AN END-TO-END AUTONOMOUS UAV SYSTEM IN GPS-DENIED AND UNSTRUCTURED ENVIRONMENTS

CAPSTONE PROJECT (MEE4099)

A PROJECT REPORT

Submitted in partial fulfillment of the
requirement for the award of the
Degree of

BACHELOR OF TECHNOLOGY

IN

MECHANICAL ENGINEERING

By

Jerrin Bright (18BME1004)

Suryaprakash Rajkumar (18BME1183)

Under the Guidance of

Dr. Arockia Selvakumar

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF MECHANICAL ENGINEERING

VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI - 600127

(APRIL 2022)

CERTIFICATE

This is to certify that the Project work titled “**An End-to-End Autonomous UAV System in GPS-Denied and Unstructured Environments**” that is being submitted by **Jerrin Bright, Suryaprakash Rajkumar** is in partial fulfillment of the requirements for the award of **Bachelor of Technology**, is a record of bonafide work carried out under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma, and the same is certified.

Guide Name

Dr. Arockia Selvakumar

The thesis is satisfactory

Internal Examiner **External Examiner**

Approved by

Head of Department

ACKNOWLEDGEMENT

We would like to thank Dr. Arockia Selvakumar for giving us an opportunity to work on the project and guiding us in all possible ways to realize the project. We would also like to thank Dr. Jegadeeshwaran S and Dr. Saravanakumar D for their guidance. This project would not have been possible without the constant support of Dr. Sreekanth Dondapati, Dean, School of Mechanical Engineering; and VIT management for giving free access to laboratories.

Jerrin Bright (18BME1004)

Suryaprakash Rajkumar (18BME1183)

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

Autonomous navigation of UAVs is vital for the safe and robust execution of intricate operations. Especially, building a reliable system that can traverse GPS-denied, unstructured and unknown environments that the UAV hasn't observed, is challenging. The complete automation of UAVs can play a vital role in several important operations including forest fire detection and control, search and rescue operations, precision agriculture, surveillance of properties, monitoring of bridges, etc., which are very difficult for a human to do. There are a lot of decision-making parameters while building a UAV considering limitations in the payload which in turn affects the number of sensors that can be used and flight time; stability; agility; and so on. Refer to APPENDIX C to know more about UAV systems, their advantages, applications, and limitations in detail.

Considering all the above-mentioned issues and necessity, we have proposed an end-to-end completely UAV system that can effectively navigate in an unknown, cluttered, unstructured, and GPS-denied environment, simultaneously inspecting the traversing environment using novel mechanisms driven by classification and detection modules. The system is made in a very modular manner, thereby resulting in the fusion of any type of inspection task based on the classes fed to the model and it is highly independent of the environment that the UAV is subject to navigate.

The robustness of the system was tested by evaluating modules of the system in different simulation environments and using various state-of-the-art datasets used to determine the robustness of the networks.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Thesis Contributions

An end-to-end completely autonomous UAV navigation system in GPS-denied and unstructured environments was proposed via this thesis. Some notable contributions of the thesis include:

1. A novel simple yet effective learning-free approach for obstacle avoidance
2. An inspection module heavily relying on a novel *sense-switch-act* approach.
3. A novel image classification network ME-CapsNet that retains important features for efficient detection.
4. A novel localizing technique leveraging inertial data and using sensor fusion via Extended Kalman Filters with ORB featured frames.
5. In-depth analysis and comparison of learning-based and learning-free obstacle avoidance approaches in terms of accuracy, computation, and agility.
6. A custom dataset consisting of more than 10,000 image frames and their corresponding navigation command and state of the UAV.
7. A modular plug-and-run inspection system capable of using for various real-time applications with ease.

Some of the thesis contributions in terms of publications include:

1. Conference Paper - Submitted to 8th IEEE CONECCT 2022
   ME-CapsNet: A Multi-Enhanced Capsule Network with Routing Mechanism [Pre-print]
2. Journal Paper - Submitted to MDPI Sensors Journal
   A Comprehensive Study on Autonomous Navigation using Learning-based Techniques for Robotic Systems.

1.2 Thesis Overview

Unmanned Aerial Vehicles or UAVs are cyber-physical systems that have been finding applications in diverse fields and can be controlled remotely or autonomously.

However, UAVs face various challenges when it comes to autonomous navigation owing to the assumptions and hardware failures in existing algorithms.

UAVs often have payload constraints as a result of their size and power consumption. Cameras have proven to be compatible especially in small UAVs, considering their size and weight. Thereby, we have used visual sensors for the successful implementation of our tasks. Understanding scenes and extrapolating tasks for the UAV is not possible from the raw frame captured from the camera. Therefore, the efficient transformation of the raw frames is vital to developing smart navigation strategies for the UAV. Also, the use of GPS tends to reduce the reliability of the UAV system considering its low cycle rate and no penetrability.

Thus, we are proposing an End-to-End System (E2ES) for UAVs in GPS-denied and unstructured environments that can simultaneously do inspection tasks. Two example tasks taken in this work include forest fire detection and search and rescue operations. The two most important modules of autonomous systems that we aim to explore here for these tasks include the navigation module and the inspection module focusing on unstructured, GPS denied environments.

For *navigation*, we developed two algorithms, each using state-of-the-art automation learning-based and learning-free techniques. The performance of both the algorithms was tested extensively to understand the significance of learning-based techniques when compared to learning-free techniques. Also, the data (images, pose, UAV angles) from learning-free obstacle avoidance was used to learn the best strategy for navigation when the learning-based technique was leveraged for autonomous navigation. Also, the network to learn the output yaw commands was leveraged from sub-modules of the inspection module, thereby proposing a compact E2ES.

For *inspection,* a sense-switch-act novel approach was adopted. The goal of this approach was to reduce the computational overhead of the inspection task. First, an image classification network is used to detect the presence of a task-specific class in the perceived field of view of the UAV. We have built a novel Capsule Network architecture leveraging strategic fusion with Squeeze-Excitation Networks. Expectation-Maximization Routing replaces the traditionally used max-pooling to retain feature information. The proposed algorithm effectively recalibrates the channel-wise and spatial-wise relationship of the features. Also, task-specific inspection is done to expand the scope of the applications of the UAV system, maintaining its accuracy. After observing the class in the environment, the image classification network is switched to an object detection model to detect the exact location of the task-specified class. This way, computation cost was reduced by maintaining the robustness of the system.

CHAPTER 2

RELATED WORKS

2.1 Obstacle Avoidance

2.1.1 Learning-Free

The learning-free approach is the most traditional obstacle avoidance strategy used. When learning-free navigation is done in GPS-denied environments, the use of visual odometry becomes mandatory. It can be done using features [1], [2], and [3] or using appearance [4], [5], and [6] or both feature and appearance-based techniques [7]. RADAR was used for obstacle avoidance in [8] and [9] and it was observed to consume more power and a significant increase in payload due to the sensor. Thus, considering the disadvantages of such sensors, visual sensors are used.

Obstacle avoidance using visual sensors can be done in three ways: monocular cues [10], [11]; stereo vision [12], [13] and motion parallax [14], [15]. Here we will be using monocular cues instead of a stereo setup due to the following two reasons: the payload of a stereo setup; and the distance of measurement are greatly limited by the baseline, beyond which it is similar to a monocular sensor. Stefan et al. [16] proposed using of stereo vision and optical flow to evade obstacles experimenting in sub-urban areas. Huili et al. [17] presented a work using depth images from image processing techniques and obstacles were estimated using EKF.

Coming to the submodules of vision-based monocular obstacle avoidance techniques, it is of three important modules. They are perception, planning, and control modules. For monocular vision, depth estimation through deep learning techniques plays a vital role. Some depth estimation techniques using monocular vision data include the works of [18], [19], and [20]. There are two path planning approaches- randomly sampling and optimal search algorithm. Some state-of-the-art path planning algorithms include [21], [22], and [23] focusing on optimal search algorithms, and [24] and [25] focusing on a random sampling approach.

2.1.2 Learning-Based

The dominance and effectiveness of learning-based approaches have been demonstrated in recent breakthroughs in AI. In 1989, Pomerleau et al. [26] introduced the first IL method for navigating in an unfamiliar environment. ALVINN was the name of the architecture, and it was able to maneuver at speeds of up to 25 m/s utilizing NAVLAB, a Chevy test van. ALVINN's performance prompted various investigations on the concept, including widening it and conducting extensive research into aligning methods to it. This section, as previously said, provides an in-depth examination of all of the most widely used IL approaches.

Muller et al. [27], inspired by ALVINN, presented a few architectural improvements to get more robust navigation results. To extract greater spatial information from the input images, the monocular frame was modified to a stereo configuration, and instead of FCNN, a 6-layer CNN was employed. In addition, rather than using road-following data like ALVINN, an off-road dataset was used for training. Following these advancements, a slew of algorithms emerged, dividing the learning of behavior policy into BC and IRL, and then further subcategories. Park et al. [28] leverage a human expert to extract the required information from an image obtained from the Gazebo simulation environment, which communicates via the ROS framework.

According to Kumaar et al. [29], the taught policy was initially tested in entertainment programs such as GTA SA and Mario Kart. Wang et al. [30] use vehicle pose and sub-goals to estimate the sub-goal angle, which is then used in conjunction with image sequences to train (using a new angle branching network). Loquercio et al. [31] developed an autonomous navigation architecture that ran at roughly 10 m/s with a 60% success rate and at 7 m/s with a 100% success rate, making it the most agile architecture conceived and tested in real-time. Bansal et al. [32] presented ChaufferNet, a unique method that employed mixed data (real-time and simulated) to drive a car in real-time while restricting exploration.

2.2 Inspection

The deep learning regime has improved significantly since the advent of deep convolutional networks [33]. Stacking layers as a result of deep networks boosted performance tremendously, but it also brought vanishing gradient issues [34]. Additionally, the use of operating layers such as max-pooling resulted in a loss of spatial information. Instead of stacking layers, CapsNet tries to tackle these problems by layering them. Deep learning research has gotten a lot of traction as a result of these new methodologies. Given its relevance and scope of improvement over older techniques, CapsNet is regarded as a giant leap forward in neural networks. Here are some examples of effective CapsNet research. Mazzia et al. [35] presented a non-iterative routing strategy that takes advantage of self-attention mechanisms. Deliege et al. [36] used a bespoke Hit-or-Miss layer to create a counterpart for the DigitCaps layer. To increase the overall performance of the network, Huang et al. [37] recommended adding two attention layers to the original CapsNet.

Yang et al. [38] propose a method for improving CapsNet's feature extraction capacity by employing a modified version of ResNet dubbed Res2Net and SENet in the base layers to improve the CapsNet's performance. This is the work that comes closest to our work in terms of relevancy. To boost the performance of the squeeze operation, we strategically placed SE blocks between several layers and used a more powerful kind of pooling.

There are a lot of object detection techniques including DSSD [39], YOLO [40], SSD [41], and G-CNN [42]. You Only Look Once (YOLO) is a powerful object detection model and tool. YOLO v3 is based on a Darknet variation that had a 53-layer network trained on ImageNet. For detection, 53 more layers are put on top of it, giving YOLO v3 a 106-layer fully convolutional underlying architecture making the model architecture suitable for object detection with high accuracy. YOLO9000 [43] was proposed over YOLOv2 architecture that is capable of detecting more classes than COCO, by using labels from both COCO and ImageNet. It uses hierarchical classification from WordNet [44], where classes are depicted in a tree-based structure.

Single Shot Detector (SSD) is a deep neural network-based method for detecting objects in images by just using a single neural network. Over various aspect ratios, the SSD technique discretizes the output space of bounding boxes into a set of default boxes. The approach scales each feature map location after discretization. To naturally handle objects of varied sizes, the Single Shot Detector network integrates predictions from numerous feature maps of various resolutions.

CHAPTER 3

OVERVIEW OF THE SYSTEM

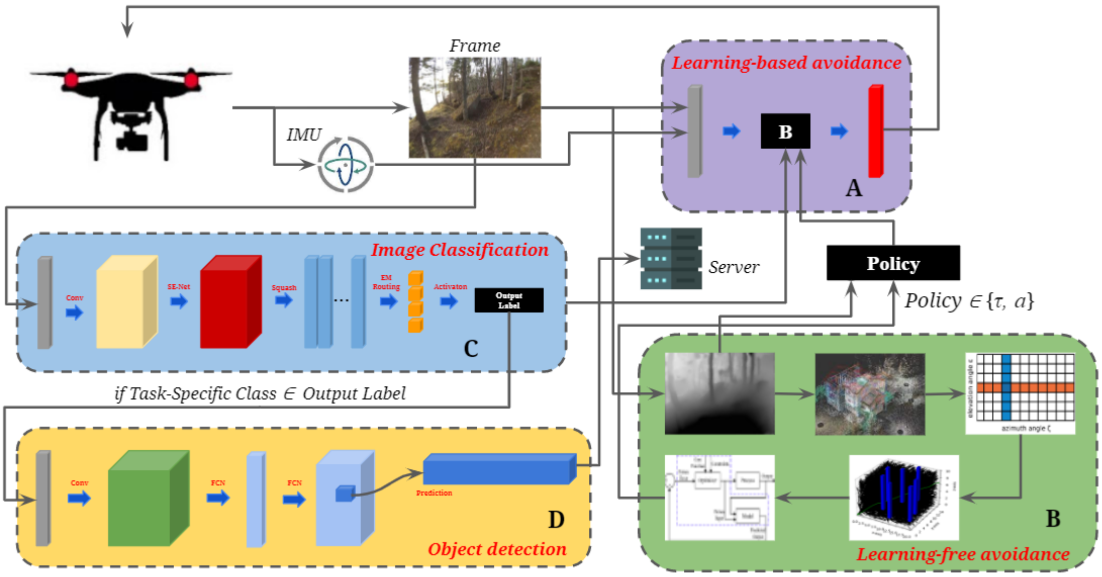


Figure 1 Overall architecture of the proposed system

In this thesis, we study the various shortcomings of autonomous navigation systems focusing on UAVs, especially when traversing in unknown, cluttered environments and building robust solutions for those. The objective is to achieve robust reliable UAV navigation by traversing to the farthest distance specified avoiding obstacles and doing a specific task in the environment simultaneously. The tasks taken in this thesis to explain the robustness of the proposed system include Search and Rescue, Forest-fire detection, and agriculture burn to name a few.

We have divided the complete architecture into four modules (A, B, C, and D) as shown in Figure 1 as four colored blocks. A and B blocks correspond to autonomous navigation with obstacle avoidance in an unknown environment using a learning-based and learning-free approach respectively. C block corresponds to image classification used for inspection tasks from the UAV perspective to detect the presence of a particular class assigned specifically to a task. D block corresponds to the object detection sub-module that identifies the exact position of the class sensed by the image classification sub-module (represented as C in Figure 1). All the four modules and its sub-modules is discussed in detail in the following chapters

CHAPTER 4

BASIC CONCEPTS

4.1 Quadcopter Dynamics

Quadrotors are 6 D.O.F high-performance systems that require proper control elements on every D.O.F for efficient and safe flight. These control elements are essential to creating a stabilized path from a start position to a target position. It is of high importance that the system must be dynamically modeled to understand the system and use it to model a control system for the aforementioned purpose.

State variables are variables that define the current state of the system. Quadrotors being a 6 D.O.F system, is defined with 12 state variables i.e., position, velocity, and attitude of an airframe in the inertial frame; and angular rates.

$$\vec{x} = \left[X,\ Y,\ Z,\ \dot{X}, \dot{Y}, \dot{Z},\ \phi,\ \ \theta,\ \ \psi,\ \ p,\ \ q,\ \ r\right]^T \qquad (1)$$

Since there we take control over all the axis, we define 4 control elements and their associated control inputs

$$\vec{\boldsymbol{u}} = \ [\,u_1, u_2, u_3, u_4]^T \ = \ [T_\Sigma, M_1, M_2, M_3]^T \qquad (2)$$

The quadcopter dynamics and mathematical modeling is used effectively to design the MPC control and kinodynamically plan an optimized path for traversing. It is explained extensively uniquely for our quadcopter system in Section 5.3.1.

4.2 Camera Calibration

Camera calibration is performed to reduce the radial and tangential distortion. Calibration helps to determine the intrinsic and extrinsic parameters along with the distortion coefficients. Intrinsic parameters include focal length and optical centers and extrinsic correspond to the rotational and translational vectors. First, the image points are determined on the checker board of pre-defined size, then the undistorted image is generated once calibration is done.

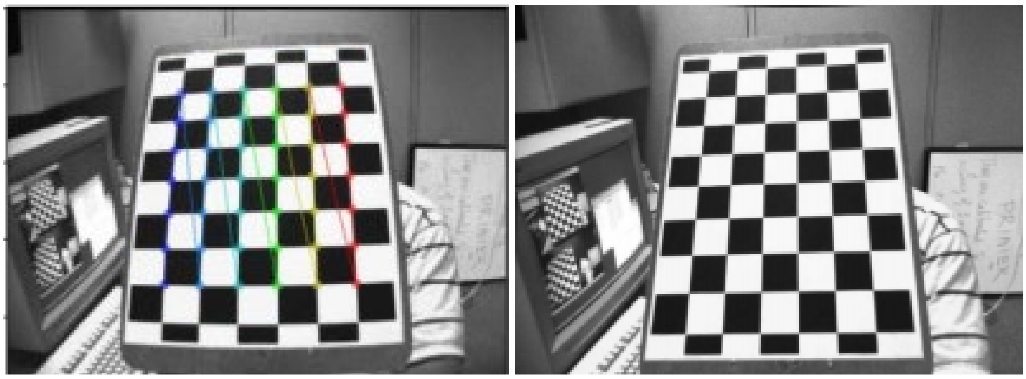


Figure 2 Camera calibration

Figure 2 represents the process of camera calibration. A checker board of predefined size is used for the process. The software captures in greyscale mode. Corner detection module helps in identifying corners and distribute several features on the checkerboard then the user moves the board to various pose in the field of view and system captures several images to approximate intrinsic and extrinsic parameter of the camera.

4.3 Image Preprocessing

Image preprocessing is done to reduce the noise observed in the image frame from the UAV visual sensor due to uncertainities in measurement and sensor bias. There are various preprocessing techniques including smoothening or blurring, rotation, warping, etc. We will be studying the performance of each preprocessing technique in the subsections. Smoothening or blurring is one of the most commonly used preprocessing techniques that helps in removing noises in the image leaving most of the image pixels intact. Some of the smoothening methods experimented with are image filtering, gaussian blurring, bilateral filtering, and median blurring.

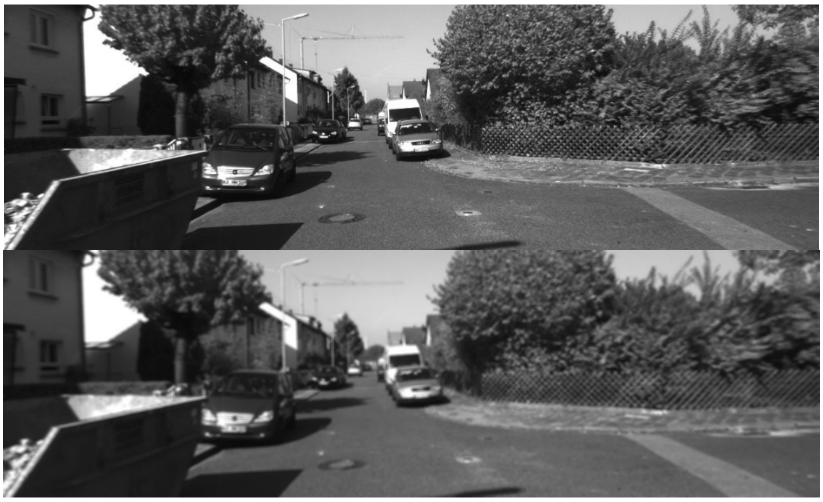


Figure 3 Image preprocessing using smoothening

Rotation and warping are a few other preprocessing techniques along with converting the frames into greyscale and normalizing the image. Many of these techniques were experimented with the system and results have been logged in subsection 7.1.

4.4 Sensor Fusion

Sensor fusion is generally done to enhance the reliability of the measurement model of the system significantly with the use of multiple sensors. We will be fusing the visual information with inertial measurements using Extended Kalman Filters.

Linear equations produce gaussian results when gaussian filter is applied but nonlinear equations will not produce gaussian results. And we also know that in the real world most problems are nonlinear in nature. Therefore, the solution for this problem will be to approximate the nonlinear model to linear model. This approximation can be done in various ways, for example by using the Taylor series. That's, Gaussian on the nonlinear function will be done followed by taking the mean first and then performing several derivatives to approximate it. The first derivative of a Taylor series is called as Jacobian Matrix. This Jacobian Matrix converts the nonlinear curve to a linear function.

The extended Kalman filter is based on a nonlinear dynamic system with a motion model and the measurement model is:

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \; z_t = h(x_t) + \delta_t \tag{3}$$

In the above equation, x_t represents the measurement model, where g is a non-linear function that depends on t and ε_t is the noise in the model. The same is with the measurement model z_t where function h is a nonlinear function and δ_t is the noise in the measurement model.

4.5 Localization

Determining the position and orientation of a robot using the data measured from the sensors is called odometry or localization. When visual sensors are used to identify the pose of the robot, it is termed visual odometry or vision-based localization. Visual odometry provides computationally cheaper while providing more accurate odometry results when compared to GPS, wheel odometry, sonar localization, or INS. Visual odometry can be done in either of two ways- feature-based or appearance-based. Here, we will be focusing on implementing feature-based visual odometry. Refer to APPENDIX F to know more about odometry and its phases in detail.

Thus, for feature-based odometry, choosing the right feature extractor is primitive. There are a lot of feature extractors including FAST [45], SIFT [46], ORB [47], SURF [48], Shi-Tomasi [49], etc. Table 1 shows all the properties of the most popularly used feature extractors. In our work, we will be using ORB feature extractors considering the proper balance between accuracy and computation. The experimentation behind selecting ORB can be studied in Section 7.2.

Table 1 Properties of different feature extractors

	ORB	**SIFT**	**SURF**	**KAZE**	**AKAZE**	**BRISK**
Origin	2011	1999	2006	2012	2013	2011
Scale Invariance	YES	YES	YES	YES	YES	YES
Rotation Inv	YES	YES	YES	YES	YES	YES
Keypoint Type	FAST	DoG	Hessian	Hessian	Hessian	-
Descriptor Type	Binary	Integer	Real	Real	MLDB	Binary
Descriptor Length	32	128	64	128	-	64

ORB stands for Oriented FAST and Rotated Brief. In ORB, an oriented FAST algorithm is used to calculate and find the key points from the image and rotated BRIEF descriptors are used. Coming to the feature detection part, the image pyramidal representation of frames is done for scale invariance, and then features from each pyramidal layer are extracted. Firstly, to estimate the orientation, a centroid will be estimated from the moment equations.

$$C = \left(\frac{m10}{m00}, \frac{m01}{m00}\right) \tag{4}$$

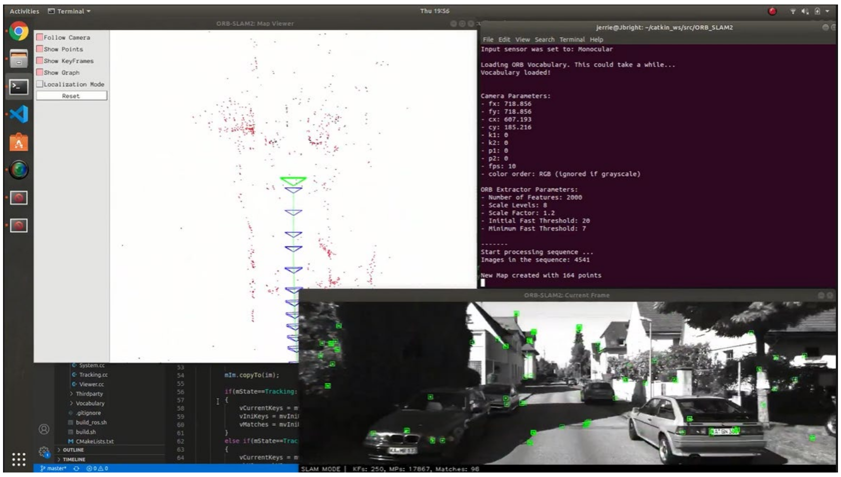
Then, an orientation patch can be obtained using the below-mentioned equation.

$$Orientation = a\ x\ tan2\ (m01, m10) \tag{5}$$

Rotated Brief descriptors are used once extraction is done. The orientation patch obtained is used to compute the BRIEF descriptor operation:

$$g(p, \theta) = f(p)|\ (x, y)\ \epsilon\ S \tag{6}$$

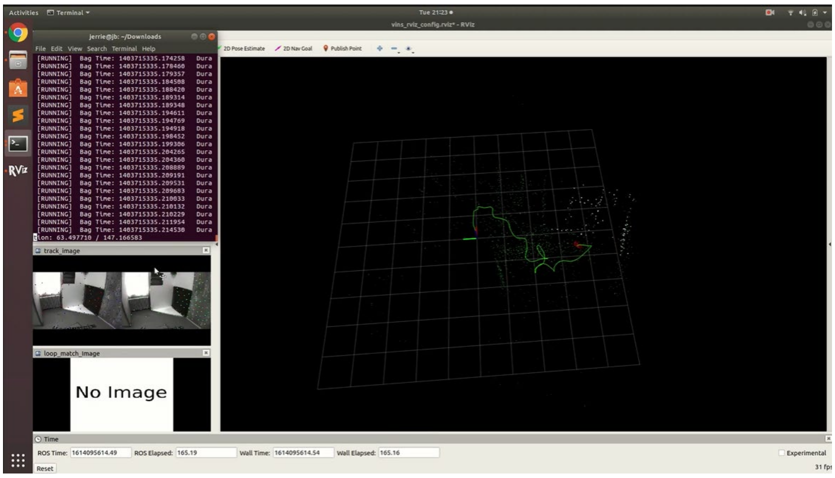
Here we used three different SLAM approaches (ORB, VINS, and RTABMap) to estimate the odometry of the robot. Refer to APPENDIX D to learn more about SLAM techniques. The best approach was then experimentally selected and used in the final system. Now we will look into the three techniques used for localization in detail. The source codes for all the approaches have been open-sourced and can be obtained from APPENDIX A.



```
Input sensor was set to: Monocular

Loading ORB Vocabulary. This could take a while...
Vocabulary loaded!


Camera Parameters:
- fx: 718.856
- fy: 718.856
- cx: 607.193
- cy: 185.216
- k1: 0
- k2: 0
- p1: 0
- p2: 0
- fps: 10
- color order: RGB (ignored if grayscale)

ORB Extractor Parameters:
- Number of Features: 2000
- Scale Levels: 8
- Scale Factor: 1.2
- Initial Fast Threshold: 20
- Minimum Fast Threshold: 7

-------
Start processing sequence ...
Images in the sequence: 4541

New Map created with 164 points
```

Figure 4 Implementation of ORB SLAM

ORB SLAM is a visual slam-based technique that uses a monocular or Stereo camera to perform localization in a 3D environment. ORB SLAM uses oriented fast and rotated brief or ORB features which are known for their efficiency and ability to match hard to define features in the environment while keeping it highly efficient in terms of computation. Similar to VINS a keyframe is initialized during the start of the process with about 200-300 features defined over the frame to initialize the reference axis and sparse point cloud map. With the use of the keyframe, the subsequent frames are compared and feature matched to triangulate the position concerning the initial keyframe. The whole process of localization using ORB SLAM is highly efficient due to the efficiency of ORB features and the ability to parallel compute the complete process. A loop closure occurs when a complete match of the frame over a bias occurs then the whole localization is aligned to create a more accurate map and odometry data.

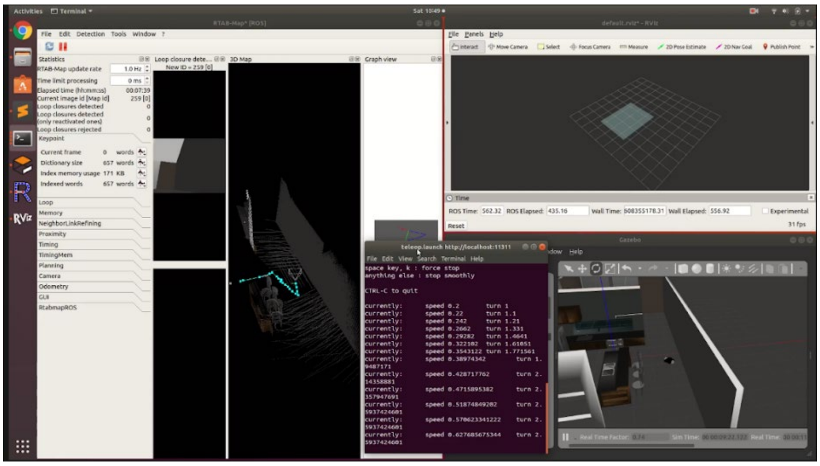


```
jerrie@jb: ~/Downloads
[RUNNING]  Bag Time: 1403715335.174258   Dura
[RUNNING]  Bag Time: 1403715335.178460   Dura
[RUNNING]  Bag Time: 1403715335.179357   Dura
[RUNNING]  Bag Time: 1403715335.184508   Dura
[RUNNING]  Bag Time: 1403715335.188420   Dura
[RUNNING]  Bag Time: 1403715335.189314   Dura
[RUNNING]  Bag Time: 1403715335.189348   Dura
[RUNNING]  Bag Time: 1403715335.194611   Dura
[RUNNING]  Bag Time: 1403715335.194769   Dura
[RUNNING]  Bag Time: 1403715335.194918   Dura
[RUNNING]  Bag Time: 1403715335.198452   Dura
[RUNNING]  Bag Time: 1403715335.199306   Dura
[RUNNING]  Bag Time: 1403715335.204265   Dura
[RUNNING]  Bag Time: 1403715335.204360   Dura
[RUNNING]  Bag Time: 1403715335.208889   Dura
[RUNNING]  Bag Time: 1403715335.209191   Dura
[RUNNING]  Bag Time: 1403715335.209531   Dura
[RUNNING]  Bag Time: 1403715335.209683   Dura
[RUNNING]  Bag Time: 1403715335.210033   Dura
[RUNNING]  Bag Time: 1403715335.210132   Dura
[RUNNING]  Bag Time: 1403715335.210229   Dura
[RUNNING]  Bag Time: 1403715335.211954   Dura
[RUNNING]  Bag Time: 1403715335.214530   Dura
tion: 63.497710 / 147.166583
```

ROS Time: 1614095614.49 ROS Elapsed: 165.19 Wall Time: 1614095614.54 Wall Elapsed: 165.16

Figure 5 Implementation of VINS SLAM

VINS stands for the Visual-Inertial SLAM system. For this implementation, a monocular camera is used along with the frame-aligned IMU for getting inertial data for the corresponding frame aligned properly with the visual frame. It uses Shi-Tomasi features and IMU data to track features in multiple frames. The initial step involves choosing the first frame as the keyframe and initializing all the features using the BRIEF descriptor. About 100-300 features are described for every frame. Once the initial frame is initialized then we proceed to subsequent frames, not just translation it also can be pure rotation as now the system is equipped with a gyroscope the system can easily triangulate the data to track the features, and with pure rotation alone, we can compute a parallax. Adding an inertial component to a SLAM system not only improves the pose and the state estimation of the system it also helps in the camera extrinsic calibration.



```
teleop.launch http://localhost:11311
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1
currently:      speed 0.22      turn 1.1
currently:      speed 0.242     turn 1.21
currently:      speed 0.2662    turn 1.331
currently:      speed 0.29282   turn 1.4641
currently:      speed 0.322102  turn 1.61051
currently:      speed 0.3543122 turn 1.771561
currently:      speed 0.38974342        turn 1.
9487171
currently:      speed 0.428717762       turn 2.
14358881
currently:      speed 0.4715895382      turn 2.
357947691
currently:      speed 0.51874849202     turn 2.
5937424601
currently:      speed 0.570623341222    turn 2.
5937424601
currently:      speed 0.627685675344    turn 2.
5937424601
```

ROS Time: 562.32 ROS Elapsed: 435.16

Figure 6 Implementation of RTABMap SLAM

RTABMap stands for Real-Time Appearance-Based Mapping. It is a Graph-based SLAM approach based on appearance-based loop closure detection. It checks how likely an image comes from the previous location. Graph SLAMs have better accuracy than FAST SLAM. Here, we used RTAB-Map with stereo only for 6 DoF mapping. Appearance-based SLAM means that the algorithm will use the data obtained from vision sensors to localize the position of the robot and simultaneously map the robot in the environment. Loop Closures are used to determine if the robot has already seen the particular frame before. Local Loop closure is dependent on Visual Odometry whereas Global Loop Closure is independent of the estimated pose that's visual odometry. Thus, when the robot moves, the map expands, and the number of images that are compared increases in turn.

4.6 Obstacle Avoidance

Obstacle avoidance techniques are predominantly divided into learning-based and learning-free approaches. The learning-free approach is the most traditional approach, which includes three modules- perception, planning, and control. But recent research works have shown the dominance of learning-based avoidance. It has mimicked close to how humans evade obstacles. The backbone of learning-based approaches is either demonstration fed as input or rewards assigned for every action. In this subsection, we will be exploring learning-free and learning-based approaches in detail.

4.6.1 Learning-free Approach

4.6.1.1 Perception

Understanding the environment that the robot is traversing is the aim of the perception module. In layman's terms, the perception module should mimic the environment in such a way, that the path planning module has to understand and navigate in that environment. This reconstructed scene is called the occupancy grid. Usually, a 2D occupancy grid is used to represent the environment instead of 3D grids to save computation. Thus, when the reconstruction performance increases, the path planning, and control modules, will tend to generate better, optimized trajectories for navigation. The perception module in general irrespective of the sensors used can be as follows: depth estimation, point cloud conversion, and occupancy grid formation.

Depth estimation refers to estimating the distance between the sensor and obstacles in front of it and the free space relative to the frame. For example, as shown in Figure 7, the dark colors indicate that the objects are far away from the robot and the light color indicates the object is close to the robot.

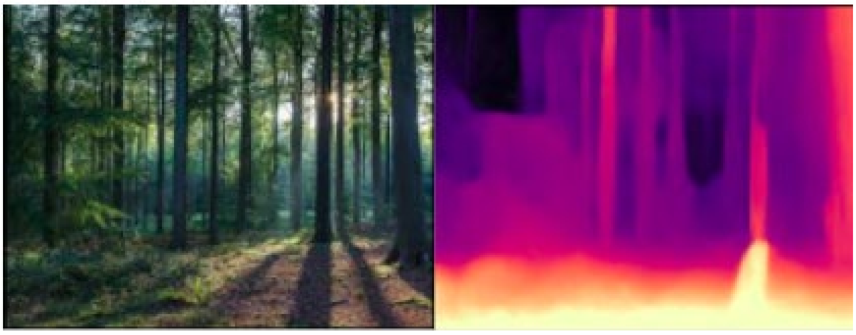


Figure 7 Monocular Depth Estimation

Next, point clouds are generated from the pixel intensities, using the absolute scale factor. The absolute scale factor can be initialized before the start of the system. Thus, we will be extracting 3D coordinates from a 2D image.

Now with the obtained point cloud data, we can create occupancy grids. Occupancy grid formation for monocular vision sensors is the trickiest, considering we don't get the absolute scale of the environment directly. For 2D grid formation, we will take the x and z coordinate and plot it in a grid-based space, thereby obtaining an occupancy grid as observed from the top. This is then fed to other modules of the learning-free obstacle avoidance.

4.6.1.2 Path Planning

Path planning is vital to determine the most suitable and permittable trajectory for the robot. In general, the input to the path planning module is the occupancy grid or the environment grid and the output is the trajectory points. Thus, control points are subsequently passed through the control module which generates the output PWM for the motors. The most suitable path is optimized in general by path planning techniques. But most of the techniques available don't take into account the dynamics of the robot. Thus, adding kinodynamic properties to the path planning technique is important for navigation in real-time. Refer to APPENDIX E to know more about different path planning techniques in detail.

Kinodynamic planning is a method of generating trajectories considering the velocity, acceleration, and force boundaries of the defined system. The dynamics of the quadrotor in all the axis is considered while defining the bounds for the planning to produce a time-optimal control solution. Generally, in the proposed planning method, the probabilistic roadmap approach is used to generate waypoints then connected by feasible trajectories. The generated trajectories are then converged to a single solution based on efficiency and the traversing feasibleness.

The two-path planning approaches extensively studied and experimented with are RRT and A* path planning algorithms. Random Rapid Trees (RRT) is a sampling-based path planning technique that randomly explores an unknown environment while traversing. It initially generates random points which are then connected based on the nearest neighbor nodes, considering the node is placed outside the obstacle boundary. The algorithm ends when a node is generated within the goal region, or a limit is hit so that an effective navigation route can be established. The connecting of nodes with neighbors and random node formations can be visualized in Figure 8.

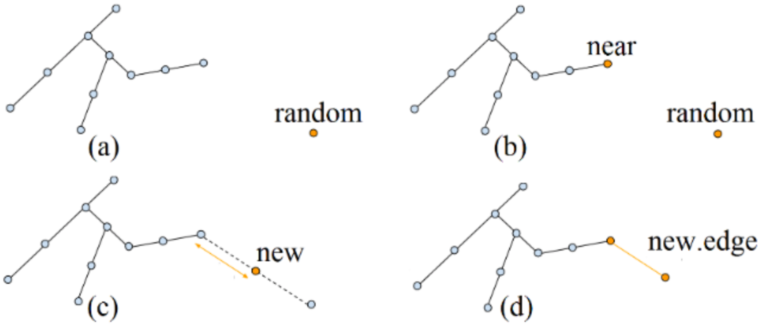


Figure 8 RRT path planning exploration

A* path planning is heuristic-based path planning which uses a node-based approach to device a safe path from start to goal position. It calculates the value of the heuristic function at each node in the work area and involves examining many nearby nodes to get the best solution with no possibility of colliding.

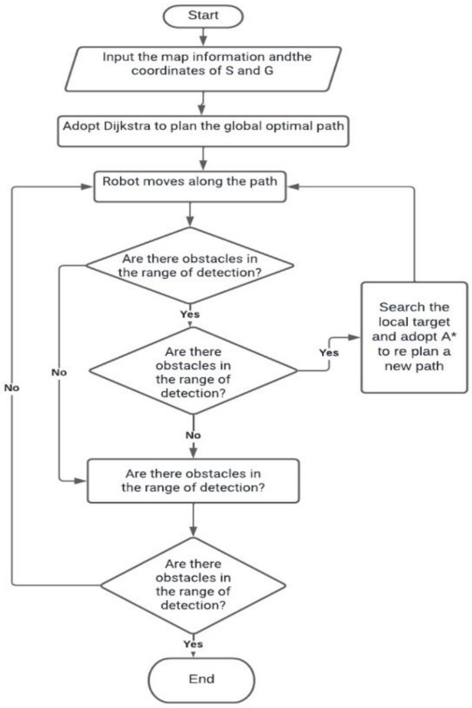

Start
Input the map information andthe coordinates of S and G
Adopt Dijkstra to plan the global optimal path
Robot moves along the path
Are there obstacles in the range of detection?
Yes
Are there obstacles in the range of detection?
Yes
Search the local target and adopt A* to re plan a new path
No
No
No
Are there obstacles in the range of detection?
Are there obstacles in the range of detection?
Yes
End

Figure 9 Workflow of A* path planning

Figure 9 shows the algorithmic workflow of the A* path planning algorithm. First, the occupancy grid consisting of all the positions of the obstacles with respect to the global frame is taken as the input. Several nodes are initialized on the global map to find a suitable path for navigation. The algorithm initially explores all possibilities through all the selected nodes and the best-optimized path is chosen for traversal. While traversing, if a new obstacle is detected, then the global path planner re-searches for a new global optimal path, thus avoiding the observed obstacle.

4.6.1.3 Control System

There are various control system architectures used for the efficient navigation of robotic systems. Here in the proposed E2ES, we will be using MPC and PID-based control systems on different aspects of the architecture.

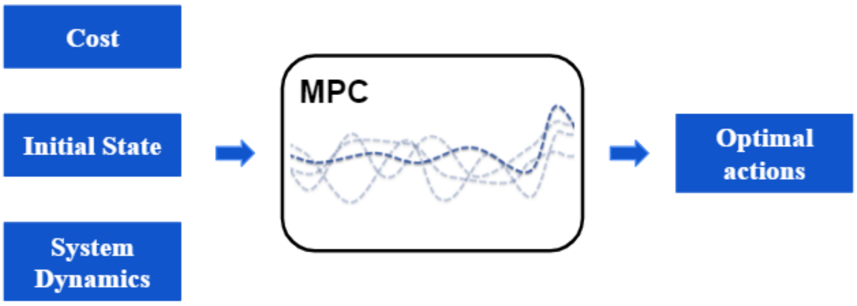


Figure 10 Idea behind MPC control block

According to Newtonian mechanics, once we modeled our system behavior, we should be able to predict the behavior under a set of constraints over some time. But due to the disturbances and uncertainties in the equipment and measurement, the prediction can be unpredictable thus making it non-linear. To create a high-fidelity controller, kinematic and dynamic modeling is done and the state variables of the system are found to predict the behavior of the system over time.

The Model Predictive Controller (MPC) is a high-fidelity controller which uses system constraints and disturbances while using the predictive nature of the system. MPC is very simple to tune while the work case can be performance-centric through processing multi-variables at the same time. MPC is considered to be nonlinear control that simply works on predicting the future states and errors.

It is widely used in the majority of fields where multi-variable control is required for the predefined application. The main drawback of MPC being it computationally intensive, but advancement in parallel computing has solved the issue in several work cases including robotics.

Quadcopters are a high-fidelity system with 6 DOF that needs a multivariable control over the roll, pitch, yaw, thrust, rate of change velocity, rate of change of angular velocity to create an agile control of the system. With control input u the defined kinematics and dynamics predict the 12 state variables of the system. For moving from the point, A to B in an agile manner, the controller uses a future state value to predict a suitable control command thereby effectively reaching the goal state. This is successfully done considering the disturbances and the uncertainties related to it.

PID is a closed-loop control system that stands for Proportional Integral Derivative controller. It is widely used in industries with a wide range of efficient applications.

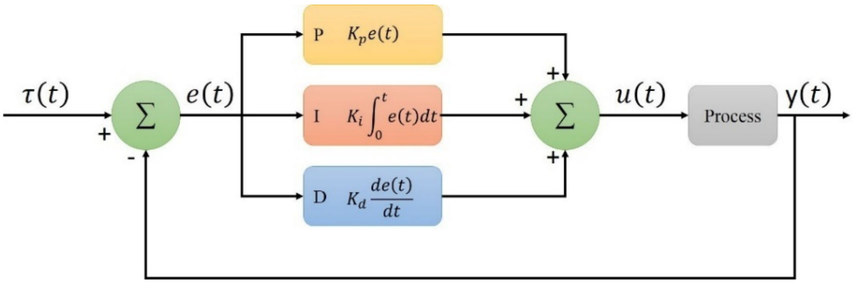


Figure 11 Idea behind PID control block

Proportional Controller (P only)- Stabilizes unstable process. It helps in reducing the steady-state error in the operation. But this controller can’t always eliminate the steady-state error. Thus, we will check along with the Derivative controller next.

$$output = k * error \tag{7}$$

Proportional Derivative Controller (PD only)- Increases the net stability of the operation. The derivative part of the control system helps in predicting the future errors of the systems based on their response. Thus, it helps in controlling the sudden shift of the operation.

$$output = k_p * error + k_i * (error - prev\_error) \tag{8}$$

Proportional-Integral-Derivative Controller (PID)- Thus, this is a very dynamic system equipped with zero state error, fast response, no oscillations, and high stability. Here in equation 9, the I_term is incremented for every estimated error value in the system.

$$output = k_p * error + k_i * (error - prev_{error}) + \ k_d * (I_{term} + error) \tag{9}$$

PID has several advantages when implemented for advanced control. It is very simple to implement since mathematics is very simple and can be implanted in almost all programming languages since it doesn't need a powerful maths solver to derive the solution. PID controller is highly effective with limited computational resources since it can also be implemented on an Arduino-based microcontroller and tuning of the control system can be done with a simple trial and error method. Model-based controllers only have an integral type of action to recover from unmeasured disturbances, but PID additionally offers proportional and derivative actions that operate instantly on an unknown disturbance. Refer to APPENDIX G to know more about different path planning techniques in detail.

4.6.2 Learning-based Approach

As the use of robotic systems evolves, so does the demand for robotic solutions to a wide range of challenges, particularly when the application demands robots to navigate in complicated, unknown settings. Humans have solved this problem by comprehending teacher comments. The teacher instructs the student by providing feedback in a variety of methods, such as rewarding proper behavior or modeling the desired behavior for the student to imitate. Learning-based techniques are the name given to this approach when it is applied to robots. Imitation learning and reinforcement learning are the two most frequent learning-based strategies. We'll look at the above-mentioned learning-based strategies in this section.

4.6.2.1 Imitation Learning

Imitation learning [50] or IL is a type of machine learning that uses a group of demonstrations to learn a control policy. Demonstrations are a collection of trajectories that depict an action leading to a specific state. The key principle of IL is learning the action and predicting from a state value. As a result, autonomous systems use IL to learn tasks in the hopes of imitating expert activities. In Illinois, the demonstrator specifies the various modes of learning. For learning, the traditional approach includes both action and state. In a navigation task, the action correlates to the PWM, while the state denotes the pose or odometry. Imitation-from-Observation, or IfO, is a type of IL in which just state is employed. Some basic representational units in IL include τ_e which represents the trajectory of the demonstration, s_t and a_t representing the state and action of the demonstration or agent at a particular time t. The objective of IL is to find the learning policies (π) from the demonstrations by the expert. Also, E and L correspond to the expert and the learner.

$$\tau_e = \{(s_t, a_t)\} \tag{7}$$

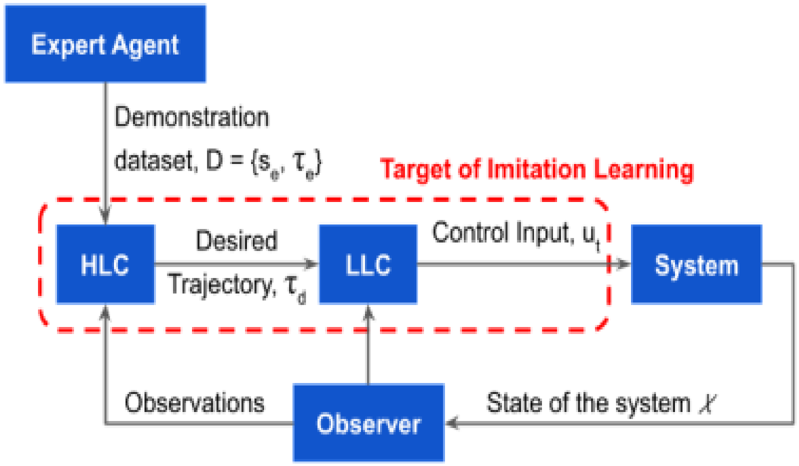


Figure 12 Workflow of imitation learning approach

As shown in Figure 12, first the expert navigates through the environment based on desired behaviors thereby generating a dataset D. The dataset D, conventionally consisting of state and action is then fed into a High-Level Controller or HLC which generates the most suitable trajectory (τ_e). The Low-Level Controller or LLC then tries to follow τ_e which will generate a new set of s_t which tries to imitate the s_t. The policy in turn will output a control input u_t for navigation.

4.6.2.2 Reinforcement Learning

Reinforcement learning or RL is a Machine Learning (ML) technique in which agents learn by completing a task in an interactive environment and receiving feedback. In general, agents learn from the feedback of their actions through trial and error. In Figure 13, the agent uses the current state data to map an action a_k, which is then used to produce a suitable reward r_{k+1} and state s_{k+1}. This is because the training environment has a Markovian property, which means that the agent's future state and reward are exclusively determined by his or her current state. A control policy is used by agents to obtain their intended states. Training labels in Deep RL or DRL are dynamic, unlike supervised learning, and hence the process cannot be done offline, necessitating constant interaction with the environment.

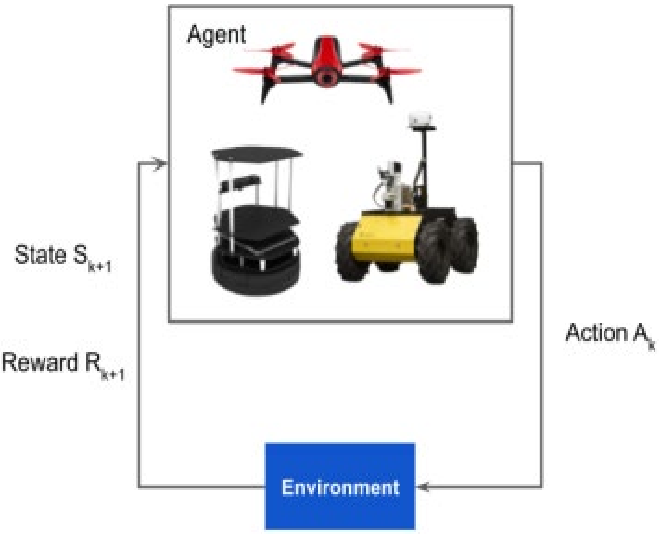


Figure 13 Workflow of reinforcement learning approach

The first step in the RL process is to set up an appropriate environment that is Markovian in nature. Following the successful creation of the training environment, the appropriate reward and policy must be developed, followed by the selection of the right training method for the specific task. The policy is then trained for multiple episodes until it maximizes the defined reward. Following successful completion of training, the policy can be validated and deployed to an appropriate agent.

4.7 Inspection

4.7.1 Convolutional Neural Network

In this subsection, the basic layers constituting the proposed novel architecture are explained in relevance to the convolutional neural network.

4.7.1.1 Convolution Layer

The objective of the convolutional layer is to capture features of both high- and low-level features including edges, corners, and gradient orientation from the input image and thus understanding the images and visualizing them just like humans [51]. So, the convolutional layer primarily does one of two things: it either reduces the spatial size of the image map by simply using kernels called Valid Padding (no padding) or it maintains the spatial distribution of the image map by augmenting the original image by padding, which is commonly referred to as Same Padding. The first convolutional layer extracts low-level features such as edges, while subsequent layers extract high-level features, resulting in a visualization similar to ours, which extracts different features such as textures, eventually learning various feature depictions of the targeted class embedded in the image tensor.

4.7.1.2 ReLU Activation

ReLU or Rectified Linear Unit is a piecewise hidden unit/ activation function that outputs the input directly if it is positive or else zero (when negative). This linear unit results in more accuracy and ease of training attesting to being a robust activator. Some of the advantages of using the ReLU activation function were explained by Albawi et al [52].

4.7.1.3 Global Pooling

Global Average Pooling (GAP) reduces the spatial dimensions of a 3D tensor ultimately dropping the number of parameters used in the model. It also helps in curtailing the overfitting observed. The GAP layers work as follows: the image feature map will be obtained and the initial size that's let's assume to be $L\ x\ B\ x\ W$ will be instantaneously reduced to $L\ x\ 1\ x\ 1$ [53]. The condensed $B\ x\ W$ feature map will be averaged, thus resulting in a simple number.

4.7.1.4 Dropout function

Dropout [54] refers to the neural network's arbitrary-selected neurons per layer dropping out. Dropout lengthens the training period while preventing data overfitting. Settings within correct thresholds also improve accuracy and steadily reduces loss.

4.7.1.5 Batch Normalization

For a consistent and computationally cost-effective training process, the input image set must be standardized. The Batch Normalization layer [55] in the system is responsible for this. This layer divides the input into mini-batches and homogenizes it.

4.7.2 Squeeze-Excitation Network

The Squeeze excitation networks were used to improve the performance of residual networks. SENets improve performance without raising computing costs by familiarising a building block perception for CNN. By introducing supplementary parameters, they directly influence the network and its layers, adaptively modifying the weights of every feature map in the channels of the convolutional block. The use of weights dramatically improves the sensitivity to advantageous map features that will be exploited in subsequent transformations. Recalibration filter replies and global responses are also supplied before the network is sent into the next transformation. The "Squeeze" and "Excitation" phases are effective in accomplishing this.

4.7.2.1 Squeeze

The basic idea of the *squeeze* phase is to reduce the spatial distribution of the feature maps using various feature descriptors. Major methods used for this reduction in CNN are Global average pooling and Global Max pooling (GAP) methods. GMP preserves the most active pixels but has a lot of noise and is independent of the neighboring pixels and GAP results in smooth averaging pixelate and don’t preserve pixel information. GAP descriptor was selected for the squeeze phase aiming to get smoother pixel information. Thus, the global information of each channel is attained by exploiting the GAP layers resulting in the reduction of the $L\ x\ B\ x\ W$ of any image map into $L\ x\ 1\ x\ 1$. Thereby, the extraction of this global information and reduction of the spatial dimensions/ distribution of the channels is the major exertion of the “squeeze phase”.

$$z_c = F_{sq}(u_c) = \frac{1}{B \times W} \sum_{I=1}^{B} \sum_{J=1}^{W} u_c(i,j) \tag{10}$$

4.7.2.2 Excitation

In the *excitation* phase, the channel value from the “squeeze” phase will be assigned adaptive scaling weights to the respective channels. σ in the below equation represents the sigmoid operator, δ represents the ReLU activation operator, $W1$ and $W2$ represent the two fully connected layers and z represents the output of the “squeeze” phase.

$$s = F_{ex}(z, W) = \sigma\big(g(z, W)\big) = \sigma\big(W_2 \delta(W_1 z)\big) \tag{11}$$

Then the $L\ x\ 1\ x\ 1$ tensor will be passed through a sigmoid activation subsequently assigning appropriate weights for respective channels of the feature map learned from the MLP of the excitation phase. The SE-Resnet will thereby, as mentioned above, add some additional layers along with its parameter increasing less than 1% of the computing cost of the system. SoftMax layers enforce importance on only one of the channels. Thus, sigmoid was chosen as it accentuates multiple channels and the output sigmoid function is represented as:

$$x_c = F_{scale}(u_c, s_c) = s_c u_c \tag{12}$$

4.7.3 Capsule Networks

4.7.3.1 Capsules

Sabour et al. [56] proposed Capsule Network or CapsNet to address the typical problems with CNN's (rotational invariance and failure to capture spatial hierarchical information) by rapidly retrieving the most critical feature information and spatial relationships. To avoid losing critical functionality, dynamic routing was used instead of pooling procedures. Also, instead of neurons, capsules were employed to efficiently transfer the spatial connection from one layer to the next. However, greater feature information in complicated datasets causes CapsNet to fail. As a result, strategically adding additional convolutional layers to the CapsNet is one solution to this problem.

If the likelihood of the feature and the feature information is sent through each neuron, appropriate activation can be delivered. As a result, these neurons are known as capsules, and instead of single feature information, they produce a vector (called an activity vector). We can easily extrapolate different feature variants in a capsule using the likelihood of a feature, decreasing the training data. The product of the pose matrix (M_i) represented by a gaussian distribution with the transformation (W_{ij}) is computed to calculate the vote (v_{ij}).

$$v_{ij} = M_i \, x \, W_{ij} \tag{13}$$

In equation 13, i corresponds to the parent capsule and j to the current capsule. The transformation is learned through back-propagation and a cost function.

4.7.3.2 Routing

Transferring information from one layer to another is called Routing. ReLU is used as the routing mechanism in fully convolutional neural networks.

$$y_i = ReLU\left(\sum_j W_{ji} \cdot x_j + b_i\right) \tag{14}$$

To transform small vector magnitudes to zero and large vector magnitudes to unit vectors, a squashing function is utilized instead of ReLU. EM Routing is the process of grouping capsules using the EM clustering algorithm. In terms of votes, parent capsules are anticipated in the pose matrix. Because it is based on proximity, the transformation matrix remains constant even when the views vary. The runtime link between the parent and other capsules is estimated using assignment probabilities (r_{ij}). Equation 15 shows the cost of all low-level capsules.

$$J_j^h = \left(\left(\ln \sigma_j^h\right) + k\right) \sum r_{ij} \tag{15}$$

Activation (A_j) in a particular capsule can be estimated using the inverse temperature parameter (λ) and parameters iteratively estimated using EM Routing by equation 16.

$$A_j = sigmoid\left(\lambda\left(b_j - \sum_h J_j^h\right)\right) \tag{16}$$

The data points fit into a gaussian model and E and M steps are triggered alternatively iterating for t steps. The Gaussian model is updated using the M step based on the r_{ij} initialized (distributed uniformly) which will then be reshaped to form the pose matrix of the parent capsule. The r_{ij} is recomputed using the E-step for every data point.

CHAPTER 5

OUR METHODOLOGY

5.1 Sensor Fusion

Odometry-based control has been established for determining the most reliable fused data results. The odometry model is defined as:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \underbrace{\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \delta_{\mathrm{trans}} \cos\left(\theta + \delta_{\mathrm{rot}\,1}\right) \\ \delta_{\mathrm{trans}} \sin\left(\theta + \delta_{r_{t_1}}\right) \\ \delta_{\mathrm{rot}\,1} + \delta_{rot_2} \end{pmatrix}}_{g(u_t, x_{t-1})} + \mathcal{N}(0, R_t) \tag{17}$$

The motion model has 3 components for finding the new $x, y,$ and θ. All the equations are derived from geometry and the function N is associated with the noise in the motion model. So, this function is a nonlinear model and has to be linearized. The Jacobian of the motion mode with respect to the state is found to be:

$$G_t = \frac{\delta g(u_t, \mu_{t-1})}{\delta x_{t-1}} \tag{18}$$

The result of the applied formula is:

$$G_t = \begin{pmatrix} 1 & 0 & -\delta_{\mathrm{trans}} \sin\left(\theta + \delta_{rot_1}\right) \\ 0 & 1 & \delta_{\mathrm{trans}} \cos\left(\theta + \delta_{\mathrm{rot}\,1}\right) \\ 0 & 0 & 1 \end{pmatrix} \tag{19}$$

Then we have to find the Jacobian of the matrix with respect to the control that is U.

$$V_t = \frac{\delta g(u_t, \mu_{t-1})}{\delta u_t} \tag{20}$$

The result of the applied formula is:

$$V_t = \begin{pmatrix} -\delta_{\mathrm{trans}} \sin\left(\theta + \delta_{\mathrm{rot}\,1}\right) & \cos\left(\theta + \delta_{\mathrm{rot}\,1}\right) & 0 \\ \delta_{\mathrm{trans}} \cos\left(\theta + \delta_{\mathrm{rot}\,1}\right) & \sin\left(\theta + \delta_{\mathrm{rot}\,1}\right) & 0 \\ 1 & 0 & 1 \end{pmatrix} \tag{21}$$

Next is the observation or measurement model. In our case, it is a range bearing model which has 2 terms (range, yaw) with respect to the sensor. So, the equation is as follows,

$$z_t^i = \begin{pmatrix} r_t^i \\ \phi_t^i \end{pmatrix} = \underbrace{\begin{pmatrix} \sqrt{\left(m_{j,x} - x\right)^2 + \left(m_{j,y} - y\right)^2} \\ \operatorname{atan2}\left(m_{j,y} - y, m_{j,x} - x\right) - \theta \end{pmatrix}}_{h(x_t, m)} + \mathcal{N}(0, Q_t) \tag{22}$$

The function is nonlinear and the function N is the noise associated with the measurement model. The measurement model is then linearized around a mean position u_t. Jacobian of the measurement model with respect to the state vector is found using the formula,

$$H_t^i = \frac{\delta h(\bar{\mu}_t, m)}{\delta x_t} \tag{23}$$

The result of the applied formula is:

$$H_t^i = \begin{pmatrix} -\frac{m_{j,x} - \mu_{t,x}}{\sqrt{q}} & -\frac{m_{j,y} - \mu_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y} - \mu_{t,y}}{q} & -\frac{m_{j,x} - \mu_{t,x}}{q} & -1 \end{pmatrix} \tag{24}$$

$$\text{where } q = \left(m_{j,x} - \bar{\mu}_{t,x}\right)^2 + \left(m_{j,y} - \bar{\mu}_{t,y}\right)^2$$

Hence, we linearized the measurement state equation, and also, we derived all the necessary parameters to apply the Extended Kalman Filter. The results of the Extended Kalman Filter algorithm are explained in the right order in the following.

Algorithm EKF Localization $(\boldsymbol{\mu_{t-1}}, \boldsymbol{\Sigma_{t-1}}, \boldsymbol{u_t}, \boldsymbol{z_t}, \boldsymbol{c_t}, \boldsymbol{m})$:

$$\theta = \mu_{\ell-1}, \theta \tag{25}$$

G_t and V_t from equations (19) and (20) is inputted here. M_t represents the uncertainties and noises produced when a movement occurs inside the system.

$$M_t = \begin{pmatrix} \alpha_1 \delta_{\mathrm{r0}t_1}^2 + \alpha_2 \delta_{\text{trans}}^2 & 0 & 0 \\ 0 & \alpha_3 \delta_{\text{trans}}^2 + \alpha_4\left(\delta_{\text{rot }2}^2 + \delta_{\text{rot }2}^2\right) & 0 \\ 0 & 0 & \alpha_1 \delta_{\text{rot }1}^2 + \alpha_2 \delta_{\text{trans}}^2 \end{pmatrix} \tag{26}$$

Equation X indicates the propagation step, where the mean estimated measurement is calculated, $\mu_t - 1$ is the mean estimate at time $t - 1$ and the matrix represents the model equation of the g function.

$$\mu_t = \mu_{t-1} + \begin{pmatrix} \delta_{\text{trans}} \cos\left(\theta + \delta_{\text{rot}_1}\right) \\ \delta_{\text{trans}} \sin\left(\theta + \delta_{\text{rot}_1}\right) \\ \delta_{\text{rot}_1} + \delta_{\text{rot}_2} \end{pmatrix} \tag{27}$$

Then, the covariance matrix which depends on the covariance at $t-1$ step and the calculated Jacobians is updated. The propagation of the uncertainties is also transferred from $t-1$ to t because if the robot is more uncertain at $t-1$ step, then it is also more uncertain at t.

$$\Sigma_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T \tag{28}$$

Then comes the measurement step of the features present in the robot environment. If there are n features/ landmarks in the environment, then there are n update iterations.

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix} \tag{29}$$

A loop is initialized here. For all observed values,

$$z_t^i = \left(r_t^i, \phi_t^i\right)^T do \tag{30}$$

$$j = c_t^i \tag{31}$$

$$q = \left(m_{j,x} - \bar{\mu}_{t,x}\right)^2 + \left(m_{j,y} - \bar{\mu}_{t,y}\right)^2 \tag{32}$$

Equation (33) calculates the expected measurement of the robot at time t which is calculated by the measurement state model.

$$\hat{z}_t^i = \left(\operatorname{atan} 2\left(m_{j,y} - \bar{\mu}_{t,y}, m_{j,x} - \bar{\mu}_{t,x}\right) - \bar{\mu}_{t,\theta}\right) \tag{33}$$

Then comes the correction step, wherein first we calculate the Kalman gain of the system to find the new estimated mean and the new estimated covariance based on the weights of the uncertainties produced by the motion model and the measurement model. H_t^i from equation (23) is inputted here.

$$S_t^i = H_t^i \bar{\Sigma}_t H_t^{iT} + Q_t \tag{34}$$

$$K_t^i = \bar{\Sigma}_t \left[H_t^i\right]^T \left[S_t^i\right]^{-1} \tag{35}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i\left(z_t - \hat{z}_t^i\right) \tag{36}$$

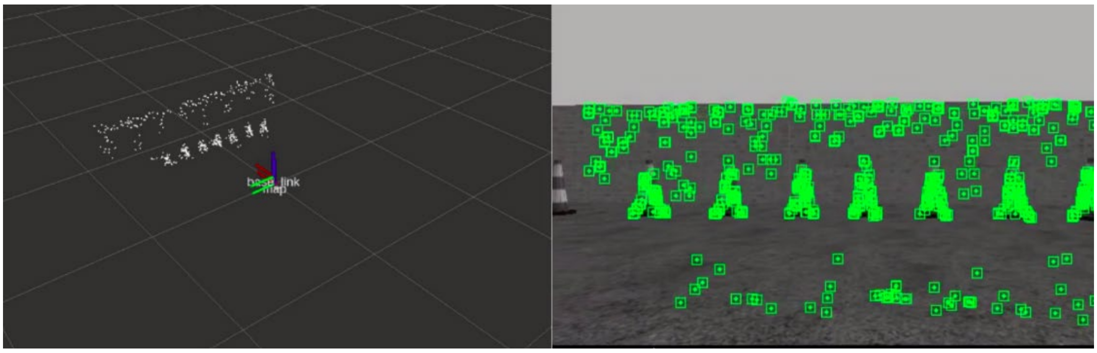
$$\bar{\Sigma}_t = \left(I - K_t^i H_t^i\right)\bar{\Sigma}_t \tag{37}$$

Then, μ_t and Σ_t are returned as output. All the steps are recursive and iterated to get the best prediction of the localization for the robot based on the sensor measurement.

5.2 Localization

We require a very computationally efficient system for localization in a flying system since all of the subsystems must be handled onboard, and complete all of the operations with limited computation resources. ORB SLAM is the best option for the specified work case since it can work in low-light and high-dynamic-range environments. The ORB feature descriptor's efficiency allows for improved picture matching and faster calculation. As a result, it is the greatest solution for quadcopter localization in a densely crowded environment.

Thus, ORB2 SLAM was used to localize the robot in the environment, which uses ORB features to analyze the environment and match features between frames to triangulate and localize itself. ORB2 SLAM produces sparse mapping and odometry, which are displayed using the RVIZ tool in Figure 14. The ORB features spread in the environment are represented by green dots contained by squares, and the sparse map associated with the features point cloud is represented by the sparse map.



(a) (b)

Figure 14 Implementation of ORB-based Localization

Figure 14 represents the simulation of a quadcopter in a gazebo simulator to localize itself in the global environment using the front-facing camera. The green dots thus represent the features as explained in the previous paragraph and it is clear they are edges, corners with texture variations, thus properties of identifying ORB features are established.

The ORB features extracted from the visual sensor are fused using sensor fusion with inertial sensors, thereby significantly increasing the reliability of the localization data. More details on how sensor fusion has been done in this work are explained in Section 5.1.

5.3 Obstacle Avoidance

In this section, we'll go through the proposed architectures for the learning-based and learning-free approaches, as well as how to most effectively avoid obstacles efficiently. In addition, the data obtained while executing the learning-free strategy was used to learn a policy for the learning-based approach. Image frames, as well as the succeeding pose and principal yaw angle, are included in the sent data for learning the policy.

5.3.1 Learning-free Approach

In this work, a simple yet efficient learning-free obstacle avoidance system is proposed for autonomous navigation. Perception, path planning, and control modules are included in the algorithm, as they are in any other technique. Figure 15 illustrates the design of the proposed system. The main advantages of this system are that it has adaptive velocity control and that it only uses one sensor for avoidance. Every module of the proposed system is thoroughly detailed in the remainder of this section.

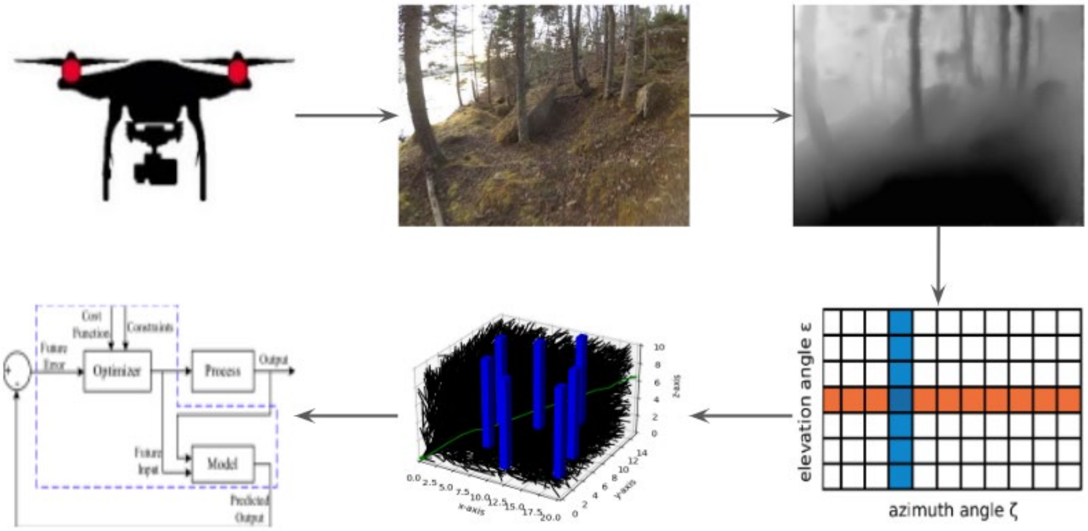
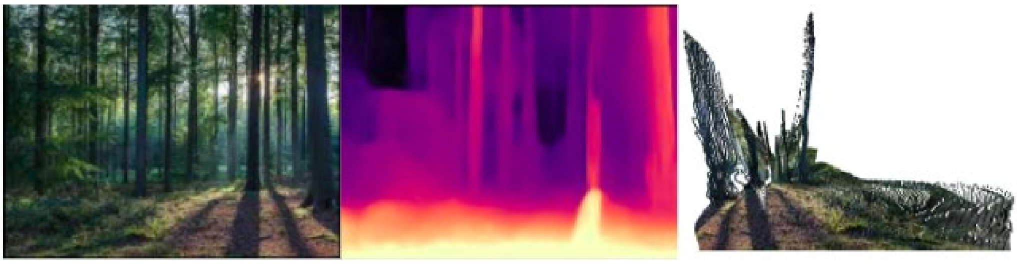


Figure 15 Architecture of learning-free approach

Figure 16 shows the perception module built from scratch for learning-free obstacle avoidance. Initially as explained earlier, a monocular depth estimation approach called MonoDepth2 [57] was used to estimate the depth from monocular images learned in an unsupervised manner. Then, using simple extrapolation and the ORB localization data, we found the scale factor with respect to the world. This data was then used to calculate the point cloud representation of the image.



(a) (b) (c)

Figure 16 Proposed perception module

Figure 17 shows the predicted point clouds projected into a 2D grid map with x and z values (to visualize from the top). The point clouds are then encircled by random borders. Then the boundaries are constrained based on overlap features and proximity. After the boundaries are created, a final experimental grid map with the perceived obstacle borders is created. This information is subsequently fed as input for the other navigation modules (planning and control).

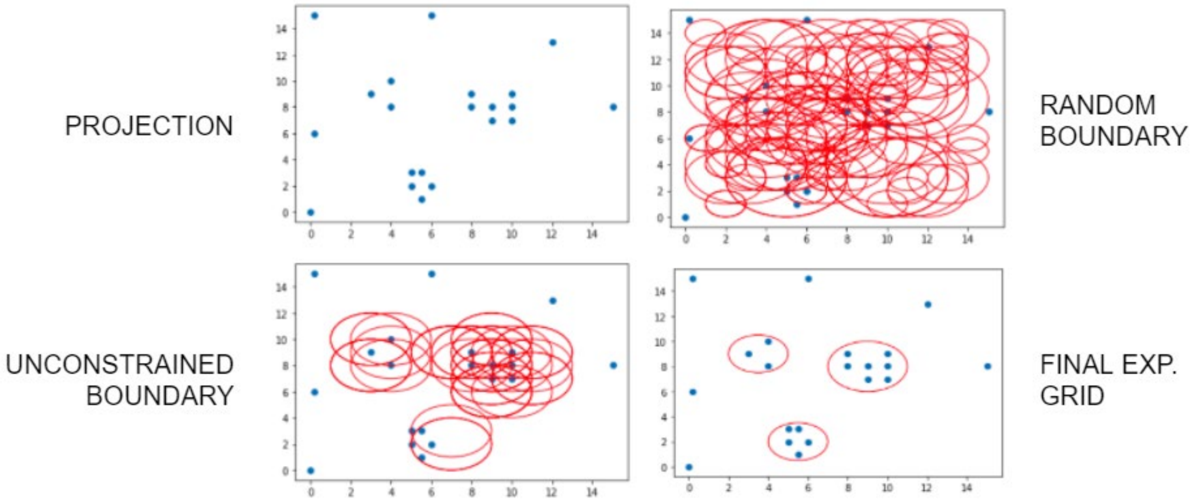


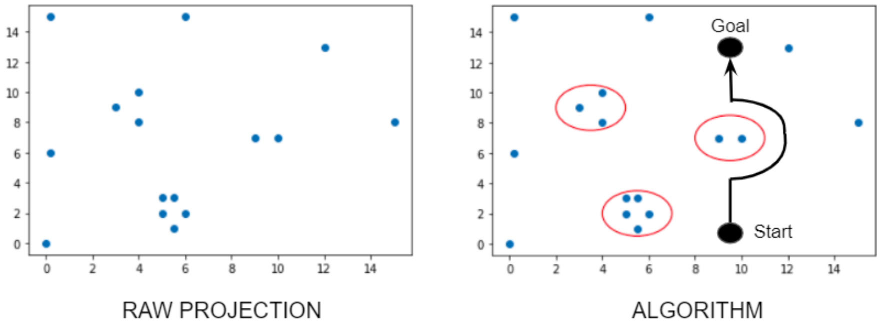
Figure 17 Selecting the obstacle boundary

The RRT exploration is performed in the resulting final experimental grid to discover the most efficient path to the objective coordinate. Instead of inserting RRT nodes at random, the cost function given below in Equation (38) is used to select them. This function act as a simple heuristic for path planning. Because the placement of nodes is not random, the system's computing cost is greatly decreased.

$$J = \frac{[\, p_{i+prev_i} * 4 + \left(\frac{1}{d_{cur-des}} + \frac{1}{d_{des-goa;}}\right) * 2\,]}{6} \tag{38}$$

In equation (38), p represents the pixel intensity, d_{cur} represents the current pose, d_{des} represents the destination distance from the current pose and d_{goal} represents the distance between the goal and the current position.

Once, the optimized control points to reach the goal are found using the RRT path planning algorithm, the MPC control algorithm is used for navigation. The mathematical modeling designed for using MPC for our system is detailed in the following pages.



RAW PROJECTION

ALGORITHM

Figure 18 Navigation strategy formation

With the kinematics and the dynamics model for a quadcopter, we will be able to estimate the state variables of the system. Equations (39) – (55) represent the state variables of an agile quadcopter and are derived using the configuration of the quadcopter and the global reference frame of the IMU.

$$\ddot{x} = \frac{-1}{m}\left[k_{t_x}\dot{x} + u_1(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)\right] \tag{39}$$

$$\ddot{y} = \frac{-1}{m}\left[k_{t_y}\dot{y} + u_1(\sin\phi\cos\psi - \cos\phi\sin\psi\sin\theta)\right] \tag{40}$$

$$\ddot{z} = \frac{-1}{m}\left[k_{t_z}\dot{z} - mg + u_1\cos\phi\cos\theta\right] \tag{41}$$

$$\dot{p} = \frac{-1}{I_x}\left[k_{r_x}p - lu_2 - I_yqr + I_zqr + I_rq\omega_r\right] \tag{42}$$

$$\dot{q} = \frac{-1}{l_y}\left[-k_{r_y}q + lu_3 - I_xpr + I_zpr + I_rp\omega_r\right] \tag{43}$$

$$\dot{r} = \frac{-1}{I_z}\left[u_4 - k_{r_z}r + I_xpq - I_ypq\right] \tag{44}$$

And the relationship between the Euler angles and angular velocity is expressed in the equations below

$$\phi = p + \mathrm{r}\cos\phi\tan\theta + \mathrm{q}\sin\phi\tan\theta \tag{45}$$

$$\dot{\theta} = \mathrm{q}\cos\phi - \mathrm{r}\sin\phi \tag{46}$$

$$\psi = r\frac{\cos\phi}{\tan\theta} + q\frac{\sin\phi}{\cos\theta} \tag{46}$$

The dynamic model of the quadcopter can be defined from four control inputs ($u = [u_1 u_2 u_3 u_4]^T$) and twelve state variables. The quadcopter being a non-linear system, can be modeled as shown in Equation (47).

$$\dot{x} = f\big((x), u(t)\big) \tag{47}$$

When we linearize the dynamic model of the quadcopter under hovering conditions where the nominal states and control inputs are $\mathcal{X}_T$ and $\mathcal{u}_T$, we get

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k \tag{48}$$

$$\Delta y_k = C\Delta x_k + D\Delta u_k \tag{49}$$

$$\Delta x_k = x_k - x_T \tag{50}$$

$$\Delta u_k = u_k - u_T \tag{51}$$

Upon using the linearized model, we predict the state variables over the time horizon using a linear quadratic estimator to estimate the change in the system for a control input u.

$$\Delta x_{k+N} = A^N \Delta x_k + A^{N-1}B\Delta u_k + A^{N-2}B\Delta u_{k+1} + \cdots + AB\Delta u_{k+N-2} + B\Delta u_{k+N-1} \tag{52}$$

$$\Delta y_{k+N} = CA^N \Delta x_k + C(A^{N-1}B\Delta u_k + A^{N-2}B\Delta u_{k+1} + \cdots + AB\Delta u_{k+N-2} + B\Delta u_{k+N-1}) \tag{53}$$

The equation can be re-written into the matrix form as represented below with these equations. Now we designed an MPC-based control for a quadcopter to traverse in a safe and agile manner between points A to B using matrixes derived from equations (39) to (55).

$$\begin{pmatrix} \Delta x_k \\ \Delta x_{k+1} \\ \Delta x_{k+2} \\ \vdots \\ \Delta x_{k+N-1} \end{pmatrix} = \begin{pmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{N-1} \end{pmatrix} \Delta x_k + \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ B & 0 & \cdots & 0 & 0 \\ AB & B & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{N-2}B & A^{N-3}B & \cdots & B & 0 \end{pmatrix} \begin{pmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+N-1} \end{pmatrix} \tag{54}$$

$$\begin{pmatrix} \Delta y_k \\ \Delta y_{k+1} \\ \Delta y_{k+2} \\ \vdots \\ \Delta y_{k+N-1} \end{pmatrix} = \begin{pmatrix} C & 0 & 0 & \cdots & 0 \\ 0 & C & 0 & \cdots & 0 \\ 0 & 0 & C & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & C \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta x_{k+1} \\ \Delta x_{k+2} \\ \vdots \\ \Delta x_{k+N-1} \end{pmatrix} + \begin{pmatrix} D & 0 & 0 & \cdots & 0 \\ 0 & D & 0 & \cdots & 0 \\ 0 & 0 & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D \end{pmatrix} \begin{pmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+N-1} \end{pmatrix} \tag{55}$$

Thus, the complete navigation system using a learning-free approach has been explained along with the novelties and advantages of this approach.

5.3.2 Learning-based Approach

For the learning-based approach, we have used a type of imitation learning idea called Behavioral Cloning (BC). The idea behind BC is very similar to supervised learning. A mapping strategy is approximated with a set of inputs to derive the desired outputs. That is, input images and poses are passed as input to map the respective yaw angles. Then, motor PWMs are estimated for efficient navigation in an unknown environment.

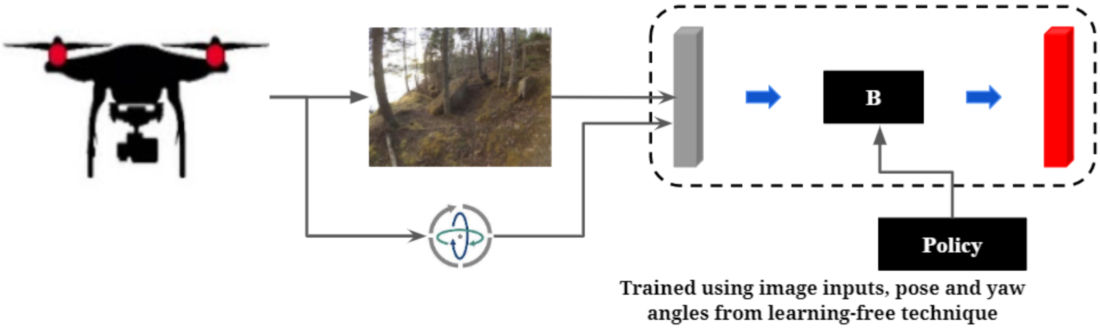

B
Policy
Trained using image inputs, pose and yaw angles from learning-free technique

Figure 19 Architecture of learning-based approach

Firstly, the dataset is gathered to train a policy that can map images and pose data to the appropriate yaw angle to avoid any collisions. This dataset gathering is carried out using a learning-free method. The data was captured and stored locally throughout the flight, and then passed to the neural network to find a suitable policy. This policy will then be applied in real-time to determine the UAV's best yaw angle.

The calculated yaw angle is subsequently provided to the control system block, which calculates the appropriate motor PWMs for the desired yaw angle successfully. The neural network deployed here takes advantage of the squeeze-excitation operators in the inspection module, extracting the most important information from the frame and calculating the most important yaw direction for navigation.

Table 2 Training data specification

	Scale	Unit
Image Resolution	120 x 160	pixels
Sampling rate	20	fps
Channel	grayscale	-
Duration	150	minutes
Control	3 (Yaw Right, Left, Straight)	-

Table 2 shows the data specifications taken into account for training. Grayscale images were used to reduce the computation significantly. Also, yaw directions were estimated to traverse in the environment instead of finding the angle itself, which was less reliable comparatively.

Different baseline networks including Residual Network, Inception Network, Alex Network VGG-16, and so on were tested. SENets improved the performance of all these baseline networks. Best results were obtained when SENets were fused with Residual baseline. Thus, we will be using SE-ResNet for training the dataset obtained from the learning-free approach. The specifics of the SENets used are explained in the upcoming section.

5.4 Inspection

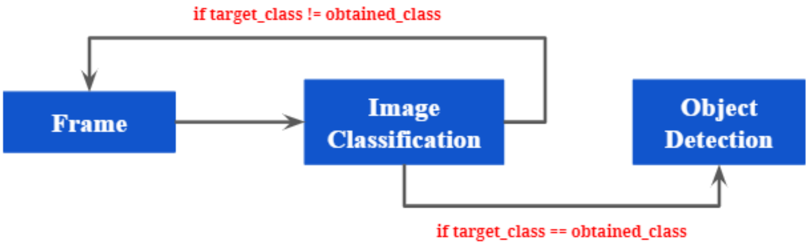

if target_class != obtained_class
Frame
Image Classification
Object Detection
if target_class == obtained_class

Figure 20 Proposed Sense-Switch-Act Mechanism

To maintain a balance between computation and accuracy, a novel sense-switch-act mechanism has been proposed for the inspection tasks as shown in Figure 20. At first, the desired target classes are parsed to the task-specific system. That is, if the task is forest fire, the target/ desired class will be fire. Then, the frame is sent to the image classification sub-module which identifies the different classes observed in the scene. Then, the classes obtained are cross verified if it matches the target classes. If yes, it is passed to the object detection sub-module which locates the exact coordinates of the target class. If there are no matches obtained, the image classification sub-module redirects to find the classes in the next frame, and the loop continues. More information on the sub-modules and the networks adapted are discussed in the following subsections in detail.

5.4.1 Image Classification

A novel network that multi-enhances the features, that is, both spatial and channel-wise information at each layer with deep convolutional layers (SENets) and CapsNet is proposed termed as ME-CapsNet (Multi-Enhanced Capsule Networks). In addition, improvements in the Squeeze phase of the originally proposed SENets by using Stochastic Spatial Sampling Pooling (S3P) are done to reduce feature information losses, computation overload and training time when compared to the originally used average pooling operation.

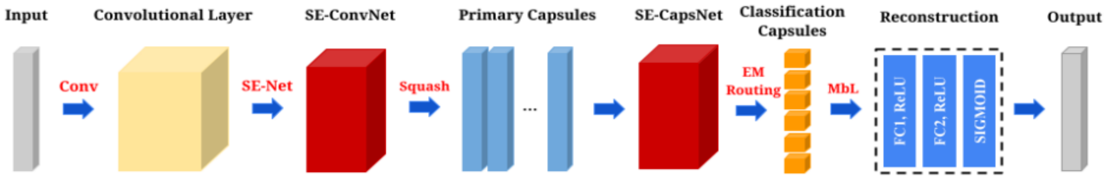


Figure 21 Architecture of the proposed image classification network

Our architecture predominantly has three blocks to multi-enhance the feature relationship of CNN: the SE Primary Capsule (SE-PC) block, SE Classification Capsule (SE-CC) block, and the Mask by Layer (MbL) block. That is, an input image in the form of tensor (U) is fed to our network, which first converts into a tensor upon transformation (U_{tr}). Then, U_{tr} is sent through the SE-PC block wherein Squeeze and Excitation operation is done which results in U_{tr}^{0} which has recalibrated weights based on channel dependence. U_{tr}^{0} is then passed through the primary capsule layer. The Squeeze and Excitation operation is done three times continuously to get the improved calibrated weights for the corresponding channels. The Squeeze operation is done using S3P [58]. Then the output of the primary layer of the SE-PC is fed as input to the SE-CC wherein features are highlighted using SENets first and then sent to the classification module. Then, we designed the MbL block inspired by the works of Huang et al. [59] for reconstructing the output capsules from the probability vectors obtained from the SE-CC block. Advantages of including the reconstruction block include regularization and the ability to regenerate data.

5.4.2 Object Detection

Object detection using YOLO detectors was used to locate the obstacles. Object detection was leveraged only to find the exact coordinate of the task-specified classes.

In other words, once a presence of a particular class is sensed by the image classification module (ME-CapsNet), we trigger the object detection sub-module to identify the exact location of the class in the global frame. For example, during forest fire operations, the UAV navigates and tries to identify the presence of fire using the ME-CapsNet sub-module. Once the presence of fire is sensed, the object detection sub-module locates the position of fire in the perceived environment. The reason to go with this switching mechanism is to reduce the computational cost of the system.

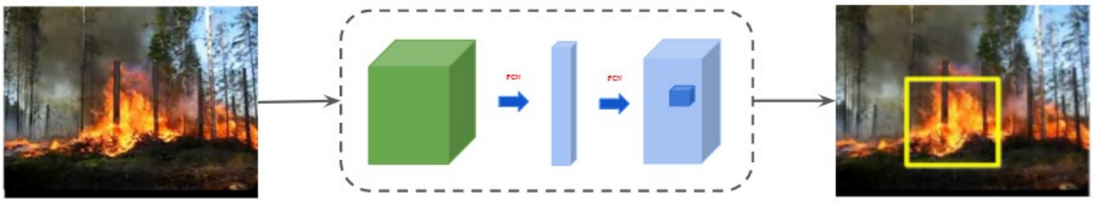


Figure 22 Architecture of the YOLO detection approach

CHAPTER 6

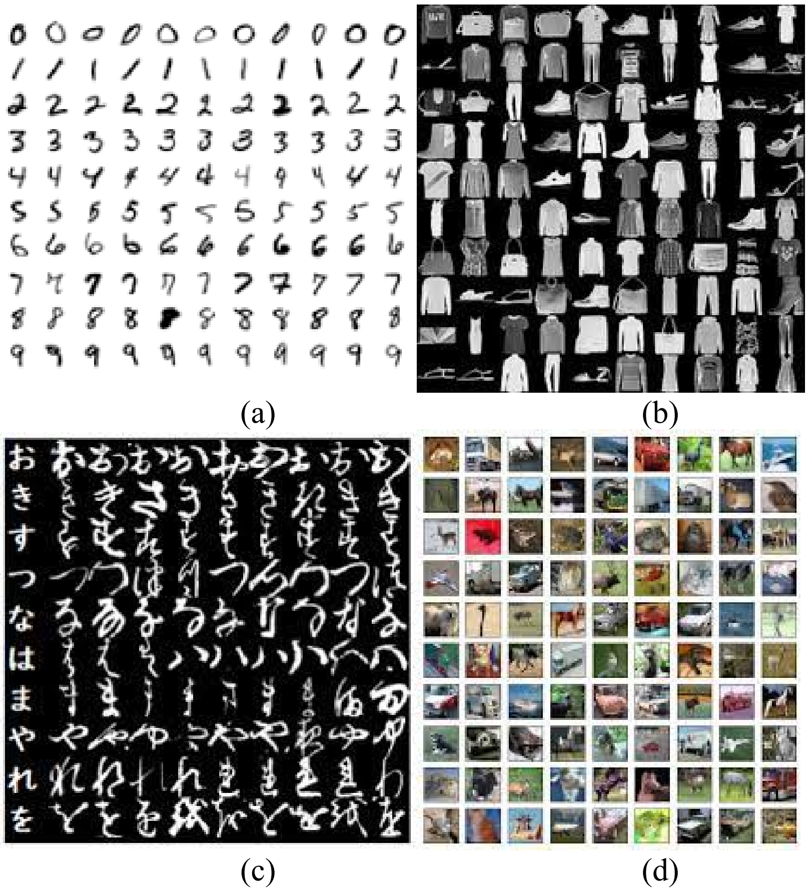
DATASET AND SIMULATORS

In this section, the datasets and simulators used to test the performance of the proposed E2ES as modules are detailed with reasons to go with these simulators and datasets.

6.1 Dataset

6.1.1 Inspection Dataset

MNIST [60] and FashionMNIST [61] both have 10 classes with 70,000 images of size 28 × 28 × 1. CIFAR10 has 10 classes with 60,000 images of size 32 × 32 × 3. KMNIST [62] dataset has 10 classes and is a drop-in replacement of MNIST and FashionMNIST, consisting of 70,000 images of size 28 × 28 × 1 just like those datasets. For MNIST, KMNIST, and FashionMNIST datasets 50,000 images were used for training and 10,000 images for testing. For the CIFAR10 [63] dataset, 60,000 images were used for training and 10,000 for testing.



(a) (b)

(c) (d)

Figure 23 Image classification datasets

For object detection, Cityscape [64] datasets were used to test the network's performance. The cityscape dataset consists of 30 classes collected from 50 different cities around a year in different periods. It has around 25,000 annotated images. Also, some videos of different tasks were used to examine the performance of the 'sense-switch-act' mechanism.

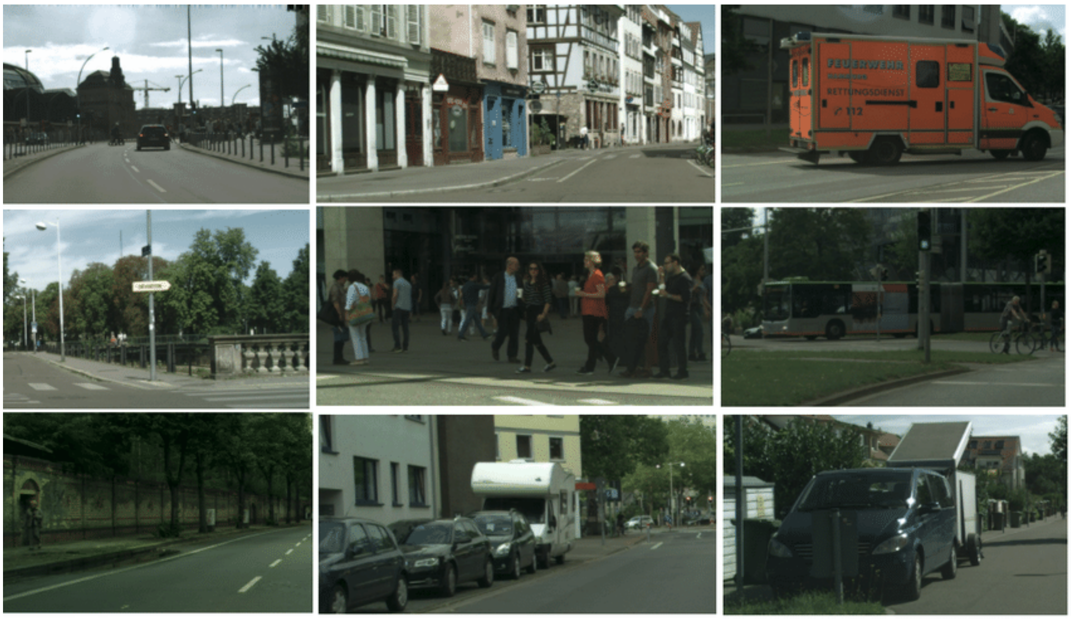


Figure 24 Cityscape dataset

6.1.2 Avoidance Dataset

We built a custom dataset for learning techniques called AIR-IL using the AIRSIM simulator. The dataset consists of images, odometry data, and the principal axes of the UAV. Thus, the aim of using this dataset is for the neural network to learn a suitable mapping technique to find the yaw angle (principal axis) from the image and odometry information of the UAV. The dataset consists of 100,000 frames per environment with its respective odometry and yaw information. The collection of data was done in three different environments, a cluttered abandoned park, an Africa Forest environment, and urban roads. The data collected in all three environments are also captured in different illuminations for testing the performance of the UAV system in varying conditions.

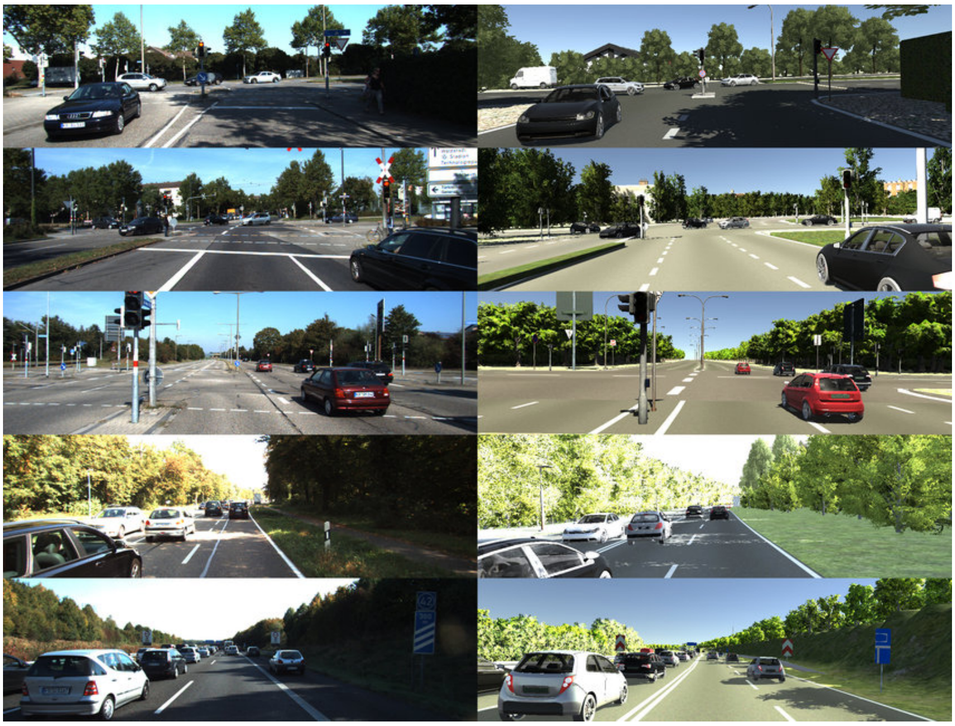


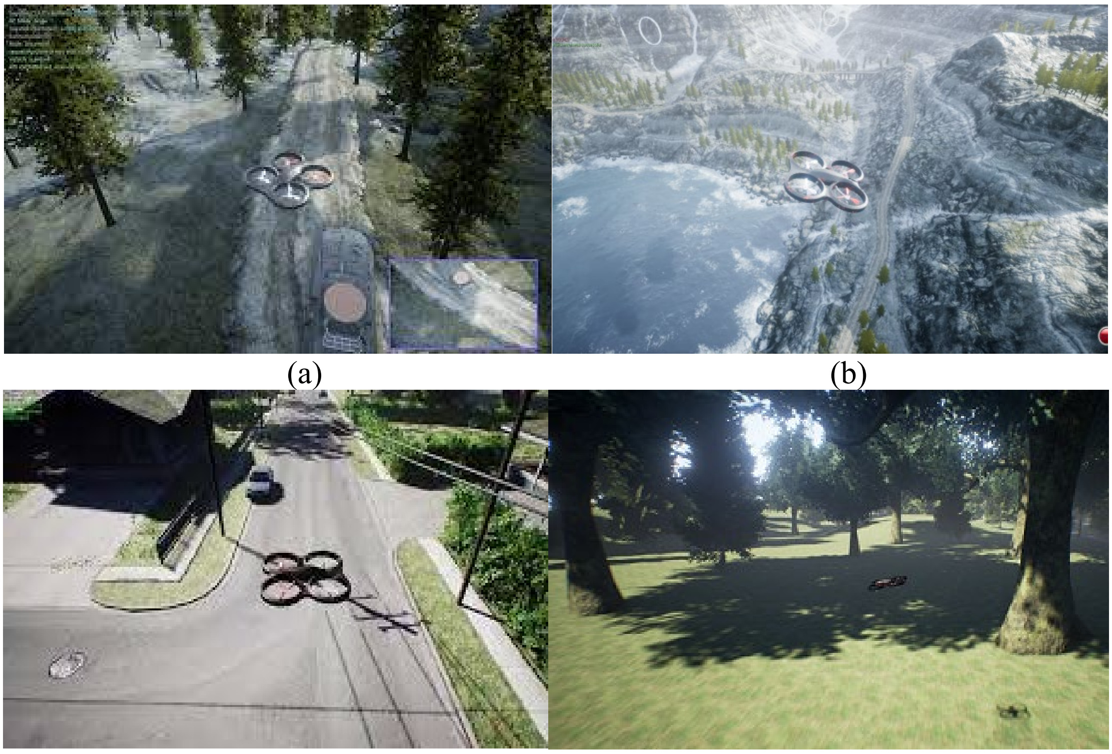
Figure 25 KITTI dataset

For testing the odometry results, we are using KITTI [65] benchmarked dataset. Some of the specs of the KITTI dataset include Grayscale frames, which consist of 22 stereo sequences (We have used the 1st sequence which has 4,500 stereo frames), 11 sequences with ground truth (for finding error/overlap ratios). We are experimenting with the 0th sequence of grayscale frames (right and left frames).

6.2. Simulator

6.2.1 AIRSIM

AIRSIM [66] was initially built using the Unreal Engine backbone, with recent releases focusing on Unity for experimentation. It is primarily built focusing on UAVs with add-ins for cars and more. It is widely used for AI research gathering data and programming in a very easy manner.



(a) (b)

(c) (d)

Figure 26 Different AIRSIM environments

The above four images shown in Figure 26 are the environments used in the AIRSIM environment to test the performance of the UAV system. The luminosity of the environments has also been changed and experimented with. The environments shown and experimented with are Africa Env, High Mountains Env, UrbanNH Env, and MSBuild Env respectively.

Python coding language can be used to communicate with the AIRSIM environment. There are inbuilt client packages, that can be leveraged for communicating and collecting data from the UAV with ease. Also, the feasibility of the simulator to closely replicate real-time environment conditions and the ability to alter various parameters, makes it stand out for robustly mimicking the UAV operation.

6.2.2 Gazebo

Gazebo simulator is popularly used among researchers to test robots efficiently in custom-built indoor and outdoor environments with both car and UAV models available with easy modeling and convenient interfaces.

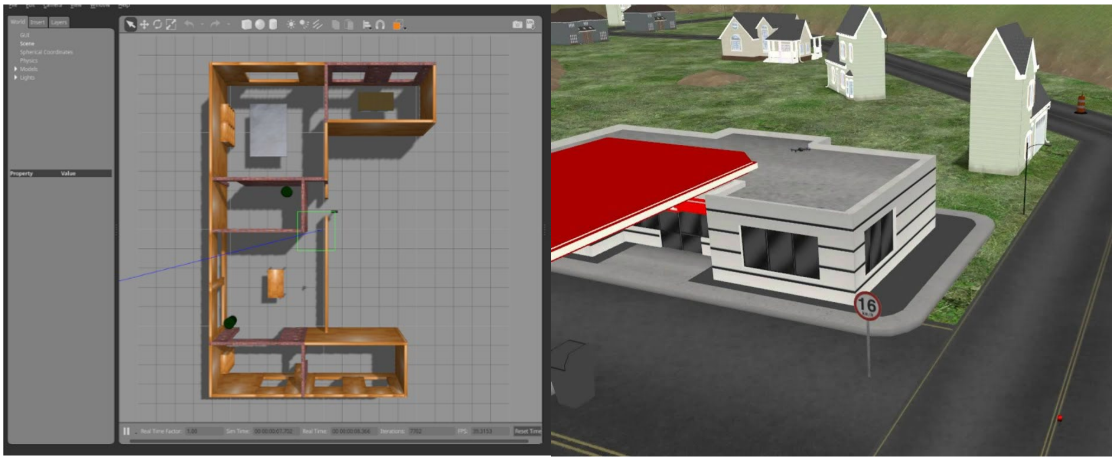


Figure 27 Different Gazebo worlds

Figure 27 shows the different worlds used for testing the performance of the obstacle avoidance capability of the proposed E2ES. Python programming language was used leveraging the Robotic Operating System (ROS). Visualization was done using an RVIZ visualizer.

6.2.3 MATLAB

MATLAB is a numeric computing and programming language that has been leveraged in our work especially for testing various architectures. Implementation and testing of various algorithms can be done in MATLAB with ease.

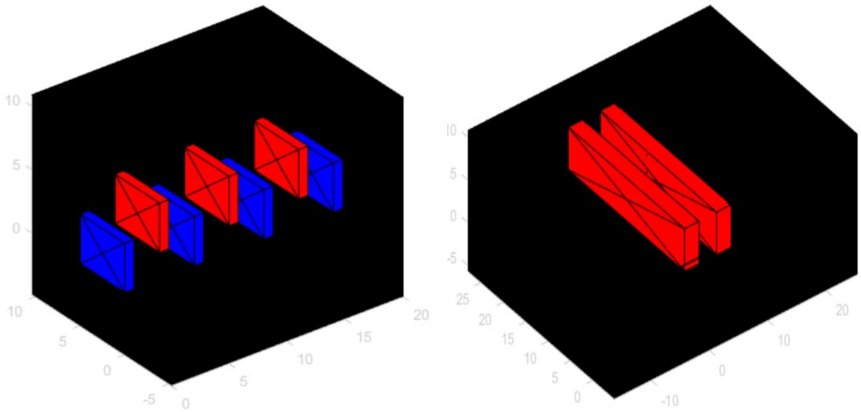


Figure 28 Different MATLAB environments

Figure 28 shows some simple environments used in MATLAB for testing the submodules, that is the path planning module’s performance with ground truth data.

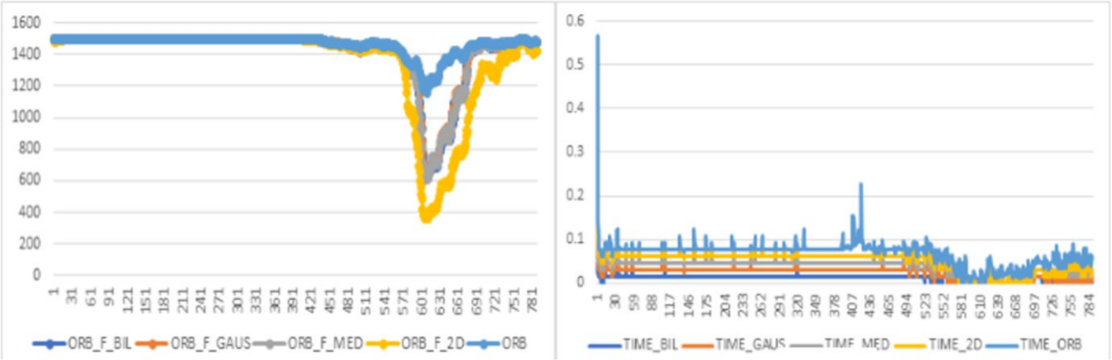
MATLAB is chosen as the software for choice for simulation since it provides an upper hand while working with mathematical functions especially multivariable mathematics related to MPC and kinodynamic planning. It is also easy to implement and deploy to test with work cases and analyze the results with interactive plots and store the data easily. It also supports using the same MATLAB functions in python with the plugin. Extensive support with ROS and Gazebo simulator is provided to ease cross-platform direct deployment in case of real-time testing.

CHAPTER 7

EXPERIMENTATION AND EVALUATION

7.1 Image Preprocessing

Image preprocessing was mandatory for all frames irrespective of the module including the obstacle avoidance module and the image classification module.



(a) (b)

Figure 29 Image Preprocessing Results

Figure 29 shows various image preprocessors tested with the input frames. Bilateral filtering, Gaussian filtering, Median blurring, and 2D image filtering were some of the preprocessors used for comparison. Figure 29 (a) represents the number of features extracted from each preprocessor and Figure 29 (b) represents the computational cost of each preprocessor. Comparatively better results were obtained when 2D image filtering was used.

7.2 Localization

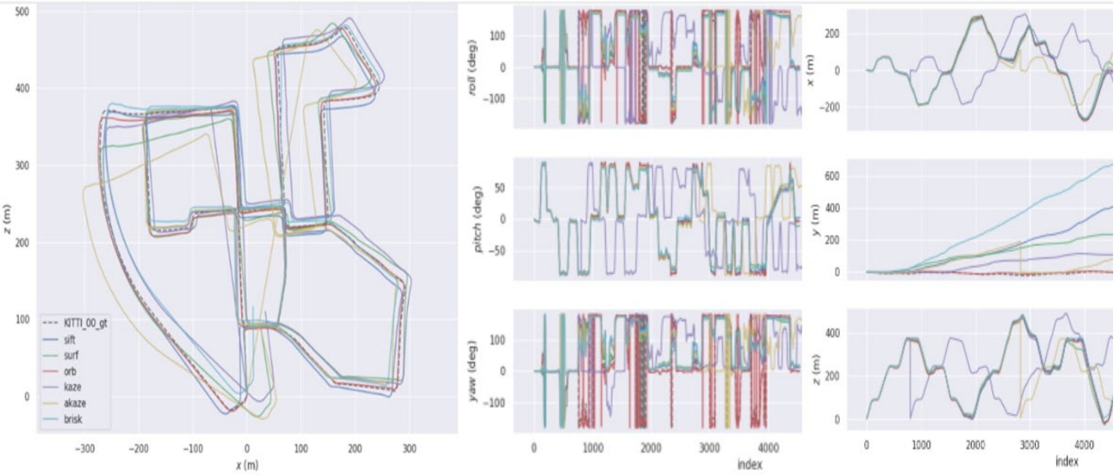
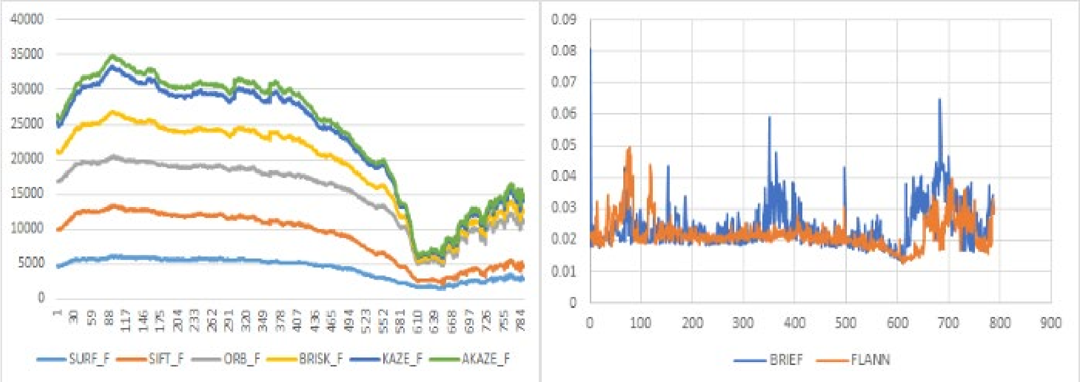


Figure 30 Trajectories using various feature extractors

Figure 30 shows the trajectories formulated in the KITTI dataset and the drift observed when different feature extractors were used for localization. The drift refers to the variation in the position variations and the pitch, roll, and yaw angles. Figure 31 (a) compares the different feature extractor's ability to grasp features in different varying lighting conditions and Figure 31 (b) shows the performance of two feature matches.



(a) (b)

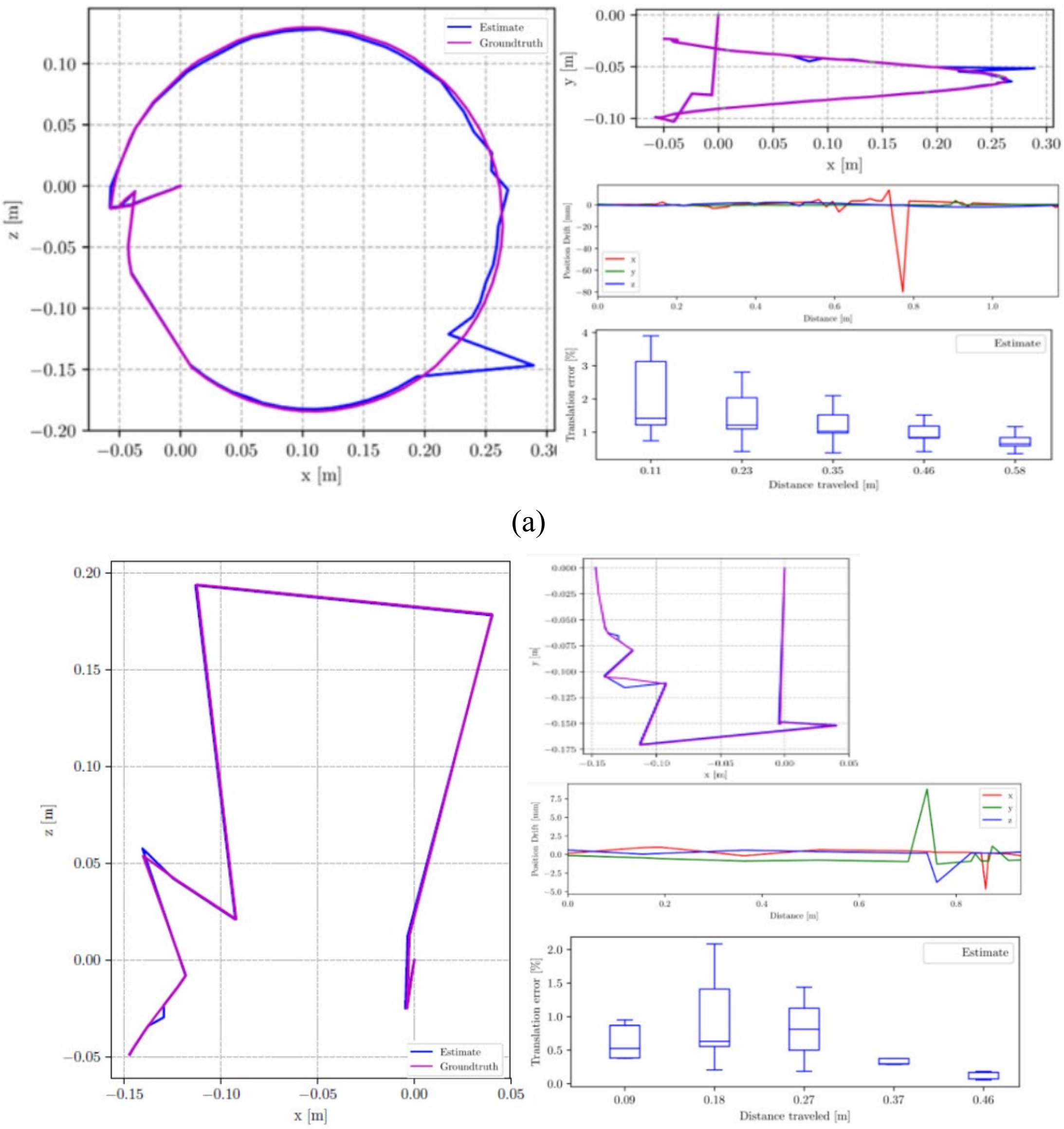
Figure 31 Comparing feature extractors and Matchers

In Table 3, we analyzed the computational power and storage required when each extractor was used for localization. Thus, we will be using ORB for localization. It has been noticed that ORB produces comparatively many features when compared to the other extractors, keeping the computation and error rate minimum.

Table 3 Performance of the odometry approaches

	ORB	SIFT	SURF	KAZE	AKAZE	BRISK
# Features	9920257	7880352	9121625	7647275	6038516	11324156
Tot. time	387.1427	1035.8436	724.0894	4087.2750	922.7840	675.3332
Tot. error	346.2213	412.6922	241.2476	164.7603	400.5994	678.8387
Storage	317.44	1000.8	584.78	978.85	744.32	724.74

Thus, the ORB feature extractor's ability to extract a good quantity of features irrespective of the luminosity of the environment with minimum computation gives it an edge over other techniques. Thus, ORB is used in our works. The maximum number of features in 640 x 480 pixels is set at 1,000. It was observed to produce an average of 640 features per frame. Most features were distributed in the regions of maximum variations in pixel intensity



(a)

(b)

Figure 32 Localization results for a circular trajectory

For analyzing the efficiency of the localization module leveraging ORB feature extractors, we perform a movement along with a circular and a random trajectory as it encompasses several rotational and translational movements as shown in Figure 32. Loop closure was observed with significant prediction accuracy. It was observed that the Root Mean Square Error (RMSE) obtained in the mentioned trajectory resulted in only 3.9456 for position when verified with the ground truth data obtained from GPS RTK.

7.3 Obstacle Avoidance

7.3.1 Learning-free Avoidance

The learning-free as explained in previous sections is split into three domains: perception, planning, and control. In this subsection, we will explain the work done in all three domains in detail starting from perception to the control system.

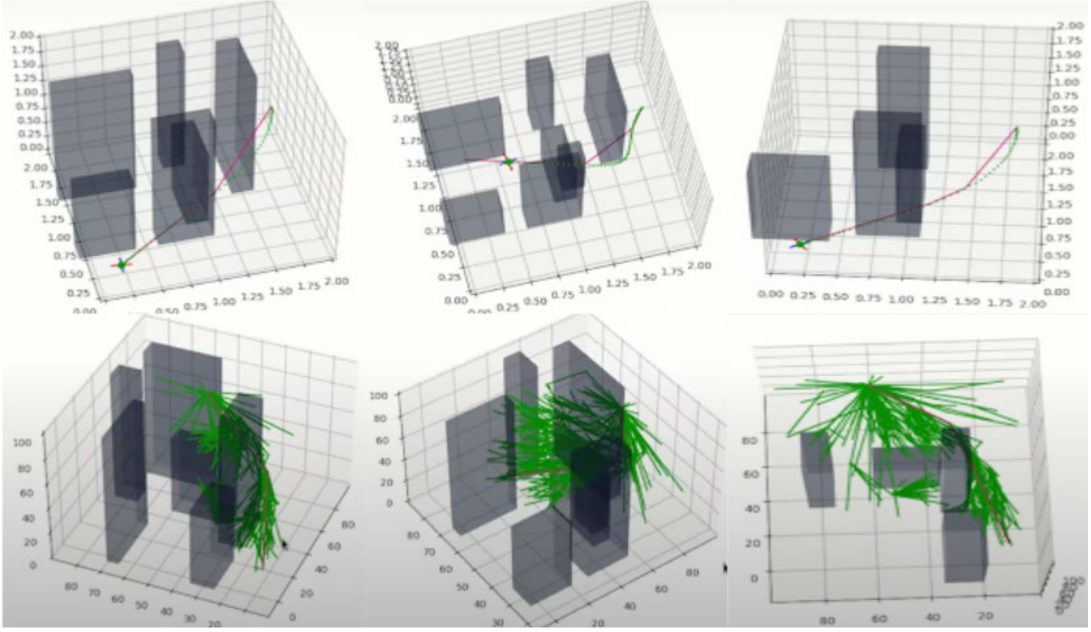
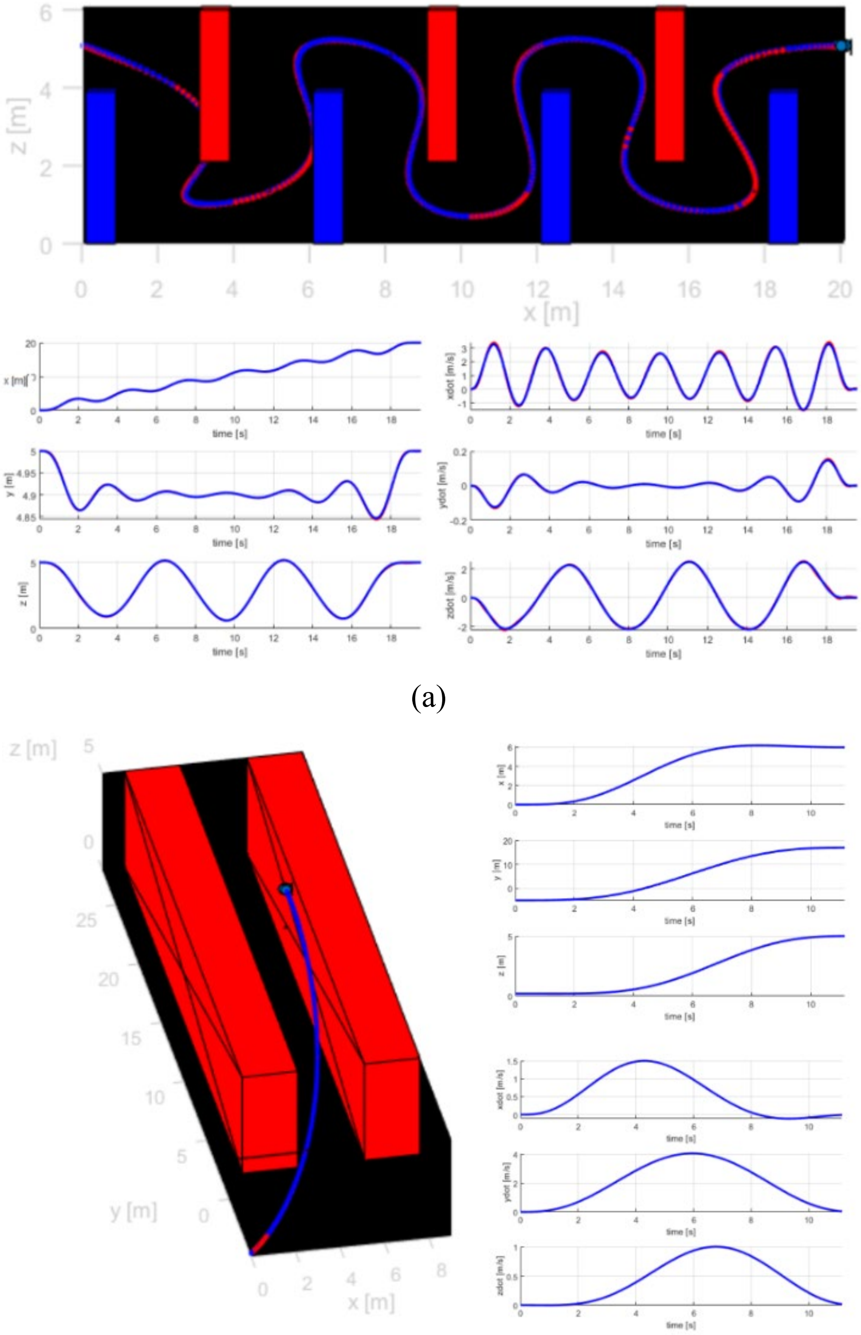


Figure 33 Kinodynamic RRT-based navigation

Similar to A* based kinodynamic trajectory planning a simulation of RRT-based kinodynamic planning was executed in a simulator-based python replicating a real-time behavior of the quadcopter. The simulation was tested in multiple obstacle maps with the number of obstacle blocks ranging from 4 to 7. As the result of the simulation, it was concluded RRT is better than A* for kinodynamic planning in terms of performance and reliability of finding the path. The average number of paths explored during the exploration was 67 and the average time it took to find the most reliable path was about 0.2 Seconds to complete the exploration. With the above info, it was concluded that RRT-based simulation yielded better performance.



(a)

(b)

Figure 34 Kinodynamic A*-based navigation

Figure 34 shows experiments performed with a defined quadcopter model in the MATLAB simulator. We use a kinodynamic planner along with the A* path planning algorithm to control the quadcopter in an agile manner from start to goal position. Several maps (Figure 34 (a) and (b)) were initialized in the planner to test the agility of the proposed global planner. The results were analyzed using the position vs time and velocity vs time graphs. It was also ensured that the slope of the curve at any given point of time was less than 0.4 thereby smooth trajectories can be generated. The red trajectory is the predicted trajectory using the A* path planning and kinodynamic trajectory generation, while the blue trajectory is the actual trajectory taken by the quadcopter over the constraints.

A huge overlap of the expected and actual trajectory was observed with several test cases as shown in Figure 34 being highly challenging since the agility of the quadcopter is tested to the maximum potential since the obstacles are placed in a very close manner. The quadcopter performed very well in the aforementioned test case with RMSE in the expected trajectory and the actual trajectory is estimated to be 0.66823. An average of 62 paths were explored and the average time of exploration is estimated to be 6.2 seconds.

A* algorithm compared to RRT is relatively slow since it checks all the nodes and all nearest edges to the start node for finding the most suitable path. While RRT explores the nearest node while keeping the computational cost low most of the time. The results of the previous experiments concluded that the path derived from A* is complete and the shortest route chosen with the time taken to find is comparatively higher. RRT has the upper hand in finding the shortest path with limited resources and in less time while A* has the most reliable path for the traversal

7.3.3 Learning-based Avoidance

As explained in the previous sections, imitation learning using SE-ResNet is used for obstacle avoidance. The performance of the complete proposed learning-based approach adopted and the quality of the network used is evaluated in this section. Also, different networks were experimented with to increase the robustness of the system.

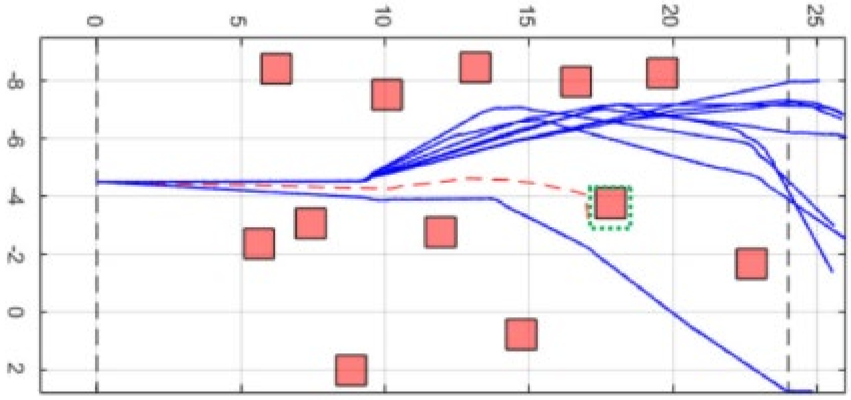


Figure 35 Trajectory taken by Learning-based UAV avoidance

Figure 35 shows the result of the learning-based technique equipped using the SE-ResNet network. The map used is Map4 as arranged in the following pages. Similarly, three more maps were formed to test the algorithm with different obstacle patterns.

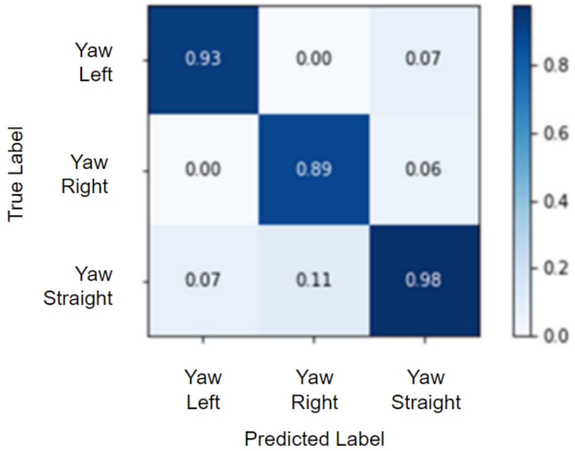


Figure 36 Prediction accuracy of the trained model

Figure 36 shows the prediction accuracy of the model trained using the input data from the learning-free technique. The output classes include yaw right, yaw left and yaw straight. These are directions, instead of angles, so that the system tends to be more reliable. Thereby locally navigating in environments as shown in Figure 34, results in better performance in terms of computation and reliability. It can be noted that more than 89% of the predicted results tend to match the true labels.

Table 4 Computation time for learning

	Computation Time
FCN	7.45
ME-CapsNet	17.34
SE-ResNet	15.39

Table 4 corresponds to the computational time required for the 3 different networks trained. FCN seems to be the fastest because it doesn't use dense layers, which ultimately results in less number of parameters. Capsule layers in the ME-CapsNet slightly increase the computation when compared to SE-ResNets.

Table 5 Performance of the networks on different test maps

	FCN	ME-CapsNet	SE-ResNet
Map1	10	45	85
Map2	15	50	90
Map3	10	60	75
Map4	15	55	90

Table 5 represents the performance of the network when put through different maps. Map 1 and Map 3 are simple maps, while Map 2 and Map 4 are a bit complex maps, with a greater number of obstacles in the way. Each network is tested 10 times on each map and the mean average is taken as the performance factor. FCN has performed the least effective of the 3 networks, which implies how important feature extraction is, for getting better performance. SE-ResNet is better than ME-CapsNet in terms of the performance factor and computation as shown in Table 4.

7.4 Inspection

As explained in Section 5.3, the mechanism behind the inspection module of the system leveraging the sense-switch-act mechanism, we will be proving the reasons behind the mechanism in terms of computational cost and the accuracy of both the models in separate subsections.

Table 6 Computation analysis of inspection module

	Classification	Detection
Forest Fire	1.27	2.34
Search & Rescue	1.34	2.79
Agricultural Burns	1.18	2.21

As observed in Table 6, object detection is computationally expensive. Thus, instead of relying on object detection in the entire course (where there could be no obstacles), we have used the above-mentioned mechanism to save computation significantly. Also, even if a class is sensed from the image classification sub-module, the object detection module will only be triggered in intervals to prevent similar frame detections as a result of hovering or slow speed. This was done to prevent redundancy in the observed detection classes.

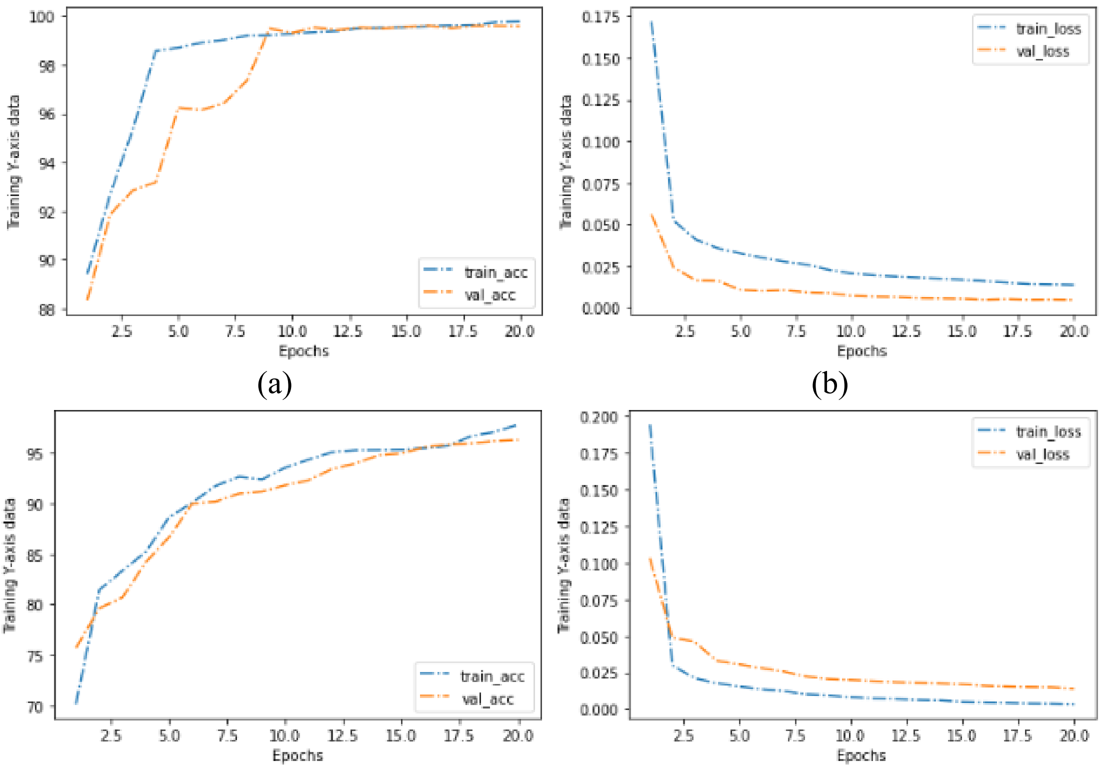
7.4.1 Image Classification

ME-CapsNet easily outperforms the research works on CapsNet in CIFAR10 datasets, thus demonstrating its ability to capture important features and routes without feature loss. The results can be visualized in Table 7. Also, 1.23% better performance in terms of accuracy was estimated using our model with the FashionMNIST dataset. Though the difference in accuracy between the proposed approach and the previous best approach is only 0.13%, computationally our approach tends to be very computationally cheaper. This is because of the less computational increase when using deep convolutional layers via SENets in between the CapsNet layers strategically with well-defined parameters.

Table 7 Comparison of the various proposed network

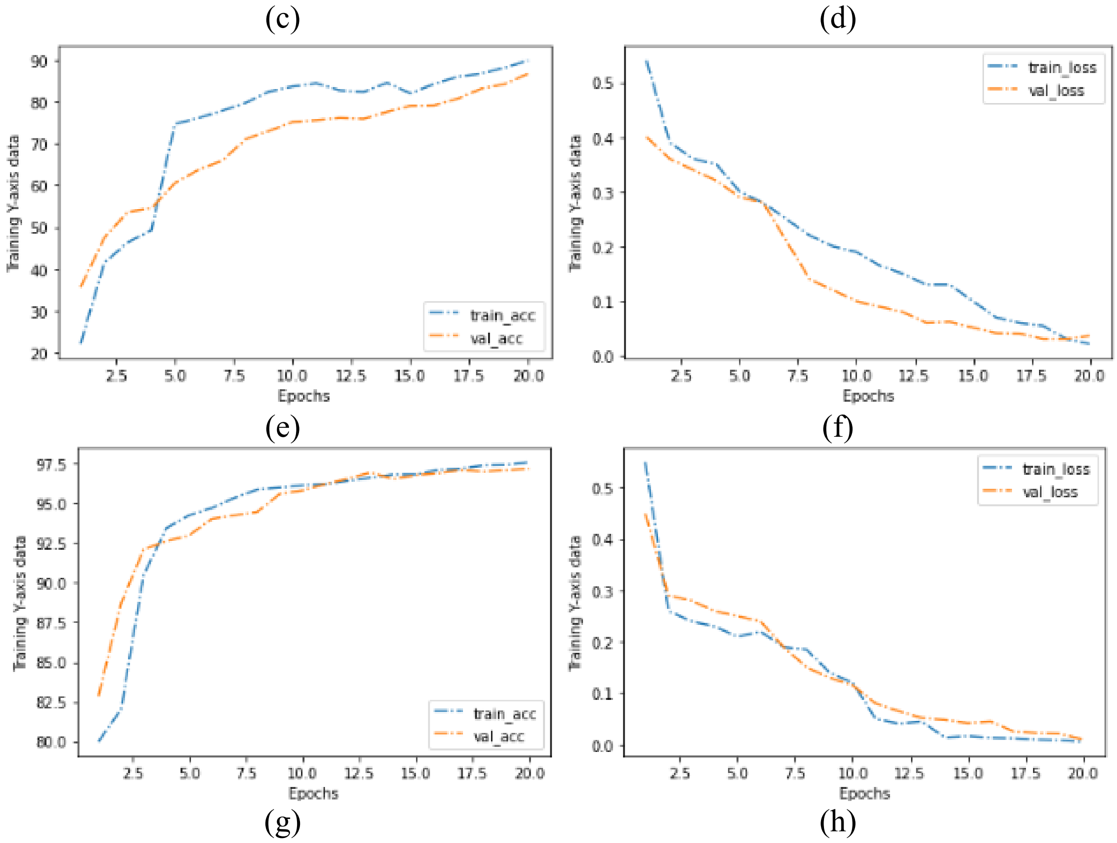
	Accuracy	# Of Params	Recon.
HitNet [36]	73.30 %	8.89 M	Yes
MS-CapsNet [67]	75.70 %	11.20 M	Yes
CapsNet Baseline	79.24 %	11.98 M	No
DeeperCaps [68]	81.29 %	5.81 M	Yes
DCNet [69]	82.63 %	11.88 M	Yes
DA-CapsNet [37]	85.47 %	7 M	Yes
Cv-CapsNet++ [70]	86.70 %	2.69 M	No
AR-CapsNet [71]	88.94 %	9.60 M	Yes
Sabour et al. [56]	89.40 %	14.36 M	Yes
DCNet++ [69]	89.71 %	13.4 M	Yes
ME-CapsNet (Ours)	**89.84 %**	**7.24 M**	**Yes**

Figure 37 shows the training and validation accuracy of ME-CapsNet when tested with state-of-the-art datasets. Figure 37 (a) and (b) correspond to MNIST datasets, Figure 37 (c) and (d) correspond to the FashionMNIST dataset, Figure 37 (e) and (f) correspond to the CIFAR10 dataset, and Figure 37 (g) and (h) correspond to KMNIST.



(a)

(b)



(c) (d)

(e) (f)

(g) (h)

Figure 37 Evaluation of the network using various datasets

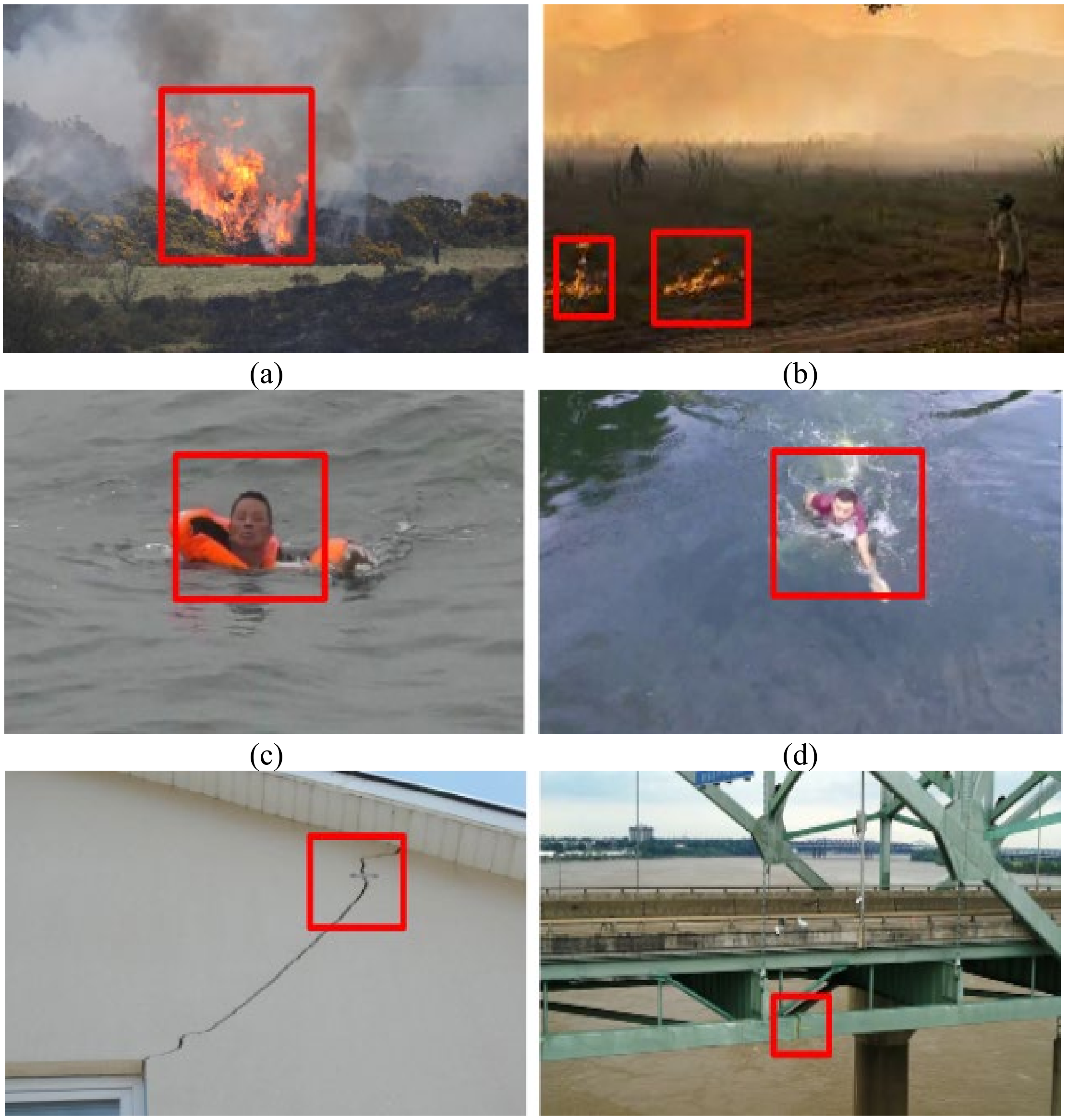
In this subsection, various backbone architectures including residual networks with various levels of layers, inception networks, and VGG networks are fused as input layers for ME-CapsNet and the respective test accuracy is logged and compared systematically for four different datasets.

Table 8 Comparison of various backbone architectures

	MNIST	FashionMNIST	CIFAR10	KMNIST
ResNet-50	99.57	94.85	88.87	96.75
ResNet-101	**99.83**	**95.63**	89.37	**97.84**
ResNet-152	99.78	95.91	**89.84**	97.11
VGG-16 [72]	98.54	95.11	87.72	96.49
Inception [73]	98.63	95.38	88.98	97.34

7.4.2 Object detection

We have tested the performance of our object detection module in different scenarios with diverse illumination conditions and it can be visualized in Figure 38. Testing was done targeting forest fire detection, agricultural burn detection, and search and rescue operations. We have obtained about 90%+ accuracy with frames in different environments experimented with.



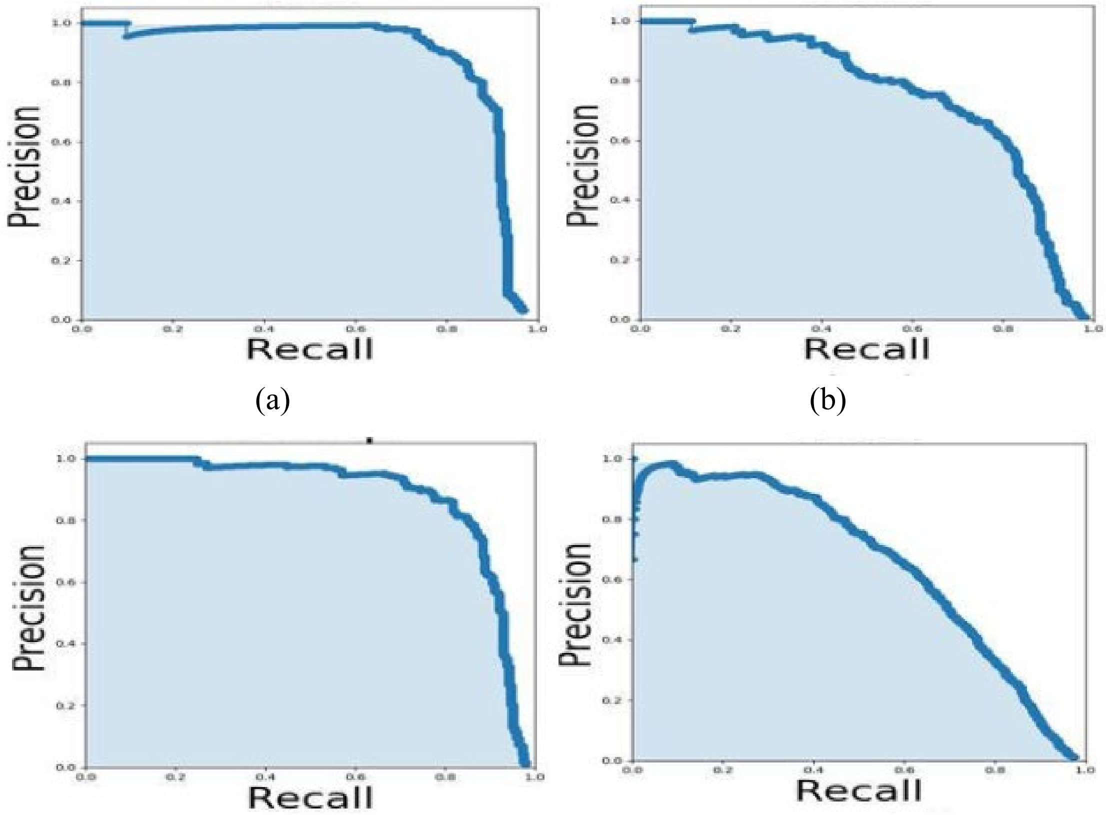
(a) (b)

(c) (d)

(e) (f)

Figure 38 Object detection using the YOLO model

In Figure 38, (a) and (b) represents fire detection for a forest fire or agricultural burns, (c) and (d) represent person detection in search and rescue operations, and (e) and (f) represent defect detection in construction and bridges. Thus, with these examples, the detection capability of the system is proved to be accurate. The system has been building such a way that, just by loading the appropriate previously trained model, the detection will take place based on the task.



(a) (b)

(c) (d)

Figure 39 Evaluation of object detection

In Figure 39, evaluation matrices are used to prove the efficiency of the object detection submodule. Figure 39 (a) – (d) correspond the performance of the object detection module with fire, person, poles, and cracks respectively. Precision and recall are the two-evaluation metrics used here. Precision is the intersection of the detected box and object divided by the detected box. Meanwhile, recall is the intersection of the detected box and object divided by the object.

CHAPTER 8

CONCLUSION

8.1 Contributions

An end-to-end autonomous navigation system for UAVs has been proposed in this thesis, which can efficiently traverse in GPS denied, unstructured environments by leveraging visual and inertial sensors only. The system is equipped with powerful deep neural network pipelines for obstacle avoidance and inspecting environments. The inspection module is a plug-and-play module, which is based on a novel sense-switch-act mechanism and can be altered based on the task in a modular way.

Two navigation modules are designed, one based on learning and another using a traditional learning-free approach. For the learning-free approach, monocular depth estimation via unsupervised learning was used for perceiving the environment, and point clouds were extracted to form the 2D occupancy grid. Over the grid, path planning using kinodynamic RRT planning and MPC control system was adapted for efficient navigation. The training data for the learning-based approach was procured during navigation via the learning-free avoidance approach, thereby learning a policy for the learning-based approach from its data.

The reliability of the proposed system has been tested by taking three tasks - search and rescue operations, inspection/ monitoring, and forest fire tasks. The inspection module is based on a sense-switch-act technique, where the object detection submodule is triggered only if a particular task-specific class is sensed by the classification module, thereby reducing computation significantly.

A custom localization framework has been designed leveraging the ORB SLAM technique. Sensor fusion using inertial sensors on the ORB features was done using Extended Kalman Filters, thereby significantly improving the reliability of the system in unstructured or cluttered environments.

A novel proposed neural network called ME-CapsNet is used for image classification, which was designed by strategically fusing the Squeeze-Excitation Network and Capsule Network over the Residual Network backbone. For the learning-based obstacle avoidance approach, the data from the traditional learning-free approach was sent through the Squeeze-Excitation Network to learn the policy thereby mapping the input to the yaw directions from the UAV perspective.

The UAV system is made agile and is highly capable of navigating in unstructured environments, which gives an added benefit of navigating at heights as low as 150 cm from the ground level. It in turn gives a proper understanding of scenes without any required rotation variants. At that height, it has been observed that the system was able to evade 95% of the obstacles (trees, people, trucks, houses, poles) with the equipped learning-based approach.

8.2 Future Works

We aim to create efficient mapping techniques in the future to fuse with the UAV system proposed here. This can assist in more robust surveillance tasks and optimizes the navigation of UAVs (shortest time). Also, removing dynamic features from the projected map and reducing the outlier using efficient sampling using RANSAC algorithms is one future research direction.

The implementation used here is based only on static obstacles, but in real-time, we can expect dynamic obstacles as well like birds, kites, etc. Thus, leveraging optical flow and semantic segmentation approaches can be used to understand the trajectory of the dynamic obstacle and evade it accordingly. This sub-module could easily be merged with the perception sub-module used in the avoidance module, considering our architecture is designed in a very modular manner.

With several improvements in model predictive control, one possible improvement could be in implementing a learning-based model predictive control. Adding intelligence to a powerful controller can make it more powerful and efficient and make the process of autonomous navigation safer and more agile.

Another possible research direction for the future could be introducing optimization techniques that can effectively reduce the reprojection error caused by odometry using visual sensors. They result in an accumulation of errors in the localization results, thereby making significant drift. Thus, improvements/ optimizations can be done to reduce the drift in the odometry data.

Another improvement to the system could be to implement the process on a Neuromorphic camera or an event camera. Event Camera is a novel bio-inspired vision sensor that outputs spikes when it observes a change in pixel brightness intensity. Instead of capturing images at a fixed rate, they asynchronously measure per-pixel brightness changes and output a stream of events that encode the time, location, and sign of the brightness changes. Considering the low latency requirement and high agile maneuverability, the Aerial system is equipped with an Event/ Neuromorphic camera to perform the process of navigation and inspection.

REFERENCES

[1] Howard A. Real-time stereo visual odometry for autonomous ground vehicles. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2008:3946-52.

[2] Nistér D, Naroditsky O, Bergen J. Visual odometry for ground vehicle applications. J Field Robotics. 2006 01;23:3-20.

[3] Jiang Y, Xu Y, Liu Y. Performance evaluation of feature detection and matching in stereo visual odometry. Neurocomputing. 2013 11;120:380-90.

[4] [4] Nourani-Vatani N, Roberts JM, Srinivasan MV. Practical visual odometry for car-like vehicles. 2009 IEEE International Conference on Robotics and Automation. 2009:3551-7.

[5] McManus C, Furgale PT, Barfoot TD. Towards lighting-invariant visual navigation: An appearance-based approach using scanning laser-rangefinders. Robotics Auton Syst. 2013;61:836-52.

[6] González R, Rodríguez F, Guzmán JL, Pradalier C, Siegwart RY. Combined visual odometry and visual compass for off-road mobile robots' localization. Robotica. 2011;30:865 878.

[7] Scaramuzza D, Siegwart RY. Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. IEEE Transactions on Robotics. 2008;24:1015-26.

[8] Kwag YK, Kang JW. Obstacle awareness and collision avoidance radar sensor system for low-altitude flying smart UAV. In: The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576). vol. 2; 2004. p. 12.D.2-121.

[9] Ajith Kumar B, Ghose D. Radar-assisted collision avoidance/guidance strategy for planar flight. IEEE Transactions on Aerospace and Electronic Systems. 2001;37(1):77-90.

[10] Kim J, Sukkarieh S. Autonomous airborne navigation in unknown terrain environments. IEEE Transactions on Aerospace and Electronic Systems. 2004;40(3):1031-45.

[11] Magree D, Mooney JG, Johnson EN. Monocular visual mapping for obstacle avoidance on UAVs. In: 2013 International Conference on Unmanned Aircraft Systems (ICUAS); 2013. p. 471-9

[12] Byrne J, Cosgrove M, Mehra R. Stereo based obstacle detection for an unmanned air vehicle. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.; 2006. p. 2830-5.

[13] Hrabar S. 3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2008. p. 807-14.

[14] Dur E. Optical Flow-based obstacle detection and avoidance behaviors for mobile robots used in unmaned planetary exploration. In: 2009 4th International Conference on Recent Advances in Space Technologies; 2009. p. 638-47.

[15] Song KT, Huang JH. Fast optical flow estimation and its application to real-time obstacle avoidance. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164). vol. 3; 2001. p. 2891-6 vol.3.

[16] Hrabar S, Sukhatme G. Vision-Based Navigation through Urban Canyons. J Field Robotics. 2009 05;26:431-52.

[17] Yu H, Beard R. A vision-based collision avoidance technique for micro air vehicles using local-level frame mapping and path planning. Autonomous Robots. 2013 01;34.

[18] Godard C, Aodha OM, Brostow GJ. Digging Into Self-Supervised Monocular Depth Estimation. CoRR. 2018; abs/1806.01260. Available from: http://arxiv.org/abs/1806.01260.

[19] Godard C, Aodha OM, Brostow GJ. Unsupervised Monocular Depth Estimation with LeftRight Consistency. CoRR. 2016;abs/1609.03677. Available from: http://arxiv.org/abs/ 1609.03677.

[20] Jung D, Choi J, Lee Y, Kim D, Kim C, Manocha D, et al. DnD: Dense Depth Estimation in Crowded Dynamic Indoor Scenes. CoRR. 2021;abs/2108.05615. Available from: https: //arxiv.org/abs/2108.05615.

[21] Musliman IA, Rahman AA, Coors V. IMPLEMENTING 3 D NETWORK ANALYSIS IN 3 D-GIS; 2008.

[22] de Filippis L, Guglieri G, Quagliotti F. Path Planning Strategies for UAVS in 3D Environments. Journal of Intelligent & Robotic Systems. 2012;65:247-64.

[23] Carsten J, Ferguson D, Stentz A. 3D Field D: Improved Path Planning and Replanning in Three Dimensions. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2006:3381-6.

[24] Fei Yan JZX Yi-Sha Liu. Path Planning in Complex 3D Environments Using a Probabilistic Roadmap Method; 2013. Available from: https://www.mi-research.net/en/article/doi/ 10.1007/s11633-013-0750-9.

[25] Yang K, Sukkarieh S. Real-time continuous curvature path planning of UAVS in cluttered environments. 2008 5th International Symposium on Mechatronics and Its Applications. 2008:1-6.

[26] Pomerleau D. ALVINN: An Autonomous Land Vehicle In a Neural Network. In: Touretzky DS, editor. Proceedings of (NeurIPS) Neural Information Processing Systems; 1989. p. 305 313.

[27] Lecun Y, Muller U, Ben J, Cosatto E, Flepp B. Off-Road Obstacle Avoidance through Endto-End Learning. Available from: http://yann.lecun.com.

[28] Park B, Oh H. Vision-Based Obstacle Avoidance for UAVs via Imitation Learning with Sequential Neural Networks. International Journal of Aeronautical and Space Sciences. 2020 9;21:768-79.

[29] Kumaar S, Sangotra A, Kumar S, Gupta M, B N, Omkar SN. Learning to Navigate Autonomously in Outdoor Environments: MAVNet. 2018 9. Available from: http://arxiv. org/abs/1809.00396.

[30] Wang T, Qin R, Chen Y, Snoussi H, Choi C. A reinforcement learning approach for UAV target searching and tracking. Multimedia Tools and Applications. 2019;78(4):4347-64.

[31] Loquercio A, Kaufmann E, Ranftl R, Müller M, Koltun V, Scaramuzza D. Learning high-speed flight in the wild. Science Robotics. 2021 10;6.

[32] Bansal M, Krizhevsky A, Ogale A. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. 2018 12. Available from: http://arxiv.org/abs/1812.03079.

[33] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. Commun ACM. 2017 may;60(6):84–90. Available from: https://doi.org/ 10.1145/3065386.

[34] Ananthi G, Arockia Selvakumar A. In: Review on Deep Learning Algorithms in Medical Devices. Cham: Springer International Publishing; 2020. p. 1-27. Available from: https: //doi.org/10.1007/978-3-030-58675-1_167-1.

[35] Mazzia V, Salvetti F, Chiaberge M. Efficient-CapsNet: Capsule Network with Self-Attention Routing. CoRR. 2021;abs/2101.12491. Available from: https://arxiv.org/abs/2101. 12491.

[36] Deliège A, Cioppa A, Droogenbroeck MV. HitNet: a neural network with capsules embedded in a Hit-or-Miss layer, extended with hybrid data augmentation and ghost capsules. CoRR. 2018;abs/1806.06519. Available from: http://arxiv.org/abs/1806.06519.

[37] Huang W, Zhou F. DA-CapsNet: dual attention mechanism capsule network. Sci Rep. 2020. Available from: https://doi.org/10.1038/s41598-020-68453-w.

[38] Yang S, Lee F, Miao R, Cai J, Chen L, Yao W, et al. RS-CapsNet: An Advanced Capsule Network. IEEE Access. 2020;8:85007-18.

[39] Fu C, Liu W, Ranga A, Tyagi A, Berg AC. DSSD : Deconvolutional Single Shot Detector. CoRR. 2017;abs/1701.06659. Available from: http://arxiv.org/abs/1701.06659.

[40] Redmon J, Farhadi A. YOLOv3: An Incremental Improvement. CoRR. 2018;abs/1804.02767. Available from: http://arxiv.org/abs/1804.02767.

[41] Liu W, Anguelov D, Erhan D, Szegedy C, Reed SE, Fu C, et al. SSD: Single Shot MultiBox Detector. CoRR. 2015;abs/1512.02325. Available from: http://arxiv.org/abs/1512.02325.

[42] Najibi M, Rastegari M, Davis LS. G-CNN: an Iterative Grid Based Object Detector. CoRR. 2015;abs/1512.07729. Available from: http://arxiv.org/abs/1512.07729.

[43] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger. CoRR. 2016;abs/1612.08242. Available from: http://arxiv.org/abs/1612.08242.

[44] Miller GA, Beckwith R, Fellbaum CD, Gross D, Miller KJ. Introduction to WordNet: An On-line Lexical Database. International Journal of Lexicography. 1990;3:235-44.

[45] Viswanathan D. Features from Accelerated Segment Test (FAST); 2011.

[46] Lowe David G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision. 2004.

[47] Rublee E, Rabaud V, Konolige K, Bradski G. ORB: An efficient alternative to SIFT or SURF. In: 2011 International Conference on Computer Vision; 2011. p. 2564-71.

[48] Bay H, Tuytelaars T, Van Gool L. SURF: Speeded Up Robust Features. In: Leonardis A, Bischof H, Pinz A, editors. Computer Vision – ECCV 2006. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006. p. 404-17.

[49] Shi J, Tomasi. Good features to track. In: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 1994. p. 593-600.

[50] Schaal S. Learning from Demonstration. Available from: http://www.cc.gatech.edulfac/ Stefan.Schaal.

[51] Mustaqeem A, Javed A, Fatima T. An Efficient Brain Tumor Detection Algorithm Using Watershed Thresholding Based Segmentation. International Journal of Image, Graphics and Signal Processing. 2012 09;4.

[52] Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. In: 2017 International Conference on Engineering and Technology (ICET); 2017. p. 1-6.

[53] Yamashita R, Nishio M, Do R, Togashi K. Convolutional neural networks: an overview and application in radiology. Insights into Imaging. 2018 06;9.

[54] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15(56):1929-58. Available from: http://jmlr.org/papers/v15/srivastava14a.html.

[55] Thakkar V, Tewary S, Chakraborty C. Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data. In: 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT); 2018. p. 1-5.

[56] Sabour S, Frosst N, Hinton GE. Dynamic Routing Between Capsules. CoRR. 2017;abs/1710.09829. Available from: http://arxiv.org/abs/1710.09829.

[57] Godard C, Mac Aodha O, Firman M, Brostow GJ. Digging into Self-Supervised Monocular Depth Prediction. 2019 October.

[58] Zhai S, Wu H, Kumar A, Cheng Y, Lu Y, Zhang Z, et al. S3Pool: Pooling with Stochastic Spatial Sampling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017

[59] Huang G, Liu Z, Weinberger KQ. Densely Connected Convolutional Networks. CoRR. 2016;abs/1608.06993. Available from: http://arxiv.org/abs/1608.06993.

[60] Deng L. The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine. 2012;29(6):141-2.

[61] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. CoRR. 2017;abs/1708.07747. Available from: http://arxiv. org/abs/1708.07747.

[62] Clanuwat T, Bober-Irizar M, Kitamoto A, Lamb A, Yamamoto K, Ha D. Deep Learning for Classical Japanese Literature. CoRR. 2018;abs/1812.01718. Available from: http://arxiv. org/abs/1812.01718.

[63] Krizhevsky A. Learning Multiple Layers of Features from Tiny Images. Citeseer, Tech Rep. 2009:32-3.

[64] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, et al. The Cityscapes Dataset for Semantic Urban Scene Understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016.

[65] Geiger A, Lenz P, Stiller C, Urtasun R. Vision Meets Robotics: The KITTI Dataset. 2013 sep;32(11):1231–1237. Available from: https://doi.org/10.1177/0278364913491297.

[66] Shah S, Dey D, Lovett C, Kapoor A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. 2017 5. Available from: http://arxiv.org/abs/1705.05065.

[67] Xiang C, Zhang L, Tang Y, Zou W, Xu C. MS-CapsNet: A Novel Multi-Scale Capsule Network. IEEE Signal Processing Letters. 2018;25(12):1850-4.

[68] Rajasegaran J, Jayasundara V, Jayasekara S, Jayasekara H, Seneviratne S, Rodrigo R. DeepCaps: Going Deeper with Capsule Networks. CoRR. 2019. Available from: http: //arxiv.org/abs/1904.09546.

[69] Phaye SSR, Sikka A, Dhall A, Bathula DR. Dense and Diverse Capsule Networks: Making the Capsules Learn Better. CoRR. 2018;abs/1805.04001. Available from: http://arxiv.org/ abs/1805.04001.

[70] Cheng X, He J, He J, Xu H. Cv-CapsNet: Complex-Valued Capsule Network. IEEE Access. 2019;7:85492-9.

[71] Choi J, Seo H, Im S, Kang M. Attention routing between capsules. CoRR. 2019;abs/1907.01750. Available from: http://arxiv.org/abs/1907.01750.

[72] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 14091556. 2014 09.

[73] Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, et al. Going Deeper with Convolutions. CoRR. 2014;abs/1409.4842. Available from: http://arxiv.org/abs/1409. 4842.

APPENDIX A

Codes made towards the completion of the thesis objective can be obtained from this appendix.

1. https://github.com/E2ES-THESIS/QuadX - Consists of ROS-based UAV built with inbuilt sensors for the purpose of the thesis.
2. https://github.com/E2ES-THESIS/Custom-Visual-Odometry - Custom Monocular Visual Odometry package built from scratch and is linked to an evaluation package.
3. https://github.com/E2ES-THESIS/Custom-RTABMAP - Implementation of custom RTABMap SLAM technique using a mobile robot to test the mapping accuracy of the technique in indoor environments.
4. https://github.com/E2ES-THESIS/Obstacle-Avoidance-RRT - Implementation of UAV obstacle avoidance using the learning-free technique.
5. https://github.com/E2ES-THESIS/Object-detection - Object detection using YOLO and MobileNetSSD.
6. https://github.com/E2ES-THESIS/Monocular-depth-estimation - Implementation of monocular depth estimation in real-time using Jetson Xavier Board and Logitech Cam.
7. https://github.com/E2ES-THESIS/Obstacle-Avoidance - Obstacle Avoidance using the learning-free algorithm, that globally plans the suitable trajectory and uses adaptive velocity based on free spaces in the UAV perspective.
8. https://github.com/E2ES-THESIS/A-Star-Path-Planning - Custom implementation of 2D A* path planning algorithm using python.
9. https://github.com/E2ES-THESIS/Image-Classification - Various implementations of image classifications are split and logged in this repository.
10. https://github.com/E2ES-THESIS/KLT-Mono-Odometry - Codebase for Monocular odometry using OpenCV, LK Feature tracking and 5-point algorithm.
11. https://github.com/E2ES-THESIS/Object-Detection-PKG-ROS - ROS Node for custom object detection written in python.
12. https://github.com/E2ES-THESIS/ORB-BFMatcher - Testing code for ORB feature descriptor and Brute-Force Matcher.

13. https://github.com/E2ES-THESIS/QuadSim-Python - Codebase for python-based quadrotor simulator. Used for testing path planning and trajectory generation techniques.
14. https://github.com/E2ES-THESIS/ImageStitcher - Ortho-Mosaic image stitching for images captured from a Drone. Performs invariant transform to merge multiple images
15. https://github.com/E2ES-THESIS/ORB2SLAM_Support_pkg - ROS-based support packages for ORB2 SLAM. Contains Launch files and visualization codes written in ROSpy
16. https://github.com/E2ES-THESIS/YOLO-Object-Detection - Yolo model-based object detection node using OpenCV and DNN. Supports object detection on the real-time video stream.
17. https://github.com/E2ES-THESIS/TeleopKeyboard - Support ROS package for teleop operation using the keyboard. Can be used for both Aerial and Ground Robots.

APPENDIX B

Demo experimentation (predominantly simulations) are attached as links in this appendix.

1. https://youtu.be/1wTaUx1P6gE - Implementation of custom RTABMap SLAM technique using a mobile robot to test the mapping accuracy of the technique in indoor environments.
2. https://youtu.be/3JyuIRXX_9E - ORB2 SLAM implementation using KITTI dataset.
3. https://youtu.be/l6-3poF67ZA - An implementation of Visual Inertial Navigation System (VINS) using EUROC dataset. Used Stereo vision and IMU for efficient navigation especially in GPS-prone environments.
4. https://youtu.be/xQxt33pqoSg - Development of python package to reconstruct indoor/outdoor environments with diverse texture contrasts using Oriented FAST and Rotated Brief feature detector, FLANN based matches, and RANSAC for outlier removal; Optical flow and PnP (DLT and Levenberg) for estimating the pose of the robot.
5. https://youtu.be/ENfY3lAuCVA - ORB stands for Oriented FAST and Rotated Brief. It uses FAST extractors and BRIEF descriptors. It is widely used in several SLAM architectures for extraction steps including VI-ORB SLAM, MSCKF, etc.
6. https://youtu.be/dEREaBp6o-g - Various Frame Processing from a UAV perspective is visualized including segmentation, depth estimation, event data along with ground truth data.
7. https://youtu.be/9YORBRRYKHM - Simulation of a quadcopter in a gazebo simulator to localize itself in the global environment using the front-facing camera. The green dots thus represent the ORB features. It can also be visualized using the RVIZ tool as shown.
8. https://youtu.be/WhJ5JbXbA6w - Using point clouds obtained from the stereo cam, 2D histograms are generated from which cost function is derived. Based on the weight values in the free space, local waypoints are manifested and obstacle avoidance is established.
9. https://youtu.be/KZcNCHoezC8 - Developed a ROSpy-based control system for a quadcopter to transverse to a set of GPS setpoint autonomously. The Control System

has two modules namely the Altitude controller and the position controller, Altitude controller stabilizes the drone at the zero error Roll, Yaw, Pitch angles using a PID based controller, the position controller takes in the target GPS coordinate has setpoint values and calculates the roll yaw pitch angles to successfully move to the setpoint coordinates. these controllers work in synchronization to autonomously fly the drone from one coordinate to another.

10. https://youtu.be/KMF9M_KPoIM - UAV autonomous navigation using the learning-free technique, adapting its velocity based on the free space in the UAV's field of view.
11. https://youtu.be/pVTG0oIB4hs - Experimentation was performed on a python-based Quadrotor simulator. Path planning algorithm was tested on multiple maps of configuration space.
12. https://youtu.be/6eGLCexWC-Y - Autonomous obstacle avoidance using a learning-based technique in AIRSIM simulator
13. https://youtu.be/eGVrUrpGMpc - Monocular depth estimation in real-time using unsupervised networks.

APPENDIX C

Basics on UAV systems, application, cons, limitations, open problems

Types of UAVs

Multirotor- It gives great control and is generally very economical. They aren't very efficient as a lot of energy is used to fight gravity and hence their flight time is comparatively less and reduces further when payload weight increases.

Fixed-wing- Use wings for getting the lift. It is more efficient as they fight only air resistance and not gravity directly. Therefore, their flight duration is also long. But the maneuverability of fixed-wing UAVs is less. They cannot vertically take off or land.

Single rotor- Has one main rotor for lift and one tail rotor to change direction. They offer better control and are more efficient because of their long blades. They can take off and land vertically and can also hover in mid-air but requires more maintenance because of their mechanical complexity.

Flight Dynamics

- *Thrust-* Upward and Downward movement of the UAV. Reducing the velocity of all motors will lower the UAV and increasing all motor velocity will increase the height of the UAV from the ground.
- *Pitch-* Forward and backward movement of the UAV. Increasing the velocity of the forward motor will move the UAV backward and vice versa.
- *Roll-* Left and right movement of the UAV. By increasing the speed of left motors, the UAV will move right and vice versa.
- *Yaw-* Rotating the UAV in the left and right direction. By changing the speed of alternate motors, the UAV will yaw.

Applications

- *Aerial videography-* Because of their compact size, and stability like in multi-rotors, they are preferred for aerial videography and live coverage of sports.
- *Shipping-* They are also used for shipping and delivery.
- *Geographic mapping-* Mapping can be made more accurate and faster.

- *Disaster management-* Because of their small size, they can be easily deployed in various locations to get viable data about the situation.
- *Military-* Used in the military as they reduce human risks considering it is unmanned.
- *Agriculture-* With their speed and technologies, they can be used to monitor crop health over a wide area/ land.
- *Surveillance-* Can be used for surveillance and in law enforcement.
- *Search and rescue-* UAVs with their ability to reach almost any terrain can be effectively used for search and reduce operations.
- *Delivery-* With the rise in e-commerce, UAVs can be used to deliver packages much faster and hence reduce traffic on land.

Common Issues

- *Flight duration-* If the battery is not charged properly, the drone might lose control at any time. Also, the battery has to be properly maintained for a longer lifespan.
- *GPS Signal-* At some places, the signal might be lost hence losing control over the UAV. Also, the cycling rate of UAVs is low which can cause issues for agile flights.
- *Wrong calibration-* If the compass is not calibrated properly, then the UAV might fly randomly.

Research Areas

- *UAV Autonomy-* Improvement of flight control. Collision and obstacle avoidance. Better path planning. Communication and improved sensors.
- *Mapping-* Using the images from drones to map the surrounding.
- *Materials-* better materials lead to increased flight duration, control, and life span.
- *Hybrids-* Design of hybrid models that incorporate all the pros of fixed-wing drones and VTOLs, such as vertical take-off and landing, hover, maneuverability, agility, and flight duration.

Open Problems

- Improving flight duration in multirotor UAVs.
- Improve payload handling capacity.
- Ability to maintain predictability or continue the task even with a minor fault or unexpected disturbances during flight.

- Improvement of swarm technologies.
- Increasing the agility in multirotor UAVs.

Limitations

- They cannot be deployed during all-weather conditions. Highly dependent on wind, speed, and rain.
- The speed of the UAV can’t exceed a limit, usually around 30mph.
- Drones can cause injuries to birds and other animals or property upon collision.
- Drones can easily breach privacy.

APPENDIX D

What is SLAM? Why Visual SLAM? What are its applications?

Simultaneously Localization and Mapping (SLAM) is the process of creating maps and localizing a vehicle or robot in a world at the same time. It creates maps from unknown areas using mapping and localization techniques. GPS doesn't work well indoors. It's precise to a few meters only in outside environments and has a slow cycle rate (thus it's not suitable for nimble flight) as well. This level of precision is insufficient for autonomous applications. As a result, SLAM was created to address these challenges. The major advantages of SLAM are:

- SLAM uses many types of sensors like laser and camera.
- Indoor mapping and location identification are easier with SLAM.
- SLAM can produce 3D images of the surroundings.

SLAM is twofold as the name suggests, it needs to construct or update the map of an environment while simultaneously keeping track of the object's location.

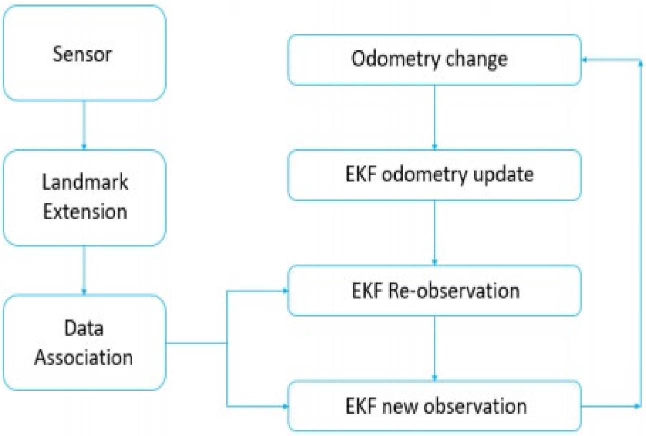


Figure 40 General architecture of SLAM

The basic steps involved in any SLAM architecture are mentioned as follows:

- Landmark Extraction - The algorithm will terminate only when the population of the particles converges the features which can be re-observed and distinguished from the environment. These are used by the robot to find out where it is (to localize itself).
- Data association - The problem of data association is that of matching observed landmarks from different scans with each other.

- State estimation - The Extended Kalman Filter is used to estimate the state (position) of the robot from odometry data and landmark observations
- State update.
- Landmark update.

Before a robot can answer the question of what the environment looks like given a set of observations, it needs to know from which locations these observations have been collected. At the same time, it is hard to estimate the current position of a vehicle without a map. A good map is needed for localization while an accurate pose estimate is needed to build a map.

Visual SLAM

Visual SLAM, or vSLAM, is the use of just visual sensors. In vSLAM, the camera's position is tracked by matching feature points or using the image's appearance, and then a 3D/2D map is constructed based on the requirement. However, because matching feature points and updating the map must be performed for all image frames, this method has the disadvantage of slow processing performance. As a result, vSLAM was improved by parallelizing the tracking and mapping process and leveraging keyframes to boost performance.

LiDAR SLAM

A LiDAR-based SLAM system maps a room using a laser sensor and an IMU, comparable to visual SLAM but with more accuracy. LiDAR uses numerous transceivers to illuminate a region and measure its distance from the robot's position. To determine position and distance, each transceiver rapidly produces pulsed light and detects the reflected pulses. Due to the speed at which light travels, very precise laser performance is required to track the exact distance between the robot and each object. Because of this necessity for precision, LiDAR is both a quick and accurate method.

APPENDIX E

What is path planning? What are its types?

One of the most basic activities required for robot navigation is path planning. As a result, once we've located our robot, we'll need to plan our path based on the occupancy grid or perceived surroundings. That is, once we have determined the destination position and localized our current pose, we must plan the robot's route to the destination. Path planning is the name for this type of planning. It can only be done if the global map is known in advance and optimized to the utmost extent possible; otherwise, it can be explored simultaneously while navigating. The robot's path must be clear of collisions. There are numerous path planning strategies to choose from, some predominantly used techniques will be explained here.

Node-based optimal algorithms

Dijkstra's Algorithm

Finds the shortest path in a graph where the weights of the edges are already known via dynamic programming using local path cost. When applying in 3D space, a 3D weighted graph must be built first; then it searches the whole graph to find the minimum cost path. It has two sets, one containing the vertices included in the shortest path tree and the other set with the ones not in the tree. The steps involved in this algorithm are:

- Creating a shortest-path tree set.
- Assigning a distance value to all vertices.
- Check the adjacent vertices of the lowest vertex.
- Pick the vertex with minimum distance.
- Update the distance value.
- Pick a vertex, not in the path tree and check its adjacent vertices.
- Repeat the steps until the path tree does include all vertices given.

A Algorithm*

A* algorithm is used to calculate the shortest distance between the source (initial state) and the destination (final state). A* algorithm has 3 parameters:

- g: the cost of moving from the initial cell to the current cell. It is the sum of all the cells that have been visited since leaving the first cell.
- h: it is the estimated cost of moving from the current cell to the final cell. The actual cost cannot be calculated until the final cell is reached. Hence, h is the estimated cost. We must make sure that there is never an overestimation of the cost.
- f: it is the sum of g and h.

Therefore, it can be said that

$$f = g + h$$

Thus, this algorithm works in the following way, it will calculate f value and find the smallest f valued cell and move to that particular cell. This process will be continued until it reaches the destination, that's the goal cell.

D Algorithm*

D* resembles A* but is dynamic (that is, its cost can change while traversing to reach the goal). It will assume that there is no obstacle in a particular grid and once it approaches the grid, it will dynamically adjust and add information to the map and if necessary, re-plan the entire path from the current coordinates. This process is repeated until the robot reaches its destination.

There are three types of D* algorithm namely original D*, Focused D*, and D* Lite. All these types are combinations of different path planning algorithms like A*. D* is an incremental search algorithm used for path planning. D* resembles A* but is dynamic (its cost can change in the traversing to reach the goal). That is, it will assume that there is no obstacle in a particular grid and once it approaches the grid, it will dynamically adjust and adds information to the map and if necessary, re-plan the entire path from the current coordinates. This process is repeated until the robot reaches its destination.

Theta Algorithm*

Theta* is an any-angle path planning algorithm that is based on the A* search algorithm. They search for a path in the space between two points and take a turn at any angle. The resulting path will be towards the goal with fewer turns. Algorithms like the A* will be limited only within the grids and thus will produce indirect and jagged paths. It interleaves smoothening with the search. There are many variants of the Theta* algorithm. Some of them are:

- Lazy Theta*
- Incremental Phi*

Sampling-based planning methods

Rapidly-Exploring Random Tree

The Rapidly-exploring Random Tree (RRT) is quite straightforward. Points are randomly generated and connected to the closest available node. Each time a vertex is created, a check must be made that the vertex lies outside of an obstacle. Furthermore, chaining the vertex to its closest neighbor must also avoid obstacles. The algorithm ends when a node is generated within the goal region or hits a limit.

Probabilistic Road Map

A new technique to deal with choosing points. Here, points are chosen randomly in the C-Space instead of uniform selection, assuming the structure of the free space from these random points is obtained. Thus, on every iteration, a new set of points are randomly assigned in the C-Space and returns 0 or 1, depending on if there is free space or collision. Thus, at every turn when it finds a free space, it will try to forge a new configuration and the closest existing sample. The problem with a probabilistic road map is that there can be a possibility that the robot might fail to find a path even if it exists. Thus, this kind of approach is not considered a complete path planning algorithm.

Bionic path planning algorithms

Ant Colony Optimization Algorithm

Ant Colony Optimization (ACO) is an intelligence-optimized algorithm that simulates the heuristic mechanism of the shortest route based on pheromone in the process of ants foraging for food. ACO uses all paths of the entire ant colony to describe the solution space of an optimization problem and obtains the best path through positive feedback based on the pheromone. The idea of ACO is that it directly maps the robot's path optimization problem, which is easy to understand and has very intuitive results.

Particle Swarm Optimization

It is optimized by continuously iterating to improve the candidate solution with regard to a given measure of quality. It will have some particles, which will be moving in the environment/ search space with a particular position and velocity. Each particle will be influenced by its local best-known position which is updated as better positions by other particles. This will move the swarm toward a better solution ultimately. It is metaheuristic as it makes few/ no assumptions about the problem optimized.

Genetic Algorithm

A genetic algorithm is a search heuristic that is inspired by Charles Darwin’s theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction to produce offspring for the next generation. The algorithm will terminate only when the population of the particles converges.

APPENDIX F

What is Visual Odometry? What are the phases of odometry?

Visual Odometry (VO) is a very vital part of autonomous navigation used with autonomous ground vehicles, aerial vehicles, and underwater or surface vehicles. Visual odometry is in layman's terms, determining the position information for accurate navigation of the robot using camera image sequences. Usage of encoders has slippage issues which in turn will reduce the accuracy of estimation. Thus, VO provides computationally cheaper and more accurate odometry results when compared to GPS, wheel odometry, sonar localization, or INS.

Feature Extraction

Features from Accelerated Segment Test

FAST stands for Features from Accelerated Segment Test. It is one of the fastest feature extraction techniques. It is best for a real-time application point of view with efficient computation. The working of FAST is as follows: Corner detection by drawing the Bresenham circle around the pixel p and labeling each of the circles from 1 to 16. Then the intensity of N random pixels is compared.

Multiple interest points are cast off using the non-maximum suppression which is based on the difference in distance between subsequent keypoints.

Scale-Invariance Feature Transform

SIFT stands for Scale-Invariance Feature Transform. Its working is as follows: First interest points are scaled and localized followed by blurring using the gaussian blur operation.

$$L = G(x, y, \sigma) \times \big(I(x, y)\big) \tag{56}$$

Comparing the interest points with neighboring octaves is done to determine keypoints based on the extrema values obtained. Rejection of edges and flat region happens next based on intensity measures.

$$Intensity \ < \ 0.03 \tag{57}$$

$$HM = \left(\frac{D_{XX}}{D_{xy}}, \frac{D_{xy}}{D_{yy}}\right), \frac{TR(HM)}{Det(HM)} = \frac{(R+1)^2}{R} \tag{58}$$

Then the orientation of the keypoints obtained is based on the high probable distribution of the orientation of every keypoint. Then the descriptors based on orientation and scale of the keypoint are assigned.

Speeded Up Robust Features

SURF stands for Speeded Up Robust Features. It is a robust and fast technique preferred for its fast computation of operators using box filters, thus enabling real-time applications such as tracking and object recognition. Consists of 2 steps- Feature Extraction and Feature Description. Feature extraction uses an integral image, which is a way of calculating the average intensity of pixels in a selected box.

$$Image\ Integral = \sum_{i=0}^{i \leq x_j = y} \sum_{j=0} I(i,j) \tag{59}$$

Surf uses the Hessian matrix because of its good performance in computation time and accuracy. Descriptors are assigned by first assigning a fixed orientation to each interest point, and then extracting the feature descriptor.

KAZE and AKAZE

The working of KAZE is as follows: Nonlinear scale-space extrema to detect the features accurately; Nonlinear diffusion filtering with Additive Operator Splitting (AOS) is used instead of gaussian blurring to keep the object boundaries intact.

$$c(x,y,t) = g\big(\nabla I_o(x,y,t)\big) \tag{60}$$

Orientation of the feature points is very similar to SIFT, but instead of aligning in a histogram, points are used in the vector space. Since the descriptors have to operate at a nonlinear scale-space, an altered version of the SURF descriptor is used here.

AKAZE was built over the KAZE algorithm. It uses an accelerated version of nonlinear scale-space called Fast Explicit Diffusion (FED). This change was made considering how computationally expensive the AOS process of KAZE was. Also, AKAZE uses a form of local difference binary (LDB) descriptors to increase the computation of the extraction process further.

Binary Robust Invariant Scalable Keypoints

BRISK stands for Binary Robust Invariant Scalable Keypoints. Similar to ORB, BRISK uses pyramidal image representation, wherein stable points are extracted using the adaptive corner detection operator. Unlike other algorithms, BRISK uses binary bitstring.

Feature Matching

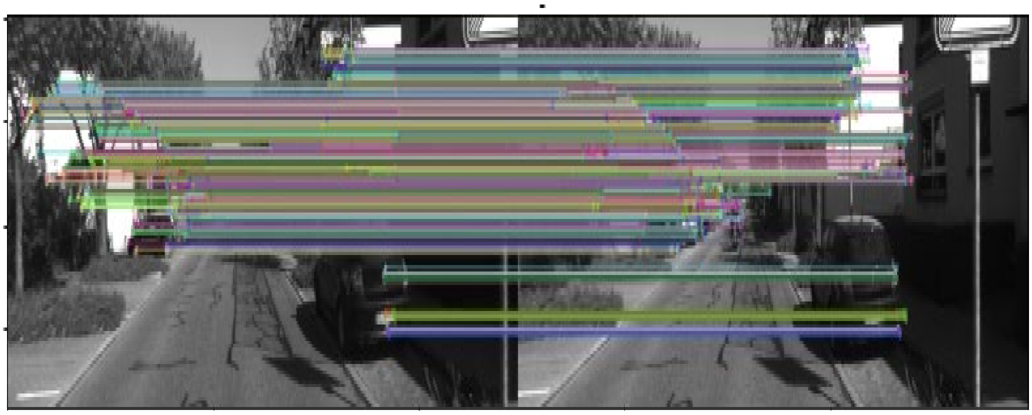


Figure 41 Matching using BF Matcher

Brute Force Matcher

It takes the descriptor of every feature of the first set and matches it with all the other features from the second set using distance calculation. It is time-consuming and matches all features. Also, Lowes's test is done after matching in general to remove outliers.

Fast Library for Approximate Nearest Neighbor

Fast Library for Approximate Nearest Neighbor or FLANN has a collection of optimized algorithms for searching the nearest neighbors in big datasets. It is faster and more accurate than BF Matcher. It has two dictionaries index and search parameters.

Outlier Removal

Random Sample Consensus

RANSAC stands for Random Sample Consensus. Deals with removing outliers from inliers contained in a data. No real-world sensor readings are perfect. Thus, we use RANSAC which is a simple trial and error method with groups the inliers and outliers separately. Thus, it helps us to throw away the outliers and work with inliers alone which will save our computation and time. It involves 4-step processing, and they are sampling, scoring, computing, and repeating.

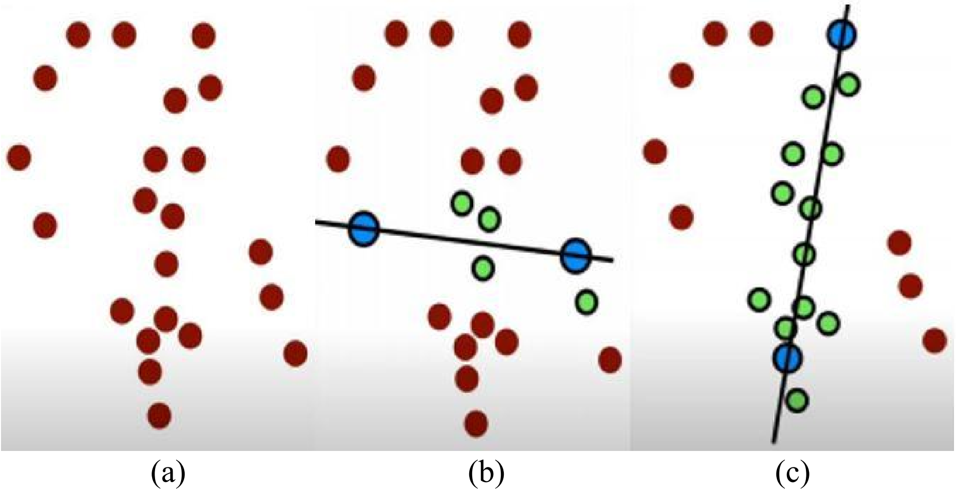


Figure 42 RANSAC Procedure

Let's assume there are 2D points (Figure 42 (a)) in which a line has to be fitted. Now we will sample and randomly take one line marked in Figure 42 (b) as our inlier. Now we will compute the number of supporting inliers satisfying the line we drew. This will be done by parallelly extending imaginary lines on both sides of the line assumed to be the inlier with a uniform distance of **δ.** Now every point lying inside these imaginary lines constitutes the scoring of the inliers. Now in Figure 42 (b) there are 4 inliers.

Now we will repeat the steps mentioned above repeatedly and the score will be overwritten with the best score after every iteration. In Figure 42 (c), that's after some iterations, we got a line that has 12 points satisfying that line model which is the best score to be obtained. Thus, it will be the best fit for the line. This way we can eliminate all the outliers easily.

Feature Tracking

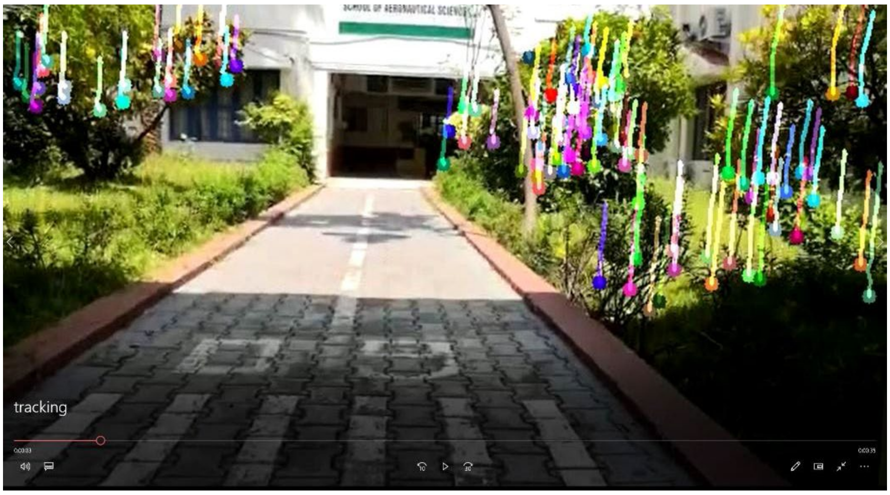


Figure 43 KLT Tracking

The apparent motion of pixels in a frame can be tracked using the Lucas-Kanade Optical Flow algorithm, which is a method for detection for feature tracking. It assumes that neighboring local pixels have small (can be considered constant) flow and solves the general optical flow equation using least square criteria (an error has a gaussian distribution of zero error). The image flow should always as follow:

$$I_x\ (q_1)V_x + I_y\ (q_1)V_y = \ -I_t\ (q_1) \tag{61}$$

Rotation and Translation

Using Mono Vision

For mono, we will first estimate the estimation matrix derived from the Fundamental matrix. Then we will use the recover pose to find the rotation and translational vectors. The fundamental matrix and Estimation matrix are used for determining the visual odometry along which RANSAC is used to estimate with minimal reprojection error.

Using Stereo Vision

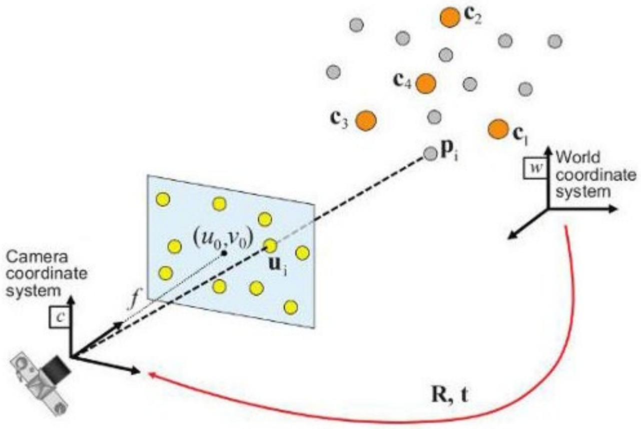


Figure 44 Camera Projection

In stereo, we will use a variant of PnP (that's Perceptive-n-Points). Some PnP variants include EPnP, MLPnP, and frame-PnP.

Some of the assumptions taken will estimate the pose of the robot using stereo vision are: Camera parameters are known. Image co-ordinate is known.

$$S\ x\ p_c = K\ x\ [R\ |\ T]\ x\ p_w \tag{62}$$

$$s[u\ v\ 1] = [f_x\ \gamma\ u_o\ f_y\ v_o\ u_o][r_{11}\ r_{12}\ r_{13}\ t_1\ r_{21}\ r_{22}\ r_{23}\ t_2\ r_{31}\ r_{32}\ r_{33}\ t_3] \tag{63}$$

The basic framework of PnP is as follows:

- Direct Linear Transform (DLT) finds the approximate value of Rv and Tv using the projection matrix.
- Levenberg-Marquardt Algorithm is a trial-and-error optimization step used to reduce the reprojection error thereby optimizing the estimated Rv and Tv values.

APPENDIX G

Proportional-Integral-Derivative (PID) Controller

Error is the difference between Process Variable (PV) and Set Point (SP).

$$Error\ (E) = PV - SP \tag{64}$$

$$P = E \tag{65}$$

$$I = I + E \tag{66}$$

$$D = Prev_{error} - E \tag{67}$$

$$PWM = k_p\ x\ P + k_i\ x\ I + k_d\ x\ D \tag{68}$$

k_p, k_i, k_d are the gain factors which will be tuned using various methods or trial and error methods. There is also a plotting tool exclusively called PlotJuggler which helps in plotting and tuning the gains in real-time using Gazebo.

Equations (64) to (68) indicate the basic equations for finding the PWM for the motors. PV means the current position resulting due to disturbances and SP is the desired position. Once the motor speeds are found, we will have to check the boundary conditions, that is *t* has to be greater than 0 and less than $2x$ depending on the bytes. So, after these conditions are taken into account, we can send the motor speeds from the controller to the ESC to the motors.

When a disturbance is made, the UAV before retaining its previous control command, the thrust will drift the UAV by a little every time it alters. Thus, we will be introducing a position controller here to solve this issue. Here instead of assuming the initial position angle references, we will use the position controllers to that for us and send it as the output to the PID loop of roll pitch and yaw mentioned earlier. We will be using the four sensors discussed in the first sheet for the position controller.

PID tuning is done here as follows:

- First, we will tune the outer loops also termed the cascade loops. For tuning, we will take all the gain terms to have a constant 0 assigned by default.
- Then, we will slowly give a range of values, so let's take the P controller of Yaw.

- So, we can tune the P terms gain and then add D and I to stabilize it properly for Yaw in the cascade loop control.
- Then, we can move to the Pitch and Roll in order and assign the values using trial and error.
- Then, once all the four PID controllers in the cascade loop are tuned, we can proceed with the controllers in the position controller following the same rules as mentioned in the above point.

In MATLAB, we can first draft the plant (the PID controller block in our case) and give values to the gain terms. We can then go to the PID tuning graph and adjust the response time and check if the desired kind of graph is plotted. Once done, check the proposed gain values at the bottom of the tuning tool and put its value in the plant done earlier. Now, check it in real-time and adjust accordingly.