Entrance and Exit Management using Video processing

Aruneswari .S , Bharadwaj and Nirali M Dave
Electronics and communication(ECE), 2nd year. B. tech,
Amrita Vishwa Vidhyapeetham Ettimadai,
Coimbatore, Tamilnadu India.
Gmail: niralidave2019@gmail.com ,bharadwaj045@gmail.com , arunasankar01@gmail.com

ABSTRACT:

A people counting method based on Haar-like detection and template-matching algorithm is presented in this paper. The aim of the method is to count pedestrians that are in a metro station automatically using video surveillance camera or any other public forum. The most challenging problem is to count pedestrians accurately in the case of not changing the position of the surveillance camera, because the view that surveillance camera uses in a metro station is always short-shot and nondirect downward view. In this view, traditional methods find it difficult to count pedestrians accurately. Hence, we propose this method.

1. Introduction

Shopping malls and Departmental stores are one of the most crowded public places available. Therefore, it becomes very tough to keep track of the number of people entering and exiting the complex. Older methods include the use of IR rays scanner which measures the count based the time duration for which the signal is interfered. This method is generally inefficient as in crowded places the signal interfered duration in very high and also expensive as the IR detector needs a fully-fledged hardware setup.

The method proposed in this paper makes use of the surveillance camera and also used the recorded video for this purpose. Using Machine learning techniques and Pre-trained detection and tracking algorithm we can implement a counter

2. Related Work

Recently, many image-processing based methods of counting people were proposed. As using in different scenes, these methods are quite different.

- Mittal et al. have proposed a traffic management system using both audio and video data.
- Hou and Pang have developed an effective method for estimating the number of people in a low-resolution image with complicated scenes in real time .
- Chan and Vasconcelos have presented an approach to the prolem of estimating the size of inhomogeneous crowds which are composed of pedestrians that travel in different directions.
- Hashimoto et al. have developed a people-counting system with human information sensors
- Amin et al. have presented a system for counting people in a scene using a combination of low cost, low-resolution visual and infrared cameras.

3. Detecting and Tracking Pedestrian

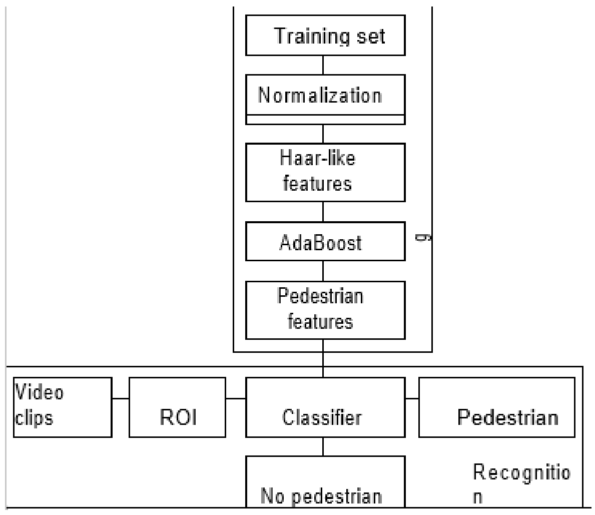
There is a fundamental difference between object detection and object tracking that you must understand.

When we apply object detection we are determining *where* in an image/frame an object is. An object detector is slower, than an object tracking algorithm. Examples of object detection algorithms include Haar cascades, HOG + Linear SVM, and deep learning-based object detectors such as Faster R-CNNs, YOLO, and Single Shot Detectors (SSDs).

An object tracker, on the other hand, will accept the input *(x, y)*-coordinates of where an object is in an image and will:

- Assign a unique ID to that particular object
- Track the object as it moves around a video stream, *predicting* the new object location in the next frame based on various attributes of the frame (gradient, optical flow, etc.)

Basic block diagram:



Combining both object detection and object tracking

Highly accurate object trackers will *combine* the concept of object detection and object tracking into a single algorithm, typically divided into two phases:

Phase 1 — Detecting: During the detection phase we are running our computationally more expensive object tracker to (1) detect if new objects have entered our view, and (2) see if we can find objects that were “lost” during the tracking phase. For each detected object we create or update an object tracker with the new bounding box coordinates. Since our object detector is more computationally expensive, we only run this phase once every *N* frame.

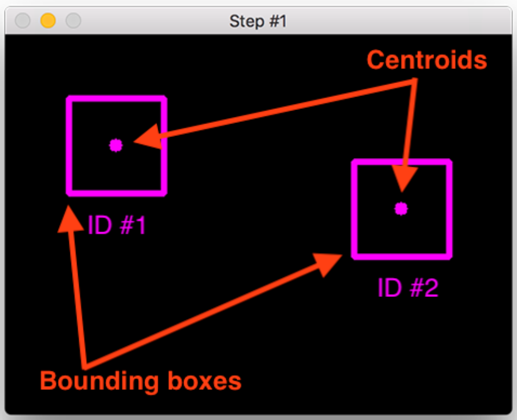
Phase 2 — Tracking: When we are not in the “detecting” phase we are in the “tracking” phase. For each of our detected objects, we create an object tracker to track the object as it moves around the frame. Our object tracker should be faster and more efficient than the object detector. We’ll continue tracking until we’ve reached the *N*-th frame and then re-run our object detector. The entire process then repeats.

Combining object tracking algorithms:

To implement our people counter we’ll be using both OpenCV and dlib. We’ll use OpenCV for standard computer vision/image processing functions, along with the deep learning object detector for people counting.

We’ll then use dlib for its implementation of correlation filters. We could use OpenCV here as well; however, the dlib object tracking implementation was a bit easier to work with for this project.

At **Step #1** we accept a set of bounding boxes and compute their corresponding centroids (i.e., the centre of the bounding boxes):

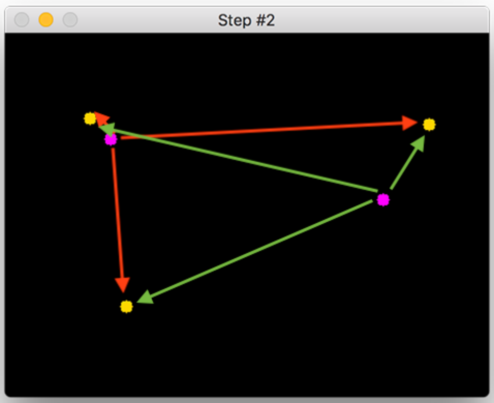


The bounding boxes themselves can be provided by either:

- An object detector (such as HOG + Linear SVM, Faster R- CNN, SSDs, etc.)
- An object tacking algorithm (mainly correlation filter).

During **Step #2** we compute the Euclidean distance between any *new* centroids (yellow) and *existing* centroids (purple):

In the above image you can see that we have two objects to track in this initial iteration of the algorithm.

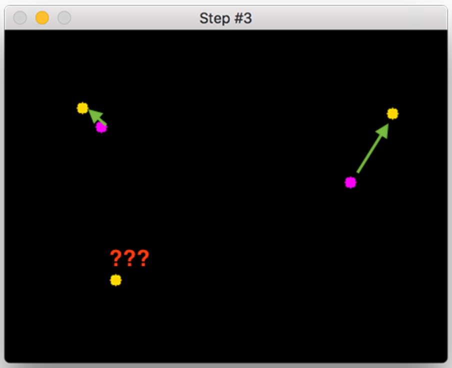


The centroid tracking algorithm makes the assumption that pairs of centroids with minimum Euclidean distance between them *must be the same object ID*.

In the example image above, we have two existing centroids (*purple*) and three new centroids (*yellow*), implying that a new object has been detected (since there is one more new centroid vs. old centroid).

The arrows then represent computing the Euclidean distances between all purple centroids and all yellow centroids.

Once we have the Euclidean distances, we attempt to associate object IDs in **Step #3**:



Registering simply means that we are adding the new object to our list of tracked objects by:

- Assigning it a new object ID
- Storing the centroid of the bounding box coordinates for the new object

In the event that an object has been lost or has left the field of view, we can simply deregister the object (**Step #5**).

Exactly how you handle when an object is "lost" or is "no longer visible" really depends on your exact application, but for our people counter, we will deregister people IDs when they cannot be matched to any existing person objects for 40 consecutive frames.

Creating a "trackable object"

In order to track and count an object in a video stream, we need an easy way to store information regarding the object itself, including:

- It's object ID

- It’s previous centroids (so we can easily to compute the direction the object is moving)

- Whether or not the object has already been counted

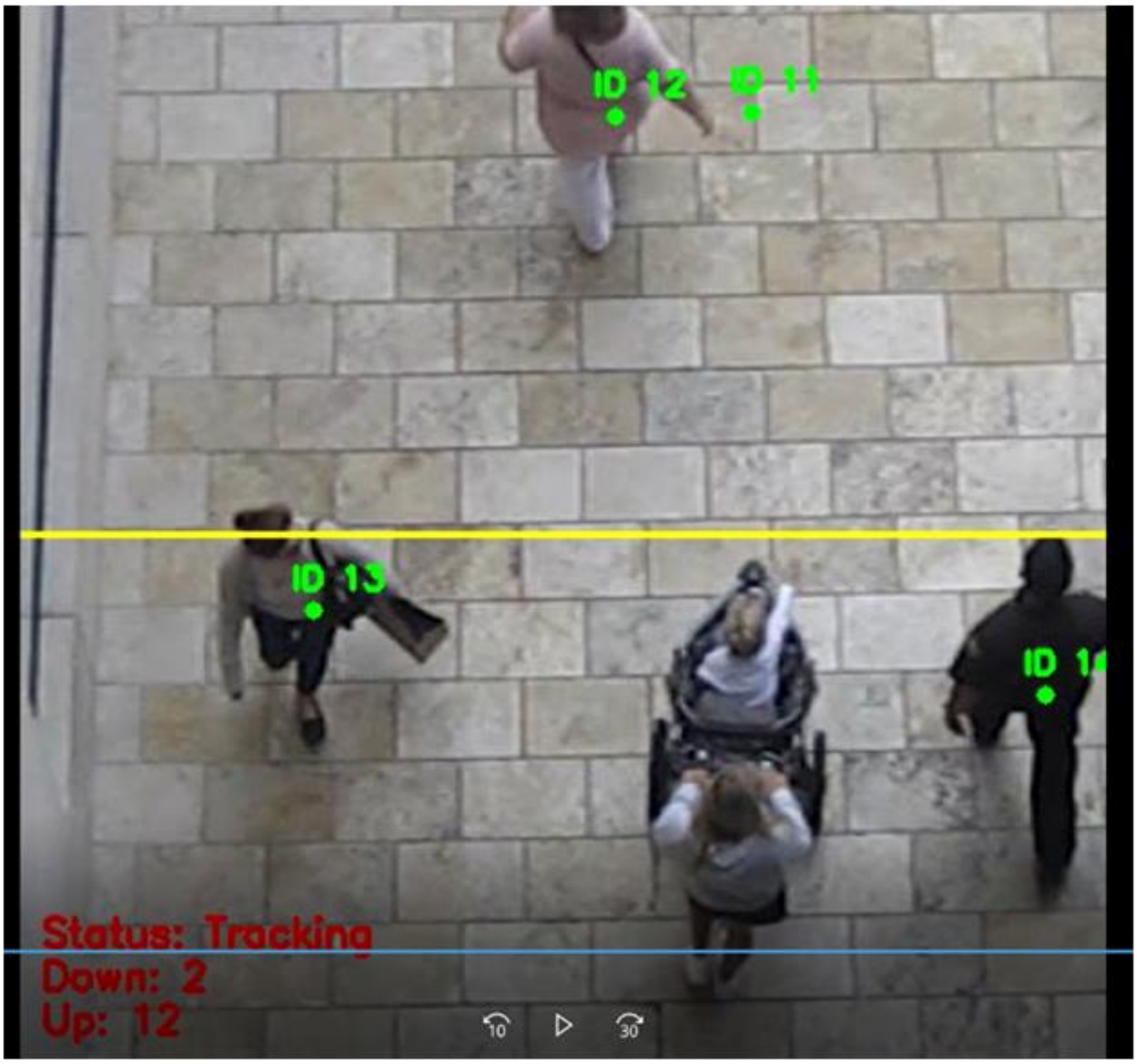
Results

To test the results of our method, we design a piece of algorithm named Entry Exit Counter. This algorithm is executed using python program using Open Cv, Dlib and Imutils library for implementation of different standard algorithm. we have made use of the Supervised Caffe model which has the maximum accuracy amongst various CNN model in object detection. This model also ensures that it only tracks the people rather than any other object. The minimum detection probability should be greater than 0.4 as it skips the weak signal which are captured in the frame.

Although the method we present in this paper makes a good performance on counting pedestrian, counting errors like mistake and missing still exist. There are two reasons of missing counting. One is due to pedestrian wearing hats with a similar color to the surrounding. The other is pedestrian’s fast speed of running. And the reason causing mistake counting is that pedestrian luggage is similar to pedestrian head. Hence, this method can meet the demands of real-time processing.

\

The output results obtained:



6. Conclusion

This paper proposes a novel method to count pedestrians in metro station based on Haar-like detection and template-matching algorithm. This new method avoids the disadvantages of other computer vision methods in short-shot and nondirect view. Thus there are the following two points between our method and the other traditional methods.

(1) To avoid counting error due to occlusions and body-shape deformation in short-shot and nondirect view, this method uses Haar-like and Caffe Model which are mature in face recognition skill to detect pedestrian.

(2) Template-matching algorithm which includes seven parameters (MIN SIZE, MAX SIZE, GOOD SCORE,GOOD SCORE RATIO, GOOD DIR AVER, GOOD SCORE MAX, and Y AXIS MIN) is presented in our method to track and count pedestrian. The value of these parameters is the key to judge a template whether a pedestrian or not; thus, it has a great influence on the accuracy of counting pedestrian. Nevertheless, we found that the accuracy results cannot meet our demands if we set these parameters with the fixed value. Therefore, a novel method to set these parameters with the threshold-curve is proposed. In other words, a threshold function is built to every parameter so as to improve the accuracy of pedestrian counting.

The experiments prove that our method can make a good performance on counting application. And the accuracy of the method with threshold-curve is nearly ninety percent which is higher than that of the fixed threshold method's seventeen percent, so the method with threshold-curve is proved to be a better method.

References

[1]A. Mittal, A. Jain, and G. K. Agarwal, "Audio-video based people counting and security framework for traffic crossings," Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, vol. 49, no. 3, pp. 377–391, 2007.

[2]Y. Hou and G. K. H. Pang, "People counting and human detection in a challenging situation," IEEE Transactions on Systems, Man, and Cybernetics A, vol. 41, no. 1, pp. 24–33, 2011.

[3]A. B. Chan and N. Vasconcelos, "Counting people with low-level features and Bayesian regression," IEEE Transactions on Image Processing, vol. 21, no. 4, pp. 2160–2177, 2012.