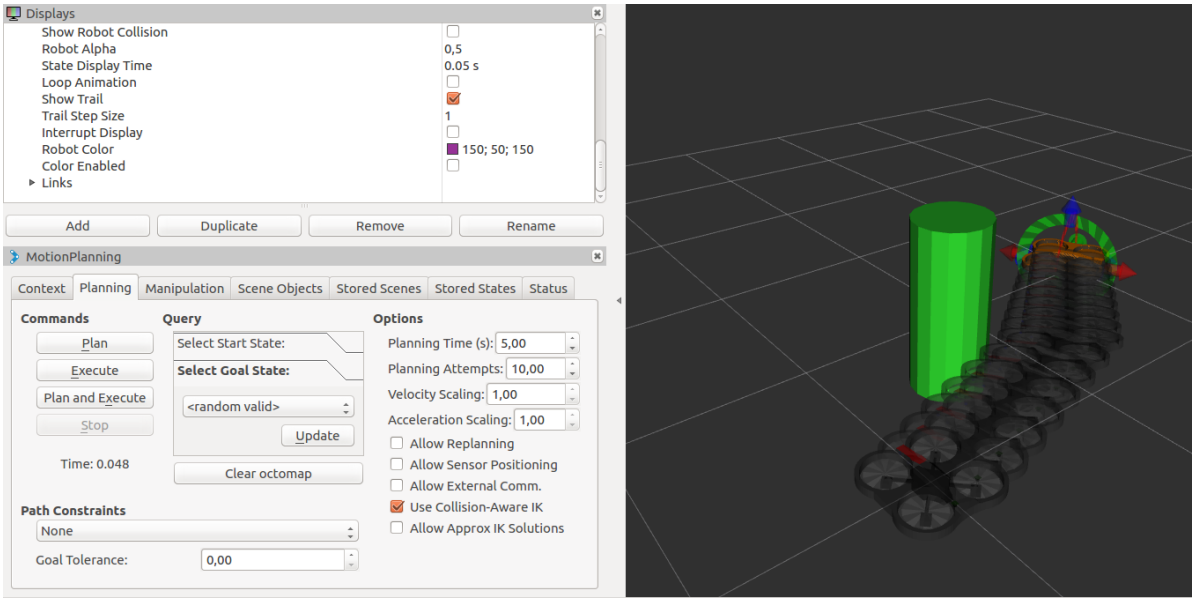
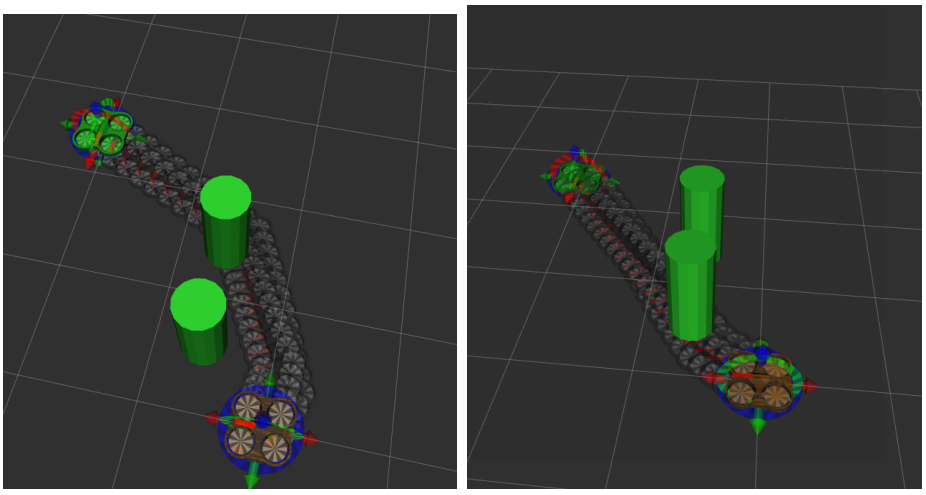


FIGURE 6.3: Path planning

In the above figure we can see that the trajectory of the drone is obstacle free, in other words words the drone is avoidance the obstacle in front of it. Hence using this we can integrate new plugin for any kind of sensor.

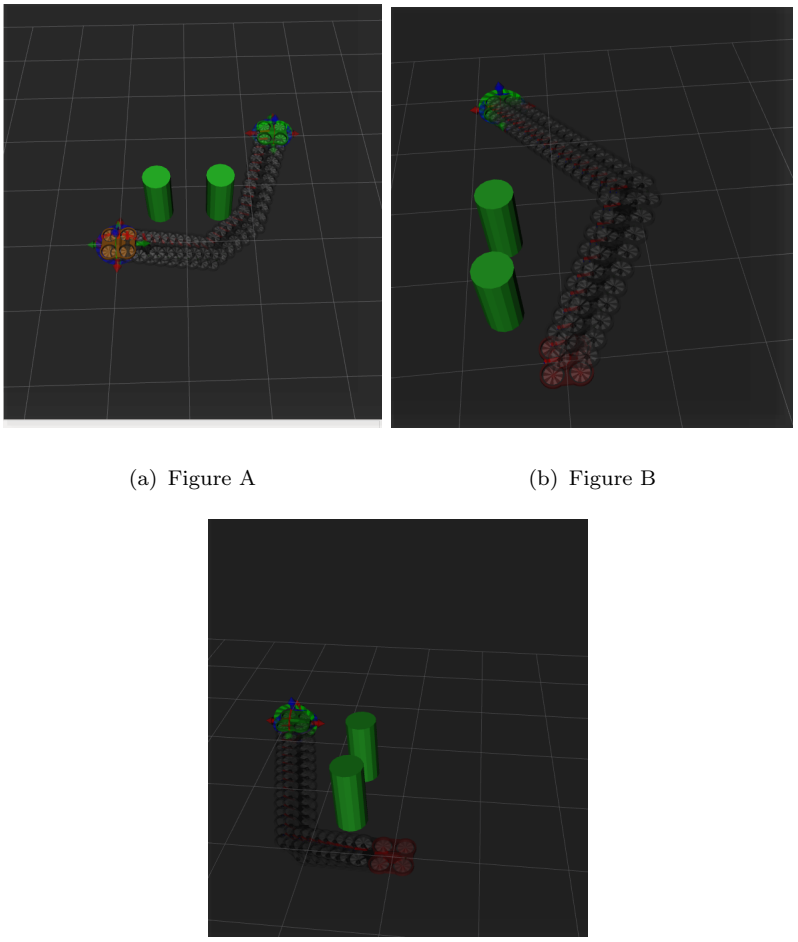
Also there were many experiment made with two cylindrical obstacles in front of the UAV and the result were quite encouraging. We saw that in few paths planned by Move It the UAV couldn't get enough space to avoid the obstacle in front and did the collision. Out of 5 trials made with 2 obstacles in front the the percentage of successful avoidance was about 70 percent which means out of 5 trials 3 were successful avoidance from the path planning done by Move It. The figure below shows the screen shots of the path made by UAV while unsuccessful avoidance.



(a) Figure A (b) Figure B

FIGURE 6.4: Collision with 2 obstacle in front (Unsuccessful avoidance)

In the above figure we can see that although the UAV tried to avoid the obstacle but the deviation away from obstacle wasn't enough to avoid them. Next lets see the successful attempts made by UAV to avoid the obstacle in front.



(a) Figure A

(b) Figure B

(c) Figure C

FIGURE 6.5: Successful attempts with 2 obstacle in front

The results with the MoveIt approach using LIDAR as sensor is tested successfully and the results were successfully validated. This approach although can only be used with 3D sensors, like depth cameras, LIDAR etc and hence is very concise to those particular sensors. Lets now analyze the results we got from our proposed approach using obstacle size.

6.3 Results - proposed method

We have taken the results with single obstacle in front of the UAV and validated the results in different perspectives. We can see how the changing size of the

obstacle information is used to simply warn the UAV about the apparent depth and avoid it by flying up until the nearest contour goes out of the ROI (region of interest).

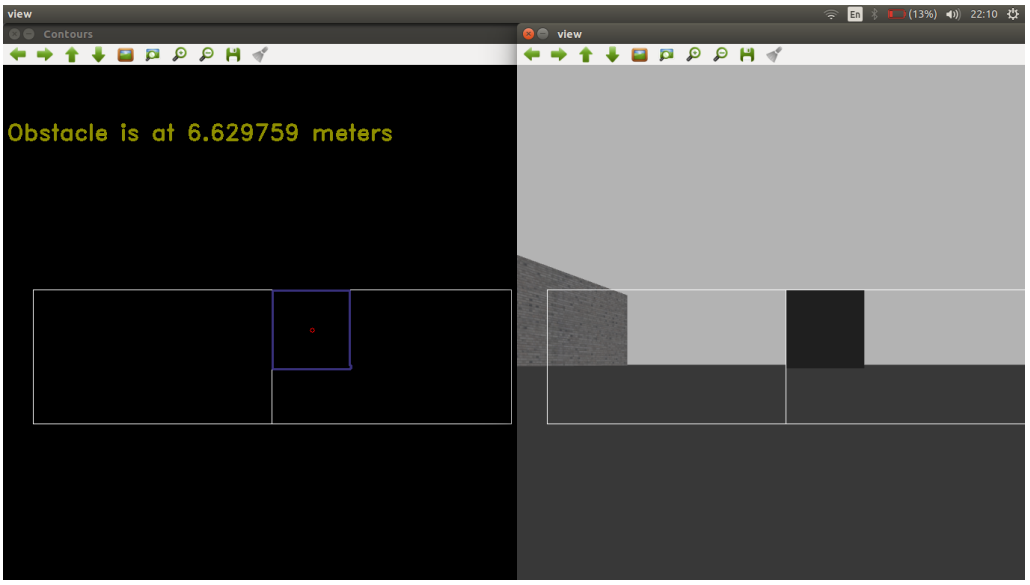


FIGURE 6.6: Distance to obstacle

In the above screen shot we can clearly see that the distance from the UAV camera and obstacle is found out to be 6.6 meters (approximate). We have also recorder a graph with distance to the obstacle vs time which can be seen in the figure below.

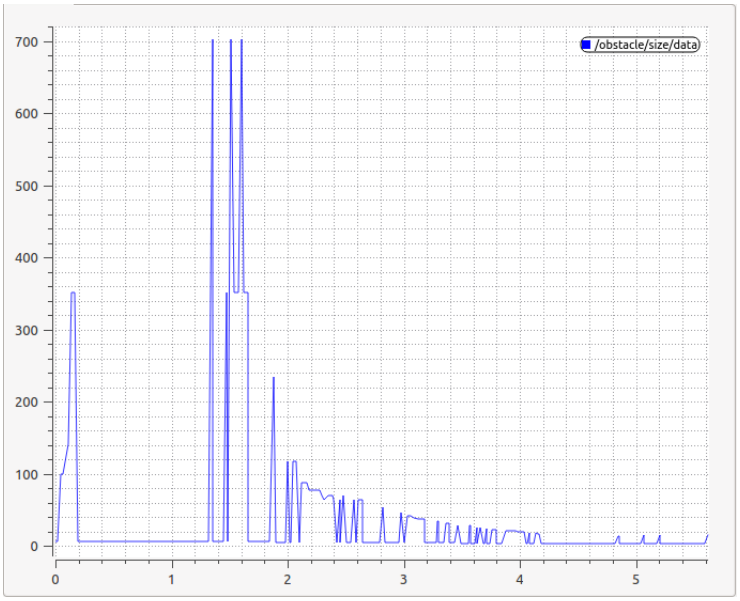
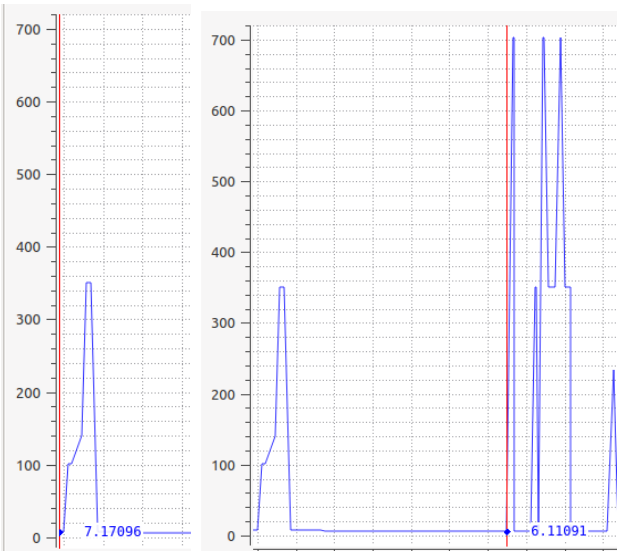
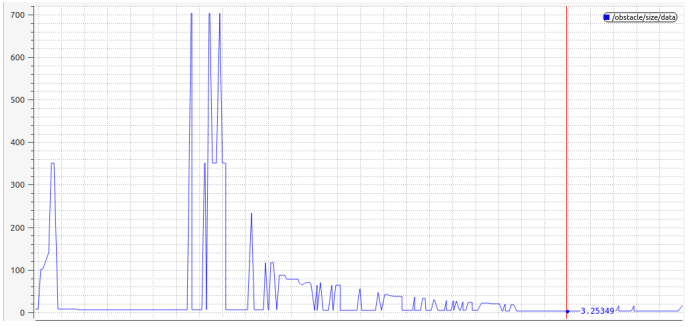


FIGURE 6.7: Distance (y-axis) vs time (x-axis)

In Fig. 6.7 we can clearly see the change in distance between the object and UAV compared with the starting point, which was approximate 7 meter. The spikes in the graph is the result of error detection for few milli seconds, which didn't had much impact over the UAV but the reading and hence we can see several spikes. To understand the reading of the distance let take a look at the graph more closely.



(a) Figure A (b) Figure B



(c) Figure C

FIGURE 6.8: Close look at the Distance reading from Fig. 6.7

As we can see in above 3 parts of the Fig. 6.7, we see that the distance reading at starting was 7.1 meters and then 6.11 meters in the middle and at last 3.25 meters before it gets stable and vanishes as the UAV did the maneuver to the upward direction to avoid the obstacle.

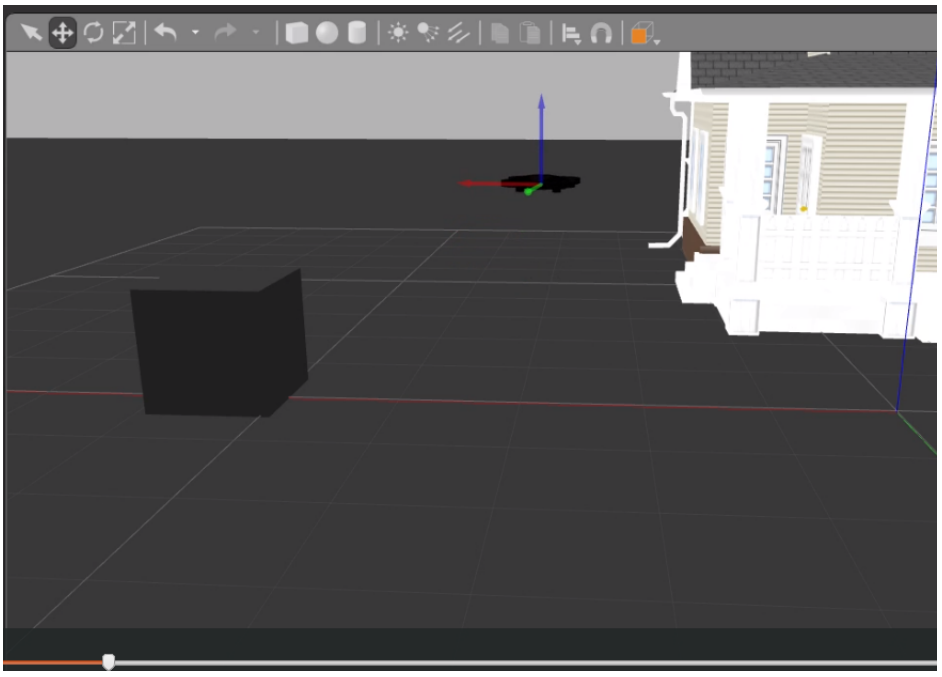


FIGURE 6.9: UAV flying upwards until obstacle is out of ROI

Hence the new approach was tested and implemented. The result with the triangulation method was quite interesting and not bad for the these test on a simulated environment. From a distance of 6 meters to fully overcome the obstacle it took about 6 to 7 seconds to complete hovering and avoiding tested over i5-7200U CPU with 2.50 GHz x 4 processor. The object was kept over a distance of approximately 6 to 7 meters from the drone on a specific marker and each time the distance from the object was with the accuracy of +-0.3 meters, although the avoidance started with almost at 3 meters of the distance, depending on the value provided in the ROS node. With few more efforts this approach can lead to real time obstacle detection with 2D camera, and hence can help in not increasing the payload of the vehicle. This approach is very new in the field of obstacle detection and hence with little more effort this can be a more effective way of obstacle detection and avoidance. We can clearly see that the results were not bad after the approach was tested in simulation. The system was tested and simulated on HP Notebook Linux machine with 8Gb RAM and Intel core i5 -7200U CPU running at 2.50GHz x 4 quad core processor.

Chapter 7

Conclusion

This thesis aimed to present an approach in the context of obstacle avoidance in UAV's. The thesis also implemented the obstacle avoidance method with laser sensor using MoveIt as the path planner, and the results we implemented and tested. However, the proposed approach used a single camera and using image processing algorithm proved that it is possible to use size as the only parameter to understand the nature of the obstacle and hence avoid it. Although this approach is for now is just a proof of concept and the results can be more overwhelming if there is more deep work done over it.

This thesis majorly focus on obstacle detection algorithm which was developed using simulation on ROS. The UAV model used was ARDrone and the path planning tool was MoveIt using one approach, while a generic avoidance was performed so as to avoid the obstacle using proposed method. In this thesis we also studied various state of art techniques of obstacle avoidance for UAV and fundamentals involved in this master thesis.

The algorithm captures images in frames from the camera and apply edge detection so as to get the boundary of the obstacles in front. This is followed by thresholding and then finding the contours over the thresholded image. If the final contour is bigger than the ROI (Region of interest) then its considered to be a closer object and if not the obstacle is considered to be relatively far, and using this information the depth (approximate) was calculated using triangle similarity

method which then helped us to understand the obstacle distance in much better way. Hence the result of this research was based on this new approach for obstacle detection and the output was gathered as a proof of concept to show that this is very simple and effective approach in the field of obstacle avoidance in UAVs.

Chapter 8

Future Work

Based on the study and results of this research the future work to be included is to add more obstacles in the environment and understand the behavior of the UAV for those obstacles. There can be multiple ways of improving this research :
The first approach may involve the addition of more obstacle and improve the image processing algorithm so as to get more clear imaging results from the camera. The second approach can be to move the UAV upward when the obstacle was seen closed to the UAV, this can help if we have 2 or more obstacles close to each other and all with different height. Using this approach if the drone sees 2 or more obstacle lying close to each other (difference between centroid of 2 contours is less) then the UAV will start flying upward so that both the obstacles are evenly avoided. More obstacles can be added in the scene and the algorithm can be made more complex in terms of how to deal with multiple obstacles. Which can really be a good study and work in this field of research.
The research can also be improved by developing a separate plugin for mono camera in the MoveIt work space and use that camera plugin to update the information of the obstacles in front and avoid them accordingly. Also kinematics of the UAV can be added to the algorithm for it us to know how much should the UAV deviate from the obstacle to avoid it.

Chapter 9

Documentory References and Other Source Of Information

[1] Tomoyuki Mori and Sebastian Scherer-"First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles"- Robotics and Automation (ICRA), 2013

[2] JeongOog Lee, KeunHwan Lee, SangHeon Park, SungGyu Im and Jungkeun Park-"Obstacle avoidance for small UAVs using monocular vision"

[3] Lazaros Nalpantidis and Antonios Gasteratos-"Stereo vision Based Algorithm for Obstacle Avoidance"

[4] Allen Ferrick, Jesse Fish, Edward Venator and Gregory S. Lee- "UAV Obstacle Avoidance Using Image Processing Techniques"

[5] Daniel Magree, John G. Mooney, Eric N. Johnson - "Monocular Visual Mapping for Obstacle Avoidance on UAVs"

[6] Obstacle avoidance drones and techniques defined here :
https://www.dronezon.com/learn-about-drones-quadcopters/top-drones-with-obstacle-detection-collision-avoidance-sensors-explained/

[7] Lazaros Nalpantidis, Ioannis Kostavelis, and Antonios Gasteratos- "Stereovision-Based Algorithm for Obstacle Avoidance"

[8] P.Daniel Ratna Raju, G.Neelima - "Image Segmentation by using Histogram Thresholding "

[9] Jaysinh Sagar and Arnoud Visser - "Obstacle avoidance by combining background subtraction, optical flow and proximity estimation"

[10] E. Jebamalar Leavline, D. Asir Antony Gnana Singh - "On Teaching Digital Image Processing with MATLAB" - American Journal of Signal Processing

[11] J. M. SAGAR - "Obstacle Avoidance using Monocular Vision on Micro Aerial Vehicles" -

[12] ROSARIA, Open Source Robotics Foundation, March 1, 2017. [Online]. Available: http://wiki.ROS.org/ROSARIA

[13] Beginner: overview, Open Source Robotics Foundation.
http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b1

[14] ROS overview, Open Source Robotics Foundation.
http://gazebosim.org/tutorials/?tut=ROS_overview

[15] MoveiT concepts : https://moveit.ROS.org/documentation/concepts/