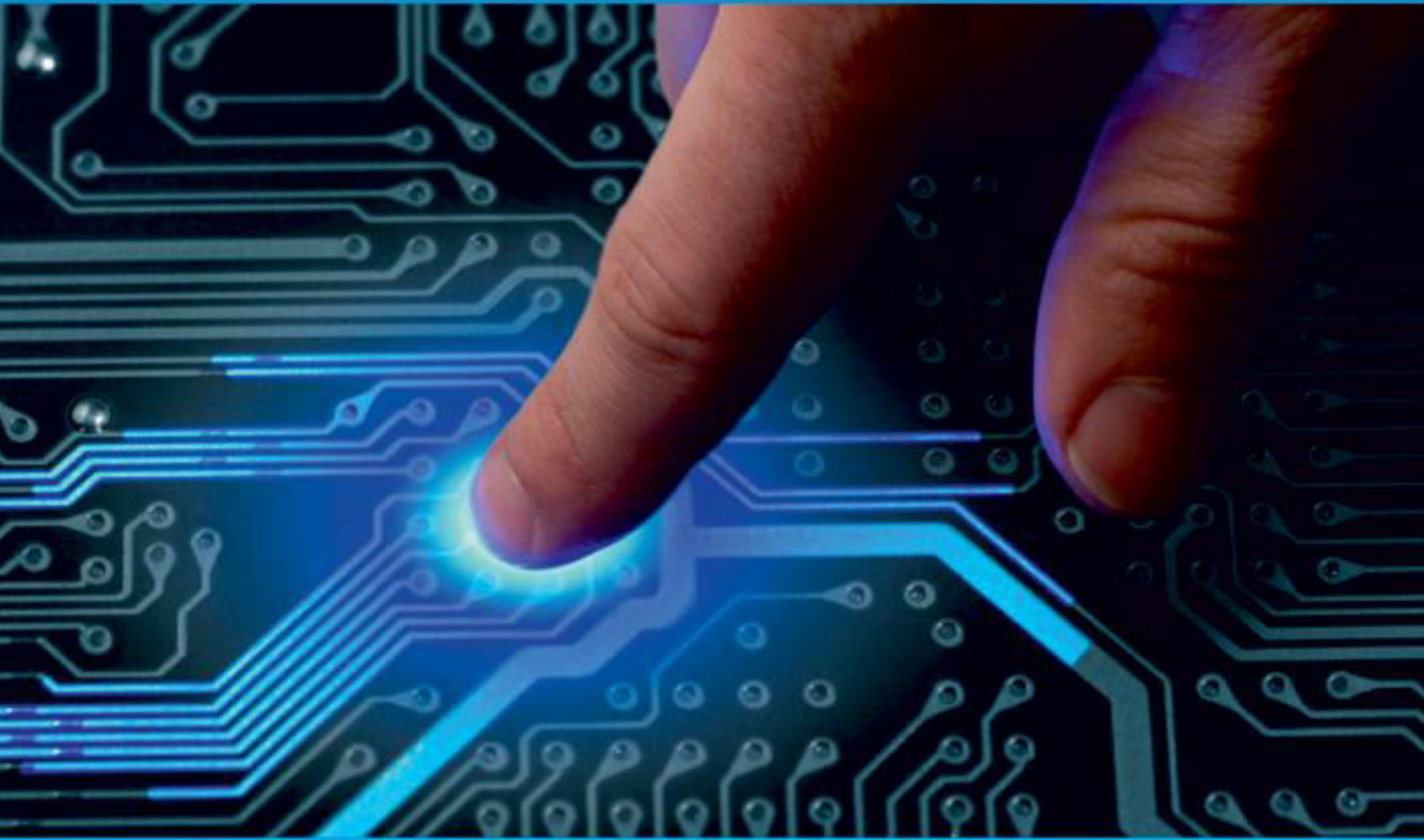
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 6, June 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006227|

Intelligence Surveillance Patrolling Drone (2022)

Abhishek K H1, Aishwarya N2, Akhel P3, Chandrashekar V3*, Dr.Vanajaroselin E.C,

Student, Dept. of ISE, The Oxford College of Engineering, Visvesvaraya Technological University, Karnataka, India

Student, Dept. of ISE, The Oxford College of Engineering, Visvesvaraya Technological University, Karnataka, India

Student, Dept. of ISE, The Oxford College of Engineering, Visvesvaraya Technological University, Karnataka, India

Student, Dept. of ISE, The Oxford College of Engineering, Visvesvaraya Technological University, Karnataka, India

Professor, Dept. of ISE, The Oxford College of Engineering, Visvesvaraya Technological University, Karnataka, India

ABSTRACT: Drones are UAVs that open up the sky for the imagination to soar, and have a wide range of applications in various industries that also have higher requirements in patrolling. The deployment of intelligence and UDP networks actuates the drones during patrolling and surveillance as per the user. After analyzing the challenges of traditional drones used in patrolling, this paper puts forward the intelligence surveillance patrolling drone prototype based on AI image recognition and identification technology and analyzes the flight range, face detection and recognition rate, tracking, network delay, and flight stabilization during patrolling. The use of these technologies can cover police patrol systems in areas where 4G/5G networks are not available.

KEYWORDS: UAV; patrolling; tracking; recognition

I. INTRODUCTION

With the faster growth of communication network and aviation technology, drones as a new type of UAV field has a wide range of applications in military and civilian fields[1]. In the military field, it can be used for reconnaissance, combat, rescue, and so on. In the civilian field, it can be used in security, transportation, energy agriculture and so on which can effectively improve the service capabilities of the industry. In most rural areas, it is not guaranteed that 4G/5G networks would be available for utilizing cloud computation during long-distance patrolling. Traditional drones can range up to 100m. After all patrolling requires an image feedback stream for viewing the scenario. The use of the UDP protocol provides the solution for this and is famous for its low cost. The range extender helps to add more flight range. Face detection and recognition is the most highly demanding application in surveillance. The use of these algorithms can help the patrol unit to get the background details of the criminal and can send the command to track the suspect using the face detection and Viola-Jones algorithm. The use of these technologies can help for easy patrolling in non-network areas during both date and night time.

II. CHALLENGES OF UAV PATROLLING

There are so many challenges during patrolling. Those challenges immediate face recognition, person/face tracking, headcount, and operating drones in areas where the network is not available to perform long-range drone operations. Among all these the pandemic is also one of the toughest challenges in patrolling. The size of a drone is also a challenge during patrolling. Thus we put our paper forward to provide a unique solution for all these challenges specified above.

III. SOLUTIONS FOR UAV PATROLLING

The patrolling system always prefers something autonomous, easy to operate, and immediate availability of personal information both day and night time. So we tried to provide the following solutions that are as follows:-
(1) UAV size and operation: There are several types of drones like tri, quad, Hexa, octa, deca, and so on[2]. In patrolling any of these types with Intel Movidius motherboard can be used. A medium-sized drone cannot be operated in perspective to small and congested areas. So our paper we are using a nano-sized drone that has an equal advantage

of producing less air drag sound from the propellers and is implemented using an Intel Movidius motherboard. Nano drones are the highest requirement drone because of their size and weight. It comes with a 5 to 10 megapixel camera and a 3.7V lithium-ion battery. These Nano sizes can be used for both cloud computing that is online and for offline purposes by adopting UDP protocols. Like other drones, the nano-size UAV have there own advantage and disadvantages. Since it is a nano-sized drone it can be operated in small areas. It can be carried anywhere inside a small box due to its lightweight. Its price is not expensive. But the problem is with respect to its range and battery capacity. It can fly for only 13 minutes.

(2) Face recognition: usually, immediate recognition of a person's face could solve so many problems in patrolling. It helps the patrol system to fix the target of the person so that they could get a clear understanding of who he is and what he's doing there. To achieve this we used a 5 megapixel camera and a face recognition algorithm. Face recognition is a series of several related problems: First, look at a picture and find all the faces in it. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person. Third, be able to pick out unique features of the face that you can use to tell it apart from other people— like how big the eyes are, how long the face is, etc[3]. Finally, compare the unique features of that face to all the people you already know to determine the person's name. As a human, the brain is weird to do all of this automatically and instantly. In fact, humans are *too good* at recognizing faces and end up seeing faces in everyday objects. Computers are not capable of this kind of high-level generalization, so we have to teach them how to do each step in this process separately.

(3) Person/face tracking: after recognizing the face and match found in the local disk, it is now time to track the face or the person by the drone[4]. This is a way more complex operation carried out than face recognition during patrolling because the drone needs to actuate its position and coordinates automatically rather than done in the ground station. Using Mask RCNN or faster Mask RCNN takes time for faster computation and thus recording the video makes an error. So In this paper, we solved this error by implementing a simple and easily understandable algorithm known as the Viola-Jones. A popular algorithm is still in use in most of daily life while taking a selfie, tagging friends on Instagram, the auto face detecting on Facebook, and so on.Operating drones: as we are using drones for patrolling each and every operation can be carried out by the drone alone. Like, suppose we want to navigate to corners that our mind wishes, then map coordinates are not at all possible since it needs a network and little knowledge to understand for pinning the location. In this paper, we are using *the djiTellopy package* and *pygames* for flying the drone. DjiTellopy is a python package developed by MIT university in the year 2018 an open-source platform that helps students code drones operation and so on. Pygames is also a python package that helps the user to control the UAV by using arrow keys. Using this we can also perform person/face tracking, face recognition, headcount functions when required. Using the DjiTellopy package we can talk to our drone, stream video, temperature, barometer values, battery percent, clockwise and counterclockwise rotation, forward, backward, left, right, up, down, 180^o flip, and so on.As we have seen so many challenges and its associated solutions, further we would like to take our discussion by explaining the algorithm's work style, and implementations in the next section.

IV. UAV ALGORITHMS

An algorithm is a step-by-step procedure to solve a particular problem by means of logical understandings and methods. In our paper, we are using the Viola-Jones algorithm for face detection, face recognition for recognizing a person's face, and the Kalman filter for drone stabilization.

(a) Viola-Jones algorithm

The method proposed by Paul Viola and Micheal Jones was a significant step forward in the face detection fields. They used machine learning algorithms to select a set of simple features that they combined into an efficient scalable classifier. Their paper introduced three key innovations that enabled their detector to achieve performance boosts over previous systems. The first was the use of the “integral image” for faster feature computation. The second was the use of the AdaBoost machine learning algorithm as a means for quickly selecting simple and efficient classifiers. The third was a method for combining classifiers into a “cascade” to quickly eliminate background regions and focus computational attention on more promising areas of the image.

(a.1) Haar-like Features

The Viola-Jones face detection method was a combination of simple Haar-Like features to classify faces. Haar-like features to classify faces. Haar-like features are rectangular digital image features that get from their similarity to Haar-

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006227|

wavelets. Working with individual pixel intensities is computationally expensive, so these features were introduced by Papageorgiou et al. Providing a method for encoding image properties in a form that can be computed much more quickly[5]. Simple Haar-like features are composed of two adjacent rectangles, located at any scale and position within an image, and are referred to as "2-rectangle" features. The features are defined as the difference between the sums of image intensities within each rectangle. Viola and Jones extended this set by defining similar features composed of 3 and 4 rectangles. Additionally, Lienhart and Maydt defined tilted versions of these features to enrich the possible feature set in an attempt to form better classifiers [6]. These simple features are quite coarse when compared to alternatives such as steerable filters, however, their computational efficiency more than makes up for their limitations.

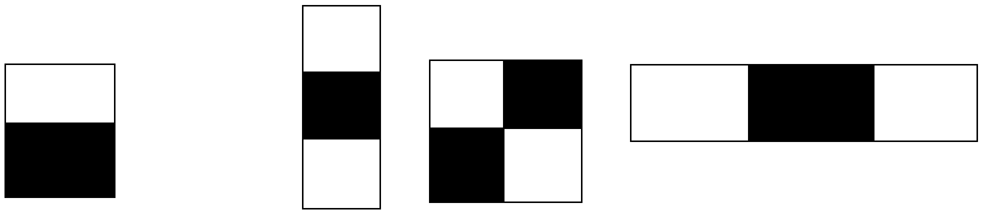


figure 1: Various types of Haar-like features

(a.2) Integral Image

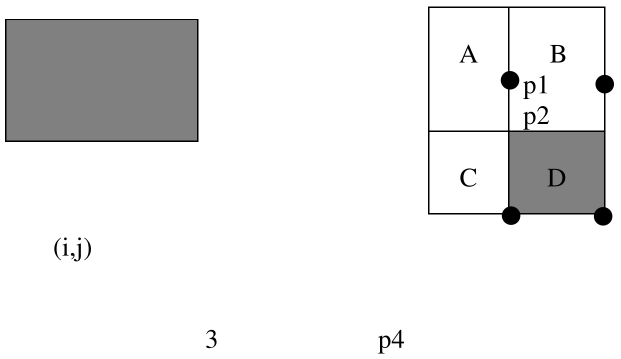
Haar-like features can be calculated extremely quickly by using an image representation called the integral image. The integral image is an application of summed-area tables; a concept introduced by Crow in 1984[4], usually used in computer graphics. The integral image can be defined as:

$ii(x, y)=(x + a)^n = \sum_{x' \leq x, y' \leq y} i(x', y')$ (1)

Where *ii(x, y)* is the integral image and *i(x, y)* is the original image intensity. The integral image can be calculated in a single pass using the following recurrences:

$$s(x , y) = s(x , y-1) + i(x , y)$$
$$ii(x , y)=ii(x-1 , y) + s(x , y) \quad (2)$$

Here *s(x , y)* is the cumulative row sum and we have the following base cases: *s(x , -1)=0* and *ii(-1 , y)=0.* Using the integral image, each features can be calculated in only four memory references.



ii(i,j)=sum of image intensities in shaded area

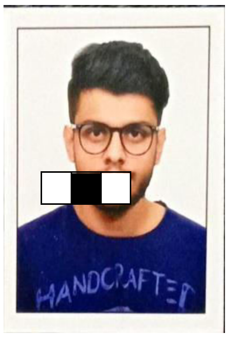
D=ii(p4)+ii(p1)-ii(p2)-ii(p3)

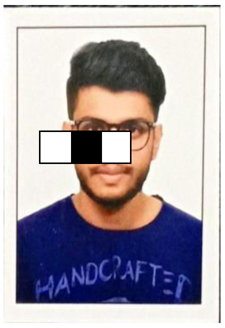
figure 2.Integral image and rectangle feature calculation.

Furthermore, because Haar-like features use rectangles that share corners, the number of tool memory references needed to compute a feature can be further reduced.

(a.3) Adaboost

As stated previously there can be a lot of features within a detector at 19x19 base resolutions that need to be calculated. After calculating all the features it is necessary to understand that only a few features will be important among all of them to detect a face.





Relevant Features **Irrelevant Features**

figure 3: Relevant and Irrelevant features of an image

After these features are discovered, a weighted combination of every one of these features is utilized in assessing and choosing whether any given window has a face or not. Every one of the chosen features is viewed as alright to be incorporated in the event that they can at least perform superior to anything arbitrary speculating. These features are called weak classifiers. Adaboost builds strong classifiers as a linear combination of these weak classifiers

$C(x) = a * C_1(x) + b*C_2(x) + c*C_3(x) + \ldots\ldots\ldots(1)$

Where:
C(x) -Strong Classifier
a , b , c -Coefficient of weak classifiers
C1(x),C2(x),C3(x)- Weak classifiers

(a.4) Cascading

The cascading classifier is used to determine whether a given sub-window classifier is definitely not a face or maybe a face.

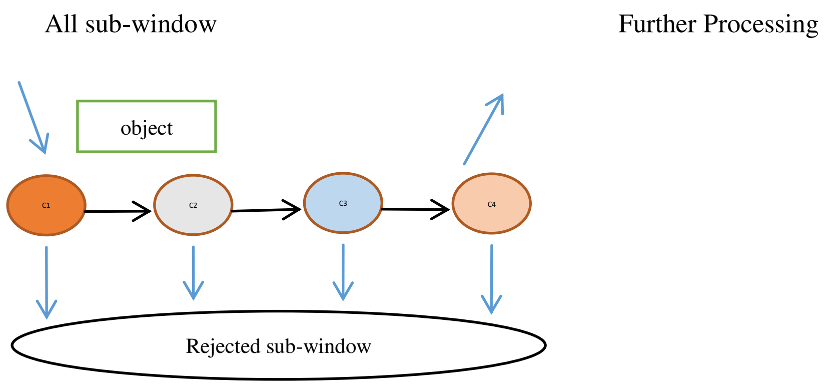


figure 4: Cascading

(b) Face recognition— Step by Step

Let's tackle this problem one step at a time. For each step, we'll learn about a different machine learning algorithm[7]. We are not going into detail about every single algorithm completely to keep this paper into a book, but we will learn how you can build your own facial recognition system in python using OpenFace and dlib.

Step 1: Finding all the Faces

The first step in our pipeline is *face detection*. Obviously, we need to locate the faces in a photograph before we can try to tell them apart!If you have used a camera in the last 10 years, you've probably seen face detection in action. Face detection is a great feature for cameras. When the camera can automatically pick out faces, it can make sure that all the faces are in focus before it takes a picture. But we'll use it for a different purpose— finding the areas of the image we want to pass on to the next step in our pipeline. Face detection went mainstream in the early 2000s when Paul Viola and Micheal Jones invented a way to detect faces that were fast enough to run on cheap cameras. However, much more reliable solutions exist now. We're going to use a method invented in 2005 called Histogram of Oriented Gradients— or just ***HOG*** for short. To find faces in an image, we'll start by making our image black and white because we don't need color data to find faces:

figure 5: black and white face image

Then we'll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it:

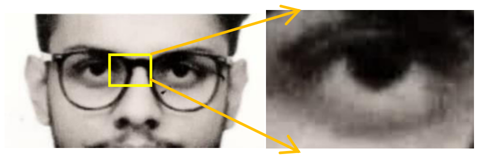


figure 6:showing every single pixel of the black and white eye image

Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:

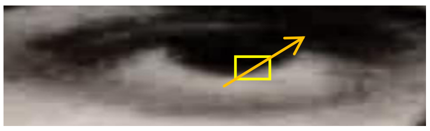


figure 7:Looking at just this one pixel and the pixels touching it, the image is getting darker towards the upper right.

If we repeat that process for every single pixel in an image, we end up with every pixel being replaced by an arrow. Those arrows are called *gradients* and they show the flow from light to dark across the entire image. This might seem like a random thing to do, but there's a really good reason for replacing the pixels with gradients. If we analyze pixels directly, really dark images and really light images of the same person will have totally different pixel values. But by only considering the *direction* that brightness changes, both really dark images, and really bright images will end up with the same exact representation. That makes the problem a lot easier to solve!. But saving the gradient for every single pixel gives us way too much detail. We end up missing the forest for the trees. It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image. To do this, we'll break up the image into a small square of 16x16 pixels each. In each square, we'll count up how many gradients point in each major direction. Then we'll replace that square in the image with the arrow directions that were the strongest. To find HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces. Using this technique, we can now easily find faces in any image.

Step 2: Posing and Projecting Faces

We now isolated the faces in our image. But how we have to deal with the problem that faces turned different directions look totally different to a computer. To account for this, we will try to wrap each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps. To do this, we are going to use an algorithm called face landmark estimation. There are lots of ways to do this, but we are going to use the approach invented in 2014 by Vahid Kazemi and Josephine Sullivan. The basic idea is we will come up with 68 specific points called *landmarks* that exist on every face— the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc[7]. Then we will train a machine-learning algorithm to be able to find these 68 specific points on any face. Now that we know where the eyes and mouth are, we'll simply rotate, scale and shear the image so that the eyes and mouth are centered as best as possible. We won't do any fancy 3d wrap because that would introduce distortions into the image. We are only going to use basic image transformations like rotation and scale that preserve parallel lines called affine transformation:

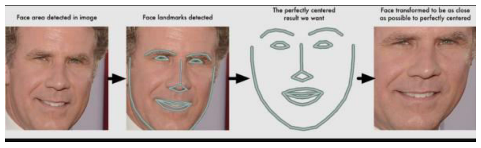

Face area detected in image
Face landmarks detected
The perfectly centered result we want
Face transformed to be as close as possible to perfectly centered

figure 8: affine transformation that preserves parallel lines

Now no matter how the faces are turned, we are able to center the eyes and mouth in roughly the same position in the image. This will make our next step a lot more accurate.

Step 2: Encoding Faces

Now we are to the meat of the problem— actually telling faces apart. This is where things are getting interesting! The simplest approach to face recognition is to directly compare the unknown face we found in step 2 with all the pictures we have of people that have already been tagged. When we find a previously tagged face that looks very similar to our unknown face, it must be the same person. Seems like a pretty good idea, right? There's actually a huge problem with that approach. A site like Facebook with billions of users and trillion photos can't possibly loop through every previous-tagged face to compare it to every newly uploaded picture. That would take way too long. They need to be able to recognize faces in milliseconds, not hours. What we need is a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, the spacing between the eyes, the length of the nose, etc.

The most reliable way to measure a face

Here, we are discussing what measurements should we collect from each face to build our known face database? Ear size? Nose length? Eye color? Something else? It turns out that the measurements that seem obvious to us humans (like eye color) don't really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure. The solution is to train a Deep Convolutional Neural Network. But instead of training the network to recognize pictures objects, we are going to train it to generate 128 measurements for each face. The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person.
2. Load another picture of the same known person.
3. Load a picture of a totally different person.

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart:

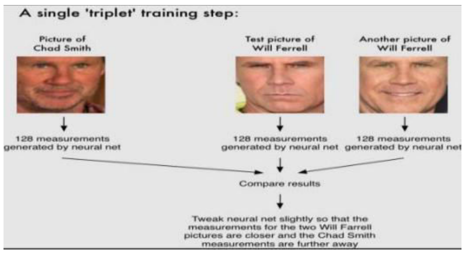

A single 'triplet' training step:
Picture of Chad Smith
Test picture of Will Ferrell
Another picture of Will Ferrell
128 measurements generated by neural net
128 measurements generated by neural net
128 measurements generated by neural net
Compare results
Tweak neural net slightly so that the measurements for the two Will Farrell pictures are closer and the Chad Smith measurements are further away

figure 9: triplet training step

After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements. Machine learning people call the 128 measurements of each face an embedding. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning. The exact approach for faces we are using was invented in 2015 by researchers at Google but many similar approaches exist.

Encoding our face image

This process of training a convolutional neural network to output face embeddings requires a lot of data and computer power. Even with an expensive NVidia Tesla video card, it takes about 24 hours of continuous training to get good accuracy. But once the network has been trained, it can generate measurements for any face, even ones it has never seen before! So this step only needs to be done once. Lucky for us, the fine folks at OpenFace already did this and they published several trained networks which we can directly use. So all we need to do ourselves is run our face image through their pre-trained network to get 128 measurements for each face. Here are the measurements for our test image:

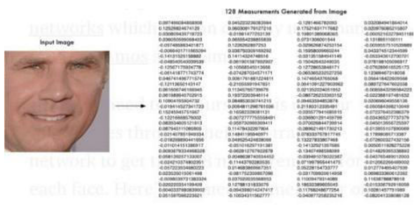


figure 10: test image measurements

So what parts of the face are these 28 numbers measuring exactly? It turns out that we have no idea. It doesn't really matter to us. All that we care about is that the network generates nearly the same numbers when looking at two different pictures of the same person.

Step 4: Finding the person’s name from the encoding This last step is actually the easiest step in the whole process. All we have to do is find the person in our database of known people who has the closest measurements to our test image. You can do that by using any basic machine learning classification algorithm. No fancy deep learning tricks are needed. We’ll use a simple linear SVM classifier, but lots of classification algorithms could work. All we need to do is train a classifier that can take in the measurements from a new test image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name of the person!.

(c) Kalman Filter (KF) The KF is not exactly a filter, it is not an electronic device that filters out unwanted noise. The KF is a mathematical estimator. It estimates the actual data that comes in and outputs the best estimate to the PID controller. Therefore, the KF does not filter a signal, but it does filter the system. The KF works by calculating the best output from the measured data with predicted data through a feedback system. We will be using the KF in our drone to accurately estimate its position and velocity[8]. The way it estimates the data is by considering the different noises introduced by each sensor. This means that both the measured and predicted data come with their own distinct noise. Figure 11 shows the noise model of both prediction and measured data.

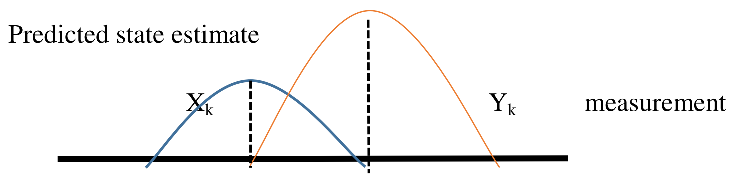


figure 11: noise models predicted and measured data calculated by the KF

Will the KF learn more about the prediction data, the measured data, or a combination of both? It depends on a series of equations within the feedback system that through a process of prediction and updates it can determine the best output. The KF will estimate the final value somewhere in between the two \ noise models by settling on the average of the two. This average depends on the shape of both noise models. This is what the KF does. Due to the noise generated by the sensors we need a KF to be implemented in our drone project. Each sensor will give a different measurement of the drone's position and velocity, so the KF must consider all these measurements. The KF calculates the noise that comes with these measurements, and from its calculations, it finds an optimal estimated value that represents the most accurate measurement. Figure 12 depicts the optimal estimated state computed by finding an average between the two noise models.

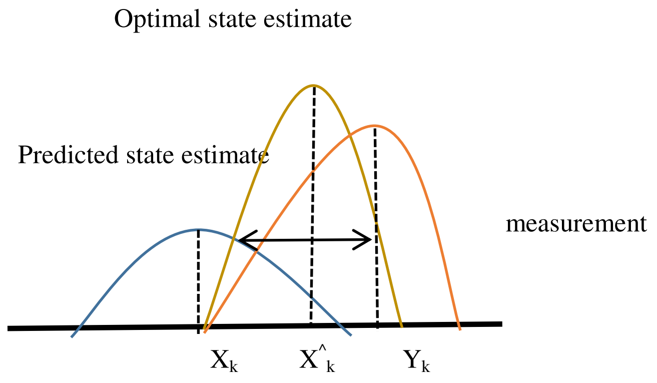


figure 12: optimal estimated state

For example, the accelerometer says the drone is 15 degrees from the vertical, while the gyroscope says the drone has not rotated away from the vertical. Without the KF, this would confuse the drone and through it off balance, but with the KF the system will find a happy medium between all the data given by the sensors.

(c.1) Extended Kalman Filter

An improvement over the KF is the EKF. The problem with the KF is that it only works for linear systems. However, the real world is nonlinear. For example, gusts of wind pushing against the drone. The EKF considers the non-linear system to calculate the best estimate between the noise models. The idea behind the EKF is basically the same as the KF, but considering non-linear systems derivatives is applied to the KF's mathematical model. Other than that, the EKF is the same as the KF. The only downfall with this implementation is that it increases the computational complexity by a little. Still, due to the advantage of solving non-linear systems is it preferable to make the switch. However, the EKF is not the only other upgrade that can be done to the KF. There is also the Unscented Kalman Filter(UKF) and the Particle Filter (PF). However, these other algorithms increase the computational complexity even more, which is unnecessary for what we need out of the drone. Therefore, the EKF will be the algorithm that will be used to help stabilize the AR 2.0 Drone.

V. LITERATURE SURVEY

Neural networks are one of the prominent machine learning algorithms. These NN and deep learning have demonstrated that they can exceed other algorithms when comes to accuracy and speed been at the same time were colossal amount of data has to be processed by them. Most of the research paper for object detection uses Mask RCNN[9]. The advantage of using this approach is the speed and accuracy in object detection and image/video segmentation by utilizing Mask rcnn neural network algorithm.In the case of person tracking one of the journals has used YOLOv3[10] for person detection, locality-constrained Linear coding(LLC) method to match the specified target person, and a Multi-task cascaded convolutional neural network for face detection. The advantage of using this is its fast speed and high accuracy and is very robust to close targets or small targets.However, using these methods have its own drawbacks when implemented in a UAV. Mask RCNN requires more computational power and an appropriate/sufficient amount of training samples to obtain accurate results. Aerial images or video that was taken, where the object is obscure due to the environmental conditions pose a great challenge to precisely detect and classify an object in an image. It is impossible to track the person at night or in dark places. A Colossal amount of data has to be processed in case of object detection.

VI. HARDWARE PLATFORM

We are using UAV nano drones. As mentioned earlier it is of low-cost hardware to test our offline-based recognition approach. The drone costs about 9,000 INR, is small and lightweight(98mm*92.5mm*41mm. Propeller: 3 inches, Built-In Functions: Range Finder, Barometer, LED, Vision System, WiFi 802.11n 2.4G, 720p live view Port: Micro USB Charging port)[3]. And can be operated indoors and outdoors. The nano drone is fitted with a single camera, an IMU, a 3-axis magnetometer, and pressure and IR-based altitude detector. The front-facing camera has a resolution of 5mp and it can take a video of resolution 1280x720 at 30 fps.

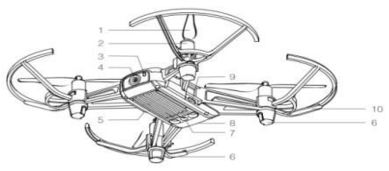


figure 13: Nano UAV architecture

VII. DEVELOPMENT OF MODEL

In this paper, we are making use of some of the libraries such as DjiTellopy, face_recognition, pygames, pyqt5 after that MySQL, dlib, visual studio C++ have to be installed which are vital for backend processing. So face detection model we have developed for patrolling in a more efficient way and is executed on the different operating systems.

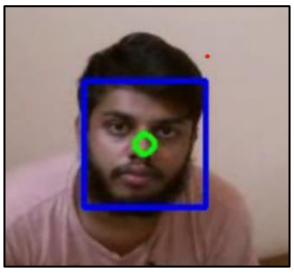


figure 14: sample result of Viola-Jones algorithm taken from our drone

1. System architecture:

We would like to further put our paper explaining the system architecture of our project work. At first, our drone includes the following components:1. Propeller, 2.Motors, 3.Aircraft status indicator, 4.Camera, 5. PowerButton ,6.Antennas, 7.Vision Positioning system, 8.Flight Battery ,9.Micro USB Port, 10.Propeller guards. Without this our hardware body is only grown for the face of it we have deployed the following:

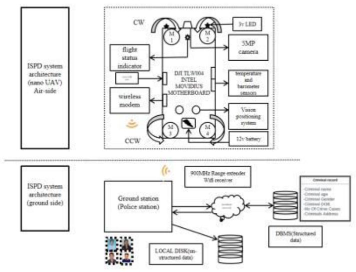

ISPD system architecture (nano UAV) Air-side
CW
3v LED
M 1
M 2
5MP camera
flight status indicator
DJI TLW004 INTEL MOVIDIUS MOTHERBOARD
temperature and barometer sensors
wireless modem
Vision positioning system
M 3
M 4
CCW
12v battery
ISPD system architecture (ground side)
900MHz Range extender Wifi receiver
Ground station (Police station)
DBMS(Structured data)
LOCAL DISK(un-structured data)

figure 15: Intelligence Surveillance Patrolling Drone System architecture

In this architecture, it is clear that the drone is controlled by a 900MHz range extender which is used to take the drone to a bit longer distance. The ground station is only having access to a local database server to fetch the data of a person who is identified from the drone camera and if the person's face is found then from the local disk which is having the unstructured data the image is popped in the screen from which the user can view. From the DBMS which is having structured data the name, age, etc of that person's detail is fetched.

2. Data-Flow Diagram

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system)[11]. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

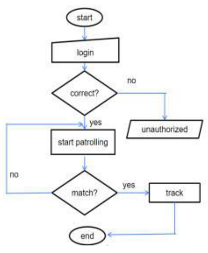


figure 16: Intelligence surveillance patrolling drone data flow diagram

At first, we start our project by log-in into the application from which we enter either through ID or password or by using face lock. If it is not accessed then it is unauthorized. If the access is not unauthorized. After which we start

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006227|

patrolling through our drone then if the criminal face is found or matched from the database then we track and end our patrolling else, till the amount of battery left we go back to the start patrolling process. Further, in detail, it is explained in the abstract diagram.

3. Abstract diagram

At first, we start our drone. When the drone is started live recording is done from the 5mp camera. As the flight goes on and the patrolling unit suspects something unusual from a person who might be a criminal, we create a bounding box around him and our drone tracks him from behind and sends the video feedback through the 900MHz wi-fi receiver as shown in the figure:

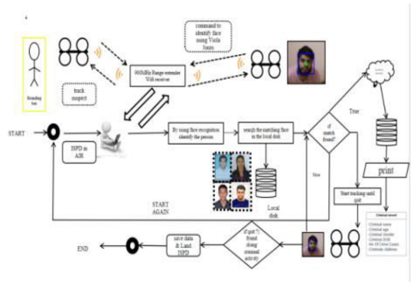

command to identify face using Viola Jones
track suspect
900MHz Range extender Wifi receiver
START
ISPD in AIR
By using face recognition identify the person
search the matching face in the local disk
if match found?
True
False
print
Start tracking until quit
START AGAIN
Local disk
if quit ?/ found doing criminal activity
save data & Land ISPD
END

figure 17: Intelligence Surveillance patrolling drone abstract diagram

Then in the ground station, we can view the live video footage from the drone thanks to the deployment of UDP protocol. Then we send the command to the drone to take the frontal picture of that person to view his face and fix the target. Once that is done we activate a command to search for the face from our system local disk. If the face is found the image of that person is popped up on the screen and his details like name, age, no of crimes are also shown from the Mysql server(structured data). Further, by making use of Viola-Jones and KF algorithm we can make use of the drone to track the criminal stabilizing the movement until we quit and save the data. If the criminal is not found but found doing a criminal activity then we record it for law enforcement proof. Suppose if he is pure then we'll destroy the target and continue the patrolling. Sending and receiving the commands from the wi-fi receiver are given in detail in the below sequence diagram.

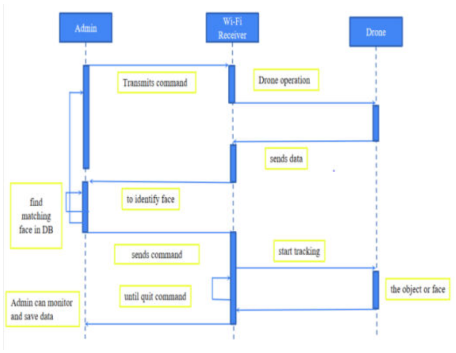


figure18: Intelligence Surveillance Patrolling drone Sequence diagram

VIII. CONCLUSION AND FUTURE WORK

Face detection, face recognition, and drone stabilization are one of the common problems during patrolling. For night vision we have soldered two LEDs in front of the camera. Obstacle avoidance, search and rescue purpose, and mapping of unknown territories. The ISPD is also capable of flying around the area during both day and night time. The drone is controlled from laptop arrow keys that allow officers to easily control the drone. ISPD helps the officers to adopt technology in patrolling and investigation that benefits the officer in quicker identification of the criminal and handling the situation carefully. It also assists the officers in monitoring the movements of the criminal, so the proper amount of

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006227|

patrolling units can be used. In the future ISPD can be trained with another machine learning model that will take decisions on its own for investigation and navigation. It can also be equipped with 360^0 view gimble camera for a better view. Our paper concludes that this technology can effectively motivate and promotes various places of work.

REFERENCES

[1] Kai Yan , Linfei Ma*, Ninth International Information Technology and Artificial Intelligence Conference [2020].
[2] Dante Tezza and Marvin Andujar ,University of South Florida [2019].
[3] Murtaza's Workshop-Robotics and AI Medium.com [2019].
[4] L. Zhi-fang, Y. Zhi-sheng, A.K.Jain and W. Yun-qiong, The Fifth International Conference on Computational Intelligence and Multimedia Application [2003].
[5]Prof.Dr. AbdulkadirErden, "Development of A Face Recognition System",[2011].
[6] Zhang K, Zhang Z, 18th IEEE internaltional Conference on Communication Technology [2016].
[7] Lwin, H.H., Khaing, A.S. and Tun,International Journal of Scientific & Technology Research,[2016].
[8] Yambor, W.S., Draper, B.A. and Beveridge, Empirical evaluation methods in computer vision [2002].
[9] Zhao, W., Krishnaswamy, A., Chellappa, R., Swets, D.L. and Weng, Discriminant analysis of principal components for face recognition.[1998]
[10] Borade, S.N., Deshmukh, R.R. and Ramu, S., Control and Automation (MED),24th Mediterranean Conference on IEEE. [2016]
[11] Priyadarsini, M.J.P., Murugesan, K., Inbathini, S,Research Journal of Applied Sciences, Engineering and Technology, [2015].