Human Detection Based on Histograms of Oriented Gradients and SVM

Haofu Liao, BSEE

April 23, 2013

Department of Electrical and Computer Engineering
Northeastern University
Contents

I. Introduction ................................................................................................................................. 3

II. Image Encoding ......................................................................................................................... 4
   A. Overview of static HOG feature extraction ........................................................................... 4
   B. Gamma Normalization and Gradients computation ............................................................... 4
   C. HOG Descriptor ..................................................................................................................... 5

III. Classifier and Learning algorithm ........................................................................................ 6
   A. Support Vector Machine (SVM) ............................................................................................. 6
   B. Window based Classifier Learning Algorithm ......................................................................... 6

IV. MATLAB Implementation And Toolbox Design ................................................................... 7
   A. File Based Human Detection Implementation ....................................................................... 8
   B. Graphic User Interface (GUI) Based Human Detection ........................................................ 8
      1. Interface and functions ......................................................................................................... 9

V. Experiment and Statistic Result .............................................................................................. 10
   A. Data Sets .............................................................................................................................. 10
   B. Experiment and Statistic Result .......................................................................................... 10

VI. Conclusion ............................................................................................................................. 11

VII. References ............................................................................................................................ 12
I. Introduction

Human detection in images is a popular topic in recent years. It has lots of applications in different areas such as pedestrian detection for smart, cars film & media analysis, and visual surveillance. However, it is also a very challenging work because of the variability in appearance due to clothing, articulation and illumination condition that are common in outdoor scenes.

Recently, Dalal & Triggs presented a human detection algorithm with excellent detection results. Their method uses a dense grid of Histograms of Oriented Gradients (HOG), computed over blocks of size $16 \times 16$ pixels to represent a detection window. This representation is proved to be powerful enough to classify humans using a linear SVM.

In this report, we first introduced some basic knowledge of HOG and SVM. Then, implemented the image encoding part and classifier learning part of human detection in MATLAB. And provide the beginner a toolbox to illustrate the technique details of human detection. Finally, we gave the conclusion and statistic results of our implementation. The overall architecture of our work is below:

This report is organized as follow: section II introduced the image encoding theory (HOG).
Section III introduced the classifier of human detection we used. Section IV designed a toolbox that can help user training and testing the human images. Conclusions are shown in Section V.

II. Image Encoding

To train our classifier, we need to build our training set first. Use original images as our training set is obvious an inefficient choice. Hence, we need to encode our images into a feature space before we use them to train our classifier. The encoding technique we used here is Histogram of Oriented Gradient (HOG). It is proved to be a powerful way to encode the image.

A. Overview of static HOG feature extraction

We encode the image in five steps: normalize gamma & color, compute gradients, weighted vote into spatial & orientation cells, contrast normalize over overlapping spatial blocks, collect HOG’s over detection window. In the first two steps, we preprocess the image to get the gradient and orientation images which are needed to compute the HOG. The following three steps are the core parts of HOG. We will talk about those five steps in details in the following subsections.

B. Gamma Normalization and Gradients computation

Gamma normalization is a image enhancement technique, which can improve the contrast of images. It does a transformation of the gray level of images. The transformation function is below:
\[ y(n_1, n_2) = x^\gamma(n_1, n_2) \]

Where, \( x(n_1, n_2) \) is the input image, and \( y(n_1, n_2) \) is the output image. Here, we set the parameter \( \gamma = 0.5 \) that can gives us a good detection performance. The following image shows the transform result of gamma normalization. We can find that the image on the right gives us a better contrast that is very useful for the further processing.

\[ \]

HOG descriptor is based on oriented gradient. The gradient is used to describe the difference of between pixels. The gradient can be computed by the following equations:

\[ y_i(n_1, n_2) = h_i(n_1, n_2) \ast x(n_1, n_2), i = 1, 2 \]

Here, \( h_i(n_1, n_2) \) is the derivative mask, the usual choice of it is

\[ h_1(n_1, n_2) = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}; h_2(n_1, n_2) = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \]

\( h_1(n_1, n_2) \) is used to compute the horizontal gradient, and \( h_2(n_1, n_2) \) is used to compute the vertical gradient. Thus, the overall gradient and orientation image can be get in the following way

\[ y_M(n_1, n_2) = \sqrt{y_1^2 + y_2^2}; y_\theta(n_1, n_2) = \tan^{-1}(y_2/y_1) \]

Figure above, shows the gradient of the image, from which we can find that the gradient image can highlight the outline of the human body.

C. HOG Descriptor

We can compute the HOG descriptor with the follow algorithm:
1) Divide the image window into a dense uniformly sampled grid of points, and for each point.
2) Divide the $\alpha\beta \times \alpha\beta$ square pixel image region centred on the point into cells
3) Apply a Gaussian window with sigma = $0.5 \times \alpha\beta$ to image gradients in the block.
4) Create a $\alpha \times \beta \times \beta$ spatial and orientation histogram.
5) For each pixel in the block, use tri-linear interpolation to vote into the histogram using
   gradient magnitude.
6) Apply $L2$-$Hys$ normalization independently to each block.
7) Collect the HOGs of all blocks lying in the image window into one big descriptor vector.

III. Classifier and Learning algorithm

After we get our feature space, we can then begin to train the classifier. The technique we
used to train our classifier is support vector machine (SVM).

A. Support Vector Machine (SVM)

The technique we used to train our classifier is support vector machine (SVM). Support vector
machines are supervised learning models with associated learning algorithms that analyze data and recognize
patterns, used for classification and regression analysis. The basic definition of a SVM classifier is

$$f(\mathbf{x}) \leftarrow \text{sgn} \left( \sum_{\mathbf{s}_i \in \text{support vectors}} \alpha_i y_i \mathbf{s}_i \cdot \mathbf{x} + b \right),$$

where $\text{sgn} = \begin{cases} -1 & \text{when } x < 0, \\ 0 & \text{when } x = 0, \\ 1 & \text{when } x > 0. \end{cases}$

Here, $\alpha_i$ is the Lagrange multiplier, $y_i$ is the labels of the support vectors, $\mathbf{s}_i$ is the support
vectors, and $\mathbf{x}$ is our input feature vector. The figure above is a sample of the SVM classifier. In
human detection, the feature space is non-separable, so we choose a soft margin SVM classifier,
where we set the penalty parameter $C = 0.01$. The kernel we choose is linear kernel, because it
can avoid costing too much run time.

B. Window based Classifier Learning Algorithm
The following figure shows our learning algorithm. First we choose our images from the training data set. Then we label those images and sample them window by window (128×64). After that we encode those windows to get the training feature space. From the training feature space we can use the SVM to train a preliminary classifier. Once we got the preliminary we need to use it to detect some false positive windows from our negative images (without people), and add those false positive windows to our feature space. Then, use the augmented feature space to train our final classifier. The reason we need those false positive windows is because it can decrease our miss rate. Although it can also increase the false positive rate but this can be reduced in the detection part.

IV. MATLAB Implementation And Toolbox Design

In this section, we will first introduce several functions and matlab script that we designed to accomplish the whole human detection algorithm. And then introduce the design of the toolbox, then function of the toolbox and how to use the toolbox. The MATLAB based human detection is support by 6 matlab files and contains about 800 matlab command lines. All the codes are my original work.
A. File Based Human Detection Implementation

There are five matlab files to help implement the human detection work.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encode.m</td>
<td>Help you to build the feature space. Choose a folder that contains your data set images to start encoding. The feature results will be stored in 'Train.mat' file.</td>
</tr>
<tr>
<td>hog_descriptor.m</td>
<td>Encode a single image with HOG. It can also gives you the a visualized HOG feature image matrix.</td>
</tr>
<tr>
<td>imload.m</td>
<td>A function that accomplish the actual encoding work</td>
</tr>
<tr>
<td>SVM_Classify.m</td>
<td>Contains two parts. First part: load feature space from 'Test.mat', 'MIT.mat', then load Classifier from ‘Train.mat’. Classify features and calculate the accuracy of the class results. Second part: choose a image file from the test set ‘~/Data/INRIAPerson/test_64x128_H96’ or ‘pedestrians128x64’ the SVM classifier will return the class result</td>
</tr>
<tr>
<td>SVM_Train.m</td>
<td>Load the feature space from 'Train.mat' frist. Then use the function 'svmtrain' to get the SVM classifier. The classifier will finally be stored int 'Train.mat'</td>
</tr>
</tbody>
</table>

Here, hog_descriptor.m and imload.m are the function that called by other script. So, to realize the human detection on MATLAB, we need to run the rest three function in the following way: first run 'Encode.m' it can help you build your feature space. Then run 'SVM_Train.m' it can help you train your classifier. Finally run 'SVM_Classify' it can help you classify your images.

B. Graphic User Interface (GUI) Based Human Detection

To give a better demonstration of my work, and also give the beginner some intuitive perspective of human detection, I also build a toolbox.
1. Interface and functions

The table below shows the functions of the toolbox.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Panel</th>
<th>Button/Listbox/Text/Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>’Encoding Options’, help user build and store the feature space</td>
<td>’Data Source’, by click this button user can import a folder of images into MATLAB.</td>
</tr>
<tr>
<td></td>
<td>’Training Information’, provide some training information</td>
<td>’Positive’ and ’Negative’ radio button, set the label of the images that user imported.</td>
</tr>
<tr>
<td></td>
<td>’SVM Weight’</td>
<td>’Encode’, start to encode the imported images into feature space.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’Default’, read the default feature spaces into MATLAB</td>
</tr>
<tr>
<td>Test</td>
<td>’Test Set’</td>
<td>’Positive SVM Weight’, Each “pixel” shows the maximum positive SVM weight in the block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>centred on the pixel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’Negative SVM Weight’, Each “pixel” shows the maximum negative SVM weight in the block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>centred on the pixel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’Data Source’, Choose a folder that contains your test images, and load them into MATLAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Listbox, Show the current test images that loaded into MATLAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Figure, the current image that user chosen</td>
</tr>
<tr>
<td></td>
<td>’Result’</td>
<td>Give the detection result, ’Y’ stand for yes, ’N’ stand for no</td>
</tr>
<tr>
<td></td>
<td>’Image Information’</td>
<td>’Gamma Normalization’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’Gradient’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’HOG features’</td>
</tr>
</tbody>
</table>

So, the toolbox should be used in the following ways: first, select the ‘Training’ mode, click ‘Data Source’ in the ‘Encoding Option’ panel and choose folder that contains positive images.
(make sure the ‘Positive’ radio button is selected), then click the ‘Encode’ button to get positive feature vectors. After that also add some negative positive images into MATLAB by click ‘Data Source’ button. Click the ‘Encode’ button as well to get negative feature vectors. User can also skip the above steps by click ‘Default’ button to load the default feature space into MATLAB. Next, click the ‘Train’ button to get the classifier. When finish the training process, change the mode to ‘Test’, click ‘Data Source’ button to add a test data set into MATLAB, then by click images shows in the listbox, user can see the detect results on the right.

V. Experiment and Statistic Result

A. Data Sets

We evaluate our human detection system with two different static human data sets. The first one is the MIT pedestrian data set, it contains 924 pedestrian images, but all of them are only front back views with a relatively limited range of poses. The second one is the INRIA Person data set. In this data set, people appear in any orientation and against a wide variety of background image including crowds.

From these two data sets we choose 2416 images (contain people) with a size of 96×160 pixels and 1218 negative images (without people) with unfixed size as our training data set. And we also choose 1126 positive images (contain people) with as size of 70×134 and 453 negative images (without people) with unfixed size. The following images are some samples from our data sets.

Samples from MIT Dataset

Samples from INRIA Dataset

B. Experiment and Statistic Result
In our experiment, we have three test data sets: MIT data set, it contains 924 positive windows. INRIA positive test samples, it contains 1126 positive windows. INRIA negative test samples, it contains 4530 negative windows. The default parameter we set in our experiment is

<table>
<thead>
<tr>
<th>Properties</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Image</td>
<td>RGB color space</td>
</tr>
<tr>
<td>Derivative Mask</td>
<td>[-1,0,1]</td>
</tr>
<tr>
<td>Orientation Bins</td>
<td>9 bins in 0°-180°</td>
</tr>
<tr>
<td>Block Size</td>
<td>2x2 cells of 8x8 pixels</td>
</tr>
<tr>
<td>Block Overlap</td>
<td>8 pixels</td>
</tr>
<tr>
<td>Block Normalization</td>
<td>L2-Hys</td>
</tr>
<tr>
<td>Detection Window</td>
<td>64x128</td>
</tr>
<tr>
<td>Classifier</td>
<td>SVM, linear kernel, soft-margin with C=0.01</td>
</tr>
</tbody>
</table>

Then, test our human detection system with the test samples we introduced above. And get the results below

<table>
<thead>
<tr>
<th>MIT Data Set</th>
<th>INRIA Positive Test Samples</th>
<th>INRIA Negative Test Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.59% (911/924)</td>
<td>98.22% (1106/1126)</td>
<td>98.01% (4440/4530)</td>
</tr>
</tbody>
</table>

From the table, above we can know our human detection has a relatively good performance.

VI. Conclusion

In this report, we first studied the basic theory of HOG, and its implementation algorithm. Then, studied some basic knowledge of SVM and its implementation algorithm in human detection. Third, we implemented our human detection system on MATLAB and build a GUI toolbox. Finally, we tested the performance of our system and find that it has a very nice performance.
VII. References


