Grant agreement no: FP7-610603

European Robotic Pedestrian Assistant 2.0 (EUROPA2)

Start of the project: 01.10.2013
Duration: 3 years

DELIVERABLE 3.1

Close-Range and Occluded Pedestrian Detection and Tracking

Due date: month 12 (September 2014)
Lead contractor organization: KULeuven
Dissemination Level: PUBLIC

1 Introduction

Deliverable 3.1 describes the initial version of the detection and tracking system for pedestrians. The final version of the system will be described at M30. The work is performed by KULeuven in collaboration with GeoAutomation.

This document describes the intermediate version of a software system for pedestrian detection and tracking. In this version, we developed an elementary but robust detector with a strong focus on efficiency. The detector is able to process a video feed in real time while running on a single CPU core. Considering the energy constraints and the limited amount of computing resources available on a mobile platform, an efficient detector offers significant advantages over power-hungry GPU based implementations. Future versions of the detector will build upon this solid base to further improve detection performance by incorporating additional data from other sensors and modules.

2 Architecture

Our pedestrian detector is based on the Aggregated Channel Features (ACF) [2] framework. ACF is an adaptation of the Integral Channel Features (ICF) [3] method by P. Dollar, which is in turn an extension of the famous Viola-Jones architecture [5].

The original Viola-Jones paper describes a framework for robust real-time face detection with the AdaBoost learning algorithm [5]. The image features are sums of pixels in rectangular boxes of a gray-scale input image. These sums can be calculated very quickly through 4-pixel look-ups in the integral image. During training, the AdaBoost algorithm builds a small amount of discriminative decision trees from the large set of possible features. These weak classifiers are then combined in a cascade. The cascade allows negative windows to be quickly discarded while spending more computation on promising windows. The final detector is applied in a sliding-window fashion on the input image, followed by a non-maximum suppression step to obtain a single detection per pedestrian.

Dollar's Integral Channel Features (ICF) [3] is an extension of the Viola-Jones architecture where the features are calculated over multiple channels instead of only the gray-scale image. Typically 10 channels are used: 3 color channels, 6 gradient orientation channels and 1 gradient magnitude channel. The use of multiple channels considerably improves detection performance.

Finally, Aggregated Channel Features (ACF) [2] is an adaptation of ICF where features are single-pixel look-ups in a so-called aggregated image instead of 4-pixel look-ups in an integral image. This means that features can be considered as sums of pixels in fixed-size squares instead of in variable-sized rectangles. Additionally, ACF uses fast feature pyramids for detecting objects at multiple image scales: instead of calculating the image features explicitly at each scale, they are approximated via extrapolation from nearby scales. These two extensions offer a significant speed improvement over ICF, and they allow for a fast CPU based implementation.

3 Specifications

The detector is written in C++ and features a simple command line interface, as shown in Figure 1. The program outputs a text file with the coordinates of the detection bounding boxes of the images in the input directory. It is also possible to store the processed images directly in an output directory. The software has dependencies with OpenCV and the Boost external libraries. Models are trained in Matlab with Piotr's Matlab Toolbox [1] and an additional script to convert the resulting model to a

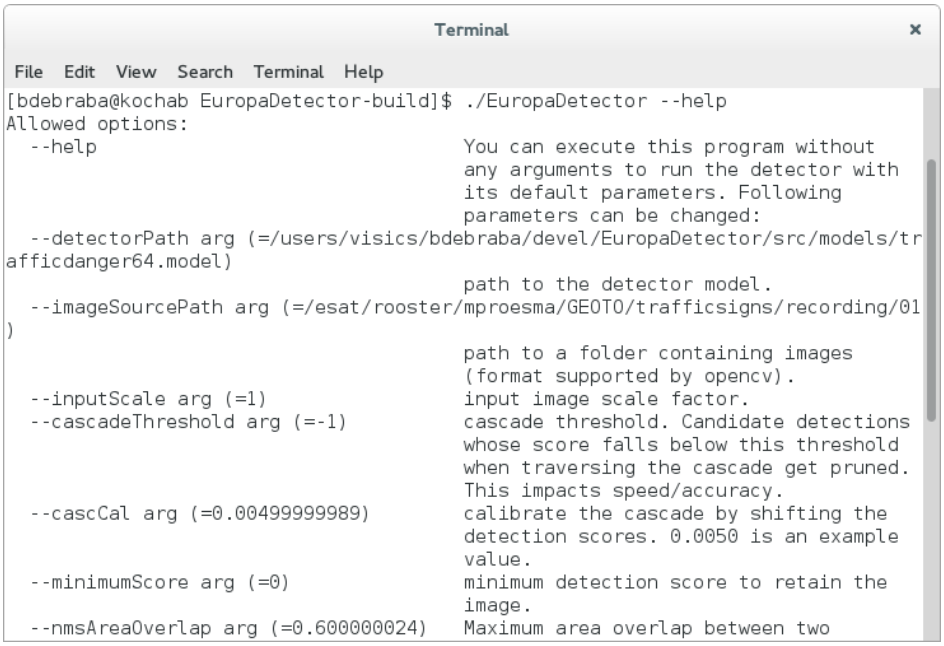

Figure 1: c++ based command line interface for pedestrian detection

binary file, which can then be loaded into the C++ detector. A frame rate of 30 fps on a single 2.8Ghz CPU core was obtained with following settings:

- image frames with a resolution of 640x480 pixels
- An image pyramid with 20 scales (8 scales per octave, of which 7 are approximated)
- A shrinking factor of 4
- A cascade consisting of 2048 level-two decision trees, with a constant rejection threshold of -1.

The log-average miss rate, a standard measure for detection performance, is 17% on the pedestrian benchmark INRIA and 45% on the more challenging Caltech dataset. The performance is on-par with the state of the art, with only a few detectors performing slightly better while mostly requiring much more computational resources. Figure 2 shows some successful detections on these data sets. Note that the detector is color-coded, according to a cold-hot color scheme. Blue-green colors indicate low responses, going over the yellow-orange range towards red-magenta that indicates large responses.

The detector has also been applied to the GeoAutomation Zurich dataset, which was recorded and processed under task 1.4 (deliverable D4.1). The sets consists of 134672 frames, for 8 individual cameras, which is about 2,5 hours of data. Processing took a bit more overhead on disk I/O because the resulting images are exported over the network. Figure 3 shows the detection results. On the left, we depict some successful detections. On the right, we show a typical false positive, where the detector is triggered by pedestrian-resembling structures, such as round traffic sign accompanied by pole structures, reflective patterns on the sides of cars, etc.

In the general case, the detector response is generally high for the people, except when occlusions are involved or when the pose deviates too much from the training data. False responses (traffic

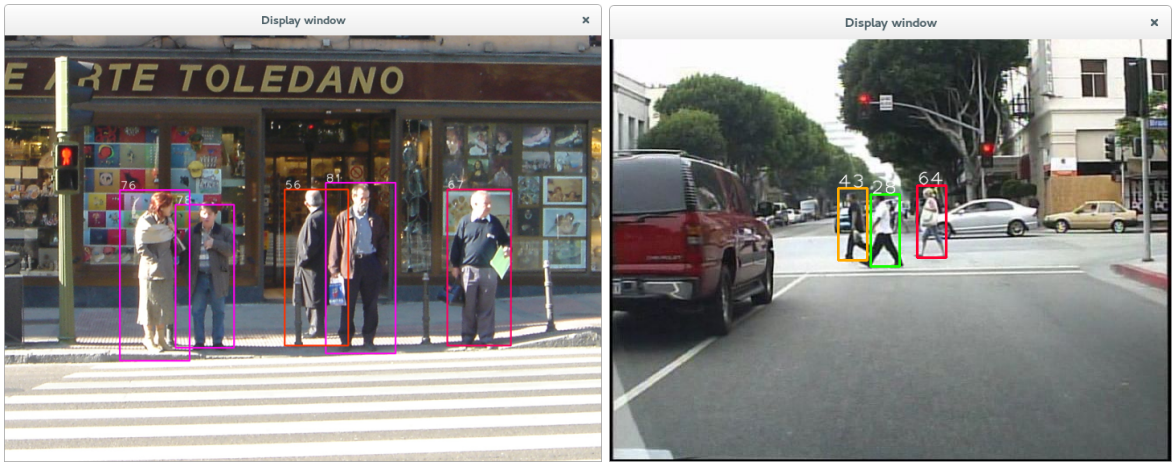

Figure 2: Detection example on the Inria (left) and Caltech (right) dataset

Figure 3: Positive detections on the GeoAutomation dataset (left). The right image shows a negative detection around a traffic sign

signs, boxes, car reflections) are generally on the lower side (indicated by blue-greenish colors on the images), and they are far less consistent over time. Figure 4 shows a consecutive set of frames indicating how the detector response behaves over time.

It was argued that such false responses could be used as negatives in a second training phase, however it appeared that this did not improve the performance significantly. We plan the handle these problems in future versions of the detector by implementing a tracking system on the detector output or incorporating information from other modules and sensors, like depth data gathered from structure from motion or stereo vision.

4 Occlusion handling

The intermediate version of this deliverable contains basic occlusion handling capabilities, which will be refined in future versions. Currently, only occlusions caused by the image border are han-

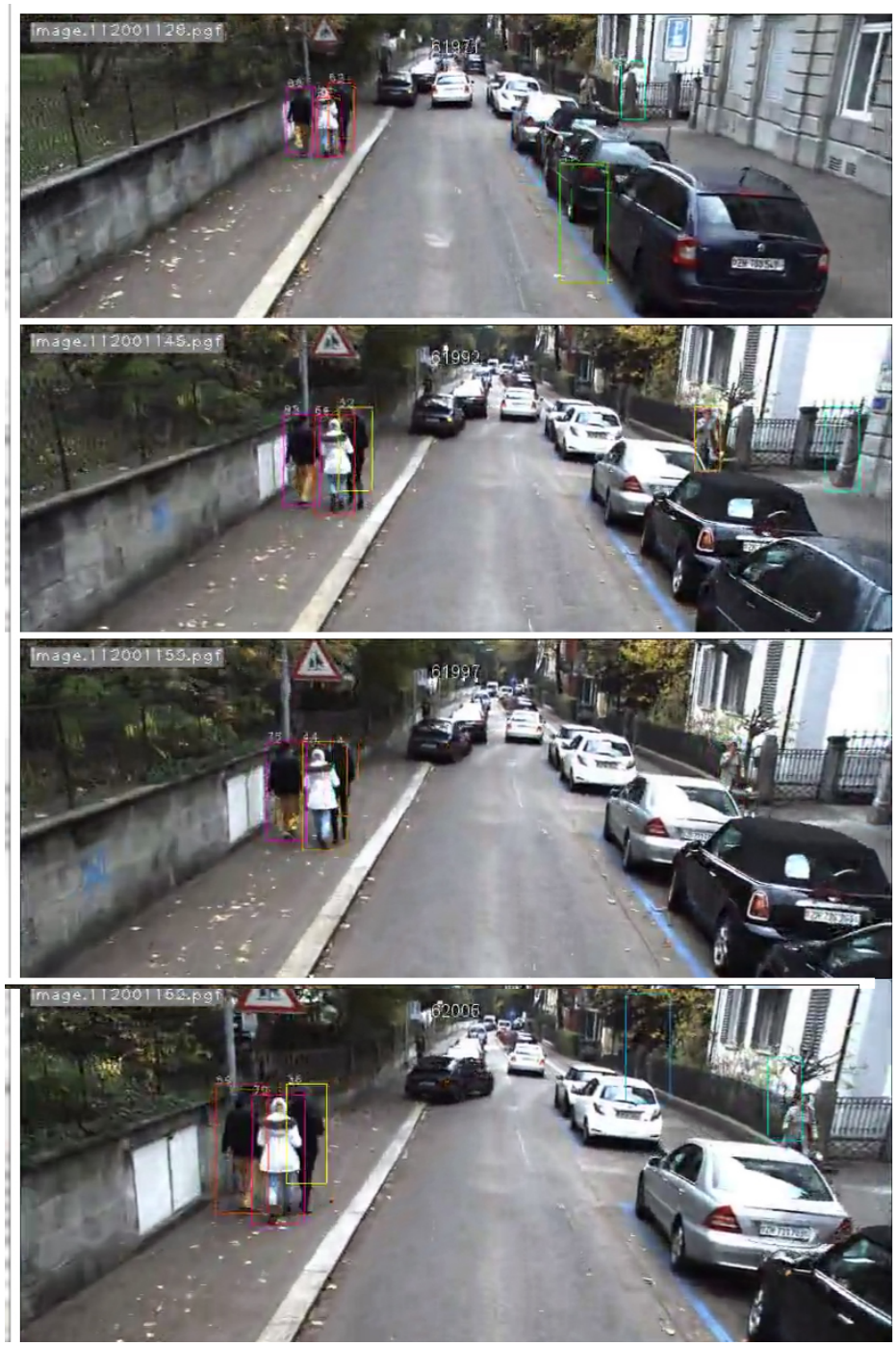

Figure 4: Detection on consecutive frames of the GeoAutomation sequence

dled. Integration with depth sensors and data from other modules will allow to detect other types of occlusions as well. The basic idea is to *cut* the model at the image border and only evaluate weak classifiers that are in non-occluded regions of the full model. Within the standard ICF framework this would not be so straightforward: the features are rectangular boxes and each one may span a large region of the model, so cutting the model will remove a large proportion of features. To this end, more sophisticated methods are needed, like Franken-classifiers [4].

With ACF, the features are more localized so simply cutting the model is a feasible idea. Another problem then arises: because windows are accepted based on fewer weak classifiers, the detector assigns lower confidence scores to positive occluded detections than to their non-occluded counterparts. As a result it is impossible to do proper non-max suppression between detections with different amounts of occlusion. To compensate for the lower confidence scores, we currently artificially boost the scores with a compensation factor based on the amount of occlusion. Alternative strategies will be investigated in future versions of the detector. Figure 7 shows how our system improves detection of pedestrians that are occluded by the lower image border, which is a frequently occurring case.

References

[1] Piotr Dollar. Piotr's image and video matlab toolbox (pmt). Software available at: `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`, 2013.

[2] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[3] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *BMVC*, volume 2, page 5, 2009.

[4] Markus Mathias, Rodrigo Benenson, Radu Timofte, and Luc Van Gool. Handling occlusions with franken-classifiers. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1505–1512. IEEE, 2013.

[5] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

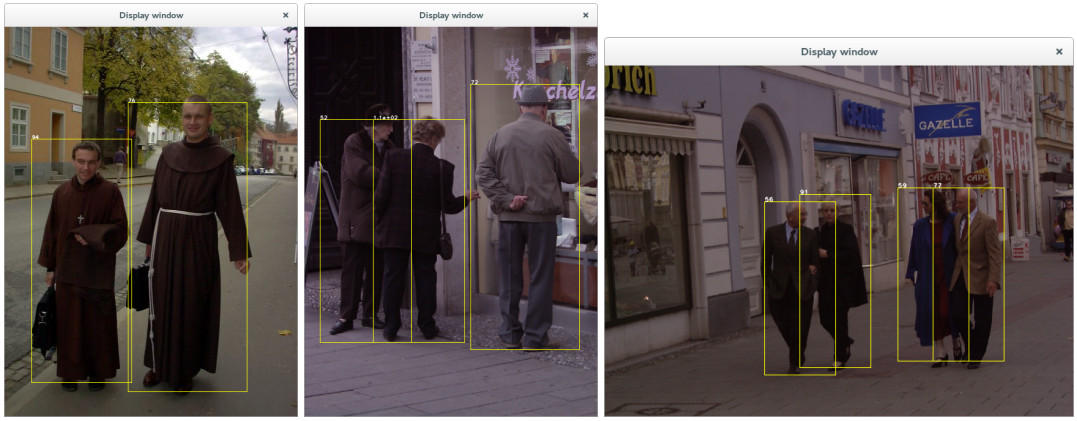

Figure 5: Non-occluded detections

Figure 6: Occluded detections without occlusion handling

Figure 7: Occluded detections with occlusion handling