# CS687 Project Report
# Pedestrian Detection in RGB-D Data

Md. Alimoor Reza
Computer Science Department
George Mason University
mreza@gmu.edu

Xing Zhou
Computer Science Department
George Mason University
xzhou10@gmu.edu

## Abstract

*This project is dealing with pedestrian detection problem in RGBD data. A set of learning algorithms: SVM, KNN, and Adaboost are experimented and compared. Besides, with the availability of consumable 3D sensor, we augment HOG with HOD, which indeed help to improve our detector performance. Finally, depth information is used to reduce the search space and eliminate false positives by plane estimation.*

## 1. Introduction

Pedestrian detection problem has been widely studied in the computer vision community, for instance, one of the state-of-art human detector was proposed by Subhransu *et al*. [1]. It takes about 6-7 seconds on a typical $400 \times 600$ image for 10,000 scanning windows with a missing rate of 0.5 at 10-4 FPPW. What's the problem is usually the proposed methods work quite well on their own dataset, while not so good on others as is shown in Figure 1.


Figure 1. Results given by Subhransy's human detector on their dataset(left) and ours(right).

What's more, with the advent of consumable RGB-D sensor like Microsoft Kinect, we can fuse that information in our algorithm to boost the original approach which only relies on color data.

## 2. Approach

In this project, we generally follow the pipeline proposed in [2], where HOG features are firstly extracted and a SVM classifier is then learned. Our differences with them are as follows.

a) We tried different learning algorithms and compare their performance with each other including SVM, KNN and Adaboost.

b) We use depth information to pruning window hypothesis, which reduce the search space in prediction stage.

c) By plane estimation from depth information, we obtain a probability map which gives us the probability of each region to occupied by a person. With this information we can decrease the false positive alarm rate during post-processing.

### 2.1. HOG: Histogram of Oriented Gradient

Histogram of Oriented Gradient (HOG) introduced by Dalas and Triggs [3] is currently one of the most performant and widely used methods for visual people detection. This method considers a fixed-size detection window which is densely divided into cells. In each individual cell, a 1-D histogram of oriented gradient is computed. By stacking those 1-D histograms within the detection window, we can obtain a representation of that window which is normally a high dimension feature vector. Figure 2 illustrates the visualization of gradient image and its corresponding HOG feature.

### 2.2. HOD: Histogram of Oriented Depth

HOD, similar to HOG, is computed from the gradient of the depth image as shown in the middle of Figure 2. In color image, gradient would come out due to illumination variance and textureness, which will worsen HOG features. This will not happen in depth image, and we will expect that

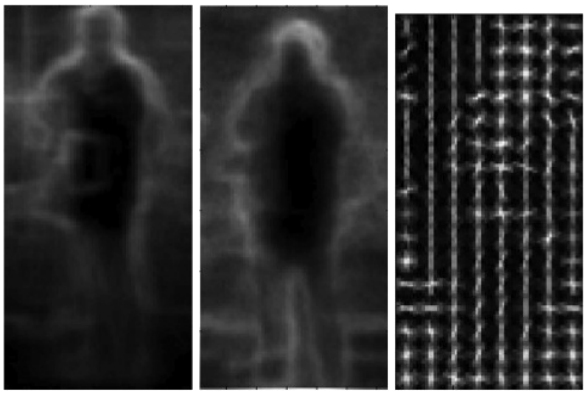

Figure 2. Left: Color gradient image; Middle: Depth gradient image; Right: HOG of color gradient image.

the performance of our detector will be improved with the use of depth information.

### 2.3. Combo-HOD

There are many different ways to incorporate the depth information into our system. One possible way is to concatenate HOG and HOD features together, and use this augmented feature to train a classifier. Another way might be training classifiers with HOG and HOD separately and combine them afterwards. In this project, we choose the later one and the final classifier is the linear combination of them.

$$p = \frac{Accu_D}{Accu_D + Accu_G} \times p_D + \frac{Accu_G}{Accu_D + Accu_G} \times p_G \quad (1)$$

where $p$ is the resulting probability of detecting a person, $Accu_D^2$ is set to accuracy of the classifier learned with HOD features, while $Accu_G$ is accuracy of the classifier learned with HOG features. This means we will put more weight to the classifier with higher accuracy.

## 3. Dataset

The dataset we are working on in this project is the RGB-D people dataset collected by Spinello [2, 4]. This dataset contains more than 3000 RGB-D frames collected in a university hall from three vertically mounted Kinect sensors. The data contains mostly upright walking and standing persons seen from different orientations an with different levels of occlusions. This dataset has also been annotated, which makes it suitable for evaluation of our work and comparison with other works. Some sample frames in this dataset in given in Figure 3.

Figure 3. Some sample frames from the dataset. First row is rgb color image, and second row is depth image.

## 4. Experiments

### 4.1. Features

HOG or HOD features are computed in a fixed-size detection window. How to decide the size of detection window? Commonly the height-width ratio of a human is more or less the same, and by exploring the dataset, we found that there are over 80% of the data have ratio 2, which is shown in Figure 4. The distribution of human height and width is also plotted on the left. In this project we choose the size of smallest detection window to be $64 \times 128$ (pixels). With the HOG cell size setting to 8, the feature dimension comes to $8 * 16 * 36 = 4608$.

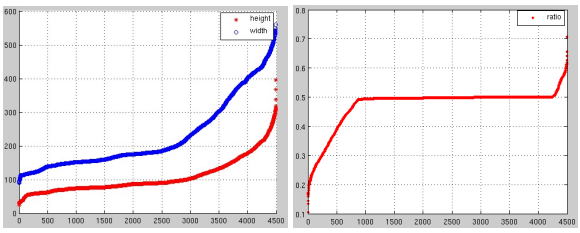


Figure 4. Left: The height and width of human; Right: Width-to-height ratio .

### 4.2. SVM: Support Vector Machine

In this project,a SVM classifier with linear kernel was trained. All the annotated data is split to two parts, 2/3 for training and 1/3 for testing. Due to the sparsity of negative examples, we choose three times the number of positive

| Method | TP Rate | FP Rate | TP Rate (Occluded) |
|---|---|---|---|
| HOG | **64.5%** | 99.8% | 53.6% |
| HOD | **65.8%** | 99.7% | 56% |

Table 1. TP and FP rate on all test data, as well as TP rate on occluded test data only.

ones. The performance on test data is given in Table 1, and the precision-recall curve is illustrated in Figure 5.

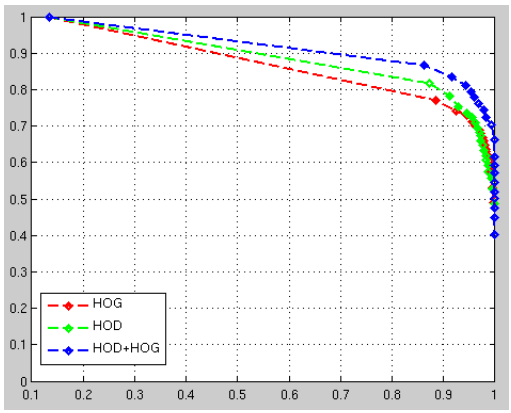


Figure 5. Precision recall of the learned SVM classifier.

There are two observations from the results. One is the learned classifier works quite well on negative examples, while the performance on positive ones are much worse. There are at least two causes that leads to the worse performance on positive examples. As the classifier was trained with more negatives, the classifier is intend to put more attention to negatives. However, through further analysis we noticed that most of the FN (False negatives) come from those test examples with occlusion. This is validated in the third column of Table 1, which shows the TP rate on occluded test examples only.

The other observation is that HOD is much more discriminative than HOG and when combing them together, the performance is the best. The reason why HOD performs better than HOG is that HOD is much more robust because the depth data can tolerate those fake gradient due to illumination variance, textureness and so forth. What's more, we can clearly see from Figure 2 that the mean gradient image of depth is much stronger than color, which makes it more robust.

### 4.3. KNN: K Nearest Neighbor

Unlike support vector machine, KNN is an unsupervised learning method. Its advantage is that it does not need training which is usually time consuming and down offline. However, KNN suffers from its space complexity. It needs to keep all the data in hand, and there does not exist any

| | K=1 | K=7 | K=11 | K=21 |
|---|---|---|---|---|
| TP Rate | 99.4% | 99.6% | 99.6% | 99.2% |
| TN Rate | 54.4% | 57.9% | 58.4%% | 59.7% |

Table 2. TP and FP rate on all test data of KNN.

theoretical way to choose the parameter $K$.

I tried different $k$'s in this project, the KNN classifier has a nice performance on positive examples, while bad on negative ones. Its performance versus different $K$ is shown in Table 2. Although the TN rate is increased when I try larger $K$, it does not mean the larger, the better. Let's take a look at the Figure 6. When one of the classes is very sparse, it's prone to assign wrong labels according to win-take-all mechanism with related to a pre-specified $K$. One possible way to overcome this problem is to use a distance threshold to define neighbors. Another way might use feature mapping to make sparse data compact in some other feature space. All of these are not addressed here.

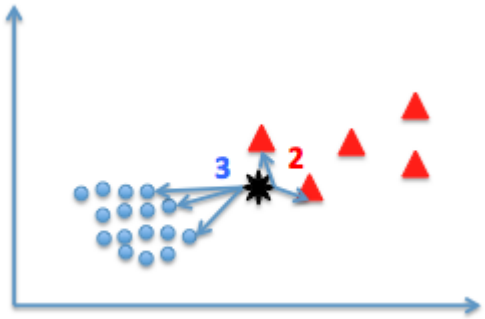


Figure 6. Problem with sparse data in KNN.

### 4.4. Ada Boost

AdaBoost is a powerful technique of combining the weak-learners to provide a more complex hypothesis that performs better as classifier [5]. For example give a 2-D data set we can learn simple linear weak classifier that perform poor separately. But in AdaBoosted settings, we can learn a collection of n-linear hypotheses h each has a learned weight alpha that are computed while learning the classifier. The goal at each stage of learning is to give more weights to currently miss-classified samples. Thereby encouraging later weak-classifier to learn them. Incorporating Decision tree as weak learner has the inherent advantage of selecting the most distinguishing feature out of the high dimensional feature vector. In general each decision tree separates the training data into desired partitions. In this project, we used the logistic regression version of AdaBoost proposed by the Hoiem et al. [6]. The weight update rule is little changed. It has the advantage of giving output as con-

| Method | TP Rate (Non-occluded) | TN Rate |
|---|---|---|
| HOG | **81.10%** | 99.57% |
| HOD | **90.61%** | 99.45% |
| HOG+HOD | **91.11%** | 99.87% |

Table 3. TP and TN rate on all test data on the AdaBoost classifier.

| Method | TP Rate (Non-occluded) | TN Rate |
|---|---|---|
| HOG | **87.86%** | 99.35% |
| HOD | **89.74%** | 99.52% |
| HOG+HOD | **94.49%** | 99.84% |

Table 4. TP and TN rate on all test data using AdaBoost on the reduced features using PCA.

fidence measure. Applying sigmoid function over this confidence gives a probabilistic output, which is very useful it classification task. In our project we learn a collection of 20 Decision tree where node weights of each trees is learned according to the algorithm 2.3 described [6]. As our SVM experiment, all the annotated data (non-occluded only) is split into two parts, 2/3 for training and 1/3 for testing. This are positive examples in our experiment. Due to the sparsity of negative examples, we choose three times the number of positive ones. The performance on test data (799 positives and 18709 negatives) is given in Table 3, and the precision-recall curve is illustrated in Figure 7.

### 4.5. AdaBoost on reduced features using PCA

Since the dimension of the HOG (or HOD) feature is large (4098), we also tested learned a classifier using the reduced dimensions of the HOG (or HOD) features. We applied PCA for the dimensionality reduction. After reduction the dimension of an exemplar became little over 800 for HOG (or 900 for HOD resp.). The performance on test data (799 positives and 18709 negatives) is given in Table 4, and the precision-recall curve is illustrated in Figure 8. Our experiments indicates that in the reduced dimension the AdaBoost performs better on the test data. The TP rate increase for all cases except the HOD in which case the TP rate (89.74%) is close to the TP rate of the previous non-reduced dimension (90.61%). TN rate remains almost the same (little difference in fractional points part).

### 4.6. Search Space Reduction and False Positive Elimination

While exploring the dataset we observed that in the indoor settings much of the image area is covered by the floor and ceiling. We applied a simple heuristic on the floor and ceiling plane in an image to reduce the search space. The depth data associated with the RGB helps us recovering the planar structure of the environment. The process of recovering the floor and ceiling area starts with over-segmenting the image into super-pixels.

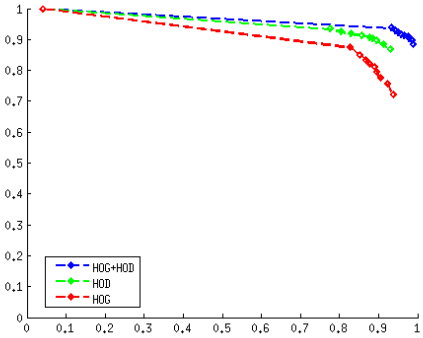


Figure 7. Precision recall of the learned AdaBoost classifier.

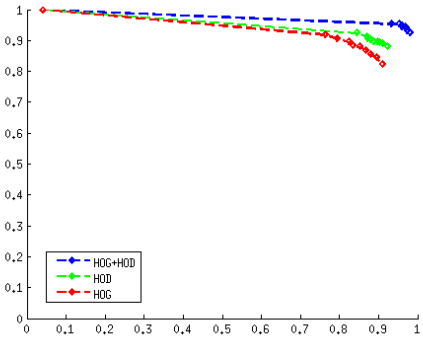


Figure 8. Precision recall of the learned AdaBoost classifier on the reduced dimension of the features using PCA.

#### 4.6.1 Super-pixels

We over-segment the RGB image using simple linear iterative clustering (SLIC) algorithm that clusters the pixels in the five dimensional color and image space by generating nearly compact and uniform super-pixels. The transformation from pixel space to super-pixels allows us to work with around 300-400 super-pixels per image. Figure 9 shows an example image after the over-segmentation where the red boundary denotes the individual super-pixels. The green and blue color super-pixels denotes the identified floor and ceiling in the image respectively.

#### 4.6.2 Plane map

Once the over-segmentation is done, we estimate SVD-based normal of the 3D point cloud that correspond to pixels inside the super-pixels. The equation of the normal for each

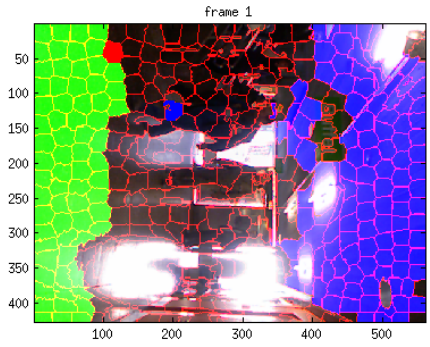


Figure 9. Superpixels and identified floor and ceiling in an image.

super-pixels has the following representation:

$$n_x + n_y + n_z + d = 0; \quad (2)$$

We estimate the floor and ceiling plane by using the sign and the magnitude of the $n_x$ co-efficient of the plane equation. This is based on the assumption that floor plane will always reside on the left side of the image and ceiling plane is on the right side. Since we want to create a map of the floor and ceil plane, we used the 1-abs($n_x$) for each super-pixels to derive a fractional number from 0 to 1 for each of the pixels in the image. Since many of the pixels the depth information are missing, we encoded the uncertainty by assigning a value of 0.5 for those pixels.Figure 10 shows the plane map we derived for the image in Figure 9.

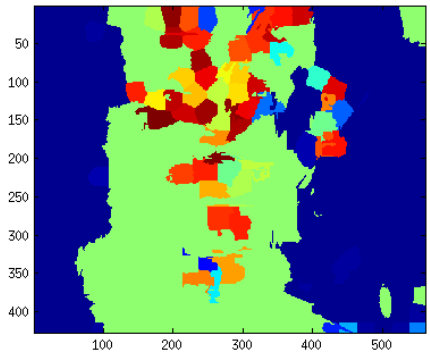


Figure 10. Plane-map

#### 4.6.3 Search-space reduction

We generate windows at 18 different scales and slide them across the image in search for the human in the image. From our observation much of the area in the image is covered by the floor and ceiling of the image. We can discard those window that significantly intersect with the area of floor and ceiling. We use the plane-map to help us prune the unwanted windows. Following table shows the percentage of overlap with floor and ceiling plane in the image and their corresponding window count. The amount of time spent is also shown in Table 5. We observed that our heuristic to reduce search space makes the detection faster on the test image without loosing any positive detection. Also some of the false positives are removed due to the fact that we do not consider a human on the detected area of the ceiling or over the floor.

| Time | # of hypothesis | % of window overlap |
|---|---|---|
| 192.149 | 11378 | No overlap |
| 158.12 | 10475 | 80% |
| 121.07 | 9170 | 60% |
| 105.31 | 8384 | 50% |

Table 5. Space reduction statistics on a sample test image.

## 5. Conclusion

Several learning algorithms are studied and compared in this project. We address the pedestrian detection problem in RGBD data, and found that depth data is not only more discriminative than color information, but also can be used to reduce the search space as well as decreasing the false positive alarm rate.

Besides, we observed that it is often the case that the distribution of positive features or negative features might be very sparse in the original feature space. This phenomenon will introduce some problems while training. This issue might be overcome by some feature mapping to make the resulted distribution more compact. This will be our future work. We applied a simple heuristic on the detected floor and wall plane. More reasoning can be done from the other detected planes that will allow us more specific places in the image to look for human detection. We would like to address this issue in the future work.

## References

[1] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. 2008.

[2] L. Spinello and K. O. Arras. People detection in rgb-d data. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, June 2005.

[4] M. Luber, L. Spinello, and K. O. Arras. People tracking in rgb-d data with on-line boosted target models. In *Proc. of The*

*International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[5] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 2002.

[6] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *Int. J. Comput. Vision*, 2007.