Top View Person Detection and Counting for Low Compute Embedded Platforms

Prashant Maheshwari
Capillary Technologies
prashant.m
@capillarytech.com

Doney Alex
Capillary Technologies
doney.alex
@capillarytech.com

Sumandeep Banerjee
Capillary Technologies
sumandeep.banerjee
@capillarytech.com

Saurav Behera
Capillary Technologies
saurav.behera
@capillarytech.com

Subrat Panda
Capillary Technologies
subrat.panda
@capillarytech.com

ABSTRACT

In this paper, we present an optimised approach for the top-view person detection and counting for low compute embedded platform. Earlier methods have used background subtraction for top-view detection which produces inaccurate results due to merging of blobs when there are multiple people in the frame. Several Deep Learning methods have been proposed for detection and tracking but they are computationally expensive to run on low compute device. We present an adaboost classifier for top-view detection and Kalman filter-Hungarian assignment for tracking which is optimised to give high accuracy on a low compute embedded platform. We achieved a counting accuracy of 97% in real-time (upto 40 FPS) on a Raspberry Pi3B. We also present a heuristic approach to handle false positives in real time through dynamic learning and unlearning of detections along with other optimisations in tracking.

CCS Concepts

•Computing methodologies→Computer vision problems Tracking • Computer systems organization→Real-time systems.

Keywords

Tracking, Real-time systems, Object detection, Multithreading

1. INTRODUCTION

People counting in various scenarios has drawn a lot of attention due to ever increasing use cases such as - surveillance, estimating the number of people in a large capacity building, business performance of any retail store through number of people visiting store. They rely on the extracted information from the detections and tracking - like the path traversed or the exact number of people crossing an area. Most of the research work has approached person detection using fronto-parallel or front view of person. However, it makes people counting a challenging task largely due to occlusion and depth perception which makes both detection and tracking inaccurate and difficult. There are numerous deep learning based approaches that have surfaced in recent times achieving state-of-the-art accuracy in both detection and tracking. These approaches are computationally expensive and requires specialised hardware to obtain results in real time. The top-view gives the advantage of almost no occlusion but the features available for detection are less compared to pedestrian or front-view and it also has perspective distortion. Therefore, detection in top-view is comparatively more challenging. The methods of segmenting foreground from background has been used for person detection in top-view [3] but it is susceptible to illumination changes, shadows and reflections. The reflections and shadows can be handled with the work by [19] and [39]. But, accurately splitting the merged blobs when there are multiple people in the frame in exact people count is a daunting task. Using stereo vision helps in splitting the blob more accurately [22], [28] but that requires extra hardware.

The people counting system has more impact if it is real-time and cost effective when it comes to mass installation. Therefore, in this paper we present a fast and resource efficient people counting algorithm for top-view monitored area which can run on low compute inexpensive embedded systems. The overhead mounted camera monitors a small area of few sq.ft, similar to that of office or shop entrances like in fig:1. To achieve reasonable accuracy and high detection FPS we overcome the following challenges : 1) to avoid occlusion we use top-view camera installation instead of pedestrian view. This slightly offloads the burden of accurate detection from our top-view detector. 2) We use a feature based top-view person detector based on Adaboost classifier which is robust to illumination and lighting changes and perspective distortion unlike background subtraction. BG-sub requires splitting blobs of people in close proximity, either heuristically which causes error in person count, or by estimating person size using stereo vision which requires extra hardware. Our detector does not have the above constraints. 3) We use Kalman Filter along with Hungarian assignment for tracking and we are also proposing numerous heuristic optimisations for better tracking, including dynamic handling of false positives through learning-unlearning of detections. 4) Lastly, we have used a multithreaded approach for our detection and tracking to achieve 40 FPS on Raspberry Pi. The paper is organized as follows: Section 2 review different solutions proposed in literature. Section 3 Proposed solution and Section 4 Experiments and Results.

ICVIP 2018, December 29–31, 2018, Hong Kong, Hong Kong
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6613-7/18/12…$15.00
https://doi.org/10.1145/3301506.3301548

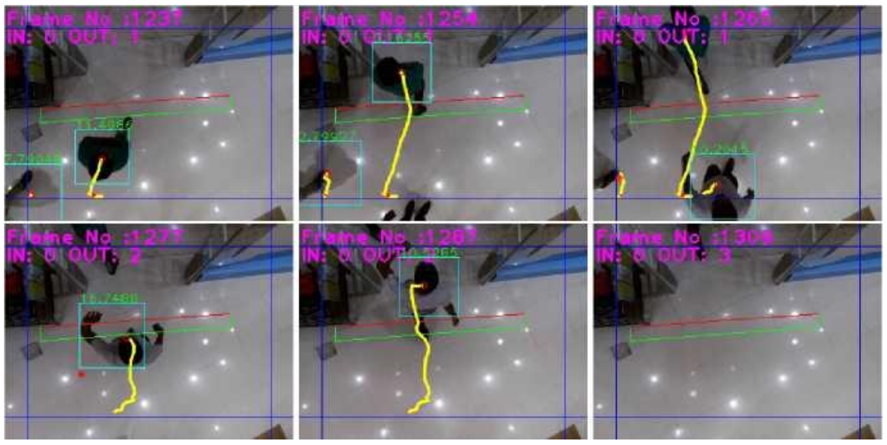

Figure 1. Fig a-f shows the detection, tracking and counting

2. RELATED WORK

Many people counting algorithms have been proposed and these approaches primarily vary with respect to the following factors: (i) Methods —detection or estimation, (ii) Person view —fronto-parallel or top view, (iii) Technology —Deep learning or machine learning. There are numerous people counting algorithms and approaches varying in terms of method, person view - front view or top view, and technology (deep learning or pure machine learning). A recent work that addresses our problem is people counting in dense crowd scenes [42], [20], [31]. In earlier approaches of pedestrian counting, person detection played an important role. They performed well as the numerous neural network architectures like [33], [34], [26], gives best of the accuracy for pedestrian detection even when occlusion occurs. But building a real time people counting system using neural networks is not feasible due to their slow performance on a low compute embedded platform such as Raspberry Pi 3B. Table 1 lists the time benchmarking of different architectures. These networks achieve less than 1 FPS which makes them unsuitable for tracking and hence for counting. However, there are other compute efficient algorithms for pedestrian detection which give good accuracy and speed. [9] and number of its modifications [44] perform well on pedestrian detection. Another notable work is by P Dollar [11], [10]. There also exist numerous people counting method based on pedestrian detection [40], [6], [25] and [4]. These methods are not very accurate as they fail to handle occlusion.

Previous work in top-view person detection [3], [36], and [24] have use background subtraction which did not produce satisfactory results as it was highly susceptible to illumination condition, change in lighting, and shadows. The variation in illumination changes creates a difference between the background pixel values and the learned background pixel values, therefore background gets detected as foreground blobs. These techniques are also very susceptible to outdoor scenarios where lighting conditions change based on weather. Similarly, the shadows also get detected as foreground blobs. Later works countered the effect of illumination, shadow and lighting using techniques like [19] and presented in [39], and [16]. Even after all these optimizations the blob detection using BG- subtract fails whenever two or more people are in close proximity in the frame producing a single fused blob and getting counted as one. Heuristic methods [32] based on human size can be used to split the blobs but such methods fail when people are carrying trolley or luggage/baggage with them. There are methods that use stereo vision to separate the fused blob of multiple people into exact count by estimating human size using depth information. There are other approaches where multiple cameras [43], [37] are used for people counting. But, the extra processing power required to process second camera feed and cost of multiple cameras makes the use of these methods undesirable. Apart from these, there are some bayesian based probabilistic approaches as suggested in [8] but these are designed to work with horizontal view, and hence fail when applied in overhead view. Similarly, Modification of HOG/SVM [2] - since only limited information is available in overhead view, relying only on HOG gives less accurate results.

Apart from detection, tracking those detection is the key in counting. Several methods exists for overhead tracking [13], [17] and [1] but these methods lack predictive nature in their approach which is key for robust system of counting. There are other single and multi object trackers like [30], [38], [7], [15] that gives state-of-the-art accuracy. However, the frame speed achieved from these methods is very minimal, MDNet and VITAL give 1 FPS and 1.5 FPS respectively with Tesla GPU. While, RDT, BACF gives 30-40 FPS but requires GPU. Achieving a real time tracking on low compute device would not be possible with such compute intensive algorithms. The tracking job needed to be computationally light and accurate as most of the computation power was already consumed by the detector. Kalman filter [23] along with Hungarian [29] assignment seems the most common method. Several complex methods for accurate tracking [41], [18] seems to be computationally expensive, hence not suitable for our use case. We used a simple Kalman tracking and Hungarian assignment combination with position and velocity as the parameters.

3. METHODOLOGY

The idea is to develop a simple system that would monitor a small area from overhead view and count the number of people passing through monitored area. With the recent success of neural networks in pedestrian detection and transfer learning, one possible approach is to re-train a fronto-parallel person detection networks for top-view detection. But deploying these computationally heavy networks on a portable low compute edge for a real-time application in cost effective manner is not a feasible task with today’s hardware. One possible solution is to have a GPU based server which can run these algorithms in real time. But, that would require sending data through internet. For a system monitoring an area and tracking each detection, it would require sending every captured frame to the server and therefore, require a lot of bandwidth and reliable up-time of network. This is not a reliable solution in cases such as latency and down time of network which can not be controlled for a real time application. The other method could be using Neural compute stick by Intel along with Raspberry Pi. But from that we could achieve 4.39 FPS [35] using SSD mobile net [21] which is not sufficient for real-time tracking. So is the case with neural network based tracker. Hence, we were required to develop a reliable detector and tracker algorithm which could be made to run on a small hardware device and can work as an offline standalone system. Detecting people in the frame using background subtraction by determining the foreground blob to be of single person or multiple people is a cumbersome task even to a human eye. In HOG-SVM approach the Histogram-Of-Gradients alone is not sufficient and more information is required for it to work on overhead view. We chose the Aggregated-Channel-Features based detector proposed by P.Dollar et al [12] because of the additional three color channels which helps capture head and partially visible face features while the remaining six channels for gradients in six directions models the physical structure of head and shoulders. After detection however a robust tracker is needed which should negate the effect of missed and false detections as explained in Figure 2. Kalman filter's ability to predict the next position of tracks without actual detection made tracking robust towards missed detections, while Hungarian assignment terminates the track initiated due to false positives. Further optimizations were made in detector and tracker to improve performance. We also discuss the multi-threaded implementation and how it boosts the frames per second throughtput of our algorithm.

The following subsections of paper shed light on optimization, multi-threaded implementation and people counting zone.

3.1 Overhead Detector

The overhead detector is based on the work done by P.Dollar et al on pedestrian detection using Aggregated Channel Features (ACF) with Adaboost Classifier. The overhead detector is trained on the images extracted from the videos taken from an overhead installed camera. To run it in real-time on low compute devices, optimizations were made on restricting detection scales and capturing video at optimal resolution. The resolution of input video is computed using average person size such that the person gets detected in first few scales. The average person size is a function of installation height and taken as input during device setup. For a camera installation height 'h' where h e H = [hi, h2, h3,..., hn}, there exist a scale 's' where s e S = [si, s2,..., si2} in which detection occurs (according to P.Dollar work these are 12). Corresponding to each scale there exist an image resolution r e R = [ri, r2,..., ri2}. For a person getting detected in scale say 's5', the computation done on scales from si - s4 is unnecessary. Therefore, the input video is recorded at a resolution corresponding to s5 such that s5 is the first scale of detection.

For a camera installation height 'h', the variation in person size in pixels is limited and is a function of person's height. Experimentally, we found that this can be covered in 4 scales of detector, therefore, saving computation of remaining 8 scales. We also found that as the resolution of image decreases - the false positive increases. Therefore, the two optimization improved the performance in terms of time and also reduced the occurrence of false positives.

3.2 Tracking

The actual In/Out count must be updated only when person enters/goes out of the view completely, we need an algorithm which is able to handle cases such as (i) people loitering in the frame should not be counted even if the person crosses the mid-line multiple times. (ii) false counts resulting from people changing their minds halfway even if they have crossed the mid-line should not be counted as they have not completed their sequence. This is especially true in the use case of retail people counter. (iii) No detector is 100% accurate. There are missed detections and false positives. Therefore, we propose a novel tracking logic which is be able to cater to all of the above situations and be robust enough to predict accurate count.

Missed detection causes premature termination of tracks, while false positives initiates false tracks and also affect the true tracks by latching (explained in figure 4. Fine tuning of kalman filter predictive feature based on displacement and velocity resolves the problem of missed detection and keeps the track from pre-mature termination. However, the problem of latching among tracks still persists due to:

1)improper detection at the edges of frame (partial view of person).

2)false positive. Latching can be understood from the following cases: 1) When the person goes out of frame, the track should end with no detection assignment, but instead, the track may be assigned to the false positive and the count is missed as shown in figure 4(i). 2) Similar situation occurs when missed detection happens, the track of true positive may be assigned to false positive and when person is detected again, a new track is initiated and the old track is lost to false positive as shown in figure 4(ii). This leads to inaccurate tracking and count miss. 3) Latching of false positive track to true positive creates false initiation point of track or worse, lead to count miss as shown in figure 4(iii). These detection and latching issues combined together creates lot of inaccuracy in tracking and the count. We present a novel approach, Figure 3, to handle false positive - dynamic learning and unlearning of false positives. We maintain the average confidence, number of missed frames and number of detected frames for every detection in tracker. If any detection crosses the threshold of number of detected frames (learning rate) then that detection is not passed to Kalman and Hungarian. This history is maintained for nearby area of detection instead of a single point to negate the effects of jitter. This solution can cause the problem of not passing the true positive to tracker for the learned region but that region is very small and can be handled by tracker. There were cases where a person standing in the frame is learned, this way we could end up with entire FOV turning into missed detection region. To differentiate between a learned true positive and false positive, we again refer to the properties of detection. A true positive, even though learned, will not be present for the entire duration, therefore, the unlearning of detection. If number of missed detection frames is greater than the threshold of learning rate, then that detection is unlearned.

At the edge of the frame, partial visibility of person gives jittery detection. The person moving out of frame has outward velocity and acceleration, but detected center remains constant due to gradually reducing detected partial rectangle size. The stationary center and jitter gives apparent deceleration in contrast to actual direction of movement. This leads to Kalman filter prediction in opposite direction. This estimation may cause the track to latch on the nearby detection, illustrated in figure 5(i). The information for the two tracks is lost due to latching, first of the actual track and second of the new detection on which track latched on. To overcome this problem, we introduced the region based tracking. These regions at the corner provides the buffer to terminate tracks gracefully and act as virtual frame boundaries beyond which if detection happens then the corresponding track is terminated thereby preventing latching as illustrated in figure 5(ii). Therefore, we avoid the error prone edges.

3.3 Counting method

We divided our FOV in different regions as Red, Yellow, Green, Pink and Cyan showed in figure 6. The counting is done based on region of first detection and last true detection. We maintained the starting region and ending region for every track and when the track is terminated count is updated. Count increment cases are as follows: Red to Green - increase IN count, Red to Cyan - increase IN count, Green to Red - increase OUT count, Green to Pink - increase OUT count. While the yellow region is the buffer zone where no change of count takes place whether detection begins or ends. Pink region is virtual boundary towards exterior. Cyan is virtual boundary towards interior.

3.4 Real Time Implementation

The accuracy of the algorithm is heavily dependent on the frame rate of the system as tracking accuracy suffers as frame rate decreases. The fps achieved with a single threaded implementation was very low. The time taken to process a frame was also dependent on the number of people in the frame as all the weak classifiers of Adaboost has to run on each block for a true positive. Hence the fps falls even further when there are multiple people in the frame bringing it below the average fps. A single threaded implementation, after all the optimization could not achieve the required accuracy. Hence a multi-threaded implementation was required.

The algorithm was implemented with an asynchronous multithreaded design illustrated in figure 7. The input frames were queued into an input buffer queue. This made sure the system was working in a constant frame rate irrespective of whether there are people in the frame or not.

The detector part of the algorithm was heaviest (taking almost 80 to 90 of percent of the processing). The detection for a particular frame was independent of any previous or future frames. Hence multiple instances of detectors could work independently.

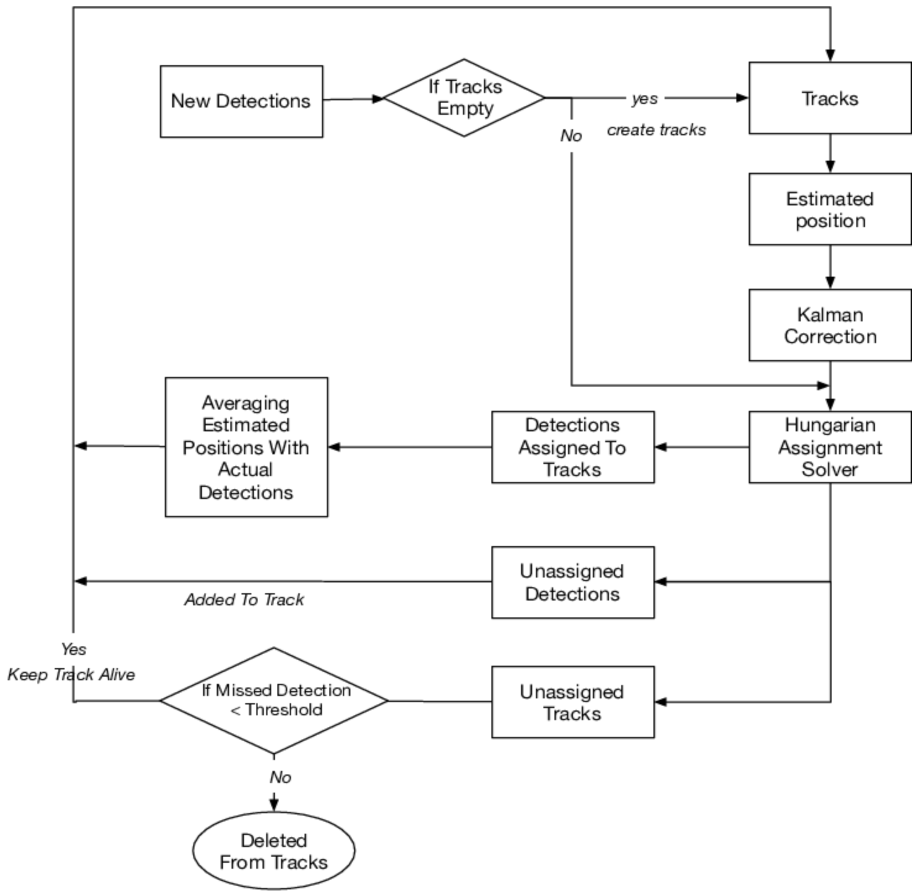

Figure 2. This figure illustrate the flow chart for Tracking algorithm

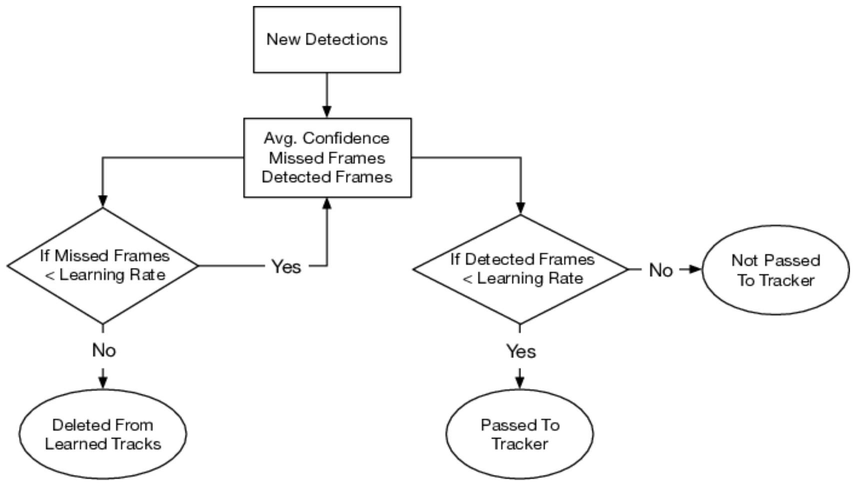

Figure 3. This figure illustrate the flow chart for following: (a) Learning/Unlearning of false positive

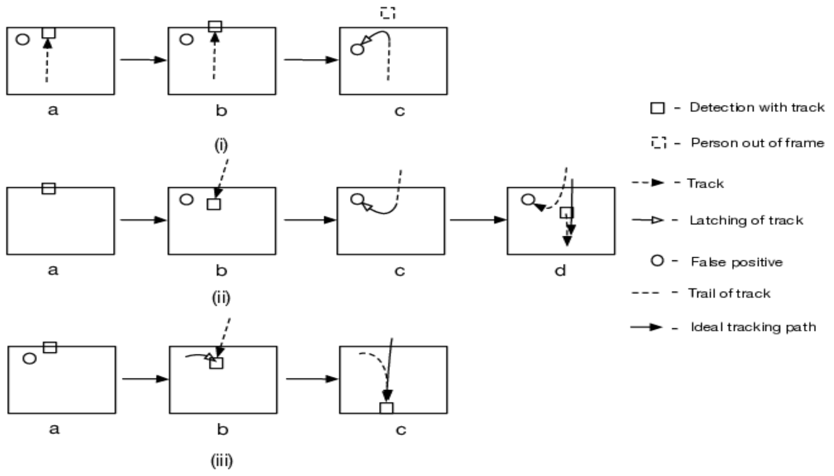

Figure 4. This illustrates Latching scenarios:® tracking jitter & deceleration at edge of the frame along with false positive causing latching. (ii) Appearance of false positive and missed detection, causing original track latching on to false positive & new track initiated for actual detection. (iii) Track initiated by false positive being latched on to true positive.

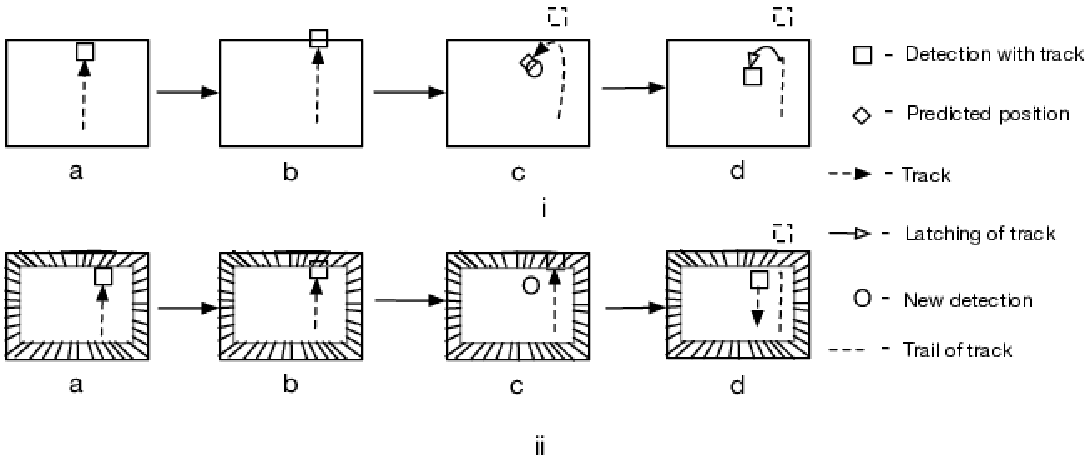

Figure 5. This illustrates the effect of buffer zone, i) In the figure, ’a’ & ’b’ represents tracks life, ’C’ shows predicted position due to deceleration and also new detection, d shows that since predicted & actual detection is close by therefore latching. ii) The shaded region is the buffer zone providing cushion to terminate tracks gracefully with no false prediction in ’c’ and in’d’ a new track is initiated for new detection.

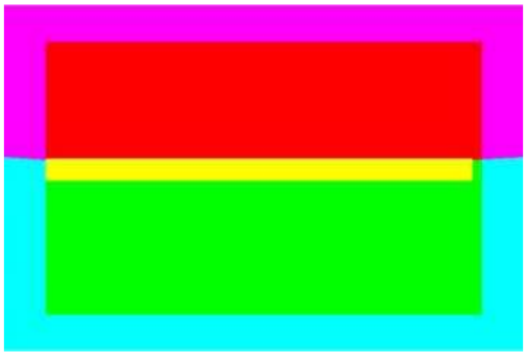

Figure 6: zonemap

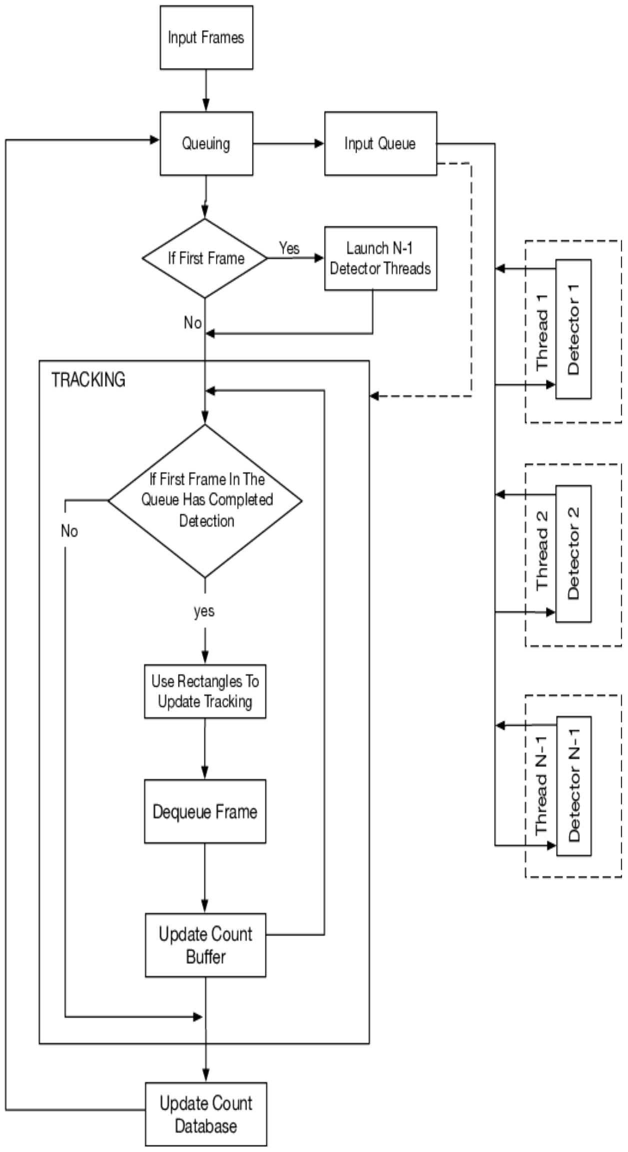

Figure 7. This figure illustrate the flow chart for multi-threaded design

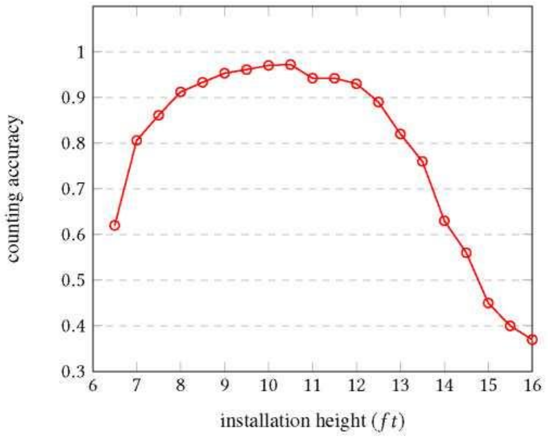

Figure 8. Counting accuracy for different installation height

The implementation was configured for N threads where N is an input parameter. Main thread does queuing of frames into the buffer and tracking job. The master thread launches N-1 threads, each performing detection on a particular frame. Once launched these threads run continuously until the exit is called. Each detector look for frames in the input queue in sequential order. It takes the oldest frame on which detection is not performed, does detection and puts it back into the queue along with detected rectangles, so that the order is maintained. Since the detection on a single frame is independent of other frames, the detectors can run asynchronously. The tracking needed to be sequential, hence there is only one instance of tracking which was performed by the main thread. The main thread after queuing an input frame, check whether the oldest frame in the queue has completed detection. If yes, it will use those rectangles to update tracking and dequeue the frame and looks for the next oldest frame. This continue until a frame on which detection is not completed is in the front of the queue.

The detector uses lot of memory for channels computation of Adaboost Classifier. Allocation and de-allocation of the memory repeatedly was adversely affecting the performance. Therefore, a custom memory management technique was implemented in which a chunk of memory was allocated once in the beginning and was reused for each frame. Similar technique used for input frame queue.

4. EXPERIMENT AND RESULTS

The state-of-the-art methods are slow on low compute-power devices. Doing real time counting is not feasible on such devices.

The novelty of the paper is the proposed methodology which gave an accuracy of 97% on device like R-Pi 3B in real time (i.e.20FPS). In this section we demonstrate the step by step improvement in accuracy due to algorithmic optimisations. The comparative study of our method could not be done with state of the art methods in terms of accuracy due to two reasons. 1) State of the art accuracy are achieved by Deep Learning based solutions. These accuracy could not be achieved in real time (20 FPS) on devices like R-Pi due to low frame rate. The FPS achieved by DL methods is mentioned in table 1. Such poor FPS discouraged us to proceed further with it. 2) The described algorithm is optimised for the presented overhead view as shown in 1. There is no dataset publicly available which has the same view. Several datasets [5], [27], [14], [12] are available for overhead view but they are not perfectly overhead view. They have an angled view where partial of face/side of head can be seen. The performance on two datasets could not be compared due to their different nature and therefore, our custom datasets (e.g in Fig:1) was used for accuracy computation and benchmarking. The accuracy is computed in two folds - algorithmic accuracy and real time accuracy on device. We created our own data-set by recording videos from overhead in different ambient conditions covering the following use cases. 1) Different installation height, 2) Changing natural lighting condition, 3) Artificial lighting, 4) Different flooring.

The implementation of detecting person in static background using background subtraction along with shadow detection produced good results under experimental setup giving an accuracy of 78%. However, when the algorithm is subjected to illumination and lighting changes, counting accuracy fell below 50%.

Table 2: Accuracy variation with various improvements in the Detector & Tracker. In the table, PD - person detector, BG-Sub - Background subtraction, OHD - Our Overhead person detector, KFHA - Kalman Filter and Hungarian assignment, IRFSD - Input resolution is the first scale of detection, BZ - Blue zone, SRD - Scale restriction in detector, FPL/U - False positive learning/unlearning, Accuracy - Bi-directional Counting Accuracy (in %)

Incremental changes in PD	Incremental changes in Tracker	Accuracy
BG-Sub	KFHA	40.9
OHD	KFHA	61.2
OHD + IRFSD	KFHA	70.8
OHD + IRFSD + SRD	KFHA + BZ	84.7
OHD + IRFSD + SRD	KFHA + BZ + FPL/U	97.2

Table 3: Multi-threaded vs single threaded (computed at frame rate of 20 FPS)

Approach	Bi-directional Counting Accuracy (in %)
Single threaded	73
Multi-threaded	97.2

Our detector is robust and efficient towards sudden lighting, illumination, shadow changes such as switching On/Off light in the middle and people coming in groups. The optimization made in detector and tracker along with parallel threaded implementation gave a bi-directional counting accuracy of 97% at 20FPS. A typical view of the detection and tracking is shown in figure 1. The counting accuracy is computed videos of 25580 minutes with total bi-directional count of more than 11946. Ground truth is maintained by manually counting the number of people in videos. Criteria for a valid count is —

1) Person should be completely visible in the frame,

2) Person should leave the frame completely

3) If person enters and leaves the frame from same side then count is not incremented

4) If person walks side-ways after entering the frame then that person is not counted.

For single threaded approach, tracking suffered due to frequent missing of frames caused by compute bottleneck and hence dip in accuracy. Therefore, we took multi-threaded approach and present the performance results in 3. We also installed our devices on 110 public locations with different ambient settings and at different installation height to assess the real time performance. We did manual counting on these locations and present the result in figure: 8. We compare results of Background subtraction with our implementation in terms of effect on counting accuracy with the suggested improvement in detector & tracking. The improvement in accuracy after each step is shown in Table 2.

5. CONCLUSION

In this paper, we presented a state-of-the-art system for real time people counting. We suggested a novel approach of using Overhead person detector along with multiple optimization to improve performance and compute time. We also made numerous improvements in tracker and suggested a novel approach of handling false positive by dynamic learning/unlearning. We achieved an accuracy of 97% in real time (20FPS) for a low computing embedded device. For a very low installation height of camera the accuracy decreases. Future work can be done on improving accuracy for low height installations.

6. REFERENCES

[1] Jonathan Owens A, Andrew Hunter B, and Eric Fletcher A. [n. d.]. *A Fast Model-Free Morphology- Based Object Tracking Algorithm.* ([n. d.]).

[2] I. Ahmed and J. N. Carter. 2012. A robust person detector for overhead views. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. 1483-1486.

[3] J. Barandiaran, B. Murguia, and F. Boto. 2008. *Real-Time People Counting Using Multiple Lines.* In 2008 Ninth International Workshop on Image Analysis for Multimedia

Interactive Services. 159-162. https://doi.org/10.1109/WIAMIS.2008. 27

[4] Tudor Barbu. 2014. *Pedestrian detection and tracking using temporal differencing and HOG features.* Computers & Electrical Engineering 40, 4 (2014), 1072 - 1079. https://doi.org/10.1016/j.compeleceng.2013.12.004

[5] A. B. Chan and N. Vasconcelos. 2008. *Modeling, Clustering, and Segmenting Video with Mixtures of Dynamic Textures*. IEEE Transactions on Pattern Analysis andMachine Intelligence 30, 5 (May 2008), 909-926. https://doi.org/10.1109/ TPAMI.2007.70738

[6] C. C. Chen, H. H. Lin, and O. T. C. Chen. 2011. *Tracking and counting people in visual surveillance systems.* In 2011 IEEE International Conference onAcoustics, Speech and Signal Processing (ICASSP). 1425-1428. https://doi.org/10.1109/ ICASSP.2011.5946681

[7] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. 2017. *Visual Tracking by Reinforced DecisionMaking.* CoRR abs/1702.06291 (2017). arXiv:1702.06291 http://arxiv.org/abs/1702.06291

[8] I. Cohen, A. Garg, and T. S. Huang. 2000. *Vision-based overhead view person recognition.* In Proceedings15th International Conference on Pattern Recognition. ICPR-2000, Vol. 1. 1119-1124 vol.1. https://doi.org/10.1109/ICPR.2000.905668

[9] N. Dalal and B. Triggs. 2005. *Histograms of oriented gradients for human detection*. 1 (June2005),886-893 vol. 1. https://doi.org/10.1109/CVPR.2005.177

[10] Piotr Dollar, Serge Belongie, and Pietro Perona. 2010. *The FastestPedestrian Detector in the West.* In Proceedings ofthe British Machine Vision Conference. BMVA Press, 68.1-68.11. doi:10.5244/C.24.68.

[11]Piotr Dollar, Zhuowen Tu, Pietro Perona, and Serge Belongie. 2009. *Integral Channel Features.* In Proc. BMVC. 91.1-91.11. doi:10.5244/C.23.91.

[12] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. 2012. *Pedestrian Detection: An Evaluation of the State of the Art. PAMI 34 (2012).* S. Duffner and C. Garcia. 2013. PixelTrack: *A Fast Adaptive Algorithm for Tracking Non-rigid Objects.* In 2013 IEEE International Conference on Computer

[13] Vision. 2480-2487. https://doi.org/10.1109/ICCV.2013.308

[14] M. Enzweiler and D. M. Gavrila. 2009. *Monocular Pedestrian Detection: Survey and Experiments.* IEEE Transactions on Pattern Analysis andMachine Intelligence 31, 12 (Dec 2009), 2179-2195. https://doi.org/10.1109/TPAMI.2008.260

[15] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. 2017. *Learning Background-Aware Correlation Filters for Visual Tracking.* CoRR abs/1703.04590 (2017). arXiv:1703.04590 http://arxiv.org/abs/1703.04590

[16] J. Garcia, A. Gardel, I. Bravo, J. L. LAqzaro, M. Martinez, and D. Rodriguez. 2013. *Directional People Counter Based on Head Tracking.* IEEE Transactions on Industrial Electronics 60, 9 (Sept 2013), 3991-4000. https: //doi.org/10.1109/TIE.2012.2206330

[17] A.B. Godbehere, A. Matsukawa, and K. Goldberg. 2012. *Visual tracking ofhuman visitors under variable-lighting conditions for a responsive audio art installation*. In 2012 American Control Conference (ACC). 4305-4312. https://doi.org/10.1109/ ACC.2012.6315174

[18] Joao F. Henriques, Rui Caseiro, and Jorge Batista. 2011. *Globally Optimal Solution to Multi-object Tracking with Merged Measurements.* In Proceedings of the 2011 International Conference on Computer Vision (ICCV '11). IEEE Computer Society, Washington, DC, USA, 2470-2477. https://doi.org/10.1109/ ICCV.2011.6126532

[19] Thanarat Horprasert, David Harwood, and Larry S. Davis. 1999. *A statistical approach for real-time robust background subtraction and shadow detection.* 1-19.

[20] Yaocong Hu, Huan Chang, Fudong Nian, Yan Wang, and Teng Li. 2016. *Dense crowd counting from still images with convolutional neural networks.* Journal of Visual Communication and Image Representation 38 (2016), 530 - 539. https: //doi.org/10.1016/j.jvcir.2016.03.021

[21] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and KevinMurphy. 2016. *Speed/accuracytrade-offs for modern convolutional object detectors.* CoRR abs/1611.10012 (2016). arXiv:1611.10012http://arxiv.org/abs/ 1611.10012

[22] Xiaoyn Huang, Liyuan Li, and T. Sim. 2004. *Stereo-based human head detection from crowd scenes.* In 2004 International Conference on Image Processing, 2004. ICIP '04., Vol. 2. 1353-1356 Vol.2. https://doi.org/10.1109/ICIP.2004.1419750

[23] Rudolph Emil Kalman. 1960. *A New Approach to Linear Filtering and Prediction Problems.* Transactions ofthe ASME-Journal of Basic Engineering 82, Series D (1960), 35-45.

[24] Damien Lefloch. [n. d.]. *Real-Time People Counting system using Video Camera.*

[25] J. Li, L. Huang, and C. Liu. 2011. *Robust people counting in video surveillance: Dataset and system.* In 2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). 54-59. https://doi.org/10.1109/ AVSS.2011.6027294

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: *Single Shot MultiBox Detector.* In Computer Vision - ECCV 2016, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 21-37.

[27]Chen Change Loy, Ke Chen, Shaogang Gong, and Tao Xiang. 2013. *Crowd Counting and Profiling: Methodology and Evaluation.* Springer New York, New York, NY, 347-382. https://doi.org/10.1007/978-1-4614-8483-7_14

[28] Ruijiang Luo and Yan Guo. 2001. *Real-time stereo tracking of multiple moving heads.* In Proceedings IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems. 55-60. https: //doi.org/10.1109/RATFG.2001.938910

[29] James Munkres. 1957. *ALGORITHMS FOR THE ASSIGNMENT AND TRANSPORTATION PROBLEMS.* (1957).

[30] Hyeonseob Nam and Bohyung Han. 2015. *Learning Multi-Domain Convolutional Neural Networks for Visual Tracking.*

CoRR abs/1510.07945 (2015). arXiv:1510.07945 http://arxiv.org/abs/1510.07945

[31] D. Onoro Rubio and R. J. Ldpez-Sastre. 2016. *Towards perspective-free object counting with deep learning.* In ECCV.

[32] Sangho Park and J. K. Aggarwal. 2000. *Head segmentation and head orientation in 3D space for pose estimation of multiple people.* In 4th IEEE Southwest Symposium on Image Analysis and Interpretation. 192-196. https://doi.org/10.1109/IAI.2000.839598

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. 2016. *You Only Look Once: Unified, Real-Time Object Detection.* In 2016 IEEE Conference on Computer Vision and PatternRecognition (CVPR). 779-788. https://doi.org/10.1109/CVPR. 2016.91

[34] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* CoRR abs/1506.01497 (2015). arXiv:1506.01497 http://arxiv.org/abs/1506.01497

[35] Adrian Rosebrock. 2018. *Real-time object detection on the Raspberry Pi with the Movidius NCS*. (2018). https://www.pyimagesearch.com/2018/02/19/ real- time-object- detection- on-the- raspberry-pi- with- the- movidius-ncs/

[36] A. K. Sahoo, S. Patnaik, P. K. Biswal, A. K. Sahani, and P. B. Mohanta. 2013. *An efficient algorithm for human tracking in visual surveillance system.* In 2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013). 125-130. https://doi.org/10.1109/ICIIP.2013.6707568

[37] Thiago T. Santos and Carlos H. Morimoto. 2011. *Multiple camera people detection and tracking using support integration.* Pattern Recognition Letters 32,1 (2011), 47 -55. https://doi.org/10.1016Zj.patrec.2010.05.016 ImageProcessing, Computer Vision and Pattern Recognition in Latin America.

[38] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson W. H. Lau, and Ming-Hsuan Yang. 2018. *VITAL: VIsual Tracking via Adversarial Learning.* CoRR abs/1804.04273 (2018). arXiv:1804.04273 http://arxiv.org/abs/1804.04273

[39] Jae won Kim, Kang sun Choi, Byeong doo Choi, and Sungjea Ko. 2002. S.-J.: *Real-time Vision-based People Counting System for the Security Door.* In: Proc. of 2002 International Technical Conference On Circuits Systems Computers and Communications, Phuket.

[40] Huazhong Xu, Pei Lv, and Lei Meng. 2010. *A people counting system based on head-shoulder detection and tracking in surveillance video.* In 2010 International Conference On Computer Design and Applications, Vol. 1. V1-394-V1-398. https://doi.org/10.1109/ICCDA.2010.5540833

[41] A. L. Yussiff, S. P. Yong, and B. B. Baharudin. 2014. *Parallel Kalman filter-based multi-human tracking in surveillance video.* In 2014 International Conference on Computer and Information Sciences (ICCOINS). 1-6. https://doi.org/10.1109/ ICCOINS.2014.6868359

[42] Cong Zhang, Hongsheng Li, X. Wang, and Xiaokang Yang. 2015. *Cross-scene crowd counting via deep convolutional neural networks*. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 833-841. https://doi.org/ 10.1109/CVPR.2015.7298684

[43] ZhognchuanZhang andF. Cohen. 2013. *3D pedestrian tracking based on overhead cameras.* In 2013 Seventh International Conference on Distributed Smart Cameras (ICDSC). 1-6. https://doi.org/10.1109/ICDSC.2013.6778235

[44] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and S. Avidan. 2006. *Fast Human Detection Using a Cascade of Histograms of Oriented Gradients.* In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06),Vol. 2. 1491-1498. https://doi.org/10.1109/CVPR.2006.119