International Institute of Information Technology, Hyderabad

Project Report

Human Detection using Cascade of HoGs

Author:
Rohit Girdhar

Supervisor:
Dr. C.V. Jawahar

November 8, 2012

Contents

Acknowledgements

I am really thankful to my guide, Dr. C. V. Jawahar for introducing me to the world of Computer Vision by the means of this project, and guiding me to reach a meaningful conclusion.

I am also very thankful to Ankit Gandhi and Natraj J. from CVIT, IIIT-H for helping me understand the problem and various concepts, giving invaluable suggestions regarding improvement of the system as well as supporting me throughout the development of the project.

Abstract

This is a report on the implementation of a Human Detection Algorithm, based on Histograms of Oriented Gradients as features, and using Adaboost classifier (with SVM as weak classifier), proposed by Zhu et.al. in [4].

It aims at developing a fast real-time human detection system. The main idea of the algorithm proposed in [4] is based on Histograms of Oriented Gradients, proposed by Dalal and Triggs in [2]. They combine this idea with the Adaboost boosting algorithm, proposed by Viola and Jones in [3], to achive a fast and accurate classifier.

1 Introduction

The paper by Dalal and Triggs [2] first proposed the use of HoGs for Human Detection. The paper by Zhu et. al. [4] integrates this approach with the cascade of rejectors approach (from [3]) to achieve the fast and accurate human detection system.

The main difference in [4] compared to [2] comes as they use HoGs of variable size blocks, and using AdaBoost, are able to select the appropriate set of blocks from a large set of blocks from the image. This allows them to use a much denser set of blocks of the image, including larger blocks that could encode meaningful parts of the human body, such as legs etc, which was not possible with fixed set of 16x16 cells. Another novel idea presented in [4] is use of integral image to compute the HoGs. Integral Images was suggested first in [3] to compute the Haar like features for face detection, and has been extended by Zhu et.al. to HoGs. This further allows for fast computation of HoG features and hence consider the large set of cells as features.

2 Papers Review

2.1 Dalal and Triggs Human Detection

In the method proposed by Dalal and Triggs in [2], they use a dense grid of 16x16 cells in the 128x64 human image, with stride of 8px. They use HoG as the feature for each block of 2x2 cells, and have a normalization step for each block to emphasize relative behaviour as opposed to absolute values.

The algorithm proceeds as follows:

Each detection window is divided into cells of 8x8 and each group of 2x2 cells is integrated into a block in sliding fashion, so the blocks overlap with each other. Each cell consists of a 9-bin histogram of oriented gradients (HoG), and each block contains a concatinated vector of all its cells, hence representing each block with a 36D vector, normalized by L2 norm. The sliding approach leads to 7x15 blocks in the image, and give a 3780D vector per window. This feature vector is then classified using linear SVM classifier.

2.2 Viola Jones Object Detection Algorithm

The main ideas borrowed from [3] are the Integral Image and AdaBoost cascade of classifiers.

The Integral Image allows very fast computation of Haar-wavelet type features, and lead to real time face detection system that was later even extended to human detection system.

In [4], the authors exploit the above to compute HoG feature. We first discretize each pixel's orientation into 9 bins, and store an integral image for each bin, and then use it to compute efficiently the histogram over any rectangular region in 4x9 operations.

3 Algorithms and Implementation Details

3.1 Training

The main training algorithm works by creating a cascade of Strong classifiers. Each strong classifier is a linear combination of weak classifiers (as proposed in [3]), and each weak classifier is a linear SVM classifier over a certain block of pixels in the detection window (image). To compute the strong classifier, we keep adding weak classifiers to the current

strong classifier until the required FP rate for that cascade stage is satisified with the required detection rate also satisfied. The StrongClassifier is then added to the cascade of classifiers, and the negative images are recomputed as the false detections from the main set of negative images (these are the 'Hard examples').

input : F_{target} : Target overall FP rate
f_{max} : Maximum FP per cascade level
d_{min} : minimum acceptable detection per cascade level
Pos : Set of positive images (64x128)
Neg : Set of negative images
PosVal : Set of positive images (Validation)
NegVal : Set of negative images (Validation)

i=0, D_i=1.0, F_i=1.0;
Cascade = ();
while $F_i > F_{target}$ **do**
Divide Pos, Neg into Training and Validation sets (Pos,Neg) and (PosVal,NegVal);
$i = i + 1$;
$f_i = 1.0$;
StrongClassifier = ();
▷ The next loop now trains a strong classifier, as one stage of the cascade. This basically follows Table1 in [3];
while $f_i > f_{max}$ **do**
1. weakClassifiers = computeWeakClassifiers(Pos, Neg, 250);
▷ This function computes 250 blocks at random, as per the sampling strategy given in paper, and trains SVM classifier accordingto the block using Pos and Neg samples;
2. b = getBestClassifier(PosVal, NegVal, weakClassifiers);
▷ This function gets the best classifier out of the previous 250 classifiers, validating on PosVal and NegVal;
3. Add b to current StrongClassifier, update StrongClassifier.threshold;
▷ Threshold calculated in adaboost manner;
4. Evaluate current StrongClassifier on PosVal, NegVal, decrease StrongClassifier.threshold until d_{min} holds;
▷ Decrease in AdaBoost style, details [1];
5. Compute f_i under this threshold on PosVal, NegVal;
end
Append StrongClassfier to Cascade;
$F_{i+1} = F_i \times f_i$;
$D_{i+1} = D_i \times d_{min}$;
Empty set Neg;
Evaluate current casacaded classifier on all negative images and add misclassified subimages to Neg;
end

output: i levels Cascade
Each level has boosted classifier of SVMs

Algorithm 1: Main Training Algorithm. Also refer Table2 in [3]

3.2 Detection

The detection works by running each detection window through the cascade, and if it passes all strong classifiers, it is labelled as positive. The whole process is done at multiple scales. To speed up the process, instead of scaling the image again and again and computing the integral image, the detector itself is scaled.

```
input  : Cascade
         TestImage
TestII = computeIntegralImage(TestImage);
Detections = ();
forall Scales do
    forall lvl in Cascade do
        s = Cascade(lvl);
                                                    ▷ s is a StrongClassifier;
        s = scaleClassifier(s, scale);
              ▷ scale the classifier block dimensions based on current detection scale;
        d = computeFeature(TestII, s);
        d = downScaleFeatureValue(d, scale);
            ▷ Scaled feature value downscaled to be able to work with the pre-trained
        classifiers;
        res = classify(d);
        if res = 0 then
            Reject Window;
        end
    end
    if lvl = LastLevel then
        Add (window position, dimensions, scale) to Detections;
    end
end
output: Detections
```

Algorithm 2: Detection Algorithm

4 Results/Observations

The training process took 5 days on a 2 GHz processor with 4 GB RAM (in CVIT Atom Cluster). The trained detector was limited to 30 stages.

The final trained detector was run on the INRIA test dataset, and gave a detection rate of 80% at FPPW rate of 8×10^{-4}. This is quite close to the detection rate of 88% at FPPW of 10^{-4} claimed by the original authors in [4].

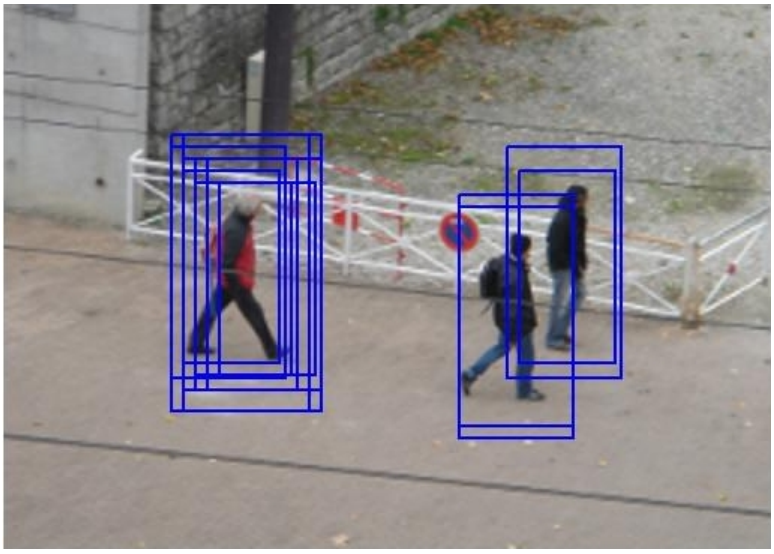
As mentioned in [4], the initial stages of the detector contained the larger sized blocks. In my case, the first block to be selected was 64x32 block from 36x29 position, which would aproximately model the main torso of the person, which is similar to the first block selected as mentioned in [4].

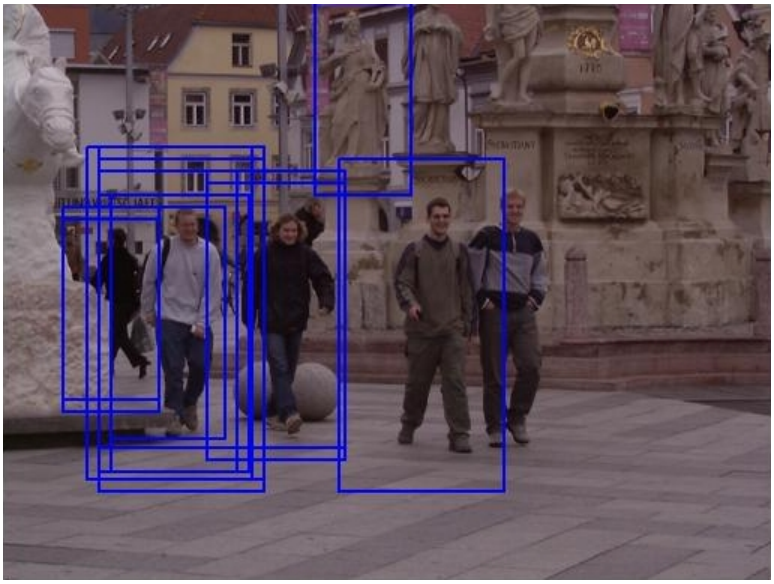
The detector was slower than what was claimed by original authors. I could achieve 0.1 FPS for a 300x200 image, while the authors claim nearly 5FPS rate. Possible reason for the same is that my implementation was in MATLAB, which is much slower than other

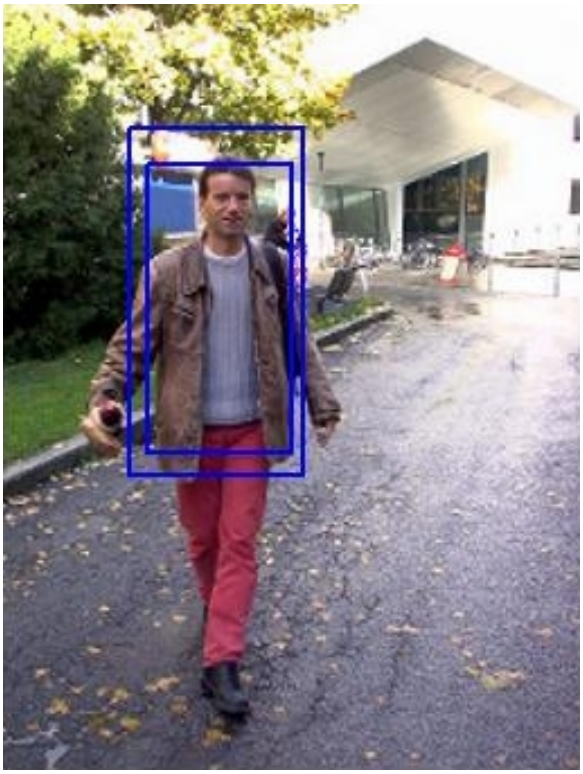
programming languages (like C/C++). Moreover, I concentrated more on achieving detection/FPPW results than the speeds, and hence used simpler algorithms at places where optimizations could be thought for.

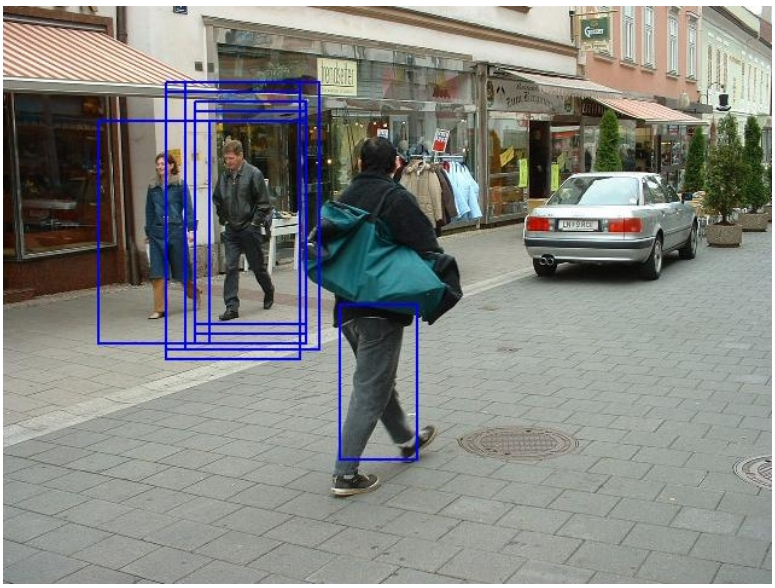
5 Sample Results/Experiments

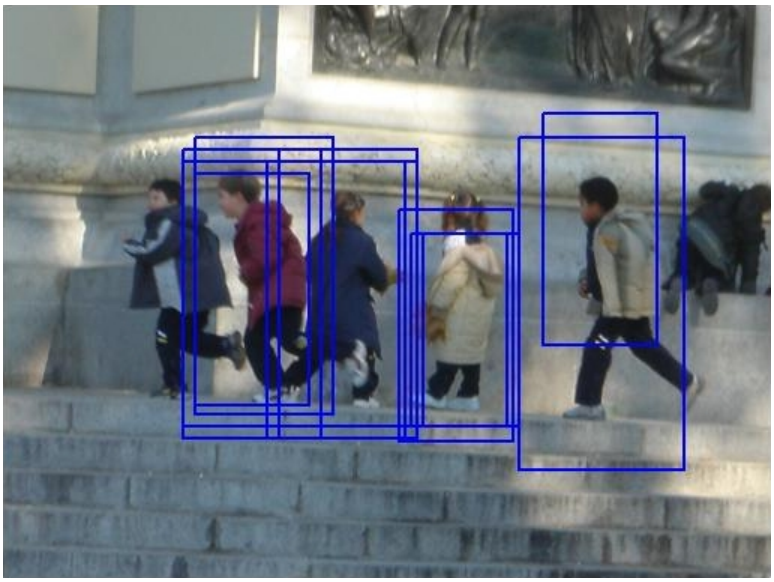
Some of the sample images are shown

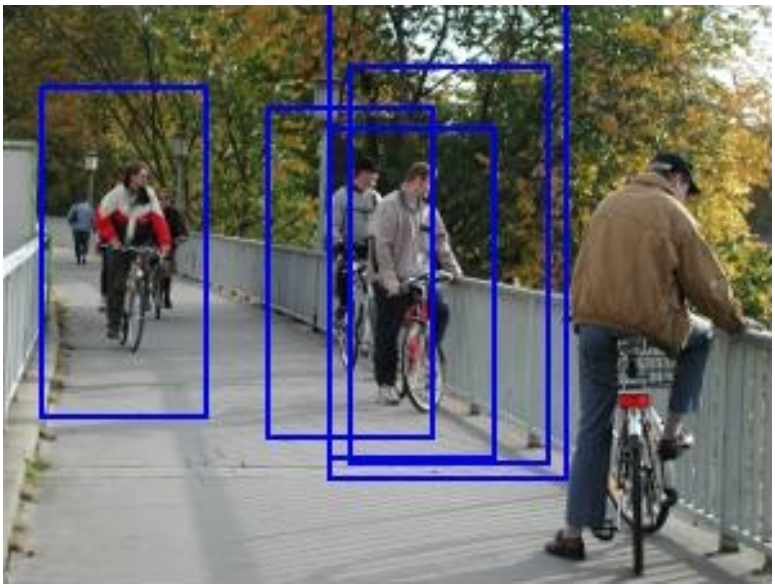


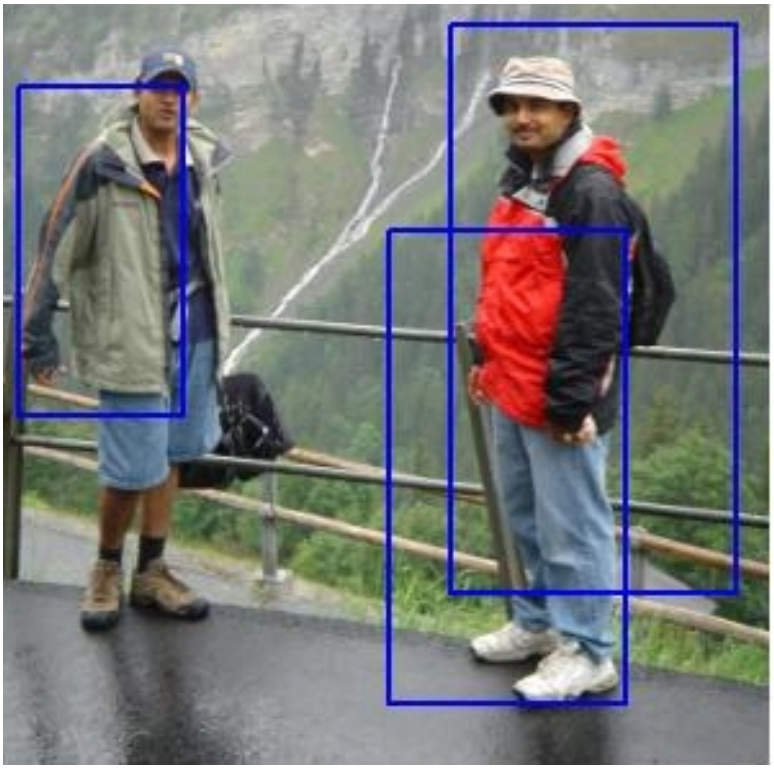












6 Difficulties Faced

- Lack of proper descriptions of algorithms

  Multiple papers I refered to assumed previous knowledge and hence refrained from giving exact details/parameter values for the various algorithms proposed. This was one major problem since I was a beginner in the field and was trying to reproduce the results stated. One example of the same was missing details in implementation of AdaBoost. Even after intense searching over literature and internet, it was considerably difficult for me to get exact information. However, I tried to contribute my research of the topic on online forums such as in [1].

7 Conclusion

I was able to implement a basic version of the algorithm proposed by Zhu, Avidan, Yeh, Kwang in [4]. It was a great learning experience, as I had to grasp concepts (to the lowest implementation details) presented across multiple papers (espl, [3], [2] and [4]), and come up with the implementation of which no other public implementation (in my knowledge) currently exists.

References

[1] An online discussion on method to reduce threshold in adaboost. http://stackoverflow.com/questions/10460265/some-details-about-adjusting-cascaded-adaboost-stage-threshold/12668159#12668159, May 2012.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[3] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[4] Qiang Zhu, Shai Avidan, Mei chen Yeh, and Kwang ting Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *In CVPR06*, pages 1491–1498, 2006.