2023

University of Huddersfield

Final-Report : CHP2524-2223 Individual Project

PROJECT TITLE : RECON-BIRD

SHIVANG CHAUDHARY [U2078485]

ABSTRACT

Surveillance has discreetly become a major factor in today's world. Nothing can be achieved without proper reconnaissance. Drones have become a catalyst for such innovative concept but alone it is just a piece of hardware with immense potential lying inside and that potential is in desperate need of usage.

This must be achieved through robust technology such as a reliable software. Thus, this project is titled '*Recon-bird*' and can carry out tasks that must be too dangerous through manual workforce.

Since always, the human workforce has been valuable for intense work environments such as factories and similar congested spaces. Thus, it constantly puts up the scenario of life and death, but those situations can be advanced for good. “The next inflection point for drones will arise with autonomous UAVs that can ‘see’ and fly intelligently. This opens the possibility for drones to play a greater role, with less human interaction.” (*Humans and Drones: A Dream Team, or a Case of “Man vs Machine”? - Technology Insights - PwC UK Blogs*, n.d.)

Therefore, this project puts forward the idea of solving the matter by using drones. This is already a practice at the industrial level and drones are being used extensively to inspect structures and locations that are difficult or dangerous for humans to access, giving their human operators a safe aerial perspective.

This will be achieved by designing a software using *python kivy* which will use Faster R-CNN object detection ‘tensorflow’ model to locate objects inside an image and display the score-confidence. Through manual inspection, those with the score-confidence lower than *50%* will be considered flawed objects and will be taken into consideration. Some testing will be done to ensure that the software can detect objects with utmost accuracy.

TABLE OF CONTENTS

INTRODUCTION

Interactive graphics for good user experience is intended for putting up presentable graphics. This is done by using '*OpenCV*' and '*KIVY*' components. Multiple screens are a requirement for the software. The first page is required to authenticate the user, this signifies an important part of this prototype since only the users knowing the *'key'* must be able to access the drone. The main screen provides for all the necessary buttons and pop-ups to properly guide the user to get well-acquainted with the software. All the widgets must be solely developed to make the user comfortable with the software. The last page must allow the user to exit the software, this page can only be opened when the drone not hovering. This way the drone flight is secured, and the user can only exit if the drone is not airborne.

The image of the object in proper shape must first be present in the memory of the machine which should be done by capturing the image of the object using the drone-camera. The software will scan the captured image and detect the score confidence in the object thus giving out the accuracy. Through manual inspection, a comparison can be made between *captured-image* and *detected-image* and that will be stored in the designated folder of the project. (Li, 2021)

A drone is a modern bird. The idea for this project was born out of the imperative necessity for the usage of AI in place of human intelligence. Not every task can be carried out by humans and not every task needs to be done. There are several examples like refineries, oil plants, and factories where workers are needed to maintain and repair the damages done every year, here some of the areas are too difficult to be visited and toxic in their natural state for the flaw to be located. In spaces like these, the drone can detect issues and flaws with utmost accuracy through proper trustworthy technology.

Following are some companies that utilise industrial drones for similar purpose: -

1. DJI camera drones
2. Parrot ANAFI
3. Delta drones
4. Scylla AI
5. Percepto

LITERATURE REVIEW

Object detection shares a similarity with object recognition with the only difference being that object recognition refers to a process to identify the correct category for an object whereas object detection is simply the way of detecting an object in camera view and locating it.

METHODS OF DATA ANALYSIS TECHNIQUES

1. Image Processing is a form of unsupervised learning, as it does not require any historical training data to teach analytical models. The models are capable of self-training themselves through input images and thus create feature maps to make predictions. Image processing does not require high GPU or 'graphical processing power' or even larger datasets for execution.

2. Deep Neural Network is a form of supervised learning in which there is a requirement for large datasets and high GPU computation power to predict object classes. It is a more accurate method for classifying objects that are partially hidden, complex, or placed in unknown backgrounds of a particular image.

APPLICATIONS OF AI VISION

Some of the important grounds that are formed for other AI Vision techniques through object detection are image classification, image retrieval, or object co-segmentation which is supposed to drive meaningful information out of real-world objects. (Wilson, 2018)

Many practical applications of these techniques are an upcoming potential need for many common people especially in the working environments including detecting pedestrians for self-driving cars, monitoring agricultural crops, and even real-time ball tracking for sports. "An object detection algorithm can help automatically detect cattle movements, traffic signals, and road lanes for self-driving vehicles to reach their destinations." ("*What Is Object Detection? Importance, Models and Types - G2*") This, in turn, eliminates the need for drivers for logistic errands.

Other important and necessary applications of object detection include its running on mobile networks which is possible by pruning the layers of a deep neural network. Its immediate uses include being used in security scanners or metal detectors at airports to detect unwanted and illegal objects. But due to its lack of cent per cent accuracy, it becomes less preferable for some other important and critical domains like mining and military.

POPULAR METHODS FOR OBJECT DETECTION

The biggest preferred and popular choices are deep learning or machine learning. Both methods work in conjunction with a support vector machine or 'SVM' to extract the features, train the algorithm and categorize objects. ("*What Is Object Detection? Importance, Models and Types - G2*")

Datasets play a vital role in object identification, this is because they cover all the major known features of an object like location, dimensions, category, or colours. (Moroney, 2020)

1. Machine Learning

The biggest advantage of this method is the manually entered data used rather than automated training data for classifying given objects making the algorithm overall error-free and more stable. Object detection being a supervised machine learning problem, needs pre-trained models to trigger object detectors. The list of classes in the training dataset of an ML algorithm must belong to a specific image or list of images. Machine learning approaches like natural language processing 'NLP' identify and classify objects based on their illumination intensity against a backdrop. Also, ML algorithms for 2D objects can be reused for detecting 3D objects in images. ("*What Is Object Detection? Importance, Models and Types - G2*")

2. DPM Object Detection

The deformable parts model or 'DPM' is also a machine learning algorithm that recognizes objects with a mixture of the graphical models and deformable parts of the image. Its main components include: -

Coarse root filter that defines several bounding boxes in an image to capture the objects. Part filters cover the fragments of objects and turn them into arrows of darker pixels. Spatial models store the locations of object fragments relative to bounding boxes in the root filter.

"A regressor is used to decrease the distance between bounding boxes and ground truth to predict objects accurately." ("*What Is Object Detection? Importance, Models and Types - G2*")

3. Deep Learning

The major difference between deep learning and machine learning that gives a USP to deep learning is that deep learning workflows come with automatic feature selection to suit your

tech stack rather than a manual selection of given features. “Deep learning approaches like convolutional neural network models produce faster and more accurate object predictions.” (“*What Is Object Detection? Importance, Models and Types - G2*”) Thus, critical needs include higher GPU and larger datasets.

Many modern-day object detection tasks carried out using deep learning include video surveillance cameras or monitoring systems that are powered by neural networks to detect unknown faces or objects successfully.

4. You Only Look Once (YOLO)

YOLO is a single-stage detection framework which is dedicated to industrial applications, with hardware friendly efficient design and high performance. Being a CNN, it is trained on large visual databases like image nets and coding capacity can be explored on platforms such as TensorFlow, DarkNet or Python. The YOLO produces state-of-the-art object detections at a lightning-fast speed of 45 frames per second. (Wang et al., 2021) The latest version that is YOLOv6 is used for training the custom datasets in ‘PyTorch’ via application programming interfaces or an 'API'. ‘Pytorch’ is a python package and one of the most preferred forms of deep learning research. “YOLOv6 is exclusively trained to detect moving vehicles on the road.” (“*What Is Object Detection? Importance, Models and Types - G2*”)

MAJOR APPLICATIONS OF OBJECT DETECTION

The current feats that object detection has achieved are in the domains like security, transport, medical and military. There exists an important process called 'data labelling' in which software companies use object detection as a method to retrieve and categorize large relational datasets automatically to increase product efficiency.

Some significant AI-powered object detection system includes:

1. Police, Forensics: Due to the critical use of object detection in locating a person, vehicle, or backpack from frame to frame, it is used by police officers and forensic professionals to inspect every nook and corner of a crime site to collect evidentiary proof.

2. Warehousing and Inventory: Logistics professionals can easily detect, classify, and pick up finished goods for transportation through real-time object detection. Some companies have even developed self-assist machines that deliver packaged items to customers' homes without establishing human-to-human contact. ("*What Is Object Detection? Importance, Models and Types - G2*")

3. Biometrics and Facial Recognition: One of the most important uses of this technology includes airport security checks which are done by employing facial recognition near the departure gates to attest to the identity of travellers. Common serious deceptions such as fraud and identity theft are prevented using facial recognition devices to compare identity documents with other biometric technologies such as fingerprints. During international transfer, immigration and customs departments use face matches to compare the portrait of the traveller with the picture in the passport.

4. Disaster Response: Other indirect and important uses include the creation of object detection applications. This is because there have been recent fluctuations in our ecosystems like the deterioration of the ozone layer, increase in greenhouse gases, and global warming.

MAJOR APPLICATIONS OF DRONE IMAGE-PROCESSING

1. Inspection of solar farms: Routine inspection and maintenance is a herculean task for solar farms. The traditional manual inspection method can only support the inspection frequency of once in three months. Because of the hostile environment, solar panels may have defects which is why broken solar panel units reduce the power output efficiency.

2. Early plant disease detection: For research purposes, a lot of researchers are mounting multi-spectral cameras on drones that will use special filters to capture reflected light from

selected regions of the electromagnetic spectrum. Stressed plants typically display a 'spectral signature' that distinguishes them from healthy plants.

3. Shark Detection: Analysis of the overhead view of a large mass of land/water can yield a vast amount of information in terms of security and public safety. For example, Australia-based Westpac Group has developed a deep learning-based object detection system to detect sharks in the water. This is because of the immediate topographical requirement. This serves as the solution to public safety in the given area.

There are various other applications to aerial imaging such as civil engineering for routine bridge inspections, power line surveillance and traffic surveying, oil, and gas for the purpose of on-and-offshore inspection of oil and gas platforms, drilling rigs, public safety for reducing risks of motor vehicle accidents, nuclear accidents, structural fires, ship collisions, plane, and train crashes. And at last, for security such as traffic surveillance, border surveillance, coastal surveillance, controlling hostile demonstrations and rioting.

All of these important uses can serve as the purpose of the drone project because aerial imaging is becoming a crucial need in today's modern world because evolution in such direction is happening at a much higher rate.

HOW TENSORFLOW MODELS WORK TO PERFORM OBJECT DETECTION

To use the object detection tactic in the software. The model types used are FASTER-CNN and SSD as a backup if either of them comes out as faulty when performing detection. In the source code, these models are being used in the software. Following are the study conducted on them and how they can work they work.

Transfer learning is performed to extract the best solution from another task, and after that, it is applied to different but related tasks. In deep learning, there are three main ways to use it.

The first method involves using *convolutional neural networks* as a fixed feature extractor, but the last fully connected layer of it is changed, and the front part is used as the fixed feature extractor for the newer image dataset.

The second way involves fine-tuning the *CNNs*, which is the same as the first method, but there is a difference. This method uses continuous backpropagation to fine-tune the weights of the pre-trained network.

The last strategy is using the pre-trained models. To have a new CNN structure, training done from scratch will be futile. Fortunately, TensorFlow model *zoo* has several pre-trained models. Researchers usually open source the checkpoint files of the CNNs they finally trained, so that the network can be used for finetuning.

This equivalent way is used to create checkpoints in the project-code. This for the sake of loading the models from it.

SINGLE SHOT DETECTOR (SSD)

The SSD approach discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. ("*Object Detection from the Video Taken by Drone via ... - Hindawi*") During the prediction time, the network will generate scores for the presence of each object category in each default box and produce adjustments to the box for the purpose of matching the object shape. Additionally, the network goes on to combine the predictions from multiple feature maps with different resolutions [ranging in 25–27] to naturally handle objects of many sizes.

SSD is simple and relative to methods that require object proposals since it eliminates proposal generation and subsequent pixel or feature resampling stages and thus encapsulates all computation in a single network. ("*SSD: Single Shot MultiBox Detector | SpringerLink*") This makes SSD quite easy to train and straightforward to integrate into systems that need a detection component.

The *SSD* training method is obtained from the 'MultiBox' method but can accommodate multiple object classes.

Following the mathematical perspective, consider:

$$\begin{gathered} x_{ij}^{p} = (1,0) \\ where\ above\ is\ the\ indicator\ for \\ pairing\ the\ i^{th}\ default\ box \\ to\ the\ j^{th}\ ground\ truth\ box \\ of\ class\ 'p'. \end{gathered} \tag{1}$$

Therefore, following is the possibility: -

$$\sum_i x_{i,j}^p >= 1 \tag{2}$$

The long-term loss function will be a weighted sum of the localization loss *(loc)*and the confidence loss *(conf)*will be :

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \propto L_{loc}(x,l,g)) \tag{3}$$

where $'N'$is the number of paired default boxes. If $'N' = 0$, set the loss to 0.

$$\begin{aligned} L_{loc}(x,l,g) &= \sum_{i\in Pos}^{N} \sum_{m\in(cx,cy,w,h)} x_{i,j}^p smooth_{L1}(l_i^m - g\prime_j^m) \\ g\prime_j^{cx} &= \frac{g_j^{cx} - d_i^{cx}}{d_i^w}, \\ g\prime_j^{w} &= log(\frac{g_j^w}{d_i^w}), \\ g\prime_j^{h} &= log(\frac{g_j^h}{d_i^h}) \end{aligned} \tag{4}$$

The loss of confidence is the softmax of multiclasses confidence *(c)* which is :

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{i,j}^p \, log(c\prime_i^p) - \sum_{i\in Neg} log(c\prime_i^0), \tag{5}$$

Where ,

$$c\prime_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{6}$$

$$and\ weight\ term\ \propto\ is\ set\ to\ 1\ through\ cross-validation.$$

FASTER REGION-BASED CONVOLUTIONAL NEURAL NETWORKS (FASTER R-CNN)

In this process, the detection process is divided into 2 stages. The first stage is Regional Proposal Network (RPN) where images are processed using a feature extractor and the features at some selected intermediate levels are used to predict class-agnostic box proposals. Next, they are fed to the remainder of the feature extractor. A prediction is obtained from each proposal, which is a class and class-specific box refinement.

The running time depends on the number of regions proposed by the RPN network, this is because a part of the computation must be run in each region. But it does not crop proposals directly from the image and re-run crops through the feature extractor, and this is further repeated.

In the project, *Faster R-CNN* is primarily used as the object-detection model. *SSD* models can pay more attention to scale, aspect ratio and predictions sampling location and compared to *Faster R-CNN*, the average time of each frame is 115 ms. But the target detection rate is low. However, *Faster R-CNN* is more accurate and finds more objects from a scene. 95% of all objects in each image can be easily recognized, but the average time of each frame is at least 140ms.

Therefore, this shows that *SSD* model on an average is faster in detection. On the contrary, *Faster R- CNN* is slower but more accurate.

The one way to speed up the *Faster R-CNN* model is to limit the number of proposed regions. (Sharif Razavian et al., 2014)

What is feature extraction and why it is essential?

Feature extraction is an essential procedure that must be carried out by neural network for the purpose of image recognition or image classification.

To be broadly specific about image recognition, the features are the groups of pixels just like edges and points of an object that the network will analyse for patterns. Features are the elements that will be fed through the network. (Guan, 2017)

Feature recognition or feature extraction is the process of pulling out the relevant features from the image inputted to the system so that the respective features can be analysed. Many images contain essential properties such as annotations or metadata about the image which in turn helps the network to find more relevant features.

The first layer of a neural network will take in all the pixels within an image for the purpose of extracting the features from the image. After all the data has been fed into the network, different filters are applied to the image, which forms representations of various parts of the image. This is the method called feature extraction and this creates the "feature maps".

This process of extracting features from an image is accomplished through a "convolutional layer", and convolution simply refers to forming a representation of a particular part of an image. It is from this convolution concept that the term *Convolutional Neural Network (CNN)*

was obtained, and this is the most common type of neural network used in image classification/recognition.

The neural networks have a parameter called *filter size.* This parameter affects how many pixels of the image are being examined at a time. The most common filter size used in *CNN* is 3. This includes both height and width and thus the filter examines a *'3 by 3'* area of pixels.

Digital images are rendered as height, width, and some *RGB* value that defines the pixel's colours. Therefore, the "depth" that is being tracked is the number of colour channels the image has.

Grayscale image have only 1 colour channel while coloured images have 3 depth-channels.

Thus, it can be assumed with utmost accuracy that for a filter of size '3', applied to a full-colour image, the dimensions of that filter will be 3 x 3 x 3.

For every pixel covered by that filter, the network multiplies the filter values with the values in the pixels themselves to get a numerical representation of that pixel. “This process is then done for the entire image to achieve a complete representation.” (“*Image Recognition and Classification in Python with TensorFlow and Keras*”)

The filter is moved across the rest of the image according to a parameter called '*stride*', which defines how many pixels the filter is to be moved by after it calculates the value in its current position. A conventional stride size for a CNN is 2.

The result of all the calculations yet is a feature map. This process is typically done with more than one filter, which helps in preserving the complexity of the image.

After creating the feature map, values are then passed through the activation function or activation layer. These values are what represent the image. Through the help of *convolution layer* the activation function takes values that represent the image, which is in a linear form. This is how their non-linearity is increased since the images themselves are non-linear. The typical activation function used to accomplish this is a Rectified Linear Unit (ReLU).

After the data is activated, it is sent through a pooling layer. Pooling "downsamples" the given image. In other words, this process involves taking in the information that represents the image to compress it, thus making it smaller. “The pooling process makes the network more flexible and more adept at recognizing objects/images based on the relevant features.” (“Image Recognition and Classification in Python with TensorFlow and Keras”) A *pooling layer* in a CNN will purposely abstract away the unnecessary parts of the image, resulting in keeping only the parts of the image that are supposed to be relevant, as controlled by the specified size of the pooling layer.

Max pooling which is the most used practice obtains the maximum value of the pixels within a single filter or within a single spot in the image. Thus, resulting in the dropping of *3/4th* of information, assuming 2 x 2 filters are being used.

For the further processing of the data, it is essential that the data is in the form of a vector, and thus the data must be '*flattened*'. The values are compressed into a long vector or a column of sequentially ordered numbers.

The final layers of the CNN are densely connected layers also called *artificial neural network* (ANN). The primary function of the *ANN* is to analyse the input features and combine them into different attributes that will assist in classification.

These layers are forming collections of neurons. "When enough of these neurons are activated in response to an input image, the image will be classified as an object." ("*Image Recognition and Classification in Python with TensorFlow and Keras*") The error, or the difference between the computed values and the expected value in the training set, is calculated by the *ANN*.

The next step in the complete concocted procedure is the *backpropagation*. Here, the influence of a given neuron on a neuron into the next layer is calculated and its influence is therefore adjusted. This is done in-order to optimize the performance of the model. This process is then repeated over and over, thus how the network trains on data and learns associations between input features and output classes.

"The neurons in the middle fully connected layers will output binary values relating to the possible classes." ("*Image Recognition and Classification in Python with TensorFlow and Keras*") It has a value of '1' for the class, believing the image is represented by the class and value '0' for the rest of the classes if more than one classes are in question.

The final fully connected layer will receive the output of the layer before it and deliver a probability for each of the classes, summing to one. The value between the range *'0-1'* will result in the score-confidence meaning, the probability of accurate detection and then multiplying it by '*100*' will give the percentage of the accuracy.

Similarly, the image classifier is trained, and images can be passed into the CNN, which will now output a guess about the content of that image. (Girshick et al., 2014)

USING OPENCV

OpenCV is an open-source library for image processing and for performing computer tasks. Its primary functions include face detection, objection tracking, landmark detection, and many more. One of the prominent features of it includes filtering images.

IMAGE FILTERS USING OPENCV

Images are used for extracting several features which highlight the properties of the image. Some of them are blur images, sharpening the images, and extracting edges from the images. Convolution is the process of applying the kernel over each pixel across the image.

A kernel is a small matrix consisting of numbers that usually sum up to unity. Depending on its values, various features from the image are extracted. A kernel is represented as a form of a matrix for identifying pixels of the image. (Pulli et al., 2012)

Consider the following as an accurate example.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{7}$$

PERSPECTIVE WARPING

Perspective warping refers to a change in viewpoint. This type of transformation will not preserve parallelism, length, and angle. But what is preserved are collinearity and incidence. Thus, implying that the straight lines will remain straight even after the transformation.

Consider,

$$\begin{bmatrix} tix' \\ tiy' \\ ti \end{bmatrix} = \begin{bmatrix} a1 & a2 & b1 \\ a3 & a4 & b2 \\ c1 & c2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{8}$$

Here, *(x',y')* are transformed points while *(x,y)* are input points. The transformation matrix *'M'* can be depicted as a combination of: -

$$\begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \tag{9}$$

-> This defines transformation such as rotation, scaling, etc.

$$\begin{bmatrix} b1 \\ b2 \end{bmatrix} \tag{10}$$

-> This defines the translation vector.

$$[c1 \quad c2] \tag{11}$$

-> This gives the projection vector.

In case of affine transformation, the projection vector is equal to 0. Thus, affine transformation is considered particular case of perspective transformation.

Since transformation matrix *'M'* is defined by 8 constants (degree of freedom). Therefore, to find this matrix, we first select 4 points in an input image and map these 4 points to desired locations in the unknown output image according to the use-case.

PROFESSIONAL, LEGAL, ETHICAL, SOCIAL ISSUES (PLESI)

The legal and ethical issues that are of concern and mainly confront society due to Artificial Intelligence are privacy and surveillance, bias, or discrimination, and potentially the philosophical challenge is the role of human judgment.

SOCIAL ISSUES

Concerns about newer digital technologies becoming a new source of inaccuracy and data breaches have arisen because of their use. ("*May Artificial Intelligence Influence Future Pediatric Research? —The ...*")

Therefore, the clients will be well-informed of the procedure and thus protecting their interests no surveillance will be carried out in unnecessary areas and no personal data of the people working in the area will be asked for and neither their images will be captured. (Rodrigues, 2020)

Similarly, the laws prevalent here will be pertinent issues that are addressed highlighting the need for algorithmic transparency, privacy, and protection of all the beneficiaries involved and cybersecurity of associated vulnerabilities. (Civil Law Rules on Robotics European Parliament Resolution of 16 February 2017 with Recommendations to the Commission on Civil Law Rules on Robotics (2015/2103(INL)), n.d.)

PROFESSIONAL ISSUES

Various real-time and critical issues include:

1. Informed consent to use data
2. Safety and transparency
3. Algorithmic fairness and biases
4. Data privacy are all crucial factors to consider

Here the emphasis is on the critical nature of a resolution calling for the immediate creation of a legislative instrument governing AI, capable of anticipating and adapting to any scientific breakthroughs anticipated in the medium term. (Coglianese & Lehr, 2019)

Therefore, there will be maximum transparency in this project meaning that the software will only store data that is immediately required by the user to store. Only the alternative images of the input demo 'image' will be used by the software to calculate the required results. (Biometric Information Privacy Act (BIPA), 2021)

Ethical Issues

There is proper planning for developing the software, thus no external help is taken only the credited sources are used for making up the modules for the software and all of them will be completely referenced.

This technology will only be used to store data of targeted object-images and nothing else to avoid any privacy breach for any other kind of data. (State of California Department of Justice, 2023)

Legal Issues

One of the ways to securely use the data for image processing is: Secure Federated Learning which is a decentralized data processing approach that uses independent nodes (devices or servers) to handle subgroups of data without sharing or exchanging information. "The processed data is then combined into a single machine learning model, but the raw data, in aggregate form, remains inaccessible to any single entity." ("*Ethical Issues in Computer Vision and Strategies for Success*")

The data captured on-site will not be divulged anywhere else and will be strictly handled by the user only keeping the client in loop of the same.

No drones will be flown in the undesignated area or where it is strictly prohibited. Very strictly the drone will only be handled by the user if he/she has the flying licence so that no official laws are broken. This is in accordance with the **UK Civil Aviation Authority.** (Overview : Flying Drones and Model Aircraft | UK Civil Aviation Authority, n.d.)

PROJECT PLANNING

USER REQUIREMENTS

Tello drone provides the intel chip and 5 MP (720p) camera that can carry out the detailed surveillance required for image processing. This will be used for the development of this project. For this software to work, it will first establish a connection to the drone and impart its controls so that it can be accessed via the main GUI.

Pycharm IDE for running the software since all the files are built using python libraries.

TOOL REQUIREMENTS

OpenCV is the open-source library framework that can process images using matrices and various mathematical formulas directly from the pixel's perspective, making the conversion of images possible. ("Virtual Sketch using Open CV")

PyCharm for using python, libraries such as 'opencv' and 'dji' to establish a communication with the drone and make the working of the software possible.

TensorFlow to build object detection file that uses deep learning methods. Models needed for the project include 'SSD' for detecting many objects with medium accuracy and 'faster R-CNN' for detecting targeted objects with higher accuracy. One of the models is needed as main model for usage keeping other as the backup model to broaden the scope of method of detecting objects.

KIVY for making up the GUI windows to authenticate user and provide an interface for the user to fulfil the usage of controlling the drone from the desktop. It is used because of the versatility of widgets that the library provides in turn resulting in creation of a robust user-interface. The key kivy components for usage are *button*, *camera module*, *label*, *kivyMD*, *switch*, *custom title-bar*, *kivy-canvas*, *boxlayout*. **(Appendix C: How python-kivy played a significant role as a framework)**

Relevance to the degree (BSc Computer Science)

The tools used for the project are one of the most common used tools among developers and thus all the tools were chosen since they were heavily taught during the duration of the course. This provides more confidence in planning the product development plan since the development tools are already well-acquainted.

In all the applications that are researched, the drone is used extensively for various industrial uses and object detection plays a major building block for the software utilised for most of the mentioned purposes. Therefore, the idea became quite clear of utilising object-detection software that requires image-feed from the drone to function. Here, the drone utilised is easily commercially available and programmable. This helps in achieving the prototype design for the software. The object detection implications are quite evident in the industry nowadays which is why the tools utilised for them are openly available. This is why it was easier to procure the libraries and code for them to build the file for performing the image detection.

Risk management

Summary

1. Risks included in the development of software are poor execution of the program, hardware failure, illegal environment for drone testing, and failure in working of the object-detection model.
2. Probability of poor execution is the highest, hence the code for the project will be thoroughly revised until the program executes as intended.
3. In-case of hardware failure, the project code will upload to a *GitHub* repository to have a backup on the development of the product.
4. Multiple *URL* will be stored in the file that executes the object-detection method in case one file does not work, another can be used as a replacement as a successful model.
5. A drone will be flown in a safe and open environment to avoid breaking any laws and for testing purposes.

RISK REGISTER

ID	Risk	Probability (out of 10)	Impact (out of 10)	Mitigation
1	Poor execution of source code	9	10	Immediately reassess the source code to make changes if the objective of the program is vividly clear.
2	Computer failure	5	7	Get the hardware assessed and repaired. Start working on backup hardware if absolutely necessary.
3	Drone crash	8	8	Though drone can be durable, minor crashes must be carefully evaluated and drone must be immediately repaired.
4	Delay in schedule	6	9	Start early rather than exact planned time to keep some extra time for reassessment as collateral
5	Poor execution of one type of model URL	4	8	Keep different URL models to be used as backup for performing object detection.
6	Failure in recollection of implementational challenges for documentation	5	7	Keep a journal while designing the software to culminate a good implementation section in the final report when designing is complete.
7	Deletion of source code files of the whole project or major fault occurrence in hardware	7	9	Keep a backup of the source code designed yet on version control git.

PRODUCT DEVELOPMENT PLAN

Duration: 5 weeks

Once the idea becomes clear, the first requirement becomes the development of software that consists of the widgets through which all the operations will be performed. Therefore initially, the *python-kivy* will be used for developing the GUI files.

Duration: 3 weeks

Next, develop the file for running the object detection on captured images. This file will contain methods to load the 'tensorflow' model that will run through the image to display the new object-detected image which will be finally stored.

Duration: 10 weeks

Depending on the duration for completion of the GUI files, additional features will be added such as special effects on the widgets, *GIF* for loading purposes (if any). Later, the time period that the model takes to load will be tested so that further improvements can be made if possible.
Testing the connection of the software with the drone to see if the drone can be controlled from the keyboard and if the camera feed is accessible.

Duration: 10 weeks

Re-visiting the code files to verify if any changes are needed and then make appropriate amendments.

Duration: 2 weeks

At last, document every change made in the source code files.

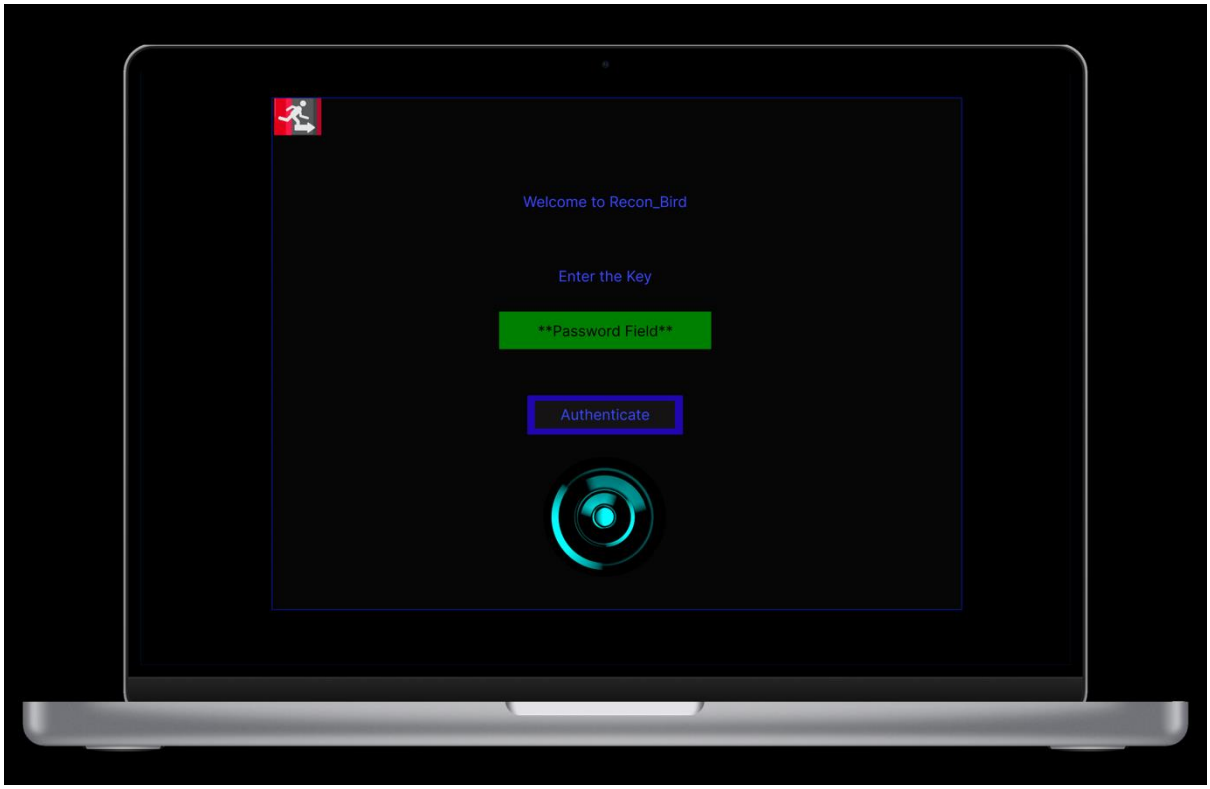
DESIGNING

PROTOTYPE WIREFRAMES

Following are the initial designs or the prototype for how each of the windows of the software will look like:

FIRST WINDOW

This window will first appear to the user when they log-in to authenticate them.

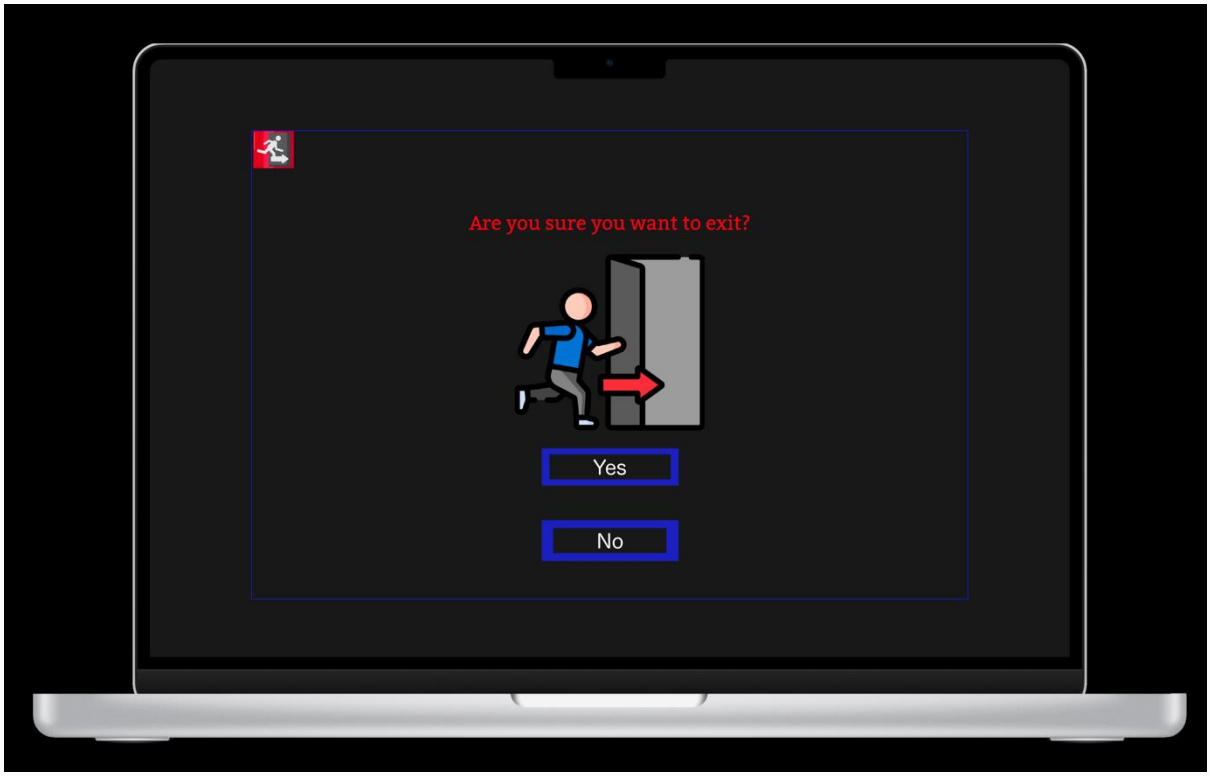


SECOND WINDOW

This is the main window that the user will log-in to after authentication and thus can access the drone-controls.

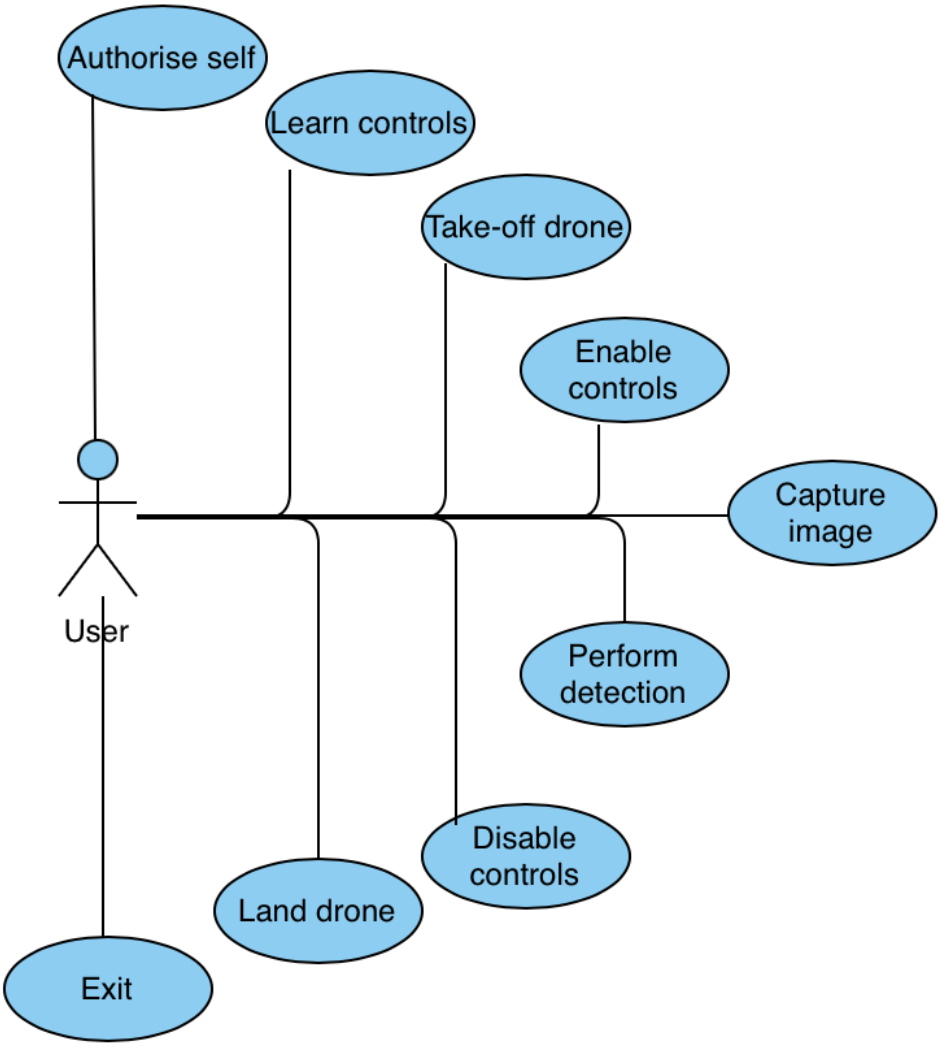
FINAL WINDOW

This the last window. It is required for the user to exit the software and can be accessed from either of the previous windows.

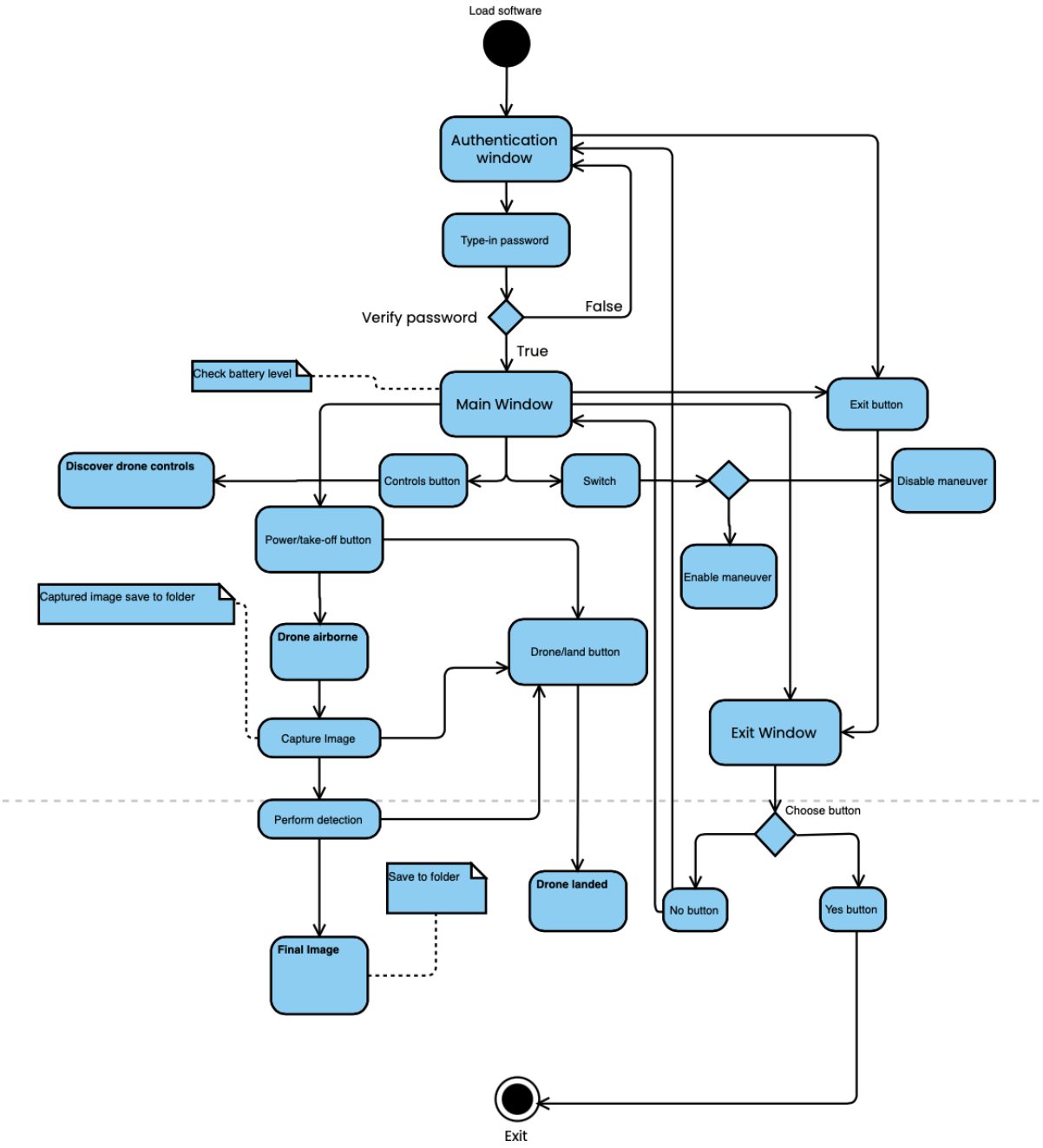


UML DIAGRAMS

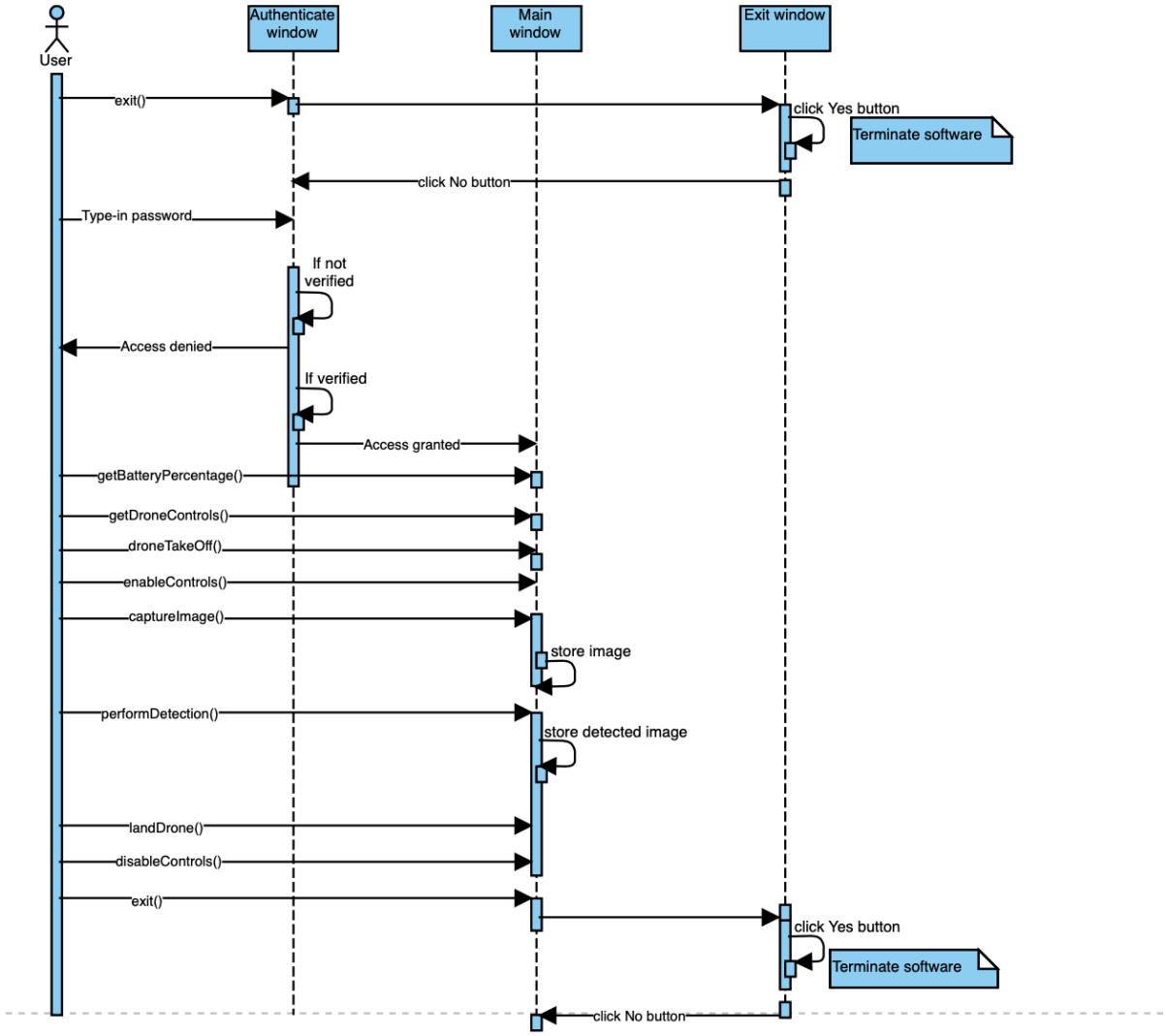
USE-CASE DIAGRAM



ACTIVITY DIAGRAM

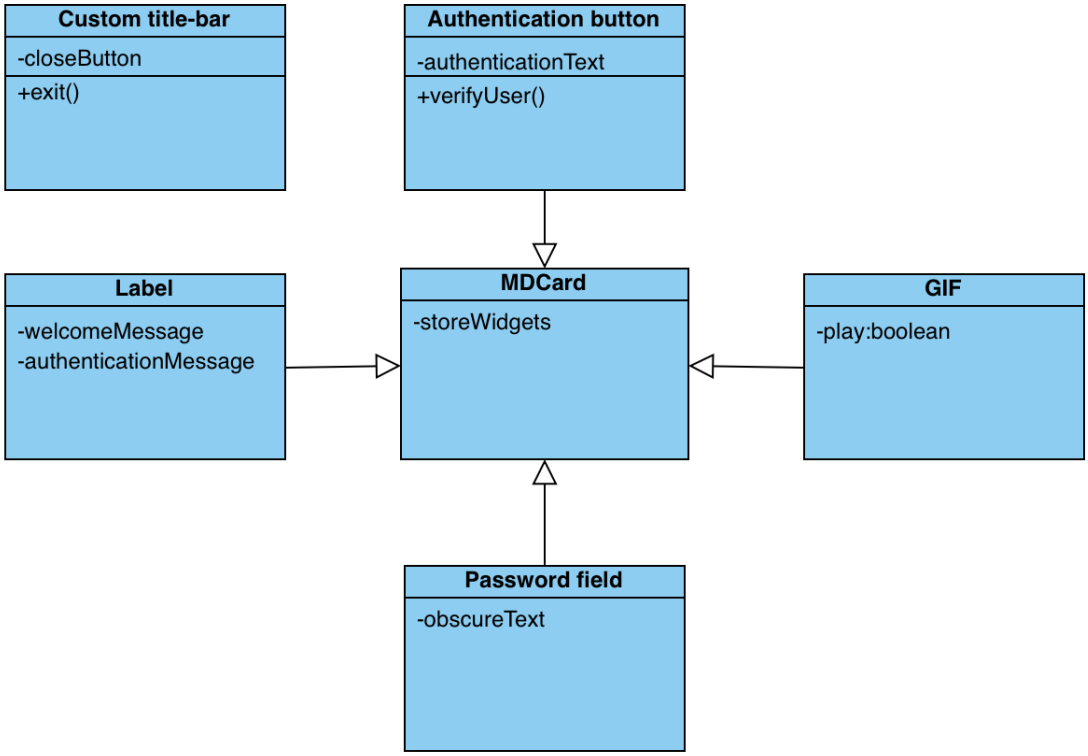


SEQUENCE DIAGRAM

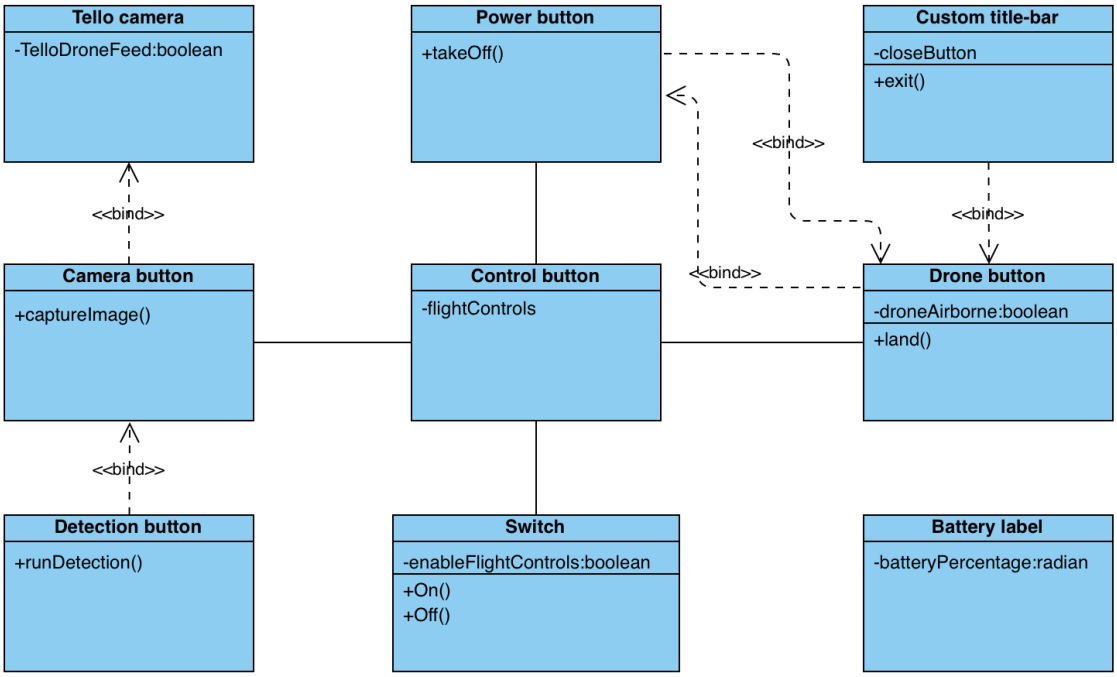


CLASS DIAGRAMS

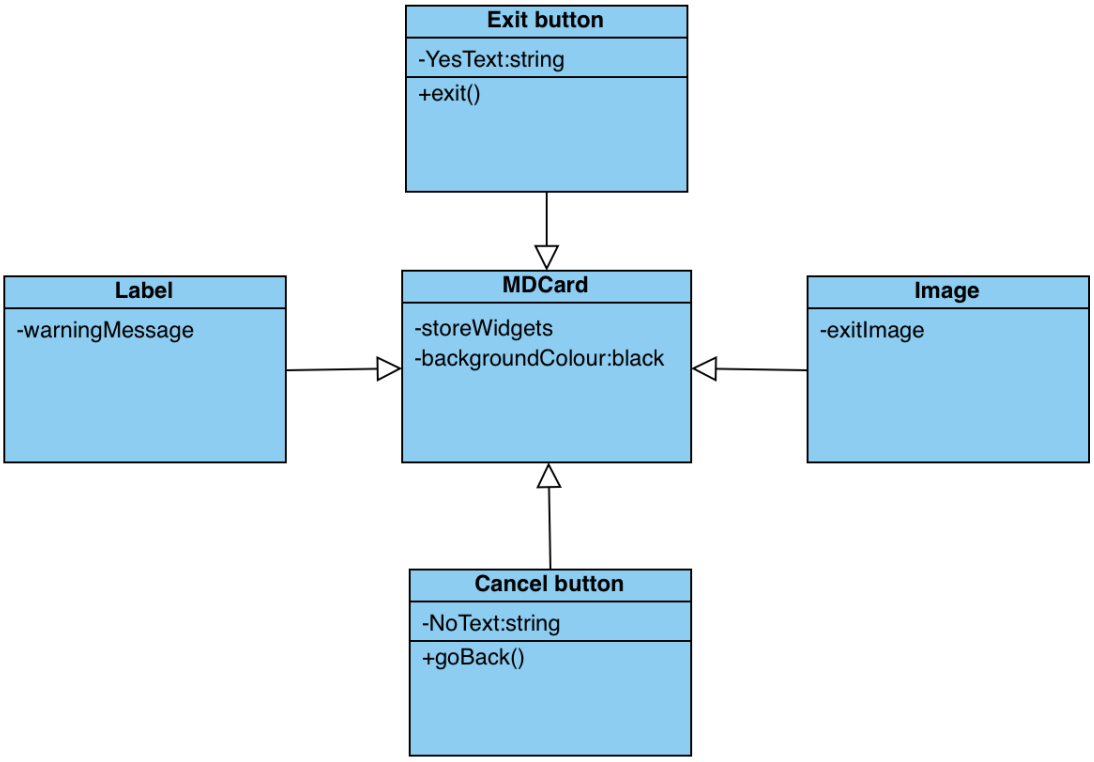
FIRST WINDOW



SECOND WINDOW



FINAL WINDOW



METHODOLOGY

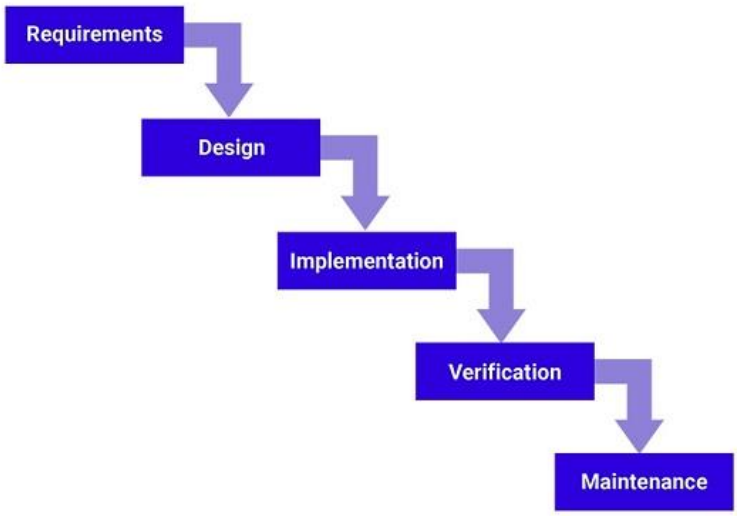
The project must be developed using a straightforward approach of building a user-interface so that the user can interact with the drone through the software. The software must be capable of building an image which can detect objects through deep-learning models.

The methodologies that can be adopted for the development of the project are:

1. Agile development methodology
2. DevOps development methodology
3. Waterfall development methodology
4. Rapid application development methodology

The approach to be taken for the project is **waterfall development method**. This methodology was chosen because of its structure. This structure is capable of providing a very efficient way to schedule the development of a project that should be done by a single person. Furthermore, given the objective of the project is noticeably clear, the method can provide much faster and efficient execution of the project as the requirements for the same are stable.

The **waterfall** development method is a rigid linear model consisting of sequential phases. Those phases which are also adapted include r*equirements, design, implementation, verification, and maintenance*. Each phase must be carefully examined and there is no going back to the previous phase but being distinct goals, each phase is constructed in a way to culminate a well-designed product. Therefore, each phase must be *100%* completed before moving on to the next phase.



The approach for this project is as follows:

1. **Requirements**: Procuring the libraries needed to develop the project which primarily includes tensorflow models for object detection (SSD and faster R-CNN), opencv, python-kivy, and most importantly *pycharm ide* for writing python code files.
2. **Design**: Modelling the project to get a clear idea of how the software must fulfil its purpose. This is done by designing prototype wireframes followed by UML diagrams which should include the class diagrams for each window in the software, activity, and sequence diagram for specifying user interaction with the software.
3. **Implementation**: Next, the user interface must be built using python-kivy which should include designing widgets for specific purposes such as enabling drone controls, disabling them, capturing images from drone feed, and performing object detection on the captured image. Various windows are required, 3 to be exact. First for authenticating the user, second for manoeuvring the drone and the last window is the exit page for the user. For embedding the object-detection ability into the software, individual files must be designed and added to the software package. A widget inside the main GUI will access this file when pressed to run object detection through the recently captured image.
4. **Verification**: Testing the project to make the necessary amendments given that the code executes perfectly. If the execution is not done properly, then making the required changes is must to move on to test rest of the remaining project parts.
5. **Maintenance**: The technical limitations for the project are just the dimensions of the window size which must be accommodated according to the feed getting from the drone. This is because the dimension of drone-feed is fixed and cannot be changed. The laws must be kept in mind to avoid leaking data and to capture images of only the objects intended. No other use for the drone is needed and neither is the drone used for. Thus, keeping a check on ethical issues that are not exploited for capturing any unneeded data. Moreover, the drone will not be tested or flown in illegal areas to avoid breaking any aviation laws.

IMPLEMENTATION DETAILS

DESIGNING KIVY FILES

1. While designing the main window that will encompass all the features, the biggest problem initially was to pre-determine the size of the window since it should not exceed user-usage and also should not be small enough from a dissatisfied perspective but rather the correct size so as to make sure the window is well- designed and very straight-forward without any superfluous features.

The **background** was already decided for the project. **(Appendix D: Background image for project)**

2. Using the '**BoxLayout'** for the main window to accommodate widgets side by side and in a proper and coherent manner. Using only 2 columns for accommodating all widgets. Later, to make the buttons more accessible and provide the feel of the touch, blinking effect is added to every button on the click. For improvement, sounds are added at the click of every button to each and every button.

3. New **folders** named - 'Captured_Images' and 'Detected_Images' are built inside the module which contains the software so that the images captured are stored immediately in the 'Captured_Images' folder and at the click of the button, the detection-performed images are stored in 'Detected_Images' folder.

4. In order to add better appealing to the widgets, **borders** are added according to the functionality of each button.

Green meaning the widget is active in performing a feature.

Red meaning it is disabled.

Blue meaning that the widget is functional, but it is more of a user-requirement feature rather than a feature that is related to drone.

5. Using **animation**, blinking effect is added to every button whenever it is clicked. The function uses a time interval from '*Clock'* module in *python* to perform the animation on the widget.

Working: In this animation, constantly a rectangle is created and destroyed around the logo of the widget. This cycle of creating and destroying the rectangle is repeated for a certain amount of time and the colour of this shape can be modified. Every cycle of animation uses different measurements. Every time a cycle is executed, a new cycle begins with new *length* and *breadth* measurements to provide a blinking effect. The colour of this animation is related to the border to add a visual effect to the widgets.

6. **Canvas** in Kivy is used to add shapes to widgets borders and further on animation such as blinking rectangles and this is done directly in the 'kv' file.

Considering the code as an example that uses the **Image** widget, the look of the button can be changed to desired logo. **(Appendix E: Canvas source code)**

7. Now after the widgets are decided, the correct **dimensions** for the window must be pre-determined in order to make sure the widgets are well-placed and there is no lag while loading the window. The initial results were not promising. **(Appendix F: Initial graphic results)**

Thus, the graphics were more appealing after making some final adjustments. **(Appendix G: Final graphic results)**

8. **Significance of border-colours**: The power button must appear green signifying that the drone is ready to take off. After it is clicked and the drone is on-air, the button is disabled and thus the border turns red meaning that the power button is disabled. The drone button turning to green means that the drone is *active* and on clicking the respective button, the drone will land, and the power button is enabled thus turning back to green. Now the drone button is turned back to blue meaning the drone is now inactive but connected to the software. The camera button is active and ready to take pictures which is why it is always green. But the detection-button is not yet green but red until the image is captured or camera-button is clicked. **(Appendix H: Enabled border widgets)**

9. The **camera-button** has basic functionality to capture the image from the live feed. The camera widget uses in- built *kivy camera function* to capture the image from the camera module. This captured image is then stored in its designated folder.

10. **Camera-Module: -**The **Tello-Camera** widget is used in 'kv' file to get the feed from the drone in the kivy window. This was placed in the *python* file as a different class to be used in the software. The *Tello-Camera* class returns the kivy image-texture. A function is then used in camera-button function in order to capture the kivy texture-image being displayed which was then stored in its designated folder. **(Appendix I: Tello-camera code files)**

11. The battery is displayed as a **rounded label**. The label consists of 3 ellipses where: -

The first ellipse will be green displaying the full battery initially. The second ellipse will be of red colour having a fixed *start-angle* but a varying *final-angle* for adjusting battery percentage where *final-angle* is battery percentage multiplied by 3.6 *(for 360 degrees angle)*. The third ellipse is adjusted in the middle of all ellipses with slightly smaller measurements. This consists of an image. The battery-label now displays the remaining battery percentage in circular format.

12. Pop-ups: -

The **first pop-up** activates whenever the control-button is pressed displaying the drone controls and then it deactivated once 'ESCAPE' key is pressed or when the 'Cross' on the window is pressed.

Second pop-up is the one that activates for a few seconds until the detection is successfully performed. On- pressing the *detection-button* the pop-up comes up for informing the user to patiently wait until the detection is performed. After the detected image is successfully loaded, the pop-up is dismissed automatically.

The pop-ups must be called using the *root.func()* in-built python kivy function after defining them inside their corresponding python code.

13. There are certain **boolean properties** being implemented in python file that are further used as conditions in 'kv' file to *disable* and *enable* buttons.

14. Adding **another window** to our project will serve as the starting window. This window specifies the authentication part of the project. Since this software must be used in industrial

environments, it is required that the data it stores in the form of images and functions it performs must be safeguarded before usage.

In the *'build()'* function of the *main.py* a screen-manager is created to manage different screens. The screen manager will store all the screens and then when the program initiates. First window is named as 'Authenticate' window.

This window takes a *key* as an input which will serve as the password for the software already declared in the *main class* and then a widget-button will validate if the entered password is correct and only then grant the user access to the '*main_window*' where the real-control for the drone lies. This way it is secure from the beginning.

The design of this 'Authenticate' window includes: -

1. *MDCard* for stacking all the widgets in the centre of the screen.
2. *MDLabel* to provide a welcome message for the users which will also display successful or unsuccessful attempts.
3. *MDRoundFlatButton* to validate the input text.
4. *MDTextField* to provide space to type in the password and the text-input is obscured.

15. Using the *'Window'* class to remove the title-bar and **disable the exit** of the window 'ESC' key is pressed. To preserve the shape of the software, the resizable function of the screen is disabled.

16. The **customised title-bar** is then made for the window to have *closing widget* to ensure the exit from the window.

The title-bar is made using *'Action-bar'* from the kivy, it was imported from **kivyMD** class. The customised title bar is set with position coordinates so that it can appear at the supposed position on screen, which is at the top of the window. Its colour is set the same as the background colour of the window. The title bar in second window is grey coloured. The closing button in title-bar navigates the user to exit-page.

17. Creating the **exitpage** for the user. This page contains the **MDCard** to accommodate a **Label**, **Image** (Displaying the exit-image), two **Buttons** that include the options of **Yes** and **No** for the user.

The label will display the final message for the user similar to "Sure you want to exit?" and below them will be **Yes** and **No** Buttons.

The yes-button will result in the closing of the window resulting in exiting the software. The no-button will take the user back to the page the user came from.

18. The '*rc.sendControls()'* function is used to send values to the drone for providing some motion to the drone when airborne.

The '*rc.sendControls()'* has 4 parameters in which: -

1. 1st parameter will send left-right flight movement. Here a positive figure will turn the drone to the left and a negative value to the right.
2. 2nd parameter is the forward/backward movement for the drone. Positive value for forward and negative for backward movement.
3. 3rd parameter is for hovering the drone. The positive figure will increase the altitude of drone-flight and the negative value will do the reverse.
4. 4th parameter is the yaw-velocity which same as the 1st parameter in which a positive figure will turn the drone to the left and a negative value to the right.

This is accommodated in the widget **switch**. Green border and sound effect is added to the switch.

19. The normal movements such as forward, backward, left and right are done using the 'WASD' keys. The hovering through 'UP' and 'DOWN' keys. For rotation in both clockwise and anti-clockwise directions, 'LEFT' and 'RIGHT' keys are used.

The keys were tied to key-inputs of the keyboard to control the drone using the keys. The '*keyboard'* from '*Window'* class which is in-built for python was used for getting the key code of every key whenever pressed and using those key-codes the key-inputs were detected and thus every movement of the drone was outputted to the drone whenever the keys were pressed.

CHALLENGES

1. There were many challenges with getting the live feed from the DJI Tello drone onto the kivy window. Since a '*dji*' drone will be used for this project, it was supposed to be quite easy to get its feed. Images must be captured as a frame from the live video the drone broadcasted onto the python window through the video that the drone itself captured. But the image module just could not receive the image from the drone in real time. **(Appendix J: Challenge in designing image widget)**

So, to solve this issue of getting a blank image, many methods were opted for before getting the correct live feed.

This part was adjusted on the window in the first row of widgets to get the feed.

At first, the issue was very straightforwardly getting a blank image in place of getting the feed. The widget used for the image was '*Image*' widget. **(Appendix J: Challenge in designing image widget)**

Here, the image widget is used for accommodating the live feed from the drone. The idea was to get each frame individually from the video and convert it into a texture using the in-built *camera-texture of kivy* so that it can be displayed. '*OpenCV*' was used in the conversion of the image into texture. But the image could not be obtained in real-time.

At first, the resolution for the widget was tested by using the desktop camera and thus using 'Camera' widget but it was then commented to use the 'Image' widget to get the feed from drone than from the desktop camera.

So, to solve that issue, the feed was first obtained from the drone before the software initialises and when it was finally launched, it was much easier to get the image displayed on the kivy-widget.

But the next issue was that image could only be displayed once when the drone was online, and it could no longer receive images once software loaded due to a technical lag and could not get any feed. So, it ended with just getting that one image at the beginning. Since this widget no longer posed a purpose, it was scratched and rather a camera-widget was used to get the feed.

2. The problem in designing the windows involved in tracking the previous window the user came from. At first, a variable was introduced in the class which is the first window "**Authenticate**", here a variable is initialised to 1 and then the count of it is increased once the user manages to successfully log-in to the next page thus increasing the variable 'page-count' to 2. Now, in the class that is the **exit-page** as annotated in the python code, the 'page-count' variable is called from the **"Authenticate"** class in-order to specify whether the user is on first-page or the second page which is the main-window.

This way the user can be successfully tracked back to the page he/she came from. That is if the user logged on to the software and is presented with the log-in page but immediately decides to exit without moving on to the second window.

On pressing the exit button on the title-bar, the user will be directed to *'exit_page'*. But now if the user wants to go back to the first window, the *No* button without any errors will take back user to the 'Authenticate' window.

Similarly, if the user has logged-in and is at the second window and now wants to exit. The user can then decide to either exit the software or go back to the main-window or the "second window" without needing to re-authorise themselves over and over and thus continue their operation of the software.

But whenever that variable was called out using the respective class in-order to get the position of the window, the variable could not be modified because such an operation is not possible in python, thus this idea was scrapped.

But since it was imperative to keep track of user-movements on both the windows and exit-page through back-and-forth movement, 2 different classes were created that were the exit-pages. Both these *classes* or *'exit_pages'* are utterly identical. But these two different but identical classes have one different function. And that is the function of the **No** button.

exit_page1 - as also annotated in the code will direct the user back to the first (*authentication*) window if the user has not gained ethical access.

The second class or **exit_page2** is called when the user wants to exit while on the main window. Thus, if needed the user will be directed back to main window if once already identified since the software has been launched. To ensure the transition is direction accurate. The exit-page will transition in the right direction while the re-direction in the initial window will transition the user in the left direction if the user decides to stay and not exit.

3. The final problem remains **movement controls of the drone** when the drone-connection is secure, and it is ‘take-off()’, and ‘land()’ controls are confirmed.

In-order to establish drone-control connection in the software, the power-button is used. Whenever the power-button is clicked, the ‘*rc.sendControls()’* function is used to send values to the drone for providing some motion to the drone when airborne.

But one major problem encountered while doing this was that while any key, for example, 'W' was pressed for moving the drone forward the drone would keep moving forward until another key was pressed and once pressed the drone would continue its movement forever for every keypress. This resulted in near-fatal crashes for the drone.

Therefore, to avoid these threads were introduced whenever any key was pressed. A thread would activate on a keypress and for a limited time, a certain velocity says "20" was passed in the respective parameter of the ‘*sendControls()*’ function and after the time meets its end, one more time the function was called in-order to send the velocity "0" to terminate the velocity of the drone. This is how its movement is controlled.

So, now the drone flight could be controlled without any technical errors using the **switch** and the program would no longer crash.

USING THE TELLOPY LIBRARY

DJI Tello Drone is the programmable drone used for this project. The library ‘*tello’* is used in python for procuring all the properties of the drone and this library was specifically made for python used along with OpenCV for image processing.

Example for using *‘tello’* library:

Import the ‘tello’ library as ‘*me*’.

Now, using 'me' import all the drone properties such as battery, drone movements which include forward, backward movements and yaw.

At last, import feed using ‘*me.frame()*’ and this feed is used in the software.

For these features to work connect the drone from the source-code using ‘*me.connect()*’ and then after establishing a connection with the drone, all the features from the drone can be imparted and used in the kivy software. The connection breaks when the program terminates.

OBTAINING KIVY TEXTURE-IMAGE

OpenCV is used for converting the image type into kivy-texture so that it can be processed by Kivy-widget.

This is done using '*Clock*' module python. Using this time interval, the '*process_frame()*' function reads the frame from the drone camera and then using OpenCV converts the format from BGR to RGB and then flips the image. Now, the camera texture is used to obtain a kivy-texture of the image. Then it is converted to byte format using '*blit_buffer*' of kivy texture and then that texture-image is returned.

Thus, now this image as a single frame of the whole feed can be displayed or captured in a format that can be processed by Kivy.

USING TENSORFLOW AND OPENCV

The '*display()*' function is used by the detection-widget from class '*Detector()*' to load a model through URL and then perform functions using those components on the captured image to perform detection. The detected- image is then stored in its respective folder.

The components of the '*Detector()*' class used are: -

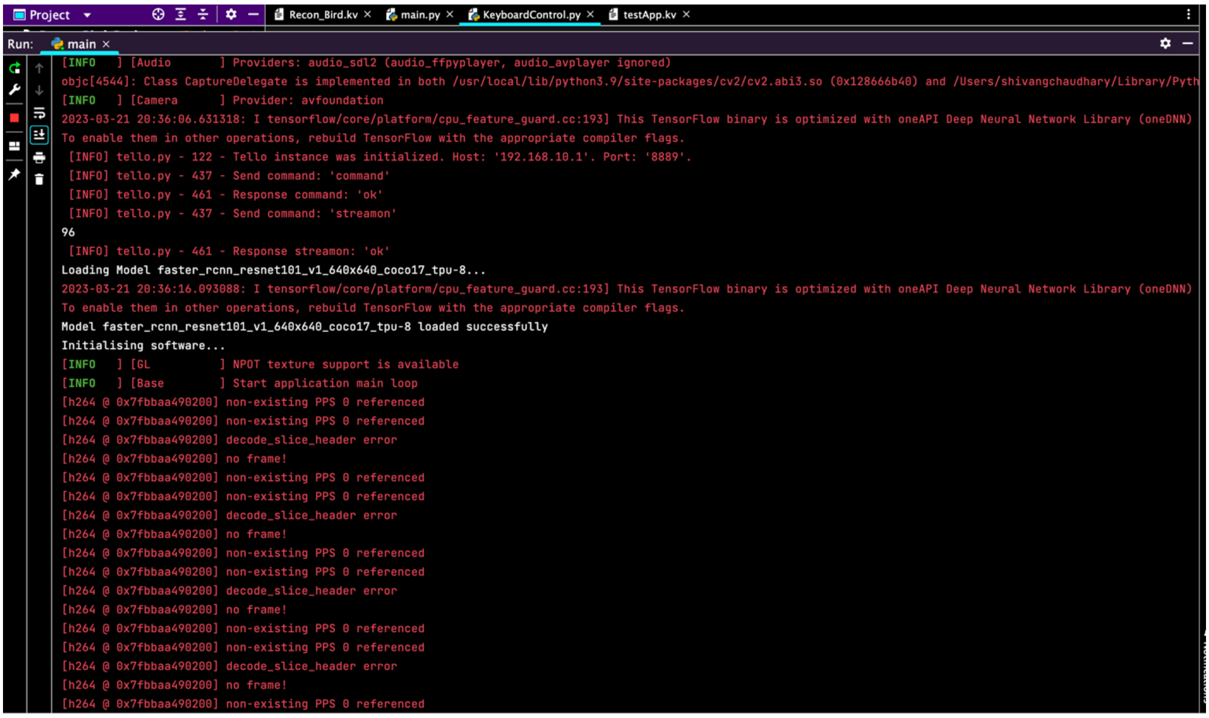
1. readClasses(): This reads all the 92 classes pre-stored in the given 'Coco.names' file to identify the object. The identified object must be among these classes and then using **OpenCV** bounding boxes are created to separate the pixels of the image of the object from the rest of the image pixels.
2. downloadModel(): Needs a model URL to download the model pre-stored in the same folder or simply locate its address.
3. loadModel(): Loads the located model and for confirmation "Model loaded successfully.." is printed as the output once the TensorFlow model without any constraints is loaded with pre-installed classes that can be detected.
4. PredictImage(): This function at the end finally detects the object, procures its name from the class file, and then puts up bounding boxes around the object in the detected image along with its score-confidence signifying the accuracy of the flaw in the object. This is essential for the primary purpose of this project and for industrial usage.

The model used in this project is 'faster_rcnn_resnet101_v1_640x640_coco17_tpu-8'. This model specifically was chosen for the fastest response time and its focused accuracy for detecting objects. Moreover, reliability in terms of score-confidence. This is pre-stored in the folder called *'pretrained Models'* along with many other samples. (Google Books, n.d.)

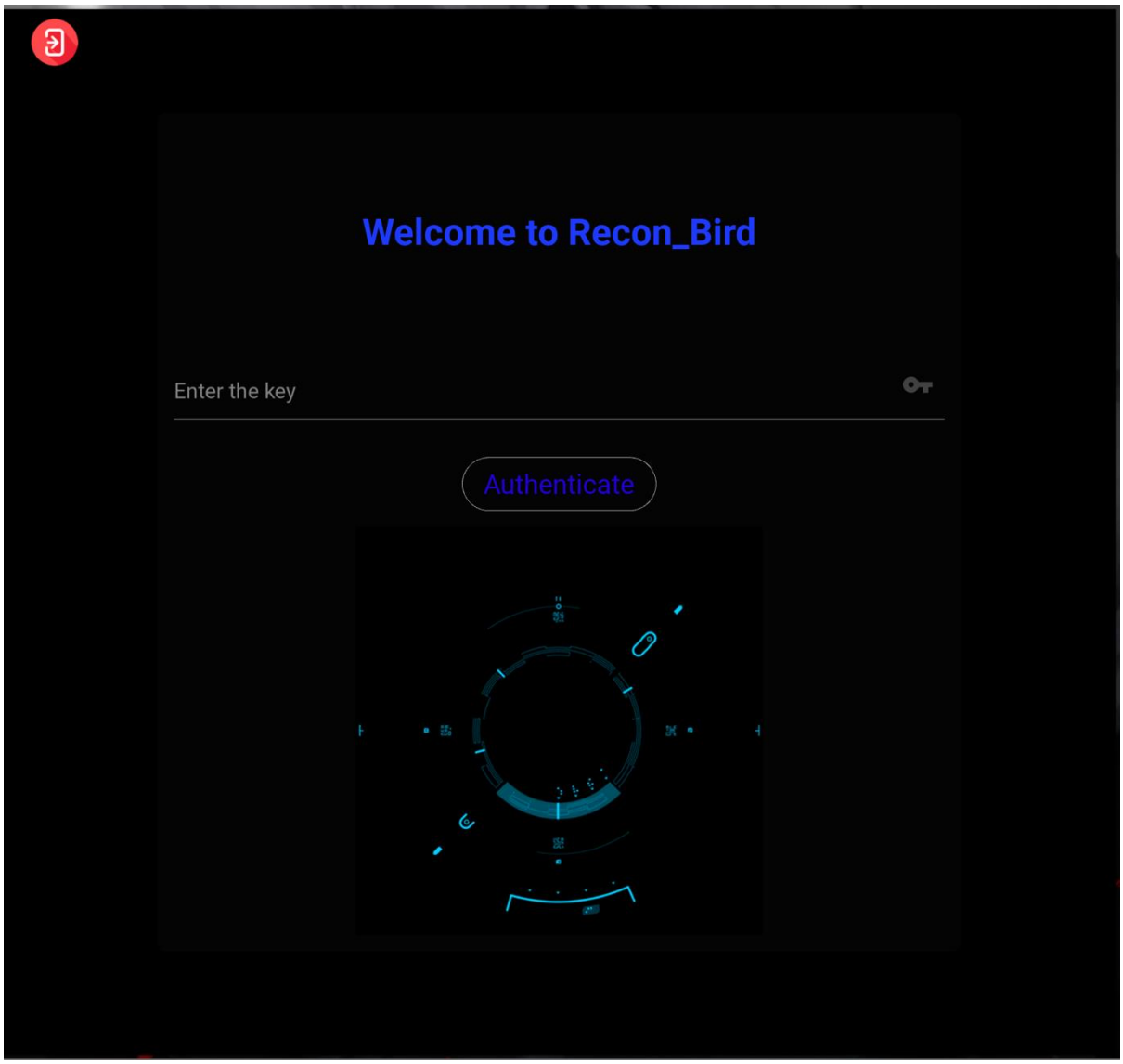
PRODUCT TESTING

Following is the working of the software, proving how the software will appear to the user when using it.

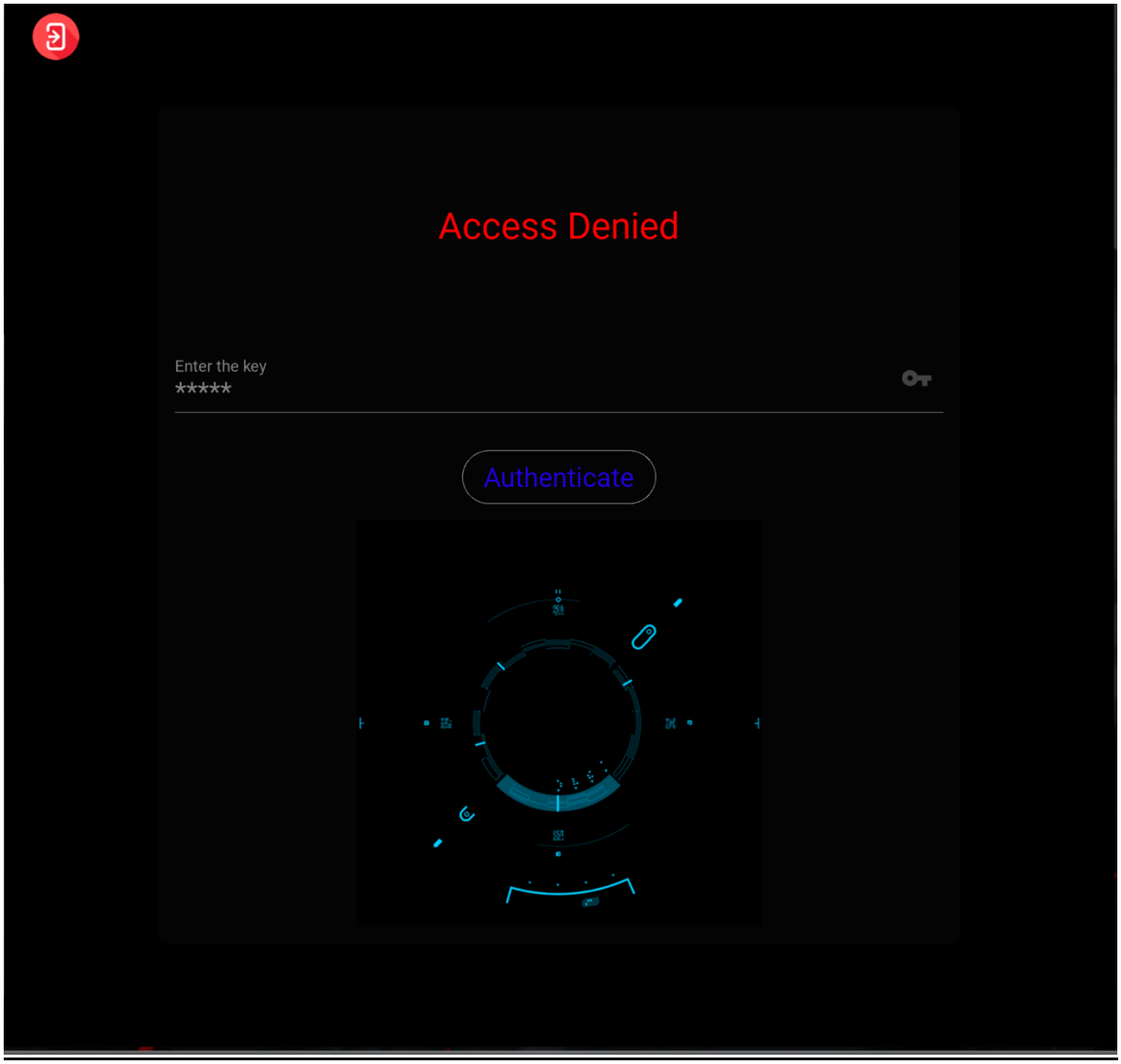
TERMINAL OUTPUT WHEN LOADING THE SOFTWARE FOR CONFIRMATION



WELCOME PAGE

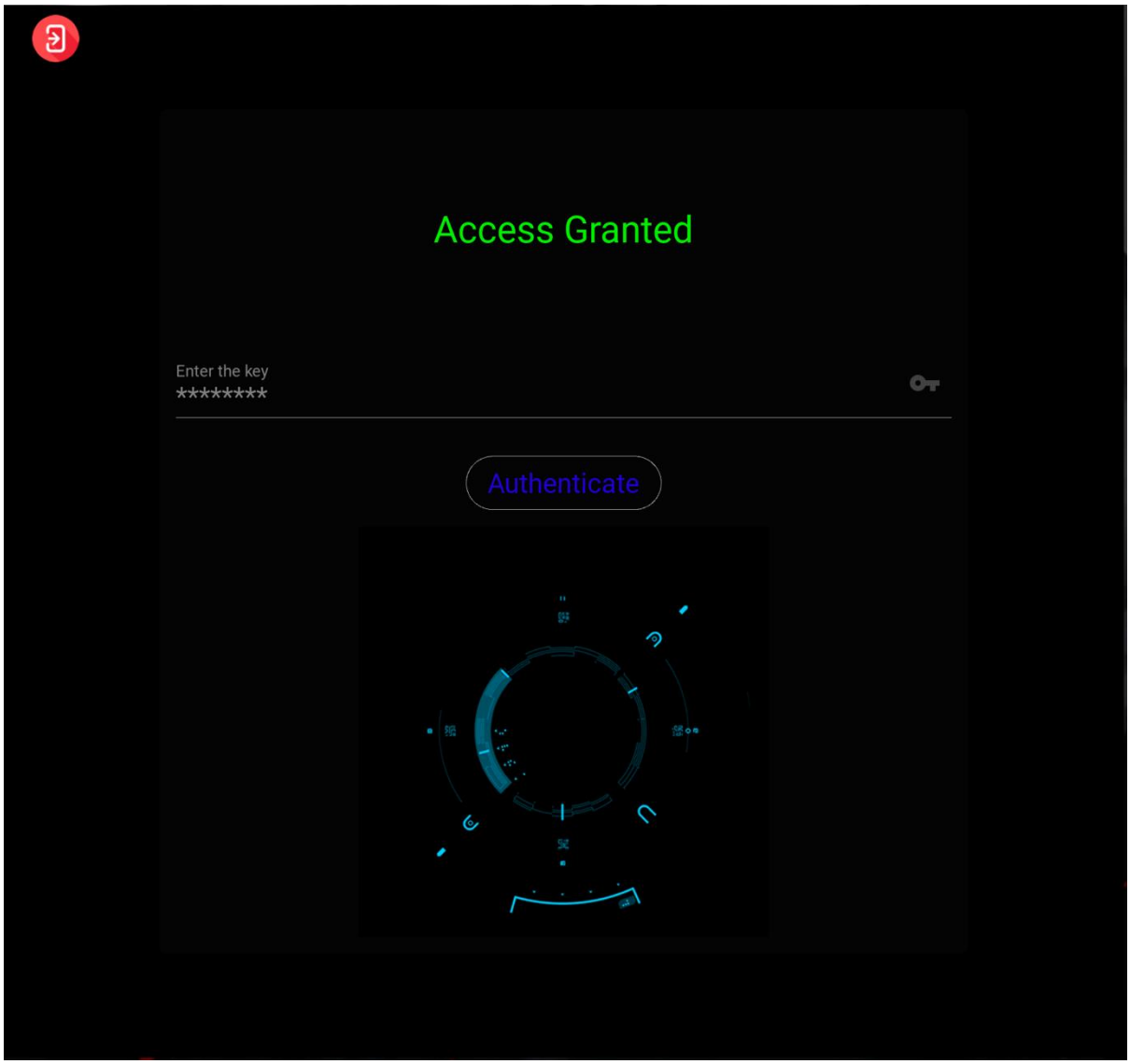


WRONG AUTHENTICATION



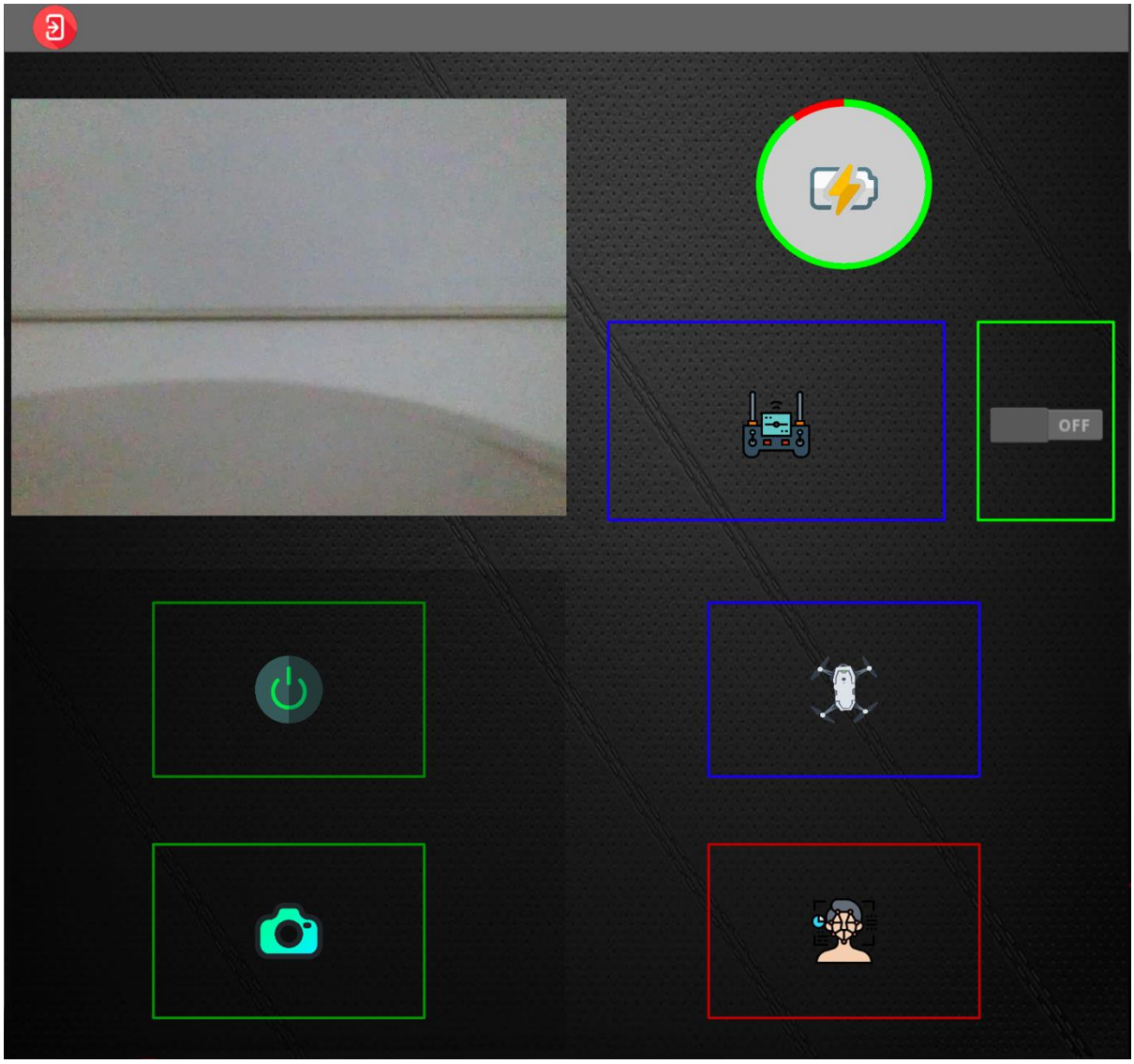
CORRECT AUTHENTICATION

The GUI below loads when the right password is entered, therefore allowing access to the forthcoming page.



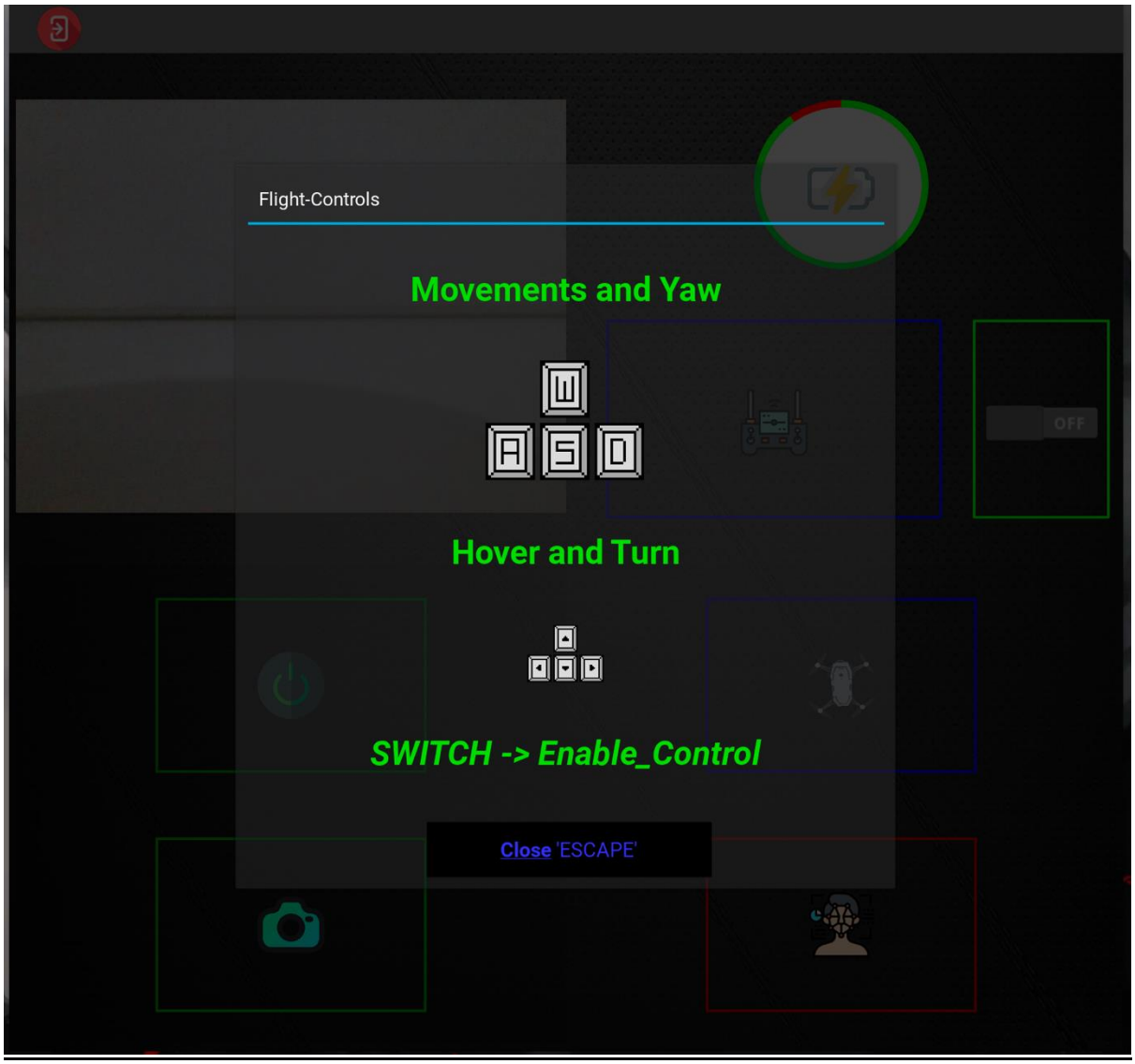
MAIN WINDOW

The battery remaining in the drone is shown by the rounded *battery* label.



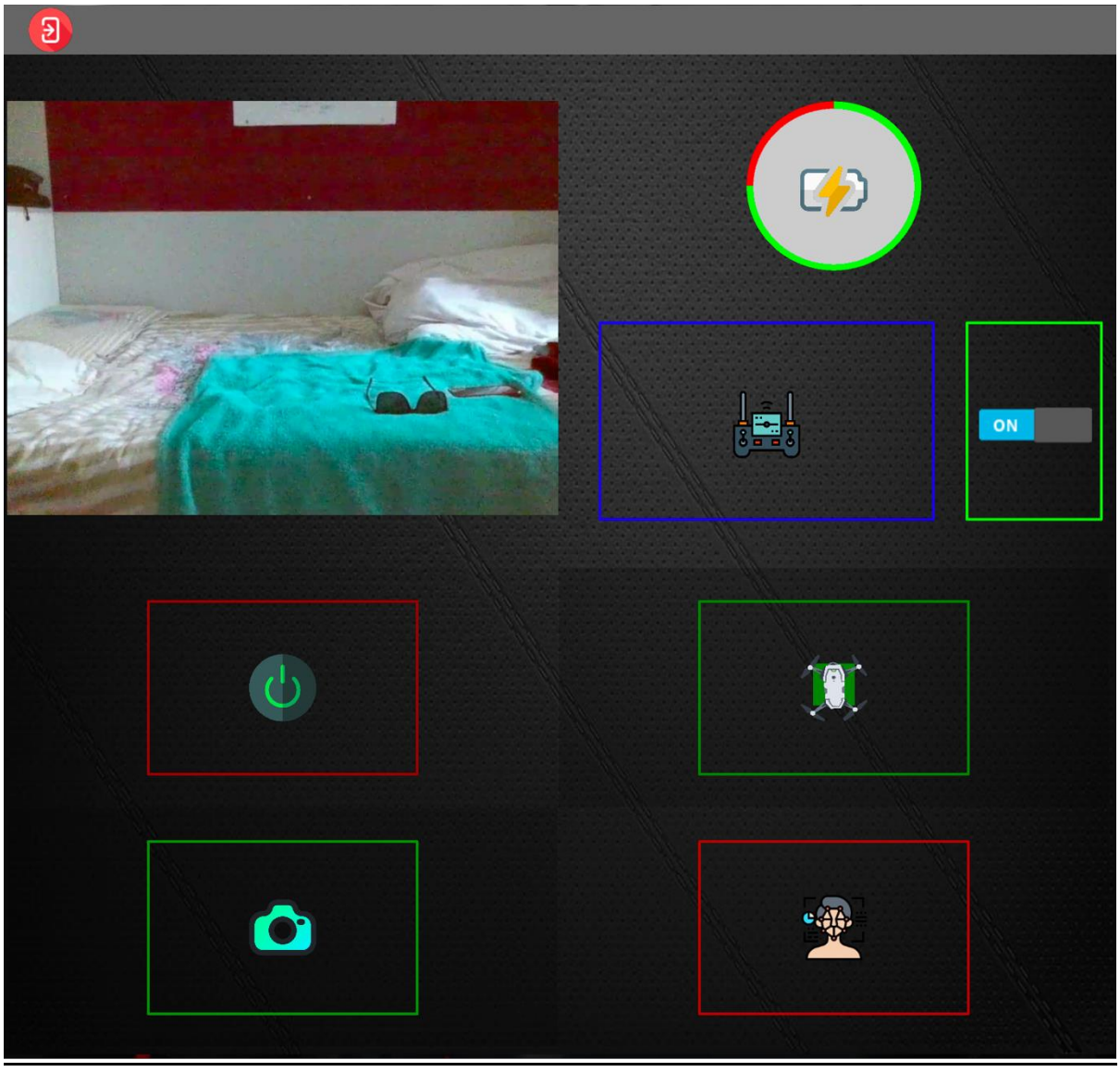
FLIGHT CONTROLS

To understand, click on the *remote* widget in-order to understand the flight controls of the drone-controlling software.

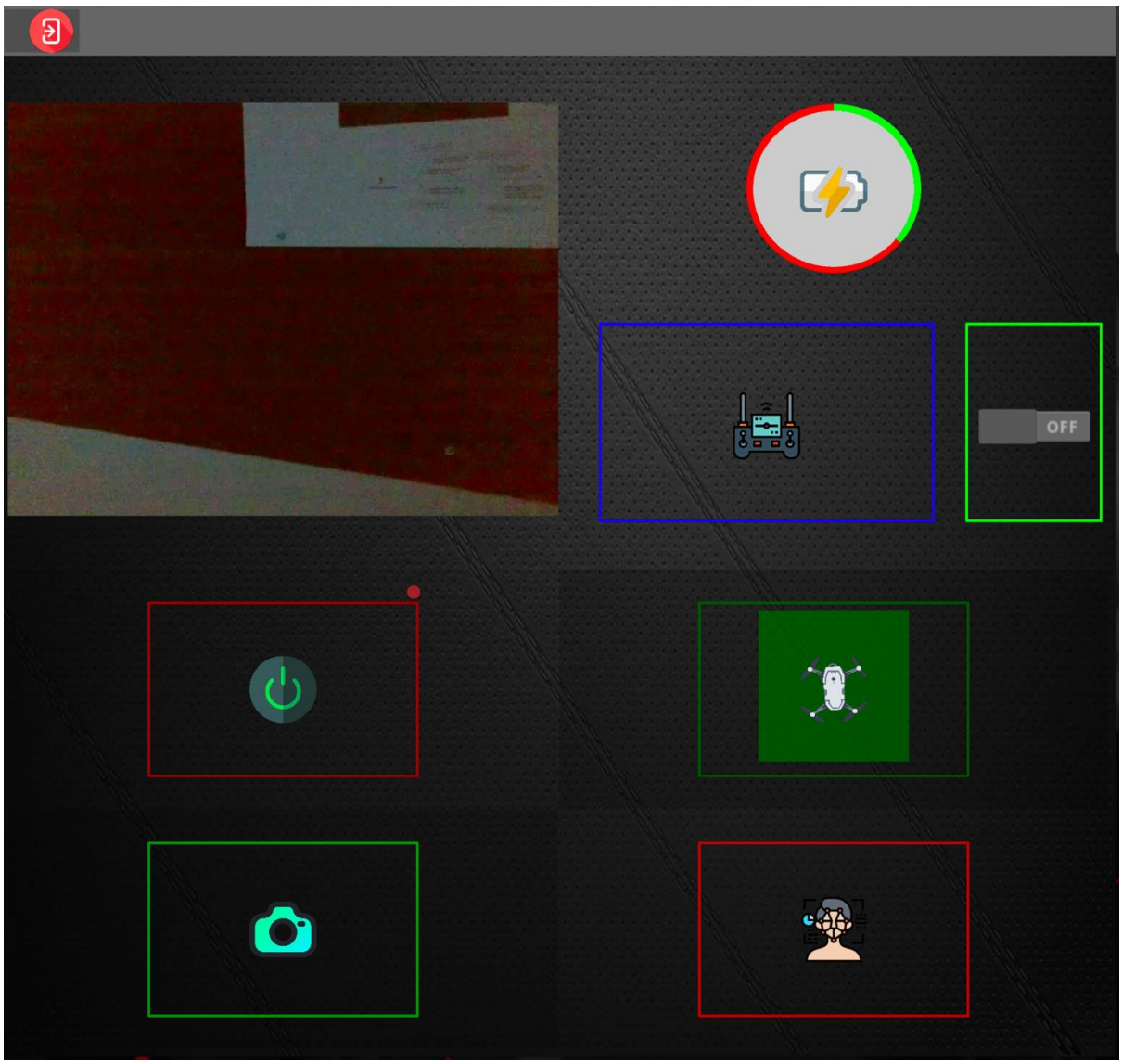


FIRST AIRBORNE FOOTAGE

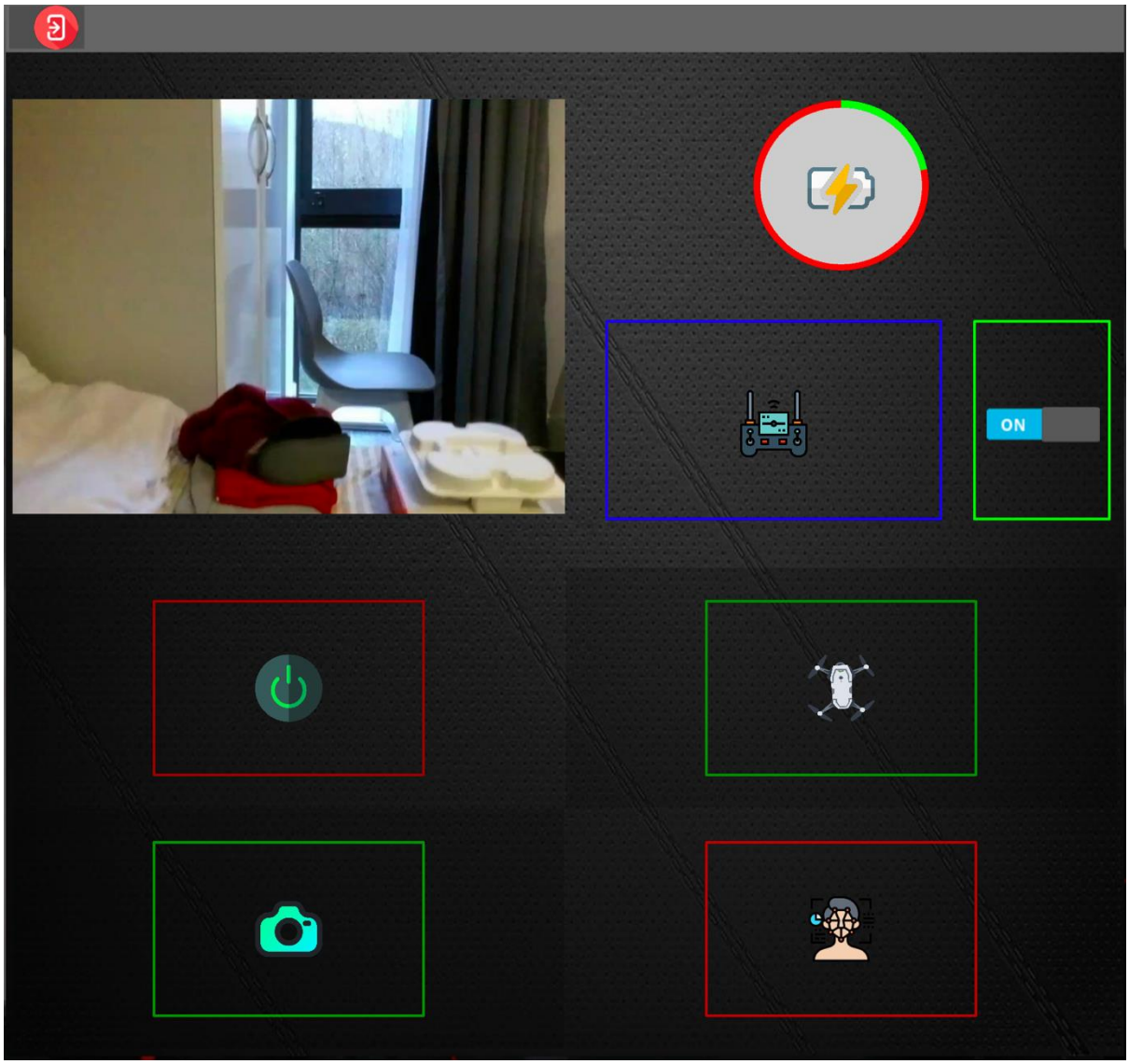
Click on power button for take-off and clicking the switch to enable drone-controls. When airborne, the *drone* widget begins blinking.



TESTING THE MANOEUVRING DYNAMICS

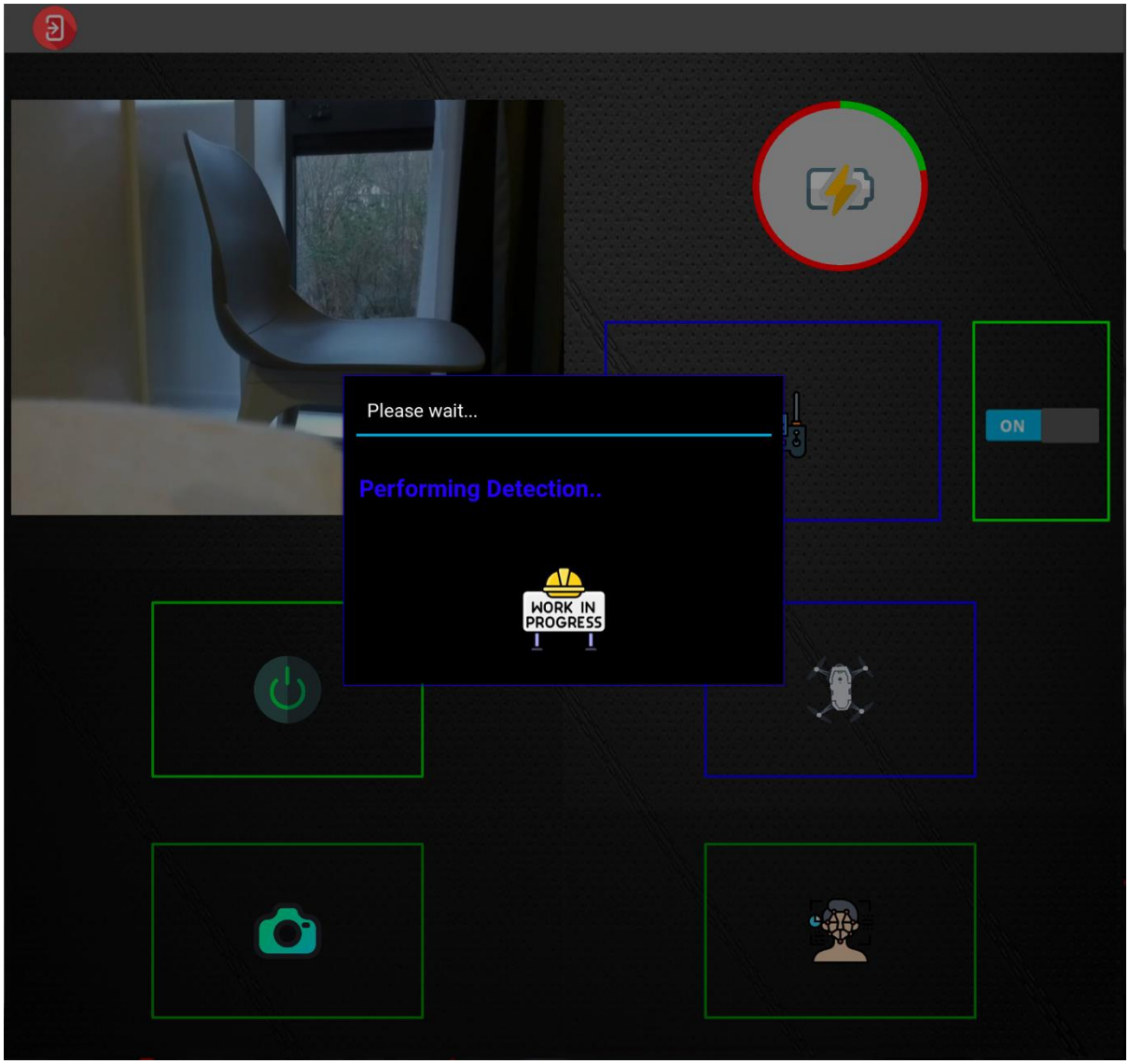


GETTING AN ANGLE TO CAPTURE IMAGE FOR DETECTION



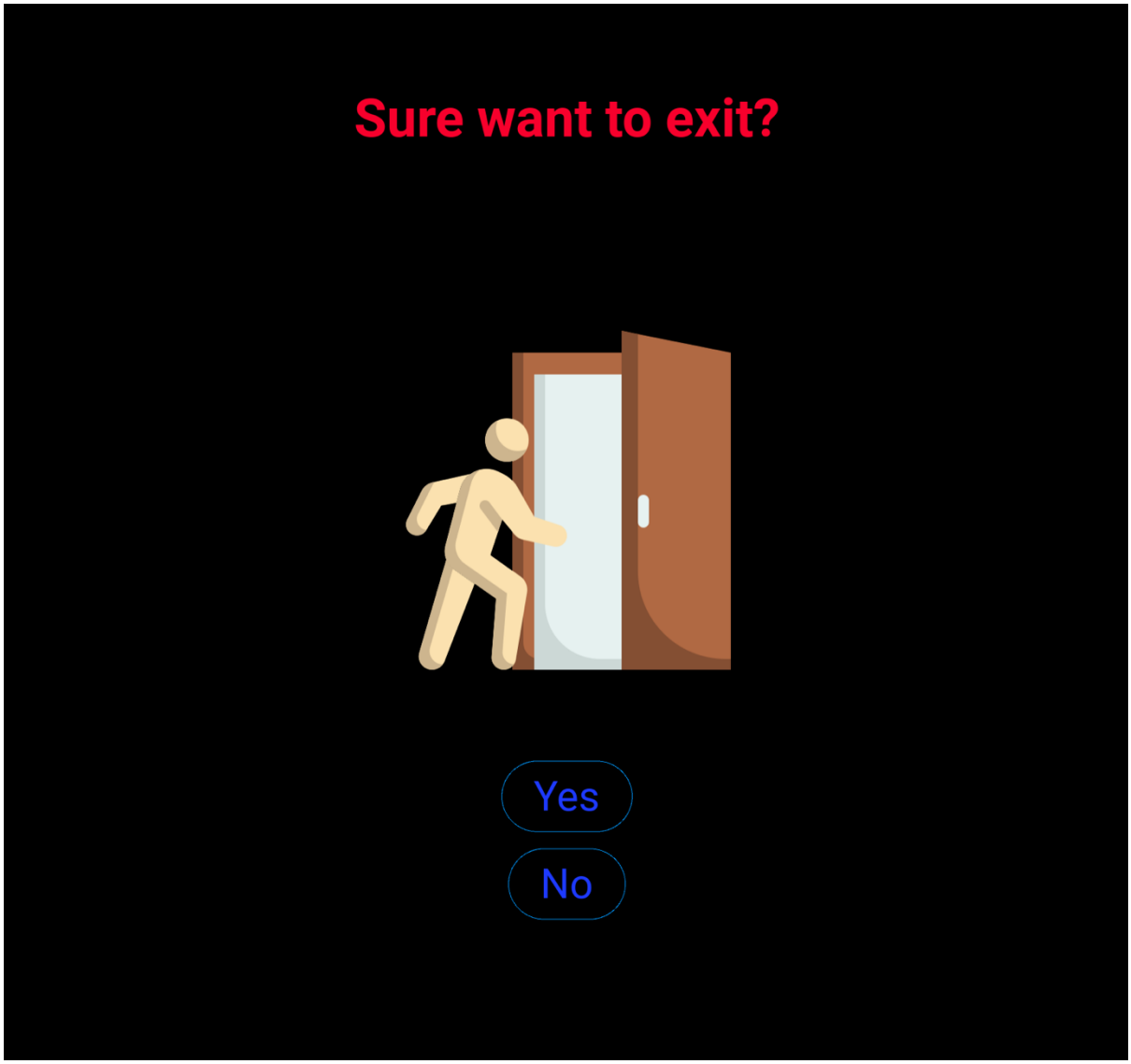
GETTING IMAGE AND PERFORMING OBJECT DETECTION ON IT

After capturing the image, landing the drone first to stabilize it. This is done by clicking on *drone* widget. The *image-detection* widget that becomes available to be clicked is used for performing image-detection.



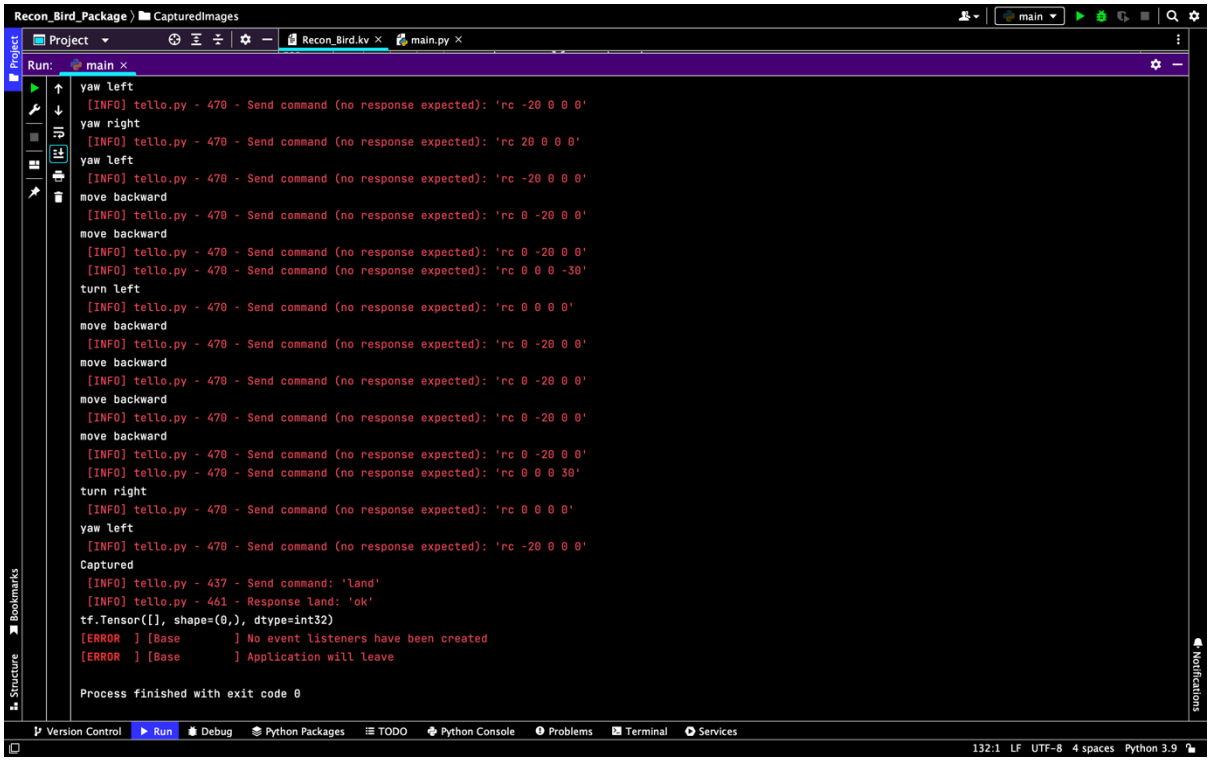
FINAL PAGE OR EXIT PAGE

Clicking on the title-bar corner button in both the windows will direct the user to this page and if clicked 'NO', it will direct the user back to the page they came from otherwise clicking on 'YES' will exit the software.

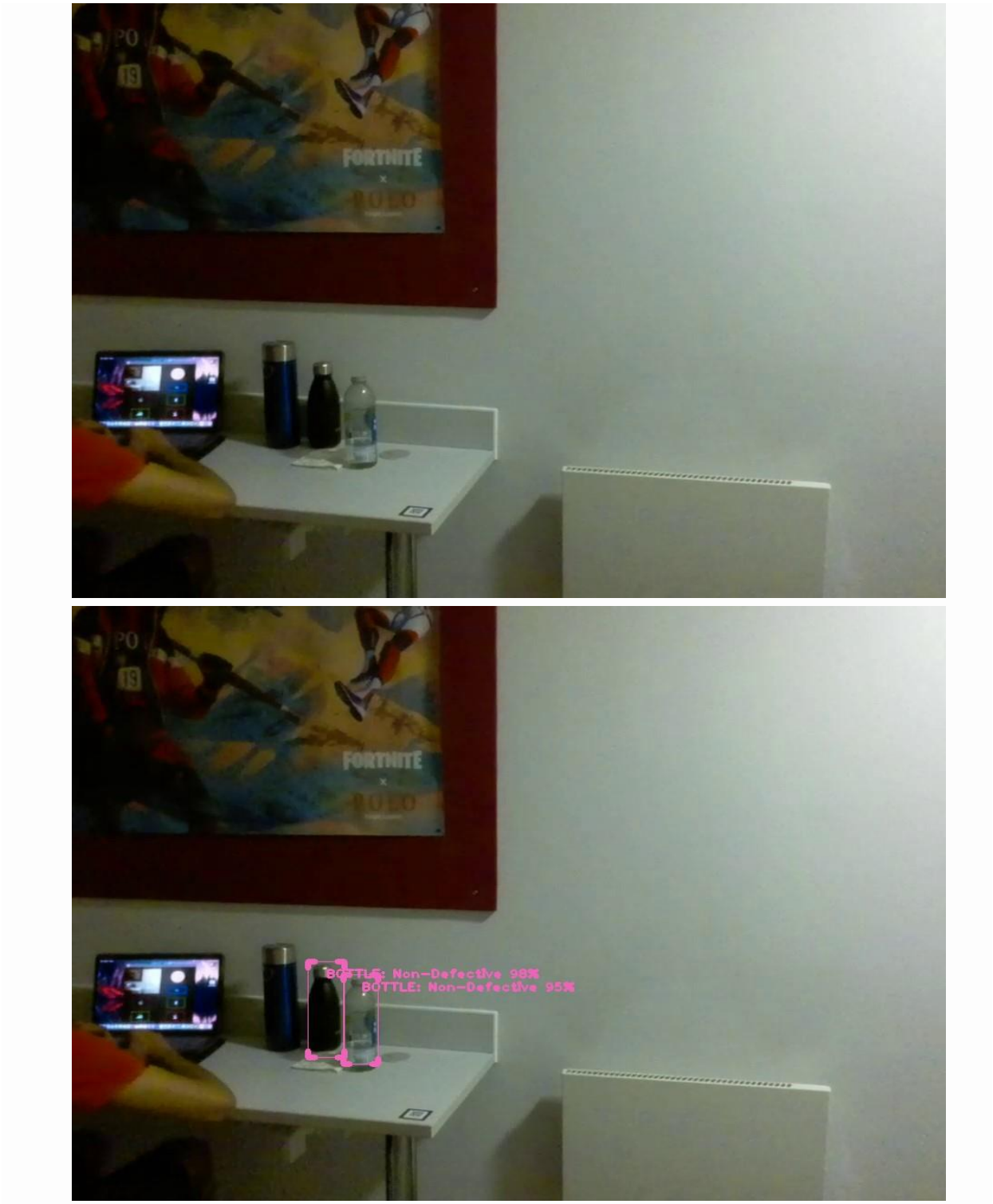


TERMINAL OUTPUT WHEN TESTING THE MANOEUVRE

Here, the program is closed by the user after usage.



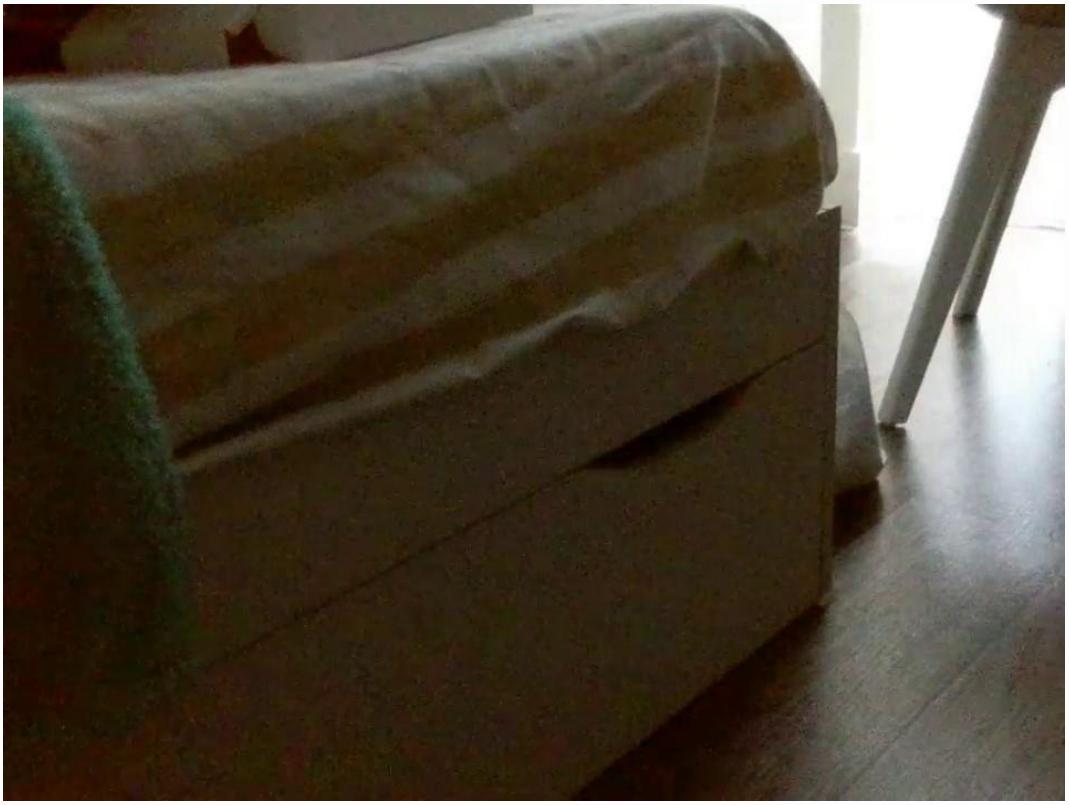
SOME TEST IMAGES CAPTURED BY THE DRONE AND DETECTION PERFORMED ON THEM

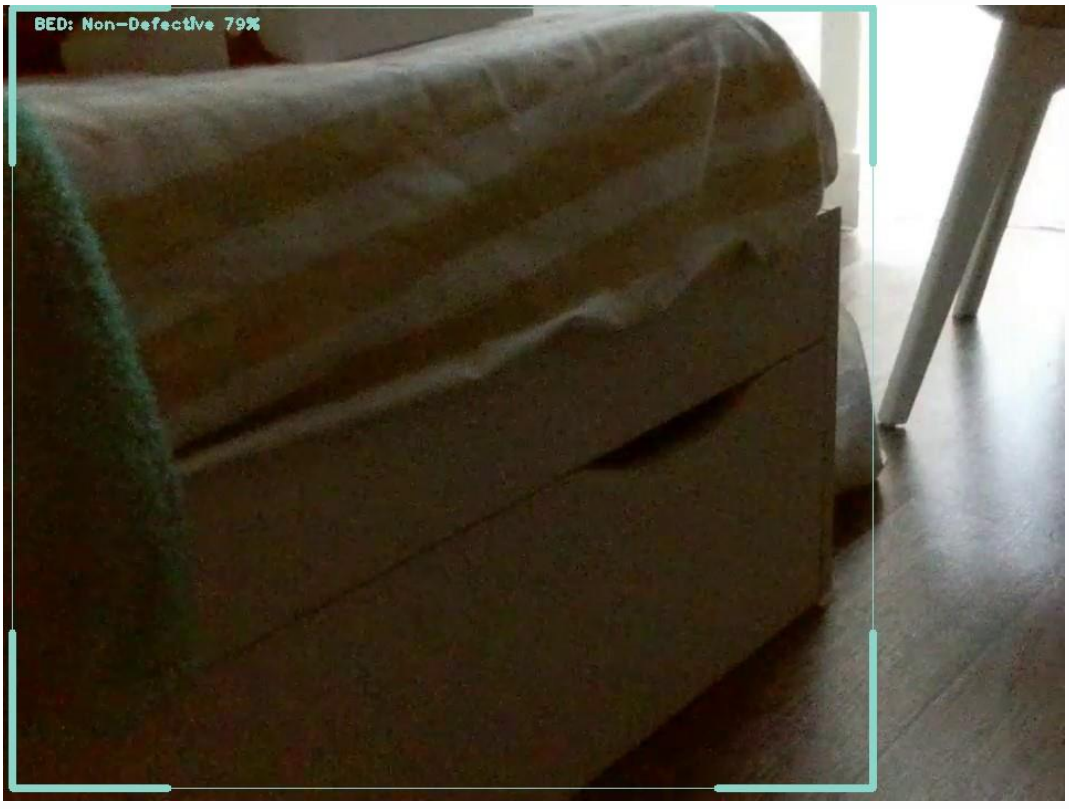


1.

2.

3.





In all the tested images above, the accuracy of the image being detected is clearly visible thus signifying whether the object is actually the intended object that is supposed to be, inside the image.

All the testing from drone was done keeping in mind the laws concerned. Through the domestic testing, the importance of the product was explained. (Overview : Flying Drones and Model Aircraft | UK Civil Aviation Authority, n.d.)

PROJECT EVALUATION

The project seemed to be the most unpredictable one from the beginning. The project at first seemed extremely complicated but when it was broken down into smaller chunks of tasks, it turned out to be an approachable piece of work. Applying the skills acquired during the complete course of study to this project was quite a tedious but imperative task. The complete development of this project required a whole new level of knowledge base, and it was extremely critical that it was done with careful consideration to achieve the highest quality of results.

The greatest challenge initially was to decide on adequate software and tools for its development since it required quite a handful of frameworks to culminate into a well-designed product. But then with the right research and sources, the programming language was finalised and then using the right frameworks, the making of the project seemed very much possible. Then came the phase of scheduling the tasks for getting the product in a user-ready state before the deadline. But all was handled with conscientious planning and thus the result was a working drone-controlling software.

The process of materialising the project required completely diverse kinds of skillsets, particularly in the field of AI and deep learning. Thus, the journey was immensely helpful in developing the required skills for implementing the deep-learning libraries and getting a better understanding of their functionality with much clarity. The biggest advantage was that the object detection model required for this project was openly available and commonly used as a deep learning library and thus it contributed towards building the required skillsets in developing the image-processing software using ‘tensorflow’ and ‘opencv’. Additional requirements included using the kivy library to design the GUI for the software's working.

All the requirements for such a drone-dependent software to have been included in the project with their functionality vividly specified. It was all possible because of the timely execution of all the tasks pre-planned. If required, more windows can be added to the software with additional features for expanding the uses of the software. This has no limit and that is its greatest benefit. This is possible due to the versatility of the programming language '*python*'.

Some other potential features depending on industrial usage that can be included in the future are:

1. Tracking of drone-path

2. Night vision (as the project needs additional light to perform accurate object detection)
3. Auto return to the user when the battery gets low following the path the software has already tracked.

Initially, the project's aims also included tracking the drone's path but later that was scrapped since it posed no purpose for the major aims discussed.

Many libraries are available in this language with more deep-learning models depending on the uses in different working environments. The main GUI window that encloses all the widgets for the functioning of the software was developed in accordance with all the features the library could provide and was utilised up to its maximum potential.

CONCLUSION

SUMMARY

The main idea behind the project was to perform object detection using an open-source computer-vision framework and deep learning model with the purpose of detecting flaws to provide the utmost accuracy in the physical analysis of the object being detected inside an image.

To detect the defect in the object through image processing, the software is designed, and the deep-learning model is trained in a way to load the model from the *URL* provided and store the model that is to be used, inside the folder called '*checkpoints*'. The software uses the drone camera to capture the image during the real-time feed for the purpose of performing object detection on it in real-time. The deep-learning algorithm was made possible by successful training of the model derived from the *tensorflow* framework. (Rodrigues, 2020)

The purpose of the project is well-served by the tools utilised to build it. However, there is no limit to progress. (Girshick et al., 2014) The *GUI* is built with clear understanding of robust requirements that must satiate the user using it. The '*modelURL'* in the code that is used for object detection can currently detect *91-92* classes with utmost accuracy.

POTENTIAL FUTURE USES OF THE SAME CONCEPT

After completing some testing, it was evident that the project can detect the accurate score-confidence for any object the software is capable of detecting. This *score-confidence* is then utilised for manual- analysis to find flaws if any that are recorded by the camera. Overall, the software can obtain real-time feed, perform real-time detection, and has complete controls over the flight controls of the drone used for this project, which is '*DJI Tello-mini'* drone. Definitely, a dream team. (Humans and Drones: A Dream Team, or a Case of "Man vs Machine"? - Technology Insights - PwC UK Blogs, n.d.)

Except for the industrial usages specified in the report, any other usages can be fulfilled by the respective software as there is a drastic upsurge in the need for drone surveillance which will only increase over time and thus this project is built to avoid its oblivion. Thus, it must at least serve for uses specified for as long as the drone mechanisms and language that this project is built on, is in demand.

Considering industrial perspective, the advantages that project provides are: -

1. Improved safety: Through remote visual inspection, workers no longer need to enter hazardous or dangerous zones unnecessarily.
2. Reduced downtime: Since, the drone is ready to be deployed to gather information within minutes, it can help in collecting crucial data of whole area within hours rather than days.
3. Lower cost: As soon as this method is opted for collection of data, traditional equipment such as scaffolding, rope access, or cranes no longer pose a necessary requirement.

Therefore, other important uses that can be fulfilled adequately by the project could be the inspection in coal-fired burners, nuclear power plants, heat recovery system generator, waste incinerator. All such environments can adapt to the updated method of area inspection supporting the safety of manual workforce and safeguarding the continuation of industrial working.

APPENDICES

APPENDIX A: POSTER PRESENTATION

APPENDIX B: ETHICAL REVIEW FORM

GUIDELINES ON COMPLETING THE ETHICAL REVIEW FORM

Questions 1, 3, 4, 6 and 7 only apply if people will be involved in your project (e.g. in providing feedback on your product, in providing input on desirable aspects of your project and/or product (e.g. through questionnaires, focus groups, etc.), etc.)

Question 2 will probably only apply if you have a real client for your project.

Questions 5 and 8 should be fairly self-evident.

Before completing question 9 you need to complete a Risk Analysis & Management form, which is attached at the end of this document, to identify risks and hazards that may arise from your project, if any.

Please have a look at the School's research ethics and integrity web pages.

Professional, Legal, Ethical and Social Issues

As part of your literature review in December you will be required to discuss professional, legal, ethical and social issues pertaining to your project. Use the space below to draft a first list of the issues you might need to consider:

UNIVERSITY OF HUDDERSFIELD
SCHOOL OF COMPUTING AND ENGINEERING

PROJECT ETHICAL REVIEW FORM

Applicable for all research, masters and undergraduate projects

Project Title:	Drone Tracking
Student:	Shivang Chaudhary
Course/Programme :	BSc. Computer Science
Department:	Computer Science
Supervisor:	Dr. Tianhua Chen
Project Start Date:	10/09/2022

ETHICAL REVIEW CHECKLIST

	Yes	No
1. Are there problems with any participant's right to remain anonymous?	☐	☑
2. Could a conflict of interest arise between a collaborating partner or funding source and the potential outcomes of the research, e.g. due to the need for confidentiality?	☐	☑
3. Will financial inducements be offered?	☐	☑
4. Will deception of participants be necessary during the research?	☐	☑
5. Does the research involve experimentation on any of the following?		
(i) animals?	☐	☑
(ii) animal tissues?	☐	☑
(iii)human tissues (including blood, fluid, skin, cell lines)?	☐	☑
6. Does the research involve participants who may be particularly vulnerable, e.g. children or adults with severe learning disabilities?	☐	☑
7. Could the research induce psychological stress or anxiety for the participants beyond that encountered in normal life?	☐	☑
8. Is it likely that the research will put any of the following at risk:		
(i) living creatures?	☐	☑
(ii) stakeholders (disregarding health and safety, which is covered by Q9)?	☐	☑
(iii)the environment?	☐	☑
(iv)the economy?	☐	☑
9. Having completed a health and safety risk assessment form and taken all reasonable practicable steps to minimise risk from the hazards identified, are the residual risks acceptable (Please attach a risk assessment form – available at the end of this document)	☑	☐

STATEMENT OF ETHICAL ISSUES AND ACTIONS

If the answer to any of the questions above is yes, or there are any other ethical issues that arise that are not covered by the checklist, then please give a summary of the ethical issues and the action that will be taken to address these in the box below. If you believe there to be no ethical issues, please enter "NONE".

STATEMENT BY THE STUDENT

I believe that the information I have given in this form on ethical issues is correct.

Signature: Shivang Chaudhary Date: 08/12/2022

AFFIRMATION BY THE SUPERVISOR

I have read this Ethical Review Checklist and I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow an informed judgement on whether further ethical approval is required.

Signature: _____ Date: _____

SUPERVISOR RECOMMENDATION ON THE PROJECT'S ETHICAL STATUS

Having satisfied myself of the accuracy of the project ethical statement, I believe that the appropriate action is:

The project proceeds in its present form	
The project proposal needs further assessment by an Ethical Review Panel. The Supervisor will pass the form to the Ethical Review Panel Leader for consideration.	

RETENTION OF THIS FORM

- The Supervisor must retain a copy of this form until the project report/dissertation is produced.
- The student must include a copy of the form as an appendix in the report/dissertation.

OUTCOME OF THE ETHICAL REVIEW PANEL PROCESS, WHERE REQUIRED

	Tick One
1. Approved. The ethical issues have been adequately addressed and the project may commence.	☐
2. Approved subject to minor amendments. The required amendments are stated in the box below. The project may proceed once the form has been amended in line with the requirements and signed by the Supervisor in the box imediately below to confirm this.	☐

I confirm, as Supervisor, that the amendments required have been made:

Signature: _____ Date: _____

3. Resubmit. The areas requiring further action are stated in the box below. The project may not proceed until the form has been resubmitted and approved.	☐
4. Reject. The reasons why it will not be possible to address the ethical issues adequately are stated in the box below.	☐

For any of the outcomes 2, 3 or 4 above, please provide a statement in the box below.

AFFIRMATION BY THE REVIEW PANEL LEADER

I approve the decision reached above by the review panel members:

Signature: _____ Date: _____

THE UNIVERSITY OF HUDDERSFIELD: RISK ANALYSIS & MANAGEMENT

ACTIVITY:			Name:	
LOCATION:			Date:	Review Date:
Hazard(s) Identified	Details of Risk(s)	People at Risk	Risk management measures	Other comments

APPENDIX C: HOW PYTHON-KIVY PLAYED A SIGNIFICANT ROLE AS A FRAMEWORK

The *'kv'* language is a dedicated language used for creating UI in python. It has been the heaviest part of this project after embedding the deep learning module by creating a class file for loading Tensorflow Module and running it once downloaded. The operations are a crucial part of this project to interact with the drone either in a static or stable state. Moreover, other parts of the kivy widgets were modified accordingly to serve the purpose of providing elegant presentable widgets to the user when operating the window.

Following are the widgets in-built which played a pivotal role in building the UI for the project:

1. **Button:** This provides a widget that has a primary function to perform an action. Any action might that be. For that to execute, a function must be defined and declared in the correspondent class in 'py' file. This function then must be declared in the *'on_press'* method of the respective in-built widget so that the action can be executed once the button is pressed. Similarly, an action can be designed when the button is released for which the in-built property of the widget is *'on_release'*. The look of the button can be changed by adding an image with a source path for the image to be present in the folder. This way the button retains its properties, but its appearances change.

2. **Camera Module:** This module was essential for getting the live feed from the drone. This was a difficult but possible procedure to get the image-texture so that it can be displayed on the kivy window. In this project, the class is named "TELLO_CAM" based on the TELLO drone being used for the project. The 'texture-image' that this module displays in real-time is also returned by the class to capture them using the *'capture-button'* present on the window.

3. **Label:** It serves the basic purpose of showing a message and nothing more. But, in this project, it was customised to its full-fledged use to maximise the presentation. The only label used in the project is battery-label which uses three ellipses, one of each green, red, and grey colour. The last ellipse which is the grey colour has the sole purpose of setting an image to highlight that the label shows the battery percentage of the drone which is received from the drone using its own in-built *'tellopy'* library. When the battery is full, only the first ellipse which is green-coloured, and the third ellipse is active meaning the drone has not lost any battery, but the second ellipse is set to new measurements when the battery starts depleting.

4. **KivyMD :** It is a newer version of Kivy with some additional properties. This was primarily used for making up the first window. The ‘kivyMD Card’ serves as an additional layout to present all the widgets it contains in rectangular format. Its shadow effect can be modified by inputting a figure in the in-built property '*elevation*' so that the ‘*Card*’ stands out in the window. In this project, it was used to store a rounded button for authenticating the user, obscured label for inputting the password or 'key' as stated in the code. It contains one Label as well to display a welcome message and to display whether the authentication was successful or not. This was also used in the *exit_page* for displaying an exit-image signifying it is the last window and two buttons stating 'Yes' or 'No' providing the user with final options to either stay on the software or leave.

5. **Switch:** As the name suggests, this provides the functionality of enabling a property given that the ‘py’ function was declared in its *'on_active'* property. When switched on, it can execute any action given to it and the same way the action is disabled when switched off. In this project, it was used for enabling drone controls.

6. **Custom Title-bar:** To materialize and thus finalize the measurements of the window being used for the software, the title-bar was removed, and a customised title-bar was used using ‘*Action-bar’* in Kivy for customising the appearance of title-bar. Along with it, the exit button was added on the title-bar which is used to navigate the user to the exit-page. It includes the possibility of disabling the exit-button when required and it was useful in prohibiting the closing of software when the drone is airborne thus serving as a good contingency feature.

7. **Kivy CANVAS:** Canvas has the usage as a root object for drawing by a widget. The project had various uses such as setting the background to every window, setting the border-colours to the widgets and most creatively from a technical perspective in the animation it executes. The property provides a lot of possibilities to add some visual character to every button also signifying that some function is performed by it and that button is unique.

8. **Layout:** The layout has primary usage, without this, any widget declared in the 'kv' file will not have any functionality. The layout must be declared to specify the order of

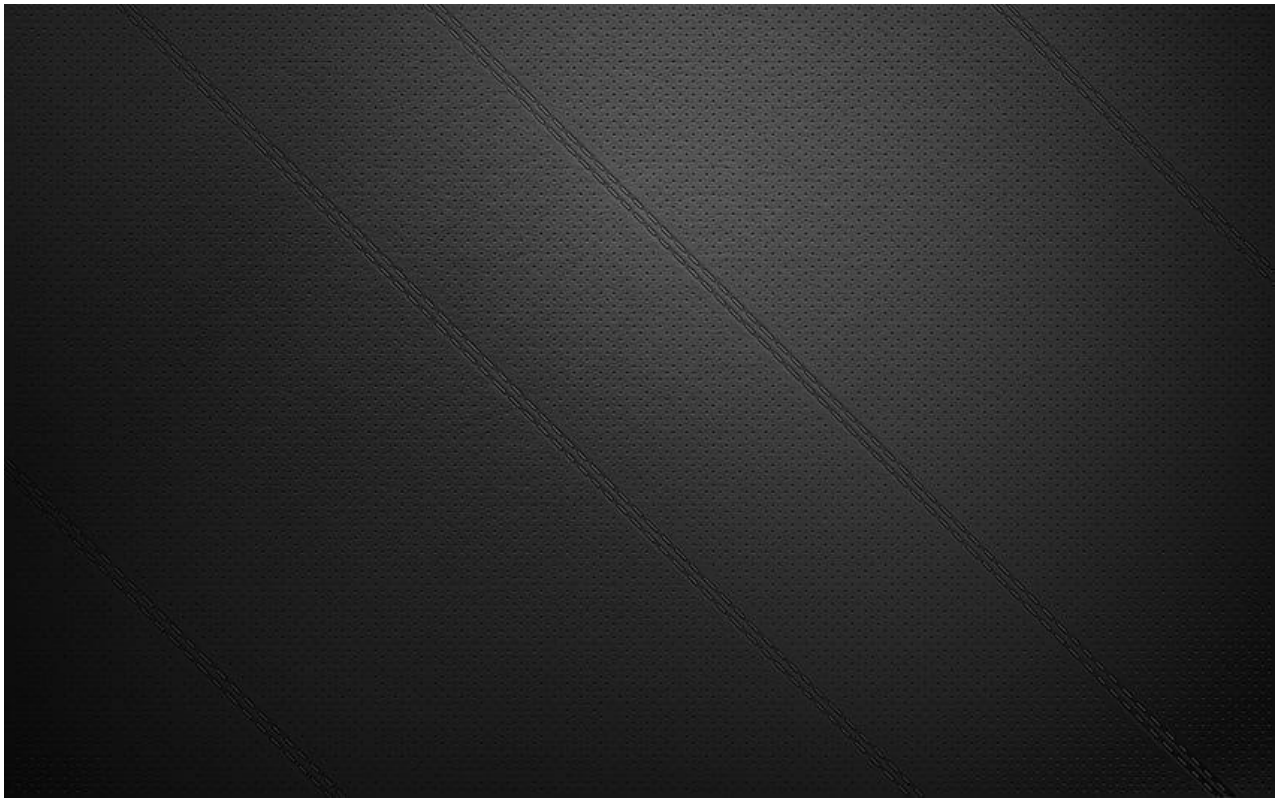
arrangement of any widget declared in the file so that they can be adjusted well in the window.

9. The options for the layout are: -

- **AnchorLayout** : For anchoring the layout in any part of the window. Example: 'Top', 'Bottom', 'Right' or 'Left'.
- **BoxLayout** : Arrangements of widgets are made in a sequential manner, either in vertical or horizontal format.
- **FloatLayout** : Widgets are unrestricted.
- **RelativeLayout** : Child widgets are declared in a relative position of the layout.
- **GridLayout** : Arrangement of widgets is done in a manner of the grid such as row and column.
- **ScatterLayout** : Here, the positioning of the widgets is the same as Relative Layout, except for they can be rotated, translated, or scaled.
- **StackLayout** : Widgets are stacked. In left to right or top to bottom manner.

The layout used most in this project are Box Layout and Grid layout which served the purpose well.

APPENDIX D: BACKGROUND IMAGE FOR PROJECT

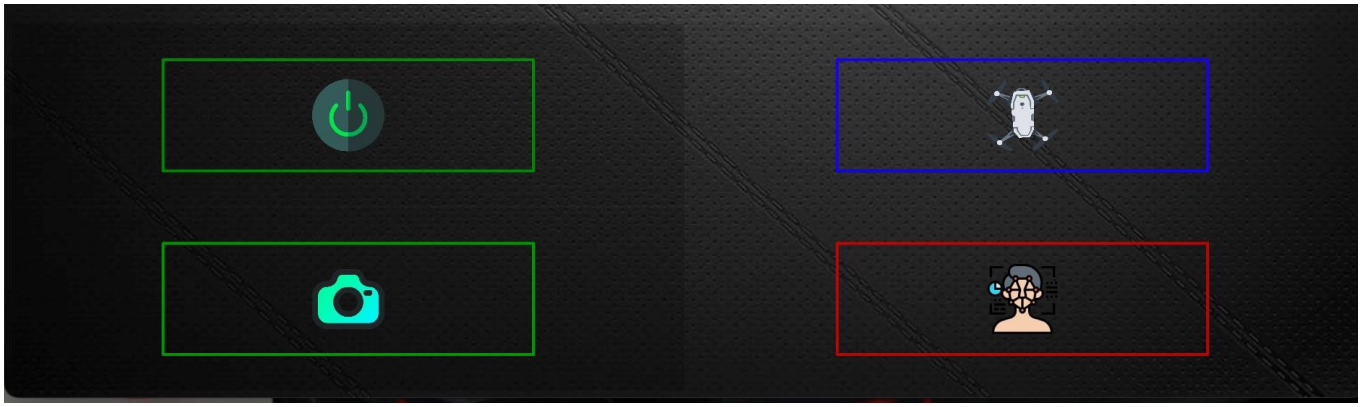


APPENDIX E: CANVAS SOURCE CODE

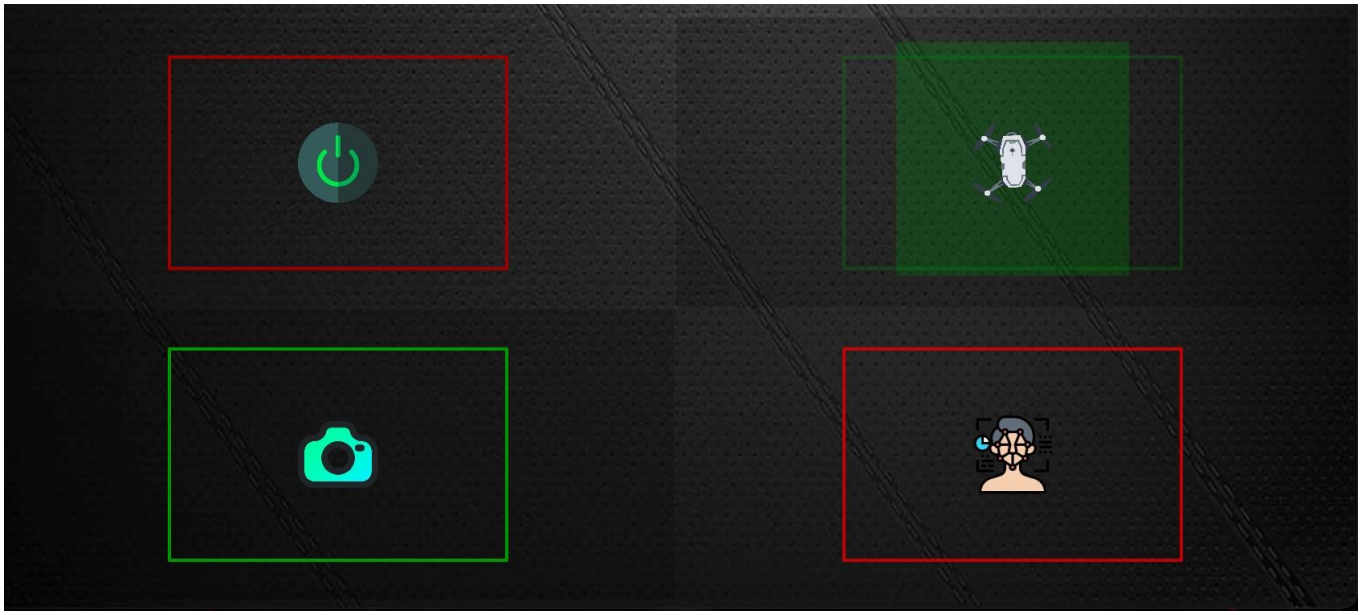
```
canvas.before:
    Color:
        rgba: self.animated_color
    Rectangle:
        size: self.blink_size, self.blink_size
        pos: self.pos[0] + self.pos[0]/2. - self.blink_size/2. -20 , self.pos[1] + self.pos[1]/2. - self.blink_size/2.-270
    Line:
        width: 4
        rectangle: self.x+198, self.y+35, self.width-395, self.height-40

Image:
    source: 'remote-control.png'
    center_x: self.parent.center_x
    center_y: self.parent.center_y
```

APPENDIX F: INITIAL GRAPHIC RESULTS

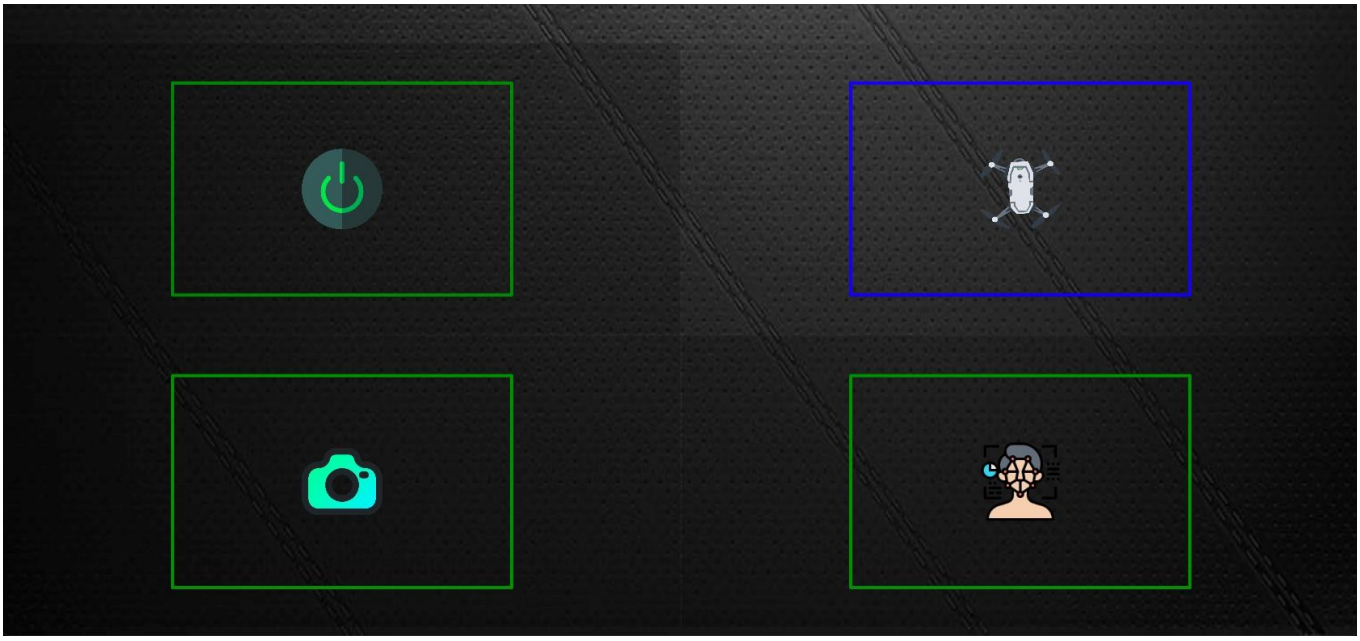
Initial results: -



APPENDIX G: FINAL GRAPHIC RESULTS



APPENDIX H: ENABLED BORDER WIDGETS

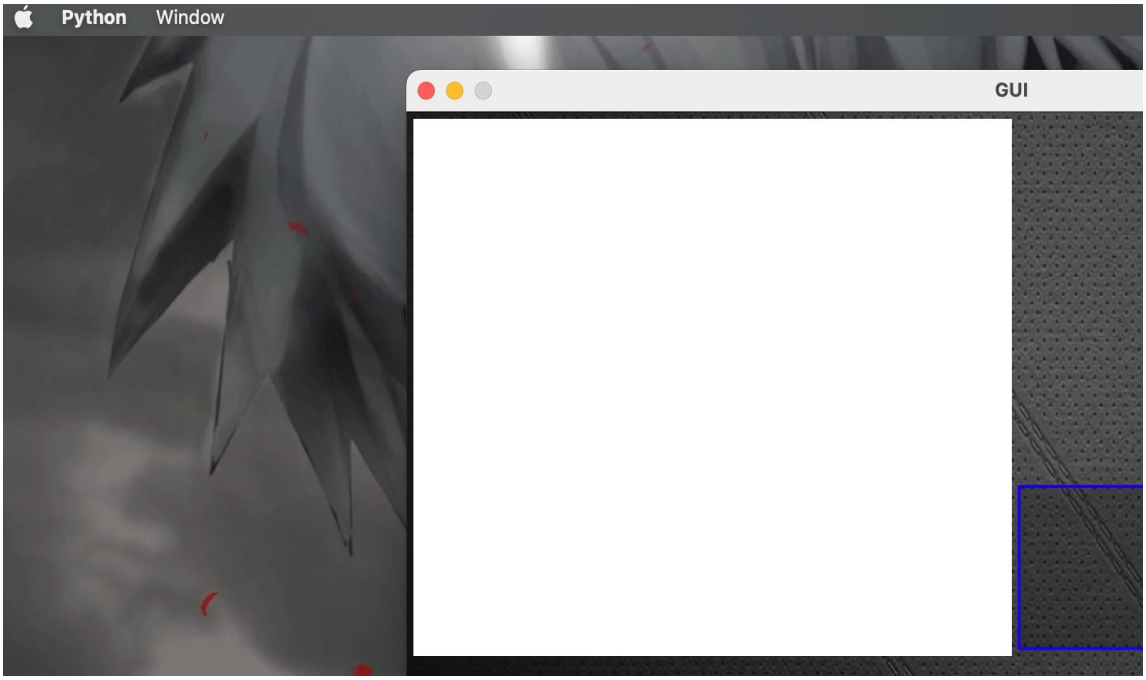


APPENDIX I: TELLO-CAMERA CODE FILES

```
GridLayout:
    padding: 10,10
    cols:2

    TelloCamera:
        id: camera
        size: self.size
        center_x: self.parent.center_x
        center_y: self.parent.center_y
```

```
Getting the feed from the drone to display in the main-window:-
class TelloCamera(Camera):

    def __init__(self, **kwargs):
        super(TelloCamera, self).__init__(**kwargs)

        # Set the resolution of the camera
        self.resolution = (640, 480)

        # Setting the callback function to process the video frames
        Clock.schedule_interval(self.process_frame, 1.0 / 30.0)

    def process_frame(self, dt):
        # Getting the video stream from the drone
        frame = me.get_frame_read().frame

        # Converting the video stream to a format which can be displayed by Kivy GUI
        img = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
        img = cv.flip(img, 0)

        # Update the image in the Kivy GUI
        texture = Texture.create(size=(img.shape[1], img.shape[0]), colorfmt='rgb')
        texture.blit_buffer(img.tobytes(), colorfmt='rgb', bufferfmt='ubyte')
        self.texture = texture
        # returning the texture-image so that it can be saved and processed through a widget.
        return self.texture


create a background class which uses the boxlayout
class Background(Screen, ButtonBehavior):

    click_disabled = BooleanProperty(True)
    power_button_enabled = BooleanProperty(True)
```

APPENDIX J: CHALLENGE IN DESIGNING IMAGE WIDGET




```
<Background>:
    BoxLayout:
        id: main_win
        orientation: "vertical"
        spacing: 10
        space_x: self.size[0]/3

        canvas.before:
            Color:
                rgba: (1, 1, 1, 1)
            Rectangle:
                source:'leather.jpeg'
                size: root.width, root.height
                pos: self.pos

        GridLayout:
            padding: 10,10
            cols:2

            #Camera:
             #   id:camera
              #  resolution: (640,480)
               # size_hint: (1,1)
                #play: True

            Image_cam:
                #source: root.build_img()
                #capture: App.Image_cam.build_img()
                center_x: self.parent.center_x
                center_y: self.parent.center_y

            GridLayout:
                padding:10,10
                cols:1

                Label:
```

REFERENCES

- Coglianese, C., & Lehr, D. (2019). TRANSPARENCY AND ALGORITHMIC GOVERNANCE. *Administrative Law Review*, *71*(1), 1–56. https://www.jstor.org/stable/27170531?casa_token=E1OSl3bLJLUAAAAA%3A8li1z58P2GWyvHhEkpEwlhkByU9I93-dbUH57NY_WQrRb_kFEnFwxCoTMyob2iqLo2u1YLIURme6Gy75aYP7DAYSBkPMkIFiTGofiFDW0sZ4jWMhq64
- Ethical Issues in Computer Vision and Strategies for Success, https://innodata.com/ethical-issues-in-computer-vision-and-strategies-for-success/.
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*. Openaccess.thecvf.com. https://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html
- Guan, L. (2017). Multimedia Image and Video Processing.In *Google Books*. CRC Press. https://books.google.co.uk/books?hl=en&lr=&id=eTfNBQAAQBAJ&oi=fnd&pg=PP1&dq=Multimedia+Image+and+Video+Processing&ots=i357oM1mhQ&sig=An26HSghyR0LXUJBoGV--KpG2bY&redir_esc=y#v=onepage&q=Multimedia%20Image%20and%20Video%20Processing&f=false
- *Humans and drones: A dream team, or a case of "man vs machine"? - Technology **Insights - PwC UK blogs***. **(n.d.). Pwc.blogs.com. https://pwc.blogs.com/technology-insights/2020/10/humans-and-drones-a-dream-team-or-a-case-of-man-vs-machine.html**
- Image Recognition and Classification in Python with TensorFlow and Keras, https://stackabuse.com/image-recognition-in-python-with-tensorflow-and-keras
- Koelle, M., Lindemann, P., Stockinger, T., & Kranz, M. (2014). *Human-Computer Interaction with Augmented Reality Advances in Embedded Interactive Systems*. ("2014 - Human-Computer Interaction With AR | PDF - Scribd")

http://www.eislab.fim.uni-passau.de/files/publications/201 4/TR2014-HCIwithAR_1.pdf

- Li, J. (2021, November 12). *How Artificial Intelligence helps make drone image processing smarter*. GeoNadir. https: //geonadir.com/drone-image-processing/
- Moroney, L. (2020). AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence. In *Google Books*. O'Reilly. https://www.google.co.uk/books/edition/AI_and_Machine_Learning_for_Coders/462OzQEACAAJ?hl=en
- Object Detection from the Video Taken by Drone via ... - Hindawi, https://doi.org/10.1155/2020/4013647
- Pulli, K., Baksheev, A., Kornyakov, K., & Eruhimov, V. (2012). Real-time computer vision with OpenCV. *Communications of the ACM*, *55*(6), 61. https://doi.org/10.1145/2184319.2184337
- Dewangan, R. K., & Chouhan, Y. (2020). *A Review on Object Detection using Open CV Method. International Research Journal of Engineering and Technology (IRJET), 7(09), 3859-3861.* https://www.academia.edu/download/64790044/IRJET_V7I9677.pdf
- Ramsundar, B., & Zadeh, R. B. (2018). TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning. In *Google Books*. "O'Reilly Media, Inc." https://books.google.co.uk/books?hl=en&lr=&id=GZ1ODwAAQBAJ&oi=fnd&pg=PA8&dq=TensorFlow+for+Deep+Learning&ots=mrIYwbnYUn&sig=DdIEsFw5ER0aLI9R6GgfAqtW7BA&redir_esc=y#v=onepage&q=TensorFlow%20for%20Deep%20Le arning&f=false
- Rodrigues, R. (2020). Legal and human rights issues of AI: gaps, challenges and vulnerabilities. *Journal of Responsible Technology*, *4*(100005), 100005. sciencedirect. https://doi.org/10.1016/j.jrt.2020.100005
- Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). *CNN Features Off-the-Shelf: An Astounding Baseline for Recognition*. Www.cv-Foundation.org. https://www.cv-foundation.org/openaccess/content_cvpr_

workshops_2014/W15/html/Razavian_CNN_Features_Off-the-Shelf_2014_CVPR_paper.html

- Virtual Sketch using Open CV, https://doi.org/10.35940/ijitee.h9262.0610821
- Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2021). *Scaled-YOLOv4: Scaling Cross Stage Partial Network*. Openaccess.thecvf.com. https://openaccess.thecvf.com/content/CVPR2021/html/Wang_Scaled-YOLOv4_Scaling_Cross_Stage_Partial_Network_CVPR_2021_paper.html?ref=https://githubhelp.com
- What Is Object Detection? Importance, Models and Types - G2, https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi-lNvzr8P-AhUHYcAKHeYrA7wQFnoECAsQAQ&url=https%3A%2F%2Fwww.g2.com%2Farticles%2Fobject-detection&usg=AOvVaw03FKj3Z8-0AIHGEfzeaNg2
- Wilson, T. (2018, January 30). *The principles of good UX for Augmented Reality*. Medium. https://uxdesign.cc/th e-principles-of-good-user-experience-design-for-augmented-reality-d8e22777aabd

LAWS CONCERNED:-

1. *Civil Law Rules on Robotics European Parliament resolution of 16 February 2017 with recommendations to the Commission on Civil Law Rules on Robotics (2015/2103(INL))*. (n.d.). Retrieved April 14, 2023, from https://eur-lex.europa.eu/legal-content/EN/TXT/ PDF/?uri=CELEX:52017IP0051&rid=9

2. *Biometric Information Privacy Act (BIPA)*. (2021, April 26). ACLU of Illinois. https://www.aclu-il.org/en/campaigns/biometric-information-privacy- act-bipa

3. State of California Department of Justice. (2023, February 15). *California Consumer Privacy Act (CCPA)*. State of California - Department of Justice - Office of the Attorney General. https://oag.ca.gov/privacy/ccpa

4. *Overview: Flying drones and model aircraft | UK Civil Aviation Authority*. (n.d.). Register-Drones.caa.co.uk. https://register-drones.caa.co.uk

VERSION CONTROL GIT: -

GitHub repository for the source code: https://github.com/S-Prince7/Recon_Bird_Package.git

(*Please find the video demonstration of the project in the 'additional files' section*)