# Haar Feature-based Cascade Classifier for Object Detection

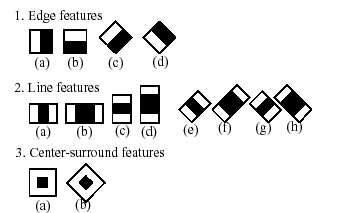
<https://www.docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html>

The object detector described below has been initially proposed by Paul Viola [[Viola01]](https://www.docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#viola01) and improved by Rainer Lienhart [[Lienhart02]](https://www.docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#lienhart02).

First, a classifier (namely a *cascade of boosted classifiers working with haar-like features*) is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.

The word “cascade” in the classifier name means that the resultant classifier consists of several simpler classifiers (*stages*) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word “boosted” means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques (weighted voting). Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost are supported. The basic classifiers are decision-tree classifiers with at least 2 leaves. Haar-like features are the input to the basic classifiers, and are calculated as described below. The current algorithm uses the following Haar-like features:



The feature used in a particular classifier is specified by its shape (1a, 2b etc.), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in the case of the third line feature (2c) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas. The sums of pixel values over a rectangular regions are calculated rapidly using integral images (see below and the [integral()](https://www.docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html#void integral(InputArray src, OutputArray sum, int sdepth)) description).

To see the object detector at work, have a look at the facedetect demo: <https://github.com/opencv/opencv/tree/master/samples/cpp/dbt_face_detection.cpp>

The following reference is for the detection part only. There is a separate application called opencv\_traincascade that can train a cascade of boosted classifiers from a set of samples.

Note

In the new C++ interface it is also possible to use LBP (local binary pattern) features in addition to Haar-like features.

|  |  |
| --- | --- |
| [[Viola01]](https://www.docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#id1) | Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001. The paper is available online at <http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_CVPR2001.pdf> |

|  |  |
| --- | --- |
| [[Lienhart02]](https://www.docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#id2) | Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002. This paper, as well as the extended technical report, can be retrieved at [http://www.multimedia-computing.de/mediawiki//images/5/52/MRL-TR-May02-revised-Dec02.pdf](http://www.multimedia-computing.de/mediawiki/images/5/52/MRL-TR-May02-revised-Dec02.pdf) |

############################################################################################

**Viola Jones face detection and tracking explained (in detail)**

[Ramsri](https://www.youtube.com/channel/UCOvza3noTqLHWdYgrsTZrbw)

Viola Jones face detection algorithm and tracking is explained. Includes explanation of haar features, integral images, adaboost, cascading classifiers, mean shift tracking and Camshift tracking. Presentation link:

https://www.youtube.com/watch?v=WfdYYNamHZ8

############################################################################################

**CascadeClassifier API description:**

**CascadeClassifier::CascadeClassifier**

Loads a classifier from a file.

**C++:** CascadeClassifier::CascadeClassifier(const string& **filename**)

**Python:** cv2.CascadeClassifier([filename]) → <CascadeClassifier object>

|  |  |
| --- | --- |
| **Parameters:** | **filename** – Name of the file from which the classifier is loaded. |

**CascadeClassifier::empty**

Checks whether the classifier has been loaded.

**C++:** bool CascadeClassifier::empty() const

**Python:** cv2.CascadeClassifier.empty() → retval

**CascadeClassifier::load**

Loads a classifier from a file.

**C++:** bool CascadeClassifier::load(const string& **filename**)

**Python:** cv2.CascadeClassifier.load(filename) → retval

|  |  |
| --- | --- |
| **Parameters:** | **filename** – Name of the file from which the classifier is loaded. The file may contain an old HAAR classifier trained by the haartraining application or a new cascade classifier trained by the traincascade application. |

**CascadeClassifier::detectMultiScale**

Detects objects of different sizes in the input image. The detected objects are returned as a list of rectangles.

**C++:** void CascadeClassifier::detectMultiScale(const Mat& **image**, vector<Rect>& **objects**, double **scaleFactor**=1.1, int **minNeighbors**=3, int **flags**=0, Size **minSize**=Size(), Size **maxSize**=Size())

**Python:** cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]) → objects

**Python:** cv2.CascadeClassifier.detectMultiScale(image, rejectLevels, levelWeights[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]]) → objects

**C:** CvSeq\* cvHaarDetectObjects(const CvArr\* **image**, CvHaarClassifierCascade\* **cascade**, CvMemStorage\* **storage**, double **scale\_factor**=1.1, int **min\_neighbors**=3, int **flags**=0, CvSize **min\_size**=cvSize(0,0), CvSize **max\_size**=cvSize(0,0) )

**Python:** cv.HaarDetectObjects(image, cascade, storage, scale\_factor=1.1, min\_neighbors=3, flags=0, min\_size=(0, 0)) → detectedObjects

|  |  |
| --- | --- |
| **Parameters:** | * **cascade** – Haar classifier cascade (OpenCV 1.x API only). It can be loaded from XML or YAML file using [Load()](https://www.docs.opencv.org/2.4/modules/core/doc/old_xml_yaml_persistence.html#void* cvLoad(const char* filename, CvMemStorage* memstorage, const char* name, const char** real_name)). When the cascade is not needed anymore, release it using cvReleaseHaarClassifierCascade(&cascade). * **image** – Matrix of the type CV\_8U containing an image where objects are detected. * **objects** – Vector of rectangles where each rectangle contains the detected object. * **scaleFactor** – Parameter specifying how much the image size is reduced at each image scale. * **minNeighbors** – Parameter specifying how many neighbors each candidate rectangle should have to retain it. * **flags** – Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade. * **minSize** – Minimum possible object size. Objects smaller than that are ignored. * **maxSize** – Maximum possible object size. Objects larger than that are ignored. |

The function is parallelized with the TBB library.

**Note**

* (Python) A face detection example using cascade classifiers can be found at opencv\_source\_code/samples/python2/facedetect.py

The function is automatically called by [CascadeClassifier::detectMultiScale()](https://www.docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#void CascadeClassifier::detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor, int minNeighbors, int flags, Size minSize, Size maxSize)) at every image scale. But if you want to test various locations manually using [CascadeClassifier::runAt()](https://www.docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#int CascadeClassifier::runAt(Ptr<FeatureEvaluator>& feval, Point pt, double& weight)), you need to call the function before, so that the integral images are computed.

**groupRectangles**

Groups the object candidate rectangles.

**C++:** void groupRectangles(vector<Rect>& **rectList**, int **groupThreshold**, double **eps**=0.2)

**C++:** void groupRectangles(vector<Rect>& **rectList**, vector<int>& **weights**, int **groupThreshold**, double **eps**=0.2)

**Python:** cv2.groupRectangles(rectList, groupThreshold[, eps]) → rectList, weights

|  |  |
| --- | --- |
| **Parameters:** | * **rectList** – Input/output vector of rectangles. Output vector includes retained and grouped rectangles. (The Python list is not modified in place.) * **groupThreshold** – Minimum possible number of rectangles minus 1. The threshold is used in a group of rectangles to retain it. * **eps** – Relative difference between sides of the rectangles to merge them into a group. |

The function is a wrapper for the generic function [partition()](https://www.docs.opencv.org/2.4/modules/core/doc/clustering.html#template<typename _Tp, class _EqPredicate> int partition(const vector<_Tp>& vec, vector<int>& labels, _EqPredicate predicate)) . It clusters all the input rectangles using the rectangle equivalence criteria that combines rectangles with similar sizes and similar locations. The similarity is defined by eps. When eps=0 , no clustering is done at all. If \texttt{eps}\rightarrow +\inf, all the rectangles are put in one cluster. Then, the small clusters containing less than or equal to groupThreshold rectangles are rejected. In each other cluster, the average rectangle is computed and put into the output rectangle list.

##############################################################################################

# Code examples for cv2.CascadeClassifier

<https://programtalk.com/python-examples/cv2.CascadeClassifier/>

############################################################################################

# [Recommended values for OpenCV detectMultiScale() parameters](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters)

<https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters>

What are the recommended parameters for [CascadeClassifier::detectMultiScale()](http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html#cascadeclassifier-detectmultiscale) and depending on which factors I should change default parameters?

void CascadeClassifier::detectMultiScale(

const Mat& image,

vector<Rect>& objects,

double scaleFactor=1.1,

int minNeighbors=3,

int flags=0,

Size minSize=Size(),

Size maxSize=Size() )

## Answer 1

Amongst these parameters, you need to pay more attention to four of them:

* **scaleFactor** – Parameter specifying how much the image size is reduced at each image scale.

Basically the scale factor is used to create your scale pyramid. More explanation can be found [here](https://sites.google.com/site/5kk73gpu2012/assignment/viola-jones-face-detection#TOC-Image-Pyramid). In short, as described [here](http://answers.opencv.org/question/10654/how-does-the-parameter-scalefactor-in-detectmultiscale-affect-face-detection/), your model has a fixed size defined during training, which is visible in the xml. This means that this size of face is detected in the image if present. However, by rescaling the input image, you can resize a larger face to a smaller one, making it detectable by the algorithm.

**1.05** is a good possible value for this, which means you use a small step for resizing, i.e. reduce size by 5%, you increase the chance of a matching size with the model for detection is found. This also means that the algorithm works slower since it is more thorough. You may increase it to as much as 1.4 for faster detection, with the risk of missing some faces altogether.

* **minNeighbors** – Parameter specifying how many neighbors each candidate rectangle should have to retain it.

This parameter will affect the quality of the detected faces. Higher value results in less detections but with higher quality. **3~6** is a good value for it.

* **minSize** – Minimum possible object size. Objects smaller than that are ignored.

This parameter determine how small size you want to detect. You decide it! Usually, **[30, 30]** is a good start for face detection.

* **maxSize** – Maximum possible object size. Objects bigger than this are ignored.

This parameter determine how big size you want to detect. Again, you decide it! **Usually, you don't need to set it manually**, the default value assumes you want to detect without an upper limit on the size of the face.

[Abhijat Biswas](https://stackoverflow.com/users/4646269/abhijat-biswas)

1641414 bronze badges

[herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao)

42k2020 gold badges109109 silver badges147147 bronze badges

* ?

Are image pyramids used in combination with sliding window techniques? Isn't it true that if we're using a sliding window that scans the image at different scales and different sizes, then we don't need to use image pyramids? – [user961627](https://stackoverflow.com/users/961627/user961627) [May 18 '14 at 13:22](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment36461041_20805153)

* ?

and what does "minNeighbors" refer to? is it about pruning excessive detections around the same face? – [user961627](https://stackoverflow.com/users/961627/user961627) [May 18 '14 at 13:23](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment36461065_20805153)

* ?

@user961627 Yes, image pyramids and sliding window techniques are combined being used. If you use a sliding window that scans the image at different scales and different sizes, you don't need to use image pyramids anymore as image pyramids just up/down-samples images when processing. For minNeighbors, yes, it is about pruning excessive detections around the same face. Please refer to the answer for the description. – [herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao" \o "41,956 reputation) [May 18 '14 at 15:50](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment36463959_20805153)

  Thanks, the explanation really helps. Do you know of any python implementation of the pruning of excessive detections? – [user961627](https://stackoverflow.com/users/961627/user961627) [May 18 '14 at 20:54](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment36469854_20805153)

  1

@user961627 There is also the Python API of OpenCV, see [here](http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html#cv2.CascadeClassifier.detectMultiScale). – [herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao" \o "41,956 reputation) [May 19 '14 at 3:28](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment36474962_20805153)

 Is this *grouprectangles* pruning function? [docs.opencv.org/modules/objdetect/doc/…](http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html#grouprectangles) – [user961627](https://stackoverflow.com/users/961627/user961627) [May 21 '14 at 11:26](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment36573975_20805153)

 1

Yes, you can use it as the pruning function or use minNeighbors parameter of detectMultiScale(). – [herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao" \o "41,956 reputation) [May 21 '14 at 12:07](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment36575638_20805153)

 1

scaleFactor depends on the size of your trained detector. e.g. if your detector windows has a training size of 20x20 and you want to detect for 21x21, 22x22 etc (each pixel size without skipping something) too, you have to choose scaleFactor = 1.05. If your trained detector is 10x10 and you want to detect for 11x11, 12x12 and so on (each pixel size), you can choose scaleFactor = 1.10 for haarcascade, your detector size is visible in the xml – [Micka](https://stackoverflow.com/users/2393191/micka" \o "15,345 reputation) [Jan 13 '15 at 13:51](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment44244211_20805153)

 @Micka Incorporate it to the answer. Thanks to make it clear. – [herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao" \o "41,956 reputation) [Jan 13 '15 at 14:29](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment44245689_20805153)

 4

@Micka Don't 100% agree on scaleFactor. In fact, you want it as high as possible while still getting "good" results, and this must be determined somewhat empirically. It's heavily dependent on the target to be detected, the type of cascade and the training; Even and a value as high as 1.1 for a 24x24 FD cascade has worked for me in the past. A too low value for either scaleFactor or minSize will result in huge computational costs because many more pyramid layers need to be generated. A factor of 1.05 requires roughly double the # of layers (and >2x the time) than 1.1 does. – [Iwillnotexist Idonotexist](https://stackoverflow.com/users/2809095/iwillnotexist-idonotexist" \o "12,117 reputation) [Jan 13 '15 at 14:37](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment44246067_20805153)

  2

@IwillnotexistIdonotexist that's obviously true, sorry I didn't mention it. I just wanted to say that it is possible to let the detector choose window sizes to someones exact needs and that scaleFactor can be computed to fulfill those needs. But it is hard to recommend a value without knowing the size of the trained window and the min/max size in combination with the constraints to computational time and detection sensitivity/specifity :) – [Micka](https://stackoverflow.com/users/2393191/micka" \o "15,345 reputation) [Jan 13 '15 at 16:09](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment44249883_20805153)

  Playing around with this and I see 0 detections... How would I go about troubleshooting? What size image is recommended (using 1080p video here, does it matter) ? I tried scale factor from 1.005, and still no result. any hint? I followed this: [docs.opencv.org/master/d7/d8b/…](http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0) – [MrE](https://stackoverflow.com/users/2494262/mre" \o "11,585 reputation) [Jan 7 '16 at 23:48](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57080295_20805153)

  @MrE Could you share your test image? – [herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao" \o "41,956 reputation) [Jan 8 '16 at 0:15](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57080820_20805153)

 just using my webcam on laptop, so it's just me on white background. is there a requirement for size of subject with regard to the size of the picture? what ratio is typical? – [MrE](https://stackoverflow.com/users/2494262/mre" \o "11,585 reputation) [Jan 8 '16 at 0:28](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57081015_20805153)

 Tried with this [download-free-wallpaper.com/img89/zjvvrfxqujoblxuzomsi.jpg](http://www.download-free-wallpaper.com/img89/zjvvrfxqujoblxuzomsi.jpg) to try with a static image, and no luck I'm using openCV 2.4.12 – [MrE](https://stackoverflow.com/users/2494262/mre" \o "11,585 reputation) [Jan 8 '16 at 0:35](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57081115_20805153)

  @MrE I can detect 3 faces for the image you provided using scaleFactor=1.05, minNeighbors=3, minSize=30. Are you sure you have successfully loaded the haarcascades model files, e.g. haarcascade\_frontalface\_alt.xml or haarcascade\_profileface.xml? – [herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao" \o "41,956 reputation) [Jan 8 '16 at 1:20](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57081842_20805153)

 I have no error in creating the cascade detector with face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml') but I just see there are 2 method signatures in the Python binding: Python: cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]) → objects Python: cv2.CascadeClassifier.detectMultiScale(image, rejectLevels, levelWeights[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]]) → objects and when using minSize=30 as you show, it complains that rejectLevels is needed – [MrE](https://stackoverflow.com/users/2494262/mre" \o "11,585 reputation) [Jan 8 '16 at 1:30](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57081979_20805153)

* if i use minSize=(30,30) it runs but detects nothing – [MrE](https://stackoverflow.com/users/2494262/mre" \o "11,585 reputation) [Jan 8 '16 at 1:32](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57082004_20805153)
* @MrE I am afraid that I cannot help you in this case as I have no experience using the Python API (C++ APIs instead). I suggest you open a new question for your specific case. Good luck. – [herohuyongtao](https://stackoverflow.com/users/2589776/herohuyongtao" \o "41,956 reputation) [Jan 8 '16 at 1:34](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57082042_20805153)

Thanks.. I found the problem: I had find and to give it the absolute path to the model file. – [MrE](https://stackoverflow.com/users/2494262/mre" \o "11,585 reputation) [Jan 8 '16 at 5:23](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57085641_20805153)

* Note that cascades based on Haar-like features (like the ones mentioned above) should not be using image pyramids -- if they do the implementation is broken. The beauty of Haar-like feature cascades is that they can be evaluated at any size on a single fixed-resolution image in constant time, giving high performance. In this case the scaleFactor is used to scale the detection window up, not scale the input image down. – [Chungzuwalla](https://stackoverflow.com/users/3667352/chungzuwalla" \o "611 reputation) [Jul 18 '18 at 4:32](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment89758766_20805153)

## Answer 2

If you have a good CPU and RAM performance or more you can set scaleFactor=1 minNeighbors=3

if you're working in an embedded system like in raspberry i recommand to choose smth like scaleFactor= 2, (Higher values means less accuracy) minNeighbors = 1, (Higher values means less accuracy but more reliability) the algorithm will run much faster otherwise it will freeze if the CPU performance and RAM are not enough .

[The Beast](https://stackoverflow.com/users/3530803/the-beast)

ScaleFactor should always be greater than 1, something like 1.05 for good CPU and RAM. – [saurabheights](https://stackoverflow.com/users/1874627/saurabheights" \o "2,684 reputation) [Jan 14 '16 at 12:33](https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters#comment57322618_31636414)

############################################################################################

# How to force detectMultiScale search on 1:1 scale?

<https://databasefaq.com/index.php/answer/92092/opencv-computer-vision-object-detection-how-to-force-detectmultiscale-search-on-11-scale>

**Question:**

How to force OpenCV [CascadeClassifier::detectMultiScale](http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html) function search only on 1:1 scale?

How many scales used by default?

**Answer:**

Number of scales used in CascadeClassifier::detectMultiScale depends on image size, original trained window size, minObjectSize, maxObjectSize and scaleFactor parameters. It loops through all the scales starting with 1 in increments of scaleFactor until one of the conditions:

* current window size is larger image size
* current window size is larger maxObjectSize

So there are several possibilities to reduce number of scales used in `CascadeClassifier::detectMultiScale:

1. Set maxObjectSize parameter equal to original trained size. It guaranties that cascade will use only 1:1 scale.
2. Set scaleFactor parameter to extremely large value (1000 for example). Thus next scale after 1 will not be used since window size is much larger than image size. It's dirty hack as for me.

Please be sure that you tune minNeighbors parameter. If you would use only one scale you will get very few candidates, so to detect something you need you must decrease this parameter.

############################################################################################

# W[hat should be the best value of “scaleFactor” to detect face in OpenCV library](https://stackoverflow.com/questions/35308142/what-should-be-the-best-value-of-scalefactor-to-detect-face-in-opencv-library)

<https://stackoverflow.com/questions/35308142/what-should-be-the-best-value-of-scalefactor-to-detect-face-in-opencv-library?noredirect=1&lq=1>

I am using following values to detect face using OpenCv, but result is not accurate for all images

faces = faceCascade.detectMultiScale(

gray,

scaleFactor=1.1,

minNeighbors=5,

minSize=(30, 30),

flags = cv2.cv.CV\_HAAR\_SCALE\_IMAGE

)

**Answer 1**

For higher accuracy, I would use a scaleFactor = 1.05. i.e 5% increment in each iteration.

 Typical face detectors have a training size of 25x25 afair (look at the xml file of your detector). So to increase this by 1 pixel for the first step you would need scale factor of 1.04 . However I dont know whether scaling is n\*scale or scale^n for iteration n. – [Micka](https://stackoverflow.com/users/2393191/micka" \o "15,345 reputation) [Feb 10 '16 at 6:37](https://stackoverflow.com/questions/35308142/what-should-be-the-best-value-of-scalefactor-to-detect-face-in-opencv-library?noredirect=1&lq=1#comment58326450_35308142)

 Try by changing minNeighbours = 3 – [Sagar Pate](https://stackoverflow.com/users/5022365/sagar-patel" \o "760 reputation)

############################################################################################

# [OpenCV detectMultiScale() minNeighbors parameter](https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter)

<https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter>

I'm currently using Haar classifiers, to detect objects. On my way, I didn't understand what is the minNeighbors parameter, what is it representing? Actually I don't understand what are the neighbors of the detection candidate rectangle. Please can anybody define the neighboring idea?

Have a look at this..<http://stackoverflow.com/questions/20801015/opencv-detectmultiscale-parameters> – [Naren](https://stackoverflow.com/users/2256349/naren" \o "1,982 reputation) [Mar 7 '14 at 12:08](https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter#comment33790706_22249579)

**Answer 1**

Haar cascade classifier works with a sliding window approach. If you look at the cascade files you can see a size parameter which usually a pretty small value like 20 20. This is the smallest window that cascade can detect. So by applying a sliding window approach, you slide a window through out the picture than you resize it and search again until you can not resize it further. So with every iteration haar's cascaded classifier true outputs are stored. So when this window is slided in picture resized and slided again; it actually detects many many false positives. You can check what it detects by giving **minNeighbors 0**. So an example here :



So there are a lot of face detection because of resizing the sliding window and a lot of false positives too. So to eliminate false positives and get the proper face rectangle out of detections, neighborhood approach is applied. It is like if it is in neighborhood of other rectangles than it is ok, you can pass it further. So this number determines the how much neighborhood is required to pass it as a face rectangle. In the same image when it is **1** :



So by increasing this number you can eliminate false positives but be careful, by increasing it you can also lose true positives too. When it is **3** a perfect result :



[yutasrobot](https://stackoverflow.com/users/1607104/yutasrobot)

1,23599 silver badges1919 bronze badges

But why in case of setting minNeighbors to 1 it does't select the marked region? [link] ([drive.google.com/file/d/0Bwc6G0Ct0fzFeURXbWJHdUswYlk/…](https://drive.google.com/file/d/0Bwc6G0Ct0fzFeURXbWJHdUswYlk/edit?usp=sharing)) – [blakeO](https://stackoverflow.com/users/3392513/blakeo" \o "339 reputation) [Mar 11 '14 at 19:17](https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter#comment33942273_22250382)

* I don't know the full details of neighborhood approach, but idea is somehow similar. Maybe overlapping is required to select as a neighbor. – [yutasrobot](https://stackoverflow.com/users/1607104/yutasrobot" \o "1,235 reputation) [Mar 12 '14 at 20:30](https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter#comment33992857_22250382)

**Answer 2**

From [OpenCV documentation](http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html#cascadeclassifier-detectmultiscale):

**minNeighbors** – Parameter specifying how many neighbors each candidate rectangle should have to retain it.

In other words, this parameter will affect the quality of the detected faces. Higher value results in less detections but with higher quality.

The idea behind this parameter is that the detector will run on a **multiple scale style** and at the same time following **sliding window strategy**. After this step, it will give you multiple responses even for a single face region. This parameter tends to filter these responses just like by setting up a lower-bound threshold, i.e. it will only be counted as a valid face if the number of responses for this face is higher than minNeighbors.

############################################################################################

# [OpenCV `detectMultiScale()` neighbours algorithm](https://stackoverflow.com/questions/31099224/opencv-detectmultiscale-neighbours-algorithm)

<https://stackoverflow.com/questions/31099224/opencv-detectmultiscale-neighbours-algorithm?noredirect=1&lq=1>

I have an implemented face detector of my own, but I faced a problem recently that OpenCV solves with the minNeighbors parameter for detectMultiScale() function: I have a lot of false positives. What OpenCV does is leaves only rectangles that have a certain amount of rectangles nearby. Faces usually have a lot of detections around them. This is [an example](https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter).

# Answer 1

you can use the OpenCV implementation by calling the function groupRectangles() and pass your own list of rectangles.

<http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html#grouprectangles>

############################################################################################

# [Meaning of Parameters of detectMultiScale(a, b, c)](https://stackoverflow.com/questions/51132674/meaning-of-parameters-of-detectmultiscalea-b-c)

<https://stackoverflow.com/questions/51132674/meaning-of-parameters-of-detectmultiscalea-b-c?noredirect=1&lq=1>

I am trying to detect multiple objects through a camera. The objects are Face, eyes, spoon, pen. Spoon and Pen are particular i.e. it should only detect the Pen and Spoon that I have trained it with. But it detects all the kind of faces and eyes as I have used the '.xml' file for face and eye detection that comes with OpenCV-Python.

There is a line in my code below which says detectMultiScale(gray, 1.3, 10). Now, I used the documentation and still couldn't clearly understand the last two parameters of the bracket.

My question:

1. Is it just a matter of trial and error adjusting these last two parameters or can one know how to change them according to the images?
2. According to me these two are really significant and make the code very sensitive as it affects false positives. How do I set them properly to reduce false positives ?

**Answer 1**

*Is it just a matter of trial and error adjusting these last two parameters or can one know how to change them according to the images?*

There is some trial and error in fine-tuning, but you should understand all the parameters and choose initial values which give a good level of performance. Then you can use some kind of automatic method for fine-tuning (i.e., iteratively re-train and re-test with different parameter values and see if detection improves or worsens, but be careful of overfitting). Since the parameters form a large multi-dimensional space, finding good parameters randomly is not practical.

Looking at the Python OpenCV bindings, it appears the two numeric parameters you use are scaleFactor and minNeighbors respectively. There is a good explanation of minNeighbours on this question: [OpenCV detectMultiScale() minNeighbors parameter](https://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter). Setting it higher should reduce your false positives, as described there.

The scaleFactor parameter determines a trade-off between detection accuracy and speed. The detection window starts out at size minSize, and after testing all windows of that size, the window is scaled up by scaleFactor and re-tested, and so on until the window reaches or exceeds maxSize. If scaleFactor is large (eg. 2.0), of course there will be fewer steps, so detection is faster, but you may miss objects whose size is in between two tested scales. But Haar-like features are inherently robust to some small variation in scale, so there's no need to make scaleFactor very small (eg. 1.001); that just wastes time with needless steps. That is why the default is 1.3 and not something smaller.

Setting minSize and maxSize is also important to maximise detection speed. Don't test windows that are smaller or larger than the size range you expect given your setup. So you should specify those in your call.

To be honest, I don't see Haar cascade classifiers being that good for detecting pens or spoons in unknown orientations (if that is your use case). Pens are long and thin which is poorly suited to a square detection window. You may have more success with LINEMOD for example.

According to me these two are really significant and make the code very sensitive as it affects false positives. How do I set them properly to reduce false positives ?

While your false negative rate and speed are OK, don't play with scaleFactor, instead work on improving your training data to reduce your high false positive rate. If speed falls to unacceptable levels while doing that (because the cascade grows to include too many classifier stages), revisit scaleFactor.

############################################################################################

#############################################################################################

## Traversing Directories and Processing Files - os.walk explained

A common programming task is walking a directory tree and processing files in the tree. Let’s explore how the built-in Python function os.walk() can be used to do this. os.walk() is used to generate filename in a directory tree by walking the tree either top-down or bottom-up. For the purposes of this section, we’ll be manipulating the following directory tree:

.

|

├── folder\_1/

| ├── file1.py

| ├── file2.py

| └── file3.py

|

├── folder\_2/

| ├── file4.py

| ├── file5.py

| └── file6.py

|

├── test1.txt

└── test2.txt

The following is an example that shows you how to list all files and directories in a directory tree using os.walk().

os.walk() defaults to traversing directories in a top-down manner:

# Walking a directory tree and printing the names of the directories and files

for dirpath, dirnames, files in os.walk('.'):

print(f'Found directory: {dirpath}')

for file\_name in files:

print(file\_name)

os.walk() returns three values on each iteration of the loop:

1. The name of the current folder
2. A list of folders in the current folder
3. A list of files in the current folder

On each iteration, it prints out the names of the subdirectories and files it finds:

Found directory: .

test1.txt

test2.txt

Found directory: ./folder\_1

file1.py

file3.py

file2.py

Found directory: ./folder\_2

file4.py

file5.py

file6.py

To traverse the directory tree in a bottom-up manner, pass in a topdown=False keyword argument to os.walk():

for dirpath, dirnames, files in os.walk('.', topdown=False):

print(f'Found directory: {dirpath}')

for file\_name in files:

print(file\_name)

Passing the topdown=False argument will make os.walk() print out the files it finds in the subdirectories first:

Found directory: ./folder\_1

file1.py

file3.py

file2.py

Found directory: ./folder\_2

file4.py

file5.py

file6.py

Found directory: .

test1.txt

test2.txt

As you can see, the program started by listing the contents of the subdirectories before listing the contents of the root directory. This is very useful in situations where you want to recursively delete files and directories. You will learn how to do this in the sections below. By default, os.walk does not walk down into symbolic links that resolve to directories. This behavior can be overridden by calling it with a followlinks=True argument.