ACCELERATED OBJECT DETECTION

Jiří Havel
Doctoral Degree Programme (1), FIT BUT
E-mail: ihavel@fit.vutbr.cz

Supervised by: Adam Herout
E-mail: herout@fit.vutbr.cz

ABSTRACT

This paper presents the improved version of a rapid object detector. This detector uses Waldboost classifier with Local Rank Differences features for face detection. Because of it's computational complexity, original detector was parallelized to take advantage of modern multicore processors. Further speedup was achieved by fixed step scaling with heavy use of SSE instruction set and template metaprogramming.

1 INTRODUCTION

This paper contributes to the research of detection classifiers done by Graph@FIT group [4, 7]. Because of complexity of this detection, various acceleration methods were developped [2, 6]. This paper extends the implementation on x86 CPUs using SSE instruction set [3].

Object detection is performed by evaluating classifier at every position in image. Classifier evaluates selected window of image (in the described program, window is 24×24 pixels). A response of the classifier is one real number, which expresses the likelihood that the window contains the detected object. If the response is greater than selected threshold, the object is present in the evauated window. The value of threshold determines false positive and false negative rate. Typically is one object detected on several nearby positions as shown in figure 1. For every overlapping group of objects, the position with the gratest response is selected. This is called non-maxima suppression.

The original version of the improved software was the implementation of [3], done by Roman Juránek. It uses Adaboost classifier to detect faces. Adaboost is algorithm which combines large number of weak classifiers to one strong [1]. Weak classifiers can be very simple like difference of two pixels from evaluated window. Final classifier is linear combination of weak classifiers, sometimes called stages. Weak classifiers are selected from large number of possibilities by automated training process. Waldboost is modification of Adaboost, which can terminate early without evaluating all stages [5]. For our 1000 stage classifier for face detection, the average evaluated count in common images is below 5. Local Rank Differences were used as weak classifiers [7].

Classifier detects only one size and rotation of objects. Therefore detection must be performed on pyramid of scaled images, ideally to the window size. Similar problem exists with object rotation. This significantly increases complexity of detection. Every processed image is con-

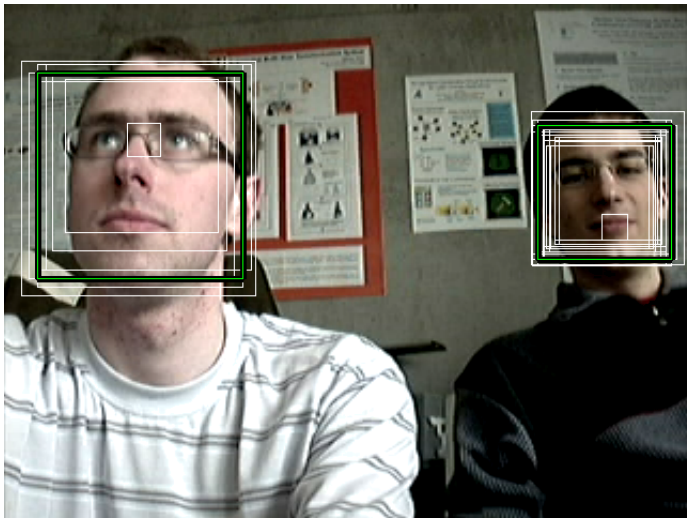

Figure 1: Detected faces and maximal responses.

verted to grayscale and optionally smoothed by gaussian kernel. For every detected object rotation, pyramid of shrunk images is built. These images are converted to classifier input format by convolving with set of simple kernels.

2 PARALLELIZATION

Detector was parallelized using industry standard OpenMP library. Frames are processed in two parallel blocks, as shown in figure 2. The outer blocks evaluates in parallel for image rotations and the inner blocks for the pyramid leveld. In the serial part remains the data input, the color conversion, the non-maxima suppression and the data output.

Scaling is performed in the outer parallel block. Moving it into the inner parallel block would be complicated because the pyramid levels dependencies. This will be further discused in section 3.

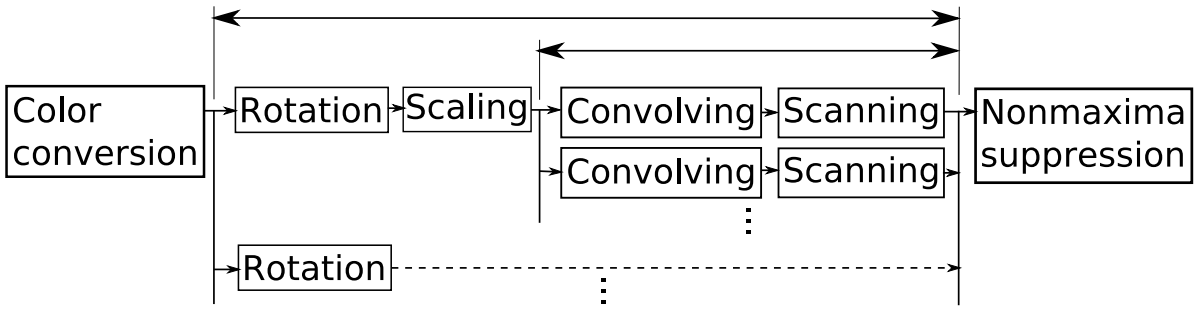

Figure 2: Processing of one frame with marked parallel parts.

Detected objects are collected separately for each thread and merged together after all threads finish. This elliminates the need for any synchronization point between processing threads. The required memory is allocated in advance to remove another synchronization point. This arrangement ensures that all cores are fully utilized.

3 SCALING

Scaling is designed as block transformation. Every operation is described by input and output block size and transformation of one block.

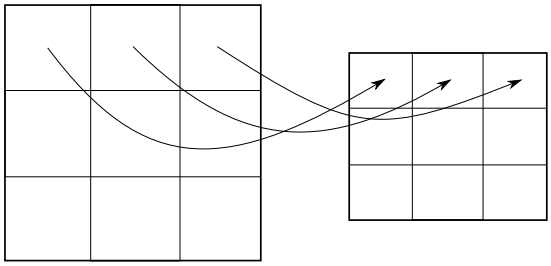

Figure 3: Image block transformation.

Source image is divided into blocks, which are processed separately, as shown in figure 3. Source image size must be divisible by input block size. If one image is used as input for more operations, its size must be aligned to least common multiple of required alignments of these operations. This is calculated at compile time by C++ template metaprogramming. Minimal output image size must be output block size multiplied by block count.

In described detector, image pyramid is built by two scale operations. One operation scales image to $7/8$ and other to $1/2$ of its size. Both operations use bilinear filtering. Every fifth level of the pyramid is built as $1/2$ of image five levels below, other as $7/8$ of previous level. This is illustrated by figure 4.

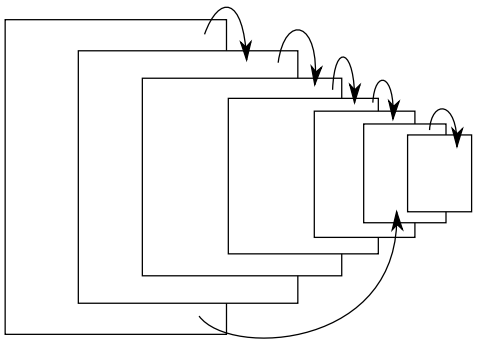

Figure 4: Building pyramid. Top arrows are $7/8$ scales, bottom arrow $1/2$.

As was mentioned in section 2, this dependency limits parallelization of pyramid building. It is possible to use some sort of synchronization. Scaling however isn't the most time consuming task. To avoid expensive synchronization, scaling was left in the outer block.

4 SPEEDUP

Program was compiled using Intel C++ compiler version 11 and run on Core 2 Duo E8200. Test data were 100 frames from videos of size 1280×480 and 720×576.

Thread count	1	2	3
Video 1	14.15	8.38	8.95
Video 2	7.17	4.65	4.88

Table 1: Processing time in seconds, depending on thread count

Table 1 shows, that parallelization improved total execution time by factor 1.6-1.7. Using more threads than processor cores didn't improve computation time any further.

Pyramid build time with fixed step funtions was compared to resize function from OpenCV library with bilinear filtering. Table 2 shows, that presented method performs two times faster than standard OpenCV function. However this restricts the pyramid to fixed steps between levels and requires strict image alignment.

	New	cvResize
Video 1	0.19	0.39
Video 2	0.12	0.27

Table 2: Pyramid build time in seconds

Measurement also shows, that pyramid build time is almost irrelevant to total processing time. 85% of total time is taken by scanning.

5 CONCLUSION

This paper presented improvements of existing object detector and evaluated the results. Parallelizing proved to be viable solution in this situation, as the object detection can be easily divided to parts and processed independently. For dualcore machine, the speedup was between 1.6-1.7. Improved scaling didn't improve computation time significantly. Although it was two times faster than OpenCV function, proportion of consumed time was too small to actually improve computational time. For applications, where image scaling takes greater part of processing time, this implementation can be very usefull.

REMARKS

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics) and by the research project "Security-Oriented Research in Information Technology" CEZMSMT, MSM0021630528.

REFERENCES

[1] Y. Freund and R. Schapire. A short introduction to boosting. *Japonese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[2] Adam Herout, Michal Hradiš, Radovan Jošth, Roman Juránek, and Pavel Zemčík. "local rank differences" image faeture implemented on gpu. In *Advanced Concepts for Intelligent Vision Systems*, Lecture Notes in Computer Science, pages 170–181, 2008.

[3] Adam Herout, Pavel Zemčík, Roman Juránek, and Michal Hradiš. Implementation of the "local rank differences" image feature using simd instructions of cpu. In *Proceedings of Sixth Indian Conference on Computer Vision, Graphics and Image Processing*, page 9. IEEE Computer Society, 2008.

[4] Michal Hradiš, Adam Herout, and Pavel Zemčík. Local rank patterns - novel features for rapid object detection. In *Proceedings of International Conference on Computer Vision and Graphics 2008*, Lecture Notes in Computer Science, pages 1–12, 2008.

[5] Jan Sochman and Jiri Matas. Waldboost " learning for time constrained sequential detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 150–156, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Pavel Zemčík and Martin Žádník. Adaboost engine. In *Proceedings of FPL 2007*, page 5. IEEE Computer Society, 2007.

[7] Pavel Zemčík, Michal Hradiš, and Adam Herout. Local rank differences - novel features for image. In *Proceedings of SCCG 2007*, pages 1–12, 2007.