Research Article

Video Analytics based Intelligent Transport System for passenger flow forecast and Social Distancing Indication

K S Gautam[1*], Vishnu kumar Kaliapan [2]

[1] Department of Information Science and Engineering, New Horizon College of Engineering, Bangalore, 560103 India, gautam2ping@gmail.com
[2] Department of Computer Science and Engineering, KPR Institute of Engineering and Technology, Coimbatore, India,vishnuaero321@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 April 2021

Abstract: Crowd density management in the transport sector is still one of the ongoing research problems. Intelligent Transportation System (ITS) is one of the branches of smart cities that aim to achieve better traffic efficiency. The intelligent transport system optimizes the traffic congestion control by acquiring real time data. Optimized traffic congestion control demands a robust system that could count the number of people inside a carrier for taking optimized decisions. In this paper we proposed an intelligent algorithm named Modified Intelligent Centroid Tracker and Counter (MICTC) that could detect, count, and measure the distance between humans in a closed and controlled environment. The proposed algorithm is vision based and the scope of the work is to optimize the congestion control inside the passenger carrier and supports countless use cases like smart transport, buildings, and other demography where social distancing is enforced. MICTC algorithm not only offers visual indication with a bounding box but also generates metadata which gives a clear picture to the concerned operational or administrative head regarding the current passenger count status. The work deployed in the public transport sector as a candid spot to operate. The algorithm delivers an adequate transport facility to the public, as it gathers information on crowd density in a public transport medium to the commuters of every region. The work gathers crowd density information and provides commuters a suggestion on availability of seats in the carrier, which then saves time, avoids catching the crowded carriers, ensures social distancing, and standardizes the public transportation system which has practical significance. On experimental analysis we could infer that the proposed approach works with accuracy of 0.81,0.83, 0.85, 0.88, 0.82, 0.89 on VISOR, Kaggle, CALTECH, Penn-Fudan, Daimler Mono and INRIA respectively.

1. Introduction

In a highly populated country like India, preferably in an urban area, population growth increases rapidly due to rising job opportunities, modernization, and comfort. It is very common to find huge crowds in public transport, which lay a critical burden on city administrators to ensure adequate transportation to the entire crowd which then drags the attention of commuters to have private transportation. Increased number of self-owned vehicles increases Global Warming and Traffic Congestion. Public Transportation plays a major role in growth as well as to make a pollution-free environment for a city. The Intelligent Transport System has become a popular area of research for this decade and it deals with enormous technologies to standardize the public transportation system. The Intelligent Transportation System (ITS) aims to improve safety, mobility, and environmental performance of road transport. This Work provides a structural prototype of ITS to elevate the passengers' comfort. To improve the passenger's comfort, it is necessary to determine the real-time crowd count inside the carriers. The proposed work deploys a camera to detect the number of passengers on the carrier and to determine the collected information to the forthcoming commuters so that they could avoid catching a crowded carrier which then enables them to save time. At once a passenger gets in or out of a public transport system, the MICTC detects, tracks, and counts the passenger. In most of the existing techniques [1], a bimodal histogram is created that discriminates the background from the foreground and as a result, a thresholded image. Contour labelling [2] is developed to the thresholded image which identifies the presence of an object and a mask is created (explain brief). Doing Boolean and operation on the original image [3] and the masked version of it the objects in the foreground [4] are numbered. For every detected region the region of interest (ROI) tracker and bounding box is updated. As detecting ROI across all the frames is computationally expensive, the ROI is detected once and the ROI tracker tracks the target across the rest of the frames [5] with bounding box [6]. This hybrid approach is highly preferred because of the MICTC algorithm where the target is easily located without computational burden.

2. The MICTC Algorithm

In the proposed MICTC algorithm, bounding boxes are drawn for the detected centroids. Such bounding boxes are drawn using an object detector and in the proposed approach, we use Haar Cascades as an object detector.

With an attempt to build a robust object detector that works invariant to lightings and postures the Haar Cascade Classifier is trained with images from benchmark datasets and manually shot images. 2000 training images of humans are captured covering all the pose variations with varying lightings from early morning to late evenings and are resized to 64(width)*128 (height) to reduce computational complexity. Fig 1 explains the workflow of the proposed MICTC Algorithm. The region of interest is detected with a bounding box, the centroid of the box is computed and each centroid is assigned a unique id as shown in Fig 2.

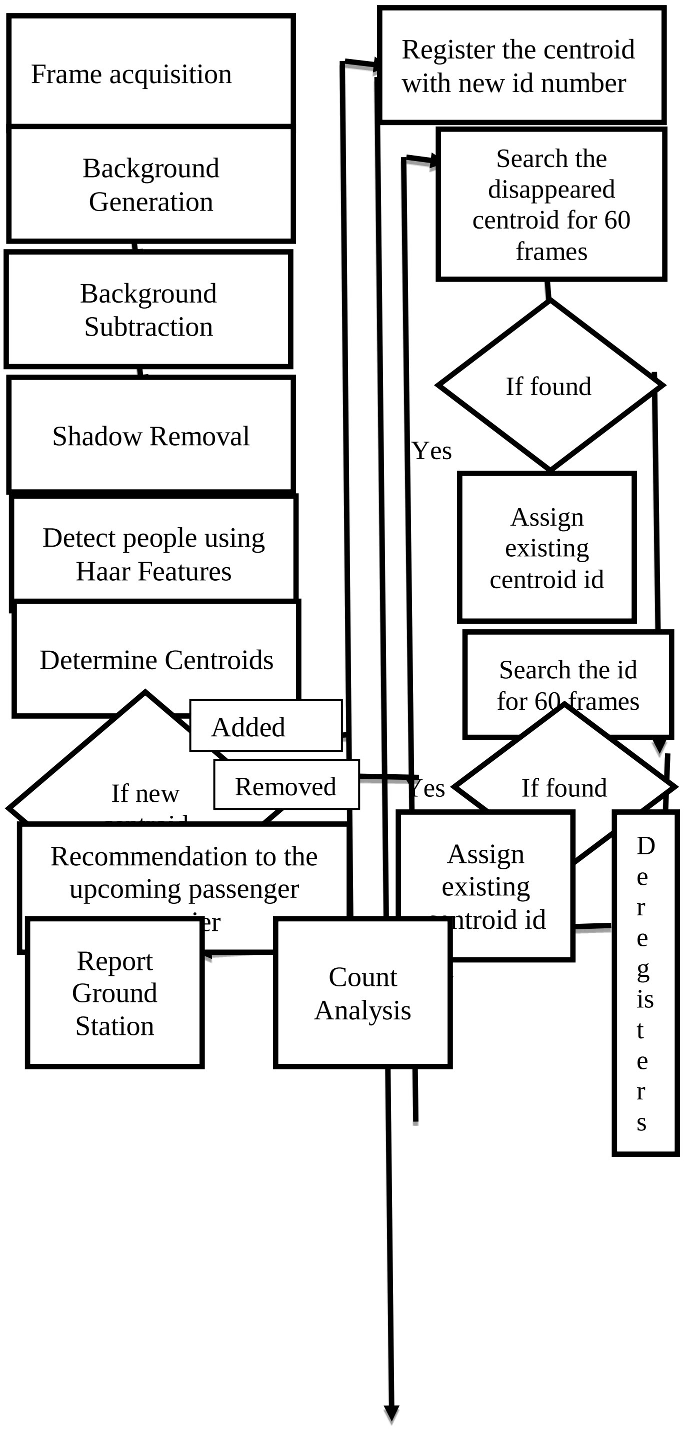
Frame acquisition
Background Generation
Background Subtraction
Shadow Removal
Detect people using Haar Features
Determine Centroids
If new centroid
Added
Removed
Recommendation to the upcoming passenger carrier
Report Ground Station
Count Analysis
Register the centroid with new id number
Search the disappeared centroid for 60 frames
If found
Yes
Assign existing centroid id
Search the id for 60 frames
Yes
If found
Assign existing centroid id
Deregisters

***Fig 1.** Theworkflow of the proposed MICTC Algorithm*

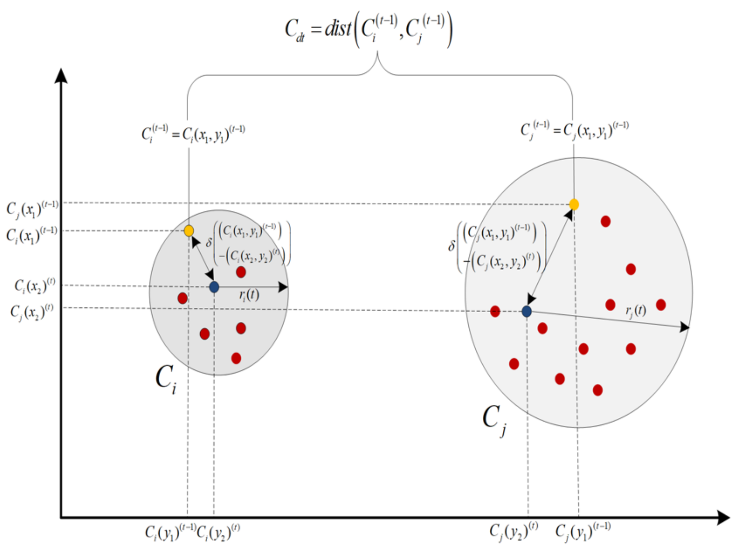
$C_{dt} = dist\left(C_i^{(t-1)}, C_j^{(t-1)}\right)$
$C_i^{(t-1)} = C_i(x_1, y_1)^{(t-1)}$
$C_j^{(t-1)} = C_j(x_1, y_1)^{(t-1)}$
$C_j(x_1)^{(t-1)}$
$C_i(x_1)^{(t-1)}$
$C_i(x_2)^{(t)}$
$C_j(x_2)^{(t)}$
$\delta\left(\left(C_i(x_1,y_1)^{(t-1)}\right) - \left(C_i(x_2,y_2)^{(t)}\right)\right)$
$\delta\left(\left(C_j(x_1,y_1)^{(t-1)}\right) - \left(C_j(x_2,y_2)^{(t)}\right)\right)$
$r_i(t)$
$r_j(t)$
C_i
C_j
$C_i(y_1)^{(t-1)}$
$C_i(y_2)^{(t)}$
$C_j(y_2)^{(t)}$
$C_j(y_1)^{(t-1)}$

Fig 2.*Distance between the clusters on Euclidean space*

2.1. Pre-processing

The video captured from the camera deployed inside the carriers are collected by the frame acquire. Let B and C be the carriers and their camera respectively. The frame acquirer captures all the frames from the corresponding carriers as shown in equation 1. In this work, the total number of carriers and cameras deployed inside is considered to be n.

$$F_n = \{B_1C_1 f_1, B_2C_2 f_2, ...B_nC_n f_n\}$$

For pre-processing, background noise removal is done across all the frames using a Gaussian filter [7]. The values that the noise possess are gaussian transformed and the noise removed frames are gathered and it replaces the input frames captured as shown in equation (2)

$$B_r f\left(f_n\right) = \{B_1C_1 f_1, B_2C_2 f_2, ...B_nC_n f\}$$

Where $n_{i,j}$ is noise value present in the frame $x_{i,j}$ in the frame. $\left(\mu_{i,j}\right)$ is the Local weighted mean and $\left(\sigma_{i,j}\right)$ is the Local weighted variance and $\left(f_{\max i,j}\right)$ is the local maxima of the pixels in the frame. The noisy frame acquired from the camera is shown in equation 3 and overall the collection of frames across all the carriers is shown in equation 4.

$$f_{i,j} = x_{i,j} + n_{i,j}$$

$$f_{i,j} = \{B, c, f\}$$

The local weighted mean (as shown in equation 5), local weighted variance (as shown in equation 6) for the pixels in the frame is calculated so that the variance of a pixel from the mean is calculated for removing Gaussian noise.

$$\mu_{i,j} = \frac{\sum_m \left(\begin{array}{l} \sum_n (m,n) W_{m,n} \rho_{i+m,j+n} \\ + \sum_m \sum_n (m,n) W_{m,n} Y_{i+m,j+n} \end{array} \right)}{\left(\sum_m^u = -u \sum_n^v = -v_{m,n}^w \right)}$$

$$\sigma_{i,j} = \frac{\sum_m \sum_n (m,n) W_{m,n} \left| \rho_{i+m,j+n} - \mu_{i,j} \right|}{\sum_m^u = -u \sum_n^v = -v_{m,n}^w}$$

To find the noisy pixel the local statistical parameters are considered using a flag and is shown in equation 7

$$flag_{i,j} = \begin{cases} 1 if \left(f_{i,j} \rangle \mu_{i,j} + B_{i,j} \right) or \left(f_{i,j} \langle \mu_{i,j} - B_{i,j} \right) \\ 0 otherwise \end{cases}$$

If the flag is 1 the classifier is used else not. After identifying the noise in the frame, the frames are sent for reconstruction where the noisy pixel $x_{i,j}$ are placed with its averaged neighbourhood pixels in equation 8.

$$\hat{x}_{i,j} = \begin{cases} \dfrac{\sum_m \sum_n \begin{pmatrix} (m,n) h_{m,n} \hat{x}_{i+m,j+n} \\ (m,n) h_{m,n} Y_{i+m,j+n} \end{pmatrix}}{\sum_m^u = -u \sum_h^v = -u h_{m,n}} if\left(flag_{i,j} = 1\right) \\ Y_{i,j} otherwise \end{cases}$$

Based on the variance of the pixel from the mean, the noise is removed and Gaussian smoothing is done using equation 9. Where Z is the normalized Gaussian filter and T is the tuning parameter.

$$h_{i,j} = \frac{1}{z} \exp\left\{ -T \frac{\sigma_{i,j}^2 \left(i^2 + j^2\right)}{\sqrt{u_{i,j} + 1}} \right\}$$

On creating a bimodal histogram the shadow pixels are identified and they are removed. The background noise removed frames are shadow removed using a shadow removal function as derived in equation 10. $B_r f\left(f_n\right)_x$ s is intensity reflected from point x in the scene wavelength. λ is the wavelength and R is the magnitude of reflection and L is the image function with shadow pixel from the lower limit X to upper limit d as per equation 10

$$B_r f\left(f_n\right)_x^{shadow(\lambda)} = L_x^d(\lambda) R_x(\lambda) + L_x^a(\lambda)$$

$$B_r f\left(f_n\right)_x^{shadow(\lambda)} = L_x^d(\lambda) R_x(\lambda) + L_x^a(\lambda) R_x(\lambda)$$

To cost a shadow on point x, the shadow intensity in equation 10 can be simplified as per the equation 11

$$B_r f\left(f_n\right)_x^{shadow(\lambda)} = a_x(\lambda) L_x^a(\lambda) R_x(\lambda)$$

Were $a_x(\lambda)$ is the attenuation factor. The shadow-free pixels and the shadow-free frameset are shown in equations 12 and 13 respectively. Thus the pre-processed image set has been gathered as shown in equation 14

$$B_r f\left(f_n\right)_x^{shadow(\lambda)} = L_x^d(\lambda) R_x(\lambda) + a_x(\lambda)^{-1}$$

$$B_r f\left(f_n\right) = \left\{ \hat{x}_{i_1,j_1}, \hat{x}_{i_2,j_2}, ...\hat{x}_{i_n,j_n} \right\}$$

Haar Cascade Classifier

The vector files for positive image sets [8] are created and .dat file for negative images [8] are created. Initial weights are chosen and the weights are updated in the decreasing direction of error. Let equation 14 be the input frames to the Haar cascade classifier

$$\left(B_r f\left(f_n\right)_x\right)_n^{sf(\lambda)} = \begin{Bmatrix} (x_1, y_1), (x_2, y_2).... \\ ......(x_m, y_m), x_i \varepsilon X, \\ y_i \varepsilon \{-1, 1\} \end{Bmatrix}$$

The weight initialization is based on the input frames taken for training and the initial weights are chosen based on equation 15, which considers the total training feature set count.

$$D(i) = \frac{1}{m}$$

M FEATURES ARE TAKEN takes m features that are given as input to the classifier for training. The training is done in such a way that +1 is the expected output for the positive images and -1 is the expected outcome for negative images, and based on the aforesaid outcome the weights are updated as per equation 16.

$$fort = 1....T$$

$$h_t : X \varepsilon \{-1, 1\}$$

Choose $\alpha\varepsilon$□ and Update weights

Z_t is the normalization function chosen so that D_{t+1} is distributed and this produces a strong classifier. The stack of week classifiers built as the result of training are shown in equation 17 and its pictorial representation is shown in Fig 6.

$$H(x)_n = \{H(x)_1, H(x)_2, \ldots\ldots H(x)_n\}$$

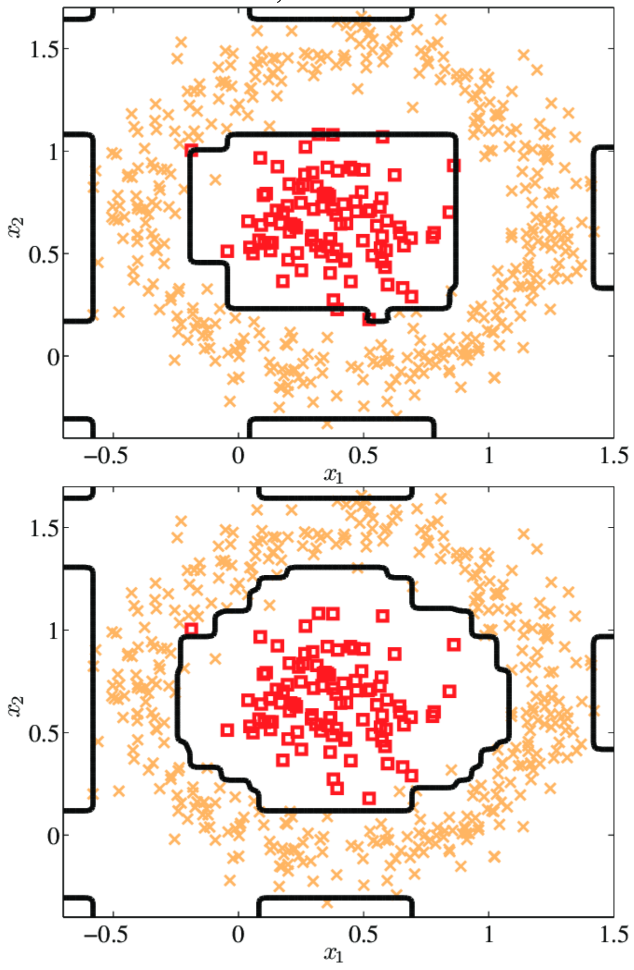

Fig 3. *Stack of weak classifiers on vector space (person vs non-person)*

Equation 18 is the condensed representation of the stack of the weak classifier.

$$H(x) = \left[\sum_{i=1}^{T} \alpha_t h_t(x)\right]$$

The trained classifier is deployed for testing where the algorithm identifies the human inside the carrier and indexes each detected human as centrode as shown in equation 19 and accumulates the total number of centroids as per equation 20.

$$Cd_{p1} = \frac{1}{n}\sum_{i=1}^{n} H(\mathrm{x})_i$$

$$Cd_{pn} = \{cd_{p1}, cd_{p2}, \ldots\ldots, cd_{pn}\}$$

As discussed in Fig 2, for each centroid Euclidian distance is calculated and if the Euclidian distance between the two centroids is below 0.5 and below it, they are considered to be a single centroid and the algorithm concludes overcrowding with positive alarm for ensuring social distancing.

$$d\left(H(x_i), H(x_j)\right) = \sqrt{\sum_{i=1}^{n}\left(H(x_i), H(x_j)\right)^2}$$

3. Experimental Analysis

The experimentation is done using Keras [9] at the front end and Tensorflow [10] as a framework at the backend. For programming real-time computer vision Opencv [11] is used. The training images are taken from the Caltech dataset [12] and manually captured images with the human upper body. 2000 human images are used for training and out of it, 900 images are the human upper body. The images taken for training are of size 128*64 and are collectively shown in Fig 7. 3700 non-human images are used for training and few of them are collectively shown in Fig 8 The computing machine used for training is MacBookAir with I7 processor and experimental analysis is done using fixed focus web camera from the computing device.

Fig 4. *Human images used for training*

Fig 5. *Non-human images used for training*

The exploratory analysis of training data on vector space is shown in Fig 9. Fig 10 shows the training data classified by the algorithm. As per the equation 19, there are weak classifiers stacked based on weights and in our model, we have 8 weak classifiers. The aforementioned stack is made after training and when the test data comes in it passes into the series of 8 weak classifiers in a sequential approach. If the test feature vector is not classified by the first classifier in the stack it is classified as class 2. The test feature vector getting positively classified as a human is passed to the rest of the classifiers to check the presence of human and if it fails at any layer of the cascade, the corresponding test feature vector is considered as non-human as shown in the Fig 6. The potential features that detects whether the test image is person or non-person are the size of a person, the shape of head, the presence of upper body and colour of hair as shown in table 1. The classifier has a stack of four weak classifiers and made a single strong classifier that could classify whether the input image is human or non-human.

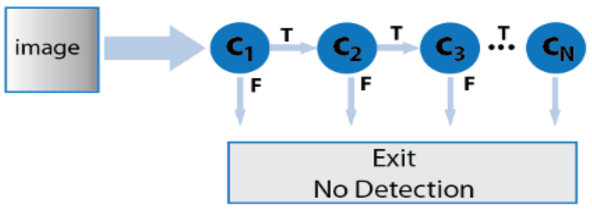

Fig 6. *Cascading weak classifiers*

Table 1.Weak classifiers and their corresponding targeted features

S.No	Cascades	Targeted ROI
1	Classifier 0.1	Height *Width
2	Classifier 0.2	Oval and Circular structure of Head
3	Classifier 0.3	Upper body
4	Classifier 0.4	Black and Brown hair on top

The classification error versus the number of weak classifiers is shown in Fig 11 and it is inferred that when the classifier stack is 4 then the error reduces below 0.1 and there is an abrupt lower drift in error after stacking 0.1 classifier which proves that the 0.1 classifier plays a major role in dicitomzing the patterns when compared to other classifiers. Hence classifier 0.1 is given more weight.

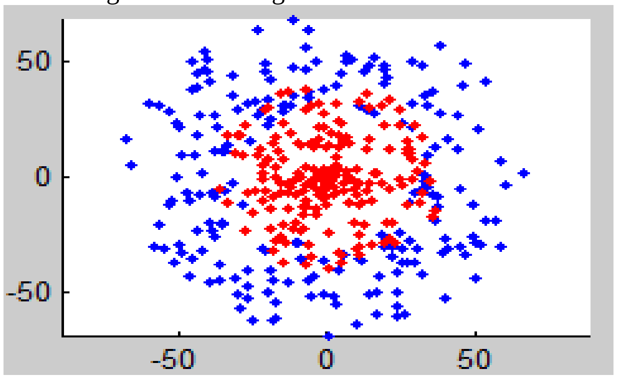

Fig 7.*Exploratory representation of training data*

180 images from Caltech dataset are used for testing the proposed system and the system is ran for 200 iterations and from the experimental analysis we could infer that after 50th iteration the training accuracy has been drastically and is shown in Fig, we could also infer that after 50 iteration the model loss is also abruptly lowered and thus we conclude that the proposed system starts intelligent learning after 50th iteration. The model accuracy and model loss while training and testing is shown in Fig 8 and 9 respectively. It is inferred that the loss function gets minimized after 110th epoch the loss in terms of mean square error. The performance of the Haar cascade classifier is compared with neural network and the section discusses the reason for choosing Haar cascade classifier to detect and discriminate human from non-human

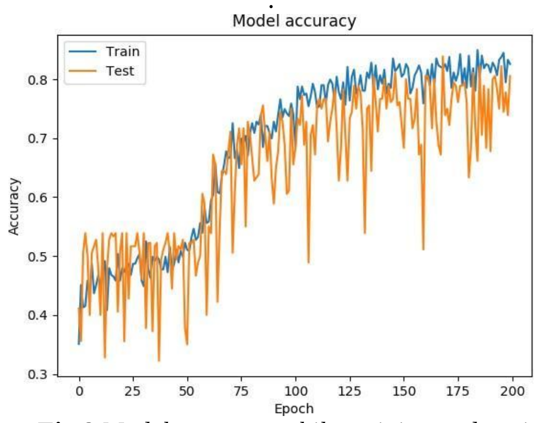

Fig 8.*Model accuracy while training and testing*

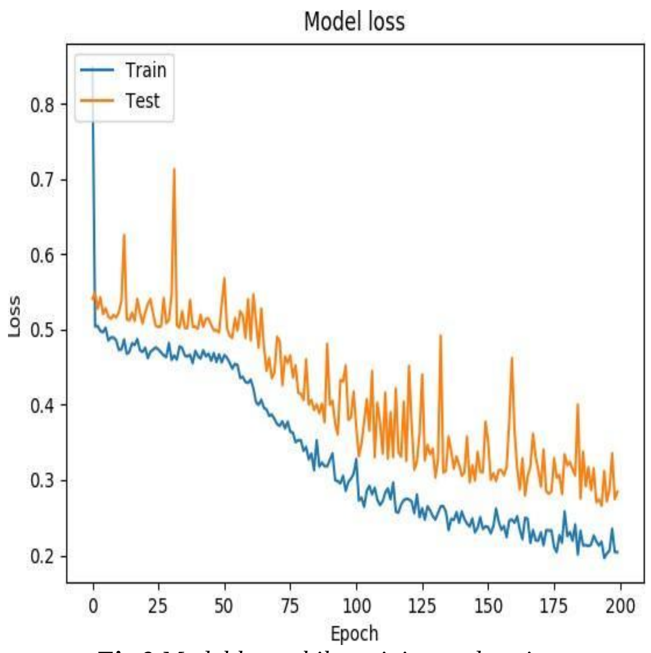

Fig 9.*Model loss while training and testing*

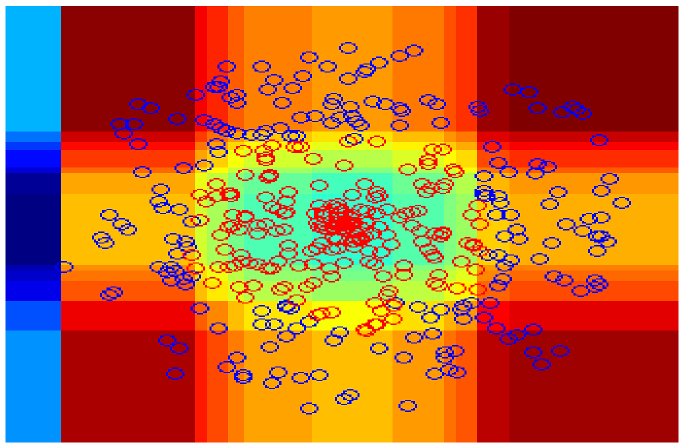

Fig 10.*Training data after classification*

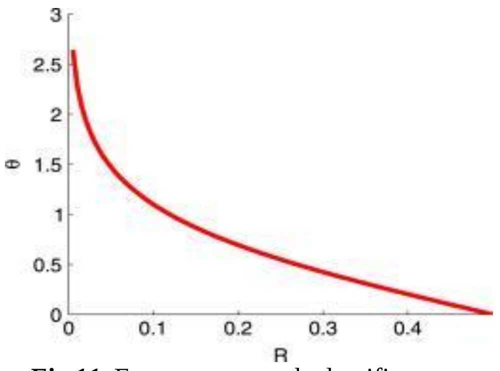

Fig 11. Error versus weak classifiers

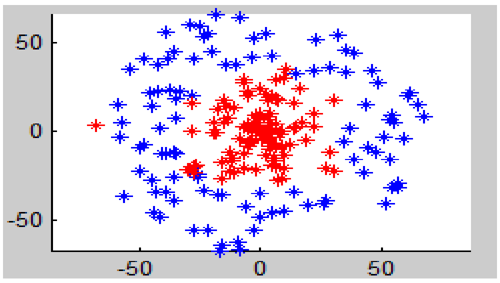

Fig 12. Test data classified by the algorithm

Haar cascade classifier is chosen when compared to other classifiers such as Neural Network [13] and Support Vector Machine [14] because Haar Cascade classifier is checks the micro level features from the region of interest using stacked weak classifiers where other classifiers takes the entire image as input and adjust weighs accordingly for minimize loss function [15]. A neural network with a sequential model is created and all the neurons are connected from one layer to another as shown in Fig 13. The feature vectors of humans from HOG feature extraction algorithm [16] is given as with dimension 8 are given as input to the hidden layer 1 and the weights are randomly initialized for eight neurons. In our work we initialize the bias to be zero and the weight at each layer follows as shown in equation 23

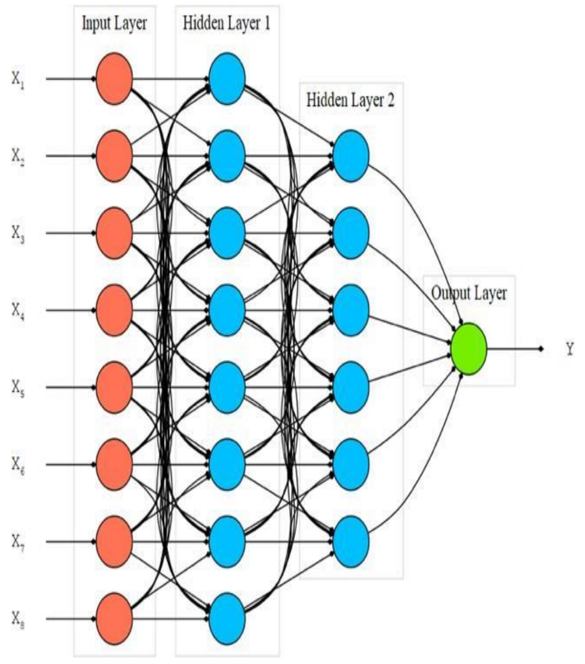

Fig 13. Architecture of the neural network

$$W_{ij} = \left[\frac{(-1)}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] = \left[\frac{(-1)}{\sqrt{8}}, \frac{1}{\sqrt{8}}\right] = [-0.08 + 0.08]$$

Instead of sigmoidal and Tan hyperbolic function Relu (Rectified Linear Units) [17] is being used because Relu ranges from 0 to x and for every weighted update the x gets updated and thus there is no space for vanishing and exploding gradient problem. As Relu is half rectified from the bottom the research fraternity uses it for model building. In this work the bias is initialized to zero. The created model is compiled using Stochastic Gradient Descent [18] learning algorithm with mean square error as loss function and with accuracy as metrics. The 8-dimension HOG feature vectors of both human and non-human images are taken for training. The initial weights to the neural network are initialized based on equation 23. The neural network has an architecture with a single input layer, two hidden layers and one output layer [19]. The first hidden layer has 8 neurons and the second hidden layer has 6 neurons and all the neurons in the network are densely connected. The learning rate is set as 0.01, which is one of the accepted rates in which weights are updated [20]. The model accuracy after training is shown in Fig 8 and we could infer that the system starts learning after the 110th epoch and there is no abrupt raise in the learning curve. The training accuracy of the neural network model built is 0.57 and the test accuracy of the model is 0.55 as shown in Fig 14 and 15.

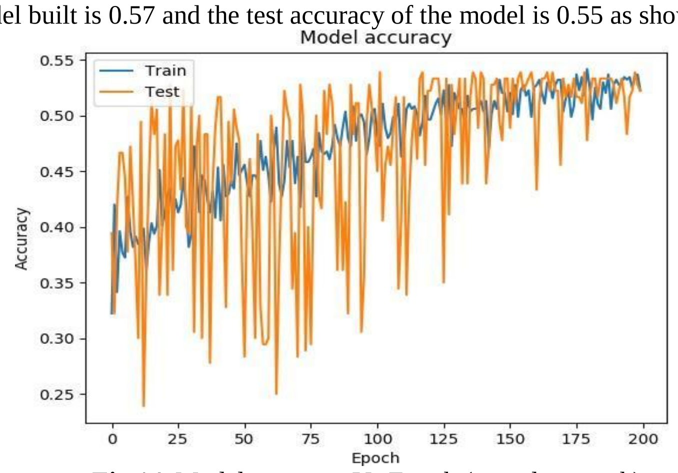

Fig 14. Model accuracy Vs Epoch (neural network)

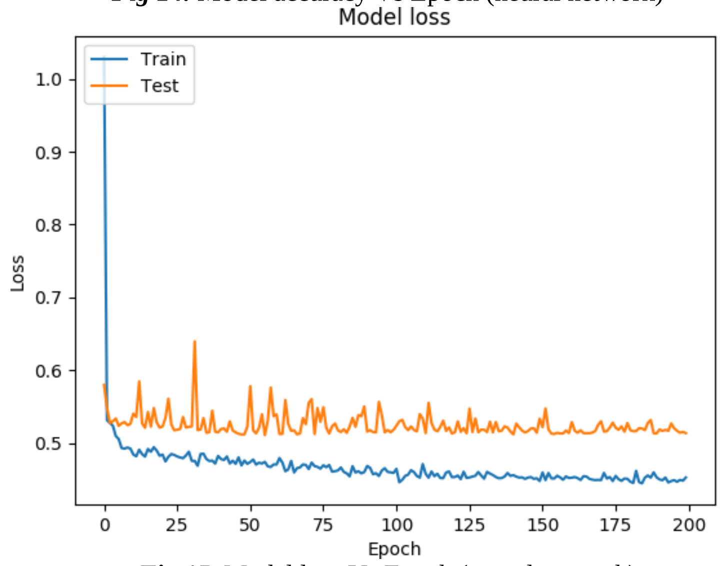

Fig 15. Model loss Vs Epoch (neural network)

Table 2.Accuracy of the proposed algorithm and Neural Network

Algorithm	True Positive	False Positive	Accuracy	False Discovery Rate	Time Complexity
MICTC powered by Haar cascade classifier	174	6	86	3.2	O(n)
Neural Network	148	32	55	18	O(n^2)

The accuracy of the proposed MICTC Algorithm is compared with the one of the universal function approximators, the neural network. From the experimental analysis we could infer that the proposed algorithm works with better Form experimental analysis: crowd image was shot and tested, the original count from the crowd is 382 and the predicted count is 384 and the results are shown in Fig 17. The working of the algorithm is tested initially in an open area where there is overcrowding and out of 28 people in the frame the algorithm gives the expected count 30 with social distancing negative as the distance between the centroids are less than 0.5. Another attempt is made to check the working of the system inside the deployment spot and here it is the bus with a single entry and exit point. Once a person enters the bus based on the direction the deployed system counts and de-counts the person inside the bus as shown in Fig 18, 19.

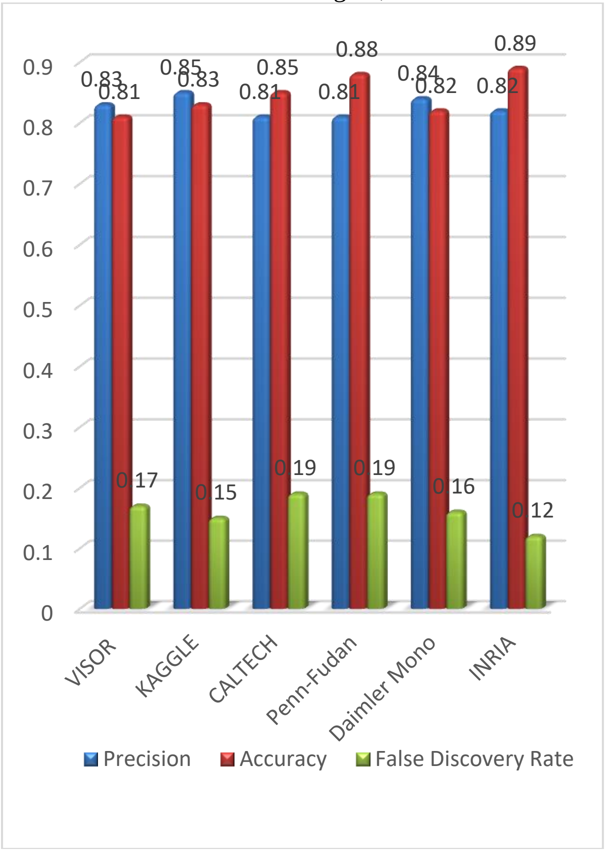

Fig 16. Performance Evaluation of the classifier

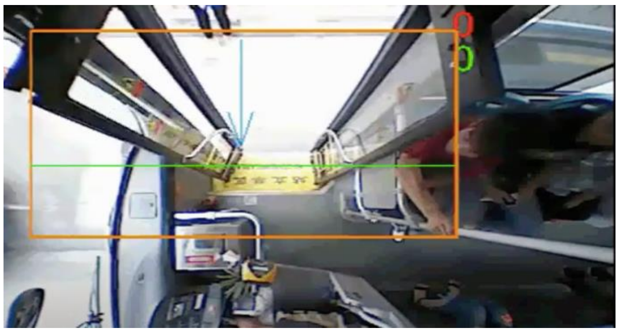

Fig 17. Testing the deployed system inside a single entry/exit passenger carrier

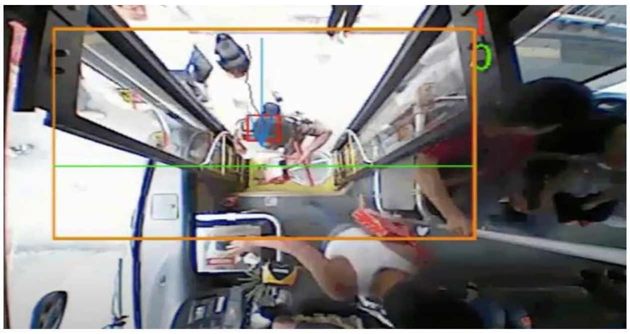

Fig 18. Testing the deployed system inside a single entry/exit passenger carrier

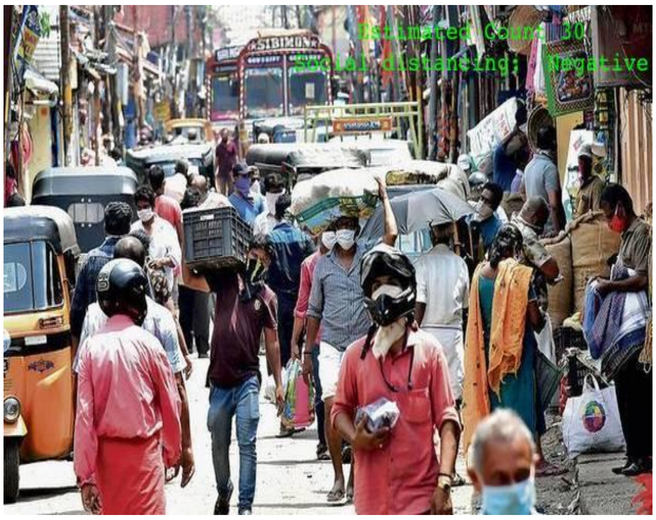

Fig 19. Testing the algorithm with overcrowded frame

The proposed algorithm is tested using varying pedestrian detection benchmark datasets such as VISOR [21], KAGGLE [22], CALTECH [12], Penn-Fudan[23], Daimler Mono [24], Indria [25] as shown in Fig 16. From the experimental analysis, we could infer that the system works with a precision of more than 0.8. We could also infer the performance drop using UCSD and KAISAT. On unravelling the reason behind we could infer that the dataset uses black and white images and is involved with complex occlusions.

4. Conclusion

The proposed system replaces human burden and automates the person count, passenger flow forecast and ensures social distancing. The performance of the system is tested with both benchmark datasets and manually captured image and video data on ground and from the performance evaluation we could infer that the proposed approach is up to the state of art. As a part of future work, we have planned to strengthened the feature set by adding infrared images and black and white images, so that the system is made robust to work in late evenings and night environments.

References

1. Gautam, K.S., Parameswaran, L. and Thangavel, S.K., 2020. Computer Vision Based Asset Surveillance for Smart Buildings. Journal of Computational and Theoretical Nanoscience, 17(1), pp.456-463.
2. Bertasius, G., Shi, J. and Torresani, L., 2015. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In Proceedings of the IEEE International Conference on Computer Vision (pp. 504-512).
3. Aliabad, F.A., Shojaei, S., Zare, M. and Ekhtesasi, M.R., 2019. Assessment of the fuzzy ARTMAP neural network method performance in geological mapping using satellite images and Boolean logic. International journal of environmental science and technology, 16(7), pp.3829-3838.
4. Lee, Y.J. and Grauman, K., 2015. Predicting important objects for egocentric video summarization. International Journal of Computer Vision, 114(1), pp.38-55.
5. Sun, Y., Sun, C., Wang, D., He, Y. and Lu, H., 2019. Roi pooled correlation filters for visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5783-5791).
6. Khandelwal, G., Anandi, V., Deepak, M.V., Prasad, V.N., Manikantan, K. and Francis, F., 2015, November. Pedestrian detection using single box convergence with iterative DCT based haar cascade detector and skin color segmentation. In 2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN) (pp. 32-37). IEEE.
7. Cadena, L., Zotin, A., Cadena, F., Korneeva, A. and Legalov, A., 2017. Noise reduction techniques for processing of medical images. In Proceedings of the World Congress on Engineering (Vol. 1, pp. 5-9).
8. K S Gautam, People_Counter, (2020), GitHub repository, https://github.com/Gautam-KS/People_Counter
9. Chollet, François. "keras." (2015).
10. Theano Development Team. "Theano: A Python framework for fast computation of mathematical expressions".
11. Bradski, G. and Kaehler, A., 2000. OpenCV. Dr. Dobb's journal of software tools, 3.

12. Griffin, G., Holub, A. and Perona, P., 2007. Caltech-256 object category dataset.
13. Gautam, K.S. and Kumar, T.S., 2016. Discrimination and detection of face and non-face using multilayer feedforward perceptron. In Proceedings of the International Conference on Soft Computing Systems (pp. 89-103). Springer, New Delhi.
14. Dadi, H.S. and Pillutla, G.M., 2016. Improved face recognition rate using HOG features and SVM classifier. IOSR Journal of Electronics and Communication Engineering, 11(4), pp.34-44.
15. Gautam, K.S. and Thangavel, S.K., 2019. Video analytics-based intelligent surveillance system for smart buildings. Soft Computing, 23(8), pp.2813-2837.
16. Singh, S. and Gupta, S.C., 2016, December. Human object detection by HoG, HoB, HoC and BO features. In 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 742-746). IEEE.
17. Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.
18. Reddi, S.J., Hefny, A., Sra, S., Poczos, B. and Smola, A.J., 2015. On variance reduction in stochastic gradient descent and its asynchronous variants. In Advances in neural information processing systems (pp. 2647-2655).
19. Jeff Heaton, (2017-06-01), The number of hidden layers, HeatonResearch,https://www.heatonresearch.com/2017/06/01/hidden-layers.html
20. K S Gautam, Senthil Kumar Thangavel, “Hidden Object Detection for Identification of threat” Vol 13, No ½, pp. 217-238,2020.
21. Vezzani, R. and Cucchiara, R., 2010. Video surveillance online repository (visor): an integrated framework. Multimedia Tools and Applications, 50(2), pp.359-380.
22. Kaggle, Pedestrian Dataset Labeled video, https://www.kaggle.com/smeschke/pedestrian-dataset
23. Penn-Fudan Database for Pedestrian Detection and Segmentation, https://www.cis.upenn.edu/~jshi/ped_html/
24. Enzweiler, M. and Gavrila, M., 2009. Daimler Mono Pedestrian Detection Bechmark Dataset. IEEE Trans. on Pattern Analysis and Machine Intelligence, pp.2179-2195.
25. Indria-Pedestrian, https://github.com/dbcollection/dbcollection/blob/master/docs/source/datasets/inria_ped.rst