

DESIGN AND FABRICATION OF POLLINATION DRONE

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Mechanical Engineering.

By

ADVAITH R (38150005)
IMMANUVEL GODWIN D (38150039)



DEPARTMENT OF MECHANICAL ENGINEERING

SCHOOL OF MECHANICAL ENGINEERING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119.

MAY – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF MECHANICAL ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ADVAITH R (38150005)** and **IMMANUVEL GODWIN D (38150039)** who carried out the project entitled "Design and Fabrication of Pollination Drone" under our supervision from **August 21 to March 22**.

Internal Guide

Dr. J. JEYA JEEVAHAN, M.E., Ph.D.

Head of the Department

Dr. G. ARUNKUMAR, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **ADVAITH R (38150005)** and **IMMANUVEL GODWIN D (38150039)** hereby declare that the Project Report entitled “**DESIGN AND FABRICATION OF POLLINATION DRONE**” done by us under the guidance of **DR. J. JEYA JEEVAHAN**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in 2018-2022.

DATE: 30/04/22

PLACE: CHENNAI

1)

R. Advaith

2)

Immanuel Godwin D

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **DR. S. PRAKASH, M.E, Ph.D.**, Dean, School of Mechanical Engineering and **DR. G. ARUNKUMAR, M.E, Ph.D.**, Head of the Department, Dept. of Mechanical Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **DR. J. JEYA JEEVAHAN, M.E, Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Mechanical Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

One of the issues that is concerning recent agricultural production is pollination. However, the current flagging of honey bees has obtained vulnerable output. This lack in pollinators has noticeably amplified the fee of agriculturalists and hiring them for cross-pollination facilities. There is an essential need to hunt for substitute pollination systems for food sustainability and economical opinion. Archaic selection approaches used in insect nursing have several difficulties, as well as that they are labor-intensive and possibly unreliable. Some of these drawbacks could be decreased using “Air Vehicles”, Computer Vision, and Deep Learning-based tactics to automate pollination. The implementation of pollination drones is a very new concept in the agricultural field. There are very few papers proposed but none was implemented in real-time which was a success. To overcome this disadvantage, we create an automated drone for artificial pollination which uses the following, Deep-Learning and Machine-Learning algorithms to recognize the flower’s size, place, state and location to direct the drone to capture and interconnect with floras for the pollination process. The novelty of the proposed “Autonomous Pollination System” discussed in this paper comes from the integration of previously developed functionalities into one unified solution. Techniques from computer vision, controls, mechanical design, trajectory generation, and localization are all involved in developing a robust autonomous pollination system. In this project, we use a Drone and the Artificial Intelligence procedures to carry the pollen grain and then perform pollination to increase yield. Drone pollination evades numerous prevailing issues with regular pollinators in agriculture, namely honeybee colony collapse disorder, killers, viruses and pollinator fleas, opposing weather, pesticide spray, and the obtainability of pollinators in an appropriate manner. Second, machines like pollinators will improve creation and value. With the reducing number of bees, “Artificial Pollination” is more in trend. The Artificial pollination is suitable and it can also increase plant yield efficiency. The effective achievement of this development will encouragingly impact the field of “non-natural pollination” in agronomy.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF SYMBOLS AND ABBREVIATIONS	v
	LIST OF FIGURES	vi
1	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 TECHNOLOGIES USED	2
	1.2.1 Embedded System	2
	1.2.1.1 Characteristics of an Embedded System	3
	1.2.1.2 Basic Structure of an Embedded System	3
	1.2.2 Internet of things (IoT)	4
	1.2.2.1 History of IoT	5
	1.2.2.2 How IoT works	6
	1.2.2.3 Benefits of IoT	7
	1.2.2.4 Importance of IoT	8
2	LITERATURE REVIEW	11
	2.1 INTRODUCTION	11
	2.2 LITERATURE SURVEY	11
	2.2.1 Towards computer vision and deep learning facilitated pollination monitoring for agriculture	11

	2.2.2 Intelligent Autonomous Pollination for Future Farming	12
	2.2.3 Visual tracking based on flower pollination algorithm	12
	2.2.4 Task Scheduling in cloud computing using various techniques	13
	2.2.5 Attentive Layer Separation for Object Classification and Object Localization in Object Detection	13
	2.2.6 Multi-Drone 3D Trajectory Planning and Scheduling in Drone Assisted Radio Access Networks.	14
	2.2.7 Communications and Control for Wireless Drone-Based Antenna Array.	14
3	AIM AND SCOPE	16
	3.1 Objective of the project	16
	3.2 Scope of the project	16
4	METHODOLOGY	17
	4.1 Existing System	17
	4.2 Disadvantage of the existing system	17
	4.3 Proposed system	17
	4.4 Advantages of Proposed system	18
	4.5 Application	19
	4.6 System Architecture	19
	4.7 Working	19
	4.8 Software Description	20
	4.9 Algorithm and Software used	21
	4.9.1 EfficientNet Algorithm	21

	4.9.2 Google Colab	22
	4.9.3 Libraries	24
5	RESULTS AND DISCUSSION	25
	5.1 Introduction	25
	5.2 Results Obtained	25
6	CONCLUSION AND FUTURE WORK	32
	6.1 Conclusion	32
	6.2 Future Work	32
	REFERENCES	33
	APPENDIX	34

LIST OF SYMBOLS AND ABBREVIATIONS

ROTS	REAL TIME OPERATING SYSTEM
A-D	ANALOG-TO-DIGITAL
D-A	DIGITAL-TO-ANALOG
DARAN	DRONE ASSISTED RADIO ACCESS NETWORKS
BS	BASE STATION
ILP	INTEGER LINEAR PROGRAMMING
AC	ALTERNATE CURRENT
ESC	ELECTRONIC SPEED CONTROLLER
BLDC	BRUSHLESS DC MOTORS
CPU	CENTRAL PROCESSING UNIT
GPU	GRAPHICAL PROCESSING UNIT
TPU	TENSOR PROCESSING UNIT

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
1.1	STRUCTURE OF EMBEDDED SYSTEM	4
1.2	EXAMPLE OF IOT SYSTEM	7
4.1	SYSTEM ARCHITECTURE	19
4.2	EFFICIENTNET ALGORITHM	22
4.3	GOOGLE COLLAB NOTEBOOK	23
5.1	DRONE	25
5.2	INITIALIZING COMMAND	26
5.3	TAKE-OFF COMMAND	26
5.4	DRONE FLYING	27
5.5	STREAM ON COMMAND	27
5.6	DRONE VISUALS	28
5.7	PASSING FORWARD COMMAND	28
5.8	DRONE MOVING TOWARDS THE FLOWER	29
5.9	PASSING BACKWARD COMMAND	29
5.10	DRONE MOVES BACK	30
5.11	PASSING CLOCK WISE COMMAND	30
5.12	COMMANDS ARE REPEATED	31

CHAPTER 1

INTRODUCTION

1.1 GENERAL

“Pollination” is an aware term to almost everybody. We know that flowers must be pollinated, usually by different types of insects, for the plant to generate fruits and seeds. We have seen captions asserting that insect pollinator populaces are decreasing, menacing those important methods. It’s a well-known fact that pollen is the yellow dust that totally covers our cars at the time of spring, and we tend to blame it for the periodic aversions. For numerous people, that’s completely as far it gets. But taking some amount of time to look into it can gradually lead to astonishing discoveries. Individual plants and flowers have advanced their individual-focused constructions and pollination methods over billions of years. The Plant-Pollinator support has progressed to become the vital facilitator of pollinator nourishment and plant replica. Pollination is basically the artificial system used by growing crops. It’s the procedure of transferring pollen from the “male-anther” to the “female-stigma” of a flower that creates the seeds and permits the reproduction process. Pollen is a dust-like particles or a powder that grows inside stamen and collect on its external surface. Pollen grains are very distinctive in their shape and sculpturing their outside surfaces because they are very hard. For affirmative pollination, the pollen should be conveyed into a “stigma” of the identical plant family at the impeccable time. When the pollen gets on a stigma, one of the pollen grain’s two main cells ‘Germinate’ and creates a pollen tube. The following cell inside the pollen grain ruptures into two different sperm cells which transportable up pollen tube to ovule. A single Sperm cell link with egg, forming an embryo, and eventually it produces a seed. The additional links with a cell in the ovule to produce endosperm, which providing the embryo’s nutrition might have one or several ovules. Only a pollinated ovules advances to the seeds. If all ovules aren’t fertilized by distinct pollen grains, less seeds grow and the following fruit is probable to be shaped unusually. The understanding of the sowed “Fruits” and “Vegetables” we eat are in fact herbal ovaries may be somewhat disgusting. Cross-pollination is major. There are profits to difference in helping plants adapt to a varying environment. Flowers reject unintentional self-pollination in many ways, with

altering the timing of pollen grain issue and stigma receptivity, the longitudinal planning of anthers and stigmas and detachment of male and female flower on similar plants. Amazingly, plants that typically had self-pollination fences in the dwelling can alternative their assembly and its chemistry to acknowledge their specific pollen if cross-pollination does not happen in a well-timed way due to weather or absence of pollinator movement.

1.2 TECHNOLOGIES USED

1.2.1 *Embedded System*

An “Embedded system” could be said to be software embedded in computer hardware. An “Embedded system” could be an autonomous system a portion of a huge system. An “Embedded system” is a microprocessor-based system which is planned to execute a particular task. E.g. a smoke detector is an “Embedded system”; it would detect only smoke.

An embedded system has 3 modules.

- Hardware.
- Application software.
- “Real-Time Operating system” (RTOS) that administers the software and delivers the procedures to make the processor execute a job as per schedule and latencies are controlled following a plan. RTOS states the methodology of the system workings. It gives the instructions through the execution of the program. A small-scale embedded system might not have “RTOS”.

So, we can state an “Embedded system” as a Micro-controller based, software-driven, real-time control system.

1.2.1.1 Characteristics of an Embedded System

- *Single-functioned* – An embedded system frequently performs a focused process and repeats the same process. For instance: A pager constantly operates as a pager.
- *Tightly constrained* – All computation methods have limitations on enterprise metrics, but which on an “embedded system” could be particularly fitted. Project metrics are a measure of an application's features such as its price, scope, control, and presentation. It must be of a size to fit on a solitary chip, must achieve fast enough to procedure data in actual time, and consume the least power to spread battery life.
- *Reactive and Real-time* – Several embedded systems essential frequently respond to variations in the structure's environment and must execute definite results in real-time without any delay. Consider a “car cruise controller”; it frequently watches and responds to speed besides brake sensors. It essentially calculates ‘acceleration’ or ‘de-acceleration’ frequently within an incomplete time; a behind calculation can outcome loss of control of the car.
- *Microprocessors based* – It must essentially be “microprocessor or microcontroller” based.
- *Memory* – It must necessarily have a ‘memory’, its software typically embeds in ‘ROM’. It does not require any subordinate memories in the computer.
- *Connected* – It must basically a have connected computer peripheral to connect both I/P and O/P devices.
- *HW-SW systems* – Software is used for additional features and flexibility. Hardware is used for computational performance and safety needs.

1.2.1.2 Basic Structure of an Embedded System

The following diagram displays the basic construction of an embedded system.

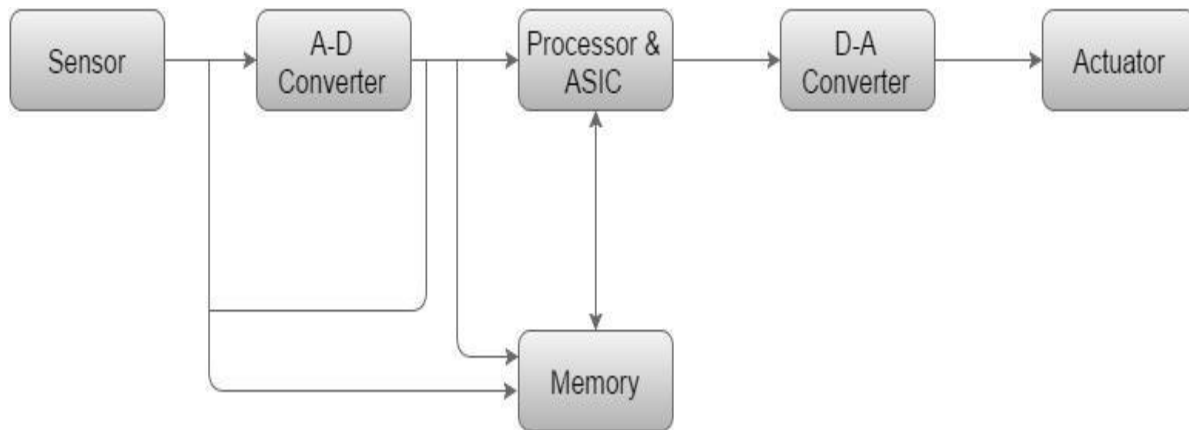


Fig: 1.1: Structure of the embedded system

- *Sensor* – It measures the corporeal amount and it converts to an electric signal which can be delivered by a spectator or by any electrical gadget like an “A2D” converter. A sensor supplies the resultant quantity in the memory.
- *A-D Converter* – An analog-to-digital converter converts the analog signal directed by the sensor into a cardinal signal.
- *Processor & ASICs* – Processors process the data to measure the results and store it in the memory.
- *D-A Converter* – A digital-to-analog converter converts the digital data served by the computer to analog data.
- *Actuator* – An actuator relates the yield given by the “D-A Converter” to the actual output stored in it and stores the agreed output.

1.2.2 Internet of things (IoT)

The “Internet of things” or “IoT”, is a scheme of consistent computer computing devices, power-driven, and digital machines, matters, animals or humans that are provided with single identifiers (UIDs) and the skill to transmit data over a system deprived of needful “human-to-human” or “human-to-computer” interaction.

An object in the “IoT” can be a human with a heart screen implant, a predator with a “biochip transponder”, a vehicle that has integrated sensors to keep the driver aware

when tire burden is a little malfunction or any other normal or artificial object that can be allocated an IP address and is talented to transmit data over a net.

Progressively, establishments in a variety of industries are making use of “IoT” to operate more proficiently, making us understand better and benefit the customers to deliver enhanced client service, advance administration, and surge the worth of the business.

1.2.2.1 History of IoT

“Kevin Ashton”, the co-founder of the “Auto-ID Center” at MIT, first declared the “Internet of things” in an exhibition he made to ‘Procter & Gamble (P&G)’ in 1999. Wanting to transport radio frequency ID (RFID) to the attention of P&G's senior management, Ashton entitled his presentation "Internet of Things" to integrate the cool fresh trend of 1999: “The Internet”. MIT professor “Neil Gerstenfeld’s” book, When Possessions Start to think, also appearing in 1999, didn't use the particular term but only if a pure vision of where the Internet of Things was led.

“IoT” has progressed from the merging of wireless techs, Microelectromechanical Systems (MEMS), micro services, and the internet. The merging has assisted slit down the storage tower amongst “Operational Technology” (OT) and “Information Technology” (IT), allowing unstructured machine-generated data to be analyzed for perceptions to drive enhancements. Though Ashton's was the initial mention of the “Internet of things”, the idea of linked devices has been around since the 1970s, beneath the monikers embedded cyberspace and prevalent computing.

The first net machine, for instance, was a “Coke machine” at ‘Carnegie Mellon University’ in the early 1980s. Using the net, a computer programmer can check the position of the device and decide whether there is a soft drink pending for them, should they go to the engine.

“IoT” grew from ‘machine-to-machine’ (M2M) communication that is machined linking to individually other via a system without humanoid interaction. M2M states to linking a trick to the cloud, handling it, and gathering data.

Taking M2M to the succeeding stage, “IoT” is a sensor grid of n number smart devices that link persons, schemes, and other uses to gather and share data. As its basis, M2M offers the connectivity that enables “IoT”.

The “IoT” is also a regular extension of “SCADA” (‘supervisory control and data acquisition’), a group of package application lists for progression control, data gathering in real-time from isolated positions to governor equipment and situations. “SCADA” systems include computer hardware and software program modules. The hardware collects and uploads data into a computer that has “SCADA” software installed in it, where it is then managed and presented it in a well-timed manner. The advancement of “SCADA” is such that final-generation “SCADA” systems developed into first-generation “IoT” systems

The idea of the “IoT” ecosystem didn't really arise on its own until the mid of 2010 when, in part, the administration of China told it will make “IoT” strategic importance in its five-year plan

1.2.2.2 How IoT works

An IoT ecology contains web-enabled clever plans that use an embedded processor, sensors, and also communication hardware to collect, direct and take action on the info they attain from their environs. IoT plans segment the device information they gather by linking an IoT gateway otherwise an advantage device where information is sent to the cloud to be examined or examined nearby. Sometimes, these devices communicate with other related devices and act on the information they get from one additional. The devices do most to function without human interference, though individuals can have interaction with the gadgets, for example, to set them up, give those orders or access the data. The connectivity, interacting and communication procedures used with these

'web-enabled' devices mainly depends on the exact IoT requests deployed.

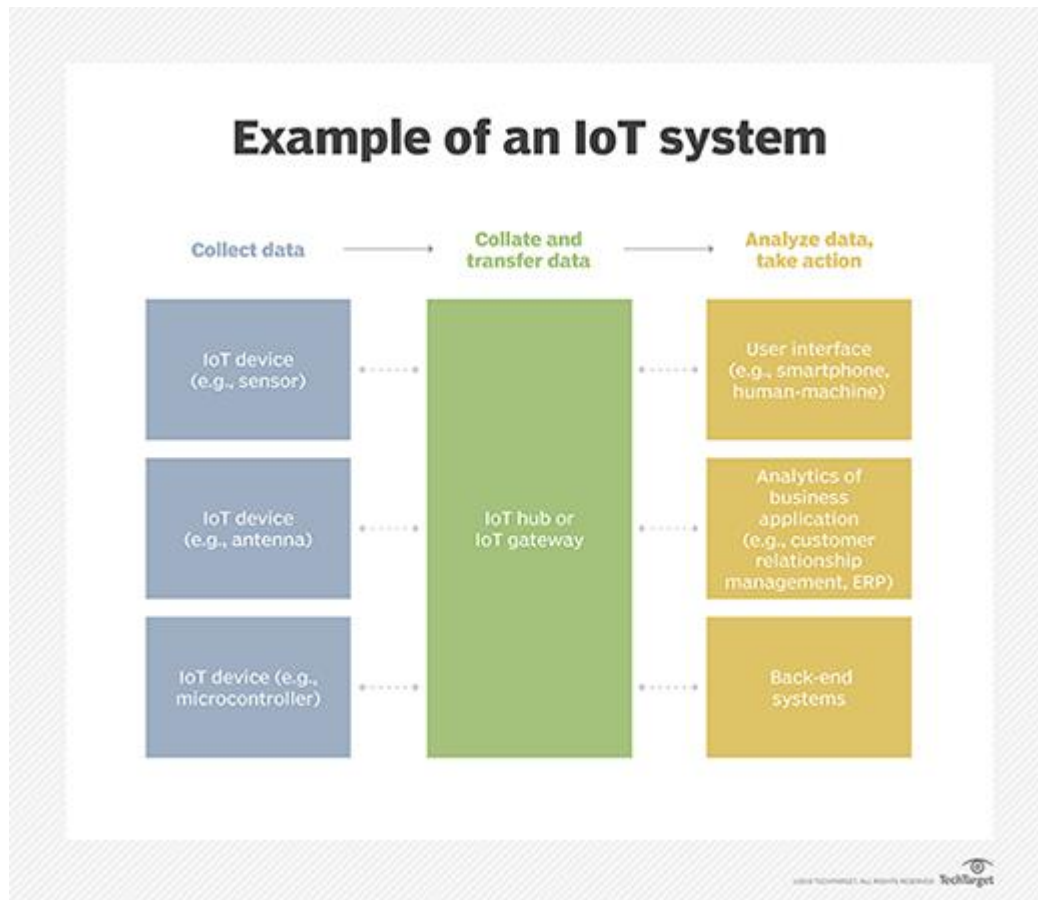


Fig: 1.2: Example of IOT system

1.2.2.3 Benefits of IoT

The IoT (internet of things) offers several benefits to that particular organization, enabling them to:

- monitor their complete business processes;
- advance the customer experience;
- save time and cash;
- improve employee productivity;
- integrate and acclimatize business models;
- make better business choices; and
- produce more revenue.

IoT stimulates companies to reconsider the ways they method their businesses,

industries & markets and gives them tools to advance their occupational strategies.

1.2.2.4 Importance Of IoT

When something is linked to the net, which technically means is that it can either send information or obtain information, or can perform both. This has the skill to direct and/or collect information makes things work smart, and here smart means good.

Let's use phones (smartphones) as an example. Right now, we can attend to any song in the world, but it's not because our phone essentially has every song in the world already stored in it. The reason is that every song that is in the world is stored somewhere else, but our phone can send information and then receive information

To be clever, a thing doesn't require to have great stowage or a super processor inside of it. All a thing has to do is attach to super storage or to a supercomputer. Being linked is awesome.

In the IoT, all the things that are actually connected to the net can be put into 3 categories:

1. Things that gather information and then send it.
2. Things that gather information and then turn according to it.
3. Things that can perform both.

And all 3 of these have enormous profits that feed on each other.

When to some degree it is linked to the net which means that it can direct information or obtain information, or both. This capability to send and/or receive information makes things smart.

Collecting and Sending Information: This refers to sensors. Sensors could be a temperature sensor, a motion sensor, a moisture sensor, an air quality sensor, a light sensor, or you name it. These sensors, sideways with a connection, allow us to mechanically gather information from the atmosphere which, in turn, lets us make more smart decisions. “Soil moisture sensor” On the farm, mechanically getting info about the soil wetness can educate the farmers exactly when their crops required be sprayed. In its place of tearing too much or watering too little, the farmer can guarantee that crops get precisely the correct amount of water. More currency for agriculturalists and more food for the world. Just as our vision, range, odor, touch, and sensitivity allow us, people, to make sense of the world, sensors allow machines to automatically make sense of the world.

Receiving and Acting on Information: We are all very aware of machinery getting data and then substituting according to it. Your printer obtains a document and prints it. Your car receives a signal from your car key and then it opens the door. The samples are endless. If it’s as simple as transferring the knowledge “turn on” or as compound as sending a 3-Dimensional model to a 3-Dimensional printer, we know that we can tell machines what to perform from far away. The real strength of the Internet of Things ascends when things can perform. Things that collect information and send it, but also obtain information and act on it.

Doing Both: Let’s rapidly go back to the farming instance. The sensors can gather information about the ‘soil moisture’ to express to the farmer how much to water the yields, but you do not nearly need the farmer to be present. In its place, the ‘irrigation system’ can mechanically turn on as required, according to how abundant wetness is in the soil. You can advance a step further. If the irrigation system gets data about the weather from its net assembly, it can also alert when it’s raining and choose not to water the crops today because they will be watered by rain. It does not stop there. All this data about the earth’s moisture, how much the irrigation system is watering the crops, and how well the crops really grow can be composed and directed to “supercomputers” that run astonishing procedures that can make sense of all this data. That is fair one type of

sensor. Enhance in additional sensors like bright, midair quality, and room temperature, and these procedures can study much more. With dozens, hundreds, thousands of farms all collecting this information, these procedures can generate incredible visions into how to make crops produce the best, serving to feed the world's increasing population.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

The following shows, a survey done for pollination and drone system. The most popular of the existing techniques is been discussed as follows.

2.2 LITERATURE SURVEY

2.2.1 “Towards Computer Vision and Deep Learning Facilitated Pollination Monitoring for Agriculture”

Spotting is beneficial to recognize insects' contribution to crop pollination. Conventional sampling approaches used in insect observing have quite a few shortcomings, that those methods are labor-intensive and possibly unpredictable. Some of the downsides might be nullified using computer vision and deep learning-based approaches to automate pollination observing. CV-based pollination observing and put forward a innovative algorithm, Polytrack, which tracks several insects at a similar time in complex agricultural atmospheres. Polytrack contains flowers recognizing section to automate the method of gathering insect-flower interface data, and a small-resolution handling mode that shrinks computational demands engaged in the processor to adapt the software towards the necessities of low-powered observing hardware.

.

Pros: Deep learning and computer vision technologies are used to recognize and track insects accurately.

Cons: Only focused on insect movements and not on pollination.

2.2.2 “Intelligent Autonomous Pollination for Future Farming”

Food safety is one of the societal challenging topics. As 1/3rd of all food consumed by people depend on 'animal pollination'. Currently, this research delivers a developing solution to food supply decrease produced by population lessening of normal pollinators,

so as to reduce its impact on environmental relations, bionetwork conservation, and constancy, the genetic difference in the crop plant public, flowery diversity, specialization, and development. This paper advances a theoretical-practical roadmap of independent fertilization for upcoming farming by means of machine like air vehicle pollinators. The research deals fresh ideas into self-governing design and production and into probable means to raise the production efficiency which restrains the period from lab to market. The independent MPs are being recognized by Artificial Intelligence and humanoid capability in the loop for the keen agricultural industry. Further, this work classifies scientific and technical advances that are predictable to translate, within future regulatory outlines, into the universal use of MPs for agricultural claims and beyond that.

Pros: Agricultural actions are completely automatic by using robots due to usage of human power is decreased.

Cons: The broad mechanization process needs a lot of research for real-time application.

2.2.3 “Visual tracking based on flower pollination algorithm”

“Flower pollination algorithm” (FPA) is a new optimization procedure that imitates the real-life methods of “flower pollination”. In this study, computer visual tracking is stated as a technique of idyllic reproduction of plants. An FPA-based tracing architecture is projected and the tracing accuracy and velocity are compared with particle filter and mean shift.

Pros: A new algorithm (Flower pollination algorithm) is proposed for analyzing real-life processes of flower pollination.

Cons: Accuracy can be improved.

2.2.4 “Task Scheduling in Cloud Computing using Various Techniques”

“Cloud Computing” is in mandate nowadays as it offers consistent, scalable, and economical IT operations. As the measure of “cloud computing” is cumulative the essential of a well-organized scheduling algorithm for implementation management of resources is also becoming vital. Scheduling unruly in the cloud is an “NP-hard” problem. “Heuristic” methods have been beneficial in as long as near-ideal solutions to the scheduling problem. In this paper, a new method for “Task Scheduling with Flower Pollination Algorithm” (TSFPA) has been announced to assign properties to the job. The purpose of this projected program algorithm is to diminish the make span. The presentation of the future system has been related to the inherited algorithm (GA), “first come first serve” (FCFS), and “round-robin” (RR) approach of scheduling using Cloudsim toolkit. Simulation results showed that the performance of TSFPA is well than GA, RR, and FCFS in relation to make span.

Pros: The routine of the “TSFPA” Algorithm is better than GA, RR, and First Come First Serve (FCFS) in terms of make span.

Cons: The competence of this system is very low.

2.2.5 “Attentive Layer Separation for Object Classification and Object Localization in Object Detection”

In object detection, object organization and object localization errands are directed. Deep learning related object detection links perform with feature charts generated by totally public networks. Though, object organization highlights the utmost object portion of the feature plan. Whereas, object confining needs a feature plan that is devoted to the whole region of the object. Complete trial comments based on the “PASCAL VOC dataset” and “MS COCO dataset” exhibited that this object detection system outperformed the “state-of-the-art” methods.

Pros: “PASCAL VOC Dataset” and “MS COCO Dataset” exposed that offered object detection network outpaced the “state-of-the-art” methods.

Cons: Implementing this concept involves more time period and money.

2.2.6 “Multi-Drone 3D Trajectory Planning and Scheduling in Drone Assisted Radio Access Networks”

3D path of numerous DBSs in the “drone-assisted radio access networks” (DARAN) where DBSs flyover associate dare as of interests and convey communications between the “Base Station” (BS) and users in Aols. Articulate the multi-DBS 3Dimensional path scheduling and planning as a mixed-integer non-linear programming (MINLP) unruly with the aim of reducing the mean DBS to user (D2U) trail loss. The 3Dimensional path differences in straight and perpendicular instructions, and the DBS-related net representations, are premeditated in the formulation. To account ” NP-hardness” of the “MINLP” and the non-protrusion problem, decouple it into numerous Integer Linear Programming (ILP) and quasi-convex sub-problems in which Aol association, D2U message planning, horizontal trails, and flying altitude of DBSs are similarly enhanced. Then, the Multi-DBS 3Dimensional path development and scheduling process to resolve the problems placed on the BCD technique. A k-means based early trajectory development and a search start slot planning is well thought-out in the projected procedure to advance arc design performance and confirm inter-DBS distance constraint, respectively.

Pros: Path strategy performance and guarantee inter-DBS distance limitation is enhanced using the proposed algorithm.

Cons: Real-time implementation is very difficult.

2.2.7 “Communications and Control for Wireless Drone-Based Antenna Array”

The real use of “manifold quadrotor drones” as an airborne aerial array that offers wireless facility to users on the ground is examined. In particular, underneath the goal of

minimalizing the airborne facility time needed for interacting with ground users, an original framework for organizing and working a drone projection array system whose fundamentals are single-aerial drones is future, the provision time is diminished by minimalizing the wireless communication time and the governor time that is wanted for the stability of the drones. To abate the communication time, the aerial array improvement is exploited by enhancing the drone space inside the array. In case, by means of agitation practices, the “drone space optimization” problem has been answered by solving succeeding, “perturbed convex optimization” problem. Then, rendering to the position of respectively field user, the ideal locations of the drones around the array’s middle are resulting in such that the communication time for the user is minimalized.

Given the strongminded ideal positions of drones, the drones apply a governor period to alter the locations animatedly so as to help many users. To diminish this fly time of the “quadrotor drones”, the speed of rotors is ideally familiar based on both the terminuses of the drones and peripheral forces.

Pros: The rapidity of rotors is ideally aware created on both the destinations of the drones and exterior forces.

Cons: The drone can solitary fly up to a firm limit.

CHAPTER 3

AIM AND SCOPE

3.1 OBJECTIVES OF THE PROJECT

- We aim to fabricate a novel pollination drone that can identify the flowers and transfer the pollen grains to different flowers.
- To improve and promote artificial pollination.
- To increase productivity and yield in agriculture.

3.2 SCOPE OF THE PROJECT

- It will be used for the artificial pollination process.
- It could be used to identify diseases in plants.
- It could be used for delivery robots.

CHAPTER 4

METHODOLOGY

4.1 EXISTING SYSTEM

Worldwide, pollinators change 35% of farm property and play a crucial part in crop creation. Therefore, observing is convenient to identify insects' contribution in crop pollination. Traditional sampler methods used in insect monitoring has some shortcomings, counting that they are need more labor and possibly unpredictable. Some of these shortcomings might be shrank using computer vision, air vehicles and deep learning-based practices to mechanize pollination. A pipeline for computer vision-based pollination observing and put forward an innovative algorithm, Polytrack, that trails numerous insects concurrently in difficult farming environments. This procedure uses foreground/ background segmentation and deep learning to spot and track insects has achieved accuracy and recollection at a very good rate. Correspondingly when we are observing honeybees foraging in investigation sites in the polytunnels of a strawberry farm. Polytrack comprises a flower recognition module to computerize the gathering of insect-flower boundary data, and a low-resolution allowance mode that drops computational configuration claim placed on the processor to grind the software towards the necessities of low-powered observing hardware.

4.2 DISADVANTAGES OF EXISTING SYSTEM

- It individually detects the program of insect pollinators in farming lands.
- It is not malleable for other submissions like fake pollination.
- Procedure is attentive to produce the location of insect pollinators.

4.3 PROPOSED SYSTEM

One of the main subjects about current farming production is crop pollination. Roughly \$74 billion per year value of crops in a bank on pollination by different pollinators. Though, the new fading of honey bees has considerably threatened output. Such a deficiency of pollinators has significantly enlarged the cost of agriculturalists and rental

them for pollination facilities. As of together financial and nourishment sustainability opinions of view, there is a crucial essential to seek substitute pollination systems. Globally, pollinators touch 35% of farming land and perform the main role in food creation. Therefore, frequent monitoring is helpful to appreciate the impact insects make on crop pollination. Traditional sampling methods in insect observing have various difficulties, as well as they are industry concentrated and theoretically undependable. Some of the drawbacks may be solved by means of computer vision, air vehicles and deep learning-based techniques to mechanize pollination. To overcome this problem this project offers a computerized drone for pollination which makes use of deep learning and machine learning algorithms to approximate the flower location, dimensions, alignment, and corporeal condition to monitor the drone to seizure and cooperate with floras for pollination. In this concept, we have used artificial intelligence and drone procedures to carry pre-collected pollen and transfer them to flower for pollination to rise the yield. Drone pollination dodges several current problems with usual pollinators in agriculture, such as “honeybee colony collapse disorder”, pollinator fleas, killers, pesticide spray, contrasting weather, and the accessibility of pollinators in a suitable manner. Moreover, robotic pollinators will improve production and fruit value. With the diminishing number of bees, false pollination is extra in program. If we take the example of “China”, 100% of floras are pollinated in non-natural way. So, we could see that non-natural pollination is favorable and can upsurge plant yield. The optimistic achievement of this plan would impact the field of fake pollination in farming.

4.4 ADVANTAGES OF THE PROPOSED SYSTEM

- Improves and promote artificial pollination
- Increase productivity and yield in agriculture
- An ESC (Electronic Speed Controller) is used which is used to control the drone in the desired direction.

4.5 APPLICATIONS

- Applicable in all areas where the drone is necessary
- Artificial pollination
- Agricultural areas where monitoring is necessary
- Seeding the farmer land
- Taking a clear picture of natural calamity affected area
- Army and police department

4.6 SYSTEM ARCHITECTURE



Fig: 4.1: System Architecture

4.7 WORKING

The drone contains of four “BLDC” motors fixed on all flying arms. The transmitter provides the knowledge to any receiver board present on the drone. As the drone accepts the command its performances consequently to hover in a specific direction. After the “take-off” command is established, the drone that transfers the pre-collected pollens commences its

“take-off” from the ground and hovers to a definite elevation, and pauses for the “stream on” command. When the stream on command is delivered, a camera that is attached on the drone turns on and starts streaming the visuals captured by the drone to the monitor. These visuals could be used to detect the movement of drone and through which we can analyze the drone’s performance and update if needed. Then the drone starts examining for the florae nearby for pollination. When any flower is recognized, the forward command is spontaneously generated to change the drone near to the flower to initiate pollination process. After pollination is completed, backward command is spontaneously generated to transfer the drone back to its original examining spot. And the right-handed command is spontaneously produced to rotate the drone in the clockwise direction so that it can detect next flower and repeat its process. Through which non-natural pollination is realized successfully. The drone flies giving to the speed given to the motors fixed on each arm of it. When the rapidity given to all the motors are increased from lower pwm voltage to higher pwm voltage, then the drone moves upwards, whereas when the pwm voltage is decreased from higher voltage to lower voltage, the drone comes down to the ground. In order to move left and right, the left motors should be slowed down whereas the right motors should be increased in speed and the same goes for the right side, such that motors in the right should be slowed down and the left motors should be given more speed. So, we could see that non-natural pollination is advantageous and could rise yield.

4.8 SOFTWARE DESCRIPTION

The purpose of the Software Requirement Specification is to produce the specification of the analysis task and also to establish complete information about the requirement, behavior, and also other constraint like functional performance and so on. The main aim of the Software Requirement Specification is to completely specify the technical requirements for the software product in a concise and unambiguous manner.

4.9 ALGORITHM AND SOFTWARE USED

4.9.1 EfficientNet Algorithm

EfficientNet is a convolutional neural network construction and the ascending way that consistently balances all sizes of depth, breadth, and resolution using a complex constant. Unlike conventional repetition that random scales these issues, the EfficientNet grading method consistently scales net depth, breadth and resolution with a set of static scaling constants.

The efficiency of prototypical scaling also depends on the starting point of the network. So, to additional improvement of its performance, they have also advanced a new baseline network by executing a neuron construction search using the “AutoML MNAS” basis, which enhances both correctness and effectiveness. The resultant planning uses portable overturned holdup convolution, parallel to ‘MobileNetV2’ & ‘MnasNet’, but is to some extent larger due to an enlarged ‘FLOP’ economical. They, then scale up the baseline network to obtain a family of models, called “EfficientNets”. They have compared their “EfficientNets” with other current CNNs on “ImageNet”. In general, the EfficientNet representations attain both sophisticated correctness and improved competence done present CNNs, dropping limit size and FLOPS by instruction of scale. However, EfficientNets complete fine on “ImageNet”, to be utmost valuable, they should also transmission to additional datasets.

The complex scaling technique can be comprehensive to current CNN constructions such as “Mobile Net” and “ResNet”. Though choosing a decent baseline network is serious for attaining the best outcomes meanwhile the compound grading technique only increases the prognostic dimensions of the networks by duplicating improper network’s underlying

convolutional processes and system structure.

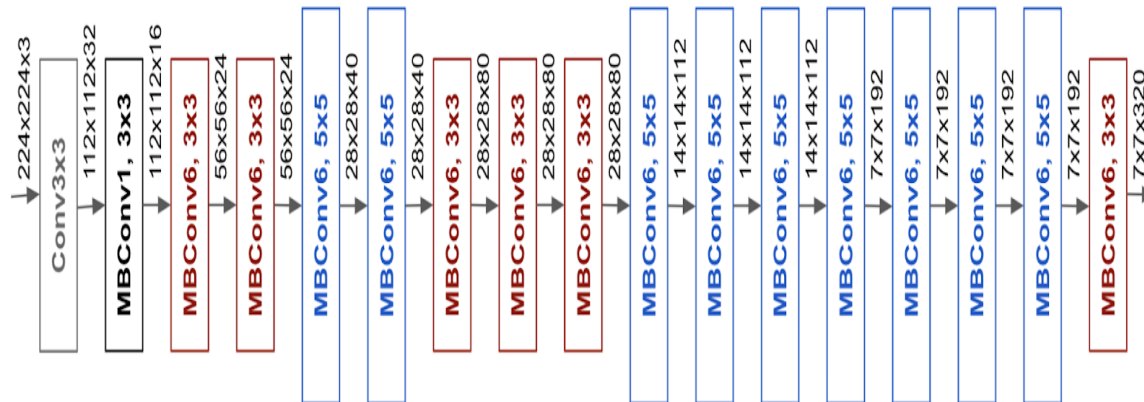


Fig: 4.2: EfficientNet Algorithm

4.9.2 Google Colab

In this project, google colab is used as an open-source IDE. Google Colaboratory is an unrestricted online cloud-based “Jupyter notebook environment” that permits us to train our machine learning, deep learning models on all CPUs, GPUs, and TPUs.

It gives us a GPU, which we can uninterruptedly run for twelve hours. For the majority of data science users, this is adequate to satisfy their computation requirements.

Google Colab provides three kinds of runtime for notebooks:

- CPU
- GPUs
- TPUs

Colab gives us twelve hours of the nonstop execution period. Later, the entire virtual engine is emptied and then we have to start over. We could run several CPU, GPU, and TPU requests at the same time, but their resources are shared amid these instances.

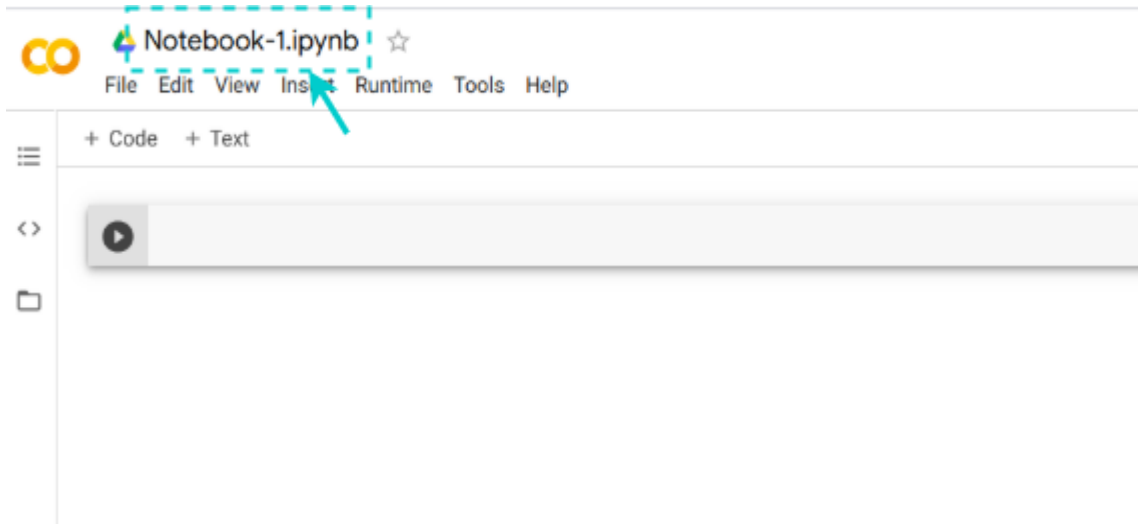


Fig: 4.3: Google Colab Notebook

Colab notebooks allow us to link feasible code and rich text in a solitary document, alongside images, HTML, LaTeX, and many. When we create our Colab notebooks, they are stored in our Google Drive account. We can easily share our Colab notebooks with colleagues or classmates, enabling them to access our notebooks or even edit them. Colab notebooks are Jupyter notebooks that are provided by Colab.

4.9.3 Libraries

- *Torch*- It delivers a supply the N-dimensional array or the Tensor, which chains elementary procedures for indexing, sharing, moving, type-casting, sharing storage, and duplicating.
- *Torchvision*- Torchvision is a collection for Computer-Vision that drives with “PyTorch”. It has values for well-organized Image and Video alterations, some normally used pre-trained models, and few datasets.
- *Efficientnet_pytorch*- EfficientNets are a group of image arrangement models, which attain state-of-the-art correctness, existence an order-of-magnitude lesser

and faster. It consists of the unique TensorFlow application, such that it is informal to load weights from a “TensorFlow” frontier.

- *TensorBoard*- TensorBoard is an instrument for providing the capacities and visualizations required through the “ML” workflow. It enables following test metrics like damage and correctness, imagining the model graph, jutting embedding to an inferior dimensional interplanetary.
- *Matplotlib*- Library of python used for visualizing the data using graphs, scatterplots, and so on. Here, we have used it for data visualization.
- *Numpy*- Library of python used for arrays computation. It has so many functions. We have used this module to change the two-dimensional array to a contiguous flattened array by using the ravel function.
- *tqdm*- tqdm is a Python library that permits you to produce a keen development bar by wrapping about any iterable. A tqdm development bar shows you how much time has passed and also shows the projected time lingering for the iterable.
- *Imutils*- A sequence of accessibility functions to create image processing utilities such as translation, rotation, resizing, and displaying Matplotlib images at ease with OpenCV.
- *Opencv python*- A sequence of suitability functions to make undeveloped image processing purposes such as conversion, revolution, formatting, framing, and showing Matplotlib images is simple with “OpenCV”.
- *DJITelloPy*- This library has different features like the execution of all tello commands, easily retrieving a video stream, receiving and parsing state packets.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 INTRODUCTION

This section discusses the practical results obtained while implementing the project.

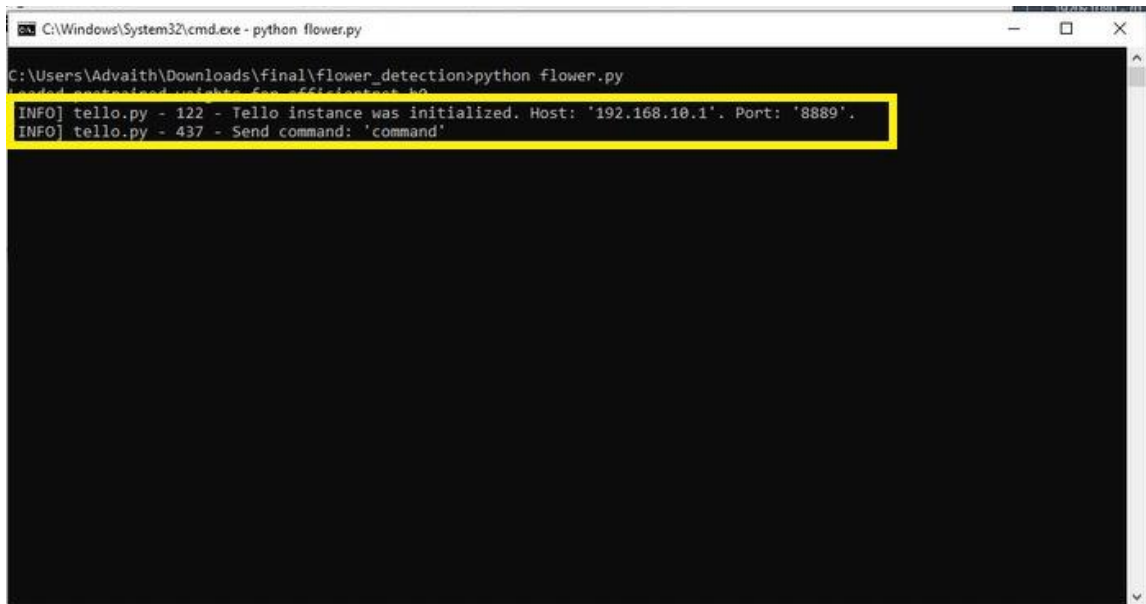
5.2 RESULTS OBTAINED

The below figure shows, the pollination drone.



Fig: 5.1: Drone

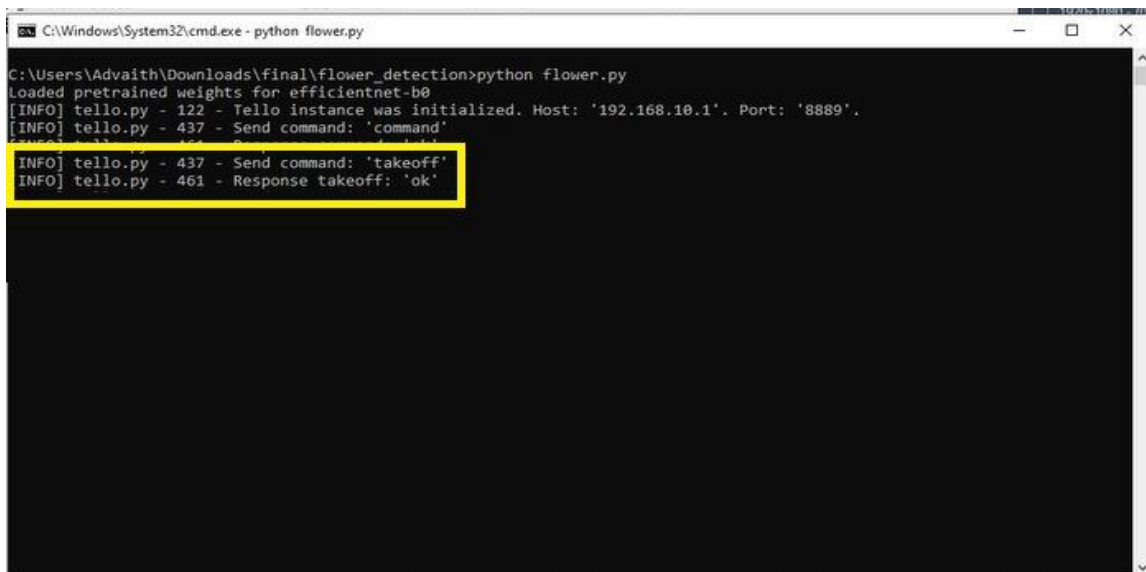
Initializing the drone to receive commands further

A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe - python flower.py". The prompt shows the command "python flower.py" being executed. The output includes two lines of log messages: "[INFO] tello.py - 122 - Tello instance was initialized. Host: '192.168.10.1'. Port: '8889'." and "[INFO] tello.py - 437 - Send command: 'command'". These two lines are highlighted with a yellow rectangular box.

```
C:\Windows\System32\cmd.exe - python flower.py
C:\Users\Advaith\Downloads\final\flower_detection>python flower.py
[INFO] tello.py - 122 - Tello instance was initialized. Host: '192.168.10.1'. Port: '8889'.
[INFO] tello.py - 437 - Send command: 'command'
```

Fig. 5.2: Initializing command

The initial take-off command is passed via the terminal to begin it takes off from the base and to reach a certain altitude.

A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe - python flower.py". The prompt shows the command "python flower.py" being executed. The output includes several lines of log messages. The last two lines are highlighted with a yellow rectangular box: "[INFO] tello.py - 437 - Send command: 'takeoff'" and "[INFO] tello.py - 461 - Response takeoff: 'ok'".

```
C:\Windows\System32\cmd.exe - python flower.py
C:\Users\Advaith\Downloads\final\flower_detection>python flower.py
Loaded pretrained weights for efficientnet-b0
[INFO] tello.py - 122 - Tello instance was initialized. Host: '192.168.10.1'. Port: '8889'.
[INFO] tello.py - 437 - Send command: 'command'
[INFO] tello.py - 437 - Send command: 'takeoff'
[INFO] tello.py - 461 - Response takeoff: 'ok'
```

Fig. 5.3: Takeoff command

Once the “take-off” command is received, the drone which transports the pre-collected pollens begins its take-off from the ground and flies to a certain altitude, and waits for the “streamon” command.



Fig: 5.4: Drone flying

“Streamon” command is passed to start the camera streaming.

```
C:\Windows\System32\cmd.exe - python flower.py

C:\Users\Advaith\Downloads\final\flower_detection>python flower.py
Loaded pretrained weights for efficientnet-b0
[INFO] tello.py - 122 - Tello instance was initialized. Host: '192.168.10.1'. Port: '8889'.
[INFO] tello.py - 437 - Send command: 'command'
[INFO] tello.py - 461 - Response command: 'ok'
[INFO] tello.py - 437 - Send command: 'takeoff'
[INFO] tello.py - 461 - Response takeoff: 'ok'
[INFO] tello.py - 437 - Send command: 'streamon'
[INFO] tello.py - 461 - Response streamon: 'ok'
```

Fig: 5.5: Streamon command

When the “streamon” command is passed, a camera that is fixed to the drone turns on and starts streaming the illustrations captured by the drone to the screen.

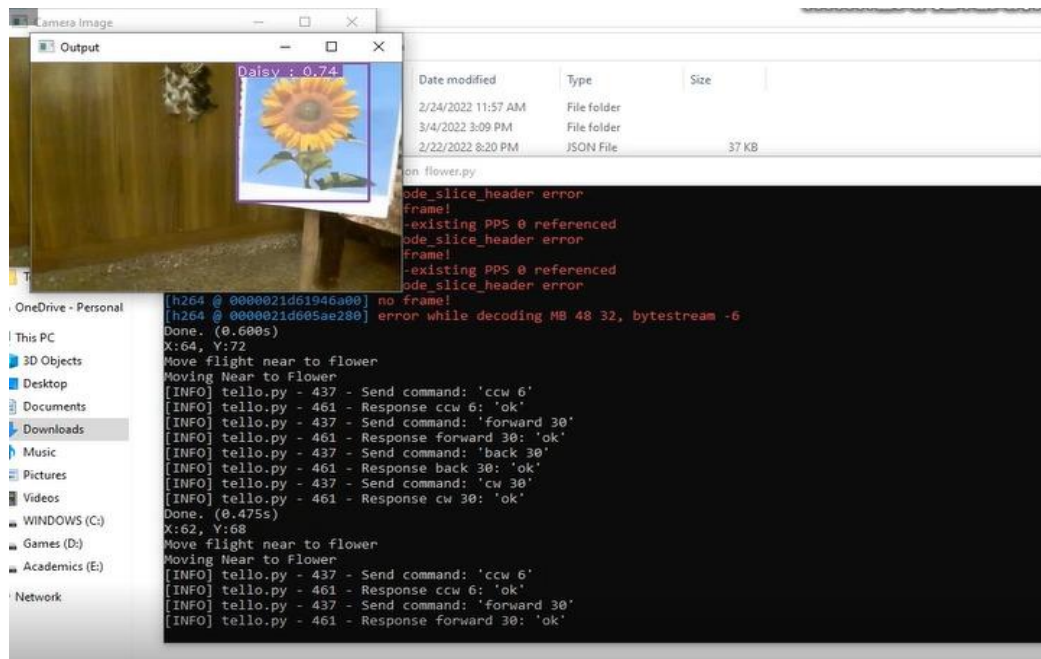


Fig: 5.6: Drone visuals

Then the drone begins to search for the flower close in radius for pollination. When any flower is sensed by the drone “forward” command is automatically produced to move the drone towards the flower to initiate pollination.

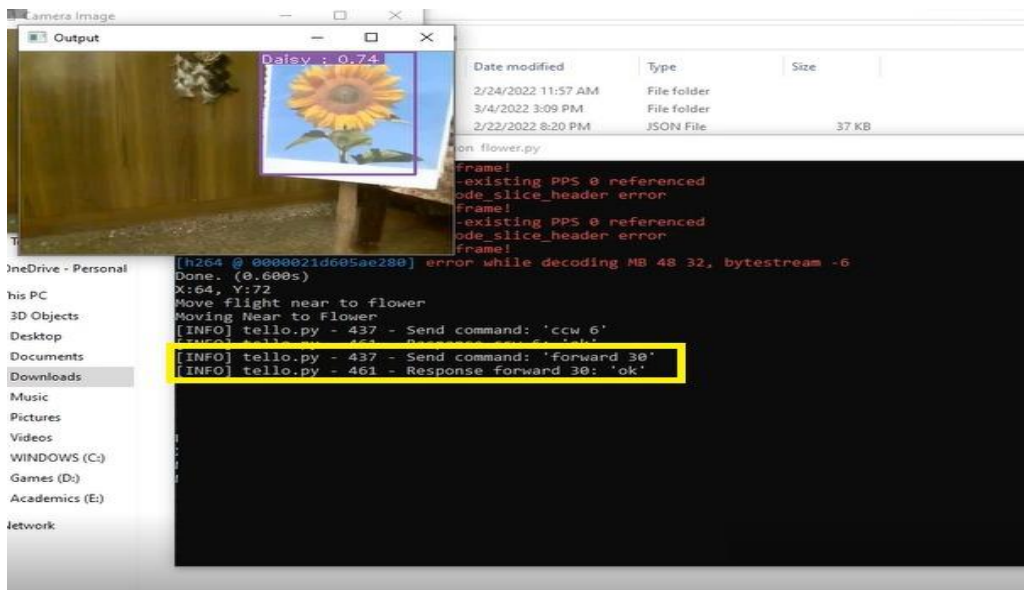


Fig: 5.7: Passing forward command

As the forward command is received the drone moves towards the flower for pollination.



Fig: 5.8: Drone moving toward the flower

Once the pollination is completed backward command is automatically produced to transfer the drone back to its initial searching position.

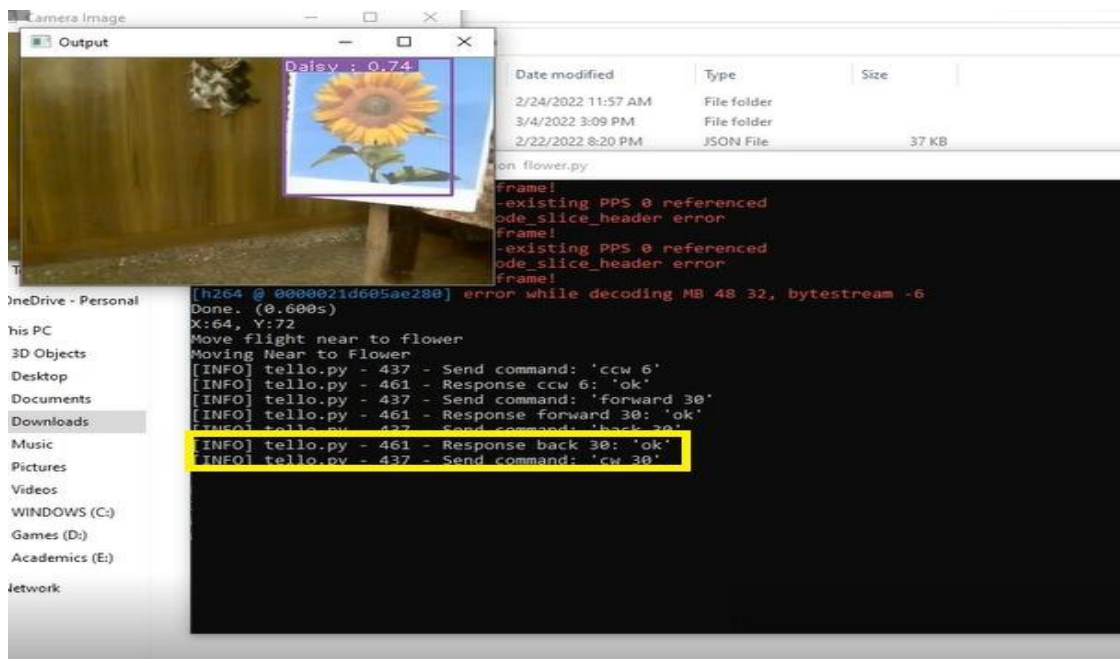


Fig: 5.9: Passing backward command

As the backward command is received the drone moves back to its initial searching position.



Fig: 5.10: Drone moves back

And the clockwise command is automatically generated to rotate the drone in the clockwise direction so that it can detect another flower and repeat its process.

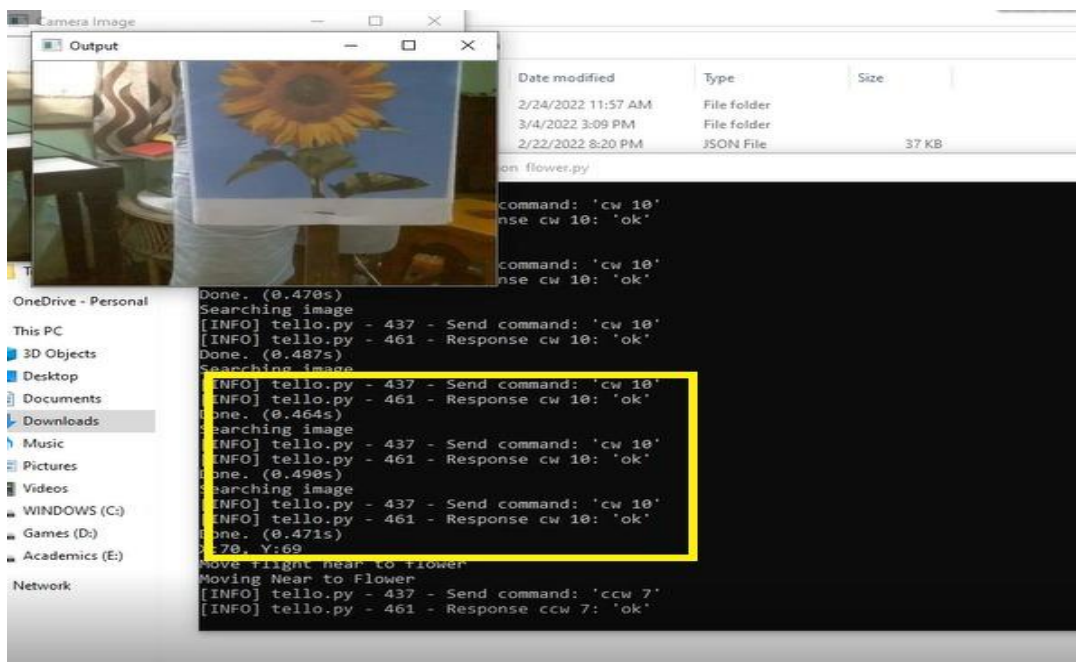


Fig: 5.11: Passing clockwise command

When the flower is detected forward and backward commands are generated again for initiating pollination.

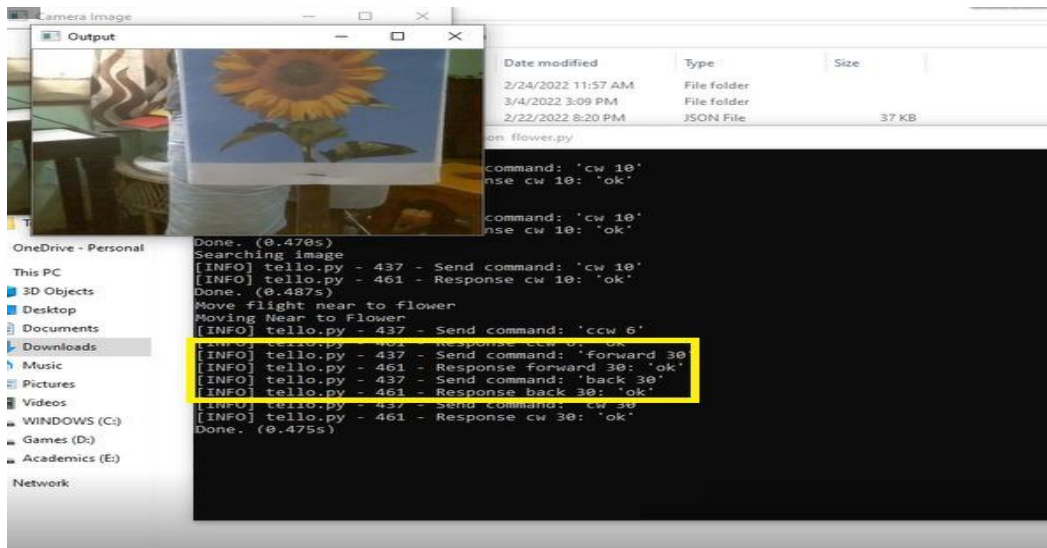


Fig: 5.12: Commands are repeated

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This project has designed a pollination drone that does artificial pollination. In the existing system, watching the insects and other complex agricultural activities has been applied using deep learning and computer vision. In this project artificial pollination method has been applied using an Efficientnet algorithm and wireless drone if any kind of obstacle comes in front of it, it will automatically turn its path in different directions. This will help to detect the obstacle. The drone has a stable performance of hovering up to 2 to 3 feet of height. Thus, by this we have successfully designed an obstacle avoidance drone for artificial pollination so, we can see that non-natural pollination is helpful and can rise plant output. The positive completion of this project will impact the field of non-natural pollination in agriculture.

6.2 FUTURE WORK

In the coming future, we review the application of drone technology. It can promote advanced obstacle pollination drone systems with more features than this system. So, there is more chance to develop or convert this project in many ways. This can be able to promote drones that will fly avoiding obstacles resulting in many real-time applications in the coming future.

REFERENCE

- [1] Jung Uk Kim and Yong Man Ro Image and Video Systems Lab, School of Electrical Engineering, KAIST, South Korea” Attentive Layer Separation for Object Classification and Object Localization in Object Detection” IEEE Transactions on Image Processing (ICIP)
- [2] Malika Nisal Ratnayake , Adrian G. Dyer, , Alan Dorin Faculty of Information Technology, Monash University, Australia School of Media and Communication, RMIT University, Australia Department of Physiology, Monash University, Australia” Towards Computer Vision and Deep Learning Facilitated Pollination Monitoring for Agriculture” IEEE Transaction on Computer Vision
- [3] Ming-Liang Gao, Yuan-Ru Zang, Jin Shen, Yu-Chen Zhang, De-Shui Yu School of Electrical & Electronic Engineering, Shandong University of Technology “Visual tracking based on flower pollination algorithm” Chinese Control Conference (CCC)
- [4] Mohammad Mozaffari¹, Walid Saad², Mehdi Bennis³, and M´erouane Debbah⁴ “Communications and Control for Wireless Drone-Based Antenna Array”, IEEE Transactions on Communications
- [5] Weisen Shi, Student Member, IEEE Junling Li, Student Member, IEEE Nan Cheng, Member, IEEE Feng Lyu, Member, IEEE Shan Zhang, Member, IEEE Haibo Zhou, Senior Member, IEEE and Xuemin (Sherman) Shen, Fellow, IEEE “Multi-Drone 3D Trajectory Planning and Scheduling in Drone Assisted Radio Access Networks” IEEE Transactions on Vehicular Technology.
- [6] Yijun yu¹(member, iee), danny barthaud¹, blaine a. Price¹, arosha k. Bandara¹, andrea zisman¹, and bashar nuseibeh^{1,2} “LiveBox: A Self-Adaptive Forensic-Ready Service for Drones” IEEE Access.
- [7] YI CHEN¹ , (SENIOR MEMBER, IEEE), YUN LI¹ ,(SENIOR MEMBER, IEEE) Dongguan Industry 4.0 Artificial Intelligence Laboratory, School of Computer Science and Engineering, Dongguan University of Technology” Intelligent Autonomous Pollination for Future Farming” IEEE Access.

APPENDIX

SOURCE CODE

```
import torch
import torch.nn as nn
import numpy as np

import time
start = time.time()

IMAGE_PREDICTION_THRESH = 0.9
THRESHOLD = 0.4
class BBoxTransform(nn.Module):

    def __init__(self, mean=None, std=None):
        super(BBoxTransform, self).__init__()
        if mean is None:
            self.mean = torch.from_numpy(np.array([0, 0, 0, 0]).astype(np.float32))
        else:
            self.mean = mean
        if std is None:
            self.std = torch.from_numpy(np.array([0.1, 0.1, 0.2, 0.2]).astype(np.float32))
        else:
            self.std = std
        if torch.cuda.is_available():
            self.mean = self.mean.cuda()
            self.std = self.std.cuda()

    def forward(self, boxes, deltas):

        widths = boxes[:, :, 2] - boxes[:, :, 0]
        heights = boxes[:, :, 3] - boxes[:, :, 1]
        ctr_x = boxes[:, :, 0] + 0.5 * widths
        ctr_y = boxes[:, :, 1] + 0.5 * heights

        dx = deltas[:, :, 0] * self.std[0] + self.mean[0]
        dy = deltas[:, :, 1] * self.std[1] + self.mean[1]
        dw = deltas[:, :, 2] * self.std[2] + self.mean[2]
        dh = deltas[:, :, 3] * self.std[3] + self.mean[3]

        pred_ctr_x = ctr_x + dx * widths
        pred_ctr_y = ctr_y + dy * heights
        pred_w = torch.exp(dw) * widths
        pred_h = torch.exp(dh) * heights
```

```

    pred_boxes_x1 = pred_ctr_x - 0.5 * pred_w
    pred_boxes_y1 = pred_ctr_y - 0.5 * pred_h
    pred_boxes_x2 = pred_ctr_x + 0.5 * pred_w
    pred_boxes_y2 = pred_ctr_y + 0.5 * pred_h

    pred_boxes = torch.stack([pred_boxes_x1, pred_boxes_y1, pred_boxes_x2,
pred_boxes_y2], dim=2)

    return pred_boxes

```

```

class ClipBoxes(nn.Module):

```

```

    def __init__(self):
        super(ClipBoxes, self).__init__()

    def forward(self, boxes, img):
        batch_size, num_channels, height, width = img.shape

        boxes[:, :, 0] = torch.clamp(boxes[:, :, 0], min=0)
        boxes[:, :, 1] = torch.clamp(boxes[:, :, 1], min=0)

        boxes[:, :, 2] = torch.clamp(boxes[:, :, 2], max=width)
        boxes[:, :, 3] = torch.clamp(boxes[:, :, 3], max=height)

        return boxes

```

```

class Anchors(nn.Module):

```

```

    def __init__(self, pyramid_levels=None, strides=None, sizes=None, ratios=None,
scales=None):
        super(Anchors, self).__init__()

        if pyramid_levels is None:
            self.pyramid_levels = [3, 4, 5, 6, 7]
        if strides is None:
            self.strides = [2 ** x for x in self.pyramid_levels]
        if sizes is None:
            self.sizes = [2 ** (x + 2) for x in self.pyramid_levels]
        if ratios is None:
            self.ratios = np.array([0.5, 1, 2])
        if scales is None:
            self.scales = np.array([2 ** 0, 2 ** (1.0 / 3.0), 2 ** (2.0 / 3.0)])

```

```

    def forward(self, image):

```

```

image_shape = image.shape[2:]
image_shape = np.array(image_shape)
image_shapes = [(image_shape + 2 ** x - 1) // (2 ** x) for x in self.pyramid_levels]

all_anchors = np.zeros((0, 4)).astype(np.float32)

for idx, p in enumerate(self.pyramid_levels):
    anchors = generate_anchors(base_size=self.sizes[idx], ratios=self.ratios,
scales=self.scales)
    shifted_anchors = shift(image_shapes[idx], self.strides[idx], anchors)
    all_anchors = np.append(all_anchors, shifted_anchors, axis=0)

all_anchors = np.expand_dims(all_anchors, axis=0)

anchors = torch.from_numpy(all_anchors.astype(np.float32))
if torch.cuda.is_available():
    anchors = anchors.cuda()
return anchors

def generate_anchors(base_size=16, ratios=None, scales=None):
    if ratios is None:
        ratios = np.array([0.5, 1, 2])

    if scales is None:
        scales = np.array([2 ** 0, 2 ** (1.0 / 3.0), 2 ** (2.0 / 3.0)])

    num_anchors = len(ratios) * len(scales)
    anchors = np.zeros((num_anchors, 4))
    anchors[:, 2:] = base_size * np.tile(scales, (2, len(ratios))).T
    areas = anchors[:, 2] * anchors[:, 3]
    anchors[:, 2] = np.sqrt(areas / np.repeat(ratios, len(scales)))
    anchors[:, 3] = anchors[:, 2] * np.repeat(ratios, len(scales))
    anchors[:, 0:2] -= np.tile(anchors[:, 2] * 0.5, (2, 1)).T
    anchors[:, 1:2] -= np.tile(anchors[:, 3] * 0.5, (2, 1)).T

    return anchors

def shift(shape, stride, anchors):
    shift_x = (np.arange(0, shape[1]) + 0.5) * stride
    shift_y = (np.arange(0, shape[0]) + 0.5) * stride
    shift_x, shift_y = np.meshgrid(shift_x, shift_y)
    shifts = np.vstack((
        shift_x.ravel(), shift_y.ravel(),
        shift_x.ravel(), shift_y.ravel()

```

```

)).transpose()

A = anchors.shape[0]
K = shifts.shape[0]
all_anchors = (anchors.reshape((1, A, 4)) + shifts.reshape((1, K, 4)).transpose((1, 0,
2)))
all_anchors = all_anchors.reshape((K * A, 4))

return all_anchors

```

```

import torch
import torch.nn as nn

```

```

def calc_iou(a, b):

    area = (b[:, 2] - b[:, 0]) * (b[:, 3] - b[:, 1])
    iw = torch.min(torch.unsqueeze(a[:, 2], dim=1), b[:, 2]) -
torch.max(torch.unsqueeze(a[:, 0], 1), b[:, 0])
    ih = torch.min(torch.unsqueeze(a[:, 3], dim=1), b[:, 3]) -
torch.max(torch.unsqueeze(a[:, 1], 1), b[:, 1])
    iw = torch.clamp(iw, min=0)
    ih = torch.clamp(ih, min=0)
    ua = torch.unsqueeze((a[:, 2] - a[:, 0]) * (a[:, 3] - a[:, 1]), dim=1) + area - iw * ih
    ua = torch.clamp(ua, min=1e-8)
    intersection = iw * ih
    IoU = intersection / ua

    return IoU

```

```

class FocalLoss(nn.Module):
    def __init__(self):
        super(FocalLoss, self).__init__()

    def forward(self, classifications, regressions, anchors, annotations):
        alpha = 0.25
        gamma = 2.0
        batch_size = classifications.shape[0]
        classification_losses = []
        regression_losses = []

        anchor = anchors[0, :, :]

        anchor_widths = anchor[:, 2] - anchor[:, 0]
        anchor_heights = anchor[:, 3] - anchor[:, 1]

```

```

anchor_ctr_x = anchor[:, 0] + 0.5 * anchor_widths
anchor_ctr_y = anchor[:, 1] + 0.5 * anchor_heights

for j in range(batch_size):

    classification = classifications[j, :, :]
    regression = regressions[j, :, :]

    bbox_annotation = annotations[j, :, :]
    bbox_annotation = bbox_annotation[bbox_annotation[:, 4] != -1]

    if bbox_annotation.shape[0] == 0:
        if torch.cuda.is_available():
            regression_losses.append(torch.tensor(0).float().cuda())
            classification_losses.append(torch.tensor(0).float().cuda())
        else:
            regression_losses.append(torch.tensor(0).float())
            classification_losses.append(torch.tensor(0).float())

    continue

classification = torch.clamp(classification, 1e-4, 1.0 - 1e-4)

IoU = calc_iou(anchors[0, :, :], bbox_annotation[:, :4])

IoU_max, IoU_argmax = torch.max(IoU, dim=1)

# compute the loss for classification
targets = torch.ones(classification.shape) * -1
if torch.cuda.is_available():
    targets = targets.cuda()

targets[torch.lt(IoU_max, 0.4), :] = 0

positive_indices = torch.ge(IoU_max, 0.5)

num_positive_anchors = positive_indices.sum()

assigned_annotations = bbox_annotation[IoU_argmax, :]

targets[positive_indices, :] = 0
targets[positive_indices, assigned_annotations[positive_indices, 4].long()] = 1

alpha_factor = torch.ones(targets.shape) * alpha
if torch.cuda.is_available():
    alpha_factor = alpha_factor.cuda()

```

```

alpha_factor = torch.where(torch.eq(targets, 1.), alpha_factor, 1. - alpha_factor)
focal_weight = torch.where(torch.eq(targets, 1.), 1. - classification, classification)
focal_weight = alpha_factor * torch.pow(focal_weight, gamma)

bce = -(targets * torch.log(classification) + (1.0 - targets) * torch.log(1.0 -
classification))

cls_loss = focal_weight * bce

zeros = torch.zeros(cls_loss.shape)
if torch.cuda.is_available():
    zeros = zeros.cuda()
cls_loss = torch.where(torch.ne(targets, -1.0), cls_loss, zeros)

classification_losses.append(cls_loss.sum() /
torch.clamp(num_positive_anchors.float(), min=1.0))

if positive_indices.sum() > 0:
    assigned_annotations = assigned_annotations[positive_indices, :]

    anchor_widths_pi = anchor_widths[positive_indices]
    anchor_heights_pi = anchor_heights[positive_indices]
    anchor_ctr_x_pi = anchor_ctr_x[positive_indices]
    anchor_ctr_y_pi = anchor_ctr_y[positive_indices]

    gt_widths = assigned_annotations[:, 2] - assigned_annotations[:, 0]
    gt_heights = assigned_annotations[:, 3] - assigned_annotations[:, 1]
    gt_ctr_x = assigned_annotations[:, 0] + 0.5 * gt_widths
    gt_ctr_y = assigned_annotations[:, 1] + 0.5 * gt_heights

    gt_widths = torch.clamp(gt_widths, min=1)
    gt_heights = torch.clamp(gt_heights, min=1)

    targets_dx = (gt_ctr_x - anchor_ctr_x_pi) / anchor_widths_pi
    targets_dy = (gt_ctr_y - anchor_ctr_y_pi) / anchor_heights_pi
    targets_dw = torch.log(gt_widths / anchor_widths_pi)
    targets_dh = torch.log(gt_heights / anchor_heights_pi)

    targets = torch.stack((targets_dx, targets_dy, targets_dw, targets_dh))
    targets = targets.t()

    norm = torch.Tensor([[0.1, 0.1, 0.2, 0.2]])
    if torch.cuda.is_available():
        norm = norm.cuda()

```

```

    targets = targets / norm

    regression_diff = torch.abs(targets - regression[positive_indices, :])

    regression_loss = torch.where(
        torch.le(regression_diff, 1.0 / 9.0),
        0.5 * 9.0 * torch.pow(regression_diff, 2),
        regression_diff - 0.5 / 9.0
    )
    regression_losses.append(regression_loss.mean())
else:
    if torch.cuda.is_available():
        regression_losses.append(torch.tensor(0).float().cuda())
    else:
        regression_losses.append(torch.tensor(0).float())

    return torch.stack(classification_losses).mean(dim=0, keepdim=True),
    torch.stack(regression_losses).mean(dim=0,
                                         keepdim=True)

```

```

import torch.nn as nn
import torch
import math
from efficientnet_pytorch import EfficientNet as EffNet
from torchvision.ops.bboxes import nms as nms_torch

```

```

def nms(dets, thresh):
    return nms_torch(dets[:, :4], dets[:, 4], thresh)

```

```

class ConvBlock(nn.Module):
    def __init__(self, num_channels):
        super(ConvBlock, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(num_channels, num_channels, kernel_size=3, stride=1, padding=1,
groups=num_channels),
            nn.Conv2d(num_channels, num_channels, kernel_size=1, stride=1, padding=0),
            nn.BatchNorm2d(num_features=num_channels, momentum=0.9997, eps=4e-5),
            nn.ReLU())

```

```

    def forward(self, input):
        return self.conv(input)

```

```

class BiFPN(nn.Module):

```

```

def __init__(self, num_channels, epsilon=1e-4):
    super(BiFPN, self).__init__()
    self.epsilon = epsilon
    # Conv layers
    self.conv6_up = ConvBlock(num_channels)
    self.conv5_up = ConvBlock(num_channels)
    self.conv4_up = ConvBlock(num_channels)
    self.conv3_up = ConvBlock(num_channels)
    self.conv4_down = ConvBlock(num_channels)
    self.conv5_down = ConvBlock(num_channels)
    self.conv6_down = ConvBlock(num_channels)
    self.conv7_down = ConvBlock(num_channels)

    # Feature scaling layers
    self.p6_upsample = nn.Upsample(scale_factor=2, mode='nearest')
    self.p5_upsample = nn.Upsample(scale_factor=2, mode='nearest')
    self.p4_upsample = nn.Upsample(scale_factor=2, mode='nearest')
    self.p3_upsample = nn.Upsample(scale_factor=2, mode='nearest')

    self.p4_downsample = nn.MaxPool2d(kernel_size=2)
    self.p5_downsample = nn.MaxPool2d(kernel_size=2)
    self.p6_downsample = nn.MaxPool2d(kernel_size=2)
    self.p7_downsample = nn.MaxPool2d(kernel_size=2)

    # Weight
    self.p6_w1 = nn.Parameter(torch.ones(2))
    self.p6_w1_relu = nn.ReLU()
    self.p5_w1 = nn.Parameter(torch.ones(2))
    self.p5_w1_relu = nn.ReLU()
    self.p4_w1 = nn.Parameter(torch.ones(2))
    self.p4_w1_relu = nn.ReLU()
    self.p3_w1 = nn.Parameter(torch.ones(2))
    self.p3_w1_relu = nn.ReLU()

    self.p4_w2 = nn.Parameter(torch.ones(3))
    self.p4_w2_relu = nn.ReLU()
    self.p5_w2 = nn.Parameter(torch.ones(3))
    self.p5_w2_relu = nn.ReLU()
    self.p6_w2 = nn.Parameter(torch.ones(3))
    self.p6_w2_relu = nn.ReLU()
    self.p7_w2 = nn.Parameter(torch.ones(2))
    self.p7_w2_relu = nn.ReLU()

```

```

def forward(self, inputs):

```

```

    """

```

```

        P7_0 ----- P7_2 ----->

```

```

P6_0 ----- P6_1 ----- P6_2 ----->

P5_0 ----- P5_1 ----- P5_2 ----->

P4_0 ----- P4_1 ----- P4_2 ----->

P3_0 ----- P3_2 ----->
,,,,,

```

```

# P3_0, P4_0, P5_0, P6_0 and P7_0
p3_in, p4_in, p5_in, p6_in, p7_in = inputs
# P7_0 to P7_2
# Weights for P6_0 and P7_0 to P6_1
p6_w1 = self.p6_w1_relu(self.p6_w1)
weight = p6_w1 / (torch.sum(p6_w1, dim=0) + self.epsilon)
# Connections for P6_0 and P7_0 to P6_1 respectively
p6_up = self.conv6_up(weight[0] * p6_in + weight[1] * self.p6_upsample(p7_in))
# Weights for P5_0 and P6_0 to P5_1
p5_w1 = self.p5_w1_relu(self.p5_w1)
weight = p5_w1 / (torch.sum(p5_w1, dim=0) + self.epsilon)
# Connections for P5_0 and P6_0 to P5_1 respectively
p5_up = self.conv5_up(weight[0] * p5_in + weight[1] * self.p5_upsample(p6_up))
# Weights for P4_0 and P5_0 to P4_1
p4_w1 = self.p4_w1_relu(self.p4_w1)
weight = p4_w1 / (torch.sum(p4_w1, dim=0) + self.epsilon)
# Connections for P4_0 and P5_0 to P4_1 respectively
p4_up = self.conv4_up(weight[0] * p4_in + weight[1] * self.p4_upsample(p5_up))

# Weights for P3_0 and P4_1 to P3_2
p3_w1 = self.p3_w1_relu(self.p3_w1)
weight = p3_w1 / (torch.sum(p3_w1, dim=0) + self.epsilon)
# Connections for P3_0 and P4_1 to P3_2 respectively
p3_out = self.conv3_up(weight[0] * p3_in + weight[1] * self.p3_upsample(p4_up))

# Weights for P4_0, P4_1 and P3_2 to P4_2
p4_w2 = self.p4_w2_relu(self.p4_w2)
weight = p4_w2 / (torch.sum(p4_w2, dim=0) + self.epsilon)
# Connections for P4_0, P4_1 and P3_2 to P4_2 respectively
p4_out = self.conv4_down(
    weight[0] * p4_in + weight[1] * p4_up + weight[2] * self.p4_downsample(p3_out))
# Weights for P5_0, P5_1 and P4_2 to P5_2
p5_w2 = self.p5_w2_relu(self.p5_w2)
weight = p5_w2 / (torch.sum(p5_w2, dim=0) + self.epsilon)
# Connections for P5_0, P5_1 and P4_2 to P5_2 respectively
p5_out = self.conv5_down(

```

```

        weight[0] * p5_in + weight[1] * p5_up + weight[2] * self.p5_downsample(p4_out))
# Weights for P6_0, P6_1 and P5_2 to P6_2
p6_w2 = self.p6_w2_relu(self.p6_w2)
weight = p6_w2 / (torch.sum(p6_w2, dim=0) + self.epsilon)
# Connections for P6_0, P6_1 and P5_2 to P6_2 respectively
p6_out = self.conv6_down(
    weight[0] * p6_in + weight[1] * p6_up + weight[2] * self.p6_downsample(p5_out))
# Weights for P7_0 and P6_2 to P7_2
p7_w2 = self.p7_w2_relu(self.p7_w2)
weight = p7_w2 / (torch.sum(p7_w2, dim=0) + self.epsilon)
# Connections for P7_0 and P6_2 to P7_2
p7_out = self.conv7_down(weight[0] * p7_in + weight[1] *
self.p7_downsample(p6_out))

return p3_out, p4_out, p5_out, p6_out, p7_out

```

```

class Regressor(nn.Module):
    def __init__(self, in_channels, num_anchors, num_layers):
        super(Regressor, self).__init__()
        layers = []
        for _ in range(num_layers):
            layers.append(nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=1,
padding=1))
            layers.append(nn.ReLU(True))
        self.layers = nn.Sequential(*layers)
        self.header = nn.Conv2d(in_channels, num_anchors * 4, kernel_size=3, stride=1,
padding=1)

    def forward(self, inputs):
        inputs = self.layers(inputs)
        inputs = self.header(inputs)
        output = inputs.permute(0, 2, 3, 1)
        return output.contiguous().view(output.shape[0], -1, 4)

```

```

class Classifier(nn.Module):
    def __init__(self, in_channels, num_anchors, num_classes, num_layers):
        super(Classifier, self).__init__()
        self.num_anchors = num_anchors
        self.num_classes = num_classes
        layers = []
        for _ in range(num_layers):
            layers.append(nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=1,
padding=1))
            layers.append(nn.ReLU(True))

```

```

        self.layers = nn.Sequential(*layers)
        self.header = nn.Conv2d(in_channels, num_anchors * num_classes,
kernel_size=3, stride=1, padding=1)
        self.act = nn.Sigmoid()

    def forward(self, inputs):
        inputs = self.layers(inputs)
        inputs = self.header(inputs)
        inputs = self.act(inputs)
        inputs = inputs.permute(0, 2, 3, 1)
        output = inputs.contiguous().view(inputs.shape[0], inputs.shape[1], inputs.shape[2],
self.num_anchors,
                                     self.num_classes)
        return output.contiguous().view(output.shape[0], -1, self.num_classes)

```

```

class EfficientNet(nn.Module):
    def __init__(self, ):
        super(EfficientNet, self).__init__()
        model = EffNet.from_pretrained('efficientnet-b0')
        del model._conv_head
        del model._bn1
        del model._avg_pooling
        del model._dropout
        del model._fc
        self.model = model

    def forward(self, x):
        x = self.model._swish(self.model._bn0(self.model._conv_stem(x)))
        feature_maps = []
        for idx, block in enumerate(self.model._blocks):
            drop_connect_rate = self.model._global_params.drop_connect_rate
            if drop_connect_rate:
                drop_connect_rate *= float(idx) / len(self.model._blocks)
            x = block(x, drop_connect_rate=drop_connect_rate)
            if block._depthwise_conv.stride == [2, 2]:
                feature_maps.append(x)

        return feature_maps[1:]

```

```

class EfficientDet(nn.Module):
    def __init__(self, num_anchors=9, num_classes=20, compound_coef=0):
        super(EfficientDet, self).__init__()
        self.compound_coef = compound_coef

```

```

self.num_channels = [64, 88, 112, 160, 224, 288, 384, 384][self.compound_coef]

self.conv3 = nn.Conv2d(40, self.num_channels, kernel_size=1, stride=1,
padding=0)
self.conv4 = nn.Conv2d(80, self.num_channels, kernel_size=1, stride=1,
padding=0)
self.conv5 = nn.Conv2d(192, self.num_channels, kernel_size=1, stride=1,
padding=0)
self.conv6 = nn.Conv2d(192, self.num_channels, kernel_size=3, stride=2,
padding=1)
self.conv7 = nn.Sequential(nn.ReLU(),
nn.Conv2d(self.num_channels, self.num_channels, kernel_size=3,
stride=2, padding=1))

self.bifpn = nn.Sequential(*[BiFPN(self.num_channels) for _ in range(min(2 +
self.compound_coef, 8))])

self.num_classes = num_classes
self.regressor = Regressor(in_channels=self.num_channels,
num_anchors=num_anchors,
num_layers=3 + self.compound_coef // 3)
self.classifier = Classifier(in_channels=self.num_channels,
num_anchors=num_anchors, num_classes=num_classes,
num_layers=3 + self.compound_coef // 3)

self.anchors = Anchors()
self.regressBoxes = BBoxTransform()
self.clipBoxes = ClipBoxes()
self.focalLoss = FocalLoss()

for m in self.modules():
    if isinstance(m, nn.Conv2d):
        n = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
        m.weight.data.normal_(0, math.sqrt(2. / n))
    elif isinstance(m, nn.BatchNorm2d):
        m.weight.data.fill_(1)
        m.bias.data.zero_()

prior = 0.01

self.classifier.header.weight.data.fill_(0)
self.classifier.header.bias.data.fill_(-math.log((1.0 - prior) / prior))

self.regressor.header.weight.data.fill_(0)
self.regressor.header.bias.data.fill_(0)

```

```

self.backbone_net = EfficientNet()

def freeze_bn(self):
    for m in self.modules():
        if isinstance(m, nn.BatchNorm2d):
            m.eval()

def forward(self, inputs):
    if len(inputs) == 2:
        is_training = True
        img_batch, annotations = inputs
    else:
        is_training = False
        img_batch = inputs

    c3, c4, c5 = self.backbone_net(img_batch)
    p3 = self.conv3(c3)
    p4 = self.conv4(c4)
    p5 = self.conv5(c5)
    p6 = self.conv6(c5)
    p7 = self.conv7(p6)

    features = [p3, p4, p5, p6, p7]
    features = self.bifpn(features)

    regression = torch.cat([self.regressor(feature) for feature in features], dim=1)
    classification = torch.cat([self.classifier(feature) for feature in features], dim=1)
    anchors = self.anchors(img_batch)

    if is_training:
        return self.focalLoss(classification, regression, anchors, annotations)
    else:
        transformed_anchors = self.regressBoxes(anchors, regression)
        transformed_anchors = self.clipBoxes(transformed_anchors, img_batch)

        scores = torch.max(classification, dim=2, keepdim=True)[0]

        scores_over_thresh = (scores > 0.05)[0, :, 0]

        if scores_over_thresh.sum() == 0:
            return [torch.zeros(0), torch.zeros(0), torch.zeros(0, 4)]

        classification = classification[:, scores_over_thresh, :]
        transformed_anchors = transformed_anchors[:, scores_over_thresh, :]
        scores = scores[:, scores_over_thresh, :]

```

```

anchors_nms_idx = nms(torch.cat([transformed_anchors, scores], dim=2)[0, :, :],
0.5)

nms_scores, nms_class = classification[0, anchors_nms_idx, :].max(dim=1)

return [nms_scores, nms_class, transformed_anchors[0, anchors_nms_idx, :]]

```

```

from tensorboardX import SummaryWriter

```

```

def count_parameters(model):

```

```

    return sum(p.numel() for p in model.parameters() if p.requires_grad)

```

```

model = EfficientDet(num_classes=80)

```

```

colors = [(39, 129, 113), (164, 80, 133), (83, 122, 114), (99, 81, 172), (95, 56, 104), (37,
84, 86), (14, 89, 122),
          (80, 7, 65), (10, 102, 25), (90, 185, 109), (106, 110, 132), (169, 158, 85), (188,
185, 26), (103, 1, 17),
          (82, 144, 81), (92, 7, 184), (49, 81, 155), (179, 177, 69), (93, 187, 158), (13, 39,
73), (12, 50, 60),
          (16, 179, 33), (112, 69, 165), (15, 139, 63), (33, 191, 159), (182, 173, 32), (34,
113, 133), (90, 135, 34),
          (53, 34, 86), (141, 35, 190), (6, 171, 8), (118, 76, 112), (89, 60, 55), (15, 54, 88),
(112, 75, 181),
          (42, 147, 38), (138, 52, 63), (128, 65, 149), (106, 103, 24), (168, 33, 45), (28, 136,
135), (86, 91, 108),
          (52, 11, 76), (142, 6, 189), (57, 81, 168), (55, 19, 148), (182, 101, 89), (44, 65,
179), (1, 33, 26),
          (122, 164, 26), (70, 63, 134), (137, 106, 82), (120, 118, 52), (129, 74, 42), (182,
147, 112), (22, 157, 50),
          (56, 50, 20), (2, 22, 177), (156, 100, 106), (21, 35, 42), (13, 8, 121), (142, 92, 28),
(45, 118, 33),
          (105, 118, 30), (7, 185, 124), (46, 34, 146), (105, 184, 169), (22, 18, 5), (147, 71,
73), (181, 64, 91),
          (31, 39, 184), (164, 179, 33), (96, 50, 18), (95, 15, 106), (113, 68, 54), (136, 116,
112), (119, 139, 130),
          (31, 139, 34), (66, 6, 127), (62, 39, 2), (49, 99, 180), (49, 119, 155), (153, 50,
183), (125, 38, 3),
          (129, 87, 143), (49, 87, 40), (128, 62, 120), (73, 85, 148), (28, 144, 118), (29, 9,
24), (175, 45, 108),
          (81, 175, 64), (178, 19, 157), (74, 188, 190), (18, 114, 2), (62, 128, 96), (21, 3,
150), (0, 6, 95),
          (2, 20, 184), (122, 37, 185)]

```

```

import os
import torch
import torch.nn as nn
from tensorboardX import SummaryWriter
import numpy as np
import cv2
import time as time
from imutils.video import FPS
from djitellopy import Tello

class Infer():
    def __init__(self, verbose=1):
        self.system_dict = {};
        self.system_dict["verbose"] = verbose;
        self.system_dict["local"] = {};
        self.system_dict["local"]["common_size"] = 512;
        self.system_dict["local"]["mean"] = np.array([[[[0.485, 0.456, 0.406]]]])
        self.system_dict["local"]["std"] = np.array([[[[0.229, 0.224, 0.225]]]])

    def Model(self, model_dir="trained/"):
        self.system_dict["local"]["model"] = torch.load(model_dir +
"/signatrix_efficientdet_coco.pth").module
        if torch.cuda.is_available():
            self.system_dict["local"]["model"] = self.system_dict["local"]["model"].cuda();

    def Predict(self, img_path, class_list, vis_threshold = 0.4,output_folder = 'Inference'):

        if not os.path.exists(output_folder):
            os.makedirs(output_folder)

        img = cv2.cvtColor(img_path, cv2.COLOR_BGR2RGB);
        image = img.astype(np.float32) / 255.;
        image = (image.astype(np.float32) - self.system_dict["local"]["mean"]) /
self.system_dict["local"]["std"]
        height, width, _ = image.shape
        if height > width:
            scale = self.system_dict["local"]["common_size"] / height
            resized_height = self.system_dict["local"]["common_size"]
            resized_width = int(width * scale)
        else:
            scale = self.system_dict["local"]["common_size"] / width
            resized_height = int(height * scale)
            resized_width = self.system_dict["local"]["common_size"]

        image = cv2.resize(image, (resized_width, resized_height))

```

```

        new_image = np.zeros((self.system_dict["local"]["common_size"],
self.system_dict["local"]["common_size"], 3))
        new_image[0:resized_height, 0:resized_width] = image

    img = torch.from_numpy(new_image)

    t0 = time.time()
    with torch.no_grad():
        if torch.cuda.is_available():
            scores, labels, boxes =
self.system_dict["local"]["model"](img.cuda().permute(2, 0, 1).float().unsqueeze(dim=0))
        else:
            scores, labels, boxes = self.system_dict["local"]["model"](img.permute(2, 0,
1).float().unsqueeze(dim=0))
            boxes /= scale;
    duration = time.time() - t0
    print('Done. (%.3fs)' % (time.time() - t0))

    try:
        if boxes.shape[0] > 0:
            for box_id in range(boxes.shape[0]):
                pred_prob = float(scores[box_id])
                if pred_prob < vis_threshold:
                    break
                pred_label = int(labels[box_id])
                xmin, ymin, xmax, ymax = boxes[box_id, :]
                color = colors[pred_label]
                cv2.rectangle(img_path, (xmin, ymin), (xmax, ymax), color, 2)
                text_size = cv2.getTextSize(class_list[pred_label] + ' : %.2f' % pred_prob,
cv2.FONT_HERSHEY_PLAIN, 1, 1)[0]

                cv2.rectangle(img_path, (xmin, ymin), (xmin + text_size[0] + 3, ymin +
text_size[1] + 4), color, -1)
                cv2.putText(
                    img_path, class_list[pred_label] + ' : %.2f' % pred_prob,
                    (xmin, ymin + text_size[1] + 4), cv2.FONT_HERSHEY_PLAIN, 1,
                    (255, 255, 255), 1)

            return duration, scores, labels, boxes

    except:
        print("NO Object Detected")
        return None

def predict_batch_of_images(self, img_folder, class_list, vis_threshold = 0.7,

```

```
output_folder='Inference'):
```

```
    all_filenames = os.listdir(img_folder)
    all_filenames.sort()
    generated_count = 0
    for filename in all_filenames:
        img_path = "{}/{}/{}".format(img_folder, filename)
        try:
            self.Predict(img_path , class_list, vis_threshold ,output_folder)
            generated_count += 1
        except:
            continue
    print("Objects detected for {} images".format(generated_count))
```

```
gtf = Infer()
gtf.Model(model_dir="trained_weight/")
```

```
count = 1
direction = 1
Search_count = 1
def flight_control_search_image():
    global count
    global direction
    print("Searching image")
    if(count >= 30 and direction == 1):
        direction = 2
        count = 1
    elif(count >= 30 and direction == 2):
        direction = 1
        count = 1
    if(direction == 1):
        tello.rotate_clockwise(10)
        count = count+1
    elif(direction == 2):
        tello.rotate_counter_clockwise(10)
        count = count+1
```

```
def move_flight_near_flower(x,y):
    print("Moving Near to Flower")
    deg = int(x/10)

    if(deg <180):
        tello.rotate_counter_clockwise(deg)
    else:
        tello.rotate_clockwise(deg)
```

```

time.sleep(1)
tello.move_forward(30)
time.sleep(1)
tello.move_back(30)
time.sleep(1)
if(direction == 1):
    tello.rotate_clockwise(30)

elif(direction == 2):
    tello.rotate_counter_clockwise(30)

```

```

time.sleep(1)

```

```

#extract class list from our annotations
import json
import imutils
import torchvision
with open('_annotations.coco.json') as json_file:
    data = json.load(json_file)
class_list = []
for category in data['categories']:
    class_list.append(category['name'])
#print(category['name'])

```

```

#cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
fps = FPS().start()

```

```

tello = Tello()
tello.connect()
tello.takeoff()
time.sleep(5)
tello.streamon()
time.sleep(2)

```

```

while True:
    x_coor =0
    y_coor =0
    frame_count =0
    image_detected = False

```

```

while (frame_count <150):
    img_path = tello.get_frame_read().frame
    #time.sleep(0.01)
    frame_count =frame_count+1

img_path = cv2.resize(img_path, (360, 240))

cv2.imshow("Camera Image", img_path)
duration, scores, labels, boxes = gtf.Predict(img_path, class_list, vis_threshold=0.7)
pred = {}
pred['boxes'] = boxes
pred['scores'] = scores
pred['labels'] = labels

import torchvision

def apply_nms(orig_prediction, iou_thresh=0.7):

    # torchvision returns the indices of the bboxes to keep
    keep = torchvision.ops.nms(orig_prediction['boxes'], orig_prediction['scores'],
iou_thresh)

    final_prediction = orig_prediction
    final_prediction['boxes'] = final_prediction['boxes'][keep]
    final_prediction['scores'] = final_prediction['scores'][keep]
    final_prediction['labels'] = final_prediction['labels'][keep]

    return final_prediction

pred = apply_nms(pred, iou_thresh=0.7)
# print(pred["boxes"][0])
if pred["scores"][0] > 0.70:
    # print(pred["boxes"][0])
    image_detected = True
    x, y, w, h = pred["boxes"][0]
    x_coor = int((w-x)/2)
    y_coor = int((h-y)/2)
    #print("Flower:"+str(pred['scores']))
    print('X:'+str(x_coor)+' Y:'+str(y_coor))

import matplotlib.pyplot as plt

plt.rcParams['text.color'] = 'blue'

cv2.imshow("Output",img_path)

```

```

if(image_detected == False):

    flight_control_search_image()
else:
    print("Move flight near to flower")
    move_flight_near_flower(int(x_coor),int(y_coor))

if(tello.get_battery() <50):
    print("Battery is low")
    tello.land()

key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
if key == ord("q"):
    tello.land()
    break

fps.update()

```