1. Introduction

Computers have been used intensively in our daily lives. In the past, people used them to speed up complex calculation. Moreover, computers nowadays are not only the big calculators, but it also can simulate human perspectives. From the virtual Turning machine, many researchers believe that capacity of human and computer are comparable if human’s thought and decision based on step by step process. Computer scientists have tried to extend computer’s capability in order to allow smart machine to substitute human in some certain jobs such as dangerous or poisonous ones. They have added artificial intelligent to computer to make it seems to have perspectives as human. Among enormous applications of computer, computer vision is the subject which draws most attention of computer scientists.

Our daily lives is filled of millions of objects ranging from big ones such as human, car, bicycle,… to tiny ones like cells. And the task of recognition and classification each object to its catalogue is the fundamental task for any intelligent based system. The difficulty is that a given class has a huge intra class variation. For example, human is usually thought as an object consists of up-right shape, two legs, two hands, and an omega shaped head. However, in reality, human probably appear in diverse shape. For example, people who sit down, stand up, lie down, or play sport have totally different shape. In addition, illumination, points of view are also the significant factors affect to recognition and detection process.

Thus, recently, the goal of researchers working in computer vision and intelligent based machine is to invent algorithms or facilities in order to allow computer has the ability to see and analysis a given images or videos. And one of the primary tasks is the detection and catalogue objects in images. Such ability allows us to have numerous applications such as human computer interaction, robotics, smart autonomous vehicle as well as image retrieval.

In this chapter …

* 1. Goal and Applications
     1. The goal

The main target of this thesis is to build up an automatic system which is possible to detect and localize pedestrians in static image. For more specific, it is the issue of create object detection from the view of point of computer, in which detector scan all the given image and bound the box around object if it appears in image. We use the approach which utilize robust extraction algorithm to extract a region of an image, and then use a classifier to decide whether this region contains pedestrian or not. In this cope of thesis, we just concentrate on how to encoding image regions into feature vectors which is robust on illumination, slight change and osculation.

Unlike matching one word with another word in which we can easily see they are identical or different, but matching object with object (for example, human) is the totally different matter. Natural object such as human, cat, dog and man-made object such as car, bicycle have diverse of shape, so it is difficult for computer to distinguish two catalogues. In this thesis, we use an approach which does not make strong assumption on context. For example, the context of car in cartoon is wider perspective than the car in show room. So, if we heavily depend on context of the car, we will miss the car in other view of point, such as car in painting or cartoon. Overall, the goal is to build a detector which can detect general object in wide perspective.

* Input: arbitrary image.
* Output: boundary box which contains pedestrian if image has that one.
  + 1. Applications

Robust extraction algorithm is not only useful in finding pedestrians in images, it also can be used to extract characteristic of any object. So, we can use this descriptor as a core in system of analyzing and cataloguing images in album. We obviously see that the advent of digital camera has allowed people to take photograph more easily. In 2-3 years, one personal digital camera can take as many as 10,000 photos, and which is impossible for human to manually search and locate these photos in short time. Consequently, Intelligent Management Software which can automatically add tags to these images to facilitate search is dispensable.

Moreover, person detectors are also being employed for detect pedestrians in smart cars. For instance, a warning message will appear in windshield to arouse drivers whenever the car tends to hit pedestrians or obstacle. Another application in smart car system is that cameras can detect the behaviors and consciousness of drivers in order to execute some proper assistance.

Information detected in multi-cameras will be fused together; and with the training knowledge in system, detectors will make reasoning decision to whether take a certain action or not. However, there is almost no detector which is good performance can execute in real time. For example, with the limited capacity of processing unit of portable devices, it is really hard for them to use good performance detectors in real time. Fortunately, in recent years, by the breakthrough in chip processing, and associating with some good detectors, building the software for smart cars is the subject that draws a lot of attention of researchers.

In biology field, one issue arises

* 1. Challenge

The most difficulty of building an object detector is the diverse of variation in images. These following factors effect on object detector are:

* Image is just a matter of pixel, and it lacks of motion knowledge like in video.
* Object in image suppress 3-D information and depend on viewpoint of camera as well as the scale.
* AS mentioned above, most natural object classes have huge variation in intra-class. Although two instances belong to one object class, they probably appear different on account for illumination, viewpoint, and shape distorting.
* Background information is also the vital key to prevent us from building robust detectors. Background clutter varies from image to image. For example, images can be taken from indoor, outdoor, and under diverse natural factors such as illumination, viewpoint. So, the desirable detectors have to have the ability of distinguishing object in complex background.
* In image, color and illumination of objects in one class probably varies considerable. Let’s think of a photograph taken in day with direct sunlight and shadows versus one taken in night with dim light, you easily see how the big gap they have. So, the robust detector must have capacity of resisting of changing color and illumination in object.
* Partial occlusion is an inevitable in real images. In this situation, just only a part of object can be visible. That is the reason why creating good performance detectors is very difficult.
  1. Some background of object detection

1.3.1 Image filter

The concept “image filter” has widely used in computer vision. It is use in edge detection, corner detection, blob detection, and noise elimination. A mask which is a matrix or a vector is used to convolve with an image. The most popular mask is Gaussian mask

1.3.2 Gradient

1.3.3 Non-maxima suppression

1.3.4 Linear SVM

* 1. Overview of our approach

In give image, we use sliding window to densely scan at all position at different scales. At each position, we get its score, and we decide this window contains object or non-object via classifier. This method is purely based on statistic approach which disregards the fore-given context of any object class. When extracting region containing object, we assume that there are some invariants which are not change dramatically within one type of object. These invariants become the main characteristics for classifier to distinguish this object class with other object class. So, by extracting invariants of object or non-object, we can represent them in high dimensional vector. And we assume that it is possible to build up a hyper-plane which separates object, non-object point as far as possible.

We just focus on method of represent robust features in order to robust from slightly changes in shape, illumination and scale. The classifier used in this thesis is Linear Support Vector Machine (stand for SVM). Recently, SVM have widely used in machine learning. And in computer vision, it is intensively used in learning process. We use SVM because it is simple, runs fast, and has good performance.

For more specific, in extracting feature process, we use locally normalized Histogram of Oriented Gradients (HOG) as a descriptor. HOG is computed from gradients of image and has the characteristic that robust to (1) small changes in image contour locations and directions, (2) significant change in image illumination and color, (3) remaining as discriminative and reparable as possible. We use weighted histograms gradient orientations over spatial neighborhood to calculate HOG features. Before calculate histogram of gradients, we do some pre-process to eliminate the effects of illumination and color changes. So, the histogram of oriented gradients has information of the contour of the object.

Once we densely scan image, we will get a bulk of windows at level classifier which means that each window is now represented as high dimensional feature vector. Note that we scan all position at multiple scales, so there are probably some windows overlap each other. After that, we suppress all window whose score below the threshold, and keep and positive windows (exceed the threshold). Because HOG is robust to slight changes in shape and contour, it is possible to have many positive windows contain same object. To resolve this problem, we fuse all positive ones and use non-maxima suppression to find only one window most likely contains object of a class. In this thesis, Mean Shift is used as a suppression algorithm in this process.

After densely scanning image, we filter windows which exceed the threshold. We assume that our extract algorithm is robust which means that the score of window is still large (not maximum) even if this window slightly off center of object. And then, we use non maximal suppression (such as mean shift) to find the mode.

In process of extract a slide window to feature vector, we use robust extraction algorithm to avoid the noise effects of background. And HOG (histogram of gradient) seems to be the best choice because it is invariance with variety of background. In realistic, HOG is declared as the state of the art in extract feature vector to distinguish object or non-object.

INRIA pedestrian dataset is intensively used in training phase and testing phase. In training phase, there are 2416 positive windows and 1220 negative images; and there are 1126 positive windows and 453 negative images.



Pedestrian/ non-pedestrian classifier

Figure 1: Overview of detection method

1. At each point in image, we densely scan with multi-scale. (b) Sliding window is detected and extracted to feature vector which is input to pedestrian/non-pedestrian classifier.

**Overview of extraction algorithm**

* Normalize gamma, using equalizeHist function in OpenCv in each channel color.
* At each color channel, compute gradient of each pixel using centered mask, the info of gradient is the weight and orientation. And at each pixel, we choose color with greatest magnitude of gradient.
* Divide window into many equal squared cells.
* In each pixel, we calculate how it contributes to histogram of cell which contain it, as well as the neighborhood cells. (spatial and orientation histogram). (See figure 2)
* In the window, blocks are created by group of cells. Blocks can overlap each other.
* Concatenate histograms of blocks to form feature vector of window.



Figure 2: Example of Spatial & orientation histogram

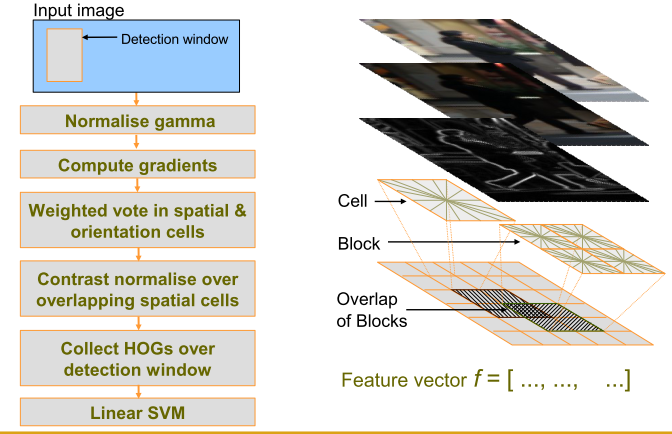


Figure 3: Overview extraction algorithm Histogram of Oriented Gradient

* 1. Overview of result

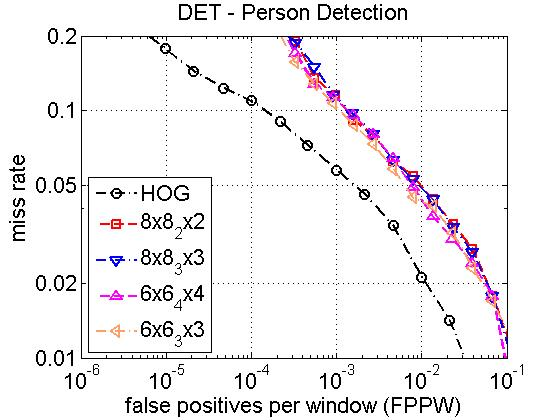


Figure 4: Performance of Dalal and us. Dalal’s curve is the first one (black).

* 1. Outline

1. Related work
2. Overview of Detection Methodology and results

In this thesis, we use a method to map local image regions to high dimensional feature spaces. To encode the static image, we use HOG approach which heavily base on image gradients. The following sections will describe in more details of detector framework of HOG.

3.1 Overall framework

Our object detector consists of two main phases called training phases and detection phase. In training phase, we use training dataset to create binary classifier which provides object/non-object decision for fixed size image’s region (usually call window). And in detection phase, we densely scan whole image at multiple scale and use the classifier derived from training phase to explore positive region in test image (positive region is the one that likely contains objects). After receiving all positive windows, they are fused together to have final detections by non-maxima suppression algorithm. The performance of final step is mostly depended on the reliability and robustness of classifier.

3.1.1 Training phase

Input image

Scan image at all position and multi scale

Extract feature over windows

Put feature vectors into linear SVM

Detection window

Normalize gamma

Compute Gradients

Weighted vote in Spatial & orientation cells

Contrast normalize over overlapping spatial cells

Collect HOGs over detection window

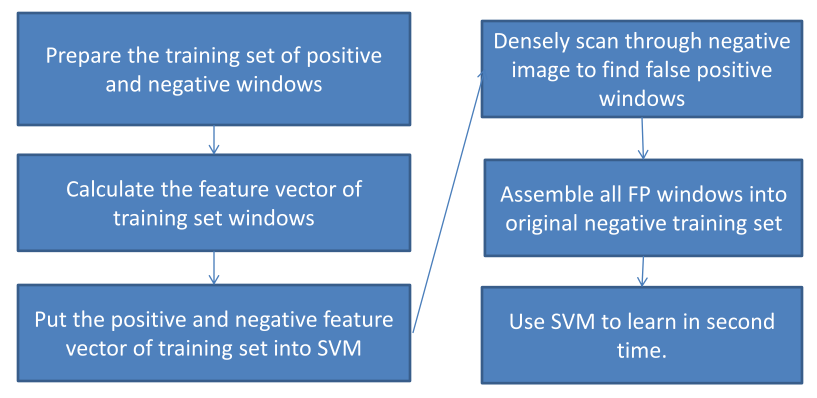
HOG

The first stage of training process is the preparation of training data. The training set consists of positive and negative windows which have the fixed and same sizes. Positive training windows contain object at the center, while negative training windows are the arbitrary sub-sample that does not contain any instance of object. The positive window is the one ideally contains only one centered object, so the number of positive training sample is very limited. On the other side, the number of negative training sample can be very huge because for example, one natural scene image can generate approximately 10,000 negative windows. As a result, the classifier will be very sensitive with negative sample. For example, let take of one image, we can generate 10,000 windows. And we assume that the false positive rate (false positive is the item indicates that window is actual negative, but the classifier says it as positive) is small as 0.1%, it can generate 10 false positive windows which can severely effect on fusing all detections to create the final one.

In this stage, we use a method called Histogram of Oriented Gradients to map each training window to high dimensional feature vector. After that we use some Machine Learning technique to build up a classifier. There are three method are put in our consideration. Initially, we thought that we can employ maximum likelihood paradigm to create a linear separator between positive and negative samples, but we find out that it is unrealistic to do that because the derived vector is huge, over than 2,000 dimensions. In maximum likelihood method, we have to calculate inverse matrix, so it is impossible for moderate computer to find inverse matrix of huge one in short period of time.

The other two learning methods are Ada-boost and linear SVM which are all common. When doing this thesis, our main target is to investigate and find out the solid descriptor that can transform image region to vector. So, we choose linear SVM because it is simple and reliable classifier. There are three properties of linear SVM which make it valuable are: (1) it converges reliably and repeatedly during training process, (2) it handles large dataset gracefully, (3) and it has good robustness towards different choices of feature sets and parameters.

We have to take into account that the number of negative windows is much larger than number of positive windows. However, it is impossible for us to put all positive and negative training windows into training set at one time because the shortage of memory of computer. That is the reason we have to train dataset multiple times. We will briefly the method which is used.



* Firstly, we gather whole positive training windows and some negative windows (the number of negative ones is as many as 5-7 times number of positive ones). Then we put them into SVM as a training data.
* Secondly, when SVM learn this dataset, it will output a classifier. This classifier cannot be used immediately because it will increase the false positive rate.
* Thirdly, we densely scan through whole set of negative images to find false positive windows.
* Fourthly, we concatenate all false positive windows derived from third step to the negative sample in training set. This stage is usually called “find hard negative sample”. And we use linear SVM to train this new training set to create new classifier.
* Finally, repeat third step if you wish to find more false positive windows. By experiment, the process of finding hard negative sample is employed one or two times will give good performance.

3.1.2 Detection phase

Scan image at all scale and location

Extract feature over windows

Run linear SVM classifier at all locations

Fuse multiple detections at all position and scale space

Object detection with boundary box

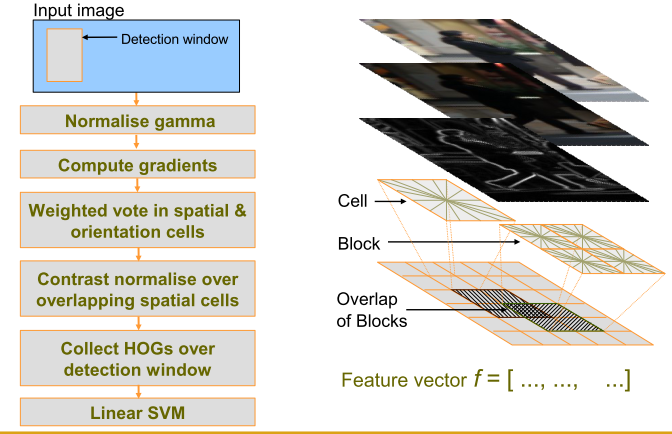
The goal of this phase to find out all positive windows of given test image. During detection procedure, the given test image is densely scanned at all scales and locations. For each scale and location (called window), we compute the feature vector of detection window. And we use the classifier derived from training phase to make the decision of whether this window is positive or negative (contain object or non-object). Because it is possible that many overlapping positive windows contain same object, it is necessary to fuse all detection windows to find the final ones.

3.2 Overview of feature sets

The feature sets used in this thesis based on dense and overlapping encoding of image regions using Histogram of Oriented Gradients descriptor. This descriptor is a statistical approach which regards the orientation of gradients in image. Dalal has proposed two types of HOG called static and motion HOG. Static HOG descriptor is used to extract image region feature, while motion HOG one is used in video. And because our target is to detect object in static image, we will use static HOG descriptor to extract image regions characteristics. In this thesis, we use term HOG instead of static HOG to indicate static HOG descriptor.

HOG which is based on the characteristics of well-normalized local histogram of orientation of gradients will be described by following steps:

* Apply normalization to the image to reduce the influence of illumination effects. In this project, we use square root method to each color channel. By observation, normalize image by square root has increased the performance a lot by prevent the effects of shadow and illumination.
* In the second step, we compute the first order image gradients. These gradients contain information of contour and some texture of object. In addition, gradient is resistant on illumination and color variation. Once completely compute first order image gradients of each channel, we choose the dominant color channel.
* Thirdly, like SIFT descriptor; local image region is encoded into high dimensional vector by concatenate many local spatial histograms of gradients. Image window is divided into small non-overlapping regions called “cell”. For each cell, we compute the histogram of gradients over all pixels in the cell by accumulating the magnitude of each pixel gradient into bins which are the range of orientation of gradients. The detail will be described in section 4.
* After receiving histograms of each cell, we take a local group of cells and normalize them. This normalization step will help the feature vector resists to variation of illumination, shadowing, and edge contrast. The group of local cells is call “block”. In this stage, many blocks can be overlapping each other, so they share some same cells. This seem redundant, but in practice this can enhance the performance of descriptor because this give us more information about image region.
* Finally, collect all HOG descriptors from all dense overlapping blocks of detection window into big feature vector for use in the window classifier.



The HOG descriptor has several advantages which are:

Figure 5: Overview extraction algorithm Histogram of Oriented Gradient

* Capture the contour information. For example, in HOG descriptor, information of edge or shape of object is stored in histogram of cells and blocks. So HOG contains the characteristic of local shape. In addition, when put together all overlapping blocks, we probably get relevant information while still maintain invariant.
* When object translates or changes a little bit, it make little different in histogram if these changes are smaller than the local spatial or orientation bin size.
* Illumination invariant is assured by gamma normalization and contrast normalization.
* The overlapping blocks has a benefit that it allows little information can be missed during the encoding process.

3.3 Fusion of multiple detections

In the detection phase, image is densely scanned at all locations and scales. This probably creates a lot of overlapping detections for one instance of object. The reason is that a detection window probably gets positive score although it is slight off object center. So, the detection windows need to be fused together. Intuitively, during the detection phase, we observe that although the number of detection windows is much larger than the number of instance object, these windows are most likely concentrate around objects. Hence, from this observation, we can employ clustering algorithm to find the right position of instance object. There are two well-know and traditional cluster algorithm which are K-mean and mean shift. And then, we decide to use Mean Shift because the number of object in image is unknown.

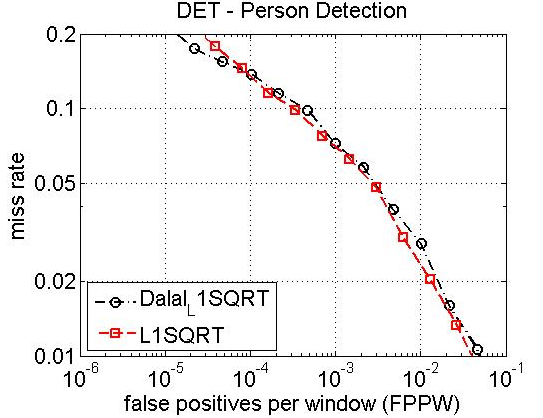
3.4 Overview of results

The main purpose of this thesis is to rebuild the HOG descriptor and improve some aspects in HOG, so we will compare our results with Dala’s work.

At first, we re-implement HOG detector of Dalal, and its performance is approximate with Dalal’s one. After that, we propose some slight contributions which are reduction of dimension of feature vector and increasing performance by adding multi-level.

The first contribution is that we try to reduce dimension of feature vector, and we get the performance still as good as Dalal’s one. In the second contribution, we enhance HOG detector performance by adding some information; and its performance are higher than original one approximately 2%. These contribution will be discussed in detail later in section 5.

(2 bieu do o day)



1. HOG

In this section, we are going to introduce in details about HOG feature sets. There are a lot of parameters in HOG feature, and their effects are very different. For example, some parameters slightly effect on HOG performance, while other some ones considerably influence on the efficiency of HOG. Overall, one conclusion we can draw out is that HOG encoding feature sets gives outstanding performance compared with other existing method such as SIFT, Haar wavelets.

4.1 Static HOG Descriptor

Histogram of Oriented Gradients is indeed the dense and overlapping description of image region. There are four HOG variants which are:

* Rectangle HOG(R-HOG): it looks like SIFT descriptor, blocks use overlapping square/rectangle grids of cells. The descriptor blocks are computed over the dense uniform grids. And each block is normalized independently. The parameters of R-HOG descriptor are ϛ x ϛ, η x η, β which are number of cells in one block, number of pixels in one cell, and number of bins respectively.
* Circular HOG(C-HOG): it seems to be similar to Shape-Context. In C-HOG, cells are defined into grids of log-polar shape instead of square or rectangle. At each center of grid point, we divide local image patch into a number of angular and radial bins. The angular bins are uniformly distributed over the circle, and bins will be increased as big as they are far from the center.
* Bar HOG: similar to HOG, but it also uses second order derivative instead of first derivative. After that, we collect histograms of both first and second order derivative. The advantage of this approach is that Bar HOG has additional information about bar and blob.
* Center-Surround HOG: in R-HOG and C-HOG, each block is normalized independently, so one cell can be normalized redundantly. It seems that optimal computation cost will not reach. To overcome this issue, in Center-Surround HOG, every cell is normalized just only one time. So it speeds up computation.

4.2 Implement and Performance Study

In this thesis, we choose R-HOG as our default descriptor because of shortage of time and its excellent performance. We now describe details about how to implement R-HOG as well as give out the effects of parameters. For all experiments, we use Detection Error Tradeoff Curve to show the performance.

As mentioned, there are several variants influence on HOG description performance. In this section, we are going to describe effects of main factors.

4.2.1 Color channel

This section will give the evaluation of pixel representations including gray-scale, RGB, LAB, and SHP. According to our experiments, the performances of RGB and LAB are similar and they are outstanding the rest. While the pure gray-scale and SHP reduces the performance 2% at 10-4 FPPW (false positive per window).

4.2.2 Color/Gamma Normalization

As mentioned above, images of dataset have to be pre-implemented before encoding HOG feature vector. A raw test image can be beautiful with human’s viewpoint, but it can be very difficult and vague for computer’s perspective due to effects of illumination and shadowing. Hence, normalizing gamma and contrast of image is necessary. There are two popular normalization methods which are “square root” and “log scale”. Their results are similar. In this thesis, we “square root” method because it is faster than the other. By experiment, performance will be boosted by 7% when using “square root” normalization method.

4.2.3 Gradient Computation

Gradient is the term that indicates the change of pixel in image. Hence, by employing gradient information, it allows us to get and encode the shape and contour of object in image.

We compute gradient by calculating first order derivative of pixels in image. In computer field, there are several ways to estimate the changes of pixels in image. There are plenty of masks used in convolution of image. However, the simple mask [-1 0 1] give the best outcome according to experiment.

* Compute first order derivative of each pixel on Ox, Oy coordinates:



One sided: Sx = (vẽ 2 hình minh họa, )



Sy =

-1

1

Corresponding mask:

(2 mask)



Two sided: Sx =



Sy =

-1

0

1

Corresponding mask:

Note: ‘h’ is usually taken as 1.

* Calculate Gradient: after getting Sx , Sy which are two first order derivatives on Ox, Oy coordinates respectively, we can use them to calculate the magnitude and orientation of pixel.



* + Magnitude:



* + Orientation:

And one importance notice is that we definitely should not smooth or blur image before computing gradients. The most likely reason is that edge informative is essential to descriptor, and if we blur image, we lose a lot of edge information.

4.2.4 Spatial/Orientation Binning

After calculating gradients, we will get a peck of gradients of pixels which consist of magnitude and orientation. And each gradient contributes a weighted vote for orientation based on the orientation of gradient itself. The orientation bins of cell will be accumulated by the weighted vote of its pixel gradients. The orientation bins can be over 0-180o (“unsigned” gradient) or 0-360 (“signed” gradient). Moreover, in order to avoid bias, we use tri-linearly interpolation voting method which is regards orientation and position matters to vote to cell bins. This idea is illustrated by below figure.

The number of bin of histogram of cell is also the factor effect a lot to performance. The performance of β=9 is significant better than of β<9 (β is the number of bins of histogram in cell). However, performance will not increase much when β exceed 9.

“Signed” or “Unsigned” gradient is also the matter put into concern. The natural object such as human, cat dog can be diverse in shape and contour. Hence, “signed” gradient is unsuitable to be used because this probably reduces the performance. At this circumstance, “unsigned” gradient give best results. In return, “signed” gradient gives very good performance for objects which are man-made because their shapes are likely constant.



Figure 6: Example of Spatial & orientation histogram

4.2.5 Block Normalization, Block Size, and Overlap

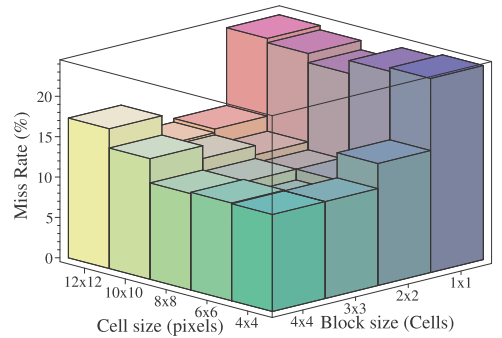
Block Normalization has a great effect on HOG descriptor. The fact that Gradients strengths vary from over a wide range due to local variations in illumination and foreground-background contrast. Hence, it is necessary to normalize block to get good performance. Block is a local group of cells, and each block is normalized separately.

We will evaluate four different block normalization types. Let **v** be the un-normalized block descriptor vector, and ε is a very small number employed to avoid division by zero. The four normalization types are :

* L2 norm: **v** 🡨 sqrt(|**v**|22 + ε2)
* L­2-Hys: L2-norm by clipping and renormalizing.
* L1-norm: **v** 🡨 **v**/(|**v**|1 + ε)
* L1-sqrt: v 🡨 sqrt(**v**/(|**v**|1+ ε))

For pedestrian detection, the performance of L­2-Hys and L1-sqrt are equal, and they are outstanding to the rest. By experiment, using L1-norm reduces performance 15%. And the value ε should be taken in range 1e-3 – 5e-2.

We are now investigating the effect of block size. For pedestrian detection, 3x3 cell blocks and 6x6 pixel cells gives best result with 11% miss rate at 1e-4 FPPW. According to Dalal’s experiment, cell size varies form 6 x 6 to 8 x 8 and block size varies form 2 x 2 to 3 x 3 gives the best performance for all kind of objects. The most likely reason is that if the size of block or cell is too big or too small, the valuable spatial information will be lost.

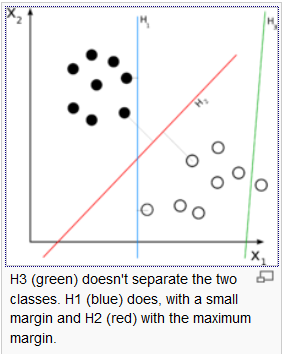


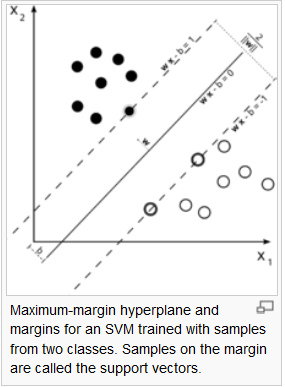
4.2.6 Classifier

Recently, classifier is the very active filed drawing a lot of researchers’ attention. Building the robust classifier is the other main key in any detector because detector is nothing if we had good descriptor but bad classifier. Among many classifiers, SVM and adaboost have very outstanding performance. We choose linear SVM as our classifier because it is simple and we have work a lot on it. Although linear SVM is simple, it runs really fast and has good performance.

**Support vector machines** (**SVMs**) are a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes the input is a member of, which makes the SVM a non-probabilistic binary linear classifier. Since an SVM is a classifier, then given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. Intuitively, an SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.





We use soft linear SVM with default c = 0.01. Non-linear SVM enhances the performance by 3% at 10-4 FPPW, but the tradeoff is that it takes much more time in computation.

4.3 Hard Samples

In HOG descriptor, a given image is densely and uniformly scanned at all locations and scales to detect positive windows. Because the size of detect windows are very small compared with the image, there are a lot of detect windows in one image. By experiments, we find out that an average image (450 x 600) has approximately 10,000 detect windows. Among these windows, just a few ones are positive, and almost are negative windows. Hence, HOG detector is very sensitive with false positive windows. As mentioned above, even a very small false positive per windows also adversely affects the performance of detector. Therefore, reducing the false positive rate is the vital thing to do.

Remind that HOG descriptor based on the well-normalized densely histogram of oriented gradients. HOG feature sets contain a group of very high dimensional vectors which occupy a lot of memory. When using SVM to learn, every feature vector has to be loaded into RAM to process. As consequent, we cannot take negative samples as many as we can because of limitation of hardware.

To overcome this obstacle, we employ a method called “Generating hard samples”. The flow work of it is illustrated below steps:

Positive samples

Negative samples

Training samples

Classifier

Training negative image set

False positive windows

* Firstly, training set is collected as the combination of positive and negative samples. Each of samples is a HOG description feature vector.
* Secondly, we use linear SVM to train the training set. The result is a classifier.
* Thirdly, we randomly pick up some points in negative image in negative sample set. After that we get a group of detection windows at all scales taken this random point as a center.
* Next, we use the classifier in second step to score the detection windows from third step. In this step, we are just interested in the window having negative score.
* Finally, we get all the false positive windows (window has negative score) and put them into the original negative samples. And we start the process again until we do not get new false positive ones, or the number of false positive ones is under some criteria.

Detector’s performance is boosted significantly during the process of finding out negative hard samples, approximately by 15% at 10-4 FPPW. We all know that the number of positive samples is limited because it takes a lot of effort to collect and annotate. On the other side, the negative samples are huge and definitely larger than positive ones. Negative samples can be collected from a lot of scenarios including indoor, outdoor, natural scene, street, and so on. Hence, high false positive per windows rate can severely affect detector’s performance.

Overall, the process of finding the hard negative samples (false positive samples) is essential for good performance detector. It can enhance a lot the performance of DET cure (Detect Error Tradeoff cure). However, there are two main drawbacks of this process. Firstly, it takes considerable time for finding out hard negative samples. According to our experiment, the more hard samples we collect, the better performance is. However, due to the enormous number of windows in image, the total number of windows which we have to scan through is very huge. From the pedestrian INRIA dataset, we have to examine approximately 2,000,000 windows to draw a few thousands of hard samples, and it takes us nearly 3 days. Secondly, while we gather hard negative samples and put them into negative training set, we also increase the miss rate of detector.

4.4 Overall Results

Our main training and testing dataset is INRIA Pedestrian dataset.

4.5 Conclusions

1. Some modified HOG

In this previous section, we see that the more overlap of blocks, the better performance is. However, the block overlapping accompanies with the size of feature vector. Hence, if we enhance performance by increasing blocks overlapping, we will also reduce the program performance because of the expanding of feature vector. Thus, we propose some methods to reduce the length of HOG feature vector. In addition, we also introduce the method called “multi-level” to increase the performance of HOG descriptor.

Because of shortage of time and limitation of computer, we just only test these new approaches on pedestrian object which is people have up-right shape and full visible. We cannot guarantee whether these approaches can be applied to the other objects or not.

5.1 Four regions based approach

This approach is only useful when the hypothesis which is the pedestrian is up-right shape and central alignment is hold. This assumption is assured in MIT and INRIA pedestrians dataset. We observe that there is a small region in the center of window which mostly contains chest and stomach is less informative. The reason is that this region usually falls into internal part of pedestrian’s body. Hence, this small region is often covered by colors of cloths without shape curve information.

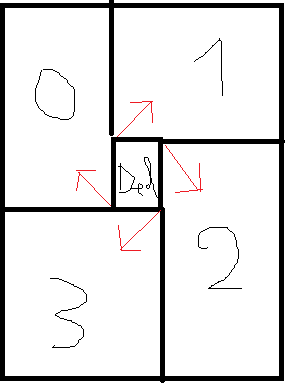


Figure 7: model of four regions based method

“Del” region is less informative than the others. We separately calculate histogram of these four regions, and concatenate them together.

By experiment, we observe that “Del” region should be less than 18.25% of window size (one and a half cell over eight cells) in order to maintain the same accuracy with original one. However, if we choose “Del” region too small, the size of feature vector of window will increase correspondently.

One more thing that significantly affects performance is the overlap of regions. The more these regions overlap to each other, the more accuracy it is. Nonetheless, percentages of overlap of regions accompanies with the size of feature vector.

The performance of this new one is approximate with the original one thought the length of new feature vector is reduced by 15-25%.

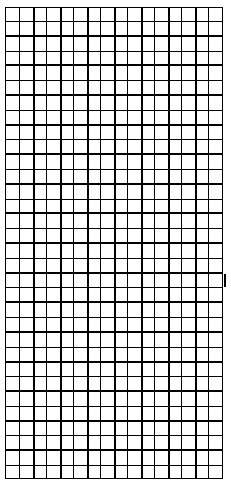


Figure 8: effect of size of "Del" region and overlap of regions on performance

5.2 Non-uniform based approach

The idea is similar to the four regions based approach, this approach also try to take advantages of less informative center region in the training image. However, instead of divide image into four region and ignore the central one, we use the distribution of grid points to catch this idea.

In the original HOG descriptor, image is divided into uniform grid of points. As a result, the author of HOG, Dalal, has assumed that every part in this image is equally important. However, in fact, the information about the background is usually less important than the body curve shape of human. Thus, from this observation, we devise a method call “non-uniform based approach” which is to concentrate grid points on the much informative regions and to loosely distribute grid points to less informative regions.



105 point

84 point

Unfortunately, the result of this approach from the experiment is below the original one. More specific, this performance is 7% lower than of the original. The reason probably is that out intuitive about less informative regions is wrong. Because of the shortage of time, we do not intensively evaluate this method on other case of grid points scatter.

5.3 Multi-level based approach

The original HOG method of Dalal is just one level. The drawback of this one is that it is less informative about structure of object. Consequently, I propose a new concept called “multi-level HOG”. The concept multi-level is not new. Many researchers use this one to enhance performance of certain extraction algorithm.

In paper “Classification using Intersection Kernel Support Vector Machines is Efficient”, author has proposed a method named multi-level which is calculated as following steps:

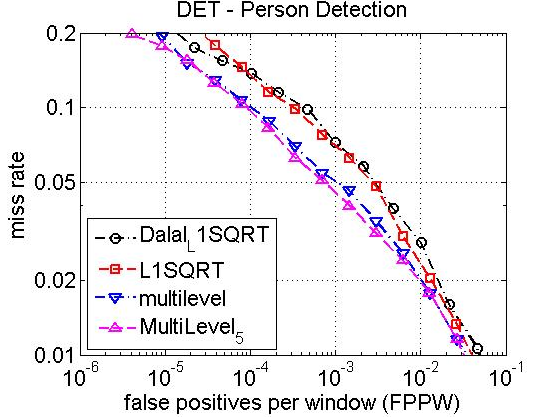
* Step 1: At level one, window is divided into non-overlap cells. And the feature vector of this level is the concatenation of HOG feature of each cell.
* Step 2: size of non-overlap cells is increased or reduced to form anther level. And it is similar to step 1 to calculate window feature at this level.
* Step 3: Concatenate all these windows feature to create final window feature vector.

However, my concept “multi-level HOG” is a bit different from the above method. At each level, instead we calculate window level feature using whole technique of Dalal and Trigg. The following steps describes in details:



* Calculate the gradient of each pixel.
* At reach level, we create uniform grid of points as above figure. Different levels have different grid as well as cell size.
* Calculate each level window using HOG.
* Concatenate all levels to form final window feature vector.

Here are the results of multi-level approach compared with the original one. At 3 levels method, we enhance performance by 2%. Moreover, we intuitively think that we will get better performance as far as we increase the number of levels, but it is not hold. The performance of 5 levels method is approximately the same as 3 levels one.

1. Multi scale object localization

Chapter 4 gives details to build up a detector to object/non-object decision. We densely scan through image at all positions and scales. This leads to several detections overlap each other. So, the fusion stage is necessary to yield the final object detections.

This section proposes a solution for fusion of multiple overlapping detections. The overall of this method has been described briefly in chapter 3. In this chapter, we will go through the detail of the method of fusing multiple overlapping detections.

6.1 Binary Classifier for Object Localization

Object detection and localization which base on scanning detection windows requires the fusion of overlapping detections. Our fusion method is held if these following assumptions are true:

* If the detector is robust, it should give a strong positive (though not maximum) response even if the detection window is slightly off-center or off-scale on the object.
* A reliable detector will not fire with same frequency and confidence for non-object image windows.

The first hypothesis assumes that the detector response degrades gradually under small changes in object position or scale, but that the maximum response occurs only at the right position and scale. The second hypothesis implies that false positives are mainly due to accidental alignments, so that their probability of occurring consistently at several adjacent scale levels and positions is low.

In addition, the fusion method is based on these following characteristics:

* The higher the peak detection score, the higher the probability for the image region to be a true positive.
* The more overlapping detections there are in the neighborhood of an image region, the higher the probability for the image region to be a true positive.
* Nearby overlapping detections should be fused together, but overlaps occurring at very different scales or positive positions should not be fused.

The third characteristic is based on the observation that the windows used to learn binary classifiers can be larger than the object to allow some context. Thus there may be scenarios where detection windows overlap for nearby objects.

We now present the method called Mean Shift which enables us to clutter the distributed points to proper groups.

6.2 Mean Shift

6.2.1 Brief introduction to Mean Shift

 m(x) = \frac{ \sum_{x_i \in N(x)} K(x_i - x) x_i } {\sum_{x_i \in N(x)} K(x_i - x)}  Mean shift is a procedure for locating the maxima of a density function given discrete data sampled from that function. It is useful for detecting the modes of this density. This is an iterative method, and we start with an initial estimate *x*. Let a kernel function ***K***(*xi* − *x*) be given. This function determines the weight of nearby points for re-estimation of the mean. Typically we use the Gaussian kernel on the distance to the current estimate,  K(x_i - x) = e^{c||x_i - x||^2} . The weighted mean of the density in the window determined by *K* is:

 K(x) \neq 0 Where *N*(*x*) is the neighborhood of *x*, a set of points for which . The mean-shift algorithm now sets x 🡨 m(x) , and repeats the estimation until *m*(*x*) converges to *x*

6.2.2 Pros and Cons of Mean Shift method

Pros:

* Does not require the prior knowledge of the number of clusters, and does not constrain the shape of clusters.
* Has good performance compared with other clustering algorithm.

Cons:

* The only parameter in Mean Shift is the radius to determine the neighborhood. And an issue is arisen how we determine the radius parameter. There is a tradeoff between the accuracy and running time when we choose radius. If we choose large radius, the program will run very fast, but the outcome’s performance cannot be guaranteed. On the other hand, if we take small radius, the performance is good, but it takes much time to run.
* Mean Shift is more complex and slower than K-Mean.

6.3 Algorithm

In our thesis, we do not use directly the original Mean Shift method. More specific, we add one more parameter to take into account on the overlapping detection windows. From experiments, we observe that one instance of object is usually detected by several nearby windows. Moreover, one important point is that false positive windows are usually distributed randomly. Hence, any cluster which contains instance of object usually has several detections. So, the new parameter is the number of detections in one cluster. If the number of entity in one cluster is lower than some criteria, we will dismiss this cluster without leaving any adverse effect on performance.

The details of algorithm are described below:

|  |
| --- |
| Input:   * Test image * Trained window classifier * Scale step, the radius of neighborhood, and the minimum number of detections in one cluster   Output:  Bounding boxes of object detections |
| HOG descriptor:   * Detect windows at all locations and multiple scales * Use HOG descriptor to extract window to high dimensional vector. * Use classifier to take positive windows. |
| Mean Shift   * Consider each window detection as a weighted 3-D point which dimension are two dimensions in image and scale. And the weight of each point is the score of itself. * At each point, we determine the neighborhood and use the equation(6.2) to calculate the mean. * Assign the mean back to the point * Iteratively for each point until it converges to the mode. * For each mode compute the bounding box from the final center point and scale. |
|  |

1. Dataset
2. Conclusion
3. Future work