

Tailgating Detection and Security Alerting System

Keerthana B¹, Kajaal R², Sahana V³

¹Keerthana B, Student, Dept. of Electronics and Communication Engineering, Sir MVIT, Karnataka, India

²Kajaal R, Student, Dept. of Electronics and Communication Engineering, Sir MVIT, Karnataka, India

³Mrs.Seema, Professor, Dept. of Electronics and Communication Engineering, Sir MVIT, Karnataka, India

Abstract – Tailgating is the one where an employee holds the office door for others to enter into the building with one access.

This leads to insecurity, wherein the unknown person can also enter the building with the access of the original employee. To overcome this, we introduce a security system that prevents tailgating that provides authentication, accuracy, flexibility and gives more convenience to the security guards. It uses Yolov3 object detection model to detect the door and the total number of people by converting each frame of the video into an image. For the first 30 frames, we detect the presence of door since a door is opened and closed within the initial part of the video and for the remaining part, we detect and count the number of people entering the building and decide if the person is being tailgated or not and alert the security accordingly

Key Words: Yolov3, door detection, person detection, security alert.

1. INTRODUCTION

The purpose of the security system is mainly to avoid malpractices or any other form of actions against security. In early times, a person needs to be available for all the time to monitor the in and out entries of all the employees or unknown person.

In this if that person is not available there may be a chance for the unknown person to enter into the restricted area. So, this can lead to many problems. Later time this is enhanced by introducing some biometric actions like fingerprint, face recognition, Iris recognition for monitoring.

This has reduced the man power and improved some sort of security. This is implemented in many places like office, banks, and private places. Other than this biometric many electronic accesses control is available nowadays RFID tags, etc. But here there may arise disadvantage like one person can take the access card of another person who is authorised and enter into the building, where in biometric this will not happen but sometimes tailgating may occur.

The entering persons is recognized and detected based on the trained datasets in database the door will open, otherwise if the unauthorized person is detected the door remains closed.

If suppose the unauthorized person goes along with access person, the second level of authentication takes place here. The camera captures the image of the unauthorized person.

1.1 Problem Statement

Given a CCTV footage of a secure door entry, detect tailgating using OpenCV Python and alert the security team. Further by image processing, report the entry of the unauthorized person by capturing the image.

1.2 Objective

Our objective is to detect tailgating using openCV and alert the security team via a spreadsheet that keeps track of the people who enter.

This can reduce the workload of a human operator and also minimize the room of errors by assisting human in making decision.

Firstly, we detect the door from the video input and extract its boundaries. When a person swipes his/her ID card or access card: Employee id, Time and Date of the swipe is noted and check if the person is authorized or not, from the spreadsheet.

If the person is authorized, we further do image segmentation to segment persons in the video to check tailgating.

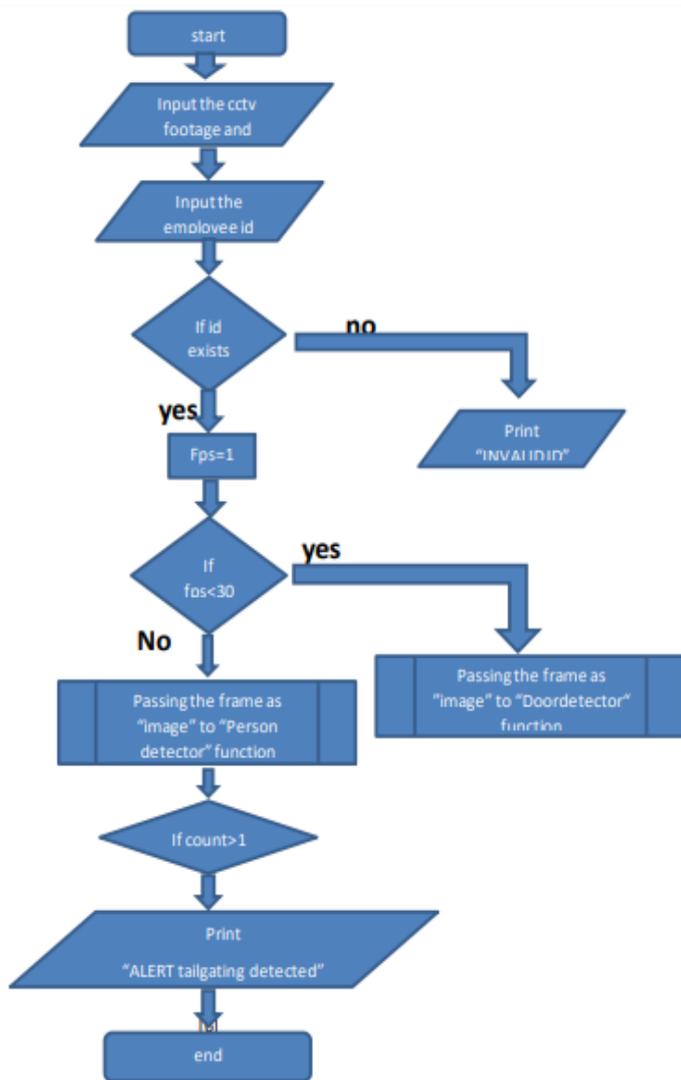
If a person is authorized, we further do image segmentation to segment persons in the video to check tailgating.

If tailgating, i.e. if two or more people crossing the boundary of the door is detected, this is reported and status is marked against the employee for further investigation.

2. METHODOLOGY FOLLOWED

2.1 Flow of the process

The flowchart below depicts the proposed system:



In this we first input the security camera footage i.e.tailgate.mp4 and we also input the employee's id, we check whether the employee with the employee id exists sing access_info.csv file.

If the employee id is present, the current date and time is stored in the file and the detection for tailgating is continued otherwise INVALID ID is printed.

To detect a person and the number of persons entering, the function 'personDetector' is called. And the function "door detector" is called to detect and extract the coordinates of the door.

Both the functions use the yolov3 object detection algorithm.

Once the confidence is greater than 50%, the person's detector function and 25% for the door detector function, their coordinates are extracted.

Then these coordinates are used to draw the rectangular or the bound box around the person and the door.

And if the count of people in the "person detector" function is more than one then ALERT! TAILGATING DETECTED is printed.

This information is stored back in access_info.csv file with the current date and time and also the status of tailgating, whether it occurred or not.

2.2 Overview of Model

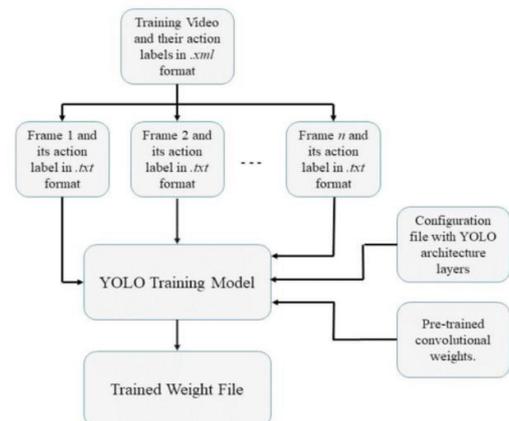


fig -1: overview of flow of the model

The inputted video is broken into frame by frame for each second, and each framed is passed into the into the yolo training model that has pre-trained convolutional weights and configuration file with yolo architecture layer.

Thus, YOLO can only detect objects belonging to the classes present in the dataset used to train the network. We will be using the official weight file for our detector. These weights have been obtained by training the network on COCO dataset, and therefore we can detect 80 categories.

Refer the appendix for code

2.3 Flow Of Detection Process

We take the input video "tailgate.mp4" and pass it as the input to the OpenCV model. We also create a data frame of all the authorized personnel to enter the building using "access_info.csv" file and through NumPy package. Once this is done, we ask the user to enter the employee id and we check if this employee id is one among the list of employee ids in the data frame created using access_info.csv file. We also store the current date and time as the timestamp to store if the employee was tailgated or not in the csv file.

Once this employee is found to be authorized, we proceed to perform object detection in the video. For this, we first check the frame per second(fps) of the video. If this fps is less than 30, it indicates it is in the initial bits of the video and since we open/close the door in the initial stages of the video, we call the door detector function if the fps is less than 30. Else, if the fps is greater than 30, we proceed to detect the person in the

video since after the initial bits, the authorized personnel will proceed to enter the building.

Door detector function :

In the door detector function, we receive an input parameter, "image" which is the image version of the current frame in the video. For this image, we define certain parameters such as the minimum confidence(probability) of the object detection that we require to accept it as a valid detection. We also set nmsthreshold, which is the minimum value for the yolo object to draw the most probable boundary box around the object detected. Since yolo, creates hundreds of boxes across the image, nmsthreshold helps us retain only one boundary box which is the most probable one among all the boundary boxes draw by yolo.

We then convert this image into a blob image, wherein a blob image is the 4D NumPy array object of the image. We also scale down the image so that the size is reduced, and the detection process is faster by yolo.

After the blob image is created, we read in the classes file in read text mode and store the list of all classes detectable in a variable called classes.

Apart from this, we have two files "config" and "weights", wherein, weights file is the file on which our object detection model, in this case, different types of doors, was trained on as images and the configuration file(config) is the output obtained on training the weights file.

We define the location of these two files in our system and we import it as modelConfig and modelWeights which are the ideal configuration and weights file for our door detection. From these files, we create a net which is nothing but a variable in which the yolo network model located on our hard disk is loaded into the OpenCV model.

We then set the blob image as input to the net which has the yolo network model. We then set the width and height for the frame passed as image. This blob image goes through several layers to detect the most appropriate class of the object. We store all these layer names in a variable called layerNames and among all these layers, we choose only the layers which give us a probability of certain class and store it as a 2D NumPy array.

We then iterate through this 2D array and for every class detected with a probability greater than the minimum confidence threshold, we find the coordinates and center points of that image.

These points are appended to draw the bounding box around that object with a label "door". For this purpose, we use NMSBoxes which defines the indices and for each indice, we are setting the top, bottom, left and right coordinates and

drawing the lines joining these points for every matching classId thereby resulting in the boundingbox.

Person detector function:

Similar to the door detector function, In the person detector function, we receive an input parameter, "image" which is the image version of the current frame in the video. We convert this image into a blob image, wherein a blob image is the 4D NumPy array object of the image.

We also scale down the image so that the size is reduced, and the detection process is faster by yolo. After the blob image is created, we set the ideal weights and configuration file and similar to the door detector function, we extract all the classes available in yolo detection and store them in a variable.

After setting these ideal configurations and weights file on which the object detection model was trained, we create a net which is nothing but a variable in which the yolo network model located on our hard disk is loaded into the OpenCV model. We then set the blob image as input to the net which has the yolo network model.

We then set the width and height for the frame passed as image. This blob image goes through several layers to detect the most appropriate class of the object. We store all these layer names in a variable called layerNames and among all these layers, we choose only the layers which give us a probability of certain class and store it as a 2D NumPy array. We then iterate through this 2D array and for every class detected with a probability greater than 0.5, we find the coordinates and center points of that image. These points are appended to draw the bounding box around that object with a label "person".

For this purpose, we use NMSBoxes which defines the indices and for each index, we are setting the top, bottom, left and right coordinates and drawing the lines joining these points for every matching classId thereby resulting in the bounding box.

This bounding box is drawn in green color. For every person detected, the count of person's us increased by 1. When this count is greater than 2, it indicates that somebody is being tailgated because only one person is authorized for one employee id and hence, it shows a message "ALERT!

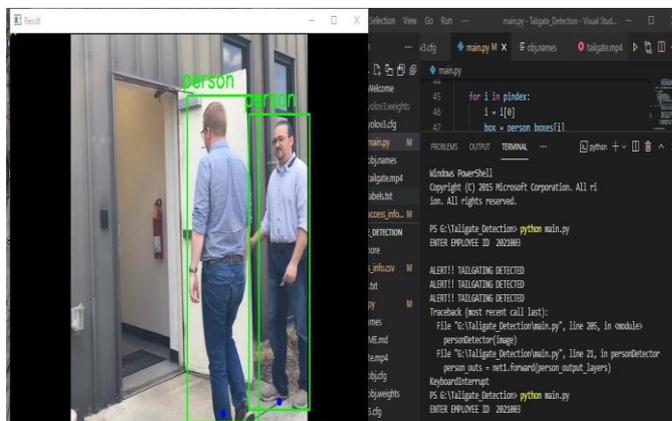
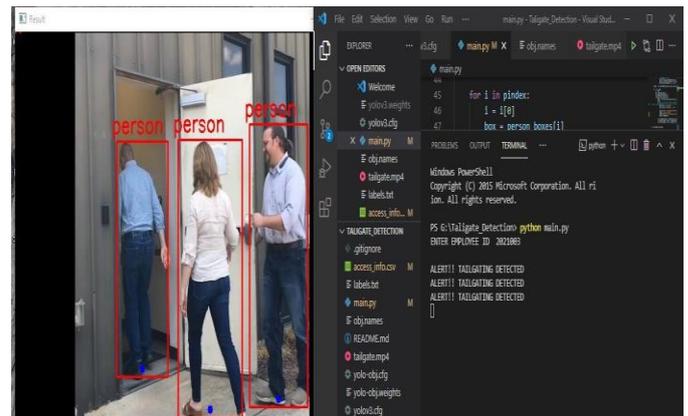
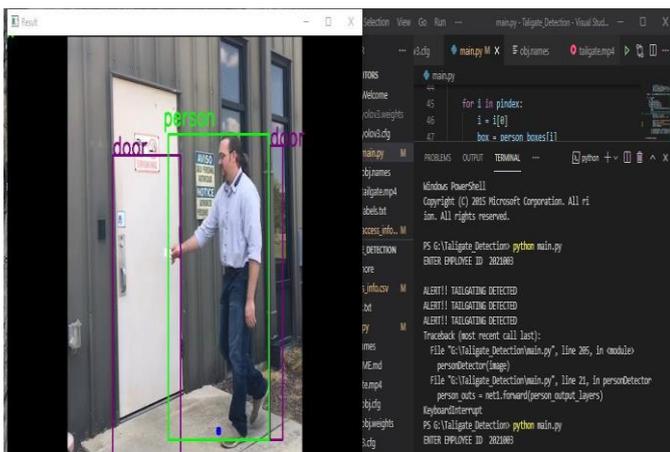
TAILGATING DETECTED" in the console and simultaneously changes the bounding box color to red indicating it is unauthorized and that the person is being tailgated. This information of whether the person was tailgated or not is stored in the csv file along with the time and date of being tailgated.

We also use a python package called “pyttsx3”, which is a text to speech converter so that every time a person is found to be tailgated, a voice message is also heard saying “alert!

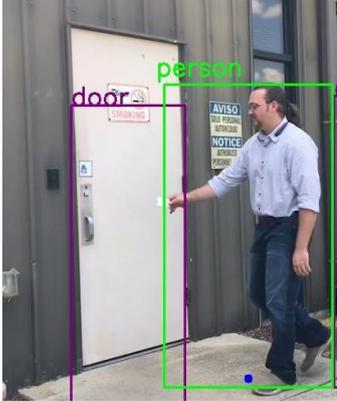
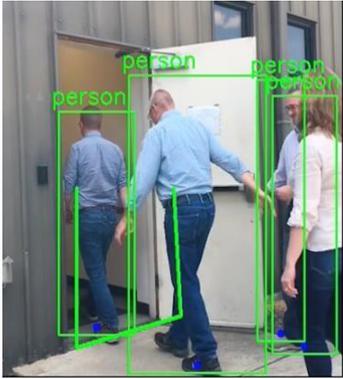
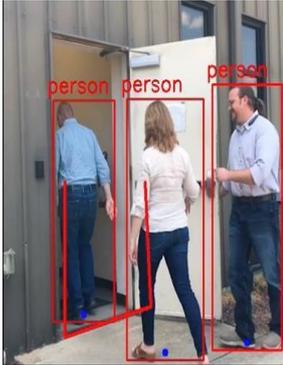
“Tailgating detected”. The program stops once the video reaches the end and the appropriate details of the result are reflected in the csv file.

3. EXPERIMENTAL RESULTS AND ANALYSIS

3.1 Output



3.2 Outcome

| Outcomes | Explanation |
|---|---|
|  <p data-bbox="300 790 584 822">Fig. 4.4.1 Door detection</p> | <p data-bbox="635 454 1340 517">This figure shows detected boundary of the door from the trained model</p> |
|  <p data-bbox="292 1292 592 1323">Fig.4.4.2 Person detection</p> | <p data-bbox="635 891 1340 1072">After drawing the boundary of the door, we verify if a person is authorized or not from the spreadsheet and take note of time and date of swipe. If authorized, we further Segment people in the video to detect tailgating. We used YOLOV-3 pre-trained model to detect people in the frame. Output of detected people in frame is shown in image.</p> |
|  <p data-bbox="264 1794 620 1825">FiFig.4.4.2 Tailgating detection</p> | <p data-bbox="635 1359 1340 1637">We use the tagged point to detect crossing of the door boundary. Concept used to detect crossing is Direction of a Point from a Line Segment using cross product. If there are 2 or more than 2 points that crossed the boundary line of the door. We can infer that there are more than 1 person entering the door. This leads to Tailgating (entry without authorization). Image shows that tailgating is detected. When tailgating is detected, this alerts the security and marks as tailgating in spreadsheet against the authorized person record by whom tailgating is detected.</p> |

4. CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

The implementation of the detection of tailgating and alerting the security team using openCV with yolov3 has been successfully done.

4.2 Future Work

We are looking forward for further improvement in this project by:

By adding other features like natural language processing (NLP) which would welcome/greet the authorized person and also say when there is security breach.

We can further use metal detector along with the existing project to check if a person carries any suspicious objects and also report that back. This can be achieved by training yolov3 with various other objects

ACKNOWLEDGEMENT

It is our earnest to express our thanks to all who contributed directly or indirectly to our project.

Firstly, we would like to thank **Sir M. Visvesvaraya Institute of Technology & Department of Electronics & Communication** for giving us an opportunity. We would like to thank our teacher & guide **Prof. Mrs. Seema S**, who initiated us to complete this project & guided us timely.

Last but not least we express our gratitude to all staff members & our friends for their excellent suggestions & coordination.

There might be some problems or bugs & additional requirements with this system. In future these problems will be corrected accordingly. For this your valuable suggestions are most welcomed.

We once again thank you all for your coordination.

REFERENCES

[1]. T.W. Chan, V.V. Yap, C.S. Soh, (2012) Embedded Based Tailgating / Piggybacking Detection Security System, IEEE Colloquium on Humanities, Science & Engineering Research (CHUSER 2012), IEEE, 277-282.

[2]. D. Nikhil Reddy, B. Mahadev, K.V. Achyuth, Sharmila Nagesawaran, (2018) Development of Security System to Prevent Tail-Gating, In 2018 International Conference on Communication and Signal Processing (ICCSP), IEEE.

[3]. R. Nandhini, N. Duraimurugan, S.P. Chokkalingam, Facial Recognition Based Attendance System, International Journal of Engineering and Advanced Technology (IJEAT) 8 (2019) 574- 577.

[4]. N.S. Tummala, P.C. Sekar, (2017) Face Recognition Using PCA and Geometric Approach, In 2017 International Conference on Computing Methodologies and Communication (ICCMC), IEEE, 562-565.

[5]. M. Sahani, C. Nanda, A.K. Sahu, B. Pattnaik, Online Embedded Door Access Control and Home Security System Based on Face Recognition, International Conference on Circuit, Power and Computing Technologies [ICCPCT], IEEE, 1-6.

[6]. www.github.com

[7]. www.towardsdatascience.com

[8]. www.medium.com