CHAPTER 1

INTRODUCTION

Unmanned surveillance on borders is now a bleeding edge technology constantly being researched and developed. Infiltration can be avoided by detection of humans or animals near the border, their identification as military/ non-military/armed/unarmed can be used to alert the border forces instantly with exact location of infiltrant by motion tracking. The system has a software generated alarm that sets off on detection of unidentified human. A relay output can be used to integrate a relay gun in case of emergency. This will not only increase the patrolling and defence efficiency but will also reduce the potential loss of human lives.

Border surveillance using Artificial Neural Networks makes use of machine learning algorithms to train the surveillance system to identify potential threats in different environments like border security or workplace security. This can also be extended to other applications like anti-theft parking, traffic monitoring, etc. The system is trained over a huge dataset to identify humans and human faces. The human recognition is used to detect border infiltration by intrusion into the no-man's land. The system output, if above a set threshold, is given to a relay gun which can start firing. In case of military base surveillance, this system is trained to identify faces of the registered employees and trained with random facial entries. This system will overcome the inefficiency of RFID tags and thus make the restricted areas more secure from unauthorized access. The current systems use either manual monitoring, if any, over the CCTV and some use sensors for alarming. But considering the fact that PIR sensors can alert even for animals or other harmless presence and unnecessarily increase redundancy of data, the idea of using artificial intelligence for surveillance has been put forward.

The existing system features are that it has a generic approach towards detection of unexpected motions in surveillance videos using video processing neural networks on non-real time basis. The approach used here is based on motion vector, or rather velocity of detected objects, rather than their nature and actual threat posed in the set environment. It can analyze the behaviour of people in the scene and send out a warning signal when dangerous behaviours occur. This has drawback that it considers only accuracy as its parameter, leading to inefficiency due to rigidity of classification algorithm. While there are some real-time NN based solutions designed for surveillance, the technology proposed by them needs a large setup and heavy

installation cost. Furthermore, most of these solutions use only user login and authentication which is individually useless in applications like border surveillance.

There are limitations that can be overcome with the use of Artificial Neural Networks. The model developed in this project makes use of Faster R-CNN, which works on feature extraction and object detection based on the trained and tested data set on a real time basis. We have trained the border surveillance system for detection of humans, while we add face recognition package to the existing system for military base authorized access. Thus the generic system can be easily modified for different environments. This model does not rely completely on accuracy metric. It is designed to meet optimum levels of accuracy, precision and consistency.

The initiation of the proposed system is easy and can be interfaced with camera or other hardware like a relay, buzzer, etc. as per requirement. Thus this system which is designed for human intrusion detection and face recognition has a generic structure that is easily expandable. Furthermore, the camera module used is a standard definition long range webcam with night vision. Thus range of camera is high and hardware requirement of installations reduces. The use of embedded software makes this system economic and secure. In the No-Man's land, amount of data in a frame is low, thus the system can work well with low computing power and high speed GPU.

BACKGROUND OF THE PROBLEM

In today's world security has become a very important aspect which cannot be compromised. Detection of moving object in an image sequences is crucial issue of a moving video. A very common difficulty in the detection of moving object is the presence of ego motion. We solve it by computing the dominant motion. Visual surveillance is a very active research area in computer vision. The use of this concept in surveillance is for security and to fight against terrorism and crime.

The main task includes detection of motion, object classification, tracking activity. The detection of moving objects in video streams is the first relevant step of information extraction in many computer vision applications. Many organizations and institutions need to secure their facilities thus need to use security and surveillance systems that are equipped with latest technology.

Intelligent video sensors were developed to support security systems to detect unexpected movement without human intervention. The important information of movement, location, speed and any desired information of target from the captured frames can be taken from the camera and can be transferred to the analysis part of the system.

Movement detection is one of these intelligent systems to which detect and tracks moving targets. An image recognition algorithm takes an image as input and outputs what the image contains. In other words, the output is a class label (e.g. "cat", "dog", "table" etc.). How does an image recognition algorithm know the contents of an image? Well, we have to train the algorithm to learn the differences between different classes.

If we want to detect cats in images, we train an image recognition algorithm with images of cats and images of backgrounds that do not contain cats. This algorithm can only understand objects / classes it has learned. We will focus only on two-class (binary) classifier that many popular object detectors (e.g. face detector and pedestrian detector) use. E.g. inside a face detector is an image classifier that says whether a patch of an image is a face or background.

PROBLEM STATEMENT

Detecting objects from a single frame is achievable but the use of it in real life scenarios is not enough. Continuous recording of frames digitally i.e. videography gives rise to a new era of surveillance system. But only capturing of frames is not satisfactory now. We are proposing a smarter surveillance system which detect the presence of a human being in a restricted area. In addition to detection we also look to make the system smarter and efficient. This will be achieved by adding various other detection parameters such as motion detection and face detection.

CHAPTER 2

Artificial Intelligence

Artificial intelligence is intelligence demonstrated by machines, in contrast to the natural intelligence (NI) displayed by humans and other animals. In computer science AI research is defined as the study of "intelligent agents". Any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving".

The scope of AI is disputed as machines become increasingly capable, tasks considered as requiring "intelligence" are often removed from the definition, a phenomenon known as the AI effect, leading to the quip, "AI is whatever hasn't been done yet". Capabilities generally classified as AI as of 2017 include successfully understanding human speech, competing at the highest level in strategic game systems (such as chess and Go), autonomous cars, intelligent routing in content delivery network and simulations. Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding.

AI research has been divided into subfields that often fail to communicate with each other. These sub-fields are based on technical considerations, such as particular goals (e.g. "robotics" or "machine learning"), the use of particular tools ("logic" or "neural networks"), or deep philosophical differences. Subfields have also been based on social factors (particular institutions or the work of particular researchers).

The traditional problems (or goals) of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is among the field's long-term goals. Approaches include statistical methods, computational intelligence, and traditional symbolic AI. Many tools are used in AI, including versions of search and mathematical optimization, neural networks and methods based on statistics, probability and economics.

Machine Learning

Machine learning is the branch of computer science that has to do with building algorithms that are guided by data. Rather than relying on human programmers to provide explicit instructions, machine learning algorithms use training sets of real-world data to infer models that are more accurate and sophisticated than humans could devise on their own.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide**.** The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Some machine learning methods

Machine learning algorithms are often categorized as supervised or unsupervised.

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- In contrast, **unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabelled data.

- **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

Within the field of machine learning, **neural networks** are a subset of algorithms built around a model of artificial neurons spread across three or more layers. There are plenty of other machine learning techniques that don't rely on neural networks.

Within neural networks, **deep learning** generally used to describe particularly complex networks with many more layers than normal. The advantage of these added layers is that the networks are able to develop much greater levels of abstraction, which is necessary for certain complex tasks, like image recognition and automatic translation.

Neural Network

Neural networks are at the cutting edge of machine learning and artificial intelligence. Implementing them requires expertise in statistical analysis, distributed systems, big data processing, and related fields. Artificial neural networks (ANNs) are biologically inspired computer programs designed to simulate the way in which the human brain processes information. ANNs gather their knowledge by detecting the patterns and relationships in data and learn (or are trained) through experience, not from programming. An ANN is formed from hundreds of single units, artificial neurons or processing elements (PE), connected with coefficients (weights), which constitute the neural structure and are organised in layers. The power of neural computations comes from connecting neurons in a network.

Each PE has weighted inputs, transfer function and one output. The behaviour of a neural network is determined by the transfer functions of its neurons, by the learning rule, and by the architecture itself. The weights are the adjustable parameters and, in that sense, a neural network is a parameterized system. The weighed sum of the inputs constitutes the activation of the neuron. The activation signal is passed through transfer function to produce a single output of the neuron. Transfer function introduces non-linearity to the network.

During training, the inter-unit connections are optimized until the error in predictions is minimized and the network reaches the specified level of accuracy. Once the network is trained and tested it can be given new input information to predict the output. Many types of neural networks have been designed already and new ones are invented every week but all can be described by the transfer functions of their neurons, by the learning rule, and by the connection formula. ANN represents a promising modelling technique, especially for data sets having non-linear relationships which are frequently encountered in pharmaceutical processes.

With the rise of autonomous vehicles, smart video surveillance, facial detection and various people counting applications, fast and accurate object detection systems are rising in demand. These systems involve not only recognizing and classifying every object in an image but localizing each one by drawing the appropriate bounding box around it.

This makes object detection a significantly harder task than its traditional computer vision predecessor, image classification. Fortunately, however, the most successful approaches to object detection are currently extensions of image classification models. Neural networks are

well-suited to identifying non-linear patterns, as in patterns where there isn't a direct, one-to-one relationship between the input and the output. Instead, the networks identify patterns between combinations of inputs and a given output.

Many media reports describe artificial neural networks as working like the human brain, but this is a bit of an oversimplification. For one, the difference in scale is tremendous: While neural networks have increased in size, they still typically contain between a few thousand and a few million neurons, compared to the 85 billion or so neurons found in a normal human brain. The other main difference lies in how these neurons are connected. In the brain, neurons can be connected to many other neurons nearby. In a typical neural network, however, information only flows one way. These neurons are spread across three layers:

- The **input layer** consists of the neurons that do nothing more than receive the data and pass it on. The number of neurons in the input layer should be equal to the number of features in your data set.
- The **output layer** consists of a number of nodes depending on the type of model you're building. In a classification system, there will be one node for each type of label you might be applying, while in a regression system there will just be a single node that puts out a value.
- In between these two layers is where things get interesting. Here, we have what's called the **hidden layer**, which also consists of a number of neurons (how many will depend on the number of neurons in your input and output layers, but don't worry about that). The nodes in the hidden layer apply transformations to the inputs before passing them on. As the network is trained, those nodes that are found to be more predictive of the outcome are weighted more heavily.

Deep learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images. Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks. The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labelled data and neural network architectures that learn features directly from the data without the need for manual feature extraction. One of the most popular types of deep neural networks is known as convolutional neural networks (CNN or ConvNet). A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images. CNNs eliminate the need for manual feature extraction, so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not pretrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

Deep learning started its journey through increased depth of artificial neural networks, but metamorphosis will probably lead to its final destination.

Convolution Neural Network:

Convolutional neural networks are very similar to ordinary neural networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, in our case, information about data from the frames captured by the surveillance camera. They perform dot product of the captured vector quantities. These values are the stored in the database.

ConvNet or CNN architecture works explicitly on images, making it easier for us to feed and process image data and encode certain video processing and feature extraction properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network. It works in 3-dimensional space, analyzing the width, height and depth of the captured frame.

F-RCNN, just like normal CNN works in the following flow:

i) Convolutional Layer
ii) Max-Pooling Layer
iii) RELU
iv) Fully connected layer

Types of CNN are:

1) R-CNN
2) SPP-NET
3) F RCNN

- R-CNN:

R-CNN is the parent version of Fast R-CNN. In other words, R-CNN really kicked things off, R-CNN, or Region-based Convolutional **N**eural **N**etwork, consisted of 3 simple steps:

1. Scan the input image for possible objects using an algorithm called Selective Search, generating ~2000 region proposals
2. Run a convolutional neural net (CNN) on top of each of these region proposals
3. Take the output of each CNN and feed it into a) an SVM to classify the region and b) a linear regressor to tighten the bounding box of the object, if such an object exists.

These 3 steps are illustrated in the image below:

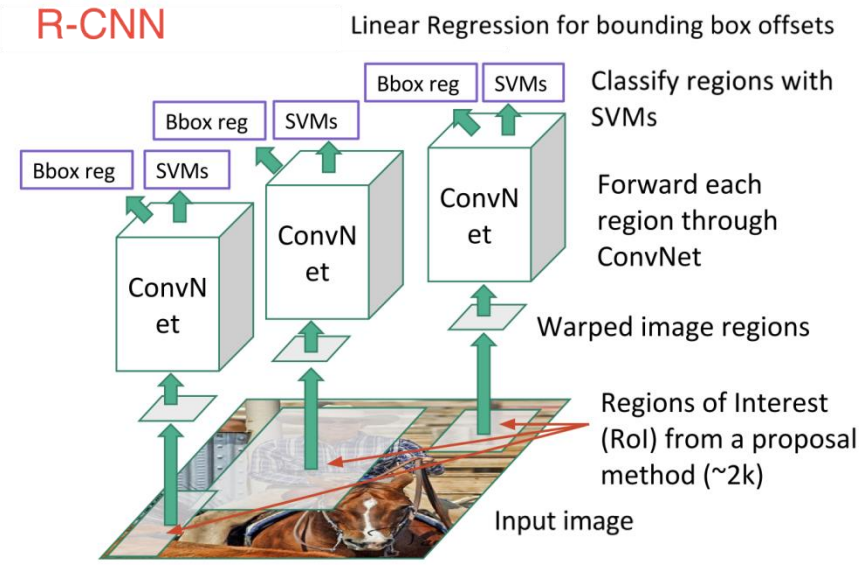


Fig1. Architecture of Region CNN

In other words, we first propose regions, then extract features, and then classify those regions based on their features. In essence, we have turned object detection into an image classification problem. R-CNN was very intuitive, but very slow.

- Fast R-CNN

Fast R-CNN is now a canonical model for deep learning-based object detection. It helped inspire many detection and segmentation models that came after it, including the two others we're going to examine today. Unfortunately, we can't really begin to understand Faster R-CNN without understanding its own predecessors, R-CNN and Fast R-CNN, so let's take a quick dive into its ancestry R-CNN's immediate descendant was Fast-R-CNN. Fast R-CNN resembled the original in many ways, but improved on its detection speed through two main augmentations:

1. Performing feature extraction over the image before proposing regions, thus only running one CNN over the entire image instead of 2000 CNN's over 2000 overlapping regions

2. Replacing the SVM with a SoftMax layer, thus extending the neural network for predictions instead of creating a new model

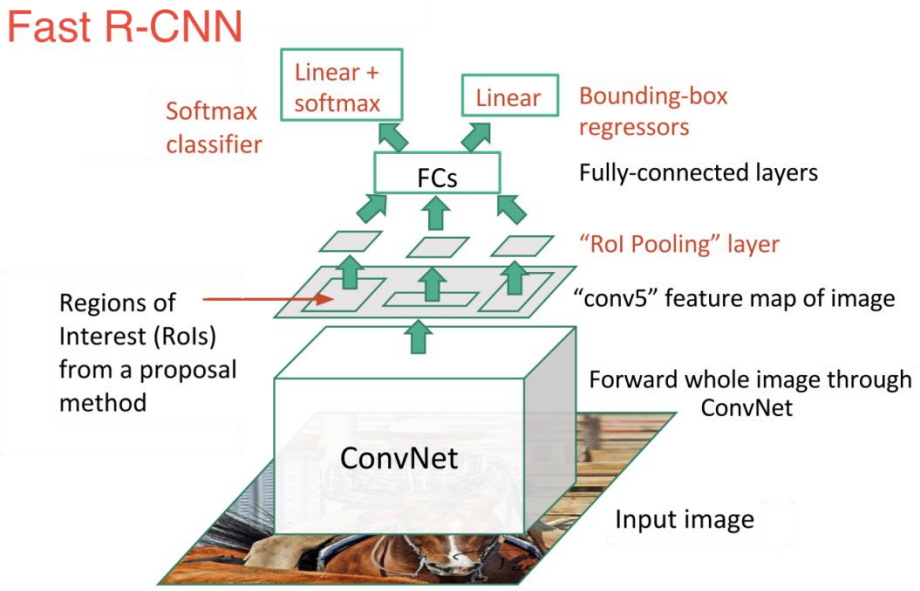


Fig2. Architecture of Fast Region CNN

As we can see from the image, we are now generating region proposals based on the last feature map of the network, not from the original image itself. As a result, we can train just **one** CNN for the entire image.

In addition, instead of training many different SVM's to classify each object class, there is a single softmax layer that outputs the class probabilities directly. Now we only have one neural

net to train, as opposed to one neural net and many SVM's.Fast R-CNN performed much better in terms of speed. There was just one big bottleneck remaining: the selective search algorithm for generating region proposals.

Thus,

Faster R-CNN, R-FCN, and SSD are three of the best and most widely used object detection models at present. Other popular models tend to be fairly similar to these three, all relying on deep CNNs (read: ResNet, Inception, etc.) to do the initial heavy lifting and largely following the same proposal/classification pipeline.

Thus, we have used this machine learning algorithm in our project. Smart surveillance system thus, has an edge over the normal surveillance system due to the fact that it reduces human intervention, thus being a literally zero sleep-time surveillance and detecting intrusion or unauthorised access in any environment.

We have trained this system for human intrusion detection in No-Man's land, which can replace manual surveillance and avoid potential loss of life. We have also developed an additional system for face recognition to prevent unauthorised access to people in restricted areas. Although, what we have created is a general prototype model, using proper acquisition devices, sensors and relays, a working system for any of these environments can be created.

IMAGE PROCESSING

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools;
- Analysing and manipulating the image;
- Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

In this lecture we will talk about a few fundamental definitions such as image, digital image, and digital image processing. Different sources of digital images will be discussed and examples for each source will be provided. The continuum from image processing to computer vision will be covered in this lecture. Finally, we will talk about image acquisition and different types of image sensors.

The model used in development of our project has been built on F-RCNN. It consists of separate modules as explained below:

CHAPTER 3

WORKING PRINCIPLE

- Video surveillance and data acquisition
    1) A 10X zoom camera with high definition is used for 24x7 surveillance. The input from video camera is given to a computer in the control room on a real time basis.
- Real-time video processing and human detection
    1) Collection of dataset (25 common objects including humans)
    2) Defining categories of objects based on feature extraction
    3) Configuration of development model (F-RCNN SSD_MobileNet_COCO) that will be required while training, validating and giving random data input. In case of pre-trained models, it is necessary to customize the model by modification of configuration util.
    4) Feeding, training and testing of dataset using the utils: train_net.py, test_net.py
    5) Another important aspect of training that is monitoring of classification performance is done by solver.prototxt and train.prototxt which describes the network structure, including number of layer, type of layer, number of neurons in each layer, etc.
    6) Since the dataset required in the case of this project is not vary large, but 8000 odd samples, a pre-trained model works and no need to develop a model from scratch.
    7) Although in case of real life implementation on borders, a dedicated model will be required to be built.
- Output of artificial neural network:
    1) The above model, after being trained and validated for detection of humans and human faces, is put to test via real-time input from the webcam.
    2) On detection of humans, the object detection model sets off a software generated alarm and activates the face detection module.
    3) The face detection model is trained in the same way as the object detection model. The face detection model has a database of all registered personnel on

the military base with permission of patrolling in the border area under surveillance.

4) The detected human's face is compared to the database entries of registered faces. If unmatched, a breach or infiltration is confirmed and the output of face recognition algorithm can be given to a relay gun that starts firing, which can be controlled manually as well.

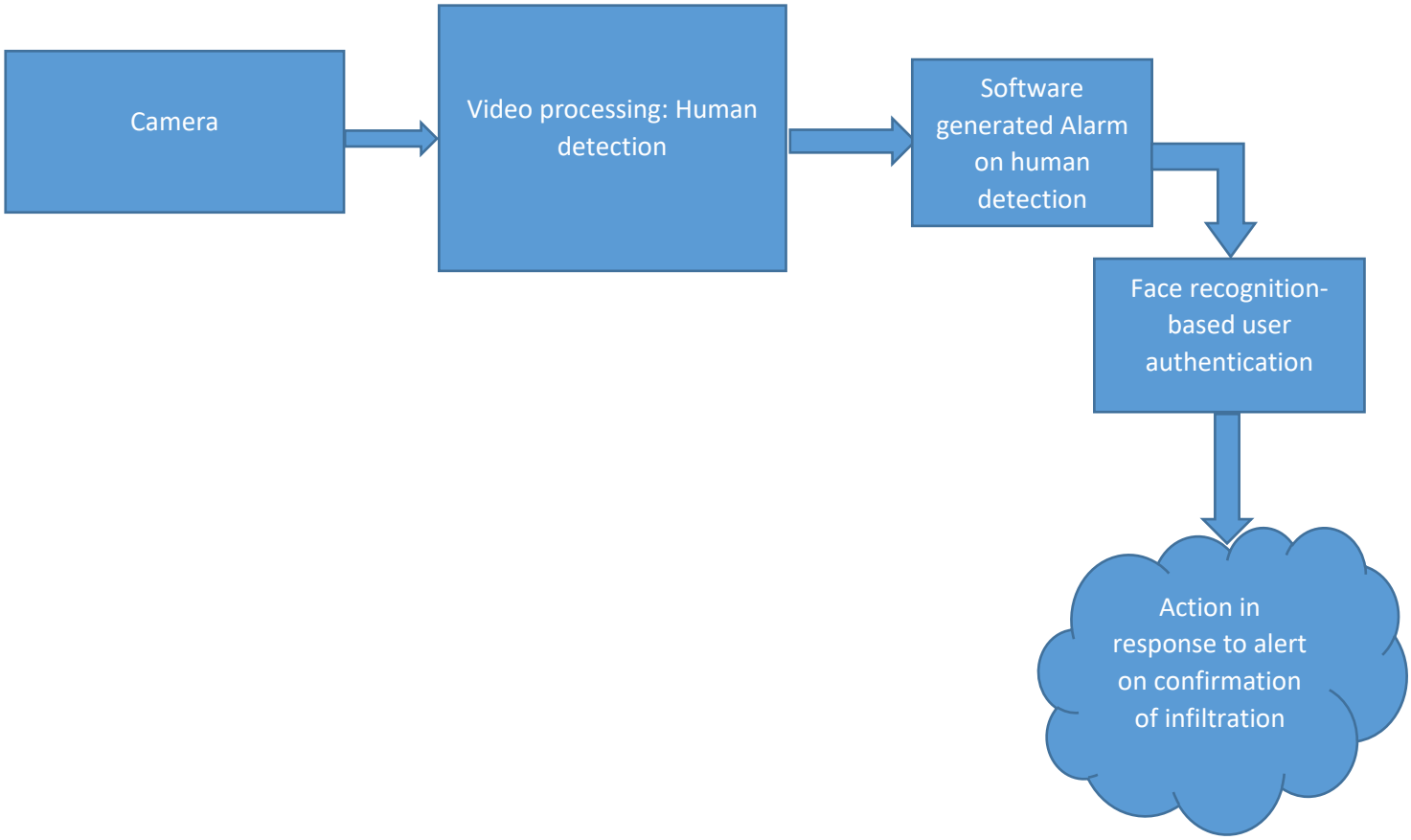
- There are various type of problems and challenges when it comes to object detection which we have solved in this project:

  Different number of objects: In a continuous picture there comes a lot of objects, many classes and classification. While training a machine learning model, we need to represent data into a fixed-sized vector. As the number of objects which in this case is humans, in the image is not known prior, we cannot detect the correct number of outputs. Due to this, some post-processing is required, which adds complexity to a model.

  Sizing: While doing simple classification, we expect and want to classify objects that cover most of the image. On the other hand, some of the objects you may want to find could be a small as a dozen pixels (or a small percentage of the original image). Traditionally this has been solved with using sliding windows of different sizes, which is simple but very inefficient.

  Modelling: Next challenge is to solve two problems at the same time, which is; location and classification into, ideally, a single model. This is where the approach of deep learning comes into the picture

BLOCK DIAGRAM



The above shown is the idle block diagram of the project. This block diagram displays the ideal working this device. It consists of as the following:

Camera

A camera is used for capturing images and videos. The area which needs surveillance is covered with eyes of this camera. There could a variety of camera completely depending on what the application calls for. Eg. For detecting in the night, a thermal camera can be used which captures the thermal energy radiated by the target if in the boundary of vision. These images are continuously sent to the video processor.

Video Processing and Face Recognition

The images are continuously received from the camera which is then processed through many trained networks such as the Convolution Neural Networks which consist of many layers. Features are extracted from the image with the help of various Image Processing operations. These features are collected and are then put as the input to the trained neural network. The new features collected are compared with the database which is already saved at the memory. If the features are matched then the target is highlighted if not it is ignored.

The final call to decide whether the object needs to be highlighted depends on the level of confidence which is calculated based upon the number of new features being matched with the old or the database. Once threshold features are matched the confidence value increases and accordingly the highlight is displayed.

Similarly, in face detection, different feature of the face, especially the value of the pixel is compared with the database. Once majority of the feature or values are matched it shows up the identified person displaying its name. The database of the same is created before the detection operation.

Alarm

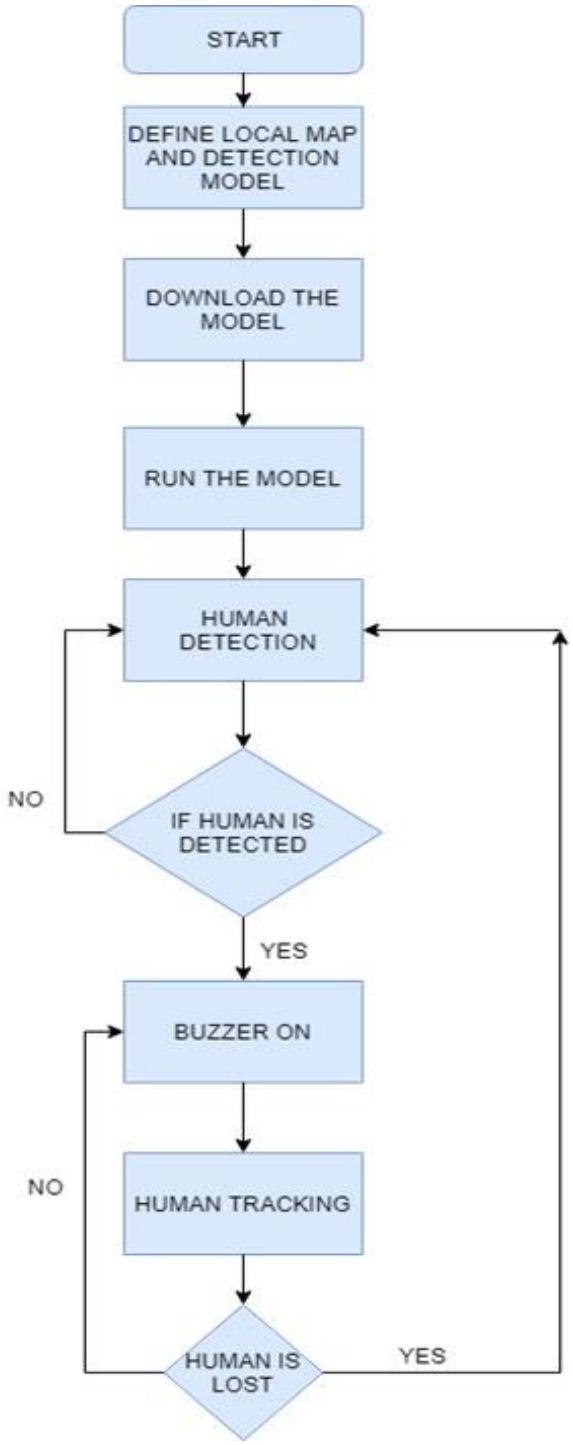
The purpose of the project is to develop a device which is smart enough to detect a human or infiltration on its own. Hence once the object is highlighted, a software alarm is called which starts to make noise and alerts the user about the infiltration making the user more alert and sharp about the surveillance. This also increases the efficiency and improves security on a higher scale.

Action in respond

To make the system more dependable and to reduce human error relays are connected which will provide actions even when the humans are absent. If a human fails to respond to any alert and warnings, the system will automatically provide actions and controls to it's machinery after the responding time for humans expire.

Actions such as, in military- activating mines, gun shots, lights, a complete lock down of the area, alerting higher command, etc. In offices- shutting the door, shutting the window, calling the police automatically, etc.

FLOW CHART (HUMAN DETECTION)



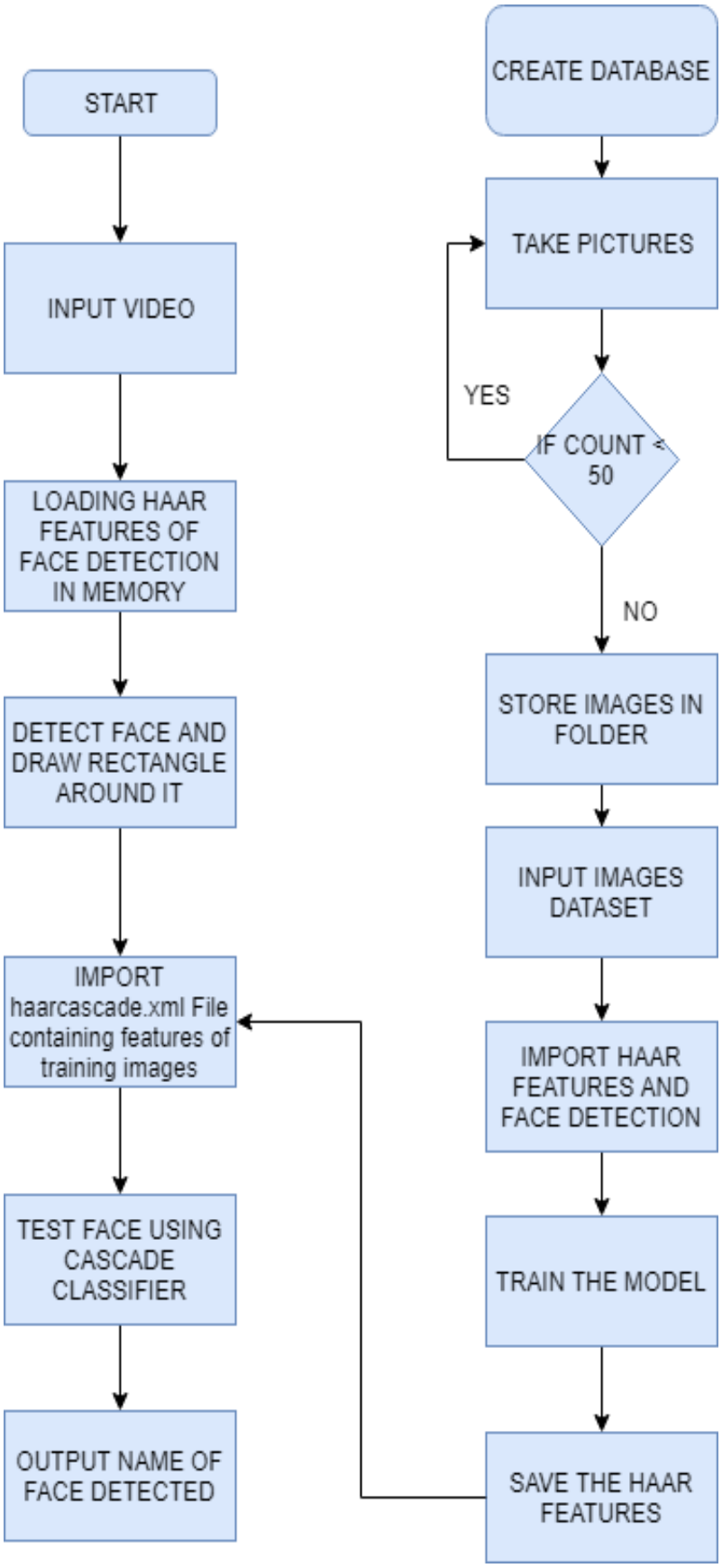
ALGORITHM (HUMAN DETECTION)

As shown in the flow chart input to our program is a video or a sequence of videos. For human detection we are going to use an "SSD with Mobilenet" model here.

The “frozen_inference_graph.pb” is the name of the actual model which we are going to use. While, mscoco-label map.pbtxt is list of strings which is used for adding labels for each box. This model was designed to detect various type of objects, but we have modified it to detect only the humans. This model can define 90 different categories which are different number of classes. But we are interested with the human detection only.First of all we have to download the model from the server without which we cannot work. Once we have downloaded the model then we have to load the actual tensorflow model which is “frozen_inference_graph.pb” and load it ion the memory.Then we will load the labels maps which are indices to category names, so that when our convolution network predicts `5`, we know that this corresponds to `airplane`. Here we use internal utility functions, but anything that returns a dictionary mapping integers to appropriate string labels would be fine.

Then we have to load the images into the numpy array and then reshape the images.
After doing this we can start the detection process. We have to take the input from the webcam and the input images are converted into numpy array and then their dimensions are expanded since the model expects images to have shape.After this, the detection of the human will take place the model will detect different humans in the image and each of these human will be represented by the boxes with different dimension which are depended on the object sizes.
As the program detects humans, a rectangle is drawn around it. In some cases, our program draws more than one rectangle for single person because of false positive cases, to minimize such false positive cases, in the next step our program uses non-maximum suppression to select a rectangle based on maximum overlapping criteria. Once the human is detected then the Software buzzer will be switched on and it will be on until the person will remain in the frame of the webcam. However, once the human is lost or gone out of frame then the buzzer will be off. Thus, the detection and tracking of the human will take place.

FLOW CHART (FACE DETECTION)



ALGORITHM (FACE DETECTION)

For face recognition we have to first create the database of the person so that if the person is present then the model can detect it. Thus, we have written two different codes. One for creating the database and the other for the face recognition. We were trying to integrate the face recognition code with the human detection one but were unable to do that so we are going to do it in different windows. We are going to run two different code files named as face_recog and other create_database for face detection purpose. The create_database helps us to create the database of the user and thus we can register them by running this file and it will collect 100 images of the users whom we want to register and then the feature extraction will take place with these images. We will train our model with these images and so if the model sees the person face again then it will be able to recognize him/her. But it's important that we have taken enough number of images of the user because then only our model will be able to train properly and we can get the face recognition with more accuracy.

These images need to be taken at different illumination scenarios so that the probability of errors will be reduced and the robustness of the system will be increased. For face recognition we have to create database by running create_database file. Thus, it will take the images of human for 100 times. We can also change the number of images taken by changing the count values in the code. After this the images will be transformed into gray scale and they are reshaped. The haarcascade.detectMultiScale is used to return the faces in the input image sand it's in the rectangle shape. . The haarcascade.detectMultiScale has scale factor which is used to create your scale pyramid. Despite a fixed size of face detection window, by rescaling the input image, you can resize a larger face towards a smaller one, making it detectable for the algorithm. Using a small step for resizing, for example 1.05 which means you reduce size by 5 %, you increase the chance of a matching size with the model for detection is found.

Then this face images are stored in the folder and then it will form the input images dataset. Then the images are used to train the model. Then the images are converted into features which will be used by the model to compare with the human face features which we want to detect and face detection will happen. Then we will run the main program of the face_recog where our model will do the face recognition. First face detection will take place. After the face detection phase, comes the phase of face recognition. But for this phase there is a prerequisite, to recognize a person firstly, we have to train our machine for certain images of that person.

Here training is done using Haar cascades. Thus, the training of the model will be performed now. The faces images which is stored in the folder database will be used to train the model. The images are converted into numpy array and the labels. Then the open CV will train the models from the images. After this, the video will be started and the model will start detecting the face so if it sees a face then it will transform the image and convert into gray scale and then the faces features will be converted by the haarcadcade.detectMultiScale. Now the featured will be compared and based on this the model will predict the person name by looking at his face. Here the cascade classifier is used for the detection of the faces. If the confidence score is more than threshold then the face label will be shown in the output. In this we have to import the haarcascade_frontalface_deafult. Xml file. This file will help us to convert the feature of the human faced so that we can convert it with the already stored features of the database. When the video starts at that time when the faces are in front of the camera then the faces will be detected and then the rectangle will be drawn around it using haar cascade.

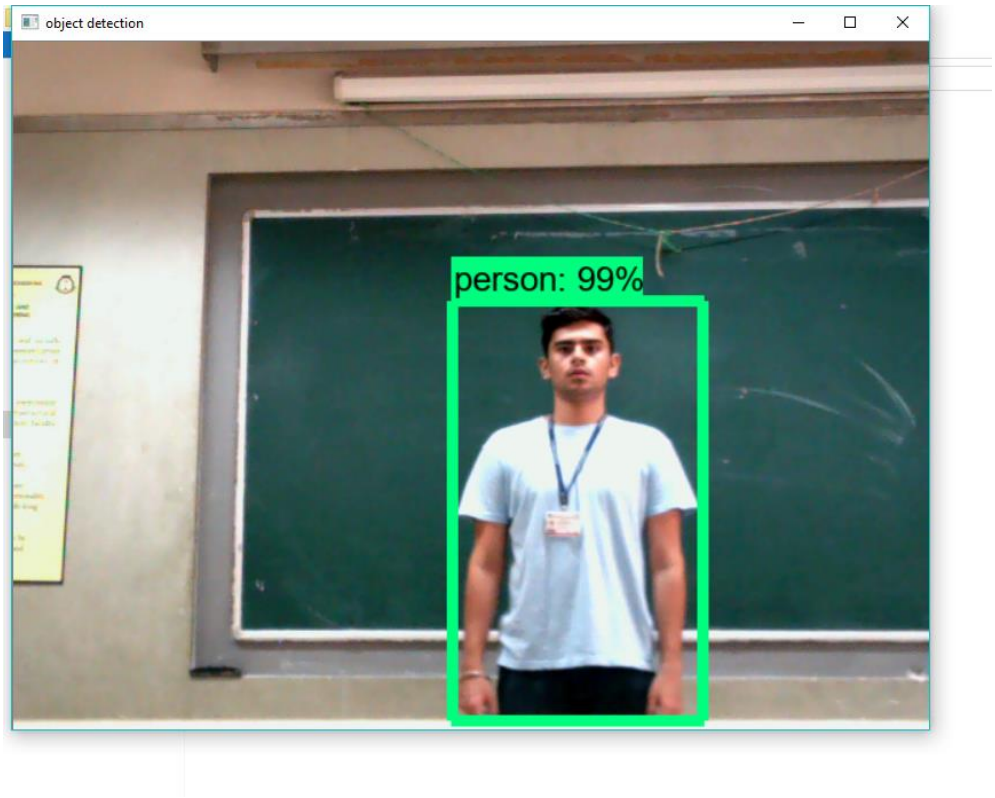
Now the feature extraction for the face will take place using haarcadcade and then these features will be compared with the featured of the trained image dataset. The comparison or classification of the face will be done by haar cascade classifier. Once faces are recognized the person is given the label of recognized face. So, each recognized person is assigned a label. The name or label of recognized person is given as output along with the confidence factor. Confidence factor is a parameter whose value is zero if the images in training set and and test case are same, higher the value of confidence factor lower the given image is similar to that person.

We have set threshold value of confidence factor, above which we can say the given image is not similar to the labeled person. Thus, our program detects human, detect faces, recognize faces and track human by recognizing this individual has passed through this camera.

OUTPUT

The generic artificial neural network created for human detection can identify 25 different objects, but our area of interest with respect to border application requires only human detection. For the same reason, the artificial neural network has been trained to identify humans in particular.

Once a human is detected, an alarm sets off and a 10X zoom-in face detection camera is activated. This camera runs for an artificial neural network specifically trained to distinguish unknown faces from the registered ones, suggesting infiltration.



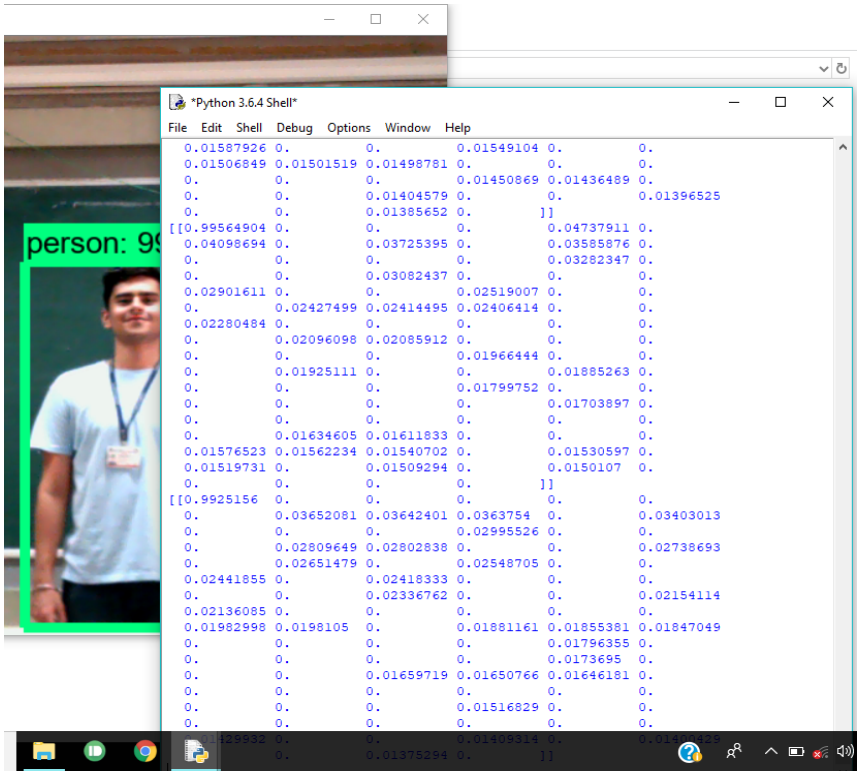
O/P1: DETECTING A HUMAN

The images shown here represent the output which we achieved by running the code over the frames provide by the camera. As shown above Human detection successfully has taken place.

The human detection model has been developed to classify humans and other objects based on their individual and relative confidence levels. Meaning thereby, if an object matches the features of two different classes, then the model reiterates its classification to find the perfect matching class using a function called: *max_num_classes*.

The label for every identified object, here human, includes its class name as well as the confidence level or essentially probability match. In order to avoid unnecessary hardware interfacing, a software generated alarm was used on human detection.

The command for this is *winsound.play ('alert'*), where 'alert.wav' was a siren sample file stored in our database. The values shown below displays the confidence value of the object being detected. If it is less than the threshold value then the detector avoids it. For detection to taken place the confidence value should be greater than the threshold value.



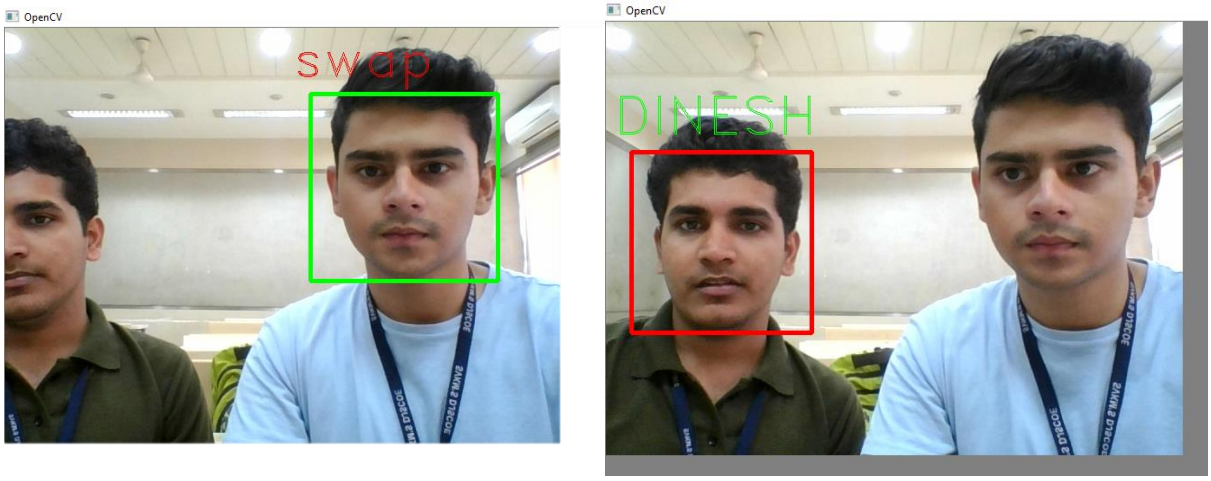
O/P2: THE CONFIDENCE VALUE BY THE MODEL ALSO THE NUMBER OF CLASS DETECTED

The achievement in this process was an unmanned surveillance and border patrolling

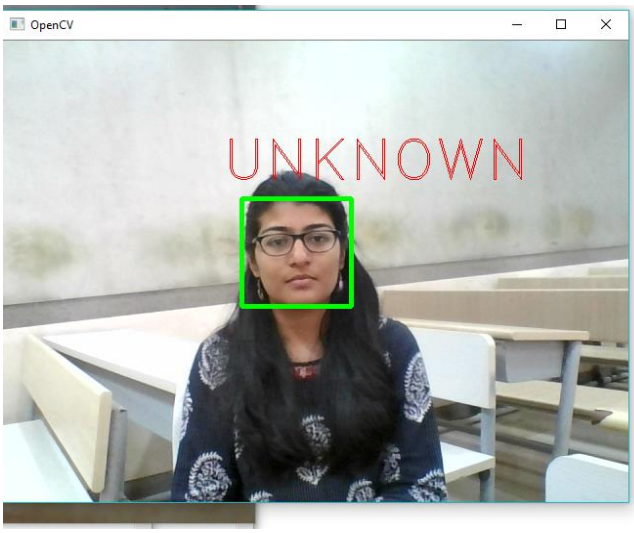
Below we have shown the screenshots of the Face Detection. In face detection we use Fisher face detector to be trained with hundreds of images which are called the data base. The database in made initially with another code which we have discussed about in the report above.

As shown below we can observe that the camera and the model is able to detect more than 1 faces and is able to differentiate them according to the database it has been trained with. The

out put of the same is achieved. Recognition of the face improves with increase in the number of images or database



O/P3: Detecting and Recognising Faces (FROM DATABASE)



O/P4: Detecting a face which is “Unknown” to the device (NOT FROM THE DATABASE)

CHAPTER 4

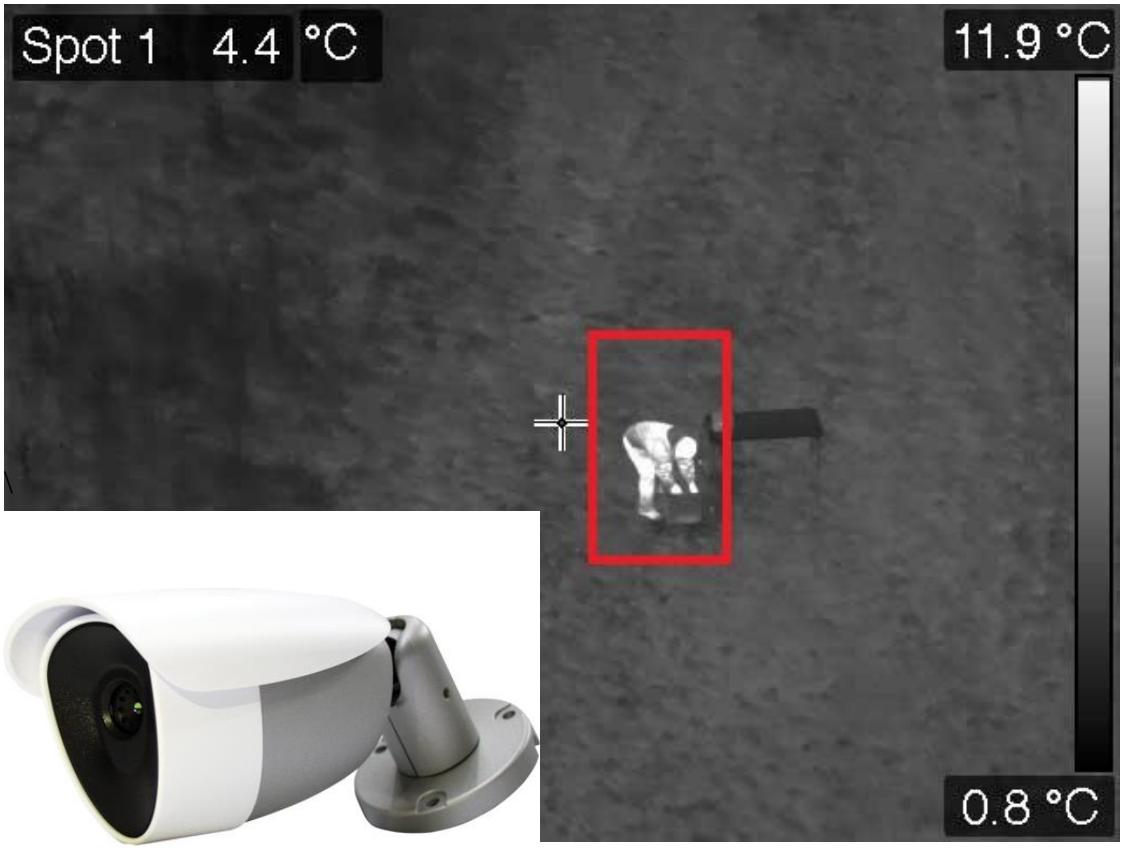
APPLICATION OF THE PROJECT

The main application of the project being border surveillance, the same system can be used as a smart surveillance system in various applications like:

1) Border Surveillance

Unmanned surveillance on borders is now a bleeding edge technology constantly being researched and developed. Infiltration can be avoided by detection of humans or animals near the border, their identification as military/ non-military/armed/unarmed can be used to alert the border forces instantly with exact location of infiltrant by motion tracking.

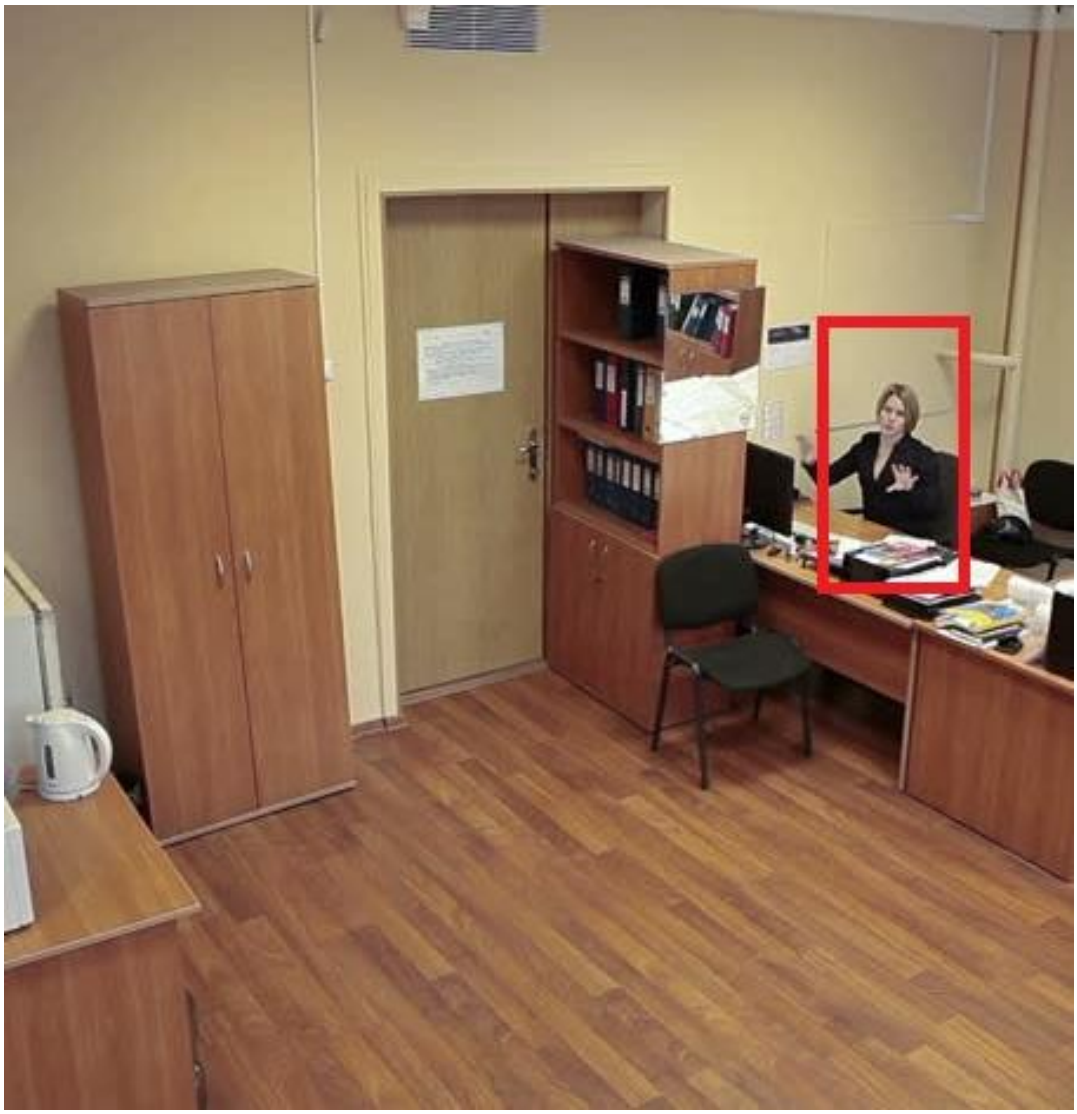
The relay output can be used to integrate a relay gun in case of emergency, in addition to immediate alert to the base station. This will not only increase the patrolling and defence efficiency but will also reduce the potential loss of human lives.



Img1. Border Infiltration

2) Office/Workplace surveillance

Offices and workplaces have restricted areas or vaults containing confidential documents. 24x7 surveillance of such sensitive areas and instant alert to authorized personnel can curb any theft or security breaches in office. In addition, as the woman safety is gaining increasing importance today, such a monitoring system can provide a secured environment to working women.



Img2. Human detected in office

3) Manufacturing unit surveillance

In a manufacturing facility like production line, there are some machines that work in extremely stringent conditions and any human intervention or entry of human in the area near such machinery can cause serious hazard. In such cases, a surveillance system with motion sensor can be used to shut down the machine in case a human enters the restricted area. It also sets off an alarm.

Img3. Humans at a factory

CONCLUSION

Through the course of the project, a detailed study of artificial neural networks was done. Considering various scenarios in border security like changing light conditions, the database for face detection was created in varied light conditions.

The generic artificial neural network created for human detection can identify 25 different objects, but our area of interest with respect to border application requires only human detection. For the same reason, the artificial neural network has been trained to identify humans in particular.

Once a human is detected, an alarm sets off and a 10X zoom-in face detection camera is activated. This camera runs for an artificial neural network specifically trained to distinguish unknown faces from the registered ones, suggesting infiltration.

The human detection model has been developed to classify humans and other objects based on their individual and relative confidence levels. Meaning thereby, if an object matches the features of two different classes, then the model reiterates its classification to find the perfect matching class using a function called: *max_num_classes*

. The label for every identified object, here human, includes its class name as well as the confidence level or essentially probability match. In order to avoid unnecessary hardware interfacing, a software generated alarm was used on human detection.

The command for this is *winsound.play ('alert'*), where 'alert.wav' was a siren sample file stored in our database.

For face detection, we achieved recognition of registered faces and also a red alert box around unregistered faces, denoting infiltration or possibility of security breach.

Both programs run simultaneously to give an almost real-time intrusion detection. This has been achieved using tools that suffice lab demonstration conditions. For real field environment, military purpose cameras with HD and night vision and high-speed GPU will be required to achieve similar output.

FUTURE SCOPE

The existing human patrolling on borders can be replaced by the above proposed technology. Further, once the infiltration is confirmed, an automated relay gun can be mounted on a servo motor, that is controlled by the neural network with provision for human control.

This will greatly save human loss on military bases and give a time efficient and accurate means of combat. Moreover, the normal cameras used in the demonstration model need to be replaced by night vision cameras for day and night surveillance while practical execution. There can also be a thermal camera instead of a night vision camera or a normal camera. The type of camera will not matter as the program itself is compatible to any kind of image or video input. Only the features of human should be provided for detection. Moreover, in real field application, the USB camera shall be replaced by wireless camera module.

APPENDIX A

PYTHON

Python is a simple dynamic programming language that can be very efficiently used to create neural networks and in image/video processing. This comes to great use in development of machine learning algorithms in Artificial intelligence as a more efficient alternative to other softwares like MATLAB, in addition to being open source. For our project we have used Python with Opencv and Tensorflow. Python interfacing with hardware is also pretty easy using just relevant libraries or Raspberry pi.

MODELS

The models used in our project are:

- Fast R-CNN- Fast R-CNN is an object detection algorithm proposed by Ross Girshick in 2015. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs a region of interest pooling scheme that allows to reuse the computations from the convolutional layers.
- SPPnet- Spatial Pyramid Matching or Spatial Pyramid Pooling is a technique that helps us to handle/use multi-scale images efficiently, especially while dealing with machine learning algorithms to address problems like classification. The technique performs pooling (ex: Max pooling) on the last convolution layer (either convolution or sub sampling) and produces a N*B dimensional vector (where N=Number of filters in the convolution layer, B= Number of Bins). The vector is in turn fed to the FC layer. The number of bins is a constant value. Therefore, the vector dimension remains constant irrespective of the input image size. Spatial Pyramid Pooling (SPP) is the key feature of the SPP Net architecture.
- Multiple Convolution (Maxpooling)- Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

FUNCTIONS USED IN PYTHON

- Convolution- There are four main steps in CNN: convolution, subsampling, activation and full connectedness. The first layers that receive an input signal are called convolution filters. Convolution is a process where the network tries to label the input signal by referring to what it has learned in the past. If the input signal looks like previous cat images it has seen before, the "cat" reference signal will be mixed into, or convolved with, the input signal. The resulting output signal is then passed on to the next layer.

- Maxpooling- Pooling creates summaries of feature information in distinct regions of each frame.

  Dimension Reduction: In deep learning when we train a model, because of excessive data size the model can take huge amount of time for training. Now consider the use of max pooling of size 5x5 with 1 stride. It reduces the successive region of size 5x5 of the given image to a 1x1 region with max value of the 5x5 region. Here pooling reduces the 25 (5x5) pixel to a single pixel (1x1) to avoid curse of dimensionality.

  Rotational/Position Invariant Feature Extraction: Pooling can also be used for extracting rotational and position invariant feature. Consider the same example of using pooling of size 5x5. Pooling extracts the max value from the given 5x5 region. Basically, extract the dominant feature value (max value) from the given region irrespective of the position of the feature value. The max value would be from any position inside the region. Pooling does not capture the position of the max value thus provides rotational/positional invariant feature extraction.

LIBRARIES

- matplotlib- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

- util-In python, util is a class that defines a set of methods that perform common, often re-used functions. This section will introduce methods to handle useful date types such as

Dynamic, Datetime, UUID, Regular Expression and URI, to query collections by Language Integrated Query, to encode and decode strings, to encrypt and decrypt data, and so on.

- _tkinter.lib- Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk. Tkinter is not the only Gui Programming toolkit for Python. It is however the most commonly used one.
- Python3.lib- Function to import libraries in Python version 3
- Pyusb- Hardware to python program USB interfacing library
- Idle- Python cding environment, just like Pycharm, Spyder, sublime text,etc.
- Import- command that imports a function or library
- Test- Test function is used to test the pre-trained model on a set of random data for validation.
- Sqlite3- It is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.
- Msilib- The msilib supports the creation of Microsoft Installer (.msi) files. The package contents can be roughly split into four parts: low-level CAB routines, low-level MSI routines, higher-level MSI routines, and standard table exposes an API to create CAB files.
- Urllib- urllib.request is a Python module for fetching URLs (Uniform Resource Locators). It offers a very simple interface, in the form of the urlopen function.
- Xml- It is a metalanguage which allows users to define their own customized markup languages, especially in order to display documents on the Internet.
- http- The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

- xmlrpc- XML-RPC is a remote procedure call (RPC) protocol which uses XML to encode its calls and HTTP as a transport mechanism. "XML-RPC" also refers generically to the use of XML for remote procedure call, independently of the specific protocol. It uses the HTTP protocol to pass information from a client computer to a server computer.
- pydoc_data- The pydoc module automatically generates documentation from Python modules. The documentation can be presented as pages of text on the console, served to a Web browser, or saved to HTML files. Here, database files.
- distutils- The distutils package provides support for building and installing additional modules into a Python installation. The new modules may be either 100%-pure Python, or may be extension modules written in C, or may be collections of Python packages which include modules coded in both Python and C.
- collections- The Collections module implements high-performance container datatypes (beyond
  the built-in types list, dict and tuple) and contains many useful data structures that you can use to store information in memory.
- asyncio- This module provides infrastructure for writing single-threaded concurrent code using coroutines, multiplexing I/O access over sockets and other resources, running network clients and servers, and other related primitives.
- dbm- The dbm module has been renamed to dbm.ndbm in Python 3. The 2to3 tool will automatically adapt imports when converting your sources to Python 3. The dbm module provides an interface to the Unix "(n)dbm" library. Dbm objects behave like mappings (dictionaries), except that keys and values are always strings.
- ensurepip- The ensurepip package provides support for bootstrapping the pip installer into an existing Python installation or virtual environment. This bootstrapping approach reflects the fact that pip is an independent project with its own release cycle, and the latest available stable version is bundled with maintenance and feature releases of the CPython reference interpreter.

APPENDIX B (code)

CODE HUMAN Detection

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import winsound

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

import cv2
cap = cv2.VideoCapture(0)

from utils import label_map_util
from utils import visualization_utils as vis_util

PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
NUM_CLASSES = 90
tar_file = tarfile.open(MODEL_FILE)
for file in tar_file.getmembers():
  file_name = os.path.basename(file.name)
  if 'frozen_inference_graph.pb' in file_name:
    tar_file.extract(file, os.getcwd())



# ## Load a (frozen) Tensorflow model into memory.
```

```
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
def load_image_into_numpy_array(image):
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)

# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)

duration = 100  # millisecond
freq = 800  # Hz

# In[10]:

with detection_graph.as_default():
  with tf.Session(graph=detection_graph) as sess:
    while True:
      ret, image_np = cap.read()
      # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
      image_np_expanded = np.expand_dims(image_np, axis=0)
      image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
      # Each box represents a part of the image where a particular object was detected.
      boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
```

```
# Each score represent how level of confidence for each of the objects.
# Score is shown on the result image, together with the class label.
scores = detection_graph.get_tensor_by_name('detection_scores:0')
classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

# Actual detection.
(boxes, scores, classes, num_detections) = sess.run(
    [boxes, scores, classes, num_detections],
    feed_dict={image_tensor: image_np_expanded})

## keeping class 1 i.e person detection valid and making other class's score to be zero
for i in range(100):
  if(classes[0][i]!=1): #if 1st class then cool
    scores[0][i] = 0;    #if not bye bye
  elif(classes[0][i]==1 and scores[0][i]>0.75): # be class 1 and threshold above 75%
      ##buzzer on
      winsound.Beep(freq, duration)



print(scores)  #get scores; percentage

# Visualization of the results of a detection.
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8)

cv2.imshow('object detection', cv2.resize(image_np, (800,600)))
if cv2.waitKey(25) & 0xFF == ord('q'):
```

```
cv2.destroyAllWindows()
break
```

CODE 2

```
# create_database.py
import cv2, sys, numpy, os, time
count = 0
size = 4

fn_haar = 'haarcascade_frontalface_default.xml'
fn_dir = 'database'
fn_name = 'DINESH'   #name of the person

path = os.path.join(fn_dir, fn_name)
if not os.path.isdir(path):
   os.mkdir(path)
(im_width, im_height) = (112, 92)
haar_cascade = cv2.CascadeClassifier(fn_haar)

webcam = cv2.VideoCapture(1)

print ("----------Taking pictures----------")
print ("----------Give some expressions----------")
# The program loops until it has 20 images of the face.

while count < 100:
   (rval, im) = webcam.read()
   im = cv2.flip(im, 1, 0)
   gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
   mini = cv2.resize(gray, (int(gray.shape[1] / size), int(gray.shape[0] / size)))
   faces = haar_cascade.detectMultiScale(mini)
   faces = sorted(faces, key=lambda x: x[3])


if faces:
```

```
            face_i = faces[0]
            (x, y, w, h) = [v * size for v in face_i]
            face = gray[y:y + h, x:x + w]
            face_resize = cv2.resize(face, (im_width, im_height))
            pin=sorted([int(n[:n.find('.')]) for n in os.listdir(path)
                 if n[0]!='.' ]+[0])[-1] + 1
            cv2.imwrite('%s/%s.png' % (path, pin), face_resize)
            cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
            cv2.putText(im, fn_name, (x - 10, y - 10), cv2.FONT_HERSHEY_PLAIN,
               1,(0, 255, 0))
        time.sleep(0.38)
        count += 1

      cv2.imshow('OpenCV', im)
        key = cv2.waitKey(10)
        cv2.destroyAllWindows()
        if key == 27:
            break

print (str(count) + " images taken and saved to " + fn_name +" folder in database ")
        cv2.destroyAllWindows()
```

CODE 3

```
# facerec.py
import cv2, sys, numpy, os,time
cv2.__version__

size = 4
fn_haar = 'haarcascade_frontalface_default.xml'
fn_dir = 'database'

(im_width, im_height) = (112, 92)
# Part 1: Create fisherRecognizer
print('Training...')
# Create a list of images and a list of corresponding names
(images, lables, names, id) = ([], [], {}, 0)
for (subdirs, dirs, files) in os.walk(fn_dir):
    print (subdirs," ",dirs," ",files)
    for subdir in dirs:
        print (subdir)
        names[id] = subdir
        subjectpath = os.path.join(fn_dir, subdir)
        for filename in os.listdir(subjectpath):
            print (filename)
            path = subjectpath + '/' + filename
            lable = id
            img = cv2.imread(path, 0)
            img = cv2.resize(img, (im_width, im_height))
            images.append(img)
            lables.append(int(lable))
        id += 1
(im_width, im_height) = (112, 92)

# Create a Numpy array from the two lists above
(images, lables) = [numpy.array(lis) for lis in [images, lables]]
```

```
# OpenCV trains a model from the images
model = cv2.face.FisherFaceRecognizer_create()
model.train(images, lables)

size = 4
fn_haar = 'haarcascade_frontalface_default.xml'
fn_dir = 'database'
(im_width, im_height) = (112, 92)
haar_cascade = cv2.CascadeClassifier(fn_haar)
webcam = cv2.VideoCapture(1)
# frame = cv2.flip(frame,1,0) to flip
print(names)
  while True:
    (rval, im) = webcam.read()
    im = cv2.flip(im, 1, 0)
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    mini = cv2.resize(gray, (int(gray.shape[1] / size), int(gray.shape[0] / size)))
    faces = haar_cascade.detectMultiScale(mini)
    faces = sorted(faces, key=lambda x: x[3])
    if faces:
        face_i = faces[0]
        (x, y, w, h) = [v * size for v in face_i]
        face = gray[y:y + h, x:x + w]
        face_resize = cv2.resize(face, (im_width, im_height))
        person,pred = model.predict(face_resize)
        if person == 0:
            cv2.rectangle(im, (x, y), (x + w, y + h), (0, 0, 255), 3)
            cv2.putText(im, names[person], (int(x -20), int(y - 20)),
cv2.FONT_HERSHEY_DUPLEX,
                2,(0, 255, 0))
        else:
            cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3) #rectangle colour
changer
```

```
            cv2.putText(im, names[person], (int(x -20), int(y - 20)),
cv2.FONT_HERSHEY_DUPLEX,
                2,(0, 0, 255))
    cv2.imshow('OpenCV', im)
    key = cv2.waitKey(10)
    if key == 27:
        break
cv2.destroyAllWindows()
```

REFERENCES

[1] Ross Girshick Microsoft Research by rbg@microsoft.com (Girshick, Ross. "Fast R-CNN" 2015)

[2] Automatic security system for recognizing unexpected motions through video surveillance

[3] doi.org

[4] Pantech smart surveillance solutions

[5] Survey paper on Smart surveillance system (github)

[6] Report on “human detection” by Jignesh S. Bhatt (IIT Vadodara)

[7] Smarter-control-border-patrol by Engineering Arizona

[8] Detection of enemy intervention in border and battle fields using smart dusttechnology.pdf

[9] Explanation of classification and feature extraction for human detector. Pdf

[10] Human detection and recognition using open source computer vision library.pdf