AUTOMATIC COUNTING OF INTERACTING PEOPLE BY USING A SINGLE UNCALIBRATED CAMERA

Senem Velipasalar [*]
Princeton University, Electrical Engineering Dept.
Princeton, NJ, 08544
svelipas@princeton.edu

Ying-Li Tian, Arun Hampapur
IBM T.J. Watson Research Center
Hawthorne, NY, 10532
yltian, arunh@us.ibm.com

ABSTRACT

Automatic counting of people, entering or exiting a region of interest, is very important for both business and security applications. This paper introduces an automatic and robust people counting system which can count multiple people who interact in the region of interest, by using only one camera. Two-level hierarchical tracking is employed. For cases not involving merges or splits, a fast blob tracking method is used. In order to deal with interactions among people in a more thorough and reliable way, the system uses the mean shift tracking algorithm. Using the first-level blob tracker in general, and employing the mean shift tracking only in the case of merges and splits saves power and makes the system computationally efficient. The system setup parameter can be automatically learned in a new environment from a 3 to 5 minute-video with people going in or out of the target region one at a time. With a 2GHz Pentium machine, the system runs at about 33fps on 320x240 images without code optimization. Average accuracy rates of 98.5% and 95% are achieved on videos with normal traffic flow and videos with many cases of merges and splits, respectively.

1. INTRODUCTION

There is an increasing need and demand for automatic, efficient and reliable counting of people entering or exiting a target region. Shopping malls and supermarkets can use this information to identify hourly traffic patterns, to optimize labor scheduling, and to monitor the effectiveness of promotional events. The counting of people is also very important for security purposes. It helps assign accurate number of security people at key places, and design efficient evacuation plans.

The existing approaches can be classified into three broad categories: Systems using contact-type counters; systems using sensors; and vision-based systems using cameras.

Systems using contact-type counters, such as turnstiles, obstruct the passage way, and can cause congestion if there is high-density traffic as they count people only one at a time. In addition, they have the limitation of possible undercounting. Systems using infrared beams or heat sensors do not block the doorways, but suffer from the same undercounting problem.

The earlier attempts in image-based systems were successful for scarce traffic of people. An overhead stereo camera is used in [1] and [2]. Multiple people can be dealt with in [1] as long as they move separately. The system in [2] has the advantage of estimating object heights, but this requires a calibrated stereo camera head and eliminates the flexibility of using a single ordinary camera.

Multiple cameras are used in [3] and [4]. It is stated in [3] that people walking together can cause problems as they cannot be resolved. The method in [4] gives an idea about the number of people in the scene, but does not track them individually. A single camera system is introduced in [5]. Two virtual base lines are used for direction detection. These lines cannot be too close in the proposed scheme and interactions that occur on or around them can cause problems.

To the best of our knowledge, the existing systems cannot handle challenging people interactions in the region of interest (ROI), especially in the vicinity of the baselines. We introduce an automatic method to count interacting people. We tested the proposed method with sequences which contain many interactions (such as merges/splits, shaking hands, and hugging) between people in the ROI. Most of these interactions occur right in the vicinity of the *entry/exit line*, thus successfully resolving them is essential to determine direction and perform counting accurately. Instead of stereo or multiple cameras, the proposed system uses a single inexpensive camera mounted overhead. It also eliminates the need for calibration, and thus creates a low-cost system which is easy to install and maintain, and more efficient. The main novelties of the method include (1) learning the size interval for a single person automatically, (2) using a single virtual *entry/exit line*, and thus detecting the entry/exit events in shorter time and minimizing the effects of appearance changes, and (3) using an efficient, two-level hierarchical tracking structure and successfully handling challenging interactions in the ROI.

2. THE AUTOMATIC PEOPLE COUNTING SYSTEM

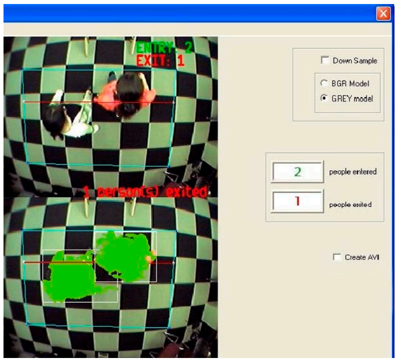
The proposed system uses only one camera that is mounted overhead to avoid the occlusions. The *person-size bounds* defining the interval for the size of a single person are the inputs to the system, which are learned in an automatic way.

The graphical interface lets the user define a ROI in the

[*] This work was done when the author was at IBM T.J. Watson Research Center.

1-4244-0367-7/06/$20.00 ©2006 IEEE

camera view so that only the people who pass through the ROI are counted. The region marked with blue lines in Fig.1 is the ROI defined by the user. Then the user needs to mark an *entry/exit line* (the red line in Fig.1), and click on the side of this line where the door is located. The door side information is used to determine the direction of the moving blob. As seen in Fig. 1, the system has two separate counters for entering and exiting people, whose numbers are displayed in green and red respectively. Moreover, each time a person enters or exits the ROI, a message is displayed on the image of the camera view. When a blob crosses the *entry/exit line*, its size is used to find the number of people forming that blob. Then, depending on the direction, the corresponding counter will be incremented by the number of people.



ENTRY: 2
EXIT: 1
1 person(s) exited

☐ Down Sample
○ BGR Model
● GREY model

2 people entered
1 people exited

☐ Create AVI

Fig. 1. A portion of the user interface. As soon as a person crossed the *entry/exit line* in exit direction, the exit counter was incremented by one and also "1 person(s) exited" message was displayed.

2.1. Automatic Learning of System Setup Parameters

The only parameters that depend on the new environment and setup are the *person-size bounds* – the upper and lower bounds for the size of a single person. For new deployments, these bounds are automatically learned in our system. After the camera is placed, a video sequence of about 3 to 5-minute length is captured in which different people go in and out of the monitoring region one person at a time. For each frame, the areas of the *improved blobs* are saved to determine the lowest, the highest, and the mean value of the single person size from the video. The *improved blobs* are obtained as explained in Section 3. If there is enough deviation, the lowest and highest values are set to be the *person-size bounds*. Otherwise, the lower and higher values are determined so that they will have equal distance from the mean and the deviation will be 0.4 (after normalization of the values by the mean value). By learning the bounds from this video sequence, we can have the necessary tolerance for different effects, such as clothing, carried items or different-height people, in our person size interval.

If this interval is denoted by $PSI = [LB \quad UB]$, then the mean will be $M = \frac{LB+UB}{2}$. Let $D = M - LB = UB - M$. Given that the area of a foreground blob is N pixels, the decision about the number of people, P, forming that blob is currently made as follows:

$$\begin{array}{llll} P=1, & if\ M-D & \leq N \leq & M+D \\ P=2, & if\ M+D & < N \leq & 2M+0.5D \\ P=3, & if\ 2M+0.5D & < N \leq & 3M+0.8D \\ P=4, & if\ 3M+0.8D & < N \leq & 4M+D \\ P=5, & if\ 4M+D & < N \leq & 5M+1.2D \end{array} \tag{1}$$

This scheme is based on experimental studies, and gives reliable results. However, to increase robustness and to remove the assumption that every moving object is a person, we will incorporate a people segmentation scheme to our system, and then cross check the decisions resulting from the intervals in (1). As we learn the person-size bounds automatically for different environments, intervals in (1) provide valuable information, and will help the people segmentation step.

3. THE COUNTING ALGORITHM

One of the challenges in people counting problem is dealing with merge and split cases. Due to background subtraction (BGS), sometimes a merge can happen even if people do not touch but only get close to each other. The proposed algorithm performs automatic, bi-directional counting of people, and can handle many challenging interactions, such as merges/splits, shaking hands etc., between people in the ROI, even in the vicinity of the *entry/exit line*. The direction of the moving people can be determined in shorter time and without needing two base-lines, which is a popular strategy in the art.

We employ a hierarchical tracking structure, where a fast and efficient blob tracking method is used as the first level. If a tracker cannot find its object, this signals the occurrence of a merge or split, and only in this case the mean shift tracker [7] is activated to resolve it. This strategy brings the following advantages: a) it is less error-prone compared to using mean shift all the way through, as in the long run mean shift can be distracted by similar colored background objects; b) the model distribution and object size are updated more reliably which is harder to perform if mean shift is used all the time; c) it is much faster and more efficient. In the experiments, it was observed that this hierarchical structure performs 5 to 6 times faster compared to using mean shift all the way.

First, people in the current video frame are segmented from background by using a BGS algorithm [6]. This step yields a foreground (FG) mask image which has bounding boxes around each FG blob. In order to fill the possible holes (which may occur due to BGS) in the FG mask, and to learn a more reliable color distribution for the FG blobs, a new mask is built which will henceforth be called the *improved mask*. First the *improved mask* is set to be equal to the original mask. Then an ellipse is fit to the FG pixel locations in each bounding box and the ellipse pixels are OR'ed with the corresponding FG pixels in the original mask. If the total number of filled pixels is less than half of the area of the ellipse, the *improved mask* is set to be the output of the OR operation. Otherwise, it is left unchanged. This area check is made to avoid joining a large background region into the mask. Fig. 2 shows how the holes in the FG blob are filled in the *improved mask*.

The overhead view of moving people can be modeled well with an ellipse, and with this method, compared to using morphological operations, the holes and missing parts can be filled successfully regardless of their area. This is a preprocessing step and for real-time performance another computationally more expensive method is not preferable.

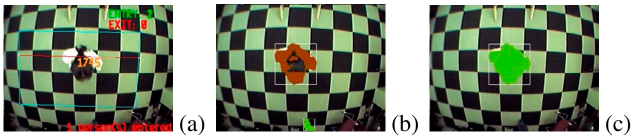


Fig. 2. Fitting an ellipse and using the OR operation fills the holes in the original mask (b), resulting in the *improved mask* (c).

3.1. First-level Blob Tracking

For a FG blob, which is in the ROI and whose size is larger than LB, the distance of its centroid to the *entry/exit line* is calculated. If this distance is lower than a predefined threshold (i.e. the person has come close enough to the *entry/exit line*), and the number of trackers is 0, the color distribution of this blob is learned, and a new tracker is created for it.

The system has two containers for trackers. One container holds the trackers which track the people who are still in the process of crossing the *entry/exit line*, and is called the *tracker container*. The other container is for the trackers that keep the color distribution of the already counted people, and is called the *already-counted tracker container*. The reason for having the second container is the fact that a blob, which is close to the *entry/exit line*, can merge with an already counted blob before its color distribution can be learned. Then, the color distribution of the counted blob, which is saved in the *already-counted tracker container*, is used to resolve this merge.

A new tracker, created for a new blob, is stored in the *tracker container*. In addition, a label is given to the blob, and the centroid of it is stored as the initial centroid. If a blob is close enough to the *entry/exit line* and the number of trackers is not 0, then it is checked if this blob can be matched with any of the existing trackers. At this stage, a fast tracking algorithm is used. First, the trackers whose bounding boxes intersect with the bounding box of the FG blob are found. Then for those trackers the Bhattacharya coefficient $\rho(\mathbf{y})$ [7] is calculated. $\rho(\mathbf{y})$ is defined by $\rho(\mathbf{y}) \equiv \rho[p(\mathbf{y}), q] = \int \sqrt{p_{\mathbf{z}}(\mathbf{y}) q_{\mathbf{z}}} d\mathbf{z}$, where $\mathbf{z}$ is the feature representing the color of the target model and is assumed to have a density function $q_{\mathbf{z}}$ while $p_{\mathbf{z}}(\mathbf{y})$ represents the feature distribution of the *improved mask* centered at location $\mathbf{y}$. The blob is assigned to the tracker which results in the highest Bhattacharya coefficient greater than a predefined threshold, and gets the label of that tracker. We used a threshold of 0.7 in all the experiments. Then, it is checked if this blob has crossed the line by inserting the centroid of the blob, and the initial centroid of its tracker into the equation of the *enty/exit line*. If the obtained signs are different and the number of pixels on the side of the current centroid is larger than the number of pixels on the other side, this means the blob has crossed the line. If the sign obtained for the initial centroid of the blob is equal to the sign obtained for the door-side point then the entry counter, if not, the exit counter is incremented. Then the tracker of this blob is removed from the *tracker container* and placed into the *already-counted tracker container*. This way, the entry/exit event is detected in a very short period of time, i.e. the tracking starts when the blob is close to the *entry/exit line* and the direction of the movement can be determined as soon as the blob crosses the line. If the blob has not crossed the line yet, its model distribution is updated. Already counted FG blobs in the ROI are tracked the same way.

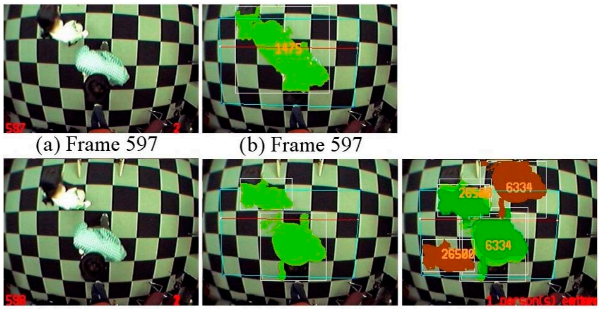
3.2. Second-level Tracking to Resolve Merges and Splits

If a blob cannot be matched to any of the trackers in the *tracker container*, then either a merge or split has occurred; or a new person/persons entered in the ROI. First, it is checked if all the existing trackers have found their matches. This is done by using the first-level tracker. If every tracker has not found its match, this means that either a merge or split has occurred. We activate the mean shift tracker [7], only when we need to resolve these cases.

Dealing with Splits: Splits present a greater challenge than merges. Because, when a split occurs, we need to find the location of the blob, formed after the split, in the previous frames to be able to decide if it is entering or exiting the ROI.

The proposed method keeps a buffer of the last 10 frames to resolve the split cases. For every tracker in the *tracker container* that could not find its match, it is checked if the area of the model mask of that tracker is greater than the area of the unmatched FG blob plus $LB/3$ (adding $LB/3$ saves time and power by avoiding a split check each time the model mask area of the tracker is slightly larger than the area of the unmatched blob). Let's denote the tracker satisfying this check by T_c. The color distribution of the unmatched FG blob is learned and the optimal location of this blob is found in the previous frame by using mean shift. The boundary box of the unmatched FG blob is moved to this location in the previous frame, and is denoted by BB_{f-1}, where f is the current frame number. If the Bhattacharya coefficient obtained from the mean shift step is larger than a threshold of 0.7 and if BB_{f-1} and the model mask of T_c overlap, this signals a split, and it is said that BB_{f-1} claims its part in the model mask of T_c. Then the location of the FG blob is found in frame $f - 10$ by mean shift. By using the current centroid of the blob and its centroid at $f - 10$, it is determined if this blob has already crossed the line. Similarly, another FG blob, which can not be matched to any of the trackers, will claim the remaining part of the model mask of T_c. After new trackers are created for the smaller blobs, T_c is removed from the *tracker container*. Fig. 3 shows a split case being resolved successfully.

Dealing with Merges: With merges, there will be no need to track back through previous frames. If the area of the unmatched FG blob is greater than $2 \times LB$, this will be a candidate case for a merge. For every unmatched tracker T_c in the *tracker container*, its optimal location in the current frame is found by using mean shift. If the Bhattacharya coefficient obtained is greater than a threshold of 0.7, and if mask of the

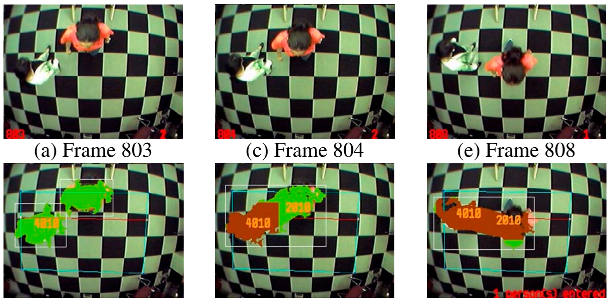


(a) Frame 597 (b) Frame 597

(c) Frame 598 (d) Frame 598 (e) Frame 598

Fig. 3. Split occurs at frame 598. (b) and (d) show the *improved mask*s (e) shows how the system successfully deals with the split. Light green, dark green and brown colors show the locations of the blob at the current, previous and 10th previous frames respectively.

T_c, after it has been moved to its found location in current frame, and the unmatched FG blob overlap, then T_c claims that part of the bigger blob. It is checked whether or not a blob have crossed the line by tracking through the merge using mean shift. Fig. 4 shows a merge case which is resolved successfully.



(a) Frame 803 (c) Frame 804 (e) Frame 808

(b) Frame 803 (d) Frame 804 (f) Frame 808

Fig. 4. Merge occurs at frame 804. The blob labeled 4010 claims its region in the bigger blob, a new distribution is learned for the remaining part of the bigger blob in (d) and is labeled as 2010.

The system empties both tracker containers when there is no one in the ROI. Thus, unless people tend to accumulate in the ROI, which is very unlikely, the number of trackers will not be large, making the matching process fast and efficient.

4. EXPERIMENTAL RESULTS

First, the proposed system has been tested for normal traffic flow i.e. people walking in and out without too many interactions around the door. The system achieved a very high accuracy rate of 98.5%. The errors are caused by BGS when the person wears clothes with similar color to the floor, and when a FG blob is divided into smaller blobs. Then, to evaluate the performance of the system in dealing with merges and splits, we captured different video sequences with various levels of difficulty. There are many interactions (such as merges, splits, shaking hands, and walking very closely) between people in the ROI. The evaluation results obtained from these sequences are summarized in Table 1, where the average accuracy rate is 95%. The experiments were performed with a 2GHz Pentium machine and the system runs at about 33fps on 320x240 images without code optimization.

		# OF ENTRIES	# OF EXITS	# OF MERGES	# OF SPLITS
VID.1	GT	16	17	4	3
	Output	16	16	4	3
	Accr.	100%	94%	100%	100%
VID.2	GT	14	16	3	3
	Output	13	16	2	3
	Accr.	93%	100%	66.7%	100%
VID.3	GT	9	9	4	2
	Output	9	10	3	2
	Accr.	100%	89%	75%	100%
VID.4	GT	12	12	–	2
	Output	11	12	–	2
	Accr.	92%	100%	–	100%
VID.5	GT	11	8	2	–
	Output	12	8	2	–
	Accr.	91%	100%	100%	–
VID.6	GT	7	7	5	2
	Output	7	6	5	2
	Accr.	100%	86%	100%	100%

Table 1. Experimental results obtained after evaluating our system on different videos. GT and Accr. denote Ground Truth and Accuracy respectively.

5. SUMMARY AND CONCLUSIONS

Considering the effort expended during the capture of the sequences to make them especially challenging, the proposed method performed very reliably and achieved high accuracy rates. The next goal is to differentiate people from other moving objects such as shopping carts. If merging people have similar colored outfits, this can be problematic for mean shift. Thus, we would like to incorporate people segmentation to strengthen the decisions made in merge/split cases and in figuring out the number of people forming a blob. We designed our system as a counting module built upon a BGS module. The performance of the algorithm is evaluated assuming that BGS performs reliably, because different state of the art approaches such as shadow removal can be used to improve the output of BGS.

6. REFERENCES

[1] K. Terada, D. Yoshida, S. Oe, J. Yamaguchi, "A method of counting the passing people by using the stereo images," *Proc. of IEEE ICIP*, vol. 2, pp. 338-342, Oct. 1999.

[2] D. Beymer, "Person counting using stereo," *Proc. of Workshop on Human Motion*, pp. 127-133, December 2000.

[3] V. Kettnaker, R. Zabih, "Counting people from multiple cameras," *IEEE Int'l Conf. on Multimedia Computing and Systems*, vol. 2, pp. 267-271, June 1999.

[4] D.B. Yang, H.H. González-Banos, L.J. Guibas, "Counting people in crowds with a real-time network of simple image sensors," *Proc. of IEEE ICCV*, pp. 122-129, October 2003.

[5] T.-H. Chen, and C.-W. Hsu, "An automatic bi-directional passing-people counting method based on color image processing," *IEEE Int'l Carnahan Conf. on Security Technology*, 2003.

[6] Y. -L. Tian, M. Lu, and A. Hampapur, "Robust and Efficient Foreground Analysis for Real-time Video Surveillance," *Proc. of IEEE CVPR*, June 2005.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," *Proc. of CVPR,* vol. 2, pp. 142-149, 2000.