# Simple Digit Recognition OCR in OpenCV-Python

I am trying to implement a "Digit Recognition OCR" in OpenCV-Python (cv2). ( It is just for learning purposes ). I would like to learn both KNearest and SVM features in OpenCV.

I have 100 samples(images) of each digit. I would like to train with them.

There is a sample letter_recog.py that comes with OpenCV sample. But i still couldn't figure out on how to use it. I don't understand what are the samples, responses etc. Also, it loads a txt file at first, which i didn't understand first.

Later on searching a little bit, i could find a letter_recognition.data in cpp samples. I used it and made a code for cv2.KNearest in the model of letter_recog.py ( just for testing):

```
import numpy as np
import cv2

fn = 'letter-recognition.data'
a = np.loadtxt(fn, np.float32, delimiter=',', converters={ 0 : lambda ch : ord(ch)-
ord('A') })
samples, responses = a[:,1:], a[:,0]

model = cv2.KNearest()
retval = model.train(samples,responses)
retval, results, neigh_resp, dists = model.find_nearest(samples, k = 10)
print results.ravel()
```

It gave me an array of size 20000, i don't understand what it is.

**Questions:**

1) What is letter_recognition.data file ? How to build that file from my own data set?

2) What does results.reval() denote?

3) How we can write a simple digit recognition tool using letter_recognition.data file ( either KNearest or SVM)?

python opencv numpy computer-vision ocr

edited Jan 28 '13 at 4:55

asked Feb 23 '12 at 12:37

Abid Rahman K
**19.9k** 8 58 96

## 2 Answers

Well, I decided to workout myself on my question to solve above problem. What i wanted is to implement a simpl OCR using KNearest or SVM features in OpenCV. And below is what i did and how. ( it is just for learning how to use KNearest for simple OCR purposes).

**1)** My first question was about letter_recognition.data file that comes with OpenCV samples. I wanted to know what is inside that file.

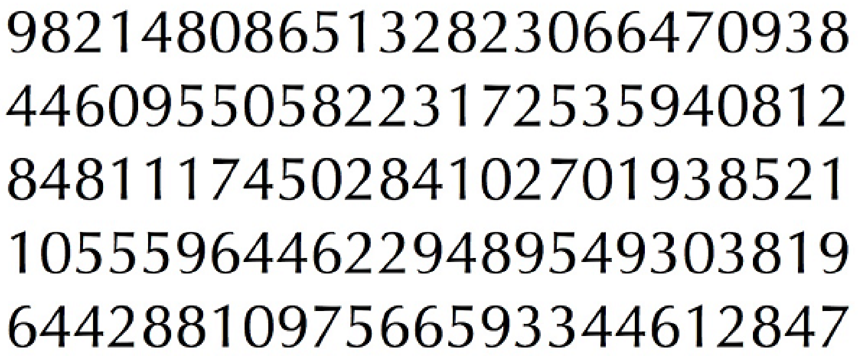
It contains a letter, along with 16 features of that letter.

And `this SOF` helped me to find it. These 16 features are explained in the paper `Letter Recognition Using Holland-Style Adaptive Classifiers` . ( Although i didn't understand some of the features at end)

**2)** Since i knew, without understanding all those features, it is difficult to do that method. i tried some other papers, but all were a little difficult for a beginner.

`So i just decided to take all the pixel values as features.` (I was not worried about accuracy or performance, i just wanted it to work, atleast with least accuracy)

I took below image for my training data:



9821480865132823066470938
4460955058223172535940812
8481117450284102701938521
1055596446229489549303819
6442881097566593344612847

( I know the amount of training data is less. But, since all letters are of same font and size, i decided to try on this).

**To prepare the data for training, i made a small code in OpenCV. It does following things:**

a) It loads the image.

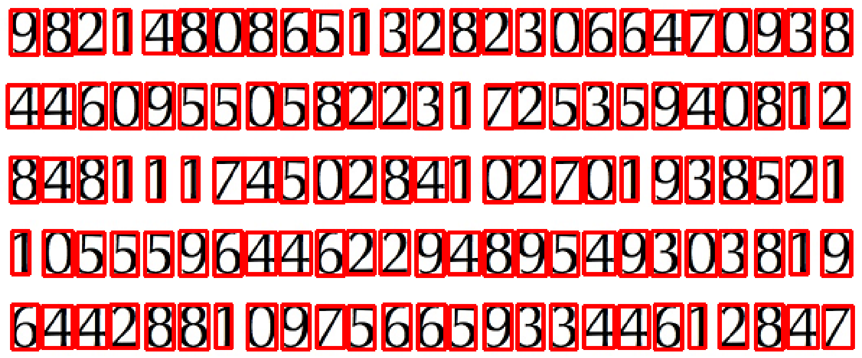
b) Selects the digits ( obviously by contour finding and applying constraints on area and height of letters to avoid false detections).

c) Draws the bounding rectangle around one letter and wait for key press. This time we press the digit key corresponding to the letter in box.

d) Once corresponding digit key is pressed, it resizes this box to 10x10 and saves 100 pixel values in an array (here, samples) and corresponding manually entered digit in another array(here, responses).

e) Then save both the arrays in seperate txt files.

At the end of manual classification of digits, image will look like below:



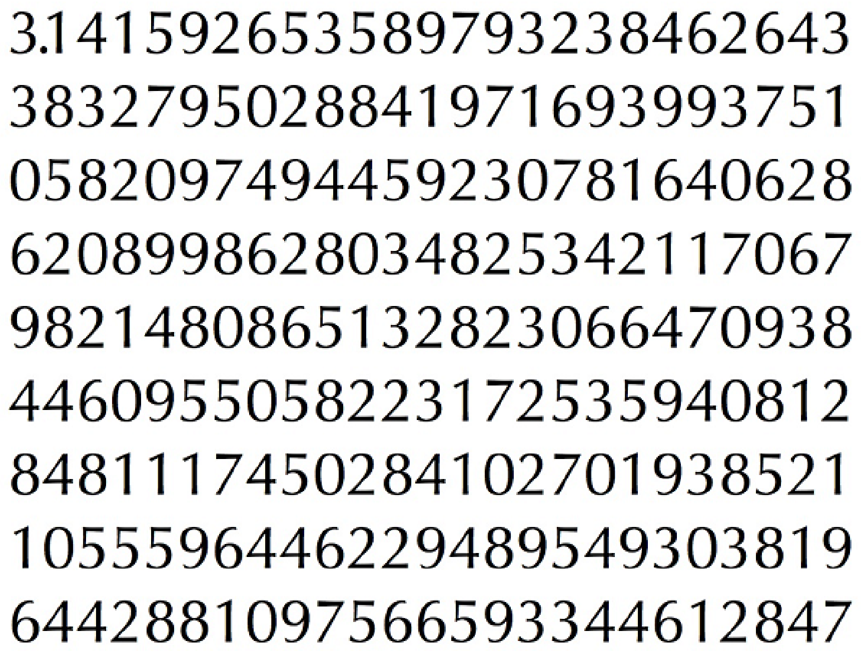
9821480865132823066470938
4460955058223172535940812
8481117450284102701938521
1055596446229489549303819
6442881097566593344612847

Below is the code i used for above purpose ( of course, not so clean):

```
import sys

import numpy as np
import cv2

im = cv2.imread('pitrain.png')
im3 = im.copy()

gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),0)
thresh = cv2.adaptiveThreshold(blur,255,1,1,11,2)

#################      Now finding Contours         ###################

contours,hierarchy = cv2.findContours(thresh,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)

samples =  np.empty((0,100))
responses = []
keys = [i for i in range(48,58)]

for cnt in contours:
    if cv2.contourArea(cnt)>50:
        [x,y,w,h] = cv2.boundingRect(cnt)

        if  h>28:
            cv2.rectangle(im,(x,y),(x+w,y+h),(0,0,255),2)
            roi = thresh[y:y+h,x:x+w]
            roismall = cv2.resize(roi,(10,10))
            cv2.imshow('norm',im)
            key = cv2.waitKey(0)

            if key == 27:  # (escape to quit)
                sys.exit()
            elif key in keys:
                responses.append(int(chr(key)))
                sample = roismall.reshape((1,100))
                samples = np.append(samples,sample,0)

responses = np.array(responses,np.float32)
responses = responses.reshape((responses.size,1))
print "training complete"

np.savetxt('generalsamples.data',samples)
np.savetxt('generalresponses.data',responses)
```

**Now we enter in to training and testing part.**

For testing part i used below image, which has same type of letters i used to train.



3.141592653589793238462643
3832795028841971693993751
0582097494459230781640628
6208998628034825342117067
9821480865132823066470938
4460955058223172535940812
8481117450284102701938521
1055596446229489549303819
6442881097566593344612847

**For training we do as follows**:

a) Load the txt files we already saved earlier

b) create a instance of classifier we are using ( here, it is KNearest)

c) Then we use KNearest.train function to train the data

**For testing purposes, we do as follows:**

a) We load the image used for testing

b) process the image as earlier and extract each digit using contour methods

c) Draw bounding box for it, then resize to 10x10, and store its pixel values in an array as done earlier.

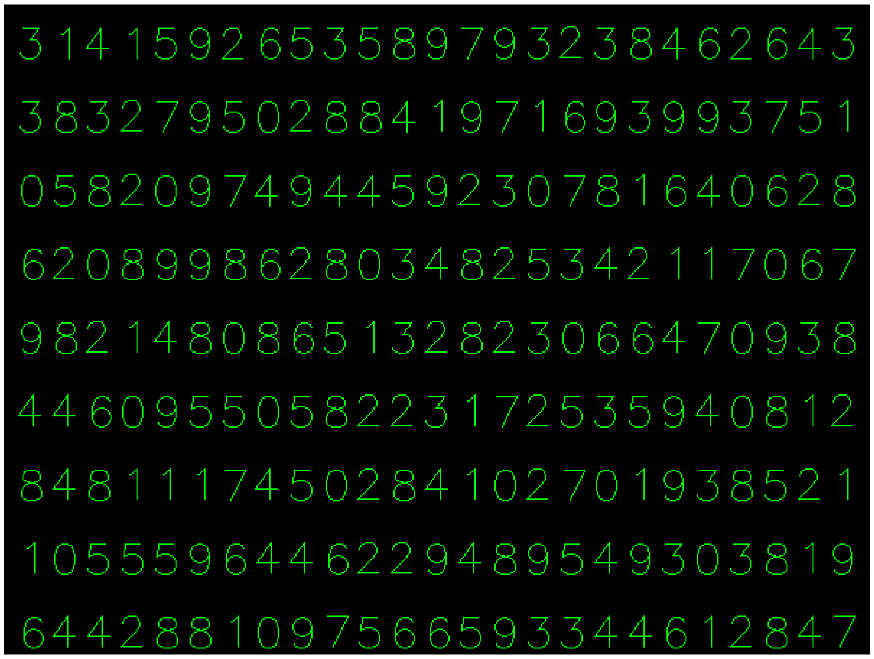
d) Then we use KNearest.find_nearest() function to find the nearest item to the one we gave. ( If lucky, it recognises the correct digit.)

I included last two steps ( training and testing) in single code below:

```
import cv2
import numpy as np

#######   training part    ###############
samples = np.loadtxt('generalsamples.data',np.float32)
responses = np.loadtxt('generalresponses.data',np.float32)
responses = responses.reshape((responses.size,1))
```

```
model = cv2.KNearest()
model.train(samples,responses)

############################# testing part  #########################

im = cv2.imread('pi.png')
out = np.zeros(im.shape,np.uint8)
gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
thresh = cv2.adaptiveThreshold(gray,255,1,1,11,2)

contours,hierarchy = cv2.findContours(thresh,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)

for cnt in contours:
    if cv2.contourArea(cnt)>50:
        [x,y,w,h] = cv2.boundingRect(cnt)
        if  h>28:
            cv2.rectangle(im,(x,y),(x+w,y+h),(0,255,0),2)
            roi = thresh[y:y+h,x:x+w]
            roismall = cv2.resize(roi,(10,10))
            roismall = roismall.reshape((1,100))
            roismall = np.float32(roismall)
            retval, results, neigh_resp, dists = model.find_nearest(roismall, k = 1)
            string = str(int((results[0][0])))
            cv2.putText(out,string,(x,y+h),0,1,(0,255,0))

cv2.imshow('im',im)
cv2.imshow('out',out)
cv2.waitKey(0)
```

And it worked , below is the result i got:



```
3 14 1592 65358979323846264 3
3832795028841971693993751
0582097494459230781640628
6208998628034825342117067
982 1480865132823066470938
4460955058223172535940812
8481117450284102701938521
1055596446229489549303819
6442881097566593344612847
```

```
Here it worked with 100% accuracy, for which the reason, i assume, is all digits are of same
kind and same size.
```

But any way, this is a good start to go for beginners ( i hope so).

edited Jan 28 '14 at 23:30
Jesse Aldridge
2,491 2 17 45

answered Mar 8 '12 at 15:35
Abid Rahman K
19.9k 8 58 96

26 +1 Long post, but very educational. This should go to opencv tag info – karlphillip Apr 17 '12 at 17:50

6 in case anyone's interested, I made a proper OO engine from this code, along with some bells and whistles: github.com/goncalopp/simple-ocr-opencv – goncalopp Oct 14 '12 at 3:01

Hi, the google docs link refered by this post doesn't work for me. – Ricardo Nov 12 '12 at 10:34

@Ricardo : edited the link, check if it works, or search for the name of the paper, first link in google turns up. – Abid Rahman K Nov 12 '12 at 11:12

3 Note that there is no need for using SVM and KNN when you have a well defined perfect font. For

instance, the digits 0, 4, 6, 9 form one group, the digits 1, 2, 3, 5, 7 form another, and 8 another. This group is given by the euler number. Then "0" has no endpoints, "4" has two, and "6" and "9" are distinguished by centroid position. "3" is the only one, in the other group, with 3 endpoints. "1" and "7" are distinguished by the skeleton length. When considering the convex hull together with the digit, "5" and "2" have two holes and they can be distinguished by the centroid of largest hole. – mmgp Jan 28 '13 at 5:23

For those who interested in C++ code can refer below code. Thanks **Abid Rahman** for the nice explanation.

The procedure is same as above but, the contour finding uses only first hierarchy level contour, so that the algorithm uses only outer contour for each digit.

## Code for creating sample and Label data

```
//Process image to extract contour
Mat thr,gray,con;
Mat src=imread("digit.png",1);
cvtColor(src,gray,CV_BGR2GRAY);
threshold(gray,thr,200,255,THRESH_BINARY_INV); //Threshold to find contour
thr.copyTo(con);

// Create sample and label data
vector< vector <Point> > contours; // Vector for storing contour
vector< Vec4i > hierarchy;
Mat sample;
Mat response_array;
findContours( con, contours, hierarchy,CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE ); //Find
contour

for( int i = 0; i< contours.size(); i=hierarchy[i][0] ) // iterate through first hierarchy
level contours
{
    Rect r= boundingRect(contours[i]); //Find bounding rect for each contour
    rectangle(src,Point(r.x,r.y), Point(r.x+r.width,r.y+r.height), Scalar(0,0,255),2,8,0);
    Mat ROI = thr(r); //Crop the image
    Mat tmp1, tmp2;
    resize(ROI,tmp1, Size(10,10), 0,0,INTER_LINEAR ); //resize to 10X10
    tmp1.convertTo(tmp2,CV_32FC1); //convert to float
    sample.push_back(tmp2.reshape(1,1)); // Store  sample data
    imshow("src",src);
    int c=waitKey(0); // Read corresponding label for contour from keyoard
    c-=0x30;     // Convert ascii to intiger value
    response_array.push_back(c); // Store label to a mat
    rectangle(src,Point(r.x,r.y), Point(r.x+r.width,r.y+r.height), Scalar(0,255,0),2,8,0);
}

// Store the data to file
Mat response,tmp;
tmp=response_array.reshape(1,1); //make continuous
tmp.convertTo(response,CV_32FC1); // Convert  to float

FileStorage Data("TrainingData.yml",FileStorage::WRITE); // Store the sample data in a
file
Data << "data" << sample;
Data.release();

FileStorage Label("LabelData.yml",FileStorage::WRITE); // Store the label data in a file
Label << "label" << response;
Label.release();
cout<<"Training and Label data created successfully....!! "<<endl;

imshow("src",src);
waitKey();
```

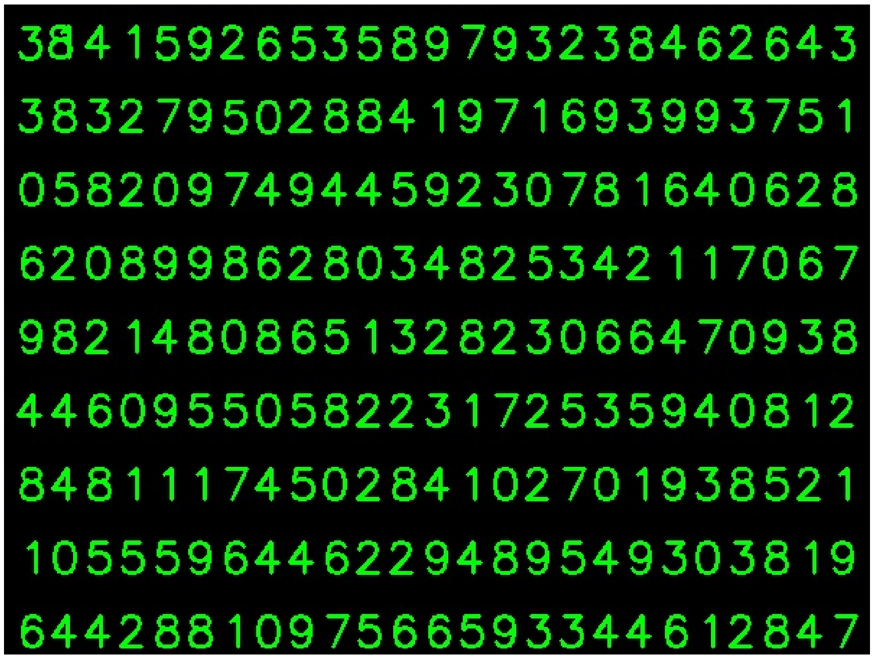
## Code for training and testing

```
Mat thr,gray,con;
Mat src=imread("dig.png",1);
cvtColor(src,gray,CV_BGR2GRAY);
threshold(gray,thr,200,255,THRESH_BINARY_INV); // Threshold to create input
thr.copyTo(con);


// Read stored sample and label for training
Mat sample;
Mat response,tmp;
FileStorage Data("TrainingData.yml",FileStorage::READ); // Read traing data to a Mat
Data["data"] >> sample;
Data.release();

FileStorage Label("LabelData.yml",FileStorage::READ); // Read label data to a Mat
```

```
    Label["label"] >> response;
    Label.release();


    KNearest knn;
    knn.train(sample,response); // Train with sample and responses
    cout<<"Training compleated.....!!"<<endl;

    vector< vector <Point> > contours; // Vector for storing contour
    vector< Vec4i > hierarchy;

    //Create input sample by contour finding and cropping
    findContours( con, contours, hierarchy,CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE );
    Mat dst(src.rows,src.cols,CV_8UC3,Scalar::all(0));

    for( int i = 0; i< contours.size(); i=hierarchy[i][0] ) // iterate through each contour
    for first hierarchy level .
    {
        Rect r= boundingRect(contours[i]);
        Mat ROI = thr(r);
        Mat tmp1, tmp2;
        resize(ROI,tmp1, Size(10,10), 0,0,INTER_LINEAR );
        tmp1.convertTo(tmp2,CV_32FC1);
        float p=knn.find_nearest(tmp2.reshape(1,1), 1);
        char name[4];
        sprintf(name,"%d",(int)p);
        putText( dst,name,Point(r.x,r.y+r.height) ,0,1, Scalar(0, 255, 0), 2, 8 );
    }

    imshow("src",src);
    imshow("dst",dst);
    imwrite("dest.jpg",dst);
    waitKey();
```

## Result

In the result the dot in the first line is detected as 8 and we haven't trained for dot. Also I am considering every contour in first hierarchy level as the sample input, user can avoid it by computing the area.



```
3841592653589793238462643
3832795028841971693993751
0582097494459230781640628
6208998628034825342117067
9821480865132823066470938
4460955058223172535940812
8481117450284102701938521
1055596446229489549303819
6442881097566593344612847
```

edited Sep 28 '14 at 16:16
Schoolboy
3,765 8 36

answered Jan 3 '14 at 11:13
Haris
4,548 1 19 32

I tired to run this code. I was able to create sample and label data. But when i run the test-training file, it runs with an error `*** stack smashing detected ***:` and hence i am not getting a final proper image as you are getting above (digits in green color) – skm Feb 14 '14 at 19:41

i change `char name[4];` in your code to `char name[7];` and i didn't get the stack related error but still i am not getting the correct results. I am getting a image like here < i.imgur.com/qRkV2B4.jpg > – skm Feb 14 '14 at 19:57

@skm Make sure that you are getting number of contour same as the number of digits in the image, also try by printing the result on console. – Haris Feb 20 '14 at 4:48

**protected** by rayryeng Feb 27 at 20:03

Thank you for your interest in this question. Because it has attracted low-quality answers, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?