cv::CascadeClassifier Class Reference

Object Detection

Cascade classifier class for object detection. More...

```
#include "objdetect.hpp"
```

Public Member Functions

	CascadeClassifier ()
	CascadeClassifier (const **String** &filename) Loads a classifier from a file. More...
	~CascadeClassifier ()
void	**detectMultiScale** (**InputArray** image, std::vector< **Rect** > &objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, **Size** minSize=**Size**(), **Size** maxSize=**Size**()) Detects objects of different sizes in the input image. The detected objects are returned as a list of rectangles. More...
void	**detectMultiScale** (**InputArray** image, std::vector< **Rect** > &objects, std::vector< int > &numDetections, double scaleFactor=1.1, int minNeighbors=3, int flags=0, **Size** minSize=**Size**(), **Size** maxSize=**Size**())
void	**detectMultiScale** (**InputArray** image, std::vector< **Rect** > &objects, std::vector< int > &rejectLevels, std::vector< double > &levelWeights, double scaleFactor=1.1, int minNeighbors=3, int flags=0, **Size** minSize=**Size**(), **Size** maxSize=**Size**(), bool outputRejectLevels=false)
bool	**empty** () const Checks whether the classifier has been loaded. More...
int	**getFeatureType** () const
Ptr< **BaseCascadeClassifier::MaskGenerator** >	**getMaskGenerator** ()
void *	**getOldCascade** ()
Size	**getOriginalWindowSize** () const
bool	**isOldFormatCascade** () const
bool	**load** (const **String** &filename) Loads a classifier from a file. More...

bool	**read** (const **FileNode** &node) Reads a classifier from a **FileStorage** node. More...
void	**setMaskGenerator** (const **Ptr**< **BaseCascadeClassifier::MaskGenerator** > &maskGenerator)

Static Public Member Functions

static bool	**convert** (const **String** &oldcascade, const **String** &newcascade)

Public Attributes

Ptr< **BaseCascadeClassifier** >	**cc**

Detailed Description

Cascade classifier class for object detection.

Examples:

facedetect.cpp.

Constructor & Destructor Documentation

§ CascadeClassifier() [1/2]

cv::CascadeClassifier::CascadeClassifier ( )

Python:

<CascadeClassifier object> = cv.CascadeClassifier( )

<CascadeClassifier object> = cv.CascadeClassifier(filename)

§ CascadeClassifier() [2/2]

```
cv::CascadeClassifier::CascadeClassifier ( const String &  filename )
```

Python:

```
<CascadeClassifier object> = cv.CascadeClassifier(          )
<CascadeClassifier object> = cv.CascadeClassifier( filename )
```

Loads a classifier from a file.

Parameters

filename Name of the file from which the classifier is loaded.

§ ~CascadeClassifier()

```
cv::CascadeClassifier::~CascadeClassifier ( )
```

Member Function Documentation

§ convert()

```
static bool cv::CascadeClassifier::convert ( const String &  oldcascade,
                                             const String &  newcascade
                                           )
```

static

Python:

```
retval = cv.CascadeClassifier_convert( oldcascade, newcascade )
```

§ detectMultiScale() [1/3]

```
void cv::CascadeClassifier::detectMultiScale ( InputArray            image,
                                               std::vector< Rect > & objects,
                                               double                scaleFactor = 1.1,
                                               int                   minNeighbors = 3,
                                               int                   flags = 0,
                                               Size                  minSize = Size(),
                                               Size                  maxSize = Size()
                                             )
```

Python:

```
objects                              = cv.CascadeClassifier.detectMultiScale(  image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]  )
objects, numDetections               = cv.CascadeClassifier.detectMultiScale2( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]  )
objects, rejectLevels, levelWeights  = cv.CascadeClassifier.detectMultiScale3( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]] )
```

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

Parameters

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, the rectangles may be partially outside the original image.
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. If `maxSize == minSize` model is evaluated on single scale.

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/python/facedetect.py

Examples:

facedetect.cpp.

§ detectMultiScale() `[2/3]`

```
void cv::CascadeClassifier::detectMultiScale ( InputArray              image,
                                               std::vector< Rect > &   objects,
                                               std::vector< int > &    numDetections,
                                               double                  scaleFactor = 1.1,
                                               int                     minNeighbors = 3,
                                               int                     flags = 0,
                                               Size                    minSize = Size(),
                                               Size                    maxSize = Size()
                                             )
```

Python:

```
objects                               = cv.CascadeClassifier.detectMultiScale(  image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]                        )
objects, numDetections                = cv.CascadeClassifier.detectMultiScale2( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]                        )
objects, rejectLevels, levelWeights   = cv.CascadeClassifier.detectMultiScale3( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]] )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

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, the rectangles may be partially outside the original image.
numDetections	Vector of detection numbers for the corresponding objects. An object's number of detections is the number of neighboring positively classified rectangles that were joined together to form the 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. If `maxSize == minSize` model is evaluated on single scale.

§ detectMultiScale() [3/3]

```
void cv::CascadeClassifier::detectMultiScale ( InputArray               image,
                                               std::vector< Rect > &    objects,
                                               std::vector< int > &     rejectLevels,
                                               std::vector< double > &  levelWeights,
                                               double                   scaleFactor = 1.1,
                                               int                      minNeighbors = 3,
                                               int                      flags = 0,
                                               Size                     minSize = Size(),
                                               Size                     maxSize = Size(),
                                               bool                     outputRejectLevels = false
                                             )
```

Python:

```
objects                          = cv.CascadeClassifier.detectMultiScale(  image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]                          )
objects, numDetections           = cv.CascadeClassifier.detectMultiScale2( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]                          )
objects, rejectLevels, levelWeights = cv.CascadeClassifier.detectMultiScale3( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]] )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. This function allows you to retrieve the final stage decision certainty of classification. For this, one needs to set `outputRejectLevels` on true and provide the `rejectLevels` and `levelWeights` parameter. For each resulting detection, `levelWeights` will then contain the certainty of classification at the final stage. This value can then be used to separate strong from weaker classifications.

A code sample on how to use it efficiently can be found below:

```
Mat img;
vector<double> weights;
vector<int> levels;
vector<Rect> detections;
CascadeClassifier model("/path/to/your/model.xml");
model.detectMultiScale(img, detections, levels, weights, 1.1, 3, 0, Size(), Size(), true);
cerr << "Detection " << detections[0] << " with weight " << weights[0] << endl;
```

§ empty()

bool cv::CascadeClassifier::empty ( ) const

Python:

retval = cv.CascadeClassifier.empty( )

Checks whether the classifier has been loaded.

Examples:

facedetect.cpp.

§ getFeatureType()

int cv::CascadeClassifier::getFeatureType ( ) const

Python:

retval = cv.CascadeClassifier.getFeatureType( )

§ getMaskGenerator()

Ptr<**BaseCascadeClassifier::MaskGenerator**> cv::CascadeClassifier::getMaskGenerator ( )

§ getOldCascade()

void* cv::CascadeClassifier::getOldCascade ( )

§ getOriginalWindowSize()

Size cv::CascadeClassifier::getOriginalWindowSize ( ) const

Python:

retval = cv.CascadeClassifier.getOriginalWindowSize( )

§ isOldFormatCascade()

bool cv::CascadeClassifier::isOldFormatCascade ( ) const

Python:

retval = cv.CascadeClassifier.isOldFormatCascade( )

§ load()

bool cv::CascadeClassifier::load (const **String** & filename)

Python:

retval = cv.CascadeClassifier.load(filename)

Loads a classifier from a file.

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.

Examples:

facedetect.cpp.

§ read()

```
bool cv::CascadeClassifier::read ( const FileNode &  node )
```

Python:

```
retval = cv.CascadeClassifier.read( node )
```

Reads a classifier from a **FileStorage** node.

Note

The file may contain a new cascade classifier (trained traincascade application) only.

§ setMaskGenerator()

```
void cv::CascadeClassifier::setMaskGenerator ( const Ptr< BaseCascadeClassifier::MaskGenerator > &  maskGenerator )
```

Member Data Documentation

§ cc

```
Ptr<BaseCascadeClassifier> cv::CascadeClassifier::cc
```

The documentation for this class was generated from the following file:

- objdetect/include/opencv2/**objdetect.hpp**