LEARN OPENCV BY EXAMPLES

OpenCV simplified for beginners by the use of examples. Learn OpenCV with basic implementation of different algorithms.

Face Detection using Haar-Cascade Classifier

class **CascadeClassifier** - Cascade classifier class for object detection

CascadeClassifier::CascadeClassifier(const string& filename) // Constructor - Loads a classifier from a file

```
CascadeClassifier face_cascade( "C:/OpenCV243/data/Haarcascades/ha
```

bool CascadeClassifier::empty() const // Checks whether the classifier has been loaded.bool CascadeClassifier::load(const string& filename) // Loads a classifier from a file

```
CascadeClassifier face_cascade;
face_cascade.load( "C:/OpenCV243/data/Haarcascades/haarcascade_front
```

bool CascadeClassifier::read(const FileNode& node) // Reads a classifier from a FileStorage node

void CascadeClassifier::**detectMultiScale**(const Mat& image, vector<Rect>& objects, double

SEARCH CONTENTS OF THIS BLOG

Search

POPULAR POSTS

1 Find Contour

2 Basic drawing examples

3 Line Detection by Hough Line Transform

Face Detection using Haar-Cascade Classifier

Perspective Transform

Sobel Edge Detection

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.

Parameters:

- **cascade** – Haar classifier cascade (OpenCV 1.x API only). It can be loaded from XML or YAML file using Load(). 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.

bool CascadeClassifier::**setImage**(Ptr<FeatureEvaluator>& feval, const Mat& image)
// Sets an image for detection
Parameters:

- **cascade** – Haar classifier cascade (OpenCV 1.x API only). See `CascadeClassifier::detectMultiScale()` for more information.
- **feval** – Pointer to the feature evaluator used for computing features.
- **image** – Matrix of the type `CV_8UC1` containing an image where the features are computed

int CascadeClassifier::**runAt**(Ptr<FeatureEvaluator>& feval, Point pt, double& weight)
// Runs the detector at the specified point. The function returns 1 if the cascade classifier detects an object in the given location. Otherwise, it returns negated index of the stage at which the candidate has been rejected.
Parameters:

- **cascade** – Haar classifier cascade (OpenCV 1.x API only). See `CascadeClassifier::detectMultiScale()` for more information.
- **feval** – Feature evaluator used for computing features.
- **pt** – Upper left point of the window where the features are computed. Size of the window is equal to the size of training images.

Steps:

Kalman Filter Implementation (Tracking mouse position)

Histogram Calculation

9 OpenCV example to convert RGB to gray / other color spaces

10 Hough Circle Detection

CATEGORIES

- Accessory
- Applications
- Basics
- Edge Detection
- Feature Extraction
- Filter
- Miscellaneous
- Morphological Operation

1. Read the image.
2. Load Face cascade (CascadeClassifier > load)
3. Detect faces (detectMultiScale)
4. Draw circles on the detected faces (ellipse)
5. Show the result.

Functions:
ellipse, detectMultiScale, imshow, imread, namedWindow, waitKey.

Example:

```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>
#include <stdio.h>

using namespace std;
using namespace cv;

int main( )
{
    Mat image;
    image = imread("lena.jpg", CV_LOAD_IMAGE_COLOR);
    namedWindow( "window1", 1 );   imshow( "window1", image );

    // Load Face cascade (.xml file)
    CascadeClassifier face_cascade;
    face_cascade.load( "C:/OpenCV243/data/Haarcascades/haarcascade_

    // Detect faces
    std::vector<Rect> faces;
    face_cascade.detectMultiScale( image, faces, 1.1, 2, 0|CV_HAAR_

    // Draw circles on the detected faces
    for( int i = 0; i < faces.size(); i++ )
    {
        Point center( faces[i].x + faces[i].width*0.5, faces[i].y +
        ellipse( image, center, Size( faces[i].width*0.5, faces[i].
    }

    imshow( "Detected Face", image );

    waitKey(0);
    return 0;
}
```