SPYDR: (UAV Ground Station For Intruder Detection And Surveillance)

By

NC Mahrez Imran

NC Ommar Aftab Hashmi

NC Usman Zubair

Supervisor

Dr. Malik Muhammad Zaki Murtaza Khan

Co-supervisor:

Dr. Saddaf Rubab

Submitted to the Faculty of Computer Software Engineering Department,

Military College of Signals, National University of Sciences and Technology, Islamabad

In partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering

JULY 2020

CERTIFICATE OF APPROVAL AND CORRECTNESS

It is to officially state that thesis work contained in this report "SPYDR: A UAV ground station for Intruder detection and Surveillance" is carried out by NC Mahrez Imran, NC Ommar Aftab Hashmi and NC Usman Zubair under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Computer Software Engineering from Military College of Signals, National University of Sciences and Technology (NUST). And is original with 14% plagiarism.

Approved by:

Signature:_____

Supervisor: Asst. Prof Dr. Malik Muhammad Zaki Mutaza Khan
MCS, Rawalpindi

Signature:_____

Co-Supervisor: Asst. Prof Dr. Saddaf Rubab
MCS, Rawalpindi

DECLARATION OF ORIGINALITY

We hereby declare that the work contained in this report and the intellectual content of this report are the product of this work. This thesis has not been formerly published in any structure not it does include any verbatim of the published resources which could be treated as violation of the international copyright decree. We also affirm that we do recognize the term ‘plagiarism’ and ‘copyright’ and that in case of any copyright infringement or plagiarism established in this thesis, we will be held fully accountable of the consequences of any such violation.

ABSTRACT

In this era of high threat, crime, and security breach concerns, one must approach surveillance from perspectives that are not confined to static circuit cameras. We need mobility, range and artificial intelligence to not only automatically detect threats without human intervention from different angles and altitudes, but also generate alerts.

In the thesis, we propose a ground station for a quadcopter that can reach altitudes, landscapes and positions human beings cannot easily access and detect threats or more specifically intruders from these areas and alert the authorities on finding anyone suspicious. SPYDR uses a State-of-The-Art Object detection and classification algorithm that has been trained to detect unauthorized presence of individuals in places under high alert or lockdown. Not only will it catch our trespasser but also generates an alarm to notify concerned parties.

ACKNOWLEDGEMENTS

We are humbly grateful and indebted to our supervisor, Dr. Muhammad Zaki Murtaza and co-supervisor, Dr. Sadaf Rubab for their sincere support in research related tasks, motivation, patience, and immense knowledge. Their guidance helped us in completing our project and this thesis on time. We would not have imagined a better supervisor and co-supervisor for our bachelor's thesis.

We also would like to thank the rest of the FYP committee and colleagues for their insights and encouragement and for their thought-provoking questions during defenses that allowed us to approach our problem from various perspectives and develop a scope that was in the best interest of team.

Last but not the least, we would like to thank our parents, brothers and sisters who supported us spiritually throughout the project timeline, thesis, and life in general. Alhamdullillah.

Table of Contents

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Drones for surveillance is the future of CCTV. Mobility, range and altitude are the key attributes that make drones such as quad copters stand out and encourage their use for monitoring and surveillance of areas where circuit cameras can't reach, or human eye fails to perceive. Considering the heightened concerns regarding scrutiny of critical infrastructure, borders or hostile demonstration environments, drones can keep a close eye for intrusions. Combined with AI, intruders can be detected from the live stream received from the drone automatically, making it highly effective for security systems.

For our BS Software Engineering final year project, our team designed a system that performs live threat/Intruder detection on the stream received from a manually controlled quadcopter during flight. The idea is to build a software that develops a udp connection with a quadcopter (3DR SOLO), fetches the live stream from the Go Pro Hero camera attached to the UAV, divides the footage into frames which are processed to detect and classify intruders or threats. So, the question is, what classifies as an intruder/threat? A regular pedestrian classifies as one at the moment, but the sky is the limit. Vehicles and weapons are an additional aim that can be achieved.

We are employing Tensorflow's Object detection API and Single Shot detector as a model. It is approximately 74 to 76% accurate with an exceptional real-time performance. With transfer learning using our own data collected in the university premises, we will further train SSD v1 mobilenet which is pre-trained on COCO dataset. The stream received will be divided into frames using python's open cv and the model will individually process each frame for detecting and classifying intruders by drawing bounding boxes around them which will then trigger an alarm.

Figure 1 Data Flow from UAV to Video Feed

The quadcopter we are using is called the 3DR SOLO from 3DR Robotics which features various aerial imaging-specific flight modes. It provides flight stability functions and autonomous flight options thanks to twin computer flight control system enabling us to

explore its potential to the fullest. It uses a GPS for navigation and additionally, gyros, accelerators and other sensors to keep the aircraft safe from obstructions. The SOLO is designed to be paired with GOPRO action camera 3+ and 4. With a battery life of 20-25 minutes, we can travel along a planned route or the circumference of a designated area under high alert and scan it for potential trespassing.

CHAPTER 2. LITERATURE REVIEW

Quadcopter surveillance is a subject under research for a long time but due to payload constraints, practical implementation was difficult. However, with low cost embedded technology, autonomous systems are easily accessible now. Not only can we build a low budget quadcopter with efficient components, we can invest in one with an embedded computer, GPS, sensors all in one. An IP camera attached to the frame along with a GPS receiver can help with object detection and localization. The data is sent repeatedly to a ground station computer via a wireless link and the images are captured, analyzed based on color to detect the presence of object [1]. Other than color recognition, pixels are also a subject of significant discussion. Object detection can be done using deep learning which is chiefly comprised of the Artificial Neural Networks (ANNs). Neural networks are normally made from a layer of interconnected neurons. In the first layer, called the information layer, every neuron relates to an information include generally a pixel. The subsequent layer neurons' data sources are the first layer neurons, third layer neurons' information sources are the second layer neuron. Preparing a neural system implies choosing the best loads for all the neuron associations. The weights are found out utilizing a calculation called back-propagation. Layers that have been utilized in deep learning include shrouded layers of an artificial neural network and sets of propositional formulae. [2]

In case of resource restrained devices such as mobile phones, a framework called the MobileNet [3] has been established for object detection. Combined with the Single Shot Detection (SSD) Framework [4], we arrive at a fast, efficient method for detecting objects. A major contribution of SSD is using default boxes of different scales on different output layers. The SSD-MobileNet was first trained on the COCO dataset and then was fine-tuned on PASCAL VOC reaching 72.7% mAP (mean average precision). Using a manually controlled quadcopter such as the Parrot AR, images are received via wireless link and the desired object can be detected [5]. Faster RCNNs [6] present an Region Proposal Network (RPN) that shares full-picture convolutional features with the detection, thus enabling almost free recommendations or YOLO (You Only Look Once) [8], which uses object detection as a regression problem that a single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation, detection algorithms and KCF tracking algorithms [7], which uses Discrete Fourier Transform to diagonalize data matrix, which is then processed to train a discriminative classifier through linear regression and kernel filter ,can aid in an inexpensive approach that can run at a higher frame rate. A system as such can be embedded on quadcopter hardware and can carry out real-time object detection. Faster RCNNs being accurate but slow and YOLO being faster but less accurate as detection algorithms and KCF being the tracking algorithm can achieve satisfactory results as well in real-time systems both on on-ground GPU systems and embedded GPU [9].

CNNs are infamous for being computationally extensive. They require high powered Graphic Processing Units either on-ground or on-board if we want to deploy them for quadcopter surveillance. Lightweight UAV's cannot handle the weight of a GPU; the overall cost also increases. One effective solution is to move the computation to a remote cloud [10]. Cloud computing permits on-request access to boundless computational resources, which is particularly helpful for blasted overwhelming computational remaining tasks at hand that intermittently require huge calculations.

CHAPTER 3. DESIGN AND DEVELOPMENT

3.1 PRODUCT FUNCTIONS

The main features of SPYDR are highlighted below:

- Live video feed is transmitted from quadcopter to computer,
- On-the-fly detection and identification of objects from the camera feed,
- Detected and Identified objects are highlighted on the screen (in modified video).

3.2 SYSTEM DESIGN DIAGRAM

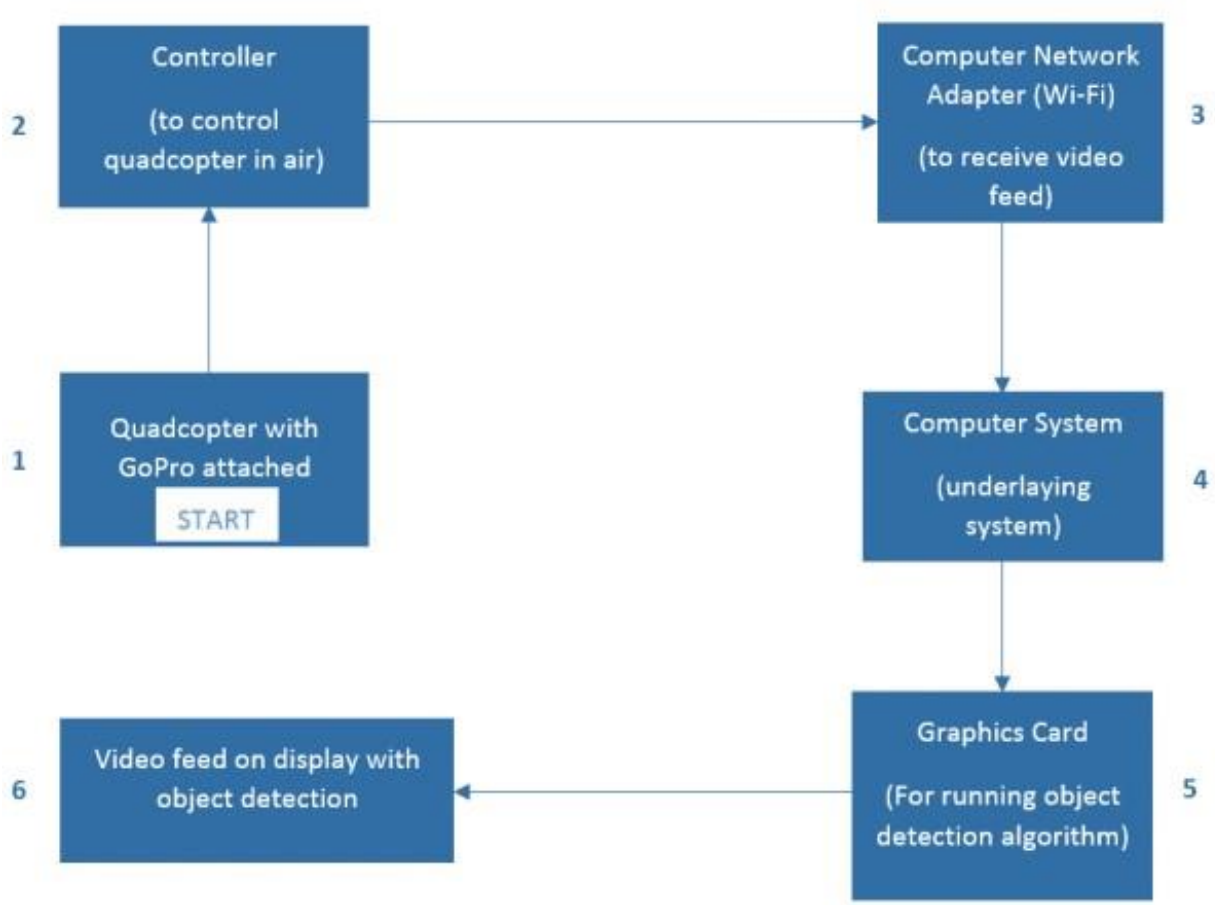


Figure 2. System Design Diagram

3.2.1 SYSTEM DESIGN DIAGRAM DESCRIPTION

The diagram above shows the steps that will take place to detect objects (particularly humans) while the live feed is constantly being streamed to the computer that is connected to the controller via. Wi-Fi. The video feed from the quadcopter's GoPro will be displayed onto the computer screen after the processing of object detection algorithm on video frames.

3.3 SINGLE SHOT MULTIBOX DETECTOR (SSD)

Our methodology, named SSD, discretizes the space of bouncing boxes into a lot of default boxes over various viewpoint proportions and scales per include map area. At

forecast time, the system creates scores for the nearness of each item class in each default box and delivers adjustments to the bounding-box to match the objects size.

Also, the system combines from numerous component maps with different resolutions to deal with objects of different sizes. Our SSD model is straightforward comparative with techniques that require object recommendations since it disposes of proposition age and resulting pixel or highlight resampling stage and embodies all calculation in a singular system. This makes SSD easy-to-train and integrate into systems that require an object-detection component.

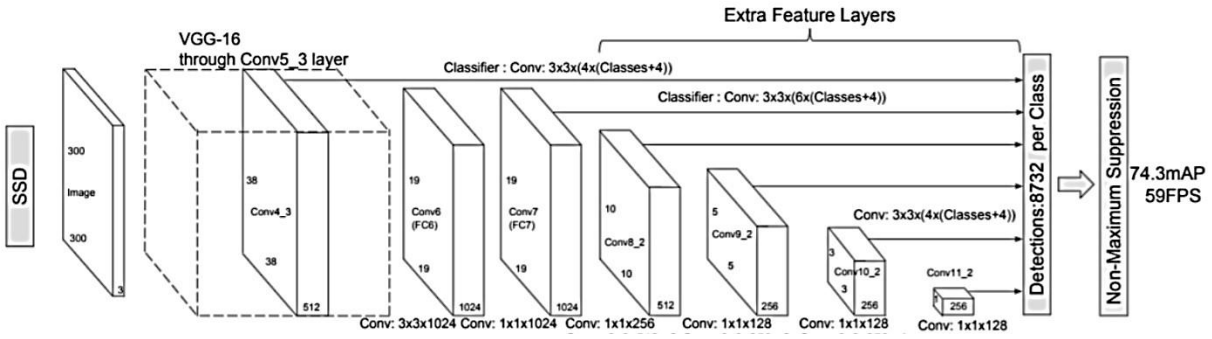


Figure 3. SSD Detection Steps

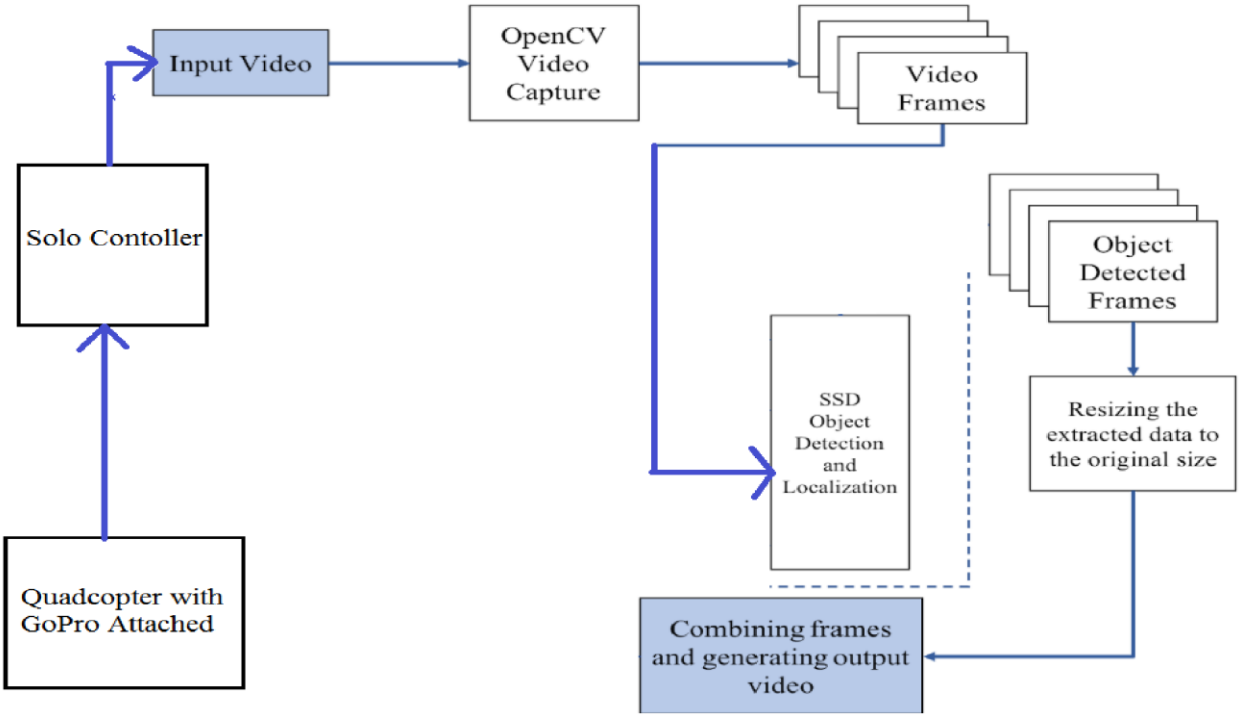


Figure 4.Object Detection on Frames

3.3.1 TENSORFLOW OBJECT DETECTION API – TRANSFER LEARNING

A custom model was trained for this project's purpose using a pretrained SSD MobileNet V2 model. The model was trained on google colaboratory for its GPU services.

Instead of training from scratch, this project's model will be built on an existing model and will be fine-tuned to detect intruders from a height.

3.3.1.1 Gathering Data

Before beginning making the object-detector, data is required which is used for training our model.

To prepare a vigorous classifier, a great deal of recordings/pictures are required which ought to contrast a ton from one another. In this way, they should have various backgrounds, arbitrary objects, and varied lighting conditions.

3.3.1.2 Labeling Data

Now after collecting enough videos/images, data is moved of around 80 percent of the recordings/pictures into the object_detection/pictures/train index and the other 20 percent in the object_detection/pictures/test registry.

To label data, a sort of labeling software is required. For this, "Labeling" is an extraordinary tool for naming.

3.3.1.3 Creating Label Map

TensorFlow requires a label map, which to be specific guides every one of the owned names to an integer. This label-map is utilized both by the train and detection process. A case of label map is as beneath:

```
item {
    id: 1
    name: 'person'
}

item {
    id: 2
    name: 'car'
}
```

Figure 5. Label Map

3.3.1.4 Creating TensorFlow Records

Now that we have generated our annotations and split our dataset into the desired training and testing subsets, it is time to convert our annotations into the so called TFRecord format.

For this, the *.csv files of each dataset are converted to *.record files (TFRecord format).

3.3.1.5 Configuring Training Pipeline

Sample pipeline configuration is downloaded for the specific model for transfer learning. For this project, we will be using ssd_mobilenet_v2_coco.cofig file.

3.3.1.6 Model Training

After downloading SSD from tensorflow's model zoo, we will use transfer learning and re-train it on colaboratory GPU.

3.3.1.7 Model Training Evaluation

Evaluation will be running in parallel to model training. An evaluation script will run to check the train directory for progress and evaluates based on the most recent checkpoint.

3.3.1.8 Training Stoppage Criteria

Based on the graphs output by Tensorboard, it will be decided to stop or to continue the training process. Usually, it will be stopped when the loss function is tapering off and no longer decreasing by a significant amount.

Image on the next page illustrates how a pre-trained model is trained on the required dataset and then used for object detection:

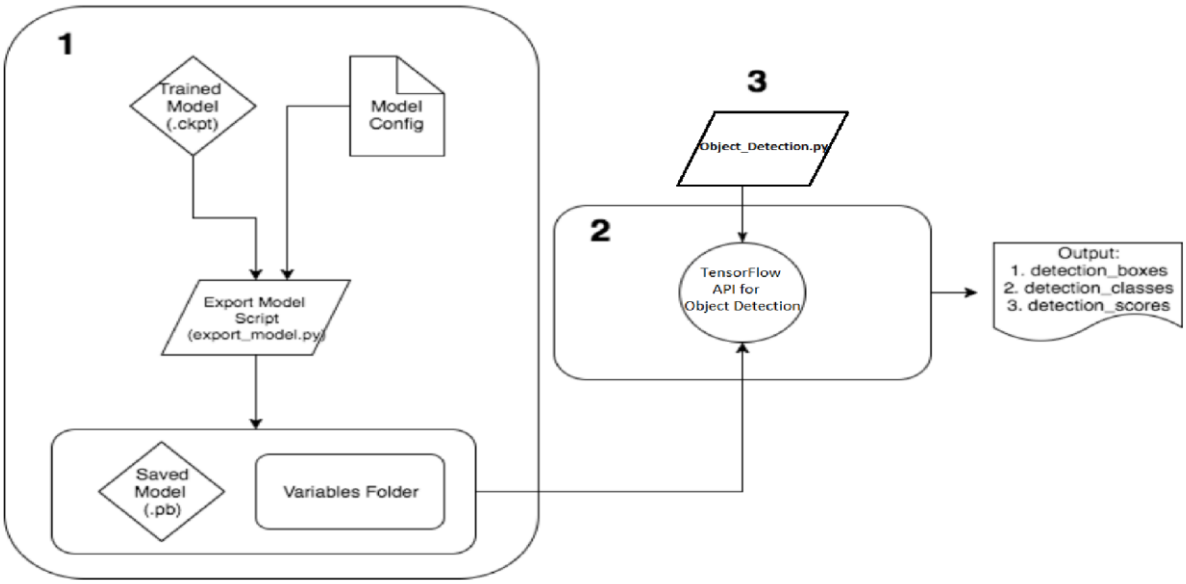


Figure 6. Transfer learning applied for Object Detection

3.4 SYSTEM ARCHITECTURE

3.4.1 ARCHITECTURAL DESIGN

Architectural design is about seeing how a system ought to be organized and designing the structure of system. It is the connection among design and software requirements engineering as it shows the main structural parts in a system and the connections between them.

1. Quadcopter
2. Quadcopter Remote Control
3. Computer

3.4.2 DATA FLOW DIAGRAM

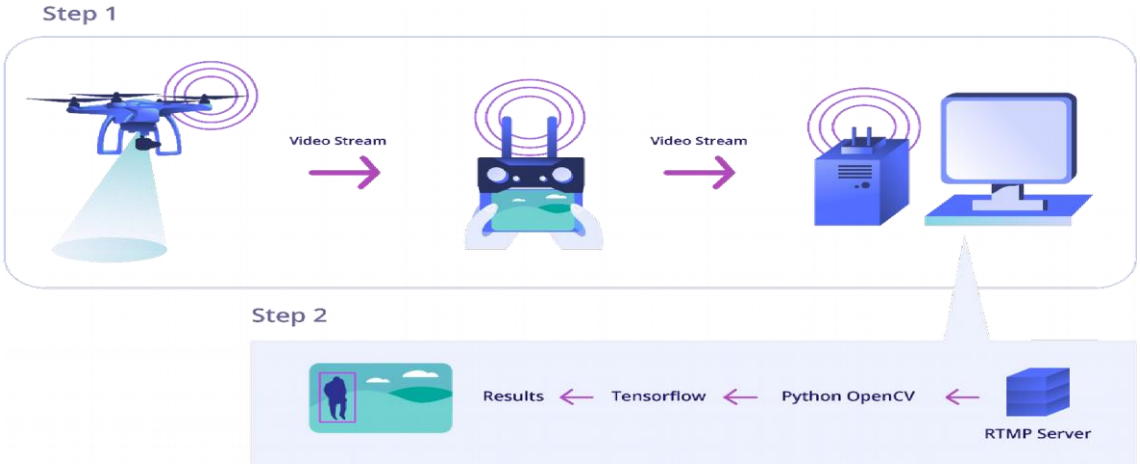


Figure 7. Data Flow Diagram

3.4.2.1 Data Flow Diagram Description

The live feed from the quadcopter is transmitted to controller, which transmits the stream to the computer. The controller is equipped with hotspot served by an access point used to connect computer to controller using Wi-Fi technology.

The live stream is fed into the RTMP server on the computer. This live stream is received onto OpenCV Python library for real-time computer vision, which transmits to TensorFlow, a math library for ML applications such as neural networks.

This computes the live feed frame by frame, showing classification, localization, and detection of objects.

The classification, localization and detection are illustrated in the picture on the next page:

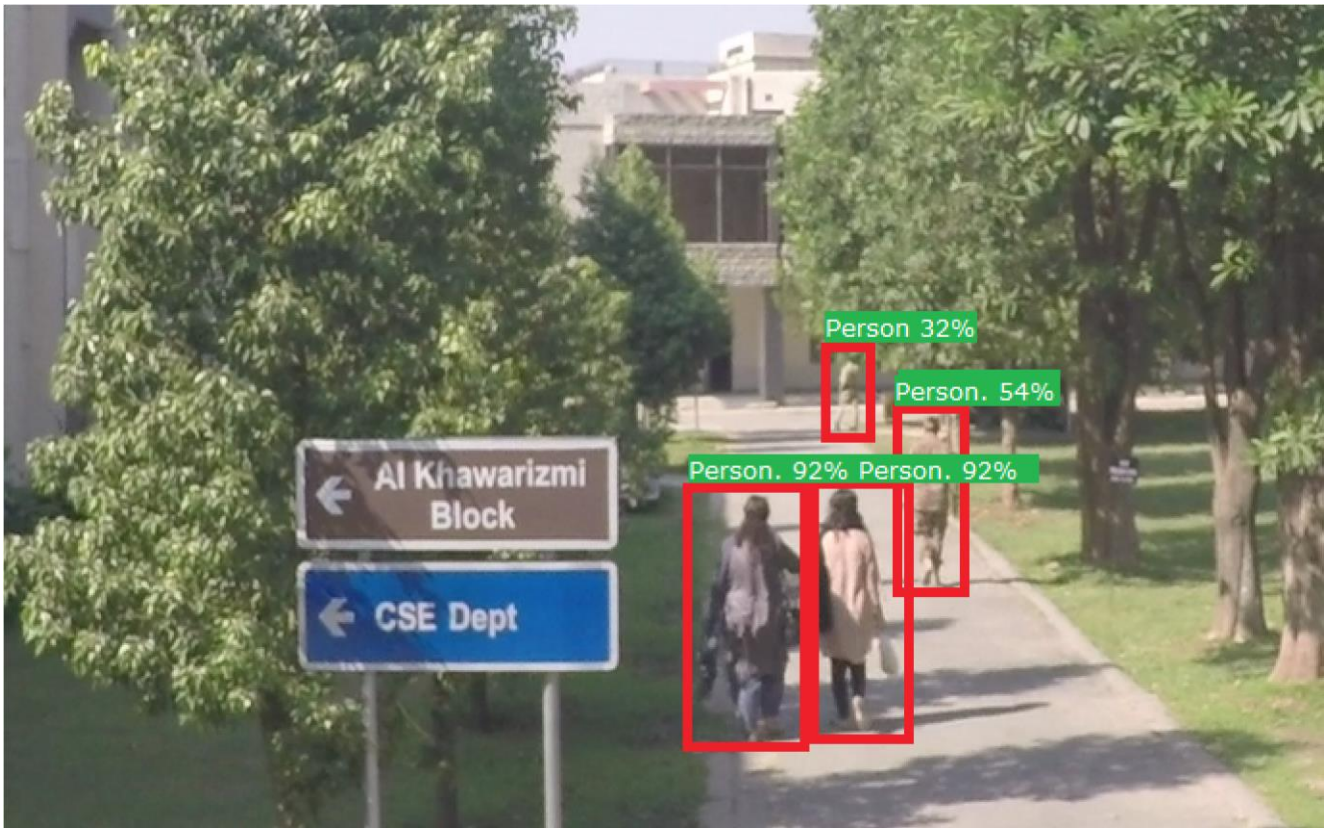


Figure 8. Classification, Localization, Detection

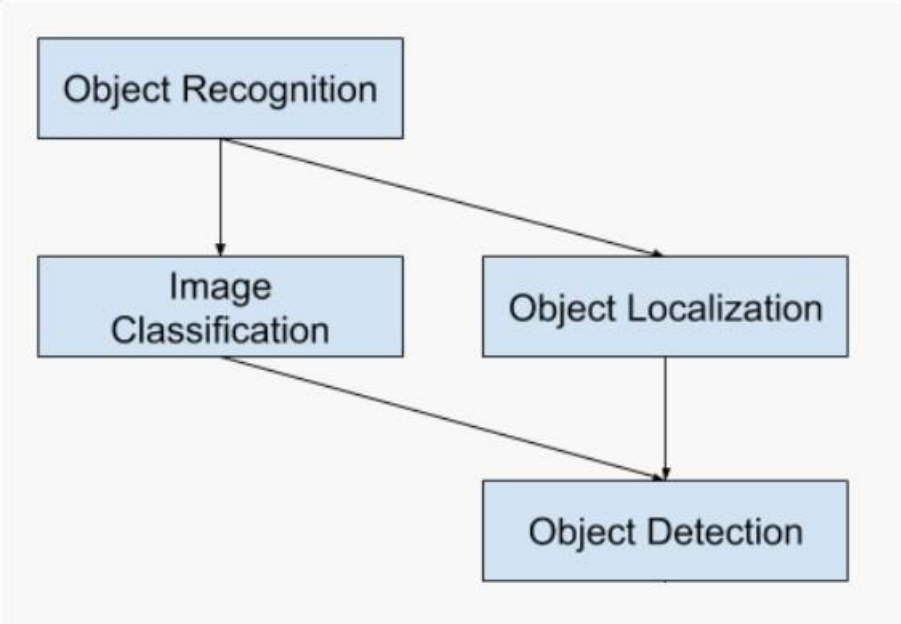


Figure 9. Steps for Object Detection

The computer vision tasks are categorized into a few simple procedures.

1 **Image Classification** - Image classification is among the most widely recognized PC vision issues. A calculation takes a gander at a picture and arranges it as an object. Picture classification performs numerous tasks, like confront detection to detection of malignant growth in medication.

2 **Object classification and localization** - The object localization calculations would not just assistance to know the nearness of an object, yet in addition the area of the object. A jumping box is drawn around the object in the picture.

3 **Multiple object detection and localization** - There could be different objects in the picture, and this is something that would be extremely basic in self-driving vehicles. The calculation would need to distinguish different vehicles as well as bikes, walkers, trees, and different objects. With regards to the setting of profound learning, the fundamental algorithmic distinction would pick the significant information sources and yields.

3.4.2.2 Stream Quality and Glitches

The video quality is set to 720p with 25FPS from GoPro mounted on the quadcopter. However, due to technical limitations and not up to the mark processor, the stream is being fetched at 5-7fps. The loss of FPS is due to a wide range of factors:

- Small buffer size of network card,
- Low clock speed of GPU,
- Mobile processor
- Low GPU dedicated graphics memory.

TensorFlow is subject to use all dedicated GPU memory when it is running it is Object Detection API. The stream is fetched continuously from the quadcopter. But due to the bottleneck in the areas above, the buffer upon getting full drops frames and the video lags with grey pattern of jagged frames.

3.4.2.3 Tradeof

The plan is to drop frames when the video is being fetched so that buffer is not filled, and it will take less time to perform object detection algorithm on the video stream. This will, however, have an impact on video quality and bitrate of the stream.

3.5 DATA DESIGN

Since project SPYDR is a real-time surveillance project, databases are not a major part of the system. There is going to be one database linked with the sign-up and login to store user information and to check whether the user that wishes to log in has been previously registered or not. The entity relationship diagram of this database is illustrated below:

The overall system, however, will use an object-oriented approach in which each module is independent of the others. To further elaborate the claim above, a class diagram is demonstrated below that gives a high-level view of the software in making:

Figure 10. Relational DB for User Credentials and Profile

3.5.1 CLASS DIAGRAM

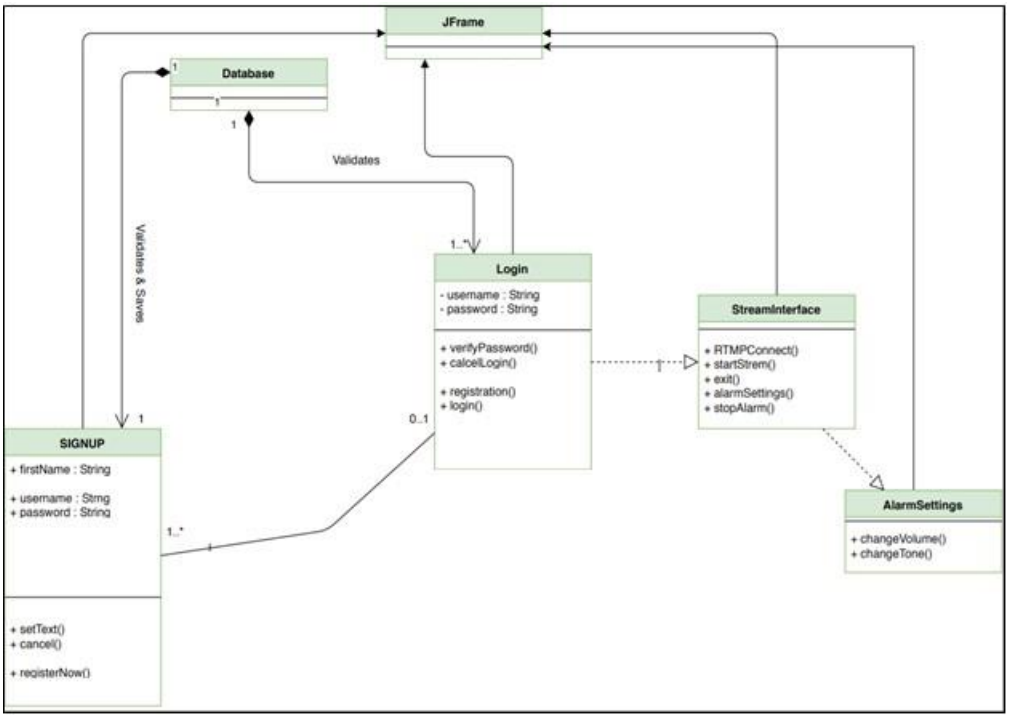


Figure 11. Class Diagram

3.5.1.1 Class Diagram Description

Class outline is a kind of static-structure diagram that depicts the structure of a system by indicating the system's classes, operations, attributes, and the relationships between them.

Classes in our system that includes:

AlarmSettings: Change alarm settings like tone and volume.

Database: Contains users profile. (user info, password, etc.), only accessible to administrator.

JFrame: Container for GUI.

Login: Verify and grants access to a registered user.

Signup: Registers a new user and makes a new row entry in the database.

StreamInterface: Responsible for displaying live stream to the logged in user, change alarm settings like volume and tone and exit out of the stream.

3.5.2 SEQUENCE DIAGRAM

3.5.2.1 Login Sequence Diagram

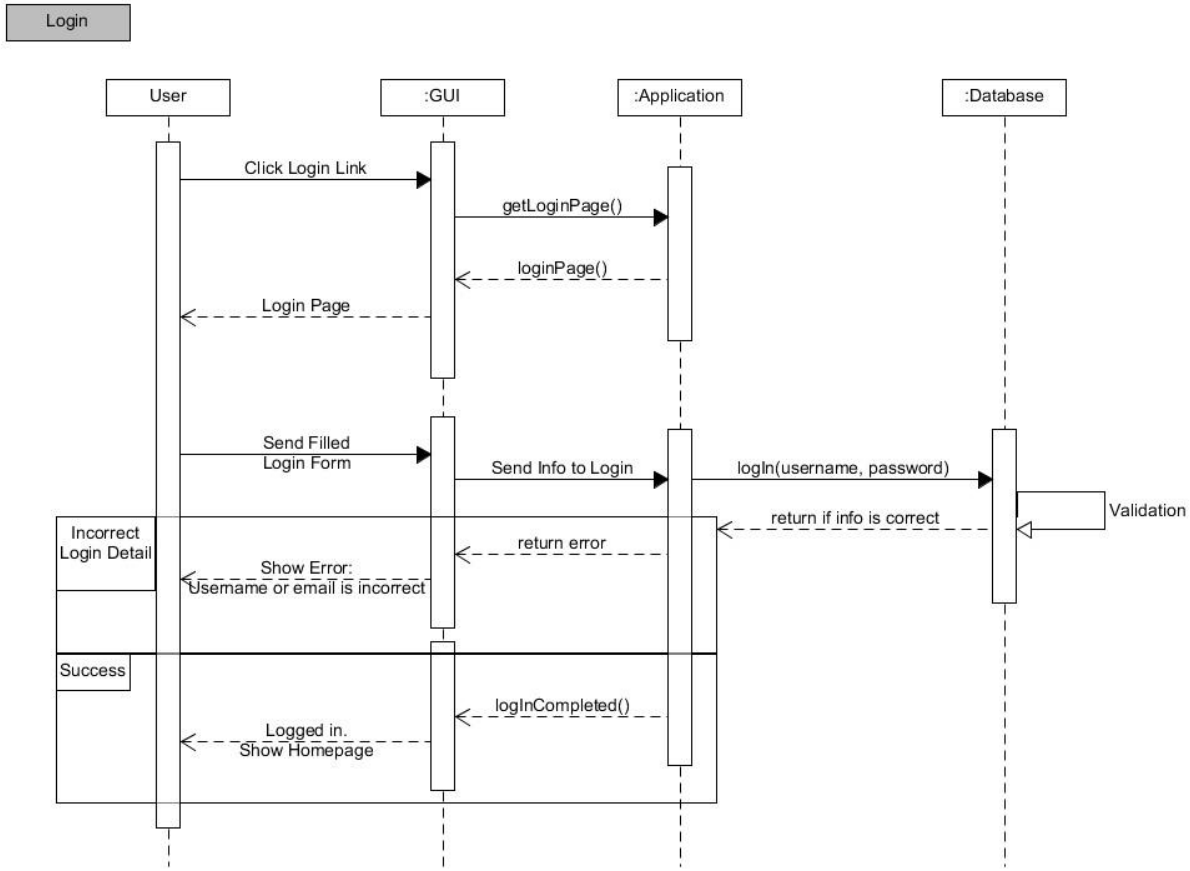


Figure 12. Login Sequence Diagram

Sequence Diagram Name: Log In

Actor: Members (Registered Users)

Preconditions: The user must have an account.

Action Steps:

1. Member opens the application, provides his/her credentials.
2. Member clicks on 'Log In' button.
3. The system checks the member's information and logs him/her to the homepage.

Post-conditions:

- If the data entered is valid, the system will enable the user into the system.
- If the data entered is invalid, the system will not let the user in and shall warn the user about the invalid information.

3.5.2.2 Signup Sequence Diagram

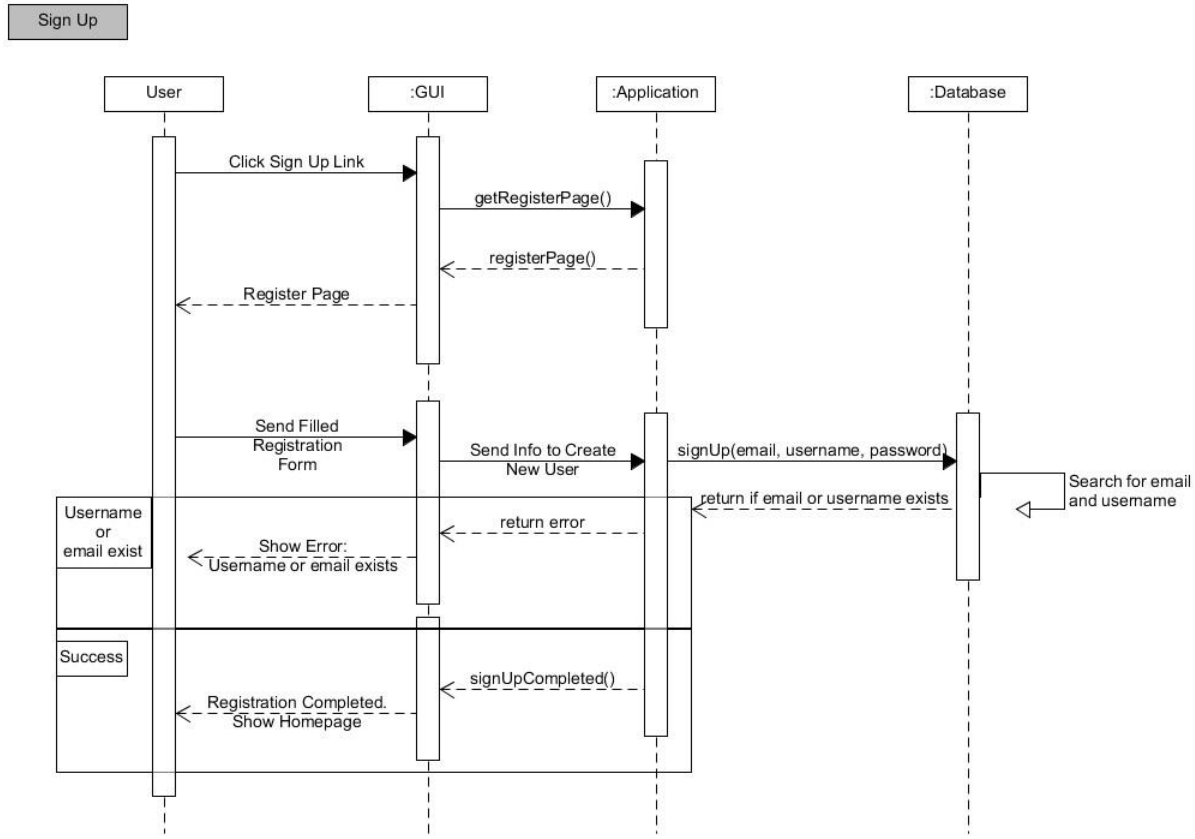


Figure 13. Signup Sequence Diagram

Sequence Diagram Name: Sign Up

Actor: Guest Users

Preconditions: None

Action Steps:

1. Guest opens application and clicks on Sign Up button.
2. Guest chooses a unique username for the account and fills relevant information.
3. Guest provides a password and a valid email address for the account.
4. Guest completes his/her registration and now can login as registered user.

Post-conditions:

- The guest user is a *member* (registered user) now.
- He/she shall be able to log in.

3.5.2.3 Establish Connection Sequence Diagram

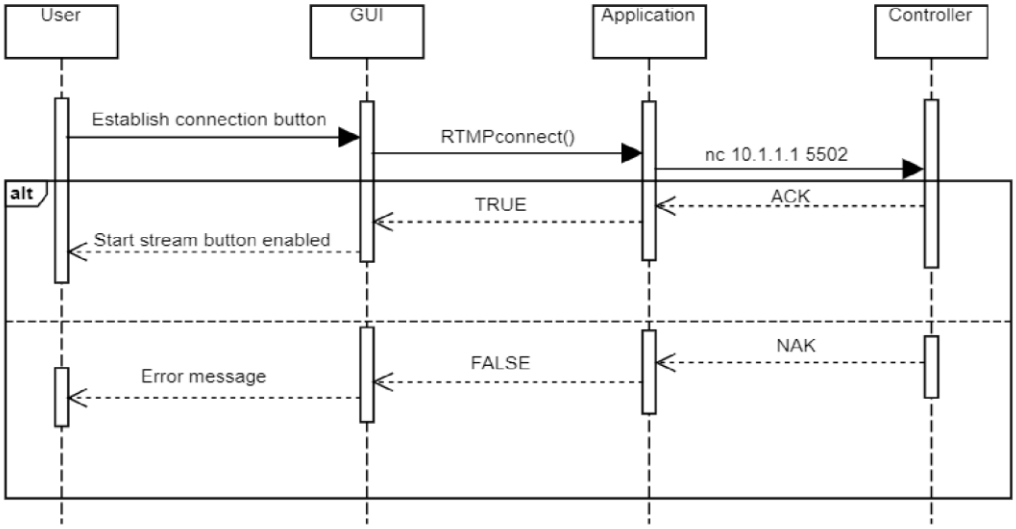


Figure 14. Establish Connection Sequence Diagram

Sequence diagram name: Establish Connection

Actor: Logged in User **Preconditions:**

- The quadcopter's motors are turned on or it is in flight mode.
- The controller is powered on.
- The camera mounted on the quadcopter is powered on.
- The system is connected to the controller's Wi-Fi.
- The quad-copter's battery is above 15%.

Postconditions:

- Start stream button is enabled.
- User can press the button to initiate the object detection module to receive processed frame on the video stream interface.

3.5.2.4 Start Stream Sequence Diagram

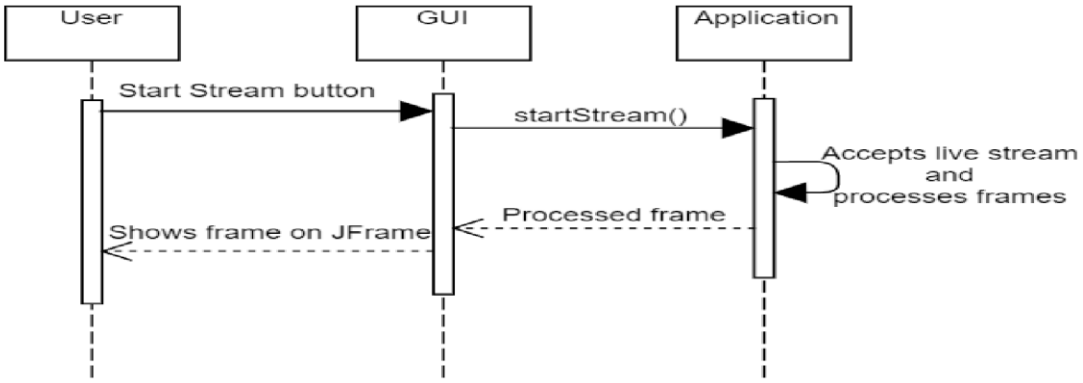


Figure 15. Start Stream Sequence Diagram

Sequence diagram name: Start Stream

Actor: Logged in User **Preconditions:**

- Connection with the quadcopter's controller is established.

Post conditions:

- Frames from the live stream after passed through the trained object detection model are displayed on the video stream interface with the required detection.

3.5.3 DEPLOYMENT DIAGRAM

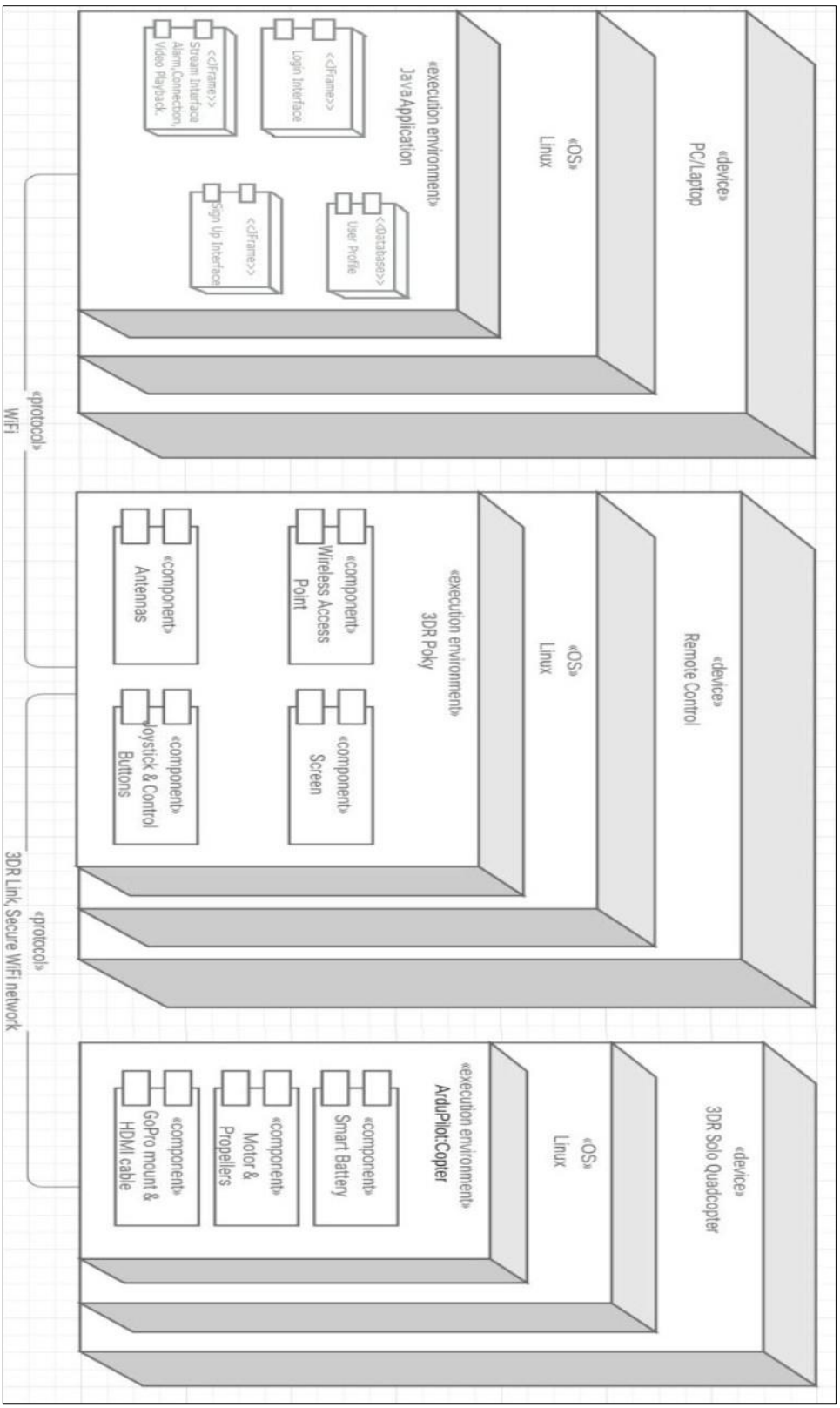


Figure 16. Deployment Diagram

3.5.3.1 Deployment Diagram Description

Deployment diagram shows the arrangement of run time nodes and components. The diagram will be utilized in displaying the physical parts of our OO system.

Deployment diagram portrays a static perspective on the run-time setup of preparing components and the segments that sudden spike in demand for those components. In other words, our sending charts show the equipment for the system, the product that is introduced on that equipment, and the middleware used to connect machines to each other.

Connection between components are represented with lines and are assigned stereotypes, for example, Wi-Fi and 3DR Link to shows the kind of connection. 3.5.4

3.5.4 USE CASE DIAGRAM

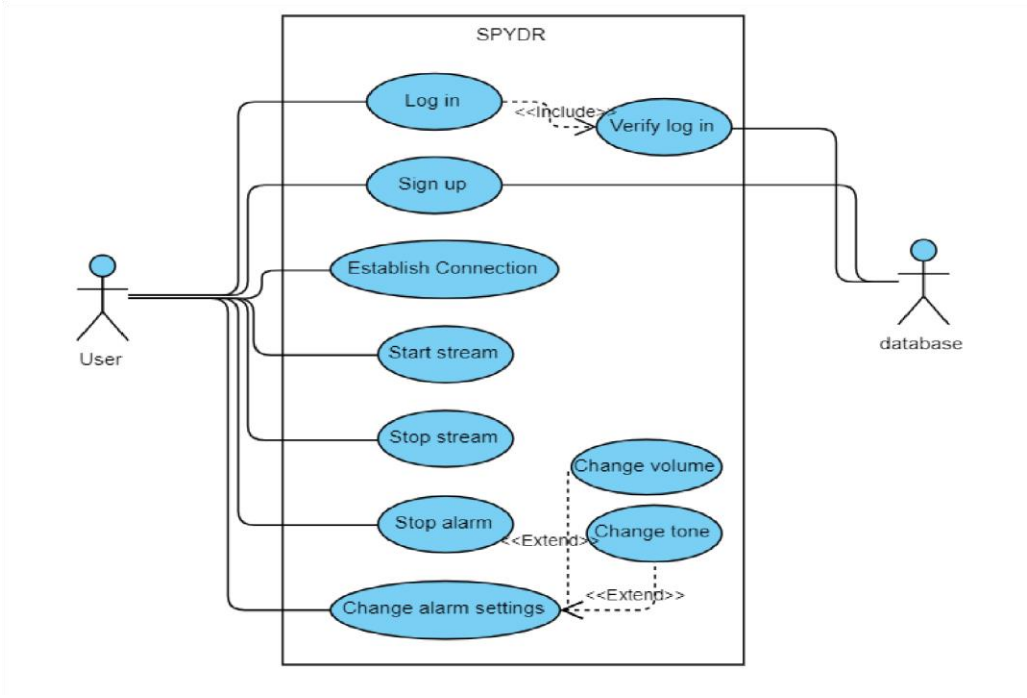


Figure 17. Use Case Diagram

Log in: User enters the credentials to access stream interface.

Verify Log in: The database check for pre-registered user.

Establish Connection: The user establishes connection with RTMP.

Start Stream: The user starts receiving livestream from the UAV.

Stop Stream: The user cuts off livestream.

Stop alarm: The users stop the alarm (only if the alarm is triggered).

Change alarm settings: The user can settings of the alarm such as volume and tone.

USE-CASES:

1. Video Acquisition

a. Description

This feature enables the system to acquire the live stream from the camera mounted on the UAV.

b. Stimulus/Response Sequences:

Normal path: The video is acquired by the system from the UAV that is surveilling a defined area.
Preconditions: The camera mounted on the UAV captures video in real-time.
Interactions: The captured video is sent to the system for processing.
Post conditions: Captured video is divided into frames for processing.
Categorization: • Criticality: High • Probability of Defects: High • Risk: High

c. Functional requirements

The system shall acquire the live stream from the UAV.

2. Video Processing

a. Description:

The incoming live stream will be divided into frames, each of which will be processed and tested for person detection separately.

b. Stimulus/Response Sequences

Normal Path: Successfully divided into frames
Preconditions: The video is sent to the system for processing
Interaction: Frames are generated from the video feed.
Post conditions: Frames are used for person detection.
Categorization: • Criticality: High • Probability of Defects: Medium

• Risk: Medium

c. Functional requirements

The system shall effectively divide the live stream into frames.

3. Person detection

a. Description

The object detection model deployed will detect the presence of people inside the frame.

b. Stimulus/Response Sequences

Normal path: People are not detected in the footage.
Preconditions: The frame does not show any people inside.
Interactions: Model finds no people to be detected in the frame.
Post conditions: The system continues to function without breaking down.
Categorization: • Criticality: High • Probability of Defects: Medium • Risk: Medium

Alternative path: People are detected in the footage.
Preconditions: Live feed is divided into frames for individual processing.
Interactions: Model recognizes people in the frame.
Post conditions: Every human in the frame is classified as a person and bounding boxes are drawn around each of them along with the class name ‘person’.
Categorization: • Criticality: High • Probability of Defects: Medium • Risk: Medium

c. Functional Requirements

The system shall use an object detection model to predict the presence of people inside the frame.

4. Person classification

a. Description

The model will classify the detected objects as humans.

b. Stimulus/Response Sequences:

Normal path: People that are detected in the footage are classified as persons.
Preconditions: The model detects people.
Interactions: Every human in the frame is classified as a person. A new frame is also displayed on a window with bounding boxes drawn around each of the detected people along with the class name ‘person’.
Post conditions: An Alarm is generated on the presence of people inside a restricted area.
Categorization: • Criticality: High • Probability of Defects: Medium • Risk: Medium

c. Functional Requirements

The system shall be able to calculate the probability of the detected object being a person and if it is higher than 0.4, it will classify it as a person and draw a bounding around it with the class name in a new frame. This new frame will be displayed on a window.

5. Alarm generation:

a. Description

An alarm will ring as soon as people enter the restricted area.

b. Stimulus/Response Sequences

Normal path: No alarm generated.
Preconditions: No people inside the frame.
Post conditions: The system will continue functioning normally.
Categorization: • Criticality: High • Probability of Defects: Medium • Risk: Medium

c. Functional Requirements

The system shall raise an alarm on detecting people which can only be turned off manually by the user.

6. Authorized access:

a. Description

The user can only use the SPYDR. System after logging in.

b. Stimulus/Response Sequences

Normal path: User gets access to the system.
Preconditions: Correct username and passwords are entered.
Interactions: The system checks its database to match the entered username and password.
Post conditions: The user may start the system and launch UAV for operation.
Categorization: • Criticality: Low • Probability of Defects: Low • Risk: Low

c. Functional requirements

- The system shall display the login page on initiation.
- The user shall enter username and password to log in.

3.6 COMPONENTS DESIGN

3.6.1 PLATFORM

One of the issues was whether to develop our project in Windows or Linux Operating System. After a whole lot of progressive discussion between team members on components and issues like:

- NVIDIA graphics card firmware,
- NVIDIA cuDNN, a GPU accelerated Deep Neural Network library,
- CUDA, toolkit for application development, ● Tensorflow and tensorflow-gpu version, ● Python version.

The above software components were checked for compatibility, availability, reliability, interoperability, and ease of implementation.

It was then decided by a whole lot of research that implementation will be done on Linux OS using its Ubuntu (16.04) based on Debian. It was decided on the factor that it has a lot of support available online and it will be in more control of out project in Linux than in Windows.

3.6.1.1 Advantages of Linux Over Windows:

- Command line interface is handy.
- Run levels and run at CLI mode if GUI is having issues.
- Usability – Easier to complete tasks.
- Support – Sufficient support available online
- Open-Source

3.6.2 QUADCOPTER

3.6.2.1 3DR Solo

The solo is a flawless aerial-video instrument. It is powerful, simple, and consistent with intuitive Smart-Shots. It does not require a professional camera crew or expertise to control it hence making it user friendly. Since it is simple to use, it will massively aid in surveilling an area of interest with minimum training.

3.6.2.2 3DR Solo Description

Solo is a UAV powered by four rotors and propellers. Solo's PCs control navigation, demeanor, and correspondences in flight while sending constant telemetry and video and accepting control contributions over the 3DR Link secure Wi-Fi organize. Solo is streamlined for catching elevated video utilizing a GoPro® camera.

3.6.2.3 Controller

The controller gives control components and shows in-flight data on a full-shading screen. Utilizing twin long-range receiving wires, the controller goes about as the focal center for all correspondence on the 3DR Link arrange, accepting all interchanges from Solo and the application, sending telemetry yields to the application, and dealing with the transmission of all control contributions to Solo.

With the overlaid telemetry, the live stream from the Go pro can be obtained on the computer after connecting to SOLO's Wi-Fi and establishing a UDP connection.

3.6.2.4 Solo System Control Diagram

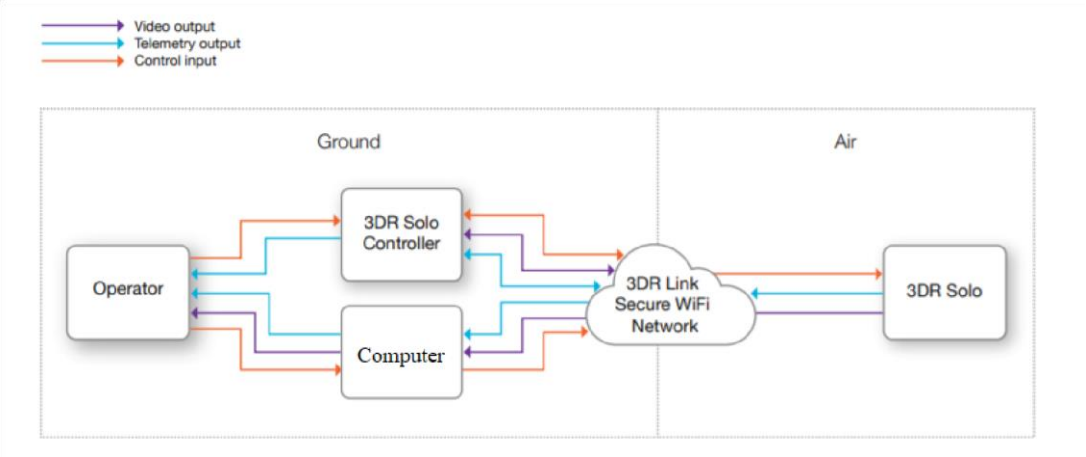


Figure 18. Solo System Control Diagram

3.6.3 SYSTEM COMPONENTS

3.6.3.1 Login

For entering user credentials and authorized access. User registered with the software will only be the ones to use the system. For a successful login, user needs to sign-up once with details such as username, password, city etc.

3.6.3.2 Detect Intruder

This component is responsible for establishing RTMP connection with the quadcopter based on the IP address and port number. As soon as the connection is established, Start stream option is enabled and live stream coming from the quadcopter is displayed on the frame and if intruders are caught in the stream, a bounding box is drawn around them and they are classified as persons (intruders).

3.6.3.3 Generate Alarm

The alarm goes off as soon as even a single person is detected in the stream. The volume and tone can be changed in the alarm settings. If the user wishes to turn it off, that can also be done.

3.7 INTERFACE DESIGN

3.7.1 OVERVIEW OF USER INTERFACE

Providing with the best UX requires a fair methodology throughout the SDLC. To ensure the balance, must not just focus on executing the implementations to finish an assignment yet in addition on how the task is accomplished through the UI. The following outlines the phases of UX/UI:

3.7.1.1 Designing

- **Functional requirements** – Administrative functions, authentication, external interfaces.

- **User analysis** – User's purpose, motive and mission was kept in mind while making the UI design.
- **Conceptual design** – How this engineering task will solve a real-life problem.
- **Logical design** – Process and information flow of application.
- **Physical design** – Implementation of logical design to solve technical and business requirements.

3.7.1.2 Implementing

- **Prototype** – Develop paper or interactive screen mockups that focus on the interface and do not include distracting visual design elements. Prototype is made because:
- Faster to use paper to get an initial prototype out
- Easier to change, to experiment with Easier to change, to experiment with
- Encourages feedback, as users do not feel like it is a big deal to change currently
- Focuses feedback on big things vs small (like the font)
- Self-documenting; can springboard from the paper description to an implementation description to an implementation
- Implementation neutral
- Allows the user to drive the design
- **Construct** – Build the product and prepare for design modification demands.

3.7.1.3 Testing

- **Usability testing** – With different users and circumstances.
- **Accessibility testing** – With available technologies and automated testing tools.

3.7.2 USER FLOW DESIGN

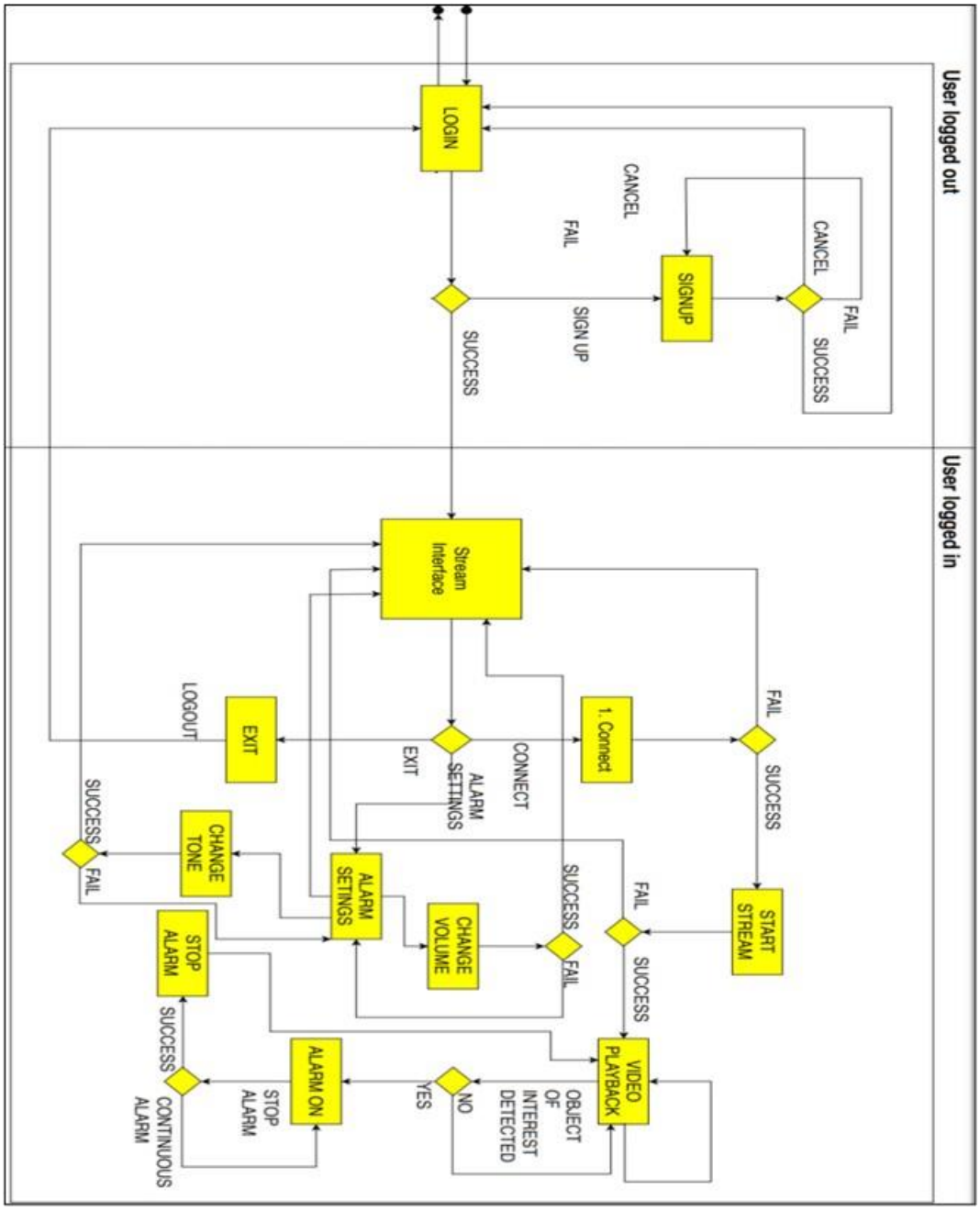


Figure 19. User Flow Design for user logged in and out of system

3.7.2.1 User Flow Diagram Description

User start from login screen, which has 3 functionalities namely **login**, **forgot password** and **signup**. The user returns to login screen if they logout from application or cancel signup or rest process.

Once the login is successful, the user moves on to the Stream Interface which is the central location that links to all system functionalities such as connect **RTMP**, **alarm settings**, **exit**, **start stream** and **video playback**.

Alarm Settings have options to change alarm volume and tone for user's preference. Alarm Stop button is enabled when object of interest generates alarm

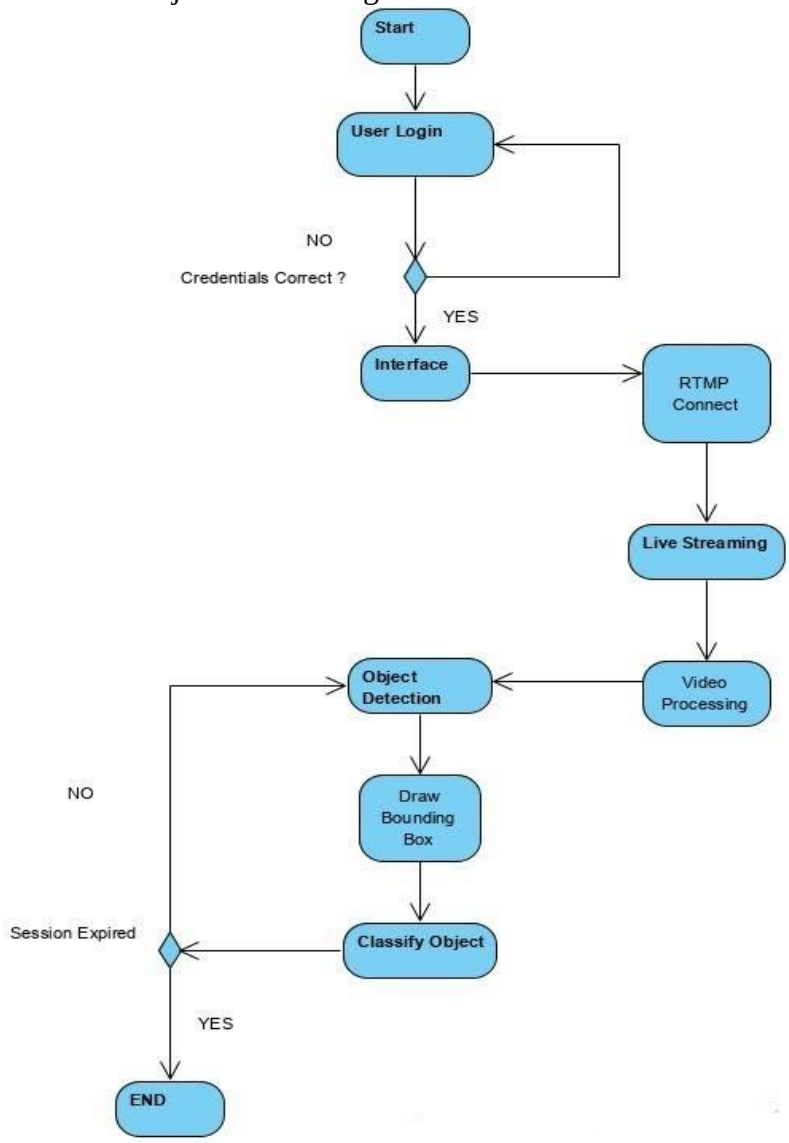


Figure 20. Activity Diagram

3.7.3 ACTIVITY DIAGRAM

3.7.3.1 Activity Diagram Description

Activity diagram is a behavioral diagram that portrays the actions taken prior to the input given to the system by user. It depicts the control flow from a beginning to a completion indicating the different choice ways that exist while the activity is being executed. In this

activity diagram, he behavior of system is depicted from the user login until his session is ended by him/her pressing the exit button.

3.7.4 UI

In design and manufacturing, a screenshot below shows scaled or fullsize working UI of the software, utilized for show, structure assessment, advancement, and different purposes. *3.7.4.1 Login*

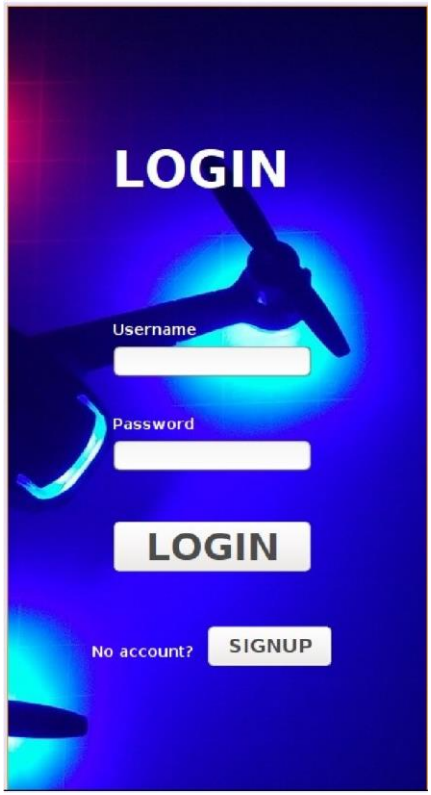


Figure 21. Login UI

The very first interface of our application the user will interact with is the login screen. The screen objects and actions associated with them is discussed in section 6.6.

3.7.4.2 Video CONTROLLER Interface

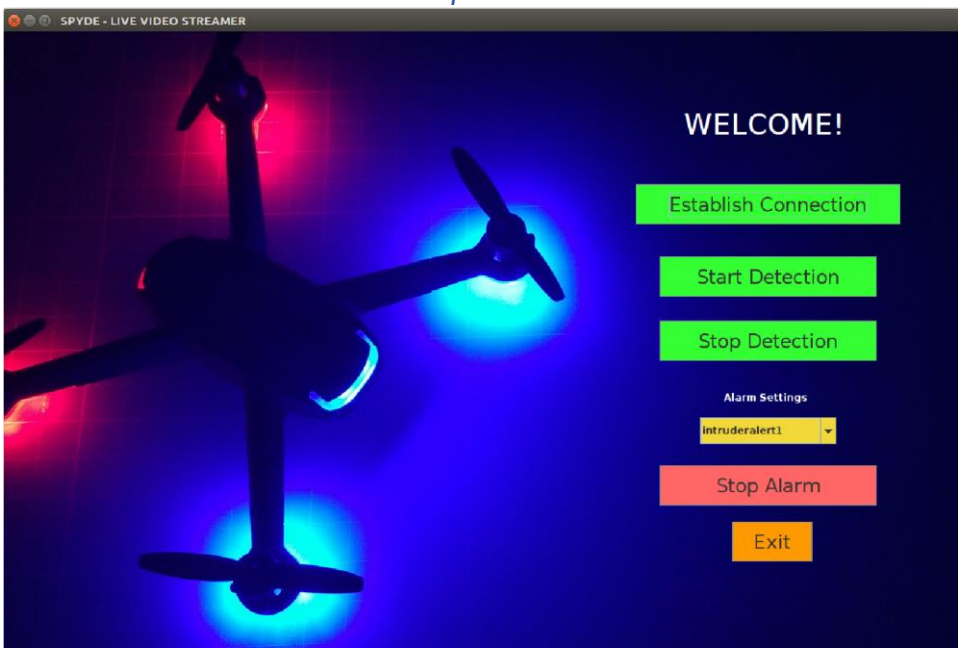


Figure 22. Video Controller Interface

This is the UI design for the second interface the user interacts upon successful login.

3.7.4.3 SIGN-UP INTERFACE

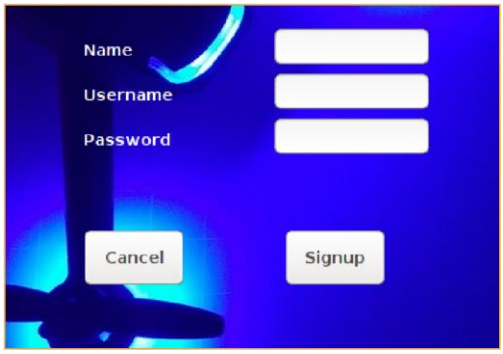


Figure 23. Signup Interface

This is the UI for signup, and it opened when user clicks on the signup button.

3.7.5 SCREEN OBJECTS AND ACTIONS

3.7.5.1 Login Screen

This implementation will provide secure and authenticated way of entering the video streaming interface. This will allow only the authenticated and verified user to use the features of this project. This user can:

- Enter his/her credentials i.e. credentials in the text fields.
- Click on **LOGIN** which will initiate credentials check from database.
- Click on **Sign up** if the user wants to register him/herself.

3.7.5.2 Video CONTROLLER Interface

After successful login, user will be shown an interface with video component and buttons, the user will be able to:

- Establish connection with RTMP,
- Upon successful connection, start detection and stop detection for detecting objects of interest,
- Setting up of alarm tone,
- Stop alarm,
- On Exit button click, the RTMP stream will close connection and program will terminate.

CHAPTER 4. PROJECT ANALYSIS & EVALUATION

The pictures below give a demonstration of the working of SPYDR.

The user is displayed with login interface which acts as first interface upon running of SPYDR:

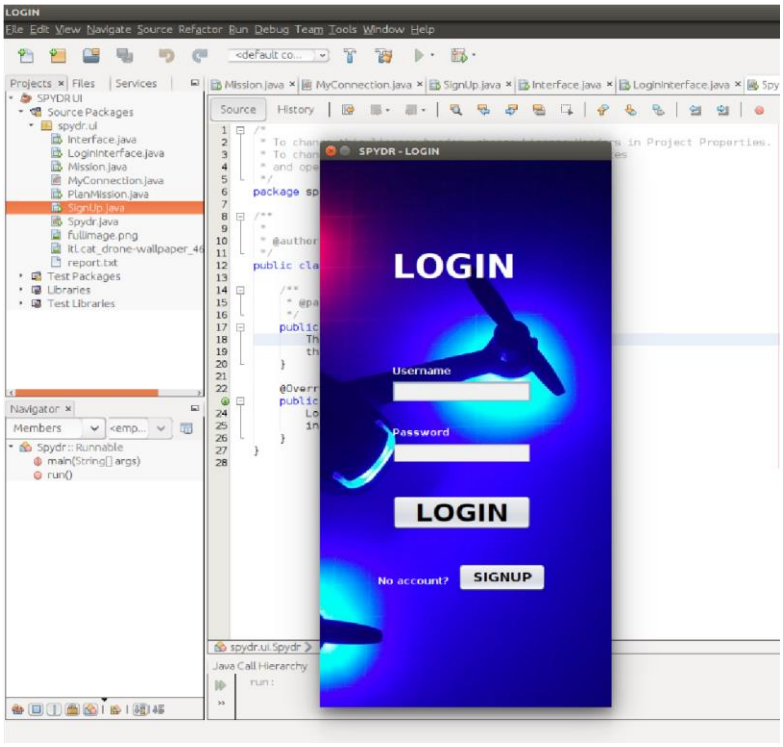


Figure 24. Login interface

If it is a new user, then the user clicks on SIGNUP:

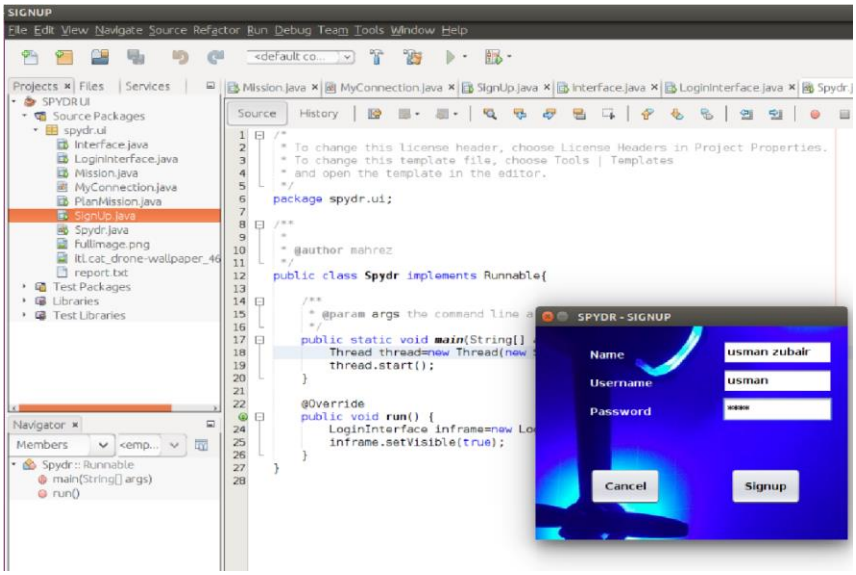


Figure 25. Signup interface

The user after entering his information clicks on Signup button:

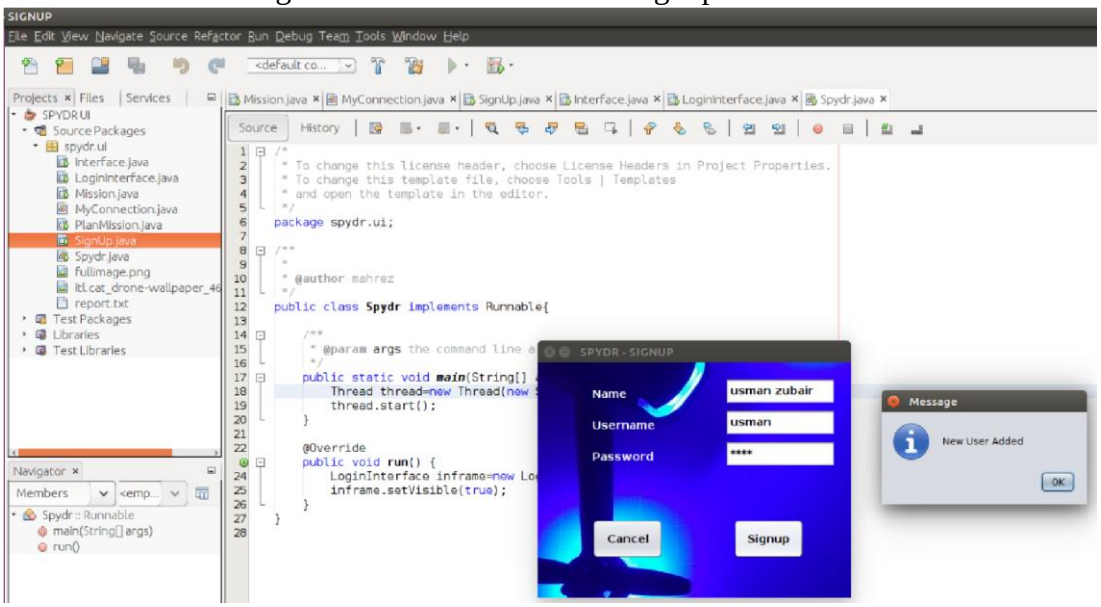


Figure 26. Successful signup

System displays a dialog box stating the addition of user in the database, now the user is ready to sign in.

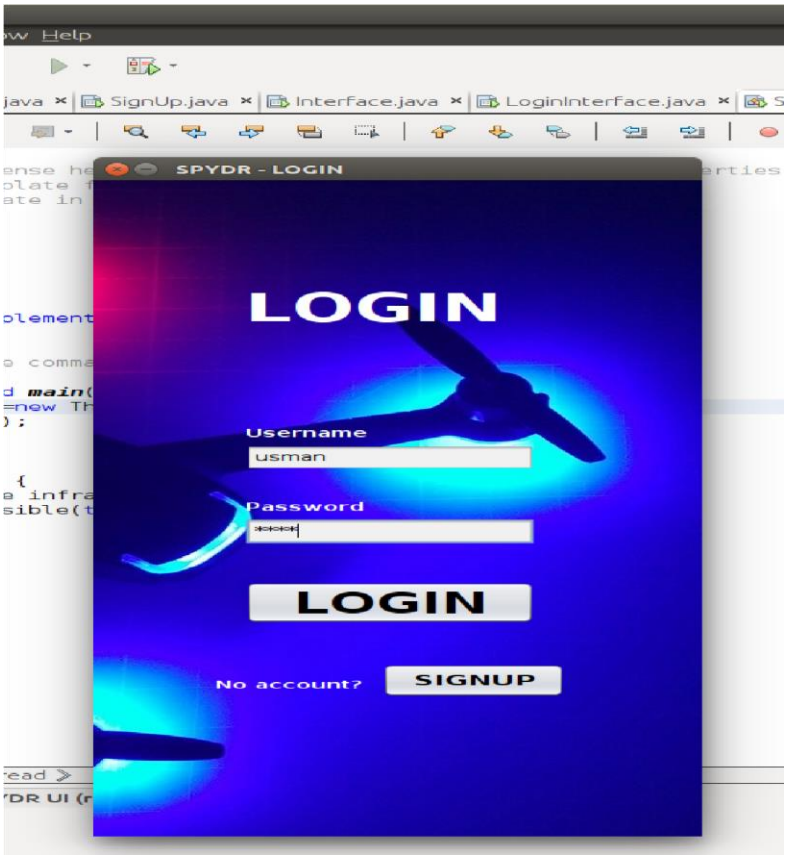


Figure 27. Back to login screen

The user enters credentials and clicks on Login this time.

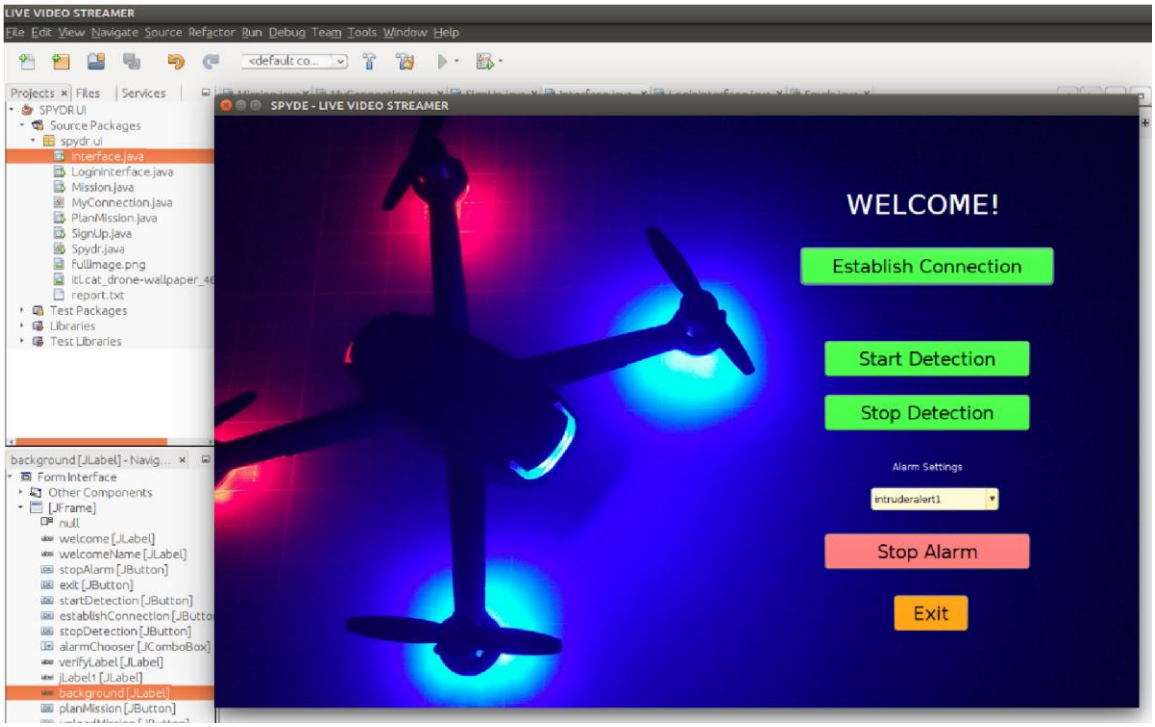


Figure 28. Control dashboard displayed upon successful sign-in

The user is displayed with a very informative control dashboard upon signing in. Now, the user must click on Establish Connection button to start a udp connection with the quadcopter so that the live video feed is steamed on the computer.

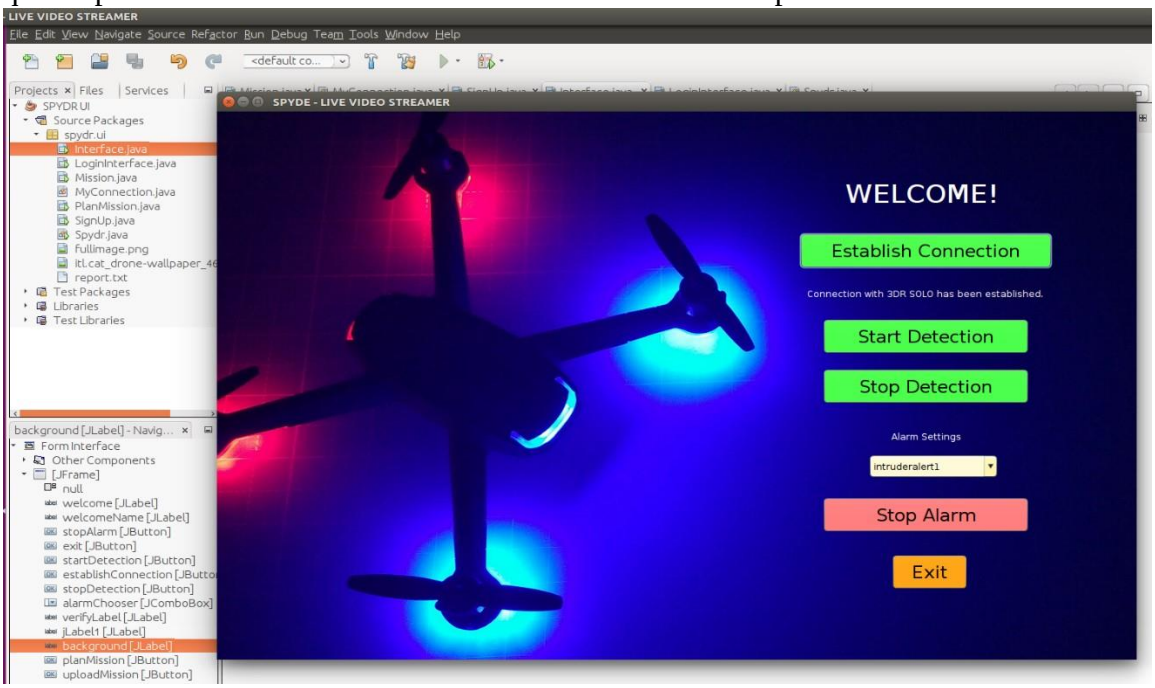


Figure 29. Connection established successfully

Upon clicking Establish Connection, text saying that connection has been establish is displayed.

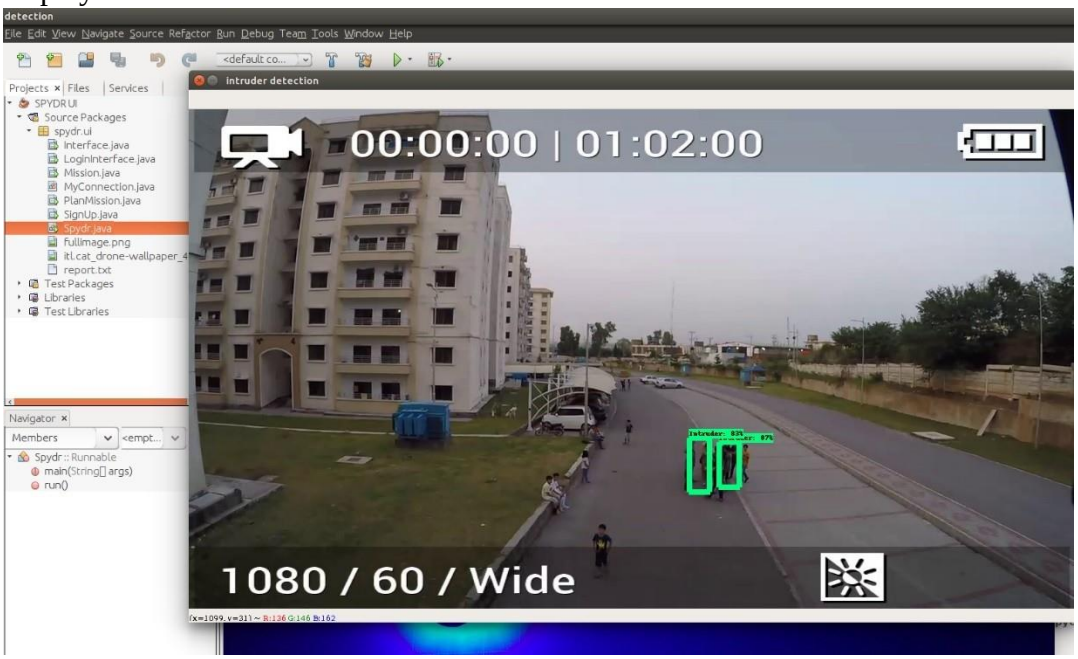


Figure 30. Video feed with intruder detection

Live video stream panel is displayed, and alarm is generated when intruders (human beings) are detected in the video feed from the quadcopter.

CHAPTER 5. FUTURE WORK

5.1 AI BASED OBJECT AVOIDANCE SYSTEM

We can improve the mobility of quadcopter when it is far out and a long way from the individual controlling the quadcopter by applying AI on quad-copter's obstacle/object avoidance.

The sensors on our quadcopter take a shot at Pixhawk, a Linux put together system that have control with respect to equipment like rotors, fan speed and altimeter.

AI execution will utilize sensors to distinguish objects in-flight inside 3-4 meters.

At the point when we will present AI, this won't just improve my flight way on map yet in addition make the quad-copter to maintain a strategic distance from obstacle/objects all through the flight consequently for example Trees, high rises and so on. This will make the flight protected, proficient and limit the danger of harming my quadcopter by crash.

UAV can identify and react to nature both progressively and statically relying upon the contribution from Range Data Unit. As navigation unit chips away at arriving at goal with briefest way, obstacle/object shirking system stresses over showing up at the goal with most secure way. It includes sensors, for instance radar, lidar, profundity cameras and

sonar. The contribution from these sensors is given to deterrent avoidance counts, for instance, thresholding or vertical field histograms.

CHAPTER 6. CHALLENGES

Image classification is hefty on a computer's processor especially if the GPU isn't powerful enough or doesn't exist to begin with. Since our team wanted to make a system that could be deployed on a laptop which could be tested in an outside environment and we didn't have a powerful machine at our disposal, we faced technical difficulties in trying to obtain and process the live stream.

Before livestreaming, we need to establish a UDP connection with the quadcopter to be able to receive its steam. As soon as we started the live object detection, the udp packets began to drop because each individual frame took time to be processed for an intruder. As a result, the UDP buffer stayed full and got memory errors. The resulting frames would be grey with glitches. The process would eventually stop responding without any acceptable results.

There were multiple solutions each of which contributed to the overall performance. Majority of them involved optimizing the code:

- Linux places very restrictive limits on the performance of UDP protocols by limiting the size of the UDP traffic that can buffer on the receive socket. So, we increased it to maximum.
- Removed any unnecessary waits/delays in our system which were originally added for efficiency purposes.
- Tested different pixel vs fps combinations in GO PRO settings with the object detection algorithm to see which one produced better result.

Pixels	Frames per seconds
4k	12.5
1080	60
1080	25
720	120
720	50
720	25

720p 25 fps worked considerably better than the other combinations with the algorithm reducing the UDP packet loss.

The most feasible solution to our problem with the limited hardware was dropping some of the frames per second altogether so the algorithm only 60% of what it was given per second. Out of the 25 fps, dropping 10 frames resulted in a reasonable tradeoff between speed and quality.

CHAPTER 7. EXTENDED FEATURES

A manually controlled quadcopter for surveillance isn't exactly feasible in a real-time environment. Wouldn't it be ideal if the copter flew on a planned trajectory following a set of pre-defined way points (coordinates) eliminating human intervention?
We have added mission panning as an extended functionality that immensely increases the value of the project. All we need to do is feed the way-point mission file to the quadcopter and shift it to auto mode. We have accomplished that using Drone-kit python (an API that allows us to communicate with the quadcopter via Mavlink protocol).

CHAPTER 8. CONCLUSION

All through in the FYP we had to face a number of constraints such as, connection establishment with the quad-copter, graying of frames due to computationally heavy deep learning tasks, packet losses due to udp buffer size constraint, random disconnection, model over-fitting during transfer learning and choosing from a wide range of available object-detection algorithms that solved our problem best, we surpassed all hurdles and found solutions suitable to the team's needs.

We conclude our Final Year Project report by pointing out how it can favor surveillance in almost any scenario one desires. Intruder detection is one aspect, adding more threats such as alien vehicles or weapons will most definitely increase our products value. The sky is the limit. What we have accomplished is a testimony to what can be further achieved if given the time, effort and necessary technology. For UAV's, one should aim higher than the sky.

CHAPTER 8. BIBLIOGRAPHY

[1] Malik A. Alsaedi, “Quadcopter Based Object Detection and Localization”, Iraqi Journal for Computers and Informatics (IJCI) doi:10.25195/2017/4317 Vol (43) Issue (1)

[2] Y. Bengio, "Learning Deep Architectures for AI" (PDF). Foundations and Trends in Machine Learning. 2 (1): 1–127, 2009, doi:10.1561/2200000006.

[3] A. G. Howard et al, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications” accessed at https://arxiv.org/abs/1704.04861

[4] W. Liu, et al, “SSD: Single Shot MultiBox Detector”, Computer Visionand Pattern Recognition, 2015.

[5] Budiharto, Widodo & Gunawan, Alexander & Suroso, Jarot & Chowanda, Andry & Patrik, Aurello & Utama, Gaudi. (2018). Fast Object Detection for Quadcopter Drone Using Deep Learning. 192-195. 10.1109/CCOMS.2018.8463284.

[6] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards realtime object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

[7] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In Computer Vision–ECCV 2012, pages 702–715. Springer, 2012.

[8] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only lookonce: Unified, real-time object detection. CoRR, abs/1506.02640, 2015.

[9] Han, Song et al. “Deep Drone: Object Detection and Tracking for Smart Drones on Embedded System.” (2016).

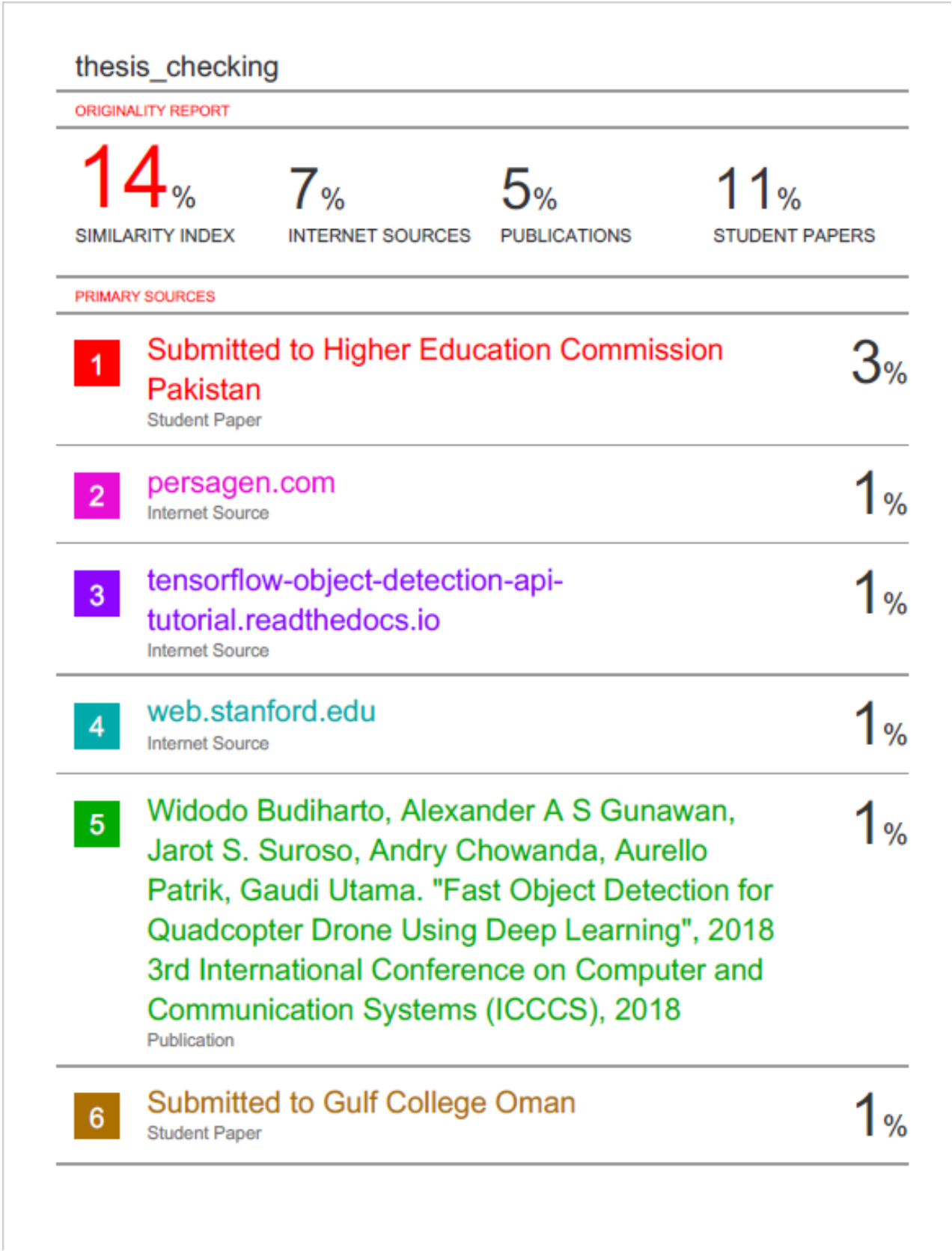
[10] Lee, Jangwon et al. “Real-Time Object Detection for Unmanned Aerial Vehicles based on Cloud-based Convolutional Neural Networks.” (2015).

[11] https://github.com/tensorflow/models/tree/master/research/object_dete ction

[12] https://github.com/Tony607/object_detection_demo

[13] https://3drobotics.github.io/solodevguide

PLAGIARISM REPORT:



thesis_checking

ORIGINALITY REPORT

14%	7%	5%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	3%
2	persagen.com Internet Source	1%
3	tensorflow-object-detection-api-tutorial.readthedocs.io Internet Source	1%
4	web.stanford.edu Internet Source	1%
5	Widodo Budiharto, Alexander A S Gunawan, Jarot S. Suroso, Andry Chowanda, Aurello Patrik, Gaudi Utama. "Fast Object Detection for Quadcopter Drone Using Deep Learning", 2018 3rd International Conference on Computer and Communication Systems (ICCCS), 2018 Publication	1%
6	Submitted to Gulf College Oman Student Paper	1%

7	docs.microsoft.com Internet Source	1%
8	honingds.com Internet Source	<1%
9	Submitted to Colorado Technical University Online Student Paper	<1%
10	Submitted to University of Technology, Sydney Student Paper	<1%
11	Submitted to Blinn College Student Paper	<1%
12	Submitted to Middle East College of Information Technology Student Paper	<1%
13	Submitted to HELP UNIVERSITY Student Paper	<1%
14	Submitted to Universiti Teknologi Malaysia Student Paper	<1%
15	en.wikipedia.org Internet Source	<1%
16	Submitted to Pathfinder Enterprises Student Paper	<1%
17	surveygizmoresponseuploads.s3.amazonaws.com Internet Source	<1%

18	link.springer.com Internet Source	<1%
19	Submitted to Informatics Education Limited Student Paper	<1%
20	Submitted to SUNY, Empire State College Student Paper	<1%
21	Submitted to University of Kansas Student Paper	<1%
22	Submitted to Royal Melbourne Institute of Technology Student Paper	<1%
23	Submitted to TechKnowledge Student Paper	<1%
24	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1%
25	Submitted to Texas A & M University, Kingville Student Paper	<1%
26	Yanyan Zhuang, Jianping Pan, Guoxing Wu. "Energy-optimal grid-based clustering in wireless microsensor networks with data aggregation", International Journal of Parallel, Emergent and Distributed Systems, 2010 Publication	<1%
27	Submitted to Asia Pacific University College of Technology and Innovation (UCTI)	<1%

	Student Paper	
28	Jimit Mistry, Aashish K. Misraa, Meenu Agarwal, Ayushi Vyas, Vishal M. Chudasama, Kishor P. Upla. "An automatic detection of helmeted and non-helmeted motorcyclist with license plate extraction using convolutional neural network", 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), 2017 Publication	<1%
29	www.pyimagesearch.com Internet Source	<1%
30	deepai.org Internet Source	<1%
31	Submitted to University of Durham Student Paper	<1%
32	Submitted to University of Nottingham Student Paper	<1%
33	Submitted to University of Wolverhampton Student Paper	<1%
34	www.scss.tcd.ie Internet Source	<1%
35	Submitted to University of Surrey Student Paper	<1%
	Submitted to University of Hong Kong	

36	Student Paper	<1%
37	Submitted to University of Bath Student Paper	<1%
38	Submitted to University of Dundee Student Paper	<1%
39	Submitted to University of Westminster Student Paper	<1%
40	Submitted to Midlands State University Student Paper	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		