STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

A COMPUTER VISION APPROACH TO OBJECT TRACKING AND COUNTING

SERGIU MEZEI AND ADRIAN SERGIU DARABANT

Abstract. This paper, introduces a new method for counting people or more generally objects that enter or exit a certain area/building or perimeter. We propose an algorithm (method) that analyzes a video sequence, detects moving objects and their moving direction and filters them according to some criteria (ex only humans). As result one obtains *in* and *out* counters for objects passing the defined perimeter. Automatic object counting is a growing size application in many industry/commerce areas. Counting can be used in statistical analysis and optimal activity scheduling methods. One of the main applications is the approximation of the number of persons passing trough, or reaching a certain area: airports (customs), shopping centers and malls and sports or cultural activities with high attendance. The main purpose is to offer an accurate estimation while still keeping the anonymity of the visitors.

1. Introduction

Researches in video processing and implicitly image processing have been very active in the last years and the result is the abundance of new techniques that have been successfully adopted in various branches of science, industry, medicine and security systems. High performance computers have become widely available at relatively low costs and this makes it possible for many businesses to track the number of people that walk over their door step using a simple camera and a PC. Counting people gives retailers knowledge about their visitors. How they can be manipulated by advertisements, weather, time of day etc. This is extremely valuable and helpful information which can be used to, for example, plan staffing and optimum times for cleaning and maintenance,

Received by the editors: June 28, 2010.

2010 *Mathematics Subject Classification.* 68T45, 68U10.

1998 *CR Categories and Descriptors.* I.5.4 [**Computing Methodologies**]: PATTERN RECOGNITION – *Applications*; I.4.6 [**Computing Methodologies**]: IMAGE PROCESSING AND COMPUTER VISION – *Segmentation*.

Key words and phrases. computer vision, object recognition, blob tracking,counting.

This work is supported by the Romanian Ministry of Education in the frame of PN2 ID-550/2007.

determine opening hours and preferred advertisements, and so forth. The application itself is based on analyzing video streams from cameras in real time or from pre-recorded video files. Earlier people counting methods are the laser beam method which is basically a counter that increments the number of people every time the beam is interrupted; the ultrasound method, which sends an ultrasound beam that bounces back of the obstacles and determines the distance from the source to them in order to determine if the people have entered a certain area or not. These methods, although they are still used in many department stores, are highly inexact in their calculations because of the number of situations in which two persons enter the store next to each other or the fact that by using the above mentioned methods you can count only the total number of people and not the number of *ins* and *outs*. The main advantages of the method proposed in this paper, over a regular counting system are the following:

- Bidirectional counting
- Flexibility of the algorithm when unexpected situations occur
- Surveillance of large entrances or large spaces
- Surveillance of high traffic areas
- Easy integration with databases or online systems
- Easy installation process that does not restrict access to the store or building

The application is structured in five main parts: the first handles the capturing of the video (reading the video stream); the second represents the processing of the individual frames in order to obtain the binary image that contains the people blobs (binary large objects); the third part handles the blob detection and tracking part; the fourth part is the actual algorithm that counts the number of people that enter or exit and the last part, the fifth, handles the actual output of the application and the data being written on the video stream.

2. Object Tracking and Counting

This section presents the methods we used to reach the final results as well as the reasons for which they are used. They are divided into three main categories:

- Image Enhancement
- Blob Detection
- Blob Tracking

All algorithms are presented using sample images that illustrate the results. Figure 1(b) shows an example of an original frame from a video. The frame shows the normal placement of the camera - usually placed on ceilings and

oriented towards the floor in order to allow object detection and to limit the field of vision such that identification of the visitors could not be possible. The counting approach is based in detecting moving objects in the camera field of vision, identify only human profiles and count them.

FIGURE 1. Original video frame from the input video.

The first problem that has to be faced here is to correctly identify only human profiles out of all items passing in front of the cameras using some kind of blob detection algorithm. The second problem to be dealt with is to correctly track the detected blobs so that we can keep trajectory information from one frame to the next and record their direction of movement. This would allow the identification and counting of objects *entering* and *getting out* of the controlled area. All these should be handled in an environment where lighting conditions are not always ideal. As the detection/measurement/counting process is usually continuous, the method should take care of variations in lighting during daylight to artificial lighting. Finally, often people tend to arrive in groups when interest area is highly crowded, thus partially covering each other even for a bird's eye view camera perspective. The following paragraphs describe the operations carried on the video flow, frame by frame in order to solve the counting problem.

2.1. **Image Enhancement Algorithms.** These algorithms represent the first step in the detection process.

2.1.1. *Background subtraction.* In order to obtain the foreground objects (in our case the people that enter the area) we need to do a background subtraction which means to subtract two images [6]. Figure 1(b) shows a video frame with people walking in the camera field of view, while 1(a) shows a frame with only static background. This method can be used when we are certain that we can get the first frame on the video to contain the static background and use that frame as a base frame and compare the real video frames against this base frame. It is however improper to assume that static background

frames with the same lighting conditions as for day, night and in-between times of the day could be obtained. Light is different according to the seasons and artificial lighting conditions, so a more complex approach is needed. The solution should not need to stop the flow of people who enter the measurement area. The adopted idea is to construct an image object that will contain an accumulation of frames over time which will represent a mean of those accumulated images. This *accumulator image* is subtracted from the current frame and thus we obtain the foreground objects. Figure 2 represents the foreground objects obtained from the above explained method.

FIGURE 2. Foreground objects after subtraction.

The image still contains a lot of noise requiring thus further processing. The subtracted resulting frame gives a pretty good approximation of the informations we are looking for. Noise should be eliminated in order to avoid its interference with the blob detection algorithm.

2.1.2. *Gray-scaling.* In order to convert any color to its approximate level of gray, one must obtain the values of its red, green and blue (RGB) primaries, in linear intensity encoding. Then, a percentage of these values are added, resulting the desired gray value between 0 and 255 (0 is black and 255 is white). These percentages are chosen due to the different relative sensitivity of the normal human eye to each of the primary colors (less sensitive to blue, more to green). More than color conversion, a gray level image only contains a lot less data to be processed -one byte per pixel as opposed to four bytes per pixel for true color images. One-channel color images are easier to be dealt with in computations. They also contain all the information required to solve the problem and any not needed data is already discarded.

2.1.3. *Binary Thresholding (Binarization).* The next steps for optimizing blobs detection is to threshold the image and transform it into a binary one. A

binary image still contains all needed information four our algorithm while further reducing data. Binarization is performed on gray images by applying a threshold and splitting the pixels into only black or white according to their value compared to the threshold. A good binarization is an essential and difficult step. One alternative is to choose a default threshold value that would work well for most videos. Statistical studies on the videos and their frames revealed that the value of 70 would be sufficient for most images. But there are special cases where the brightness highly differs from the average value such that no default threshold could be used. Leaving the task of choosing the right value to the user is not convenient in this case, as the high level of automation for the application would be damaged. Basically, after binarization we get a frame that only has two colors: black and white. The gray disappears because after binarization, depending on the thresholding value (in our case 70), everything that is under that value becomes black and everything over that value becomes white. Figure 3(a) represents the frame from Figure2 after thresholding is applied. At a closer look one can still observe some minor noise in the image.

FIGURE 3. Binary image(a), Eroded image(b).

2.1.4. *Erosion and Dilation.* Erosion is used in order to remove the noise in images and make the people blobs more *exact*. It applies a kernel to the image that erodes the borders of the connected components in a binary image based on the values from the neighborhood. Erosion is an iterative method, that can be applied multiple times on an image. Figure 3(b) describes the frame from Figure 3(a) on which erosion was applied. Erosion is applied in order to separate lightly connected blobs that are probably part of different blobs connected by noise or some other overlap or link situations. Dilation applies a kernel on the image in order to correct irregular pixel-based shapes. The dialtion is necessary because we want to fill the black regions within the blobs and make them contiguous. Like erosion, dilation is also an iterative method,

so it can be applied as many times as the user specifies. Figure 4 represents the frame from Figure 3(b) on which dilation was applied.

FIGURE 4. Dilated image-blobs are well separated and compact.

2.2. **Blob Detection.** Blob detection (blob analysis) involves examining the blob image and finding each individual foreground object [6, 3, 4]. This may in some special cases become a rather difficult task, even for the human eye. Figure 5(a) represents two blobs, which are easily distinguished from one another. When the moving persons have similar colors as the background, the blobs become rather distorted [1]. Some abnormal situations appear when blobs contain holes as shown in figure 5(b). Dilation was not sufficient in this case to completely compact the blob. Blobs are found by finding all adjacent pixels and putting them together so that the pixels form the blob object [1]. Analyzing contiguous blobs is a straightforward process. Fragmented blobs, like the one shown in figure 5(c), are harder to analyze since it is difficult to see if the blob is in fact a blob composed from multiple people. It becomes impossible to detect that in crowded areas. Still, large area blobs are likely to be groups of either people or something else that is moving. People standing in groups are hard to deal with, even for the human eye. When they stand close together or hold hands they form one big blob, like the one shown in Figure 6. It is hard to determine how many blobs there actually are in the image.

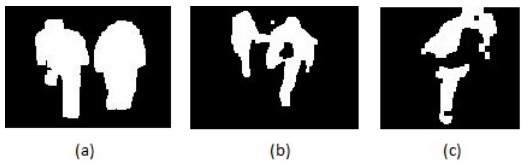

FIGURE 5. Normal blobs(a), Blobs with holes(b), Fragmented blobs(c).

FIGURE 6. Multiple people form a larger blob.

It may seem relatively straightforward to divide the larger blobs into smaller blobs by splitting the blobs with respect to height and width. This is not as easy, especially in an environment where not only people pass in the front of the camera (ex: in a mall or shop). People coming from grocery stores or other similar shops often push trolleys ahead of them. These trolleys usually appear to be of the same size as a human being. This makes splitting blobs by height or width a hard task, which must take movement into account. Two blobs moving horizontally appear larger in height, while moving vertically has larger width. In the current approach these are not dealt with yet. Blob analysis is all about identifying the blobs, grouping together adjacent pixels and representing them using a good data structure [5]. The size of the blob is not the issue. Even though the blob consists of numerous people it will still be represented as one blob. This can give rise to other problems such as with the fragmented blobs which will make one blob appear as many [7]. If the background subtraction is configured correctly, it should be relatively rare that a blob will be very fragmented. Therefore, all individual blobs, consisting of more persons will be represented as one blob in an appropriate data structure. Simple properties of a blob can be its position, width and height. A good way to combine position with width and height is to keep information about the coordinates of the smallest X and Y value, as well as the largest X and Y values [8]. In this way, one can easily know the position of the blob. It can find the width and height by subtracting the smallest X coordinate from the largest and similarly for the Y coordinates. The limitation of this approach is that the blob's shape will be known only by a rectangular approximation of the area surrounding it. This is shown in Figure 7. The point of origin for the blob is the smallest (X,Y) coordinate.

When blobs can easily be recognized one needs some sort of a data structure to hold all of the blob instances [9]. Trying to organize them so that one will quickly match the blobs from older frames to newer ones is a good idea. To do this in a fast and effective way one could use some advanced structure like a skip list to categorize them by, for example area or direction. The overhead of this method and the varying sizes of blobs on each location

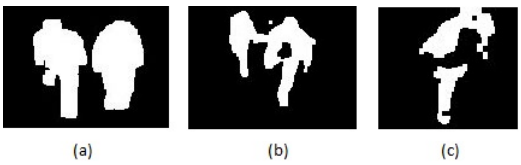

FIGURE 7. Bounding rectangle on a detected person in the video.

make this an ineffective way to compare blobs quickly. The fastest way to process the blobs is to simply put them into a list as they appear. This means actually categorizing them by their X and Y coordinates since the blob image is systematically processed starting at coordinates (0,0). We use an easy manageable list so that blobs could be quickly added, merged and deleted. Since the amount of blobs varies with time a dynamic list is required. With these reasons in mind, we choose a linked list in order to keep track of all the blobs.

2.3. **Blob Tracking and Counting.** The last step in order to count persons moving underneath the camera is to see whether a blob passes over a previously defined *boundary* that is basically a one pixel wide horizontal line across the middle of the video frame. The idea is to count a person who travels from the top towards the bottom of the frame as *IN* if and only if it passes over that line and count a person who travels from the bottom towards the top as *OUT*, again, only if it passes over that line. Some blobs only move under the camera and never enter or exit the store. These are in most cases not counted. The application also retains the track described by a blob in order to identify each unique blob across successive frames. A frame with blobs in a previous frame will indicate the blobs movement [6, 4]. If we were to count people going through a large entrance at the top of the image, knowing the path of the blobs will show that the middle blob can be counted as entering, the rightmost blob as exiting and the leftmost blob should not be counted. Analyzing the path of the blobs is the goal of blob tracking [2]. Blob tracking was performed with the help of the cvBlob library [10] which labels detected blobs in order to help tracking them from one frame to another. Figure 8 shows the algorithm applied on a video sequence.

3. Results and Conclusions

We used two video sets in order to statistically determine some of the parameters for the detection process. Then we applied heuristics for improving these parameters on video sequences of a few hours. This resulted in an accuracy of over 90% for the blob detection process. In the two initial video sets we detected 12 out of 13 people blobs. From the 20 people blobs that appear

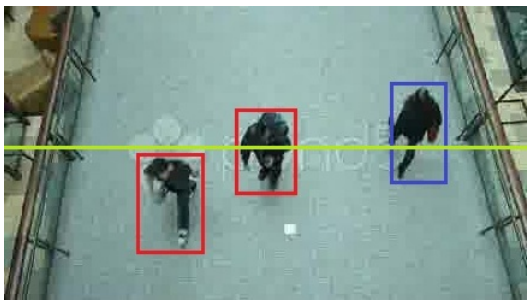

FIGURE 8. People detected as blobs, tracked and counted as *in* and *out*.

in the second video sample, only 1 remains undetected. On the experimental video sequences the counted blobs reached an accuracy of 92%. Out of 3000 people passing only 40 where incorrectly detected - mostly because they passed in very compact groups in front of the camera. The results also showed that segmentation can be the most difficult stage in blob detection because its success depends on the quality and brightness of the image generated by the illumination conditions.

In this paper we propose an algorithm for real-time detection of objects and their moving direction in a camera video sequence. Out method allows for bidirectional counting of moving objects as compared to other exiting methods. For this current version of the application, the blob detection process is done by scanning, in a binary image, pixel by pixel until a white pixel is found. When the white pixel is discovered, we search the area around it in all four directions in order to find other white connected pixels. This method can be very resource intensive. A future version of the application will scan areas instead of pixels. Dividing the frame in a number of areas, like a grid, and perform the search on the grid areas should allow for an important reduction in the CPU workload.

REFERENCES

[1] Araujo, H., Mendonca, A., M., Pinho, A., J., Torres, M. I.: Pattern Recognition and Image Analysis, Springer, Portugal, 2009

[2] Comaniciu, D., Kanatani, K., Mester R., Suter, D.: Statistical methods in video processing, Springer, Prague, Czech Republic, 2004

[3] Huang, D., Jo, K., H., Lee, H., H., Kang, H., J., Bevilacqua, V.: Emerging Intelligent Computing Technology and Applications, Springer, Ulsan, South Korea, 2009

[4] Lindeberg, T.: Scale-space theory in computer vision, Kluwer Academic Publishers, Netherlands, 1994

[5] Niels, H., Niels da Vitoria L.: Visual event detection, Kluwer Academic Publishers, USA, 2001

[6] Langdon, P., Clarkson, J., Robinson, P.: Designing Inclusive Futures, Springer, London, UK, 2008
[7] Louban, R.: Image processing of edge and surface defects, Springer, Berlin, Germany, 2009
[8] Stiefelhagen, R., Garofolo, J.: Multimodal Technologies for Perception of Humans, Springer, Berlin, Germany, 2006
[9] Sunderam, V., S., Dick van Albada, G., Sloot, P., M., A., Dongarra, J., J.: Computational Science ICCS 2005 Part One, Springer, Atlanta, USA, 2005
[10] http://code.google.com/p/cvblob/

Babes Bolyai University, Faculty of Mathematics and Computer Science, Cluj Napoca, Romania
E-mail address: dadi@cs.ubbcluj.ro