



Histogram of Oriented Gradients and Object Detection

by **Adrian Rosebrock** on November 10, 2014 in **Machine Learning, Tutorials**

15

34



If you've been paying attention to my Twitter account lately, you've probably noticed [one](#) or [two](#) teasers of what I've been working on — a Python framework/package to rapidly construct object detectors using Histogram of Oriented Gradients and Linear Support Vector Machines.

Honestly, I really can't stand using the Haar cascade classifiers provided by OpenCV (i.e. the Viola-Jones detectors) — and hence why I'm working on my own suite of classifiers. While cascade methods are extremely fast, they leave much to be desired. If you've ever used OpenCV to detect faces you'll know exactly what I'm talking about.

Free 21-day crash course on computer vision & image search engines

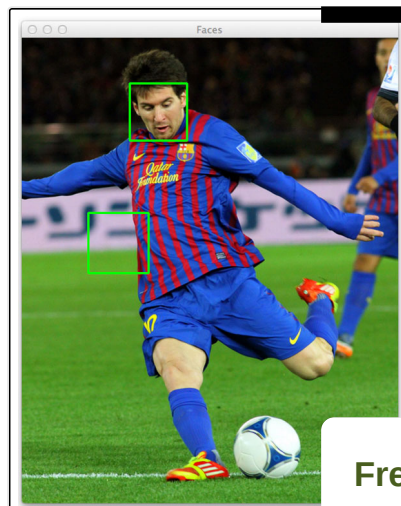


Figure 1: Example of falsely detect in an image. This is a common problem using `cv2.detectMultiScale`

In order to detect faces/humans/objects/whatever in OpenCV (and removing tuning the `cv2.detectMultiScale` parameters. And again, there is no going from image-to-image. This makes batch-processing large datasets for faces concerned with either (1) falsely detecting faces or (2) missing faces entirely on an image basis.

There is also the problem that the Viola-Jones detectors *are nearing 15* Cabernet Sauvignon I might be pretty stoked right now. But the field has the Viola-Jones detectors were state-of-the-art and they were certainly a advances we have in object detection today.

Now, the Viola-Jones detector isn't our only choice for object detection. We have object detection using keypoints, local invariant descriptors, and bag-of-visual-words models. We have Histogram of Oriented Gradients. We have deformable parts models. Exemplar models. And we are now utilizing Deep Learning with pyramids to recognize objects at different scales!

All that said, even though the Histogram of Oriented Gradients descriptor for object recognition is nearly a decade old, it is still heavily used today — and with fantastic results. The Histogram of Oriented Gradients method suggested by Dalal and Triggs in their seminal 2005 paper, *Histogram of Oriented Gradients for Human Detection* demonstrated that the Histogram of Oriented Gradients (HOG) image descriptor and a Linear Support Vector Machine (SVM) could be used to train highly accurate object classifiers — or in their particular study, human detectors.

Histogram of Oriented Gradients and Object Detection

I'm not going to review the entire detailed process of training an object detector using Histogram of Oriented Gradients (yet), simply because each step can be fairly detailed. But I wanted to take a minute and detail the general algorithm for training an object detector using Histogram of Oriented Gradients. It goes a little something like this:

Step 1:

Sample P positive samples from your training data of the object(s) you want to detect and extract HOG descriptors from these samples.

Step 2:

Sample N negative samples from a *negative training set* that **does not contain** any of the objects you want to detect and extract HOG descriptors from these samples as well. In practice $N \gg P$.

Step 3:

Train a Linear Support Vector Machine on your positive and negative samples.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!

Free 21-day crash course on computer vision & image search engines

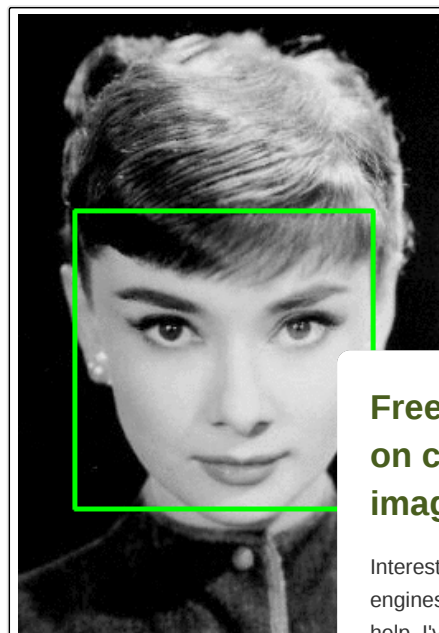
Step 4:

Figure 2: Example of the sliding window approach, where we slide a window across the image from left-to-right and top-to-bottom. *Note: Only a single window is shown. In practice this window would be applied to multiple scales of the image.*

Apply hard-negative mining. For each image and each possible scale of the object, apply the sliding window technique and slide your window across the image. At each window, apply your classifier. If your classifier (incorrectly) classifies a given window as a positive (false-positives), record the feature vector associated with the false-positive classification. **This approach is called *hard-negative mining*.**

Step 5:

Take the false-positive samples found during the hard-negative mining stage, sort them by their confidence (i.e. probability) and re-train your classifier using these hard-negative samples. *(Note: You can iteratively apply steps 4-5, but in practice one stage of hard-negative mining usually [not not always] tends to be enough. The gains in accuracy on subsequent runs of hard-negative mining tend to be minimal.)*

Step 6:

Your classifier is now trained and can be applied to your test dataset. Again, just like in Step 4, for each image in your test set, and for each scale of the image, apply the sliding window technique. At each window extract HOG descriptors and apply your classifier. If your classifier detects an object with sufficiently large probability, record the bounding box of the window. After you have finished scanning the image, apply non-maximum suppression to remove redundant and overlapping bounding boxes.

These are the bare minimum steps required, but by using this 6-step process you can train and build object detection classifiers of your own! Extensions to this approach include a [deformable parts model](#) and [Exemplar SVMs](#), where you train a classifier for *each positive instance* rather than a *collection of them*.

However, if you've ever worked with object detection in images you've likely ran into the problem of *detecting multiple bounding boxes around the object you want to detect in the image*.

Here's an example of this overlapping bounding box problem:

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!

Free 21-day crash course on computer vision & image search engines

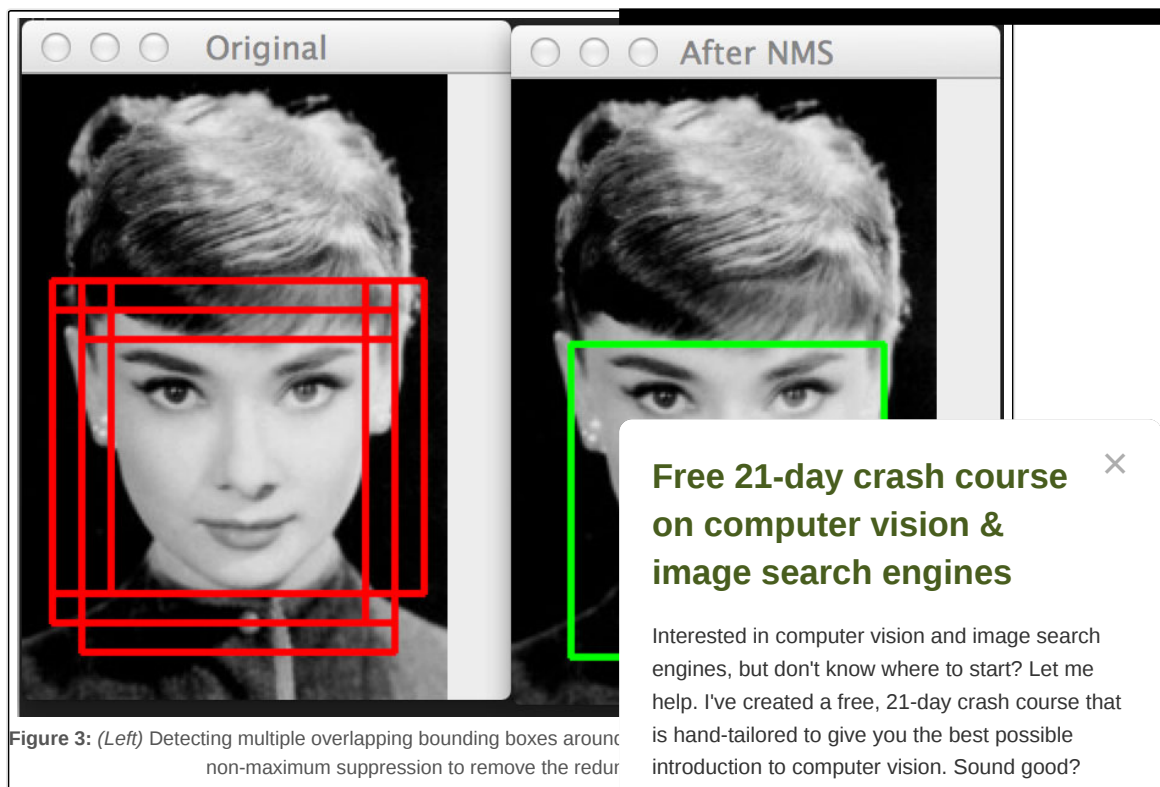


Figure 3: (Left) Detecting multiple overlapping bounding boxes around a face. (Right) The result after non-maximum suppression to remove the redundant boxes.

Notice on the *left* we have 6 overlapping bounding boxes that have correctly detected the face. However, these 6 bounding boxes all refer to the same face — we need a method to suppress the redundant boxes, keeping only the largest one, as seen on the *right*.

This is a common problem, no matter if you are using the Viola-Jones based object detector or a deep learning based object detector.

There are multiple ways to remedy this problem. Triggs et al. suggests to use the [mean-shift algorithm](#) to detect multiple modes in the bounding box space by utilizing the (x, y) coordinates of the bounding box as well as the logarithm of the current scale of the image.

I've personally tried this method and wasn't satisfied with the results. Instead, you're much better off relying on a *strong classifier* with *higher accuracy* (meaning there are very few false positives) and then applying non-maximum suppression to the bounding boxes.

I spent some time looking for a good non-maximum suppression (sometimes called non-maxima suppression) implementation in Python. When I couldn't find one, I chatted with my friend [Dr. Tomasz Malisiewicz](#), who has spent his entire career working with object detector algorithms and the HOG descriptor. There is literally *no one* that I know who has more experience in this area than Tomasz. And if you've ever read any of his papers, you'll know why. His work is fantastic.

Anyway, after chatting with him, he pointed me to two MATLAB implementations. [The first](#) is based on the work by [Felzenszwalb et al.](#) and their deformable parts model.

The [second method](#) is implemented by Tomasz himself for his [Exemplar SVM](#) project which he used for his dissertation and his ICCV 2011 paper, [Ensemble of Exemplar-SVMs for Object Detection and Beyond](#). It's important to note that Tomasz's method is *over 100x faster* than the Felzenszwalb et al. method. And when you're executing your non-maximum suppression function millions of times, that 100x speedup really matters.

I've implemented both the Felzenszwalb et al. and Tomasz et al. methods, porting them from MATLAB to Python. Next week we'll start with the Felzenszwalb method, then the following week I'll cover Tomasz's method. While Tomasz's method is substantially faster, I think it's important to see both implementations so we can understand exactly *why* his method obtains such drastic speedups.

Be sure to stick around and check out these posts! These are absolutely critical steps to building object detectors of your own!

Summary

Free 21-day crash course on computer vision & image search engines

In this blog post we had a little bit of a history lesson regarding object detectors. We also had a sneak peek into a Python framework that I am working on for object detection in images.

From there we had a quick review of how the Histogram of Oriented Gradients method is used in conjunction with a Linear SVM to train a robust object detector.

However, no matter what method of object detection you use, you will likely end up with multiple bounding boxes surrounding the object you want to detect. In order to remove these redundant boxes you'll need to apply Non-Maximum Suppression.

Over the next two weeks I'll show you two implementations of Non-Maximum Suppression that you can use in your own object detection projects.

Be sure to enter your email address in the form below to receive an announcement. Non-Maximum Suppression is absolutely *critical* to obtaining an accurate and robust object detector. You don't want to miss these posts!

Resource Guide (it's totally free).



Enter your email address below to get my **free PDF**. Uncover **exclusive techniques** that I don't want to share on my search engines of your own!

DOWNLOAD THE GUIDE!

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!

🔖 **histogram of oriented gradients, hog, linear svm, nms, non maxima suppression, non maximum suppression, svm**

< [How to Display a Matplotlib RGB Image](#)

[Non-Maximum Suppression for Object Detection in Python](#) >

152 Responses to *Histogram of Oriented Gradients and Object Detection*



SHI Xudong November 10, 2014 at 11:20 pm #

REPLY ↩

Hi Adrian,

Thank you for your great post.

I am currently working on object detection too. My goal is to make the detection faster, because object detectors are usually slow.

In your post, the object detection framework you are using is a 1 model, N image scales. However, there are alternatives, such as N models, 1 image scale; 1 model, N/K ($K > 1$) image scales (FPDW approach); N/K models ($K > 1$), 1 image scale.

I am thinking about creating a unified framework, which can include all these frameworks, but have no idea about the implementation now. Could you please make some comments:

- Is it worth to try?
- how will you do if you are doing it?

References:

1. Benenson, Rodrigo, et al. "Pedestrian detection at 100 frames per second." Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.
2. Dollár, Piotr, Serge Belongie, and Pietro Perona. "The Fastest Pedestrian

Free 21-day crash course on computer vision & image search engines



Adrian Rosebrock November 11, 2014 at 7:07 am #

REPLY ↩

Hi Shi! You are absolutely right, object detectors can be painfully slow, especially when you using the sliding window technique. And when expensive features such as HOG need to be computed, it can really kill performance.

And you are correct, I am utilizing the N image scales model for this framework. However, I have plans to create something similar to the FPDW approach by Dollar et al.

Right now performance using the N image scales model is actually pretty good for my framework, but not great. The reason is because I have distributed the image pyramid to all available cores of the system — this is an obvious solution where making the HOG sliding window computation run in parallel can dramatically speedup the code.

However, doing something like FPDW will further increase the speed (I

Send me an email and I would be happy to chat more about implementing a ton of great information in here.

**Free 21-day crash course
on computer vision &
image search engines**



Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Eyy November 10, 2014 at 11:28 pm #

Really good illustration. Thanks! waiting for your following posts



Varun Kumar November 11, 2014 at 5:15 am #

Thanks Adrian for this and other wonderful posts. I have been following you say you are doing a wonderful job. Keep it up!



Adrian Rosebrock November 11, 2014 at 7:08 am #

REPLY ↩

Thanks, I really appreciate it! 😊



P Derep November 11, 2014 at 8:36 am #

REPLY ↩

La Fin Du Monde... that's quite a good beer! I'm impressed! 😊 Kudos for the interesting article, BTW.



Wajih Ullah Baig November 11, 2014 at 12:24 pm #

REPLY ↩

Good post, waiting for some nifty code to mess with 😊
Your blog is now my standard technology for teaching!



Adrian Rosebrock November 11, 2014 at 2:35 pm #

REPLY ↩

Thank you for such an awesome compliment! 😊



Wajih Ullah Baig November 13, 2014 at 2:53 am #

REPLY ↩

Thank you, not me for such an awesome blog!!

**Free 21-day crash course on computer
vision & image search engines**

**Niv** November 17, 2014 at 9:29 am #

REPLY ↩

Nice post, thanks.

In phase #5, the false positives are taken along with their probabilities and then sorted by their probabilities in order to further retrain the classifier. I have two questions about which I would appreciate to get a clarification:

1. why do you need their probabilities in that retraining phase? linear SVM doesn't require their probabilities but rather merely their taggings (negatives in this case).
2. Why do you need them to be sorted? linear SVM doesn't require any sorting of the training samples.

**Adrian Rosebrock** November 17, 2014 at 10:49 am #

REPLY ↩

You would want to keep track of the probabilities for two reasons: physical memory to store the positive samples, negative samples, and case you are limited by RAM, you would want to sort the samples by the classifier performed "worst" on, discarding the rest. Secondly, you may

**Free 21-day crash course
on computer vision &
image search engines**

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!

**asaaki** December 18, 2014 at 3:26 am #

Will Steps 4 and 5 work for a multiclass cascade classifier as well example you're dealing with LSVMS.

**Adrian Rosebrock** December 18, 2014 at 7:45 am #

Potentially, but there are other approaches that are more robust.

**Karthik** January 6, 2015 at 1:33 am #

REPLY ↩

According to the paper published by Dalal and Triggs, they suggest gamma correction is not necessary. I have the doubt about whether correcting gamma is a good option to go for or not. If Gamma correction is necessary what is the gamma value I have to take for better performance. Awaiting for reply!!! thanks in advance!!

**Adrian Rosebrock** January 6, 2015 at 7:38 am #

REPLY ↩

That is correct, gamma correction is not necessary. Normalization however is quite often helpful. You can normalize by either taking the log or the square-root of the image channel before applying the HOG descriptor (normally the square-root is used). Another method to make HOG more robust is to compute the gradient magnitude over all channels of the RGB or L*a*b* image and take the maximum gradient response across all channels before forming the histogram.

**Karthi** January 6, 2015 at 2:40 pm #

REPLY ↩

Thank you for your reply!! But why can't we consider converting to grayscale and processing would be the better option?

**Adrian Rosebrock** January 6, 2015 at 2:52 pm #

REPLY ↩

You can certainly convert to grayscale and compute the maximum response over the individual RGB or L*a*b* channel

**Free 21-day crash course on computer
vision & image search engines**

mileage may vary depending on your dataset.



Karthik January 6, 2015 at 3:45 pm #

And we are actually stretching the contrast while we are gamma correcting, so which gamma value you think will be providing the higher performance? gamma of 2?



Adrian Rosebrock January 6, 2015 at 4:02 pm #

Please see my previous comment. I would not recommend gamma correction or square-root normalization should suffice.



Sarah April 10, 2015 at 11:34 pm #

In opencv the gamma correction can only be set or not set. I believe gamma correction makes images which can be taken at night. I believe gamma correction makes



Adrian Rosebrock April 11, 2015 at 8:55 am #

Hi Sarah, thanks for the comment. However, log-base 10 normalization is better for SVM performance. The square-root (in most cases) or simple

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Karthik January 6, 2015 at 10:22 pm #

REPLY ↩

Ya, I got you. And I would like to thank for your valuable blog !!! really helpfull!!



Morné January 26, 2015 at 2:43 am #

REPLY ↩

Hi Adrian, this is great stuff. Do you have any tutorials on implementing HOG descriptors. Having that is really helpful for learning the essentials. Thanks.



Adrian Rosebrock January 26, 2015 at 6:43 am #

REPLY ↩

Hi Morné, check out the scikit-image implementation of [HOG](#). That is a great starting point!



James January 28, 2015 at 3:08 am #

REPLY ↩

Hi Adrian,

Great post! Are you still working on this or is it already completed?



Adrian Rosebrock January 28, 2015 at 6:28 am #

REPLY ↩

Great question! The framework is 100% complete at this point. I am working on. Check out the [Kickstarter](#) for more information!

Free 21-day crash course on computer vision & image search engines



Jeff February 12, 2015 at 11:18 am #

REPLY ↩

I have a suggestion for speeding up the search, and would like your opinion.

What if you applied the KPM, or Boyer-Moore to the search ? Meaning, convert the histogram 'coefficients' to a string representation, and do the same with the image being searched. Then, look at the 'suffixes' first.

I've used the Boyer-Moore, because it's about 10x faster for string searching. In this application, it would be like looking at the right edge of a rect first, and then skipping over pixels when there isn't a match.



Jeck March 14, 2015 at 1:39 pm #

Hi Adrian, awesome tutorial btw. Im quite a beginner in openCV. I am able to detect license plates and read them. However, I don't really know where to start with. So far, I was able to create the document scanner in your tutorial. Thanks in advance.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Adrian Rosebrock March 14, 2015 at 4:17 pm #

Hi Jeck, great question. I'll be covering license plate detection [course](#). License plate detection and recognition is not an easy task, but it can be broken down into easy to digest components, similar to other tutorials on the PyImageSearch.com website.



Miriam March 17, 2015 at 6:10 am #

Hi , i'am working at detection face and i have a problem. the algorithm didn't detect the faces in in different position. what's the problem ? Thank you



Adrian Rosebrock March 17, 2015 at 6:43 am #

REPLY ↩

Hi Miriam. Unfortunately, there could be many, many reasons why faces are not detected and your question is a bit ambiguous. Here are just some example questions for you to consider: Are you using HOG + Linear SVM? Have you chosen the optimal HOG parameters for your descriptor? What about your Linear SVM? Are you using a soft-classifier? Do your training examples sufficiently represent the faces that you want to detect in images? Are you performing hard negative mining to increase accuracy? Are you using image pyramids so you can detect faces at multi-scales? Or using a sliding window so detect faces in different positions of the image? Be sure to consider all avenues.



Miriam April 7, 2015 at 7:26 am #

REPLY ↩

Ok

Thank you Adrian.



Milos March 24, 2015 at 4:33 am #

REPLY ↩

Hey Adrian, i really want to thank you for outstanding lessons you taught me about computer vision. I was wondering, can we expect something about image stitching. Something like that. All best to you sir! 😊

Free 21-day crash course on computer vision & image search engines



Adrian Rosebrock March 24, 2015 at 6:36 am #

REPLY ↩

Hi Milos, thank you for such a kind comment! And we will absolutely be doing image stitching and panoramas within the next few months. I can't set an actual date yet since I have a series of posts coming out soon, but I'll be sure to let you know when the image stitching post goes live.



fansari March 25, 2015 at 5:33 am #

REPLY ↩

if my dataset is on colored images and color is a demarcating feature in my classification then should I use grayscale image or should I find gradient resultant on all channels of RGB. Will it help



Adrian Rosebrock March 25, 2015 at 6:48 am #

Dalal and Triggs actually recommend converting from the RGB to grayscale by computing the gradient magnitude for each of the L*, a*, and b* channels and then averaging the magnitude across all channels. In many cases this can improve performance.



Yiwen Wan April 10, 2015 at 11:29 pm #

Thanks for your great work. I have a question about training on images of different size. Do you try to resize them to a predefined size? I concern about orientation depending on how it is resized. Looking forwards to your answer.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Adrian Rosebrock April 11, 2015 at 8:56 am #

REPLY ↩

Indeed, each object of each class should be resized to a pre-defined size to ensure the resulting HOG feature vectors are the same dimensionality.



Yiwen Wan April 14, 2015 at 7:50 pm #

Thanks. I am aware of that. Just want to make sure whether it will give distortion or not. If it does bring distortion to the gradient orientations. Maybe compensation is needed? Or maybe the distortion is too little to consider.



Adrian Rosebrock April 15, 2015 at 9:40 am #

It will absolutely bring in some distortions since we are ignoring the aspect ratio of the image during the resizing, but that's a standard side effect that we accept so we can get features of the same dimensionality.



Méabh Loughman April 17, 2015 at 5:01 am #

REPLY ↩

Hey Adrian,

Just wanted to say thank you. I am a student studying computer vision for the first time as part of a taught masters and this blog really helped me so much – you have a knack for explaining things intuitively 😊

Thanks again,
Méabh

Free 21-day crash course on computer vision & image search engines



Adrian Rosebrock April 17, 2015 at 7:03 am #

REPLY ↩

Thanks so much Méabh 😊 I'm happy to help. Best of luck with your masters program!



Rish April 18, 2015 at 9:40 am #

REPLY ↩

First I am just thank you for your wonderful and super easy to understand tutorials and perhaps the best available.

I have one question on Object Recognition using sliding window and SVM. detection I get multiple windows where I need to apply Non-Maximum Suppression. However, the linear SVM output is a hard decision of +1 for objects and -1 for background. NMS as all weights (considering the prediction response) are same.

I read a paper from Platt, 1999 to convert the prediction response to probabilities. Is there any simpler or better way to achieve this perhaps.

Thank you for your time.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Adrian Rosebrock April 18, 2015 at 1:33 pm #

Hey Rish, thanks for the comment. Does your SVM library not do. I don't do much work in C++ (literally none), but I know the scikit-learn class labels.

Also, you can still do NMS without probabilities. Instead of using the probabilities, I do in this post on [fast non-maximum suppression](#). It's obviously better still work for you.



Rish April 19, 2015 at 11:23 pm #

REPLY ↩

Hi Adrian, thank you for your prompt reply. I checked the LibSVM and it does return probabilities while SVM in OpenCV C++ does not. Hence I changed to using LibSVM.

Regarding the NMS, as your tutorial is implemented using bottom right-hand corner, I am little confused how to use Bounding Box probabilities instead. I guess maybe like this but if I am wrong than could please guide me on the right path.

→ Instead of computing the overlap should I directly use the probabilities of classifier output?



Adrian Rosebrock April 20, 2015 at 6:54 am #

REPLY ↩

You'll need to modify my NMS code to accept a list of probabilities that corresponds to the bounding boxes. Then you'll need to sort those probabilities (Line 27) rather than the bottom-right corner.



Rish April 23, 2015 at 12:42 pm #

Hi Adrian. Thank you very much I managed to get it working with LibSVM. However, I noticed that getting the probabilities from SVM is computationally more expensive compared to the label prediction (hard decision). I wonder if this happens or did I just got something wrong? Mathematically, I think estimating probability needs more operations. However, my question here is that if I don't use probabilities but just use label prediction i.e (+1/-1) than use your NMS to get the final output. Are Probabilities are important or does it improves results? (

Free 21-day crash course on computer vision & image search engines

Secondly, about the HOG descriptor. Say for example I trained a classifier to detect standing bottle (say sliding window of 64×128) = 3780 dimension feature vector. During detection, I will use this window size to detect the bottle. However, my question is when the bottle is resting on its side (now say the dimension is 128×64) but our detector uses the window of 64×128 , how do I cater for this issue? I read paper on Part based model but I think PBM is computationally expensive. Perhaps I thought to divide the training image into 4 parts (say 16×32) and train this. Do you have any suggestion on this?



Adrian Rosebrock April 23, 2015 at 1:08 pm #

Hey Rish, you're absolutely correct — using probabilities does increase the computational cost. However, if you have the probabilities you'll likely get better results. It depends on your application and the level of tolerance you have.

As for your second question, HOG based methods, including "structure" to the image. For example, if you trained your classifier to detect wheels, handlebars, a seat, etc. But if this bike were rotated, the HOG descriptors are structural descriptors and hence in your case the object is rotated.

If you're really concerned about finding all bottles in the image, you could rotate each input image and rotate it 0, 90, 180, and 270 degrees. This definitely comes at a considerable computational cost, but it works.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Rish May 4, 2015 at 12:54 pm #

Hi Adrian. Sorry I was busy with my Finals so was unable to respond earlier. I want to change your NMS from Bottom Right Corner to probability sorting as you mentioned.

I think I am a bit lost, I removed all the bounding box params from your NMS and use the Probability instead. I am confused like in line 51 where you do the overlapping computation. How can I change this to probability instead? Do I need to do something like Gradient Descent? Sorry I am just confused, please help me out. Thanks



Adrian Rosebrock May 4, 2015 at 1:15 pm #

No need for gradient descent or anything fancy. When you make a call to your NMS function you'll need to pass in three variables: the bounding boxes, the overlap threshold, and a probability of detection that corresponds to the bounding box. Then you need to sort (from highest probability to lowest probability) the bounding boxes based on their probability. From there, everything will be the same. Like I said, the only code that you'll need to change is the code that sorts on probability rather than the bottom-right coordinate.



Siva Krishna January 30, 2016 at 5:08 am #

Hi Adrian,

I have a few doubts regarding NMS algorithm using probabilities

1. What does probability mean? Is it the detection score by SVM (i.e. $Wx+b$ value), isn't it?
2. You said that we have to sort in decreasing order of probabilities. But I think we have to sort in increasing order, since we are picking from last.

Thank you for your wonderful posts.



Adrian Rosebrock January 30, 2016 at 11:57 am #

Free 21-day crash course on computer vision & image search engines

The probabilities actually come from a [modified logistic regression](#). As for the sorting of the probabilities, that depends entirely on the actual implementation of the NMS algorithm.



Russell May 2, 2015 at 3:24 pm #

REPLY ↩

I am new to sklearn. I see many of you have advanced experience in this area. I am therefore asking for your assistance so that I can catch up 😊

I have a number of images of sedans, SUV's and Trucks that pass outside my house. I want to train a SVM classifier to detect whether a vehicle is a one of these three. I have cropped all of the images and successfully created Histogram Oriented Gradient images of these vehicles. My question is Do I have to write code to reshape the image and format it to get the data into a format useable by scikit-learn or is there code already written to do tl



Adrian Rosebrock May 2, 2015 at 3:36 pm #

Visualizing a Histogram of Oriented Gradients image versus a feature vector are two completely different things. While you can visualize training a classifier — it simply allows you to visually inspect the gradient need to extract the HOG descriptor using something like [scikit-image](#) as mentioned in an email to you, I'll be covering all this inside the [PyImage](#)

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Russell May 2, 2015 at 3:59 pm #

Hi Adrian,

Yes, I understand that the HOG image is not useable for integrating reshaped into a matrix of row vectors so that we have number of samples vs number of features. Are you telling me that this is done in scikit-image? Thanks for your help.



Adrian Rosebrock May 2, 2015 at 4:07 pm #

REPLY ↩

Yes, take a look at the [documentation/example](#). If you take a look at the Handwriting Recognition chapter of Case Studies, you'll learn how to extract the HOG feature vector. Secondly, the HOG image does not need to be reshaped — the HOG image is essentially useless for anything but visualization. You'll need the HOG feature vector (which scikit-image provides you with) to create your classifier.



Jay July 29, 2016 at 7:29 am #

Hi Adrian, it occurred to me that OpenCV's hog may be a better choice. I looked into scikit-image's source code of hog and found it does not implement weighted vote for histogram orientation binning. It only does simple binning based on magnitude value falling into which bin, which is not quite an accurate way for computing hog.

Actually, I found in my experiments that object detection based on OpenCV's implementation is more accurate than with the one from scikit-image, and runs faster.



Adrian Rosebrock July 29, 2016 at 8:24 am #

Thanks for sharing your experience Jay! I've been using a personal modification of the scikit-image HOG that I've been meaning to create a pull request for. It handles various types of input transformations (square-root, log, variance) along with multi-channel image across the channels (this tends to work better in practice).

Free 21-day crash course on computer vision & image search engines



TimothyUntan May 5, 2015 at 9:47 pm #

REPLY ↩

Dang, you're awesome man... That one helps a lot for my final year project... Cheers...



Adrian Rosebrock May 6, 2015 at 7:06 am #

REPLY ↩

I'm happy it helped Timothy! And congrats on your (impending) graduation!



pal May 7, 2015 at 5:30 am #

great post!

I just want to know that how many training images are required to make a g



Adrian Rosebrock May 7, 2015 at 7:16 am #

The answer is, it depends! Some problems are quite simple a problems are very, very complex and require many training examples. examples of what you want to detect, followed by 500-1000 negative training and select another 500-1000 false positive examples. This approach but it will give you a baseline to further tune your parameters. Training your first attempt to give you the best performance.

Free 21-day crash course on computer vision & image search engines



Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Pal May 8, 2015 at 12:06 am #

REPLY ↩

Thanks a lot! 😊



abbas May 22, 2015 at 12:48 am #

REPLY ↩

Hi Adrian,

I am a student of BS computer science i have started working on human detection in image using HOG descriptor. I have a problem in testing phase step 4 and 5. Are you telling me what i do i use a matlab as a tool.



Adrian Rosebrock May 22, 2015 at 5:09 am #

REPLY ↩

Hi Abbas, I honestly have not used MATLAB in a very, very long time so I am probably not the right person for that question.



ALI June 25, 2015 at 4:59 am #

REPLY ↩

Thank you for your work , May I sak you to direct me to some implementation of hard-negative mining. Actually I didnt get the point of how to reuse the false negative data ! . Should I use multi instance Svm ; I mean like clustering the false positive .

In case I get false positive by my trained classifier on negative train data, s them with negative samples ? I am a little bit confused on what the proced

Free 21-day crash course on computer vision & image search engines

you help on that !

Cheers ,



Adrian Rosebrock June 25, 2015 at 6:19 am #

REPLY ↩

So basically, hard negative mining is all about finding the patches of images (and their associated HOG feature vectors) that the Linear SVM gets totally and completely wrong. For example, let's pretend we are training an object detector to detect faces in images. The first thing we'll do is extract HOG features from a positive dataset (that contains lots of examples of faces) and a negative dataset (which contains absolutely no faces at all, just a bunch of random images). We then train our Linear SVM.

However, this classifier may falsely detect faces in images where there are negative data (which contain no faces what-so-ever), and we collect all reports as a face. This process is called "hard-negative mining". In your classifier you want to **keep** that data since you can use it to better improve the classifier.

Finally, we need to re-train our classifier, which is just a 1-vs-1 SVM: either we take the HOG feature vectors from the original face dataset, the HOG feature vectors from the negative dataset, or a combination of both.

We then let this classifier train, with the end goal (normally) being that it only detects faces and not non-faces.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



vin July 3, 2015 at 2:15 pm #

hi adrian!

in your view, do you think the success of convolutional NNs have made other methods obsolete? thanks!



Adrian Rosebrock July 3, 2015 at 4:10 pm #

REPLY ↩

Definitely not. CNNs, and deep learning in general, is a tool just like anything else. There is a time and a place for it to be used. In some situations it's warranted. In some situations it's overkill. And in other it's just the flat out wrong choice. Take a look at my post on the [deep learning "band wagon"](#) mentality that rises up every 5-7 years.



ngapweitham July 7, 2015 at 8:02 am #

REPLY ↩

Hi, I have two questions after reading this post

1 : Would you show us step 1~step 6 in your blogs just like the pokedex example did?I found step 6(<http://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/>) but no step 1 ~step 5.Or I have to wait until the course " PyImageSearch Gurus " start?

2 : The SVM of opencv do not provide probability estimates, could I use Random tree of opencv to replace SVM?Or I should find another SVM libraries to develop the algorithms?

Again, thank you for your brilliant posts



Adrian Rosebrock July 7, 2015 at 8:47 am #

REPLY ↩

1. I plan to review Steps 1-6 inside the PyImageSearch Gurus course. I might cover these steps on the blog later on, but right now I am planning on detailing every step inside PyImageSearch Gurus.

2. I would recommend using the [scikit-learn implementation of the Linear SVM](#).

Free 21-day crash course on computer vision & image search engines

**ngapweitham** July 7, 2015 at 11:50 am #

REPLY ↩

Thanks for your suggestion, I have some questions to ask(new to the field of machine learning)

1 : what are the pros and cons of using the scikit-learn linear svm vs the random forest of opencv3.0?I think the random forest of opencv provide probability estimation too(<http://stackoverflow.com/questions/28303334/multi-class-image-classification-with-probability-estimation>)?

2 : Since HOG is a feature descriptor(am I right?), is it possible to use another descriptor to describe the object and feed to svm?Like akaze, kaze brisk, freak(In truth, I do not know their different) and so on

**Adrian Rosebrock** July 7, 2015 at 1:31 pm #

Simply put, a Linear SVM is very fast. A random forest classifier requires computing N number of trees. It also requires training. Linear SVMs excel at. SVMs can also return probability estimation:

2. HOG is indeed an image descriptor. You could certainly use any classifier, such as Local Binary Patterns, Zernike moments, Haralick SIFT, etc.; however, this requires the added step of creating a bag-of-features. I am covering all of these topics in detail inside my book. I haven't played around with machine learning, [Practical Python and](#)

Free 21-day crash course on computer vision & image search engines



Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!

**ngapweitham** July 10, 2015 at 6:17 am #

According to the dlib document(http://dlib.net/fhog_object_detection.html), I have two questions

1 : what is semi-rigid object?

2 : smoke is rigid-object or not?should I use HOG + SVM to classify smoke and non-smoke object?

**Adrian Rosebrock** July 10, 2015 at 6:27 am #

REPLY ↩

A semi-rigid object is something that has a clear form and structure, but can change slightly — consider how a human walks. We put one foot in front of the other, maybe move our arms a bit. The form is clearly the same, but it does change a little bit. HOG can be used to detect semi-rigid objects like humans provided that our poses are not too deviant from our training data. Finally, smoke is definitely *not a semi-rigid object*. *Smoke diffuses into the air and has no real shape or form.*

**ngapweitham** July 10, 2015 at 11:02 pm #

REPLY ↩

Thanks for your reply.What kind of descriptors and machine learning tools you will recommend if you want to classify smoke?

The thesis "Forest smoke detection using CCD camera and spatial-temporal variation of smoke visual patterns" use random forest

Another one "SMOKE DETECTION USING TEMPORAL HOG/HOF DESCRIPTORS AND ENERGY COLOUR STATISTICS FROM VIDEO" use HOG + HOF(I do not what is this yet)

I haven't go through the details of these papers yet, the second paper claim that the HOG can separate rigid object and non-rigid object very well.They use HOF to estimate the motion of the smoke, but it would be a problem if the video is not stable

Some implementation(Video-based Smoke Detection: Possibilities, Techniques and Challenges) machine learning at all

Free 21-day crash course on computer vision & image search engines



Adrian Rosebrock July 11, 2015 at 6:30 am #

REPLY ↩

I don't have any experience with smoke detection, so take this with a grain of salt, but I would apply something like a spatial pyramid, some really basic color channel information in the L*a*b* and then I would create an experiment where I try a bunch of image descriptors such as Local Binary Patterns (I would be really curious about this for this problem), Haralick texture, and I would even try HOG to see how it would perform. My guess is not great, but that's why we perform experiments.



ngapweitham July 11, 2015 at 8:23 am #

Thanks for your suggestion, I will try to find out which solution fits

After studying the cases of Plant classification, Handwriting recognition and I have more sense about what are this post talking about.

Free 21-day crash course on computer vision & image search engines



Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Douglas July 20, 2015 at 9:12 am #

Hi Adrian,

Awesome job, your posts are the best posts about OpenCV that I found!! C I have one question: Did you ever work with SVM for ONE_CLASS? I tri



Adrian Rosebrock July 20, 2015 at 10:14 am #

Hey Douglas, thanks for the comment. Personally, I prefer to use the [scikit-learn](#) machine learning implementations. I find the ones implemented in OpenCV to be a bit cumbersome and not as customizable.



Marine July 27, 2015 at 8:02 am #

REPLY ↩

Hi Adrian,

Thank you for this great tutorial !

I'm trying to train a svm for a face detection system. I've got positive and negative examples.

Do you know the size I should choose for the hog extraction ? I know it's often 64*128 for human detection, with blocksize =8. But I don't know for face problem.. I'm trying to calculate hog features on 25*35 images, with the function `hog.compute(...)` but it's not working...

Any help ? 😊

Plus, how do you know the optimal parameters for svm ?

Thanks !



Adrian Rosebrock July 27, 2015 at 10:17 am #

REPLY ↩

You normally would experiment with the optimal values for the HOG descriptor, the same goes for the parameters of the SVM. I actually prefer to use the [scikit-image implementation of HOG](#) and the [scikit-learn implementation of the Linear SVM](#). You can then apply a grid search and cross-validation to find the optimal values.

Ahmad August 7, 2015 at 6:46 am #

Free 21-day crash course on computer vision & image search engines



Hi Adrian,

As a research problem, I want to apply HOG for Content based image retrieval and definitely it is computationally expensive for large datasets. Please guide what to implement and test for efficient image retrieval. I've been tested the effects of changing block, cell, bin, orientation angel etc. , it give a bit performance gain but how does is this sufficient for defense?

Thanks,
Ahmad



Adrian Rosebrock August 7, 2015 at 7:04 am #

REPLY ↩

Hey Ahmad — can you let me know what the computational b
feature extraction itself? Or is it the actual search that is slow? If it's the
linear scan of the dataset (i.e. comparing the query features to every o
possible, I would suggest using an approximate nearest neighbor data
search.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Ahmad August 7, 2015 at 9:53 am #

Thanks Adrian for very quick reply.

Actually I'm pursuing my PhD. For my research contribution the ab
with PCA and classify using SVM at Corel dataset. Then I tuned Hi
but it is not a contribution. I'm stuck at how can I add my contributi
Ideas:

> To focus 'rotation invariance' in HOG, use HOG+LBP is worthabl
> What to apply to achieve more efficiency , recall rates with HOG'
New your valuable input on it!

Thanks,
Ahmad



Adrian Rosebrock August 8, 2015 at 6:38 am #

REPLY ↩

Nice, congrats on working on your PhD, that's very exciting. As far as rotation, HOG is definitely not rotation invariant. In you use only uniform LBPs you can achieve some level of rotation invariance. I don't recall any papers off the top of my head that have combined both HOG and LBP together, although it seems like there should be papers on the topic — it might be worth doing some more research in the literature and perhaps trying an experiment to see what your results look like.



vin September 14, 2015 at 11:08 am #

REPLY ↩

hi adrian,

what do you think is the next evolutionary step for deformable parts model? it seemed like it was the darling of the cv community 3-4 years ago...



Adrian Rosebrock September 15, 2015 at 6:09 am #

REPLY ↩

The next evolution is already here and (no pun intended), it's called Convolutional Neural Networks. CNNs are very accurate for image classification and object localization. However, the largest drawback is the incredible amount of data it takes to learn even a *single class*. All supervised machine learni
particularly data hungry. In those cases where data in minimal and dat
methods still work very well.

Free 21-day crash course on computer
vision & image search engines



Chi November 4, 2015 at 5:42 am #

REPLY ↩

Thank you for this great tutorial !



Adrian Rosebrock November 4, 2015 at 6:29 am #

REPLY ↩

Thanks Chi! 😊



Philippe Dykmans December 1, 2015 at 5:00 pm #

Hi Adrian,

While I totally agree with you that the classifiers provided by OpenCv are poorly trained. When you see these cascades come back with mere patches entirely, it's clear that the training images had poor alignment to start with. I then told the training program that these were faces... Yeah, well... I think they merely serve as demonstration material, and that they were never optimized for the Viola Jones detector, however, can yield amazing results.

That said; I'm interested to test your Python framework one day. Sounds good!

Grtz,
Philippe

Free 21-day crash course on computer vision & image search engines



Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Rahul December 30, 2015 at 12:26 am #

REPLY ↩

Hi Adrian,

Will it produce good (accurate) results, If we use HOG features and SVM for vehicle licence plate detection? I only need to detect the rectangle region where license plate is present, recognition is not necessary.



Adrian Rosebrock December 30, 2015 at 6:59 am #

REPLY ↩

It really depends on your license plate dataset. If the license plates are skewed or rotated, it will be harder for HOG + Linear SVM to return good results (since HOG is not rotation invariant).



Nicolò January 9, 2016 at 7:43 pm #

REPLY ↩

Hi Adrian,

Thank you for all your great tutorials!

I'm trying to implement Exemplar SVMs with some friends for a school project but we got stuck. We followed your tutorials but our classifier doesn't detect anything. It seems that each single classifier doesn't train properly: as the number of negative sample grows, we get only 2 support vectors (could it be a problem?). We tried `linearSVC(loss='hinge')`, `SVC(kernel='linear')` and `linear_model.SGDClassifier()` but nothing changes. Maybe we're doing something wrong with features extraction? Any hint?



Adrian Rosebrock January 10, 2016 at 7:44 am #

Free 21-day crash course on computer
vision & image search engines

For help with Exemplar SVMs, I suggest reaching out to [Tomasz Malisiewicz](#), who authored the original paper on Exemplar SVMs. It's hard to say what your exact problem is, but when you read through the original paper you'll see that they use a "calibration" step where they tune the support vectors for each classifier. I have never implemented Exemplar SVMs before, but it seems like this could also cause a problem with your classification. It's also hard to tell if feature extraction is your problem without knowing your actual feature extraction process.



sruthi February 22, 2016 at 8:46 am #

REPLY ↩

Sir,

I read all the above information regarding HOG. I am currently working on HOG for human detection. Do you have any MATLAB codes for HOG with which I can go on with the project.

Thank you in advance



Adrian Rosebrock February 22, 2016 at 4:19 pm #

No, I do not have any codes related to MATLAB for HOG detection. I have implemented this HOG + Linear SVM framework inside the [Pylma](#)

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Manez February 22, 2016 at 1:45 pm #

Hi Adrian ! Awesome tutorial !

I am using the HOG descriptor from skimage to create a traffic stop sign detector. It takes about 0.5s for a (255, 496) frame).

Did you use the same descriptor as me or the one from OpenCV-Python ?

Do you have some tips for increasing the speed ?



Adrian Rosebrock February 22, 2016 at 4:17 pm #

REPLY ↩

I use the same scikit-image descriptor as well. As for increasing speed, try splitting processing of the image pyramid to different cores of the processor, that way each core can process a separate layer of the pyramid independently.



Manez February 23, 2016 at 9:26 am #

REPLY ↩

Ok !

I have registered to the PyImageSearch Gurus recently. Are you going to talk about multiprocessing module in the course ?



Adrian Rosebrock February 23, 2016 at 3:21 pm #

REPLY ↩

I don't directly cover it in the course, although it's an easy add-on once you've seen the source code for implementing an object detector from scratch. Of course, I also share more resources to make multi-processing easier inside the course 😊



Shahab March 9, 2016 at 6:33 am #

REPLY ↩

Hi Adrian,

Free 21-day crash course on computer
vision & image search engines

Thank you so much for this perfect step-by-step instruction. I'm working on recognizing traffic signs using SVM + HOG. The algorithm is great and the accuracy is very reasonable; however, the speed is deriving me crazy. For each image, of hundreds, it takes more than 5 minutes to yield the results. I really need to shorten this time to at most a minute or less. The sliding window is the main problem. How can I handle the slow moving of sliding window?

Thank you in advance.



Adrian Rosebrock March 9, 2016 at 4:39 pm #

REPLY ↩

The sliding window itself isn't the actual problem — it's the time associated with extracting the HOG features and then passing them through the SVM. Your pyramid scale may also be hurting performance. I would suggest increasing the size of the image pyramid along with the step size of the sliding window. The larger the pyramid scale and the sliding window step size are, the less



Dmitry Zaytsev April 19, 2016 at 4:49 am #

What about training a Haar (well, LBP) cascade first, lower the threshold about false-positives. Then, instead of sliding the window just go through the image once. I didn't tried that myself, but maybe you did? Or is there is something



Adrian Rosebrock April 19, 2016 at 6:49 am #

There isn't anything "wrong" with that approach, but it's not the best. You can tune any-and-all parameters associated with them. It would be more accurate HOG classifier or pushing the computation to the GPU. The "region proposals" which is the process of iteratively finding "interesting" regions of the image to classify rather than applying an exhaustive sliding window + image pyramid approach.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



romanzo April 27, 2016 at 3:11 am #

REPLY ↩

Hi Adrian great post!

I'm trying to build my own SVM and detect object them on test image using `hog.detectMultiScale`.

I tried to build a SVM classifier from sklearn and opencv using crossvalidation. It gives a good accuracy during the cross validation and also while using other test images (however the opencv one just gives me one support vector which is kinda of weird).

If i use sklearn SVM, I load the `coef_` attribute of my SVC object (which corresponds to the primal sv) using `setSVMDetector` but the detection gives a lot of false positive and not even a true positive on the images that were well detected with the sklearn predict...

Have you trained your own svm model and used it with `detectMultiScale`? Any ideas of what I could do wrong.

Thanks a lot!



Adrian Rosebrock April 28, 2016 at 3:23 pm #

REPLY ↩

I personally don't like using OpenCV to train a custom HOG detector from scratch. Instead, I prefer using scikit-learn as it gives me more control. I detail the general process in the post we're currently commenting on and then [demonstrate how to implement HOG + Linear SVM inside PyImageSearch Gurus](#).



Siva Krishna May 14, 2016 at 11:38 am #

REPLY ↩

Free 21-day crash course on computer vision & image search engines

Hi Adrian,

I am working on an object detector for fallen people using HOG. As a part of that I have collected some data manually and used some data available online. But I am not obtaining good accuracy. However my data set size is of size train(fall – 400 neg – 1031), test(fall – 246 neg – 503). After initial training I did hard mining also. I have used HOG parameters like cell size – 8×8, Block size – 16×16 Window size – 128×64 Bins – 9. How to improve accuracy?. Accuracy depends on the available data set size, doesn't it? Should I play with cell size and other parameters? Thanks in advance.



Adrian Rosebrock May 15, 2016 at 11:04 am #

REPLY ↩

In general, the more data you have the better. In terms of negative samples, I would try to increase this dramatically. Ideally you should ~10x more negative samples than positive. You will need to play with HOG parameters, but I think this is less of a concern.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Rabelani Netshifhire May 17, 2016 at 4:34 pm #

Hi Adrian. Where is the actual implementation and Code for HOG SVM, i see in this post it was supposed to be done next week



Adrian Rosebrock May 19, 2016 at 6:11 pm #

Sorry for any confusion, but the actual implementation of HOG is in the [crash course](#).



Bill May 19, 2016 at 5:39 am #

REPLY ↩

Is it possible to define a distance function between the HOG (or other) descriptors for two images? My application is just to iteratively find interestingly similar images, without seeing duplicates.



Adrian Rosebrock May 19, 2016 at 5:59 pm #

REPLY ↩

Absolutely! Typically you would use either the Euclidean distance or the chi-squared distance. Both are implemented in NumPy/Scipy.



Bill May 23, 2016 at 5:43 am #

REPLY ↩

Given two sets of descriptors, one interesting metric might be the https://en.wikipedia.org/wiki/Hausdorff_distance which would be the worst closest match distance between set features, or maybe the average of the closest matches.



Adrian Rosebrock May 23, 2016 at 7:20 pm #

REPLY ↩

For comparing histograms, I would recommend utilizing the chi-squared distance — this will give you reasonable performance.



Michael June 29, 2016 at 7:41 pm #

Thank you Adrian

Free 21-day crash course on computer
vision & image search engines



vani July 21, 2016 at 3:10 am #

REPLY ↩

hi ,

feeling somewhat hard to learn..

trying to implement the code..

a hats off to you..



faryad September 11, 2016 at 2:39 pm #

REPLY ↩

Thank you Adrian it's very good

Good Luck



Romanzo September 19, 2016 at 2:15 am #

Hi Adrian,

In your post you said number of negatives samples >> number of positives internet and usually people tend to use balanced data for a binary classification unbalanced data in this case?

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Adrian Rosebrock September 19, 2016 at 1:02 pm #

Indeed, that is a great question. Normally we would use a balanced dataset so many more negative examples in the HOG + Linear SVM framework?

The reason is false-positive detections. If we don't use a lot of negative examples we get *many* false-positive detections.



Romanzo September 20, 2016 at 3:37 am #

REPLY ↩

Thanks for the answer. I asked because I'm wondering if the unbalanced or balanced data could actually be the cause of some of the errors I'm encountering .

First I'm doing cross validation to confirm i have good training tests. Even with unbalanced data, i get a low FNR and FPR during the training. But when using the detectMultiScale on a video I get much more FP.

Does that mean my model is overfitting during training or my datasets are not good enough?

Secondly, I'm trying to reduce this number of FP by using hard data mining. The optimal number of hard negative samples N and the optimal C of the new retrained model are found via random research. But usually the lowest FNR and FPR i get is using 0 hard negative. Which surprise me.

Does that mean my model was really overfitting and adding some more examples is worst. Should i based my choice of optimal C and N by testing another set rather than getting a FNR and FPR on my training set during cross validation? Any other suggestions? Thanks for your time!



Adrian Rosebrock September 21, 2016 at 2:14 pm #

REPLY ↩

It's hard to say without seeing your datasets. I would start by asking yourself if your training data is representative of your testing data. If your training data doesn't look anything like your testing data then you can expect to get strange results.

As you suggested, it's normal to set your number of samples and SVM value of C based off a random search or grid search — this is a pretty standard procedure. Given that you hard-negatives is used leads me to believe that your training d

Free 21-day crash course on computer vision & image search engines



Romano September 27, 2016 at 3:02 am #

For the positive training set, i'm using images from a scene view by a fixed camera. The test images are also taken from this camera. But the negative training set, I'm using the one from INRIA. Should a negative training set be whatever scene without a positive instance or should it reflect the scene (I'm working indoor so should my negative set be only a view from the camera without my object)?



Adrian Rosebrock September 27, 2016 at 8:38 am #

It really depends on where you depend on deploy very common to use negative instances that simply do not said, if your production environment will have a lot of varyii using negative views from your camera.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



sonu September 23, 2016 at 12:39 am #

hi

Which pyramid should be used while implementing with HOG for person de and laplacian etc



Adrian Rosebrock September 23, 2016 at 6:47 am #

If review image pyramids for object detection [in this blog post](#). You should actually stay away from blurring your images in the image pyramid when applying the HOG descriptor was that will decrease your accuracy.



Optimus1072 September 24, 2016 at 6:16 am #

REPLY ↩

Bro! I don't like this site. I come here to read about one topic and ends up spending an hour or two reading different articles 😞



Gabriel October 6, 2016 at 2:35 pm #

REPLY ↩

Hey man,

Great site. I have a question for you. If you train your svm with a certain kind of object size in mint (say 60X180) , then how does the detectMultiScale method figure out what kind of crops it needs to take from the image input so that the descriptor from the compute method will be a vector that's the same size as the ones used in the training process ?

Thanks,
Gabriel



Adrian Rosebrock October 7, 2016 at 7:31 am #

REPLY ↩

If you want to train your own custom object detector using HOG + Linear SVM then I wouldn't recommend using OpenCV. I would instead use [methods I detail inside the PyImageSearch Gurus course](#). The Python + OpenCV bindings for HOG are not very friendly. You also don't have much control over the training process.

Gabriel October 7, 2016 at 8:30 am #

Free 21-day crash course on computer vision & image search engines



I just found out today the hard way that `setSVMdetector` actually requires a vector instead of a `Svm` object. At this point it seems easier to just build my own hog detector with sliding windows and pyramid images and use that with a `sklearn svm`. Thanks a lot for the reply and i'll say it again that i really appreciate the info on this

site



Adrian Rosebrock October 11, 2016 at 1:18 pm #

REPLY ↩

No problem Gabriel, I'm happy to help. I know I already mentioned this so I don't want to beat it to death, but I do demonstrate how to train your own HOG detector using sliding windows, image pyramids, and `scikit-learn's SVM` module inside the [PyImageSearch Gurus course](#). If you get stuck implementing on your own (or would like to learn from well documented tutorials) then I would definitely su

Free 21-day crash course on computer vision & image search engines



Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Hoang October 17, 2016 at 11:51 am #

Hi Adrian,

Great site for OpenCV and Image Processing.

I have some questions:

1. As you said in Step 2: Sample N negative samples from a negative train want to detect and extract HOG descriptors from these samples as well. In So example if I want to train a smile detector, the positive images contain r faces. It's mean that both positive and negative are include face, so is it co

2. In case of smile detector, I want to make a classifier with 3-classes: norm Viola-Jones algorithm and it's not good. Now, i'm using Logistic Regression opening mouth, normal – smile. It's better than Viola-Jones, but it still get n HOG+Linear SVM better? Or could you suggest any approach?

3. With HOG + Linear SVM Model or any Model you suggestion, can I save this model to re-use it in mobile, example save model as xml file and load it in Objective-C (iOS) or Java (Android)? So I'm just predict later, not training again. Finally, may I run realtime in mobile (30fps) with your suggestion model on mobile device?



Adrian Rosebrock October 17, 2016 at 3:58 pm #

REPLY ↩

1. Semantically this is not a contradiction, but I think you'll run into issues if you use your frowning lips as negative samples. HOG is good at discriminating between objects, but lips are still lips and look similar. I would experiment with both, but my guess is that you'll find better performance having the detectors trained on datasets that do not contain examples from the other set. Overall, I would suggest you take a look at "facial landmarks" (keypoints detected on the face). The geometry of these points might help you build a more accurate classifier.

2. HOG + Linear SVM tends to majorly beat out the Viola-Jones detector in terms of both accuracy and false-positives. If you're getting many false-positives you should be applying more aggressive hard-negative mining.

3. I'm not sure about this since I don't do much mobile development. You might have to train the model on a library specific to iOS/Android, but once it's trained, you can deploy it into the production devices.



Walid Ahmed November 4, 2016 at 1:32 pm #

REPLY ↩

Thanks a lot



Saloni Mittal November 10, 2016 at 7:12 am #

REPLY ↩

Hi Adrian,

I am doing a project on traffic sign detection. Ideally how many hog feature if the dataset contains about 2000 positive images and 5000 negatives? Al

Free 21-day crash course on computer vision & image search engines



Adrian Rosebrock November 10, 2016 at 7:29 am #

REPLY ↩

The number of extracted HOG features is entirely dependent on your dataset. A good rule of thumb is to have approximately 5-10x as many negatives as your positives and then perform hard-negative mining. As far as a kernel goes, a linear kernel tends to be standard due to how fast it is. Given your current dataset I would suggest performing an initial experiment to obtain a baseline. From there you can try adjusting the number of negatives and hard-negatives.



Gerardo Rosiles November 10, 2016 at 2:07 pm #

REPLY ↩

Hi Adrian,

Regarding the detection performance. How do you measure it as compare metrics like precision and recall?

It seems to me you need to first evaluate the classifier itself with a set of ability of the classifier and then evaluate detection performance in some of

My impression is that the detector feeds a very large number of "non-object negatives and a few (or none) false positives plus a few true positives reprocessed with non-maxima suppression). I think the typical ML metrics are given the in-balance between positive and negative test vectors.

So, I guess my question is, how do you evaluate your models after training

Thanks,

Gerardo

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Adrian Rosebrock November 14, 2016 at 12:20 pm #

REPLY ↩

Hey Gerardo — I actually cover the answer to that question in my latest blog post. You use the [Intersection over Union](#) metric to measure the quality of a detection.



cam nguyen November 12, 2016 at 11:02 am #

REPLY ↩

Hi Adrian, thank you so much for your great article !

I'm learning about human detection using HOG descriptor + Linear SVM.

I am using Verilog HDL to build my system on FPGA board.

My huge issue is : how to train the input dataset by using HOG descriptor? The detailed steps? Because verilog HDL does not have library for any the mathematical function? So, I am hopefully received your detailed steps ?

Thanks so much!



Adrian Rosebrock November 14, 2016 at 12:09 pm #

REPLY ↩

Hi Cam — I can't speak for the FGPA side of things (since I don't have experience in that area), but I provided *very detailed* steps + source code to train your own custom object detector using the HOG + Linear SVM method [inside the PyImageSearch Gurus course](#).



Yash Karnawat November 14, 2016 at 7:46 am #

REPLY ↩

Free 21-day crash course on computer vision & image search engines

Hi Adrian ,

I am a huge fan of your blog and really appreciate what you do . I am creating my own object classifier for sign language detection . I am facing some problem with localizing the hand of the person in an image. The hand can be in a square box of size in the range 70 to 220px . I wanted to know how can I apply both the image pyramid technique and hard-negative mining. I mean for which square size do I train my binary classifier and how do I go about the process.



Adrian Rosebrock November 14, 2016 at 12:02 pm #

REPLY ↩

The actual size of the hand doesn't matter as long as the aspect ratio (ratio of width to height) is the same. That is the entire point of applying a sliding window + image pyramid — to detect objects at various scales and locations of an image. I detail a thorough solution to training a custom HOG + Linear SVM classifier in my [Gurus course](#). I would suggest starting there.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!



Saloni Mittal November 22, 2016 at 10:41 am #

Hi Adrian,

How can we retrain an existing svm without starting from scratch the retrain While performing hard negative training I got a Memory Error in the end, at the numpy array size! (My dataset contains 10K positives and 60K negative negatives. The number of hog features extracted from every image is 1568 Thanks in advance! 😊



Adrian Rosebrock November 22, 2016 at 12:21 pm #

There are machine learning algorithms that can be trained in less than one of them. In this case, you'll need to reduce the size of your training dataset. I would suggest reducing the size of the true negatives and keeping only the most confident hard-negatives.

Trackbacks/Pingbacks

[Non-Maximum Suppression for Object Detection in Python - PyImageSearch](#) - November 17, 2014

[...] If you remember, last week we discussed Histogram of Oriented Gradients for Objection Detection. [...]

[Capturing mouse click events with Python and OpenCV - PyImageSearch](#) - March 9, 2015

[...] In this example we'll click and drag a rectangular Region of Interest (ROI) and crop it from our image. This technique is especially helpful if you are labeling data as input to an image classification algorithm. [...]

[Image Pyramids with Python and OpenCV - PyImageSearch](#) - March 16, 2015

[...] see, a few months ago I wrote a blog post on utilizing the Histogram of Oriented Gradients image descriptor and a Linear SVM to detect objects in images. This 6-step framework can be used to [...]

[Sliding Windows for Object Detection with Python and OpenCV - PyImageSearch](#) - May 11, 2015

[...] fact, both sliding windows and image pyramids are both used in my 6-step HOG + Linear SVM object classification [...]

[Pedestrian Detection OpenCV - PyImageSearch](#) - November 9, 2015

[...] OpenCV ships with a pre-trained HOG + Linear SVM model that can be used to perform pedestrian detection in both images and video streams. If you're not familiar with the Histogram of Oriented Gradients and Linear SVM method, I suggest you read this blog post where I discuss the 6 step framework. [...]

[HOG detectMultiScale parameters explained - PyImageSearch](#) - November 17, 2015

[...] accomplish this, we leveraged the built-in HOG + Linear SVM detector that OpenCV ships with, allowing us to detect people in [...]

[Detecting machine-readable zones in passport images - PyImageSearch](#) - November 30, 2015

[...] to detect the MRZ region of a passport that you need a bit of machine learning, perhaps using the Linear SVM + HOG framework to construct an "MRZ detector" — but that would be [...]

[Intersection over Union \(IoU\) for object detection - PyImageSearch](#) - November 7, 2016

[...] is interested in building a custom object detector using the HOG + Linear SVM framework for his final year project. He understands the steps required to build the object detector well [...]

Free 21-day crash course on computer vision & image search engines

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Resource Guide (it's totally free).



Click the button below to get my **free 11-page Image Search Engine techniques** that I don't publish on this blog and start building image search engines.

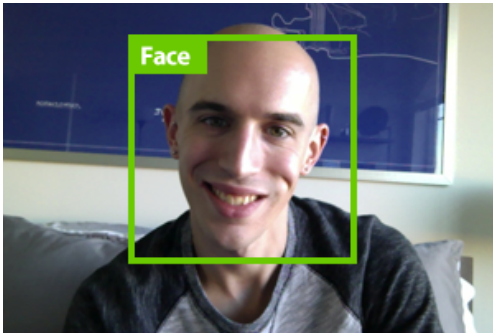
[Download for Free](#)

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

LET'S DO IT!

You can detect faces in images & video.



Are you interested in **detecting faces in images & video**? But **tired of Googling for tutorials** that *never work*? Then let me help! I guarantee that my new book will turn you into a **face detection ninja** by the end of this weekend. [Click here to give it a shot yourself.](#)

[CLICK HERE TO MASTER FACE DETECTION](#)

PyImageSearch Gurus: NOW ENROLLING!

Free 21-day crash course on computer vision & image search engines

The PyImageSearch Gurus course is **now enrolling!** Inside the course you'll learn how to perform:

- Automatic License Plate Recognition (ANPR)
- Deep Learning
- Face Recognition
- *and much more!*

Click the button below to learn more about the course, take a tour, and get 10 (FREE) sample lessons.

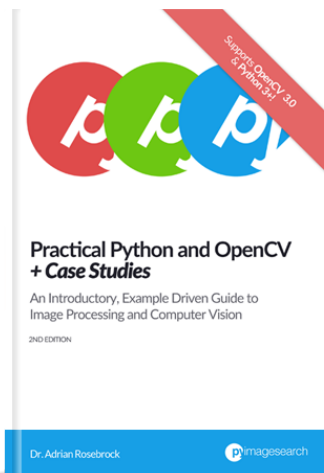
TAKE A TOUR & GET 10 (FREE) LESSONS

Hello! I'm Adrian Rosebrock.



I'm an entrepreneur and Ph.D who has launched two successful to share my tips, tricks, and hacks I've learned along the way.

Learn computer vision in a single weekend.



Want to learn computer vision & OpenCV? I can teach you in a **single weekend**. I know. It sounds crazy, but it's no joke. My new book is your **guaranteed, quick-start guide** to becoming an OpenCV Ninja. So why not give it a try? [Click here to become a computer vision ninja.](#)

[CLICK HERE TO BECOME AN OPENCV NINJA](#)

Subscribe via RSS



Never miss a post! Subscribe to the PyImageSearch RSS Feed and keep up to date with my image search engine tutorials, tips, and tricks

POPULAR

Install OpenCV and Python on your Raspberry Pi 2 and B+
FEBRUARY 23, 2015

Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox
JUNE 1, 2015

How to install OpenCV 3 on Raspbian Jessie
OCTOBER 26, 2015

Free 21-day crash course on computer vision & image search engines

Basic motion detection and tracking with Python and OpenCV MAY 25, 2015	
Install OpenCV 3.0 and Python 2.7+ on OSX JUNE 15, 2015	
Install OpenCV 3.0 and Python 2.7+ on Ubuntu JUNE 22, 2015	
Accessing the Raspberry Pi Camera with OpenCV and Python MARCH 30, 2015	

Search

Search...

Find me on [Twitter](#), [Facebook](#), [Google+](#), and [LinkedIn](#).
© 2016 PyImageSearch. All Rights Reserved.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Free 21-day crash course on computer vision & image search engines