Real-Time DSP Implementation of Pedestrian Detection Algorithm Using HOG Features

Akshay Chavan
Electrical and Computer Engineering
Texas Tech University
Lubbock, USA
chavanakshay9@gmail.com

Senthil Kumar Yogamani
Advanced Driver Assistance Systems
Texas Instruments Inc.
Dallas, USA
ysenthil@ti.com

Abstract— **Pedestrian Detection is the most critical safety application in automotive driver assistance systems. Histogram of Oriented Gradients (HOG) features is known to produce the state of the art results for this application. This feature is very compute-intensive and it is difficult to achieve real-time performance by direct porting of community software like OpenCV. In this paper, we discuss an efficient DSP implementation of this algorithm and also demonstrate how architecture aware design choices can lead to huge performance improvements. The algorithm was implemented and profiled on a Texas Instruments' C674x DSP, achieving a performance of 20 fps for a VGA resolution video sequence. Compared to OpenCV's HOG function, the proposed implementation is 130X faster without a significant loss of accuracy.**

Pedestrian detection; Embedded Vision; DSP Optimization; OpenCV.

I. INTRODUCTION

A pedestrian is a moving obstacle for an automotive vehicle and a quick response time is required to detect the presence of a pedestrian on the road and react by stopping or slowing down the vehicle to avoid accidents. Driver assistance systems which detect pedestrians can aid the driver in reducing accidents. Vision based systems are commonly used for this purpose to detect pedestrians on the road or on sidewalk. Pedestrian detection or people detection is applicable in areas like surveillance, robotics, tracking players in multiplayer games like soccer to develop or detect game strategies, etc. Detecting people in surveillance videos can also be used for market research for example, in monitoring super market aisles. It is challenging to develop a real-time vision algorithm because of variations in clothing, pose, lighting conditions and dynamic backgrounds.

In this work, we develop and implement a real-time pedestrian detection system on an embedded processor with reasonable accuracy. The platform chosen is the low power and high precision fixed/floating-point TMS320C674x™ VLIW DSP.

A. Related Work

Gerónimo [1] and Gavrila [2] discussed various state of the art pedestrian detection algorithms, evaluated them for their performance in terms of accuracy and execution time. Results of the study showed that the HOG-based linear SVM approach [3] significantly outperformed in detection accuracy with the second best being the Haar features approach [4]. HOG feature based approach, though accurate, is computationally expensive. The aim of our literature review was to select an algorithm that is accurate and feasible for a real-time implementation. There have been several attempts of developing real-time pedestrian detection systems using template based techniques [5], using IR cameras for night time driver assistance [6], LIDAR based approach [7] and several HOG feature derivatives. Recent attempts making use of HOG feature derivatives in [8] and [9] obtained a performance of ~10 fps on a standard PC platform. Desktop GPU implementation in [12] achieved 40 fps.

The rest of the paper is divided into the following parts. Section II describes the standard HOG based pedestrian detection process in detail. Sections III and IV discuss the algorithmic design and DSP optimization techniques. Section V discusses the results obtained using the proposed implementation.

II. IMPLEMENTATION DETAILS OF HOG FEATURES BASED PEDESTRIAN DETECTION

Dalal [3] studied the Histograms of Oriented Gradient (HOG) based feature descriptors using Linear SVM. This algorithm produces state-of-the-art results and has become the standard feature for many object detection problems. We use the OpenCV implementation of HOG + SVM as our reference to compare performance and accuracy. Figure 2 shows a block diagram of this implementation. Image pyramid is employed to detect pedestrians at different scales. Refer to [3] for more details.

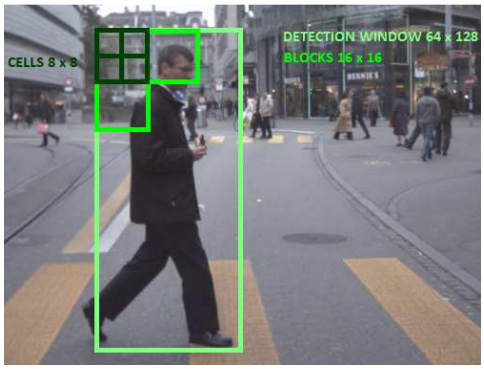

Figure 1. Illustration of 8x8 HOG cells(dark green), 16x16 HOG blocks consisiting of 4 cells(light green) and 64x128 detection window(lighter green)

For the best performance of the algorithm it is recommended to use RGB color space images where each pixel's gradient vector is chosen from the color channel with the largest norm. Optimal parameters include detection window size of 128×64 pixels, [-1, 0, 1] gradient filter with no smoothing, 9 orientation bins for the range 0-180^o, block size of 16×16 pixels containing 4 cells of 8×8 pixels, Gaussian spatial window with standard deviation of 8 pixels and L2-Hys block normalization and a linear SVM classifier. Figure 1 shows the sizes of cells, blocks and detection window which was used. Each block is weighted with a Gaussian window to give higher weights to the pixels in the center than on the edges, and the orientations are grouped into evenly spaced bins over 0-180^o to form an orientation histogram. Histograms from each block are normalized using L2Hys norm to correct for the local illumination changes in the image. The final descriptor is a vector of concatenated histograms of all the blocks in the detection window.

III. ALGORITHMIC IMPROVEMENTS

This project aims at developing a pedestrian detection system on a floating point TI C674x DSP embedded platform. The OpenCV's C++ implementation of people detection tested on this gave an execution time of over a minute per frame according to the number of clock cycles obtained from the cycle accurate simulator. C++ is inefficient for any optimizing compiler and rewriting the code in C providing a 5X improvement. As this is not algorithmic improvement, we do not use this factor for performance comparison of our algorithm. The DSP benchmark for the C code was approximately 15 seconds per frame.

Dalal [3] quotes a 1.5% reduction in performance at 10^{-4} FPPW(False Positives Per Window) with the use of grayscale images instead of RGB images. The use of grayscale images was intended to make use of the Y luminance channel of the YCbCr image provided by the camera without any computations. The use of RGB image required more than thrice the number of computations per pixel while computing magnitude and orientations, in comparison to the use of grayscale image. Hence we chose to use only the grayscale image.

Figure 2. Illustration of the Detection Window Pruning Scheme. The above sequence of figures are sample detection window, magnitude of Gx, edge-linking filtered & thresholded image, three rectanglular windows (in green) where number of edge pixels are counted.

Second major change was to remove the Gaussian spatial weighting applied to the blocks in the detection window. Dalal [3] states that a Gaussian spatial window to each pixel before accumulating orientation votes into cells improves performance by 1% at 10^{-4} FPPW. The removal of these computations reduced the accuracy of the classifier from 1 false positive for every 10^{-4} negative detection windows to 1.01015 false positives for every 10^{-4} negative detection windows, which is negligible considering the large reduction in the number of computations after the removal of Gaussian weighting.

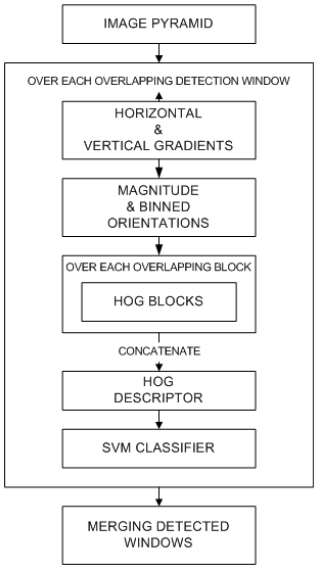

Figure 3. Block diagram of the people detect algorithm

A. Pruning of Detection Windows

The classification step is a computationally expensive step. A standard VGA image of 640 by 480 pixels will have more than 12500 detection windows which require a dot product of two vectors of length 3780. Therefore to reduce the number of dot products used, a coarse detection algorithm to reduce the number of detection windows was devised.

A typical scene from a front camera in a car consists of many regions with low texture corresponding to road, sky, etc. A simple detector can reject these regions easily with few computations. As the gradients are already computed, it is a common practice to count the number of edges and reject windows with a small number of edges. A threshold is applied on the gradient magnitude to obtain a binary edge image and the number of edges can be efficiently calculated using Integral images. A suitable threshold is experimentally chosen to allow pedestrians.

We improve this method in two ways. Firstly, we observed that the edges corresponding to pedestrian silhouette are predominantly vertical or have a strong vertical component. Hence we use thresholded horizontal gradients instead of gradient magnitude. Secondly, we also make use of the fact that these edges in the silhouette are strongly linked spatially. So for the shortlisted edges which have a strong horizontal gradient, we also check if there is another shortlisted edge in the gradient orientation direction on top and bottom. After this pruning of edge pixels, we use three windows as shown in Figure 3 and count the number of remaining edges in each of

these and use 3 different thresholds to make sure each region has significant number of edges. This ROI scheme rejects about 40% of the detection windows and was observed to be more robust in allowing pedestrian windows.

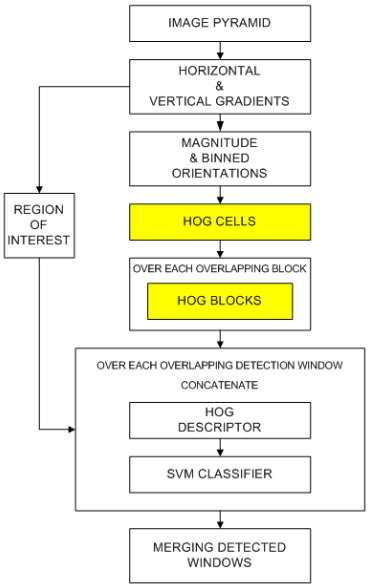

Figure 4. Block diagram of the proposed pedestrian detection algorithm. The blocks in yellow are moved out of the detection window and converted to frame-level functions with re-use of computations

B. *Design Choices and Restructuring of HOG of Blocks computation to enable computation re-use*

The OpenCV implementation has many redundant (repeated) calculations across detection windows. Computation of the histogram of the blocks and normalization was repeated for each overlapping window and overlapping block. This led to atleast 87% and 93% of repeated computations for every horizontal and vertical slide of the detection window by 8 pixels respectively.

To prevent the repetitive calculations, the algorithm was redesigned to pre-calculate the computations required for the classification step (dot product with SVM). The technique of pre-calculating and storing the computations for later use requires the cell's side, the block-stride, the window-stride, and both the sides of the detection window to be in multiples of the stride (number of pixels shift between the two adjacent detection windows).

Figure 4 shows the data flow of the modified algorithm after algorithmic restructuring. The C code implementation was modified to have the stride value set to 8 pixels. This modification gave a reasonable performance when compared to the detection results from strides of 2 and 4. Additionally, the authors suggest block spacing stride of 8 pixels to get a 4-fold coverage of each cell for optimal performance. This modification of setting the block and window stride value to a particular value allows the use of computations done for HOG descriptor in a detection window to be reused for the adjacent window. The cells in a frame are not overlapping and a fixed stride value results in prior knowledge of the number and the position of overlapping blocks in a frame. So the pre-computing step is designed to store HOG cells and normalized overlapping HOG of blocks for a frame before entering the classification step.

Further a pre-computation step was added in the image rescaling function. The image rescaling was implemented using bilinear interpolation. The rescaled coordinates for the longest side of the image was pre-computed rather than calculating them for each pixel. This pre-computation reduced the execution time of the rescaling function by half.

IV. OTHER DESIGN AND IMPLEMENTATION DETAILS

A. *DSP Specific Kernel Optimization*

Kernel based optimization was applied for the TI C674x DSP. The iterative functions like division, square-root are replaced by the functions from the C674x fastRTS library. Lookup table was created for the iterative tan inverse function for equally spaced bins from 0 to 180 degrees and a floating point lookup table for reciprocal of horizontal gradient to avoid dividing the horizontal gradient with the vertical gradient. For some fine optimization the dependency on the math library was removed (removed floor and ceil, macro for abs) to avoid function calls and were replaced by macros. Some of the other criteria for optimization on the platform were to choose proper compiler options like the optimization_level (–o3) and –mt option for optimization. When opt_level=3 is used, the compiler will try to perform all the optimization techniques it is capable of carrying out (software pipelining) based on the knowledge that is available to it. The –mt option tells the compiler that, for any function in the entire application, the pointer-based parameters of this function will never point to the same address [10].

B. *Floating-point to Fixed-point conversion*

Even on a floating point DSP, fixed point arithmetic is typically faster. As mentioned earlier, the classification step is the most computationally expensive step because of the dot product of two vectors of length 3781 for each detection window. The floating point SVM vector is converted to fixed point vector to facilitate the use of the fixed point c64+ DSP library which gives a 4 times faster dot product calculation. The image rescaling function was converted to a fixed point function from a floating point function using Q point conversion to get a speed up of almost 2X after the pre-computation step. The DSP has Very Long Instruction Word (VLIW) architecture which enables SIMD instructions. The library used makes use of the SIMD instructions for better vectorization and parallel computations.

V. RESULTS AND DISCUSSION

The aim of the project was to develop a real time pedestrian detection system on an embedded platform. The OpenCV's implementation is best suited for computer platforms. The modifications done of the implementation were to improve the data flow and efficiently utilize the DSP resources to obtain best performance. The 130 times improvement is on the base implementation when ported on a DSP system. We determined that by designing the loops in the implementation to facilitate

better data flow any implementation can be optimized to give best performance on embedded platform. The table below shows the approximate speed-ups obtained in the optimization process. Figure 5 shows a sample result of our algorithm using the challenging test data BAHNHOF sequence [13]. The algorithm was tested over ten 5 minute automotive video clips and accuracy was very similar to OpenCV's implementation. For these videos, we obtained a miss rate of 0.05 for a FPPW of 0.1.

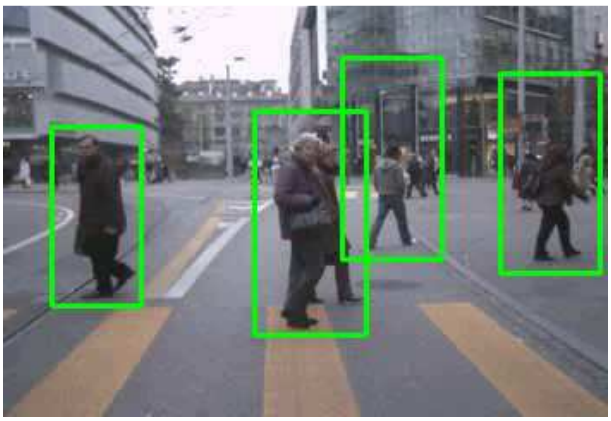

Figure 5. Example result on BAHNOF sequence

Further improvements could be done by enabling DMA and ping-pong buffering. Designing the data flow for block based implementation over the frame that help in reducing the memory overheads can be considered as a part of algorithmic optimization. Frameworks mentioned in [11] can be used for implementing memory efficient block-based schemes.

TABLE I. PERFORMANCE IMPROVEMENT FACTORS

Process	**Individual Factors**	**Cumulative Factors**
C++ → C	5	-
Grayscale & Uniform Weighting	1.5	1.5
Pre-computations	10	15
DSP libraries	1.4	21
SVM – fixed point / Fixed point lib	1.8	39
Pre-computations Image rescaling	1.9	72
Image rescaling - fixed point	1.3	94
Region of Interest	1.4	130

A. Comparison with Adaboost + Integral Histogram

A common fast alternative to HOG + SVM is Integral Histogram features + Adaboost. Zhu [14] uses this approach and reports a 70X with a significant loss of accuracy. From their comparison figure, the miss rate can be 2-3X worse compared to HOG+SVM. In addition to the loss of accuracy, the algorithm cannot be mapped efficiently on the DSP. Integral Histogram requires large additional memory (scales linearly with the number of histogram bins) which increases the memory requirements significantly and inhibits block-based operations on the DSP. Although this is not a concern on large memory systems like a PC, this severely impacts performance on a DSP. Additionally, Adaboost classifier uses a dynamic cascade classifier which inhibits the static scheduling required for a VLIW processor. Hence the parallel DSP units cannot be efficiently used. In summary, Adaboost + Integral Histogram approach works well on a non-parallel processor with large memory. But on a parallel architecture embedded system, HOG + SVM performs better in terms of speed and accuracy.

VI. CONCLUSION

We demonstrate a real-time pedestrian detection system which runs at 20 fps on an embedded DSP. Performance improvement of 130X over OpenCV shows that there is a lot of scope for achieving significant performance improvements using embedded vision techniques. Higher performance can be achieved by incorporating detection window tracking and other automotive scene heuristics. Better accuracy might be achieved by using intersection kernels for SVM.

REFERENCES

[1] D. Gerónimo, A.M. López, A.D Sappa, T Graf, "Survey of Pedestrian Detection for Advanced Driver Assistance Systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.32, no.7, pp.1239-1258, July 2010.

[2] M. Enzweiler, D.M. Gavrila, D.M, "Monocular Pedestrian Detection: Survey and Experiments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.31, no.12, pp.2179-2195, Dec. 2009.

[3] N Dalal, B. Triggs, B, "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol.1, no., pp.886-893 vol. 1, 25-25 June 2005.

[4] P. Viola, M. Jones, D Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance," *International Journal of Computer Vision, 2005. Springer Netherlands*, vol.63, no., pp.153-161 vol. 63, 2 July 2005

[5] D. Gavrila, D. Vernon, "Pedestrian Detection from a Moving Vehicle," *Computer Vision — ECCV 2000. Springer Berlin / Heidelberg*, vol.1, no., pp.37-89349 vol. 1843. 2005

[6] J. Ge, Y. Luo, G. Tei , "Real-Time Pedestrian Detection and Tracking at Nighttime for Driver-Assistance Systems," *Intelligent Transportation Systems, IEEE Transactions on* , vol.10, no.2, pp.283-298, June 2009

[7] C. Premebida, O. Ludwig, U. Nunes, "LIDAR and vision-based pedestrian detection system". Journal of Field Robotics 2006, 26: 696–711.

[8] G. Xu, X. Wu, L. Liu, Z. W, "Real-time pedestrian detection based on edge factor and Histogram of Oriented Gradient," *Information and Automation (ICIA), 2011 IEEE International Conference on* , vol., no., pp.384-389, 6-8 June 2011.

[9] M. Bansal, S. Jung, B. Matei, J. Eledath, H. Sawhney, "A real-time pedestrian detection system based on structure and appearance classification," *Robotics and Automation (ICRA), 2010 IEEE International Conference on* , vol., no., pp.903-909, 3-7 May 2010.

[10] S.K. Yogamani, " A Tutorial on Optimizing Vision Algorithms on TI DSPs," Texas Instruments Application Report SPNA165, August 2012.

[11] B.R. Kiran, K.P. Anoop, Y.S. Kumar; "Parallelizing connectivity-based image processing operators in a multi-core environment", International Conference on Communications and Signal Processing (ICCSP), 2011.

[12] V.A. Prisacariu, I. D. Reid, "fastHOG - a real-time GPU implementation of HOG", University of Oxford, Technical Report 2310/09.

[13] A. Ess, B. Leibe, K. Schindler, L. van Gool, "A Mobile Vision System for Robust Multi-Person Tracking", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08).

[14] Q. Zhu, M.C. Yeh, K.T. Cheng, S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06).