Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Detecting and counting people using real-time directional algorithms implemented by compute unified device architecture

Yasemin Poyraz Kocak[a], Selcuk Sevgen[b,*]

[a] Istanbul University, Vocational School of Technical Sciences Department of Computer Programming, 34850 Istanbul, Turkey
[b] Istanbul University, Faculty of Engineering Department of Computer Engineering, 34850 Istanbul, Turkey

ARTICLE INFO

Article history:
Received 2 May 2016
Revised 22 July 2016
Accepted 26 August 2016
Available online 8 March 2017

Keywords:
Graphic processing unit
Compute unified device architecture
Image processing
People counting

ABSTRACT

This paper implements a real-time and directional counting algorithm using the Graphic Processing Unit (GPU) Programming for the purpose of detecting and counting people. We use the Compute Unified Device Architecture (CUDA) as the environment of the GPU programming. The proposed algorithm is implemented for detecting and counting people employing the single virtual line and two virtual lines, respectively, using three video streams and two GPU graphic cards GeForce GT 630 and GeForce GTX 550Ti. We first test the video streams on the algorithm by using GeForce GT 630 together with applying the single virtual line and two virtual lines, respectively. Then, we repeat the same procedures for the GPU graphic card GeForce GTX 550Ti. The obtained experimental results show that our proposed algorithm running on GPU can be successfully programmed and implemented for people detecting and counting problems.

1. Introduction

In recent years, image processing has found many applications areas such as medicine, security, military, meteorology, etc.

People detecting and counting is an important concept in image processing. In recent years, many researchers have dealt with the problem of detecting and counting people [1–9].

People detecting and counting problem can be studied by using two main approaches. The first approach is the line of interest (LOI), in which a camera is fixed on a virtual line and people crossing this line are counted [10,11]; the second approach is the region of interest (ROI) [12,13], in which people in a certain area are counted. In this paper, we use the LOI approach as the video streams are captured from a high-position camera.

In recent years, a great deal of results concerning with the implementation of algorithms for people detecting and counting have been proposed. In [1], the automatic bidirectional counting algorithm was proposed for the pedestrian flow. The authors of [1] use the GMM algorithm for background division, apply the morphological processes to reduce the noise, and exploit the component labelling method for distinguishing the connected pedestrians. They also use the geometric properties such as location, shape and centre of mass (centroid) for detecting and counting processes.

In [2], a different approach is proposed for detecting and counting problem in which the camera is located to a higher position for using the up-down perspective, which reduces the possibility of detecting closer people as a single person. This study of [2] is basically considered for a crowded human population. They use the crowd segmentation and person segmentation algorithms together with the frame difference, morphological processing, region growing and color-based body compensation processes.

In [3], a new approach is presented for people counting in a crowded environment. This method is implemented in three steps. In the first step, the foreground pixels are determined from background with motion-detection algorithm. The second step is the foreground information fusion where the foreground pixels obtained by different camera perspectives are combined. The third step is the detecting and counting of people.

There have been some attempts to solve the detecting and counting problem by using infrared imaging, which has a high cost in terms of hardware implementation. However, in [4], the detecting and counting problem has been achieved using infrared imaging with low hardware cost.

In [5], the people counting process in a crowded environment is implemented by the energy-based model where in each image frame, the histograms are obtained on the X and Y axis, respectively. Using these processes, the probability distribution of foreground object is calculated and the crowd entropy value is determined. In the method of [5], a fixed threshold value from the histograms is determined, which enables one count not only fixed directional people but also abnormal motioning people.

In [6], the detecting and counting problem is carried out with an algorithm which is realized in three steps. In the first step, the

* Corresponding author.
E-mail address: sevgens@istanbul.edu.tr (S. Sevgen).

http://dx.doi.org/10.1016/j.neucom.2016.08.137
0925-2312/© 2017 Elsevier B.V. All rights reserved.

background subtraction is formed, then, the Expectation Maximization (EM)-based method is considered for the clustering. In the final step, using the clustering properties, they make the detecting and counting of people.

Ref. [7] presents a new approach for detecting and counting people moving to different directions without any object segmentation and tracking process. In this method, firstly, the crowd is segmented with the mixture of dynamic-texture motion model. Then, from each segmented field, the holistic low level features are subtracted. Finally, the number of people is estimated with Bayesian Regression Mode.

In [8], people counting is implemented by a hybrid radial basis function neural network while [9] counts people with a field-programming gate array (FPGA)-based a low-level head-detection method.

There are a only few papers in the literature dealing with the problem of detecting and counting people with GPU programming. For instance, in [14], the objects are counted in a series of still images instead of video streams, which differs from our approach as we use video streams.

The main objective of this paper is to implement a real-time and bi-directional people detecting and counting algorithm where the main problem is considered to be the background subtraction. In the recent literature, various background subtraction algorithms such as Difference Algorithm [15,16], Running Gaussian Average [17], Temporal Median Filter [18], Gaussian Mixture Model (GMM) [19], Kernel Density Estimation (KDE) [20] and Kalman Filter [21] have been proposed. In our studies, we will employ the Difference Algorithm.

Since image processing applications require a huge computational power, image processing is still a challenging problem for classical single-thread computer architectures. On the other hand, the existing image processing techniques and methods may not be capable of processing of images of huge data [22]. Therefore, parallel computing techniques and methods are of great interest in solving image processing problems. In this context, graphic processing unit (GPU) programming, providing alternative approach to designing future multi-core processor architecture, is an important tool in the concept of parallel computing techniques. In this paper, we will implement a real-time and directional counting algorithm using the Graphic Processing Unit (GPU) Programming for the purpose of detecting and counting people. The CUDA C, the programming language of Compute Unified Device Architecture (CUDA), is formed by using the expansions of C and C ++ programming languages. We should point out here that the parallel programming functions in a CUDA C are called kernels. A CUDA C program executes serial codes on the CPU and executes parallel kernels on the GPU. The programmer organizes these threads into a hierarchy of thread blocks and grids [23,24]. When compared to the central processing unit (CPU), GPU has more cores and transistors in the kernel. This makes GPU more effective than CPU parallel processing problems. In this study, we use three video streams taken in day-light. These video streams have different resolutions and length and the people in these video streams are moving with a fixed background.

The rest of the paper is organized as follows: in Section 2, the steps of the people counting algorithms are explained; in Section 3, the experimental results are presented and comparisons are made and finally in Section 4, the conclusions are given.

2. Proposed people counting algorithm

In order to implement an algorithm by CUDA, it is necessary to perform the required steps in parallel. The parallel implementation of the detecting and counting algorithm is given in Fig. 1.

We will now explain the steps of our algorithm.

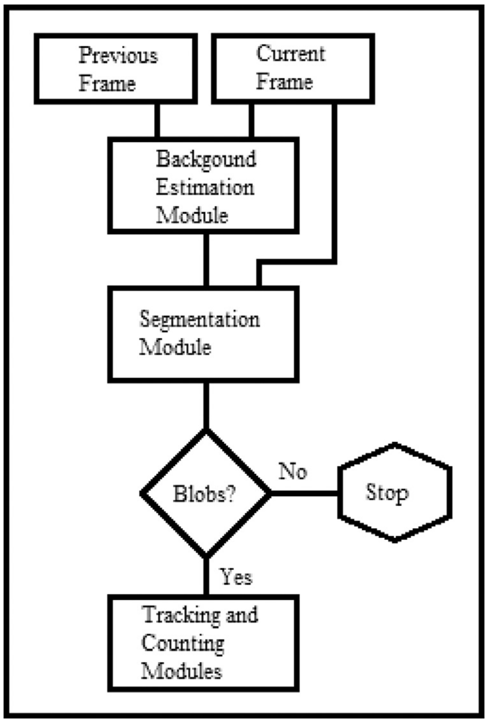

Fig. 1. The detecting and counting algorithm flowchart.

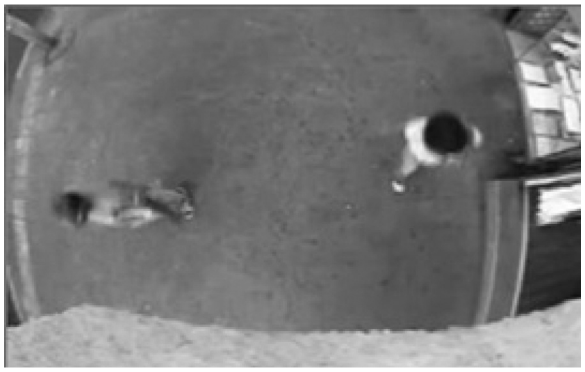

Fig. 2. An example of gray-scale video frame (Captured from the first video stream).

2.1. Capturing images

The function, capturing images, is executed in real-time and is run on the host CPU. The video stream is acquired by the function cvCreateFileCapture() in OpenCV. By the use of this function, the properties of each video stream such as the number of the frames and height-width of the frames are obtained. All steps of the algorithm in Fig. 1 are executed iteratively during each video stream.

2.2. Gray-scale transform

Each image of every video streams is transformed into the gray-scale by using the *R,G,B* (0–255) values which are acquired from each pixel using the following equation:

$$Gray - Scale = R * 0.3 + G * 0.6 + B * 0.1 \quad (1)$$

All of the pixels are transformed into the gray-scale simultaneously with the parallelly running rgb-to-grayscale kernel. In Fig. 2, an example of a gray-scale video frame is depicted.

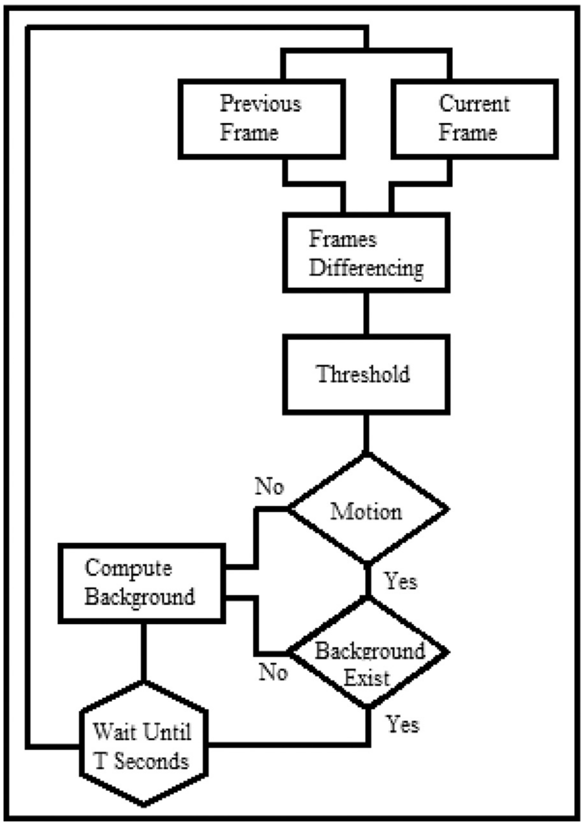

Fig. 3. The frame difference algorithm flowchart.

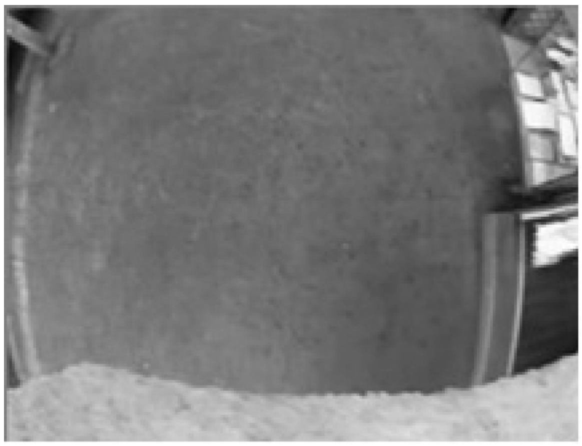

Fig. 4. An example of background image frame.

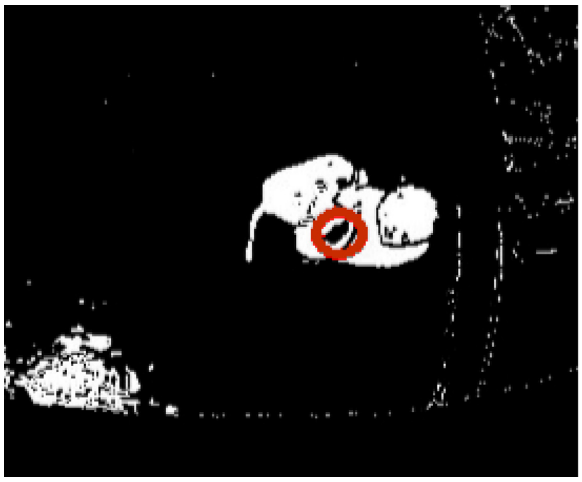

Fig. 5. An example of noisy frame.

2.3. Background substraction

Since the background subtraction process affects the success rate of the algorithm it is an important step in the people counting algorithm. In our applications, the background subtraction process is based on people detection in foreground. As we have pointed out in the previous sections, there are many different background subtraction algorithms in the literature. Obviously, the success of the algorithm crucially depends on the background subtraction process. A better background subtraction process will result in better foreground people images. For the sake of a precise comparison of our results with the previous literature results, we use the Difference Algorithm for background subtraction process. The flowchart of the Difference Algorithm is given in Fig. 3 [25,26].

In this algorithm, we first use the mean filter method to minimize the noise arisen from the camera and the light changes.

The average of the pixel values of all image frames, which are in the same position in the frame, is calculated, and these average values represent the background image frame (Fig. 4).

Then, we establish new image by taking the absolute value differences of consecutive image frames. The non-zero values in this new image indicates a motion. In some cases, even if there is no motion in the image, due to the quality of the camera and light changes may cause as if there is a motion in the image. In order to avoid this case, we use a threshold. If the difference value is greater than the threshold then this pixel is considered as a foreground and the pixel value is set to 1, otherwise it is considered as a background and the pixel value is set to 0.

2.4. Morphological operations

The threshold values determined to constitute binary images cannot eliminate all noises. Therefore, we use opening operations in the algorithm. As an example of this operation is shown in Fig. 5 as a red hole. This opening operation has two operators which are the erosion and dilation. The purpose of erosion is to shrink borders of foreground objects. Hence, the foreground objects become smaller and the holes in the objects become larger. The dilation operation aims to expand the borders of the foreground objects. Thus, the foreground objects become larger and the holes in the objects become smaller or disappear completely. Since small objects are eliminated by erosion, they cannot appear in dilation operation again.

The equations of erosion and dilation operations are defined as follows [25]:

Erosion equation:

$$\epsilon_B(X) = X \ominus B = \{x \backslash B_X \subset X\} \tag{2}$$

Dilation equation:

$$\delta_B(X) = X \oplus B = \{x + b \backslash b \in B, x \in X\} \tag{3}$$

where X is a subset of ϵ and B is a structural element.

We apply respectively the erosion and dilation operators 3×3 structural elements to the images. The obtained results are shown in Figs. 6 and 7.

Erosion: 3×3 structural elements are moved on foreground pixels with having the values of 1. If all values of 3×3 neighborhood of foreground pixel are 1, then the current pixel value does not change (Fig. 6).

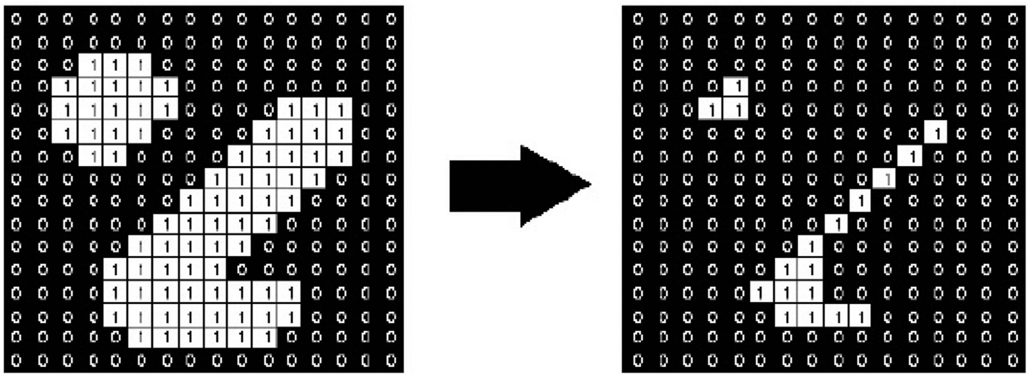
Fig. 6. Eroded image.

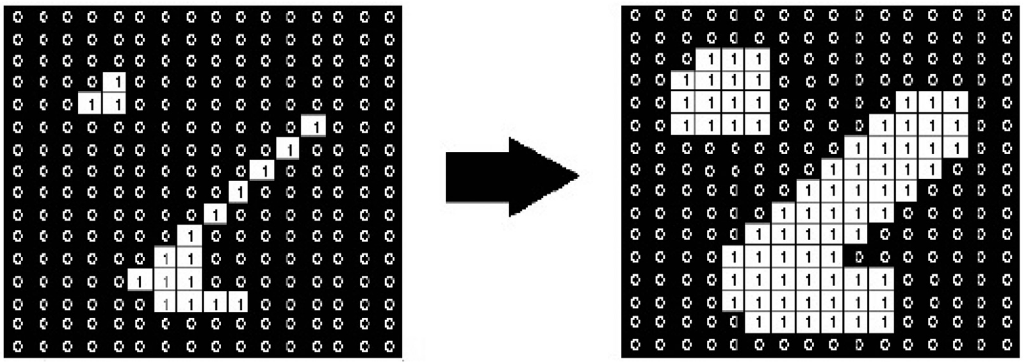
Fig. 7. Dilated image.

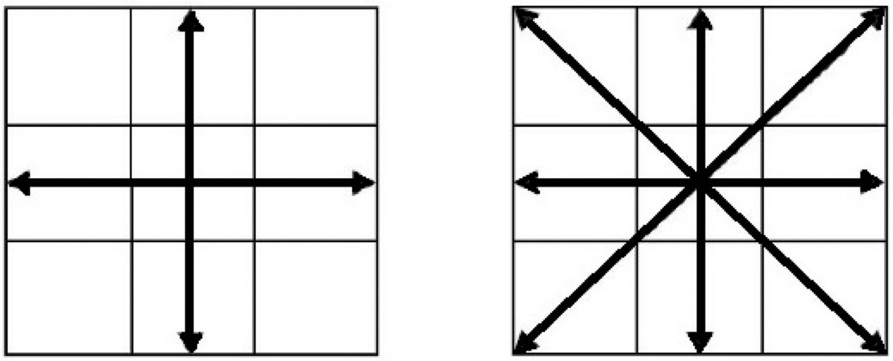
Fig. 8. Two rules of adjacent pixels.

Dilation: 3 × 3 structural elements are moved on all pixels. If at least one of the values of 3 × 3 neighborhood of foreground pixel is 1, then the current pixel value is set to 1 (Fig. 7).

2.5. Blob analysis

In order to track objects, it is necessary to define all objects and their properties. This operation is called blob analysis. In this study, the blob analysis is used for object tracking. In order to determine whether the object is a person or not, an average size for the people to be detected is decided according to the position of camera. In the blob analysis algorithm, there are two different ways of connecting the pixels. One way is to connect vertical-horizontal adjacent pixels. The other way is to connect diagonal adjacent pixels (Fig. 8).

The performance of blob analysis basically depends on the segmentation process of the images. An inappropriate segmentation algorithm may detect noises on the images as the blobs. Therefore, it is important to make the use of the central position, convex-hull, height and width of the blob in the algorithm.

2.6. Counting process by using a single virtual line

The counting process is performed by a single virtual line separating the entrance and exit area completely. At least two successive images are needed to count people. If a blob in an image is below the virtual line and it is above adjacent image, then, the counter is increased by one [16,27–29].

2.7. Counting process by using two virtual lines

In some cases, people can walk very close to each other. The closeness can be examined in four different ways when they are in the camera's field of view. These cases are explained in Fig. 9 [27,30,31]:

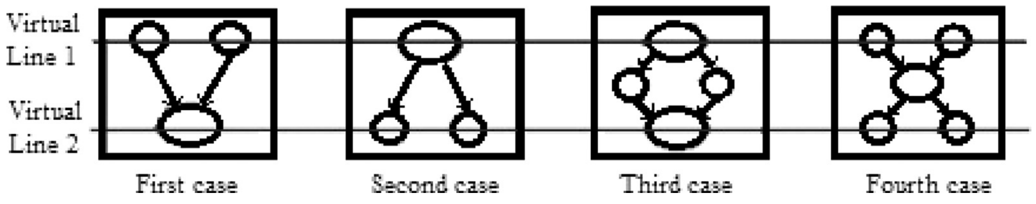

Fig. 9. The positions of people.

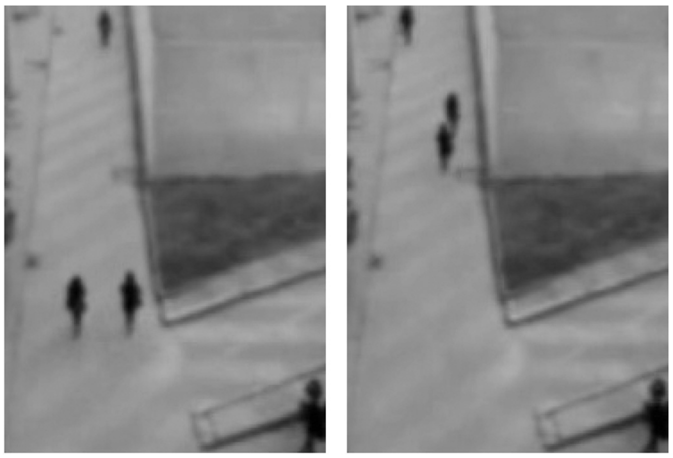

Fig. 10. Example images of first case (Captured from the third video stream).

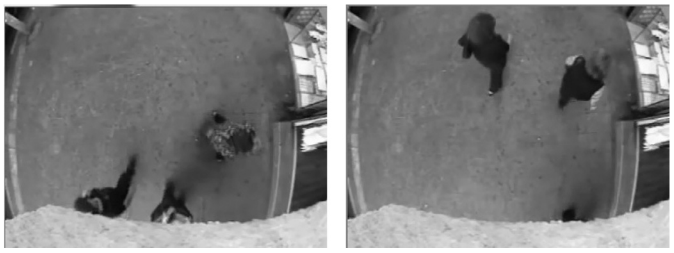

Fig. 11. Example images of second case (Captured from the first video stream).

- First case: people can get closer to each other while they are walking separately (Fig. 10).
- Second case: people can separate from each other while they are walking together (Fig. 11).
- Third case: people can walk closely, they can separate, and they can get closer.
- Fourth case: people can walk separately, get closer to each other and, then, separate again.

These cases in Fig. 9 are for both top-down and bottom-up directions. The top-down and bottom-up human passing through are checked on Virtual Line 1 and Virtual Line 2, respectively. Hence, the classification of close people as a single person is avoided.

3. Experimental results

In this section, we implement the people detecting and counting algorithms for three video streams. The properties of the video streams are given in Table 1.

In the implementation of the algorithms, we use GPU graphic cards, GeForce GT 630 2 GB and GeForce GTX 550 Ti, whose properties are shown in the Table 2.

In Tables 3 and 4, the running times of each algorithm are shown for each step.

Table 1
Properties of three video.

	Number of frame	Duration (s)	Resolution
1. Video	2456	98	288×384
2. Video	3181	106	320×448
3. Video	8999	300	320×240

Table 2
Properties of two GPU graphic cards.

Device properties	GeForce GT 630 2 GB	GeForce GTX 550 Ti
Total memory	2048 MB	1024 MB
GPU clock rate	1.62 GHz	1.82 GHz
Total texture size	65536	65536
Total constant memory	65536	65536
Total shared memory per block	49152	49152
Warp size	32	32
Number of thread	1024	1024
Dimension of a thread block	< 1024, 1024, 64 >	< 1024, 1024, 64 >
Dimension of a grid	< 65535, 65535, 65535 >	< 65535, 65535, 65535 >
Number of CUDA core	96	192

Table 3
Running times of algorithms of videos on GeForce GT 630 2 GB.

	1.Video (ms)	2.Video (ms)	3.Video (ms)
Gray-scale transformation	0.067	0.099	0.052
Background estimation	1.955	2.280	0.708
Erosion	0.099	0.140	0.076
Dilation	0.211	0.298	0.166
Blob analysis and counting Using a virtual line	32.087	164.287	31.166
Blob analysis and counting Using two virtual line	64.649	646.925	46.527

Table 4
Running times of algorithms of videos on GeForce GTX 550 Ti.

	1.Video (ms)	2.Video (ms)	3.Video (ms)
Gray-scale transformation	0.030	0.041	0.024
Background estimation	0.880	1.571	0.502
Erosion	0.045	0.063	0.035
Dilation	0.095	0.135	0.076
Blob analysis and counting Using a virtual line	21.901	70.010	21.776
Blob analysis and counting Using two virtual line	47.824	468.622	39.621

From the results of Tables 3 and 4 we can conclude that the running times are directly proportional to the resolution. The total running times for each algorithm are also given in Tables 5 and 6.

The same algorithms have been run in the MATLAB platform, and the total processing times have been calculated as follows: 31 min for first video stream, 93 min for second video stream and 375 min for third video stream. Based on the results obtained in MATLAB and in GPU programming, for total processing times, it can be concluded that GPU programming technique for people detecting and counting is faster than the classical sequential programming.

Table 5
The total running times (processing time + video duration) on GeForce GT 630 2 GB.

	One virtual line (s)	Two virtual line (s)
1. Video	84.537 + 98	164.505 + 98
2. Video	531.557 + 106	2066.829 + 106
3. Video	280.480 + 300	427.713 + 300

Table 6
The total running times on GeForce GTX 550 Ti.

	One virtual line (s)	Two virtual line (s)
1. Video	56.367 + 98	120.034 + 98
2. Video	228.459 + 106	1496.444 + 106
3. Video	201.694 + 300	362.281 + 300

For the one virtual line algorithm, we have correctly counted 22 people out of 31 people for first video stream, 29 people out of 36 people for second video stream and 29 people out of 46 people for third video stream. For the two virtual line algorithm, we have correctly counted 24 people out of 31 people for first video stream, 30 people out of 36 people for second video stream and 34 people out of 46 people for third video stream. From the above analysis and results, we can conclude that the success rate in detecting and counting people in our applications basically depend on the distance between the counted people, the resolution of the video streams. In references [4,32] and [33] the success rates for counting people are obtained to be 95 %, 87 % and 90 %, respectively. The best success rate of our results is about 80 %. When only the success rates are considered, the results of [4,32] and [33] seem to be better than our results. However, [4,32] and [33], the success rates are determined without mentioning the speed of the their systems and the total processing times. Therefore, we can only express the success rates of the previous results since we are not able to make any precise comparisons between the results. We should mention here that our current paper is an extensively improved version of [34].

4. Conclusion

In this paper, we have implemented a real-time and directional counting algorithm using the Graphic Processing Unit (GPU) Programming for the purpose of detecting and counting people. Our approach has proved to be a faster working system and have less total processing times in detecting and counting people. We have used two different GPU graphic cards to demonstrate the performance of the GPU graphic cards on the classification problem. It has been revealed that the properties of the GPU graphic cards play a key role in detecting and counting problems. We have also observed that the resolution of the video streams has a great impact on the performance of the implemented algorithm.

In future work, the method and the algorithm we have proposed for detecting and counting people can be improved by new algorithms to be implemented on GPU graphic cards. Another possibility to improve our presented results may be the use of more virtual lines other than single virtual line and two virtual lines.

References

[1] J.L. Raheja, K. Sishir, J. Pallab, L. Solanki, A robust real time people tracking and counting incorporating shadow detection and removal, Int. J. Comput. Appl. 46 (4) (2012) 51–58.
[2] C.C. Ho, C.T. Yi, W.D. Jinn, C.T. Jie, A cost effective people-counter for a crowd of moving people based on two-stage segmentation, J. Inf. Hiding Multimed. Signal Process. Ubiquitous Int. 3 (1) (2012) 12–23.
[3] A. Mousse, C. Motamed, E.C. Ezin, People counting via multiple views using a fast information fusion approach, Multimed Tools Appl. (2016), doi:10.1007/s11042-016-3352-z.
[4] I.J. Amin, A.J. Taylor, F. Junejo, A. Al-Habaibeh, R.M. Parkin, Automated people counting by using low-resolution infrared and visual cameras, Measurement 41 (2008) 589–599.
[5] G. Xiong, J. Cheng, X. Wu, Y.L. Chen, Y. Ou, Y. Xu, An energy model approach to people counting for abnormal crowd behavior detection, Neurocomputing 83 (2009) 121–135.
[6] Y.L. Hou, K.H. Pang, People counting and human detection in a challenging situation, IEEE Trans. Syst. Man Cybern.-Part A: Syst. Hum. 41 (1) (2011) 24–33.
[7] A.B. Chan, N. Vasconcelos, Counting people with low-level features and Bayesian regression, Trans. Image Process. 21 (4) (2012) 2160–2177.
[8] D. Huang, T.W.S. Chow, A people-counting system using a hybrid RBF neural network, Neural Process. Lett. 18 (2003) 97–113.
[9] A.G. Vicente, I.B. Muñoz, P.J. Molina, J.L.L. Galilea, Embedded vision modules for tracking and counting people, IEEE Trans. Instrum. Meas. 58 (9) (2009) 3004–3011.
[10] J. Barandiaran, B. Murguia, F. Boto, Real-time people counting using multiple lines, in: Proceedings of the 9th International Workshop on Image Analysis for Multi. Interactive Services, 2008, pp. 159–162.
[11] S. Yu, X. Chen, W. Sun, D. Xie, A robust method for detecting and counting people, in: Proceedings of the International Conference on Audio, Language and Image Processing ICALIP08, 2008, pp. 1545–1549.
[12] D. Ryan, S. Denman, C. Fookes, S. Sridharan, Crowd counting using multiple local features, in: Proceedings of the Digital Image Computing: Techniques and Applications DICTA09, 2009, pp. 81–88.
[13] A.G. Vicente, I.B. Munoz, P.J. Molina, J.L.L. Galilea, Embedded vision modules for tracking and counting people, IEEE Trans. Instrum. Meas. 58 (9) (2009) 3004–3011.
[14] K. Bjerge, Dynamic counting objects in images optimized for data-parallel computing (20097553), Aarhus University, Department of Computer Science, 2009.
[15] E. Guler, B. Gecer, in: People counting, 2015. http://www.ebubekirguler.com/goruntu-isleme-yontemleri-ile-insan-sayma/.
[16] S.A. El-Azim, I. Ismail, H.A. El-Latiff, An efficient object tracking technique using block-matching algorithm, in: Proceedings of the Nineteenth National, Radio Science Conference, 2002, pp. 427–433.
[17] C. Wren, A. Azarhayejani, T. Darrell, A.P. Pentland, Pfinder: real-time tracking of the human body, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 780–785.
[18] B.P. Lo, S.A. Velastin, Automatic congestion detection system for underground platforms, in: Proceedings of the International Symposium on Intelligent Multimedia, Video and Speech Processing, 2001, pp. 158–161.
[19] C. Stauffer, W.E.L. Grimson, Adaptive background mixture models for real-time tracking, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, 2, 1999. 246–25
[20] A. Elgammal, D. Hanvood, L.S. Davis, Nonparametric model for background subtraction, in: Proceedings of the European Conference on Computer Vision, 2000, pp. 751–767.
[21] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russell, Towards robust automatic traffic scene analysis in real-time, in: Proceedings of the International Conference on Pattern Recognition, 1994, pp. 126–131.
[22] S. Saxena, N. Sharma, Parallel image processing techniques, benefits and limitations, Res. J. Appl. Sci. Eng. Technol. 12 (2) (2016) 223–238.
[23] NVIDIA CUDA. (2007–Last updated 2015) C Programming Guide Version 6.0, 2007, http://docs.nvidia.com/cuda/cuda-c-programming-guide/.
[24] NVIDIA CUDA. (2007–Last updated 2015) CUDA C Best Practices Guide, 2007, http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/.
[25] D. Lefloch, Real-time people counting system using video camera (Master thesis), Gjøvik University College, 2007.
[26] D. Gutchess, M. Trajkonic, E. Cohen-Solal, D. Lyons, A.K. Jain, A background model initialization algorithm for video surveillance, in: Proceedings of the 8th IEEE International Conference on Computer Vision, 2001, pp. 733–740.
[27] T.H. Chen, T.Y. Chen, Z.X. Chen, An intelligent people-flow counting method for passing through a gate, in: Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems, 2006, pp. 573–578.
[28] T.H. Chen, Y.F. Lin, T.Y. Chen, Intelligent vehicle counting based on blob analysis in traffic surveillance, in: Proceedings of the IEEE International Conference on Innovative Computing, Information and Control (ICICIC-07), 2007.
[29] I. Karaulova, P. Hall, A. Marshall, A hierarchical model of dynamics for tracking people with a single video camera, in: Proceedings of the British Machine Vision Conference, 2000, pp. 352–361.
[30] C. Chen, Y. Chang, T. Chen, D. Wang, People counting system for getting in/out of a bus based on video processing, in: Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications, 2008, pp. 565–569.
[31] T. Chen, C. Hsu, An automatic bi-directional passing-people counting method based on color image processing, in: Proceedings of the IEEE 37th Annual 2003 International Carnahan Conference, 2003, pp. 200–2007.
[32] C. Tsong, C. Chao, W. Da, C. Tsang, Real-time counting method for a crowd of moving people, in: Proceedings of the Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2010, pp. 643–646.
[33] K. Yam, W. Siu, N. Law, C. Chan, Effective bi-directional people flow counting for real time surveillance system, in: Proceedings of the IEEE International Conference on Consumer Electronics, 2011, pp. 890–891.
[34] Y.P. Kocak, S. Sevgen, Real-time people counting application by using GPU programming, in: Proceedings of the 22nd International Conference on Neural Information Processing (ICONIP 2015), 2015, pp. 540–547.

Yasemin Poyraz Kocak received the M.Sc. degree in Computer Engineering from Istanbul University in 2014. She is a Ph.D. student in the same department of Istanbul University now. Currently she is a research assistant at Technical Sciences Department of Computer Programming in Istanbul University. Her research interests image processing and GPU programming.

Selcuk Sevgen is currently an Assistant Professor at the Department of Computer Engineering in Istanbul University. He received his M.Sc. and Ph.D. degrees in same department in 2003 and in 2009, respectively. His main interests are neural networks, image processing and computer vision.