A Real-time Cascade Pedestrian Detection based on Heterogeneous Features

Zhaowei Cai and Nuno Vasconcelos
University of California San Diego
9500 Gilman Drive
La Jolla, California, USA
{zwcai, nuno}@ucsd.edu

Abstract—**In this paper, we are introducing a real-time pedestrian detector in traffic scenes. To improve the running efficiency of the whole detection system, a cascade framework is implemented. Three heterogeneous types of features are used, including aggregate channel feature (ACF), the responses of self-similarity and checkerboard filters on ACF. Based on the observations that 1) these three types of features take different complexities to compute, and 2) most of the false positives are rejected in the former cascade stages, we propose to divide the cascade into 3 parts, and each type of features is selected in each cascade part according to the feature complexity. The ACF features are computed for the whole image before any cascade stage, and the other two types of features are computed when they are needed at specific stages. This strategy significantly outperforms ACF-only features with only a little loss of speed. Finally, we achieve a pedestrian detector with running speed of 10 frames per second on a 640×480 image.**

Keywords-pedestrian detection; cascade; aggregate channel feature; self-similarity; checkerboard filter

I. INTRODUCTION

Pedestrian detection is one of the most important computer vision applications, since recent years many companies, e.g. Google, Baidu, etc., devote significant efforts on developing driver assistance systems and self-driving automobiles. However, pedestrian detection is a difficult problem because the detection system should be very fast and highly accurate for safety. For example, these systems must detect all pedestrians in the field of view at a very low false positive rate.

Based on the two concerns mentioned above, many real-time pedestrian detectors with acceptable accuracy resort to the cascade structure of [1]. In this architecture, usually a bunch of weak classifiers are trained with efficient features, and the weak classifiers are combined to produce a strong classifier. In detection process, the strong classifier is performed on each patches on the image in a sliding window way. Instead of processing all weak learners for every sliding window patch, the weak learners could be split into different stages, where each stage only has one or a few weak learners. Some patches that are easily classified as background could be rejected by only a few stages, so that the whole system could run in real-time. However, the performance of the framework is limited by the features. To improve the accuracy, some approaches [5] [6] propose to use more complicated features, which will impair the running speed. In this paper, we focus on how to keep the running speed, even when heterogeneous and complicated features are incorporated, so that a better trade-off between speed and accuracy is achieved.

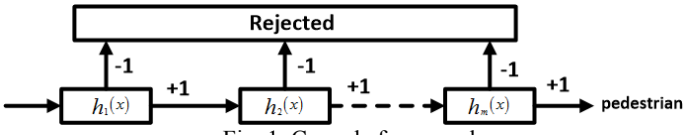

Fig. 1. Cascade framework.

II. PROPOSED SYSTEM

A. Cascade Detection Framework

In this pedestrian detection system, we use cascade framework, which is shown in Fig. 1. The input is features extracted from the test images. As the cascade progresses, more and more background patches will be rejected, and only a few patches with high confidence scores will be survived. This strategy makes the real-time pedestrian detection available.

While the detection process is fast in cascade framework, the speed of the whole system still depends on the speed of feature extraction. Usually, the features used in real-time detection task are homogeneous and efficient, e.g. Haar [1] and ACF [4]. However, the simplicity of the features will limit their representation capacity. To improve detection accuracy, more heterogeneous and complicated features are needed, which will impair the speed at the same time. In this system, we design a new cascade framework that significantly outperforms the traditional one with a small loss of speed. In the proposed approach, multiple heterogeneous complicated features with varied complexity are used, but different features are selected in different cascade stages. As we known, the majority of background patches are rejected in the first few cascade stages, and only a small portion will be survived. Based on this observation, the computationally cheap features should be used in the former cascade stages, and as the cascade moves on, more expensive features are used, and so on. For example, in a cascade with 2048 stages, the first 1024 stage uses the cheapest features, the next 512 stages the second cheapest feature, and the last 512 stages the most expensive features. In this way, the cheap features will be performed on the majority of patches, so that the computation amount is almost the same with using cheap features only.

This work was supported by NSF grant (NSF IIS-1208522) and the Technology Development Program for Commercializing System Semiconductor funded By the Ministry of Trade, industry & Energy (MOTIE, Korea). (No. 10041126, Title: International Collaborative R&BD Project for System Semiconductor)

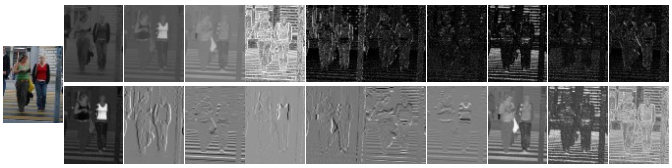

Fig. 2. The left image is the input, the top row is the ACF feature channels and the second row is some CB feature channels.

B. Heterogeneous Features

The cheapest features we use in the first part (1024 stages) of cascade are aggregate channel features (ACF) [4]. There are three types of channels in ACF: color channel (LUV), gradient magnitude and gradient histograms. To increase robustness, the features are averaged downsampled by 4 times. For a 64×32 pedestrian template the process is: extract the 10 64×32 feature maps from the input image, then downsample them to 16×8 feature maps. In total, there are 10×16×8=1280 ACF features for a pedestrian template. The computation of ACF is very efficient (approximately 50 fps for a 640×480 image).

The second feature set we use is self-similarity (SS) [7] on ACF feature channels. SS is the difference between two ACF feature values. Instead of computing every SS between any two ACF features, we compute SS only on a 12×6 grid of the 16×8 ACF channels. In total there are 10×72×71/2=25,560 SS features for a pedestrian template. SS feature set could have a more power representation than ACF. Since every computation of SS involves 2 ACF features, SS is more expensive than ACF. Thus, they are only selected in the second part (512 stages) of the cascade.

The third feature set is the convolution responses of checkerboard filters (CB) [5] on ACF channels. The eight CB filters we use are of 2×2 size, with +1 or -1 values. They significantly extend the feature pool of ACF. In total, there are 10×8×16×8=10,240 CB features for a pedestrian template. Since each CB convolutional response needs 4 computations over ACF, they are even more expensive than SS. Therefore, they are located in the last part (512 stages) of the cascade.

Some ACF and CB channel features are shown in Fig. 2, in which they show strong representation of the pedestrian template.

III. EXPERIMENTS

The algorithm is implemented with C++. We compared the proposed system to a set of state-of-the-art pedestrian detectors on the Caltech Pedestrian dataset [2]. The cascade is composed of 2048 decision trees of depth 2. For training, the positives are the cropped pedestrian patches, and bootstrapping technique is used to exploit the hard negatives. Bootstrapping happens at stages {32, 128, 256, 512, 1024, 1536}, and all the weak learners are stacked together as the whole cascade. The performance of the proposed system was evaluated with the toolbox of [2]. The comparison is based on detecting pedestrians of height at least 50 pixels in 640×480 images. This

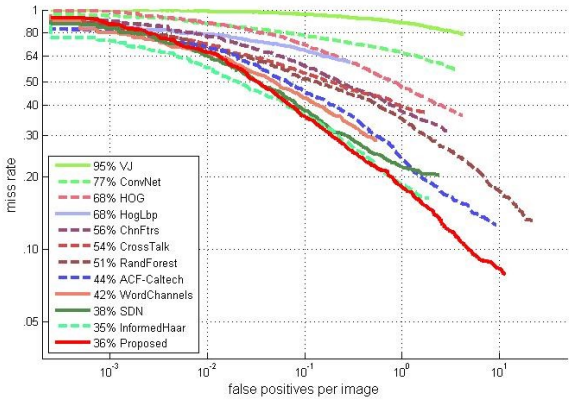

Fig. 3. Miss rate vs. FPPI rate for of various pedestrian detectors. The number on the left of each legend is the log-average miss-rate.

is equivalent to detecting pedestrians about 40m away from the vehicle. Fig. 3 presents the results of this comparison. The numbers shown on the left of the legend summarize the log-average miss-detection rate. The proposed algorithm is compared with some popular architectures, such as SVM based detectors [3], cascade based detectors [1] [4], and some recently popular deep learning based methods [8]. The proposed method outperforms all detectors other than InformedHaar [6], which spent a lot of time to design the filters. The benchmark detector is ACF-Caltech [4], which uses a similar detector and features. The experiments show introducing heterogeneous features significantly improve the accuracy (about 8%). Meanwhile, benefited from the special design of the cascade, the proposed detector achieves the speed of 10 frames per second.

REFERENCES

[1] P. Viola and M. Jones. "Robust real-time object detection". Workshop on Statistical and Computational Theories of Vision, 2001.

[2] P. Dollar, C. Wojek, B. Schiele, and P. Perona. "Pedestrian detection: An evaluation of the state of the art". IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(4):743–761, 2012.

[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. "Object detection with discriminatively trained part-based models". IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010.

[4] P. Dollar, R. Appel, S. Belongie, P. Perona. "Fast Feature pyramids for Object Detection". IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.

[5] S. Zhang, R. Benenson, and B. Schiele, "Filtered channel features for pedestrian detection," In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 1751–1760, 2015.

[6] S. Zhang, C. Bauckhage, and A. B. Cremers, "Informed haarlike features improve pedestrian detection," In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 947–954, 2014.

[7] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[8] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 3626–3633, 2013.