Real-Time Pedestrian Detection and Tracking

Elad Roichman, Yaniv Solomon, Yair Moshe
Signal and Image Processing Laboratory (SIPL)
Department of Electrical Engineering, Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel
http://sipl.technion.ac.il

Email: yair@sipl.technion.ac.il

Abstract

This paper presents an embedded prototype system for pedestrian detection and tracking in outdoor scenes. The system uniquely combines two common techniques for motion detection - temporal differencing and background subtraction. The results of these two techniques are adaptively averaged and thresholded to result in a binary motion image. Motion regions are efficiently segmented and non-target regions are removed. Detected pedestrians are tracked based on their location in previous frames. Both motion detection and tracking were designed with an embedded environment in mind and were optimized for the C6000 family of DSPs. A fixed-point version of the human detection and tracking algorithm was implemented using the DM642 EVM board. The proposed system and algorithm were verified to be accurate and robust by a set of experiments at real scenarios. A real-time implementation for D1 resolution shows the efficiency of the algorithm and proves the suitability of the DM642 chip for such a system.

1. Introduction

Human detection and tracking has been a very active research area over the past two decades. One of the main reasons for the massive research activity in this area is its important applications, especially for surveillance and monitoring. For many of these applications a low-cost, accurate and robust detection and tracking solution is required [1][2].

Though a well studied area, detecting humans in video still holds many scientific and technological challenges. Maintaining a low false detection rate is hard mainly due to the lack of distinguishing visual features that characterize human appearance. Tracking people is not easy as well since humans move non-rigidly and can make a wide variety of body poses. Movement through cluttered areas, occlusions, shadows, lighting changes, moving of elements in the scene and more, make the situation even more difficult. For embedded applications, avoiding excessive calculations and large memory consumption is another challenge.

There are three classical approaches for moving target detection: temporal differencing, background subtraction and optical flow [1][2]. Temporal differencing attempts to detect moving regions by differencing between consecutive frames in the video sequence. It is adaptive to dynamic environments but is inaccurate in extracting all relevant moving pixels. Background subtraction attempts to detect moving regions by differencing between current image and an image that models the background of the scene. This approach is more accurate but is sensitive to dynamic scene changes such as variations in lighting. Optical flow is used to describe coherent motion of points or features between video frames. Using this technique it is possible to detect independently moving targets in the presence of camera motion.

However, optical flow incurs a high computational burden; therefore it is usually impractical for real-time implementation.

The prototype described in this paper assumes a pre-calibrated mostly static camera and attempts to track and detect walking human pedestrians. The chosen solution comprises the advantages of both temporal differencing and background subtraction by combining them. This allows accuracy in detection and tracking together with robustness to dynamic scene changes.

Section 2 outlines the proposed pedestrian detection and tracking algorithm. Sections 3 and 4 describe its implementation and results respectively. Finally, some conclusions are drawn in section 5.

2. Pedestrian Detection and Tracking

A high-level block diagram of the proposed algorithm is depicted in Figure 1. Temporal differencing is performed by subtracting each frame from its previous frame and results in a difference frame. Background subtraction is performed by background modeling and subtracting the current frame from the background image model and results in a background difference frame. A background image model is updated using a running average, calculated according to:

$$B_n(x,y) = \alpha B_{n-1}(x,y) + (1-\alpha) I_n(x,y) \qquad (1)$$

where I_n is the nth input image, B_n is the nth background image model, and α is a learning factor in the range [0, 1]. The background model at each pixel location is based on the pixel's recent history. α is adaptively computed for every pixel of the background image model using the corresponding value in the difference frame. Larger difference results in a higher value of α, thus giving more weight to background pixels, and vice versa. The adaptive tuning of α makes the background model robust to fast changes. The background image model is reset when motion is detected in more than 80% of the difference frame pixels. This enabled correct handling in case of unexpected movement of the camera or sudden scene changes. To improve robustness even further, the difference frame and background difference frame are averaged according to the average difference level of the pixels in the difference frame in which movement occurred, as formulated by:

$$Diff(x,y) = w \cdot BackDiff(x,y) + (1-w) \cdot ConsFramesDiff(x,y) \qquad (2)$$

where $Diff$ is the averaged difference frame, $BackDiff$ is the background difference frame, $ConsFramesDiff$ is the difference frame between consecutive frames and w is the weight given to the background difference frame. w is calculated according to:

$$w = MinWeight + (MaxWeight - MinWeight) \cdot AverageDiffLevel \qquad (3)$$

where $MinWeight$ is the minimum weight given to the background difference frame, $MaxWeight$ is the maximum weight given to the background difference frame and $AverageDiffLevel$ is the average difference level in $ConsFramesDiff$, normalized to the range [0,1]. The higher the average difference level is, the more weight is given to the background difference frame.

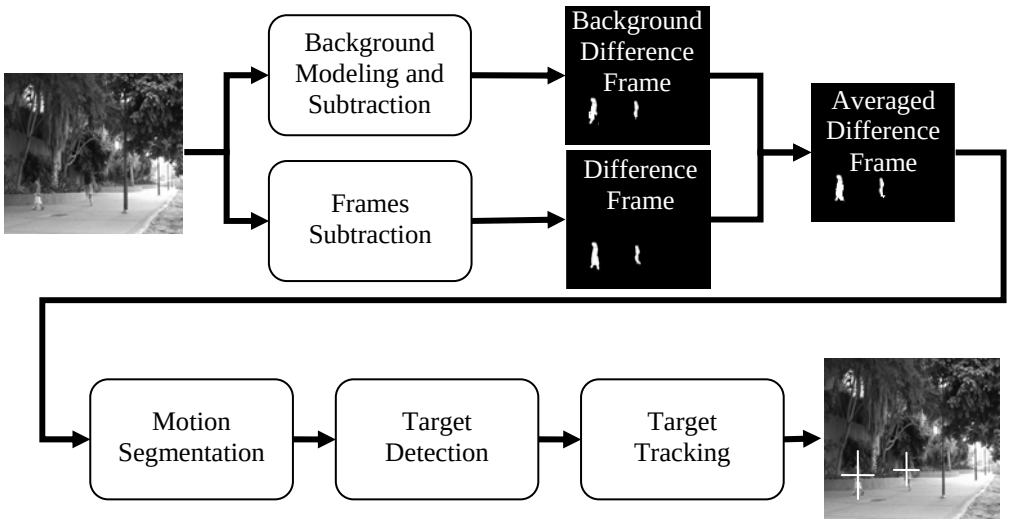

Figure 1: Pedestrian detection and tracking algorithm.

Figure 2 demonstrates the robustness of the background modeling. In the sequence shown in this figure, the background is static and suddenly a car and two men are entering the scene and cover a large part of the image. Nevertheless, the background image model remains faithful to the true scene background.

The averaged difference frame is used for segmentation of moving pixels. First, the averaged difference frame is low-pass filtered. In this manner undesirable small segments attributed to isolated moving pixels and to isolated edge pixels are removed. Then, a motion threshold is adaptively computed using the averaged difference filtered frame histogram, as described in [3]. A linear line is constructed between the maximum of the image histogram and its minimum. The threshold is two times the brightness value for which the distance to the drawn line is maximal. The threshold is applied to the averaged difference frame, resulting in a binary image representing pixels in motion.

Target detection is performed by connected component labeling and removal of non-target pixel segments. The connected component labeling algorithm is based on the technique described in [4] that uses only simple localized memory actions. These attributes make it suitable for real time systems. The removal of non-target pixel segments eliminates labeled components that are considered non-human targets due to their size or height-width ratio.

Target tracking is accomplished by calculating possible locations at which the target could move to according to its motion vector. A weighted SAD (Sum of Absolute Differences) measurement around each of the predicted target locations is used for target tracking. A weighted SAD measurement is used since target outline pixels have lower probability to be correctly detected compared to target non-outline pixels. Therefore lower weight is assigned to SAD values at the target outline pixels while higher weight is assigned to SAD values at the target non-outline pixels. This is accomplished by applying an elliptical weighting mask to the target SAD values. At the end of the tracking process a new target location is set at the location for which the weighed SAD is minimal.

(a) (b)

(c) (d)

Figure 2: Demonstration of background modeling robustness. (a) Frame N-10 (b) Frame N (c) Frame N+10 (d) Background image model for frame N.

3. Implementation

A prototype system was implemented using the DM642 EVM board [5]. This board comprises a DM642 chip [6] running @ 720 MHz, level-one program and data caches of 16KB each, 256KB of internal SRAM which can also be configured as second-level unified program/data cache, video input and output ports and many other peripherals.

The algorithm was designed and the code implementing it was written in consideration of its designated platform, hence a DSP-friendly algorithm was chosen and implemented. An example of a DSP-friendly implementation is the background modeling phase. A moving average scheme was chosen due to its low computational complexity, low memory consumption, and accurate results. Another example for a DSP-friendly implementation is the connected component labeling phase, characterized by linear running time complexity and local sequential memory access. It is especially designed to minimize CPU stalls due to its sequential operation and incurs only a low memory complexity.

Most parts of the pedestrian detection and tracking algorithm produce accurate results even if not performed for every input frame. In order to divide the workload more evenly in time, many parts of the algorithm are performed in several iterations where each iteration is performed only for part of the input frame. For example, in each iteration the background model is updated based only on part of the input frame. The number of iterations can be easily controlled. This method of dividing computation tasks across several iterations allows

better flexibility under tight time constraints and is being used throughout the implementation of the algorithm.

DSP-oriented coding style was used in order to help the compiler in producing a highly optimized code. Branches were avoided, if possible, or carefully written. If any apriori knowledge on a branch is available, it was used for writing the branch efficiently. Divisions were performed by shift operations with numbers which are a power of 2. Frequently used function values, such as *Cos* and *Sin*, were pre-calculated and saved in tables of constants. The code was organized and written for good data locality in order to allow good cache utilization. Hence, different code snippets that use the same memory areas were moved together as long as the algorithm consistency remained intact. Small size data types were chosen so that memory usage is minimized and better parallelism is achieved. Additional parallelism was achieved by loop unrolling. *#pragma* directives were used in order to allow better compiler optimizations.

Video input/output raises many performance issues in transferring the video data in real-time. In order to deal with these performance issues, the EDMA (Enhanced Direct Memory Access) mechanism of the DM642 chip was used, hence allowing faster I/O through video peripherals without any CPU computation directly involved.

Implementation was performed in C without any assembly code. Specific functions that were identified as performance bottlenecks were specifically optimized. An example for such a function is the 2D convolution, where running time was reduced from approximately 50% of the total processing time (before optimization) to 25% (after optimization), meaning 100% improvement in the function's running time.

4. Results

Field testing the system with the DM642 EVM board and a standard video camera at different real scenarios showed accurate pedestrian detection and tracking, demonstrated in Figure 3.

The algorithm should be pre-calibrated to the specific environment and camera location in order to achieve optimal results. Calibration affects different thresholds that control the tradeoffs between correct detection rate and false detection rate. One such threshold is for the ratio between suspected target height and width. While a strict threshold removes many non-human targets, a too strict threshold may discard legitimate targets, especially at complex scenarios. One example of such a complex scenario is a scene where multiple human targets are moving together and are detected as a single moving target with non-legitimate height-width ratio. Another example, shown in Figure 4, is a pedestrian walking together with a large object (a stroller) and hence is discarded for the same reason.

Figure 5 shows the improvement in the algorithm running time achieved by optimizations. The optimized version of the algorithm runs almost x6 faster than the non-optimized version. Currently the prototype system runs at twelve D1 (720x576) frames per second. This rate is enough to achieve accurate detection and tracking and meets real- time requirements. If only one field of the interlaced input image (720x288) is used, the system runs at 14 frames per second, without any significant degradation in tracking results accuracy.

Figure 3: Pedestrian tracking example.

Figure 4: Imperfect pedestrian tracking example. The left pedestrian is detected together with the stroller and is discarded due to non-legitimate height-width ratio.

Profiling shows that most of the running time is spent on CPU stalls due to extensive use of slow external memory. Based on these results it could be estimated that with a more efficient memory organization and wise use of EDMA, algorithm running time is expected to substantially improve. Reducing the size of the low-pass filter in static scenarios is expected not to hurt algorithm accuracy while assisting in reducing running time even further.

Partitioning of the algorithm running time between its parts is shown in Figure 6. Target detection is the most computationally complex part while target tracking is relatively computationally easy.

5. CONCLUSIONS

This paper presents a fixed-point low-cost algorithm for pedestrian detection and tracking together with its implementation using the DM642 chip and EVM board. The proposed algorithm combines a dynamic background model together with temporal subtraction in order to achieve high robustness and scalability. The system was examined in different real scenarios and showed accurate pedestrian detection and tracking.

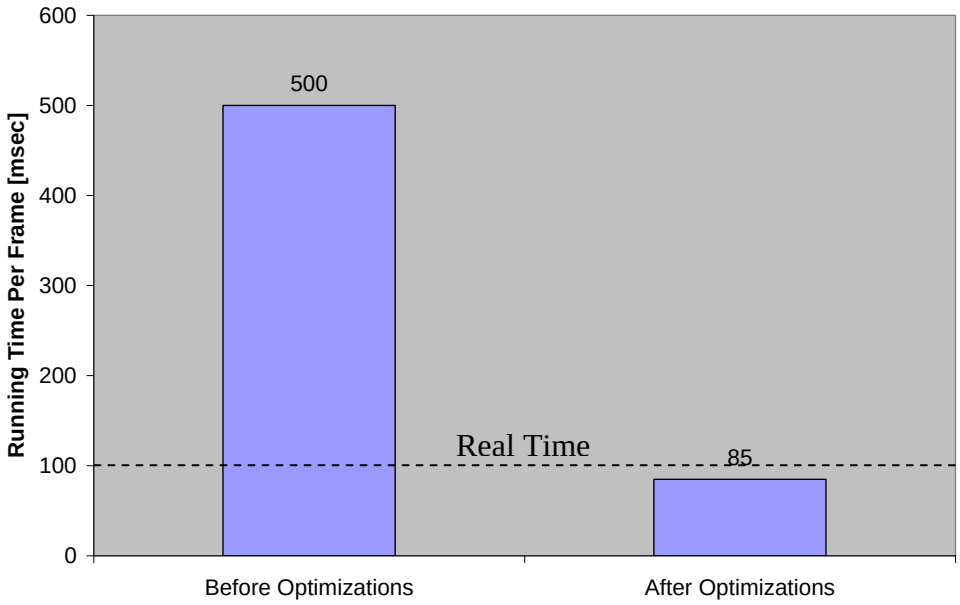

Figure 5: Algorithm running time.

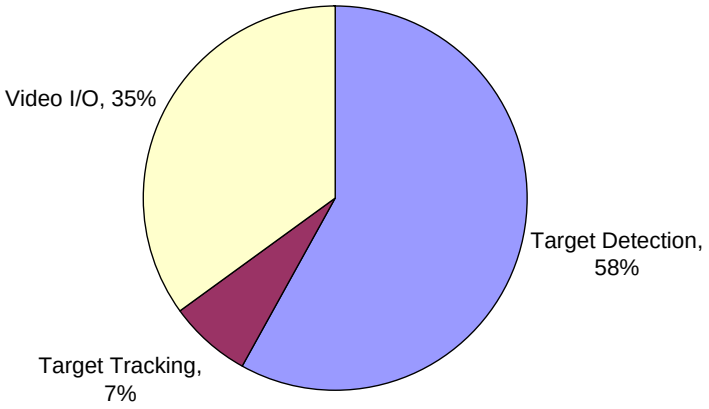

Figure 6: Partitioning of algorithm running time.

After initial tuning and optimizations, the system runs at real-time for D1 resolution with good detection and tracking results. This proves the suitability of the DM642 DSP for implementing such a system. Current performance bottlenecks are due to extensive use of external memory and non-optimal cache utilization. A substantial performance gain can be achieved by further optimizing these aspects of the system.

6. ACKNOLEDGMENTS

The embedded prototype system that was described in this paper would have not seen its final state without the help and assistance of several people.

The authors would like to thank Yuval Shabo and Roni Papo and their supervisors, Johanan Erez and Eli Appelboim, from the Vision and Image Science Laboratory (VISL) in the Department of Electrical Engineering of the Technion for supplying an initial version of the algorithm.

The authors would also like to express their special gratitude and appreciation to the staff of the Signal and Image Processing Laboratory (SIPL) in the Department of Electrical Engineering of the Technion, where this project has been initiated and performed, for their continuous support and assistance in the various aspects of creating such a system. In particular, we would like to thank Prof. David Malah for his helpful comments, Nimrod Peleg, for providing the means to create such a prototype system, Ziva Avni and Avi Rosen, for helping in all the technical aspects, enabling an excellent working environment.

This project was supported in part by Texas Instruments DSP ELITE universities program.

REFERENCES

[1] L. Wang, W. Hu, T. Tan, "Recent Developments in Human Motion Analysis," Pattern Recognition, vol. 36 (3), pp. 585–601, 2003.

[2] T. B. Moeslund, A. Hilton, V. Krüger, "A Survey of Advances in Vision-based Human Motion Capture and Analysis," Computer Vision and Image Understanding, vol. 104 (2–3), pp. 90–126, 2006.

[3] G. W. Zack, W. E. Rogers, and S. A. Latt, "Automatic Measurement of Sister Chromatid Exchange Frequency," Journal of Histochemistry and Cytochemistry, vol. 25(7), pp. 741-753, 1977.

[4] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time Connected-component Labeling Based on Sequential Local Operations," Computer Vision and Image Understanding, vol. 89 (1), pp. 1-23, 2003.

[5] Spectrum Digital, "TMS320DM642 Evaluation Module with TVP Video Decoders Technical Reference (Rev. B)," http://c6000.spectrumdigital.com/evmdm642/V3/docs/dm642evm_TechRef.pdf, 2004.

[6] Texas Instruments, "TMS320DM642 Technical Overview," http://focus.ti.com/lit/ug/spru615/spru615.pdf, 2002.