Development of Automated People Counting System using Object Detection and Tracking

https://doi.org/10.3991/ijoe.v19i06.38515

Chee Jia Hong, Muhammad Hazli Mazlan(✉)
Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia
mhazli@uthm.edu.my

Abstract—The emergence of automation in the current economic trend promotes the usage of computer vision systems in various applications. Counting people in a specified area or on the street can bring many benefits in terms of security and marketing. The people counting system is one of the applications that utilize the computer vision system to count people with higher reliability and accuracy. Thus, this project is to develop an offline automated people counting system based on captured video file input using MATLAB software and a notification system to update and send notifications about the number of occupants in a target area using ThingSpeak. For project development, simulation and development of coding for object detection that involves deep learning approach, object tracking and counting, and development of notification system have been done. Three videos were taken to be used for three trials to evaluate the functionality and performance of the developed system. Based on the results and analysis, the system can perform people detection, people tracking and people counting on the recorded input videos with high accuracy of 94.45%, visualize the data on the ThingSpeak platform and send notifications through Twitter.

Keywords—people counting, MATLAB, ThingSpeak, deep learning, object detection, object tracking, Mask R-CNN

1 Introduction

Nowadays, computer vision is introduced to duplicate human vision abilities to understand and process digital images and video [1]. Image and video processing algorithms have recently become more mature with lots of new technologies and their usage has been growing rapidly. This causes the emergence of the necessity for the development of tools to analyze digital imagery content [2]. People counting is also one of the common applications of computer vision systems [3]. A people counter is an electronic device that counts and monitors people entering a building or estimates the total number of people within a specific area [4]. The system is widely applied at the entrance of a building to record the total number of visitors. Besides, people counting systems are widely used and beneficial for security, tracking, and marketing research. The crowd population density can also be known using the system [5].

A vision-based system that can automatically count people has been widely used for people counting systems due to its reliability and accuracy [3] [4]. However [7] [8], with current technologies, there are still many challenges that cause counting people using a vision system to be a difficult task. For example, the main challenges include people occlusion, which means difficult to distinguish or track people who are too close to each other [6]. In some cases, the system also considers moving non-human objects as a count. Other challenges include background clutter, environment illumination, and the angle of the camera [3] [5]. Besides, the existing commercialized people counting systems using video cameras by various companies are very accurate and reliable but expensive [4]. Moreover, it is noticed that the existing counting systems mostly do not operate with a notification system that can update the number of occupants counted in an area.

Therefore, this project aims to develop an automated offline people counting system based on captured video file input using MATLAB software. The system will also be equipped with a notification system that can display the counted people and notify the user about the number of counts. The performance of the system will be analyzed by comparing it with a manual. The system should be able to perform image processing, object detection using deep learning algorithms, object tracking, and object counting automatically to process the input from a video camera.

2 Literature review

A people counting system can be defined as a system that measures how many people pass through a particular path, entry, or door. It is primarily used to record the total number of visitors to a specific location, typically at the entrance [1]. There are various types of people counting systems, including contact-type systems, sensor-type systems, and vision-based systems [3]. With the rapid development of technology, vision-based systems that can provide better results have become increasingly appealing. Therefore, this review will focus on the most cutting-edge approaches and previous related work related to vision-based systems. To count people, M. Cruz [9] applied deep learning algorithms such as You Only Look Once (YOLOv3) and DeepSORT for object detection and classification and used the Kalman filter for object tracking, using Python and OpenCV. The YOLOv3 algorithm is frequently used by M. Ahmad [11] and I. Ahmed [14] also used it to detect objects. In I. Ahmed's work [13], new algorithms such as SiamMask and Faster R-CNN were used for better object detection performance. In addition, M. Ahmad [15] used Generic Object Tracking Using Regression Networks (GOTURN) tracking algorithm to track people in videos and the Faster R-CNN algorithm for object detection. For object detection and classification, I. Ahmed [18] used machine learning algorithms such as Rotated Histogram of Gradient (HOG) and Support Vector Machine (SVM). Table 1 summarizes the most relevant research on the previous related work.

Table 1. Summary of related previous work

Ref No.	Algorithm/ Techniques	Software/ Hardware Platform	Performance Evaluation	Year
			Accuracy	
[9]	YOLOv3, DeepSORT, Kalman filter, Bounding Box Centroid, Threshold	Python, OpenCV	82.76%	2020
[10]	Morphological operation, LBP-Based Adaboost classifier, Mean Shift Algorithm	–	96.8%	2020
[11]	YOLOv3	OpenCV	95%	2019
[12]	Single Shot Detector (SSD)	–	95%	2019
[13]	SiamMask algorithm	Python	95%	2021
	Kernelized correlation filter (KCF)		80%	
	Median Flow		80%	
[14]	YOLOv3, Deep SORT	Python, OpenCV	96%	2021
[15]	Faster-RCNN, GOTURN tracking	OpenCV	94%	2020
[16]	Background subtraction, Frame differencing, Connected-component labelling	OpenCV	87%	2016
[17]	EagleSense system, Top-View Depth-Sensing	–	>90%	2017
[18]	Rotated HOG, SVM	OpenCV	94%	2019

3 Methodology

The section will present the methodology that describes the overall procedures for the research from the beginning to the end in order to achieve the objectives of the project. The subsection below describes the details of designing and developing the proposed system by using a block diagram and flowchart.

3.1 Flowchart of the developed system

Figure 1 shows the flowchart of the proposed system. First, the system is reset, and all existing variables are cleared to prevent them from affecting the results. Then, the MATLAB platform is connected to ThingSpeak to exchange information. The recorded video footage is then fed into the system, and system objects are created and set up to facilitate the coding process for video frame reading, object detection, and object tracking. Before the object tracking process starts, an empty array is created to be filled with the tracking values obtained later. After the setup is completed, the system reads the input video frames and checks whether they exist or not. If the frame does not exist, the process ends. If the input video frames are detected, the system proceeds to the pre-processing, object detection, object tracking, counting, and results-displaying processes. A loop is created in the code to detect whether or not the video frames are detected. Finally, the number of people counted in the sample frames

is sent to ThingSpeak and displayed. If the number of people counted is less than 5, the notification sent is about the current people count at the respective time. If the number is more than 5, an alert message is sent with the respective time. The system assumes that the area is crowded if the number of people is more than 5.

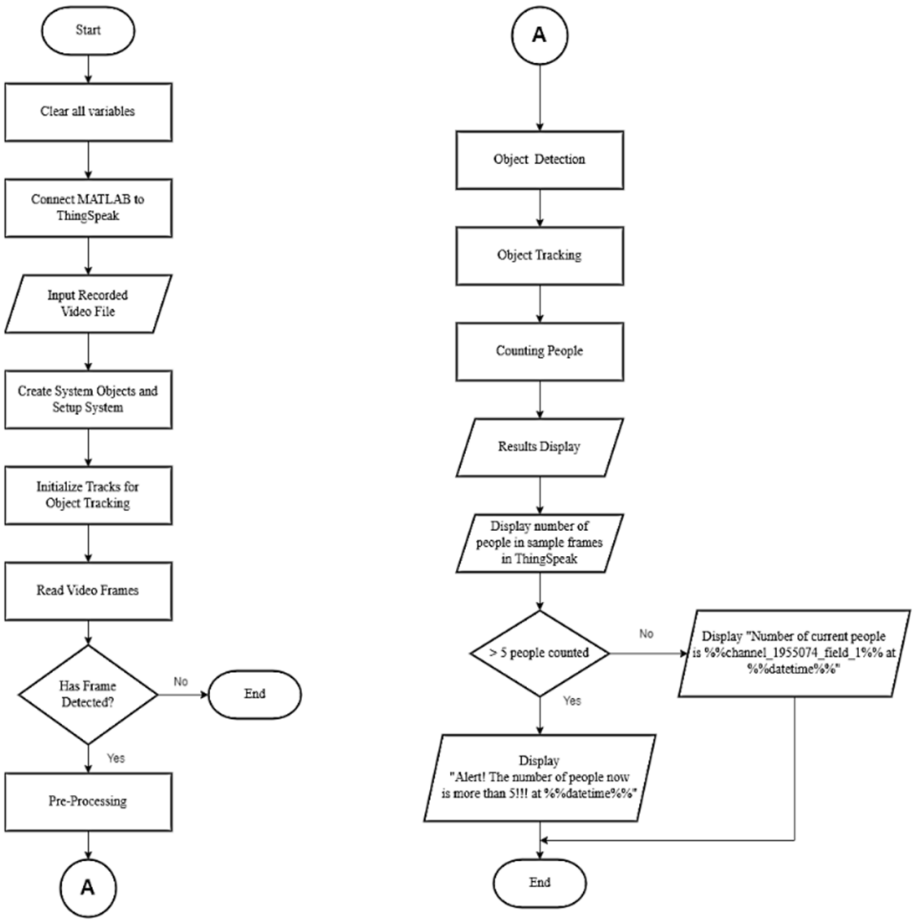

Fig. 1. Flowchart of the system

3.2 Block diagram of the developed system

The block diagram of the proposed system is shown in Figure 5. It provides more detailed information on how the system works by outlining the methods and steps involved in the main processes, which include pre-processing, object detection, object tracking, counting, and result display.

For pre-processing, the system acquired and read video frames. Additionally, the frame rate of the video was increased to reduce processing time, and the video quality was improved to obtain better processing results. The code used to create objects to

read video and write video files to adjust the frame rate and quality of the video is shown in Figure 2.

```
TA.Reader = VideoReader(InputVideo);
TA.Reader2 = VideoWriter(InputVideo);
TA.Reader2.FrameRate = 90;
TA.Reader2.Quality = 95;
```

Fig. 2. Coding to read and write video data from the input video

The system used several methods to perform object detection, including instance segmentation using Mask R-CNN, background subtraction using Gaussian Mixture Models (GMM), blob analysis, and centroid setting. The code used to declare the system objects and execute these tasks is shown in Figure 3. Before instance segmentation is performed, the Mask R-CNN object detector must be loaded using the 'maskrcnn' function. A pre-trained Mask R-CNN object detector trained on the MS-COCO dataset with a ResNet-50 network as the feature extractor was used in this project. The loaded detector was passed to the 'segmentObjects' function to detect objects in an image, which segments individual instances of objects using the detector. The function returns useful information about the frames, including the detected object masks, the labels assigned to the detected objects, the scores for each of the detected objects, and the locations of the segmented objects, also known as bounding boxes. A detection threshold was set to reduce false positives to determine the return of detections with scores less than the threshold value. Additionally, hardware resources were specified to reduce the processing time by setting the 'ExecutionEnvironment' to use a GPU (Graphing Processing Unit). The video frames were resized to [800 1200 3] to input into the 'segmentObjects' function. The 'classNames' was specified as the names of the object classes that the detector is trained to detect instead of using all object classes.

The next method used was background subtraction using a Gaussian mixture model. The system object was created, its properties were set, and the foreground detector function was used to perform this. By comparing each video frame to the background model at the pixel level, the function determines whether the pixels are part of the background or foreground. It then computes a foreground mask using a Gaussian mixture model and returns a binary mask as output. The desired results were achieved by setting the number of Gaussian modes in the mixture model, the number of initial frames used to train the background model, and the threshold used to determine the background model. The binary mask obtained from the foreground detector was used as input for blob analysis. This was done by creating the system object, setting its properties, and using its MATLAB function, as shown in Figure 3. The function obtains statistics, including centroid coordinates and bounding box coordinates, for groups of connected pixels in the binary image. The minimum blob area in pixels was set using 'MinimumBlobArea' to determine which blobs to retain. The output coordinates returned by the function were then used for object tracking.

```
% Create System objects for people detection
TA.DL_detector = maskrcnn("resnet50-coco");

% Create System objects for foreground detection
TA.Detector = vision.ForegroundDetector('NumGaussians', 3, ...
  'NumTrainingFrames', 40, 'MinimumBackgroundRatio', 0.7);

% Create System objects for blob analysis
TA.BlobAnalyzer = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', true, 'CentroidOutputPort', true, ...
    'MinimumBlobArea', 400);
function [centroids, bboxes, masks, labels, scores] = detectPeople(videoFrame)
  % Run system algorithm to detect people
  I = imresize(videoFrame,'OutputSize',[800 1200]);
  classNames = "person";
  [masks,labels,scores,boxes] = segmentObjects(TA.DL_detector,I,Threshold=0.92,...
      ExecutionEnvironment="gpu");

  % Run system algorithm to detect foreground
  bmask = TA.Detector.step(videoFrame);

  % Perform blob analysis to find connected components
  [~,centroids,bboxes] = TA.BlobAnalyzer.step(bmask);
end
```

Fig. 3. Coding for object detection

The object tracking process involved the use of the Kalman filter and track maintenance algorithm. The code for executing these tasks is presented in Figure 4. The Kalman filter was used to predict the location of each track in the video frame and assign detections to each track. Meanwhile, the track maintenance algorithm was responsible for updating assigned and unassigned tracks, deleting lost tracks, and creating new tracks for unassigned detections. The Kalman filter is used to predict the location of each track and assign detections to the appropriate track to estimate the motion of each track. The filter predicted the new location of an existing track by updating its centroid location and bounding box coordinates. The cost of assigning a detection to each track was computed using the Euclidean distance between the predicted centroid of the track and the centroid of the detection, as well as the confidence of the prediction. The 'assignDetectionsToTracks' function used the cost matrix and the cost of non-assignment to determine the indices of assigned and unassigned tracks and detections, respectively.

The filter was worked in conjunction with the track maintenance algorithm. After the detections were assigned to tracks in a video frame, the corresponding tracks were updated with the assigned detections. However, some detections or tracks may remain unassigned, in which case they were marked as invisible, and a new track will begin for those detections. Additionally, the number of consecutive frames for which the track remains unassigned was counted to determine whether the object has left the field of view. The object's track was considered lost and deleted if it exceeded the set threshold.

```
while hasFrame(TA.Reader)
    videoFrame = readFrame(TA.Reader);
    [centroids, bboxes, masks, labels, scores] = detectPeople(videoFrame);
    TracksNewLocationsPrediction();
    [assignments, unassignedTracks, unassignedDetections] = AssignDetectionToTrack();

    UpdateAssignedTracks();
    UpdateUnassignedTracks();
    DeleteLostTracks();
    CreateNewTracks();
end
```

Fig. 4. Coding to call each subroutine functions in a loop

Lastly, counting was done by accessing the array that stores the results of bounding boxes and reading the size of the bounding boxes at respective frames.

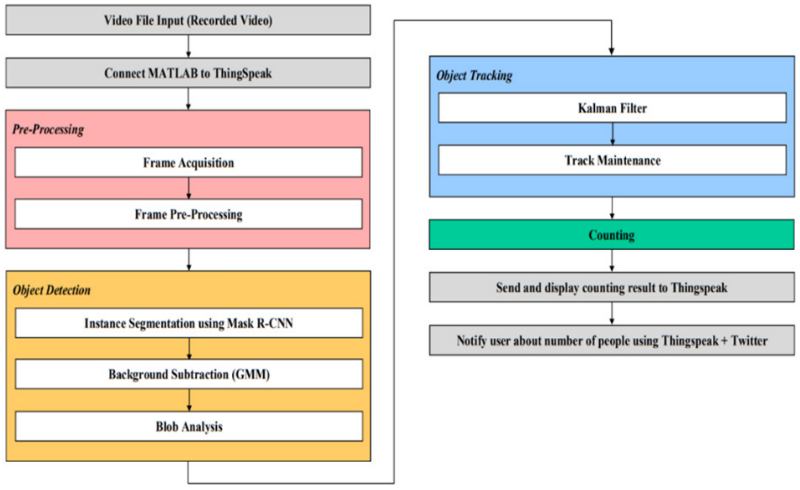

Fig. 5. Block diagram of the system

3.3 Performance evaluation

This project applied a new deep-learning approach to detect and count people moving in the video. This section presents the equations and methods to calculate and evaluate the proposed system's performance. To assess the performance, metrics evaluation and analysis were done. Besides, the main findings of this paper were compared with the other methods from previous works [9]–[21]. The accuracy and percentage error was calculated by [19]:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{1}$$

$$Error\ Rate = \frac{Number\ of\ wrong\ predictions}{Total\ number\ of\ predictions} \tag{2}$$

4 Results and discussion

In this section, the results of the developed system based on the methodology described in the previous chapter will be discussed. All the progress and findings that have been made, as well as analysis, will be stated.

4.1 Results on input video footage in MATLAB

The results of the system were obtained by testing it in three different trials. The first and second trials were conducted under simple conditions, while the third trial was conducted under more complicated conditions. Figures 6–8 illustrate the results of the trials. In the first trial, the system detected and counted people when they appeared in the frames but did not count when there was no person. The second trial showed that the system could detect multiple objects with partial occlusion but failed to detect fully occluded people. The third trial, which was conducted in complex backgrounds, showed that the system could not accurately detect and track people. To evaluate the performance of the system, the simulation counting results were compared to manual counting results. The system counting accuracy for each trial was found to be 100%, 97.62%, and 85.73%, respectively. These results indicate that the system performed well under simple conditions, but its performance decreased when the conditions became more complex.

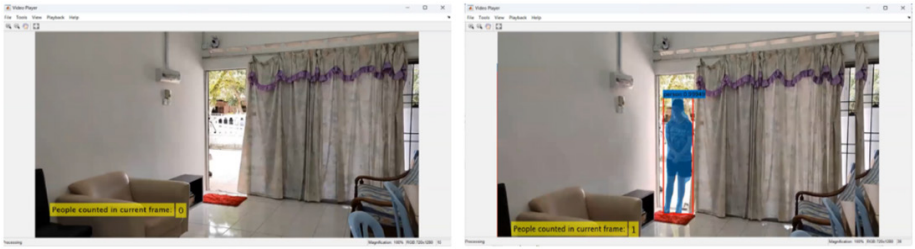

Fig. 6. Examples of trial 1 results on MATLAB

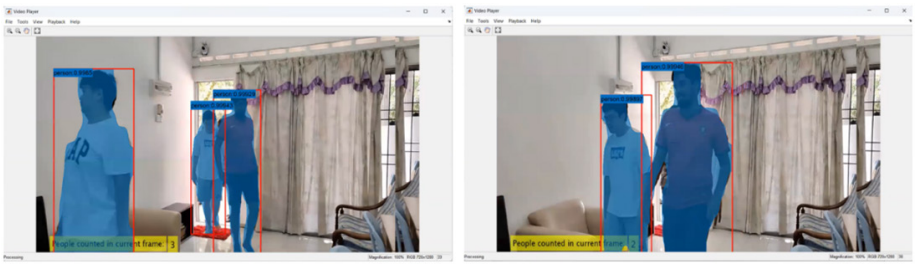

Fig. 7. Examples of trial 2 results on MATLAB

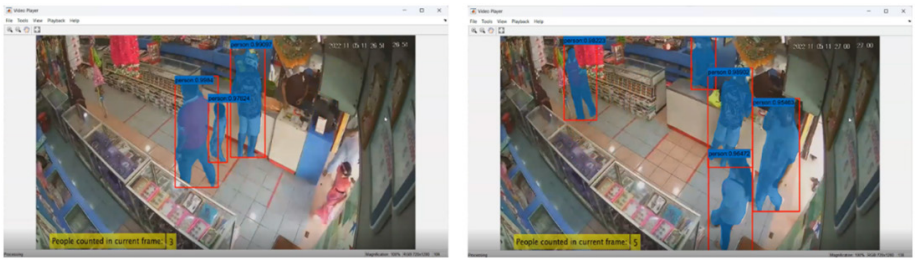

Fig. 8. Examples of trial 3 results on MATLAB

4.2 Results on the ThingSpeak platform and Twitter

In this subsection, the notification system in the ThingSpeak platform was tested to store and visualize the counting results in a graph for each trial. Besides, it was tested to send a notification to users through Twitter based on the condition set. The results for each trial are shown in Figure 9.

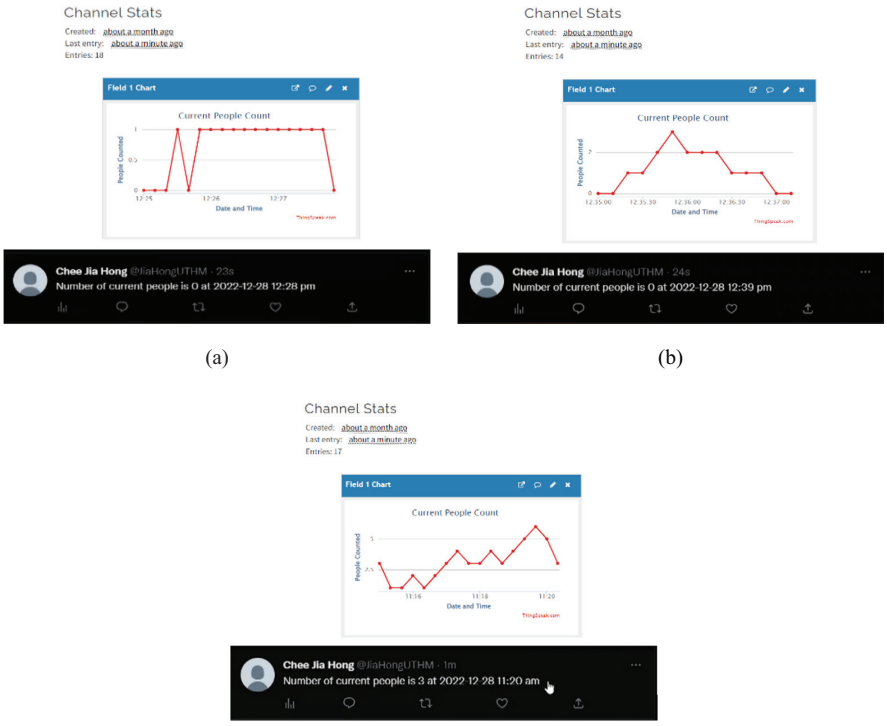

Fig. 9. Notification system results (a) Trial 1; (b) Trial 2; (c) Trial 3

4.3 Discussion

In this subsection, the discussion on the system's performance in detecting and counting people that passed through the monitoring area in video frames will be done based on the results of the three trials. From Table 2, the system accuracy and percentage error for each trial were tabulated and overall accuracy and percentage error were calculated.

Table 2. Overall system accuracy and percentage error

Trial	Accuracy (%)	Percentage Error (%)
1	100	0
2	97.62	2.38
3	85.73	14.27
Overall	94.45	5.55

Based on the results of the first trial, when the system is tested with the simplest condition, which is people enter and leave the field of view one by one with a simple background, the system can detect people in the video frames accurately and in good condition. This can be proved by the result in Table 2, which shows the system accuracy to be 100% in the first trial. The result also shows that the system does not detect anything non-human objects in the scene as people when there is no person. Besides, it works well to detect and count people correctly when they enter and leave the view. Moreover, the first trial results also show that the displaying and labelling of detected objects in the frames are working in good condition.

Based on the results of the second trial, when the system is tested with a more complex condition where the people enter and leave the field of view together in the simple background, the system is proved to be able to detect and count multiple people in the scene in high accuracy of 97.62% by referring to Table 1. Besides, the system can detect people that appear in partially occluded condition. However, the system cannot detect the person that appears in a fully occluded condition. This can be the reason of the accuracy of the second trial is lower than the first trial.

Based on the results of the third trial, when the system is tested with a complex condition where the people enter and leave the field of view together in complex background, the system is proven to be able to detect and count multiple people in the scene in lower accuracy of 85.73% by referring to Table 2. In this trial, as same as the results in the second trial the system can detect and count people well in partially occluded conditions but cannot work well in fully occluded conditions. However, for several detected people that are partially occluded in complex conditions, the results show that the system may have errors in detecting and counting the people. This can be the reason why the accuracy of the third trial is lower than the previous trials.

Overall, from Table 1, the system is shown to have an overall accuracy of 94.45% which is considered high accuracy. It shows the system can work perfectly in simple conditions and simple backgrounds and can work with high accuracy in complex conditions and complex backgrounds. Besides, the results on the ThingSpeak platform and Twitter show that the notification system also works well.

5 Conclusion

In conclusion, an offline automated people counting system was developed using MATLAB for this project. Image and video processing techniques have been used to develop the people counting system, including pre-processing, object detection, and object tracking. For object detection, the main technique applied to increase the system accuracy is instance segmentation using Mask R-CNN. The developed system can process the input recorded videos, count people that appear in video frames, send data to the cloud medium, ThingSpeak and send notifications through Twitter. The MATLAB simulation results were compared to other existing related works to ensure the accuracy of the results. Based on the results and analysis, the accuracy of the developed system was tested and evaluated. It was also proved to be operating at high accuracy, which was 94.45% which was quite successful through comparison. However, several limitations affect the accuracy and functionality of the system. Further projects can be done using other software like Python and OpenCV to overcome the limitations of the system. Besides, to improve the system's accuracy, other object detection and tracking algorithms can be applied to develop a similar system. The system should also be improved to operate in real-time and accurately count people entering and leaving an area.

6 Acknowledgement

Communication of this research is made possible through monetary assistance by Universiti Tun Hussein Onn Malaysia and the UTHM Publisher's Office via Publication Fund E15216.

7 References

[1] A. Murat Tekalp, Digital Video Processing, 2nd ed., New York: Prentice Hall, 2015.

[2] M.S. Sruthi, "IOT Based Real Time People Counting System for Smart Buildings," International Journal of Emerging Technology and Innovative Engineering, vol. 5, no. 2, pp. 83–86, 2019.

[3] Chakravartula Raghavachari, V. Aparna, S. Chithira and Vidhya Balasubramanian, "A Comparative Study of Vision based Human Detection Techniques in People Counting Applications," Procedia Computer Science, pp. 461–169, 2015. https://doi.org/10.1016/j.procs.2015.08.064

[4] Damien LEFLOCH, "Real-Time People Counting System using Video Camera," Gjøvik University College, Norway, 2007. https://doi.org/10.1117/12.766499

[5] Md Israfil Ansari and Shim Jaechang, "People Counting System using Raspberry Pi," Journal of Multimedia Information System, vol. 4, no. 4, pp. 239–242, 2017.

[6] Mohammad Ali A. Hammoudeh, Mohammad Alsaykhan, Ryan Alsalameh and Nahs Althwaibi, "Computer Vision: A Review of Detecting Objects in Videos – Challenges and Techniques," International Journal of Online and Biomedical Engineering, vol. 18, no. 01, pp. 15–27, 2022. https://doi.org/10.3991/ijoe.v18i01.27577

[7] Brahim Jabir, Noureddine Falih and Khalid Rahmani, "Accuracy and Efficiency Comparison of Object Detection Open-Source Models," International Journal of Online and Biomedical Engineering, vol. 17, no. 05, pp. 165–184, 2021. https://doi.org/10.3991/ijoe.v17i05.21833

[8] Evelyn Santana Mantuano, Washington Xavier García-Quilachamin and Jorge Anchundia Santana, "A Systematic Review of Algorithms in People Images Detection Based on Artificial Vision Techniques for Energy Management in Air Conditioners," International Journal of Online and Biomedical Engineering (iJOE), vol. 17, no. 01, pp. 17–33, 2021. https://doi.org/10.3991/ijoe.v17i01.17899

[9] Meygen Cruz, Jefferson James Keh, Ramiel Deticio, Carl Vincent Tan, John Anthony Jose and Elmer Dadios, "A People Counting System for use in CCTV Cameras in Retail," in 2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Manila, Philippines, 2020. https://doi.org/10.1109/HNICEM51456.2020.9400048

[10] Vinay Kaithwas and Lokesh Parashar, "A People Counting Method Based on Machine Learning," International Journal of Science, Engineering and Technology, vol. 8, no. 5, pp. 1–5, 2020.

[11] Misbah Ahmad, Imran Ahmed and Awais Adnan, "Overhead View Person Detection Using YOLO," in IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, 2019. https://doi.org/10.1109/UEMCON47517.2019.8992980

[12] Misbah Ahmad, Imran Ahmed, Kaleem Ullah and Maaz A, "A Deep Neural Network Approach for Top View People Detection and Counting," in IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, 2019. https://doi.org/10.1109/UEMCON47517.2019.8993109

[13] Imran Ahmed and Gwanggil Jeon, "A Real Time Person Tracking System Based on SiamMask Network for Intelligent Video Surveillance," Journal of Real-Time Image Processing, vol. 18, pp. 1803–1814, 2021. https://doi.org/10.1007/s11554-021-01144-5

[14] Imran Ahmed, Misbah Ahmad, Awais Ahmad and Gwanggil Jeon, "Top View Multiple People Tracking by Detection using Deep SORT and YOLOv3 with Transfer Learning: Within 5G Infrastructure," International Journal of Machine Learning and Cybernetics, vol. 12, pp. 3053–3067, 2021. https://doi.org/10.1007/s13042-020-01220-5

[15] Misbah Ahmad, Imran Ahmed, Fakhri Alam Khan, Fawad Qayum and Hanan Aljuaid, "Convolutional Neural Network–based Person Tracking using Overhead Views," International Journal of Distributed Sensor Networks, vol. 16, no. 6, 2020. https://doi.org/10.1177/1550147720934738

[16] Jau-Woei Perng, Ting-Yen Wang, Ya-Wen Hsu and Bing-Fei Wu, "The Design and Implementation of a Vision-based People Counting System in Buses," in International Conference on System Science and Engineering (ICSSE), Taiwan, 2016. https://doi.org/10.1109/ICSSE.2016.7551620

[17] Chi-Jui Wu, Steven Houben and Nicolai Marquardt, "EagleSense: Tracking People and Devices in Interactive Spaces using Real-Time Top-View Depth-Sensing," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, 2017.

[18] Imran Ahmed, Misbah Ahmad, Awais Adnan, Awais Ahmad and Murad Khan, "Person Detector for Diferent Overhead Views using Machine Learning," International Journal of Machine Learning and Cybernetics, vol. 10, pp. 2657–2668, 2019. https://doi.org/10.1007/s13042-019-00950-5

[19] Pang-Ning Tan, Michael Steinbach and Vipin Kumar, Introduction to Data Mining, United States: Pearson Education Limited, 2014.
[20] M Al-Nawashi, OM Al-Hazaimeh and Mohamad Saraee, "Geometrical-based Approach for Robust Human Image Detection," Multimedia Tools and Applications, 2019. https://doi.org/10.1007/s11042-018-6401-y
[21] Malek Al-Nawashi, OM Al-Hazaimeh and Mohamad Saraee, "A Novel Framework for Intelligent Surveillance System Based on Abnormal Human Activity Detection in Academic Environments," Neural Computing and Applications, vol. 28, no. Suppl 1, pp. 565–572, 2017. https://doi.org/10.1007/s00521-016-2363-z

8 Authors

Chee Jia Hong is a Bachelor's degree student at the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Johor, Malaysia (email: de190133@student.uthm.edu.my).

Muhammad Hazli Mazlan is a Senior Lecturer and Principle Researcher at the Faculty of Electrical and Electronic Engineering Universiti Tun Hussein Onn Malaysia. He is currently working in biomedical engineering research areas specializing in orthopaedic implants and image processing analysis (email: mhazli@uthm.edu.my).

Article submitted 2023-01-18. Resubmitted 2023-02-21. Final acceptance 2023-02-24. Final version published as submitted by the authors.