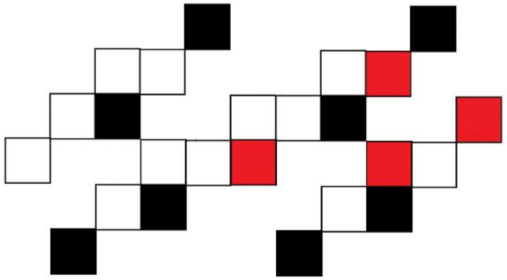
Open CV

Basics and Samples



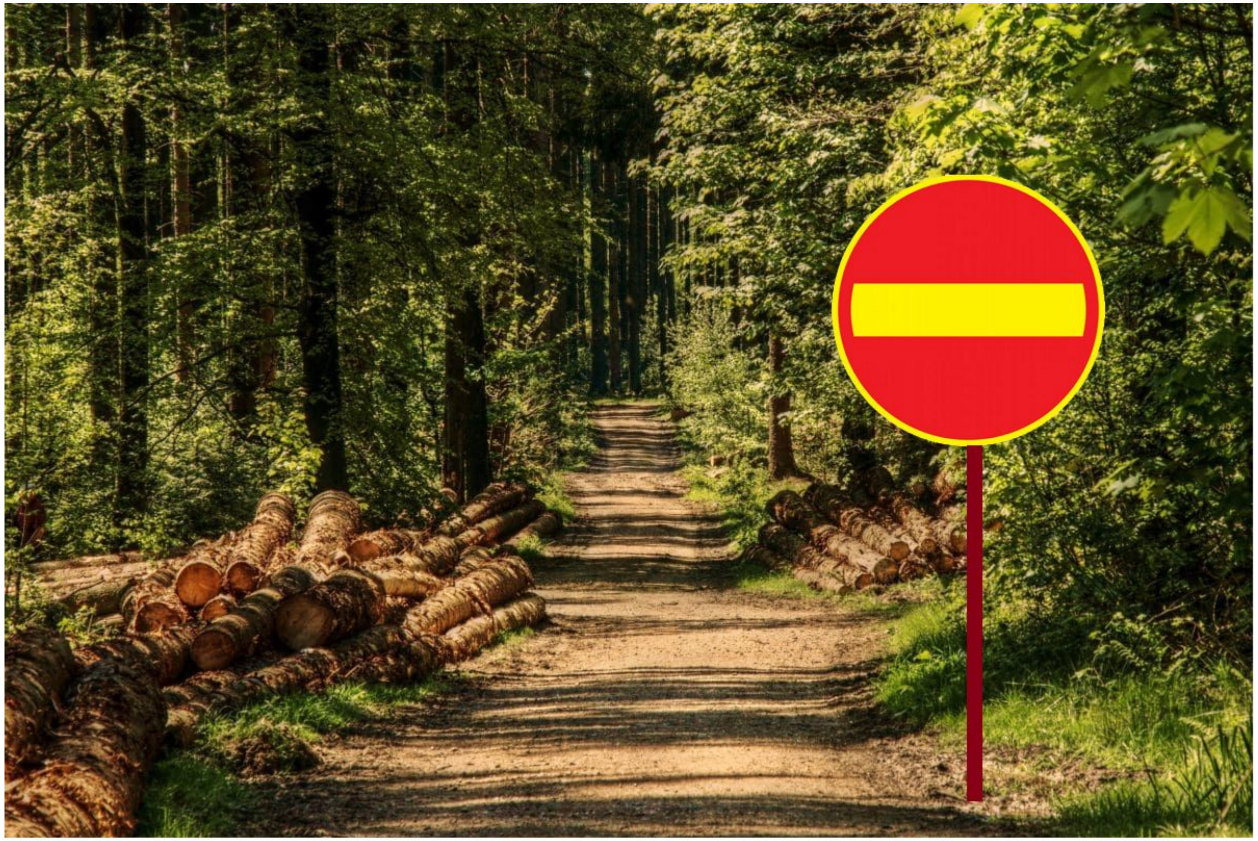
Open CV demonstration

Traffic sign detection

Open CV demonstration

Traffic sign detection

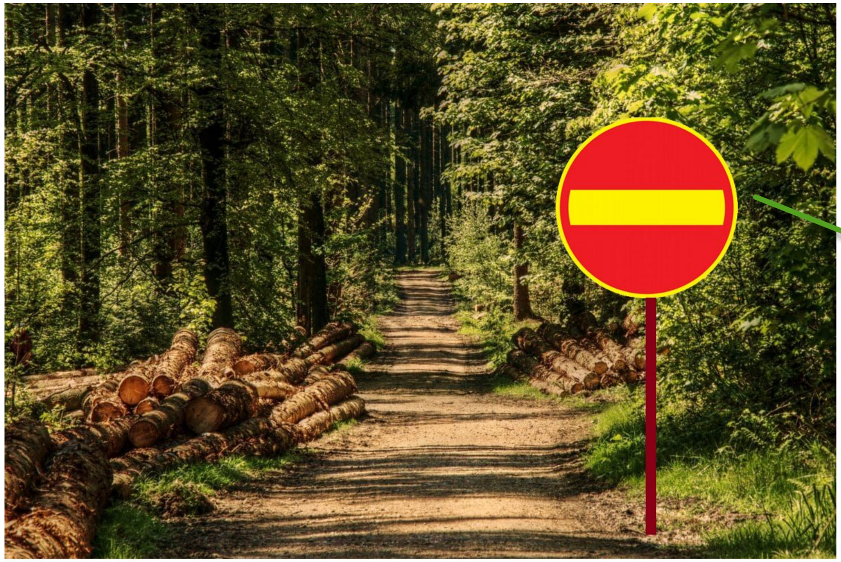
Can computer detect a circle shape from the picture

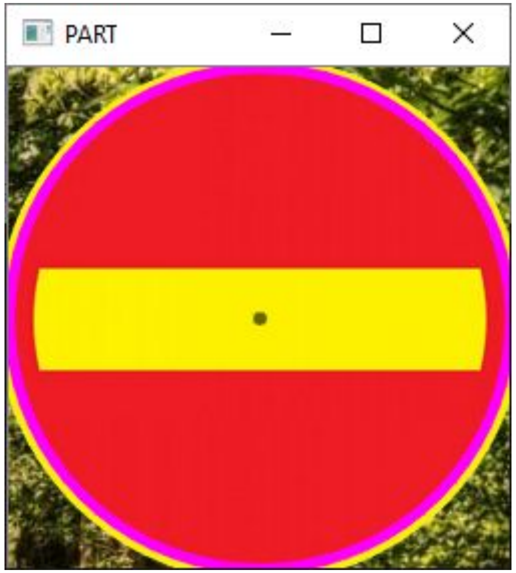


Open CV demonstration

Traffic sign detection

Can computer detect a circle shape from the picture





Now we wanted to detect only a circrle from the left picture:
It was found.
Next step could be to detect colors…

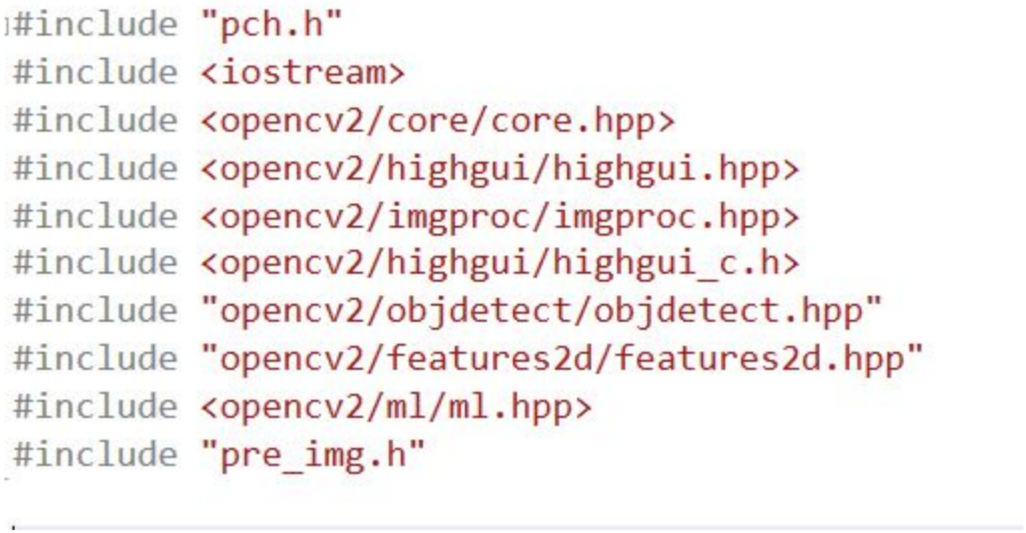
Open CV demonstration

Traffic sign detection

Codes

These header files were added

We have a Visual Studio console app and use C++



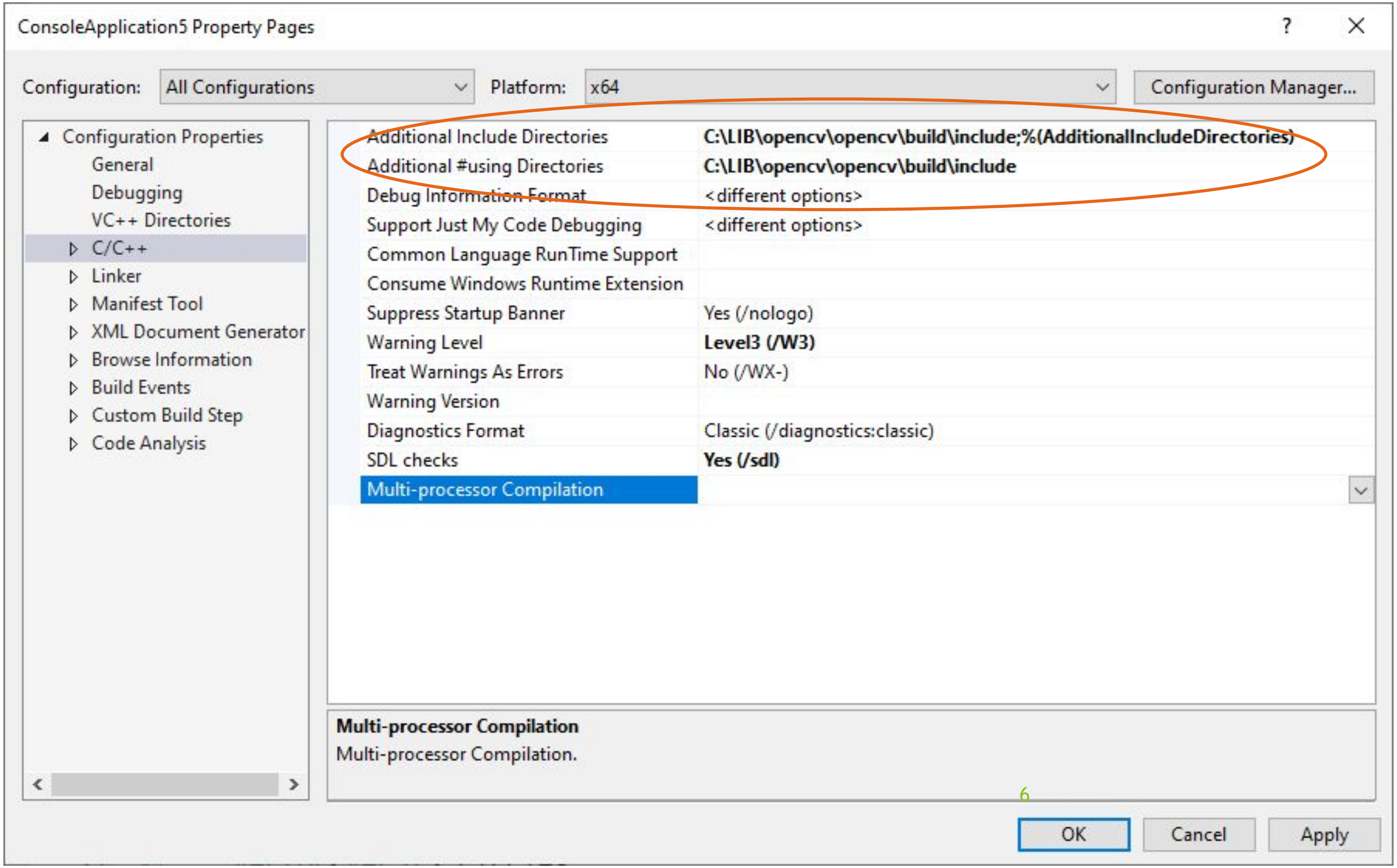
```
#include "pch.h"
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui_c.h>
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/features2d/features2d.hpp"
#include <opencv2/ml/ml.hpp>
#include "pre_img.h"
```

Open CV demonstration

Traffic sign detection

Codes

Take these libraries to VS project



ConsoleApplication5 Property Pages

Configuration: All Configurations

Platform: x64

Additional Include Directories	**C:\LIB\opencv\opencv\build\include;%(AdditionalIncludeDirectories)**
Additional #using Directories	**C:\LIB\opencv\opencv\build\include**
Debug Information Format	<different options>
Support Just My Code Debugging	<different options>
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Classic (/diagnostics:classic)
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	

Multi-processor Compilation
Multi-processor Compilation.

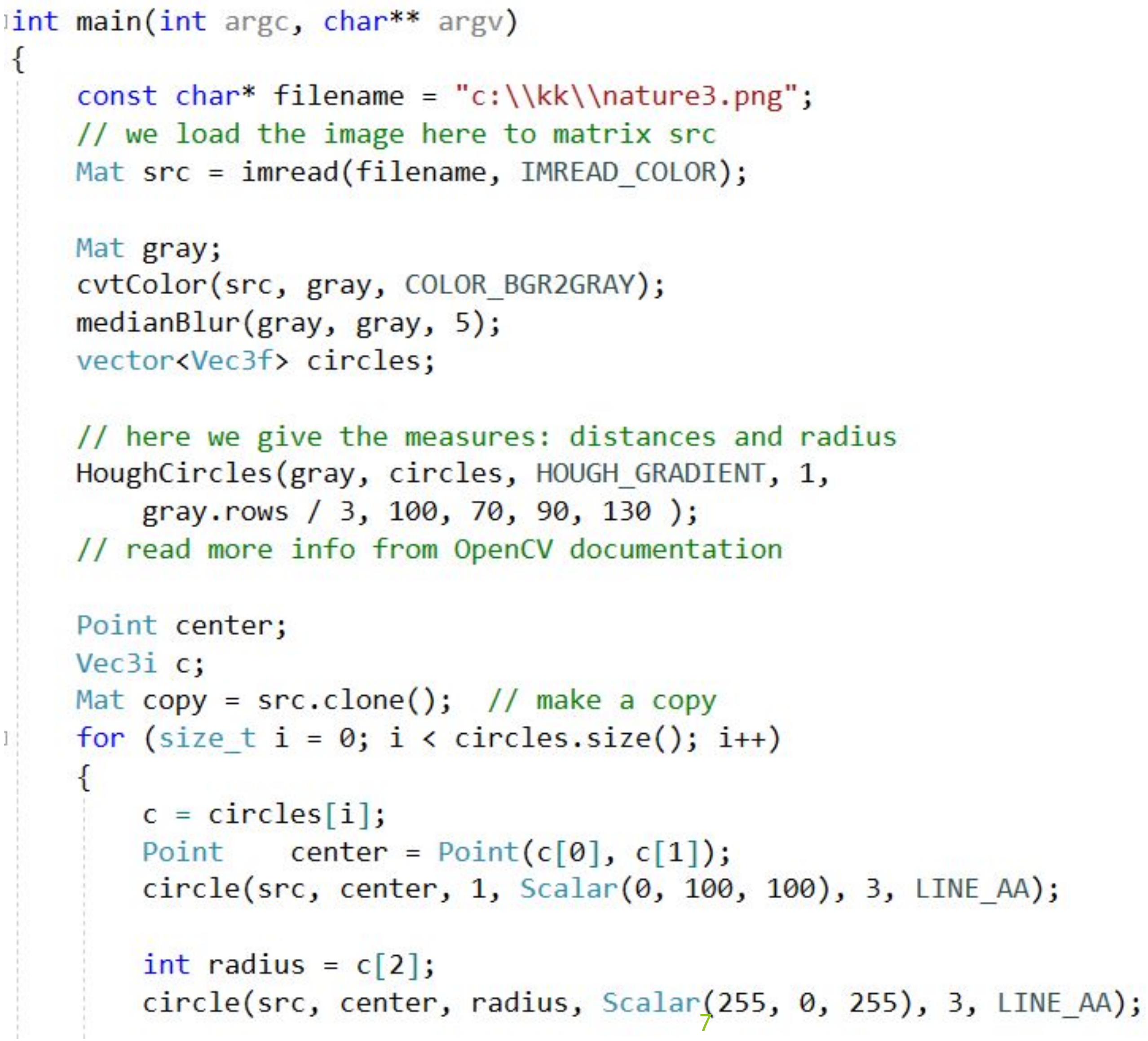
Main(): part 1

Open CV demonstration

Traffic sign detection

Codes

We have a Visual Studio console app and use C++



```cpp
int main(int argc, char** argv)
{
    const char* filename = "c:\\kk\\nature3.png";
    // we load the image here to matrix src
    Mat src = imread(filename, IMREAD_COLOR);

    Mat gray;
    cvtColor(src, gray, COLOR_BGR2GRAY);
    medianBlur(gray, gray, 5);
    vector<Vec3f> circles;

    // here we give the measures: distances and radius
    HoughCircles(gray, circles, HOUGH_GRADIENT, 1,
        gray.rows / 3, 100, 70, 90, 130 );
    // read more info from OpenCV documentation

    Point center;
    Vec3i c;
    Mat copy = src.clone();  // make a copy
    for (size_t i = 0; i < circles.size(); i++)
    {
        c = circles[i];
        Point    center = Point(c[0], c[1]);
        circle(src, center, 1, Scalar(0, 100, 100), 3, LINE_AA);

        int radius = c[2];
        circle(src, center, radius, Scalar(255, 0, 255), 3, LINE_AA);
```

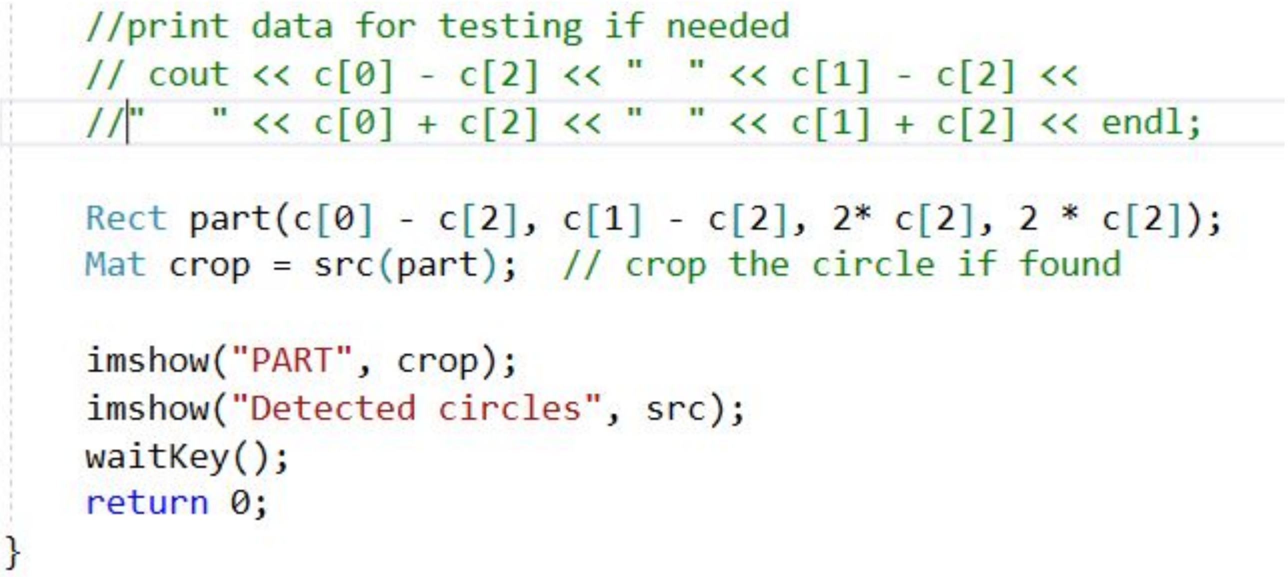
Open CV demonstration

Main(): part 2

Traffic sign detection

Codes

We have a Visual Studio console app and use C++



```
    //print data for testing if needed
    // cout << c[0] - c[2] << "   " << c[1] - c[2] <<
    //"    " << c[0] + c[2] << "   " << c[1] + c[2] << endl;

    Rect part(c[0] - c[2], c[1] - c[2], 2* c[2], 2 * c[2]);
    Mat crop = src(part);  // crop the circle if found

    imshow("PART", crop);
    imshow("Detected circles", src);
    waitKey();
    return 0;
}
```

Coding basics

Try it.

Make it better.

Detect colours ->
get more info
make decisions using that more detailed info!

Open CV demonstration

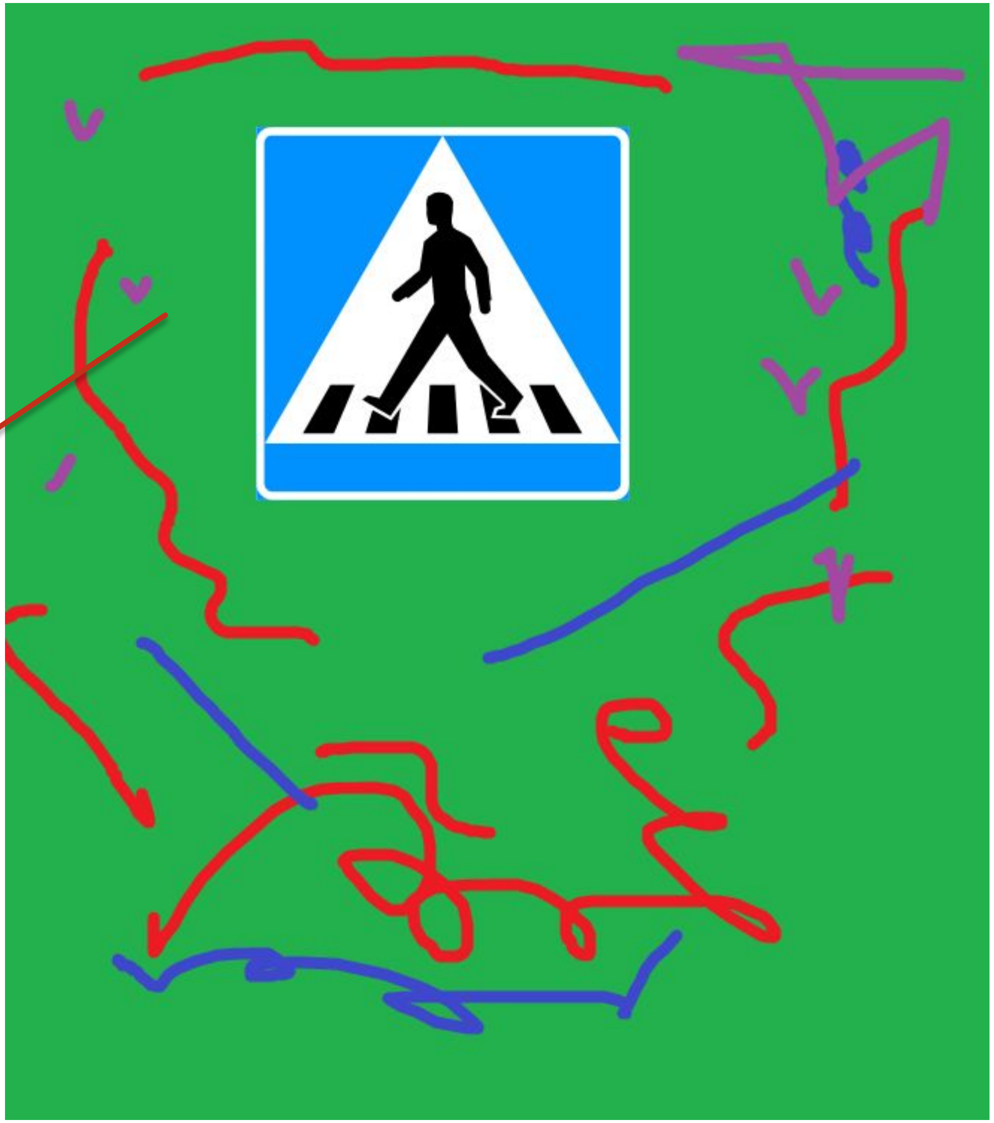
Traffic sign detection, proto 2

Open CV demonstration

Traffic sign detection, proto 2

Official sign

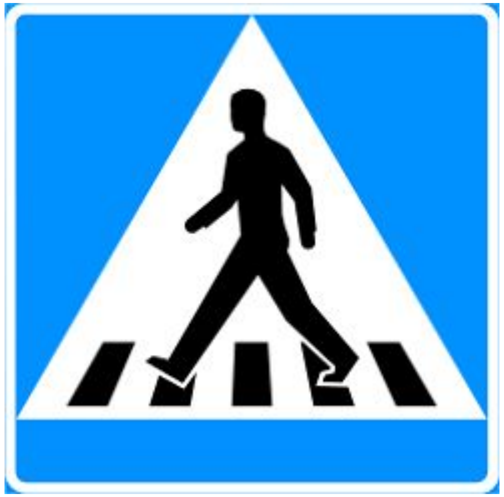
Do we detect It?



Open CV demonstration

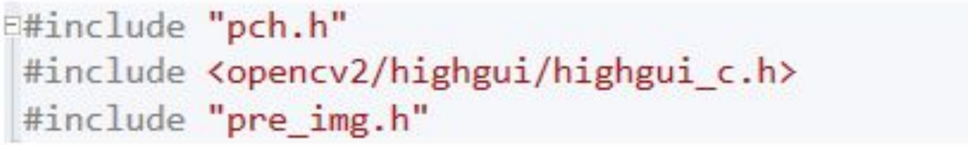
Traffic sign detection, proto 2

Official sign

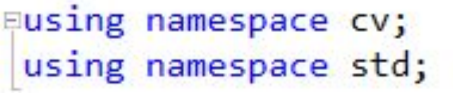


Codes
Source: opencv documentation

These are used in Visual Studio
C++ Console project



```
#include "pch.h"
#include <opencv2/highgui/highgui_c.h>
#include "pre_img.h"
```



```
using namespace cv;
using namespace std;
```

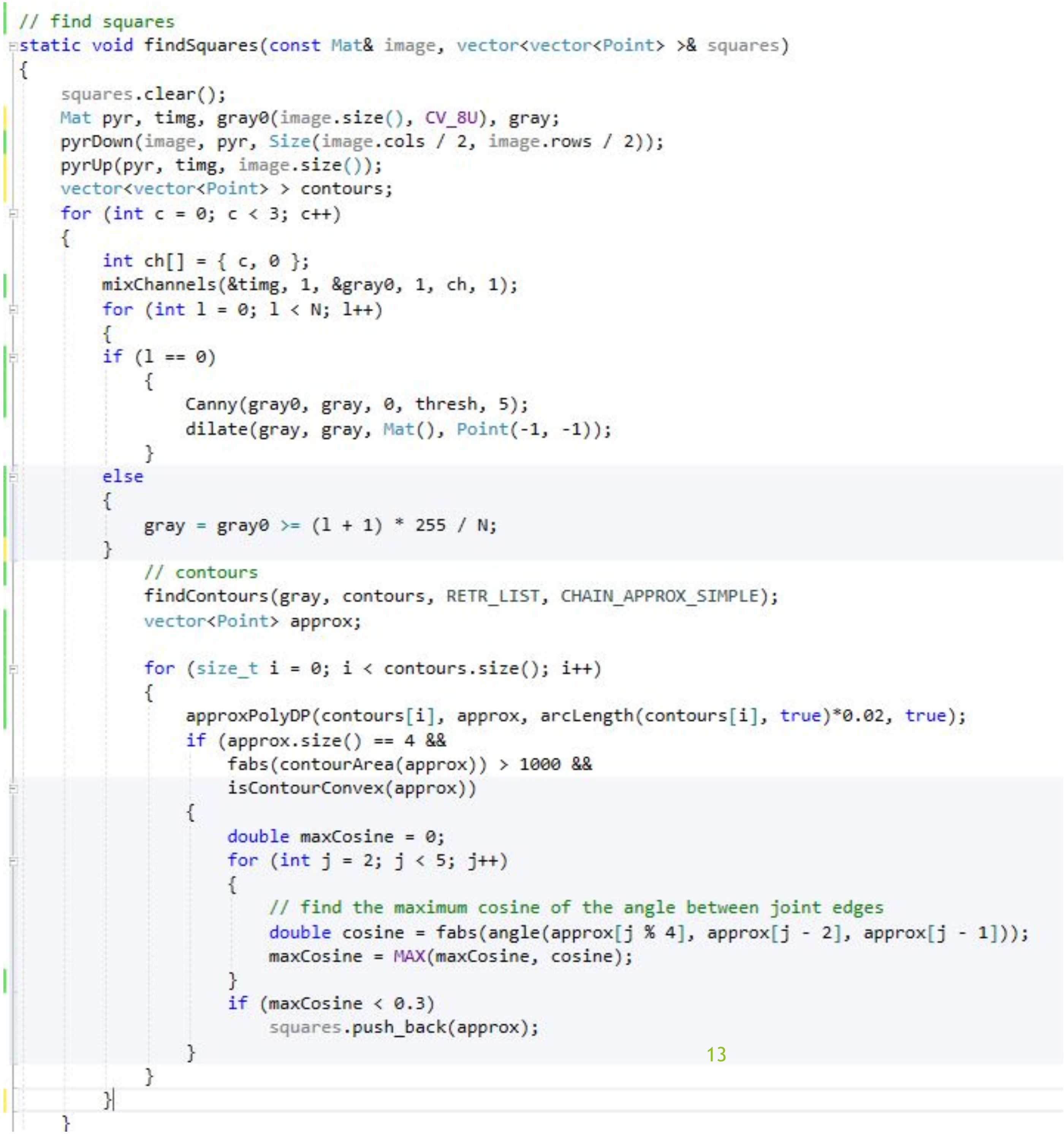
Open CV demonstration

Traffic sign detection, proto 2

Codes

Source: opencv documentation

Function that detects squares



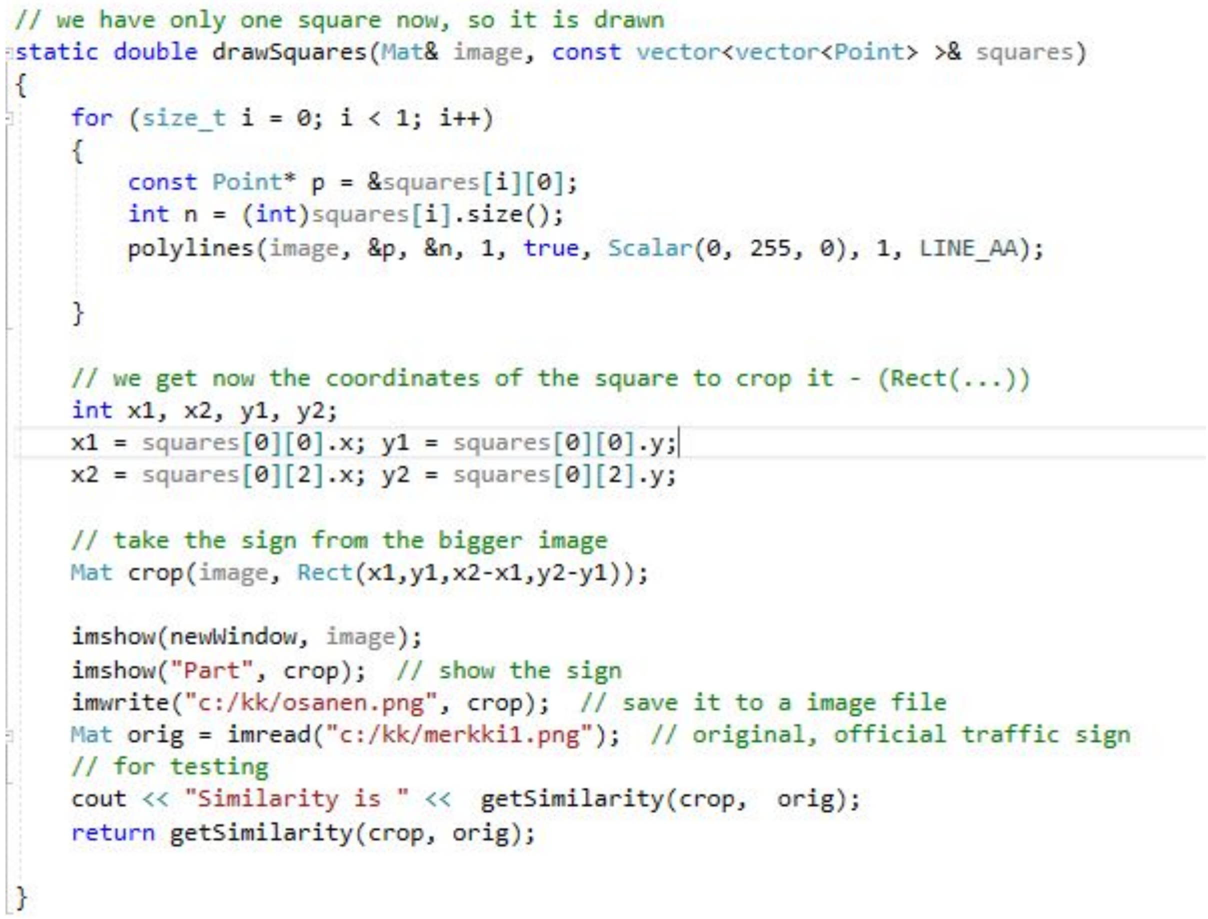
```
// find squares
static void findSquares(const Mat& image, vector<vector<Point> >& squares)
{
    squares.clear();
    Mat pyr, timg, gray0(image.size(), CV_8U), gray;
    pyrDown(image, pyr, Size(image.cols / 2, image.rows / 2));
    pyrUp(pyr, timg, image.size());
    vector<vector<Point> > contours;
    for (int c = 0; c < 3; c++)
    {
        int ch[] = { c, 0 };
        mixChannels(&timg, 1, &gray0, 1, ch, 1);
        for (int l = 0; l < N; l++)
        {
        if (l == 0)
            {
                Canny(gray0, gray, 0, thresh, 5);
                dilate(gray, gray, Mat(), Point(-1, -1));
            }
        else
        {
            gray = gray0 >= (l + 1) * 255 / N;
        }
            // contours
            findContours(gray, contours, RETR_LIST, CHAIN_APPROX_SIMPLE);
            vector<Point> approx;

            for (size_t i = 0; i < contours.size(); i++)
            {
                approxPolyDP(contours[i], approx, arcLength(contours[i], true)*0.02, true);
                if (approx.size() == 4 &&
                    fabs(contourArea(approx)) > 1000 &&
                    isContourConvex(approx))
                {
                    double maxCosine = 0;
                    for (int j = 2; j < 5; j++)
                    {
                        // find the maximum cosine of the angle between joint edges
                        double cosine = fabs(angle(approx[j % 4], approx[j - 2], approx[j - 1]));
                        maxCosine = MAX(maxCosine, cosine);
                    }
                    if (maxCosine < 0.3)
                        squares.push_back(approx);
                }
            }
        }
    }
```

Open CV demonstration

Traffic sign detection, proto 2

Codes
Source: opencv documentation

Crop and draw



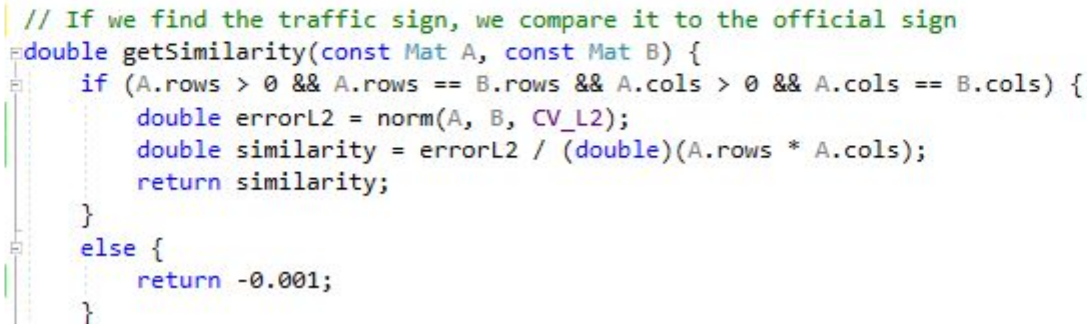
```
// we have only one square now, so it is drawn
static double drawSquares(Mat& image, const vector<vector<Point> >& squares)
{
    for (size_t i = 0; i < 1; i++)
    {
        const Point* p = &squares[i][0];
        int n = (int)squares[i].size();
        polylines(image, &p, &n, 1, true, Scalar(0, 255, 0), 1, LINE_AA);

    }

    // we get now the coordinates of the square to crop it - (Rect(...))
    int x1, x2, y1, y2;
    x1 = squares[0][0].x; y1 = squares[0][0].y;
    x2 = squares[0][2].x; y2 = squares[0][2].y;

    // take the sign from the bigger image
    Mat crop(image, Rect(x1,y1,x2-x1,y2-y1));

    imshow(newWindow, image);
    imshow("Part", crop);  // show the sign
    imwrite("c:/kk/osanen.png", crop);  // save it to a image file
    Mat orig = imread("c:/kk/merkki1.png");  // original, official traffic sign
    // for testing
    cout << "Similarity is " <<  getSimilarity(crop,  orig);
    return getSimilarity(crop, orig);

}
```

Open CV demonstration

Traffic sign detection, proto 2

Codes
Source: opencv documentation

Are images similar



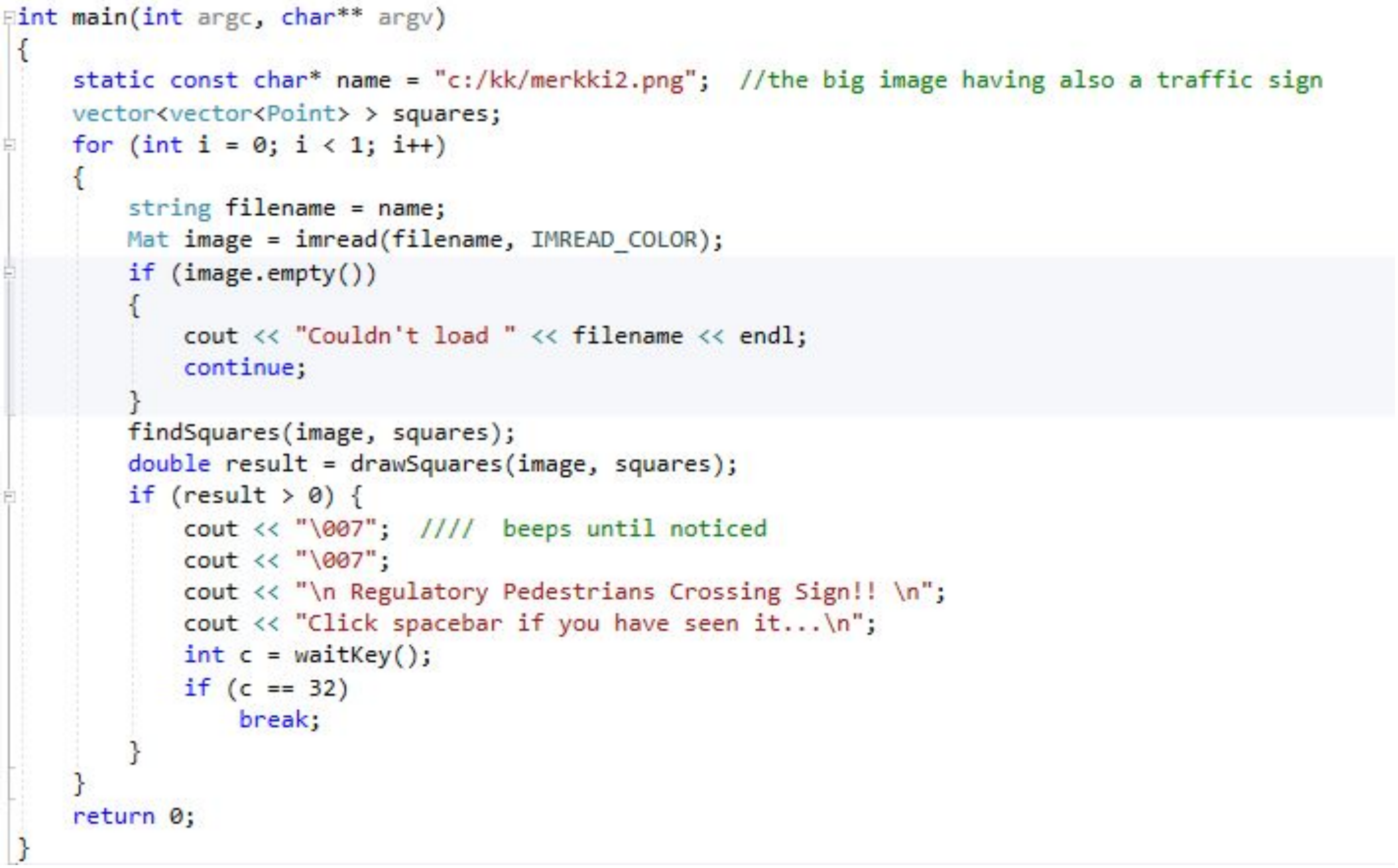
```
// If we find the traffic sign, we compare it to the official sign
double getSimilarity(const Mat A, const Mat B) {
    if (A.rows > 0 && A.rows == B.rows && A.cols > 0 && A.cols == B.cols) {
        double errorL2 = norm(A, B, CV_L2);
        double similarity = errorL2 / (double)(A.rows * A.cols);
        return similarity;
    }
    else {
        return -0.001;
    }
```

Open CV demonstration

Traffic sign detection, proto 2

Function main()

Codes
Source: opencv documentation

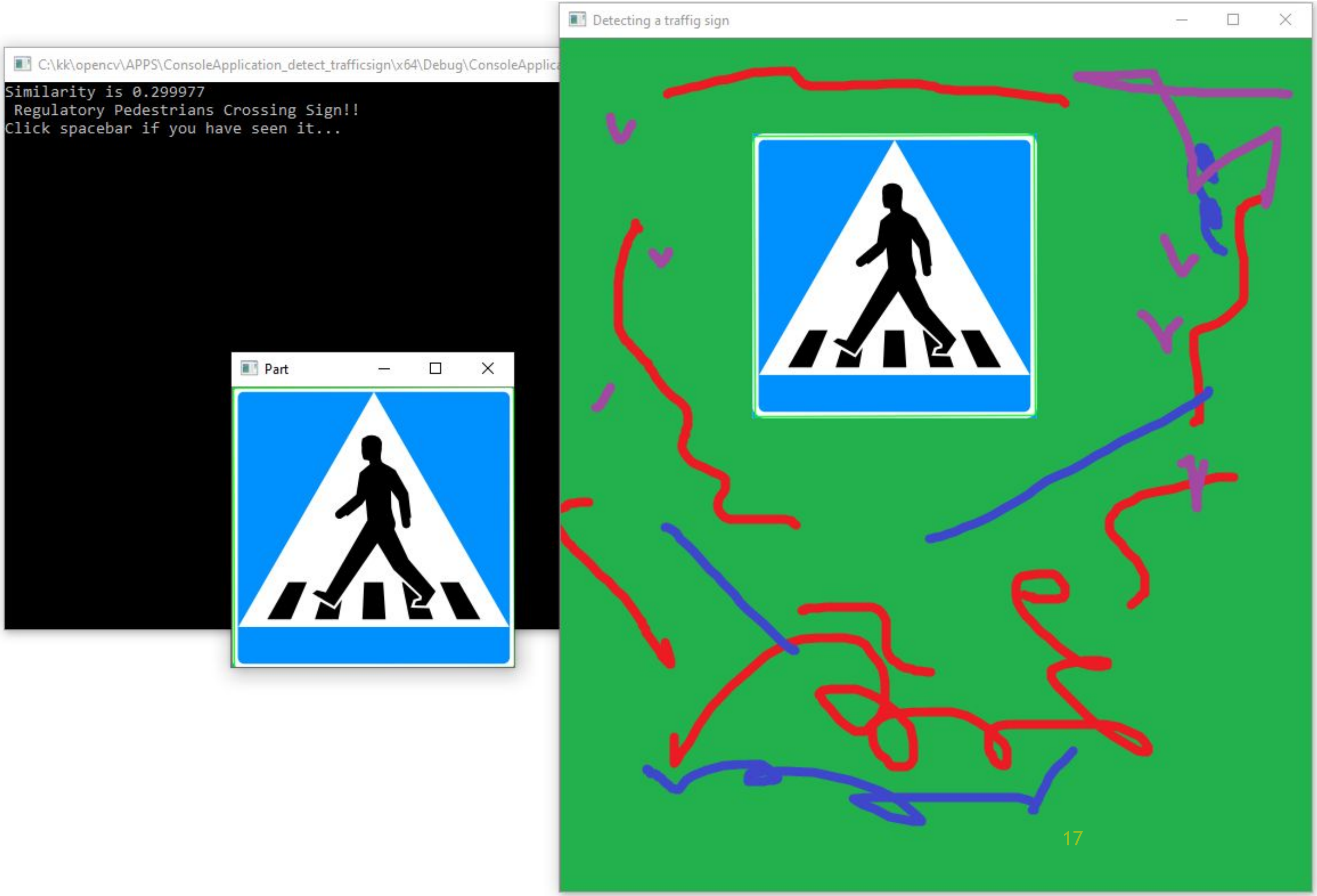


```
int main(int argc, char** argv)
{
    static const char* name = "c:/kk/merkki2.png";  //the big image having also a traffic sign
    vector<vector<Point> > squares;
    for (int i = 0; i < 1; i++)
    {
        string filename = name;
        Mat image = imread(filename, IMREAD_COLOR);
        if (image.empty())
        {
            cout << "Couldn't load " << filename << endl;
            continue;
        }
        findSquares(image, squares);
        double result = drawSquares(image, squares);
        if (result > 0) {
            cout << "\007";  ////  beeps until noticed
            cout << "\007";
            cout << "\n Regulatory Pedestrians Crossing Sign!! \n";
            cout << "Click spacebar if you have seen it...\n";
            int c = waitKey();
            if (c == 32)
                break;
        }
    }
    return 0;
}
```

Open CV demonstration

Test run

Traffic sign detection, proto 2



C:\kk\opencv\APPS\ConsoleApplication_detect_trafficsign\x64\Debug\ConsoleApplic
Part
Detecting a traffig sign

```
Similarity is 0.299977
 Regulatory Pedestrians Crossing Sign!!
Click spacebar if you have seen it...
```

Open CV demonstration

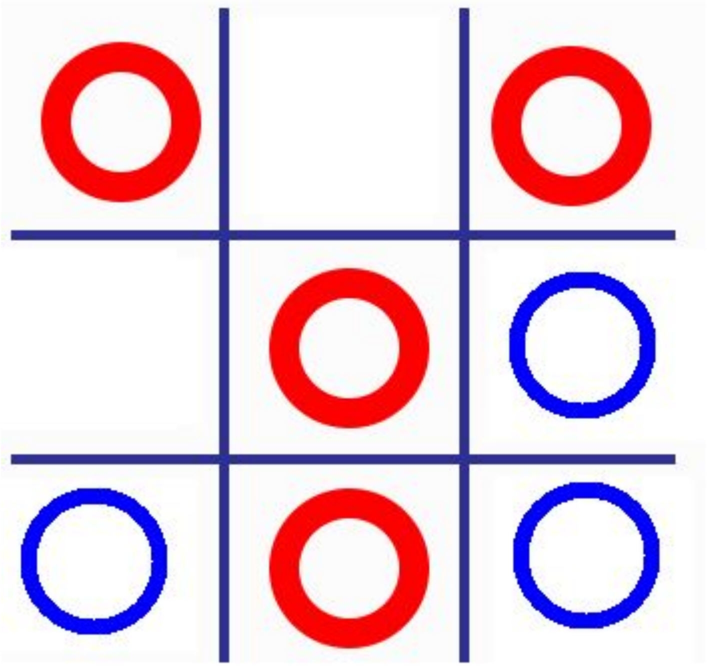
Detect colors

Source:
OpenCV documentation

Open CV demonstration

Detect colors

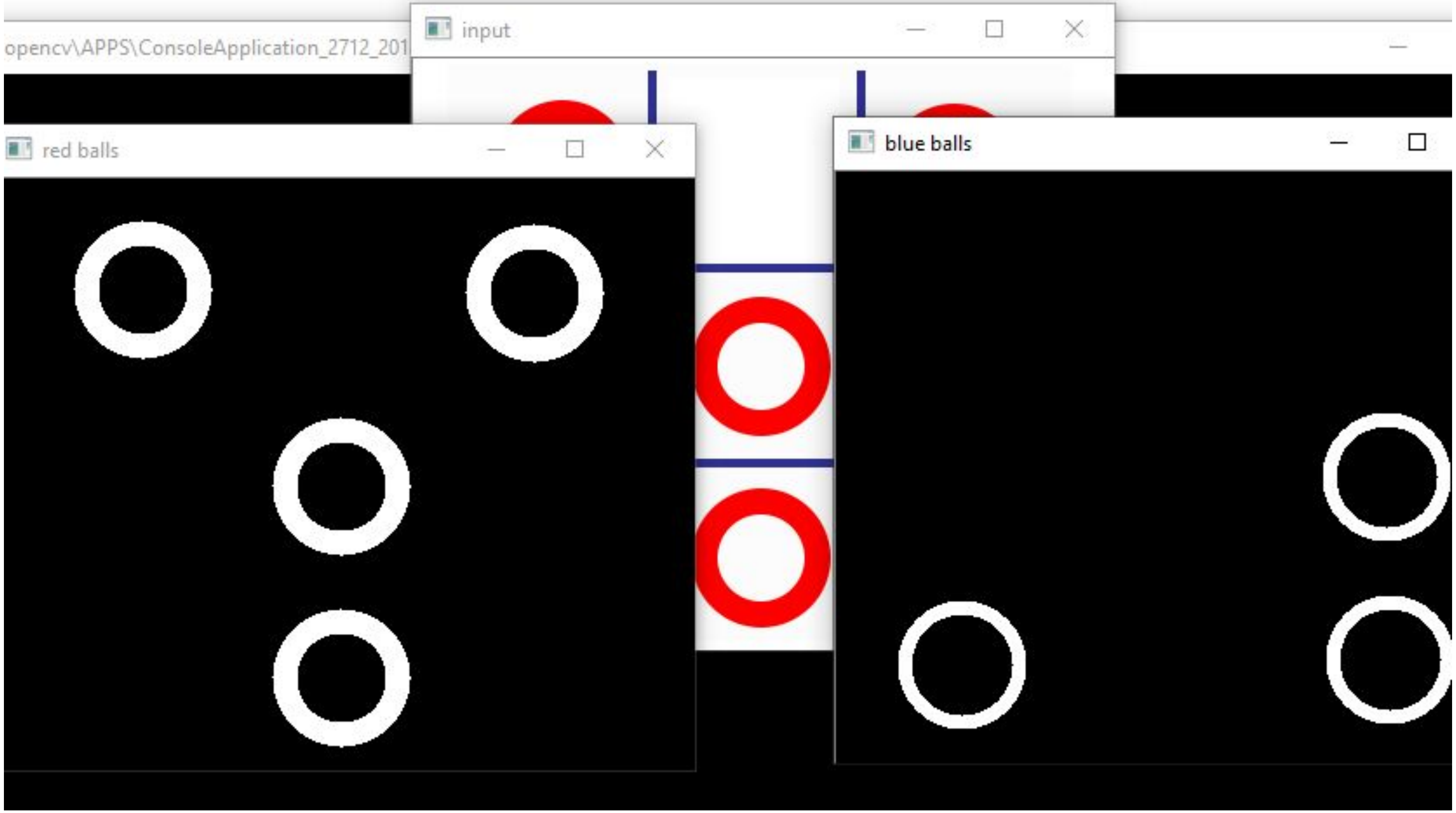
Tictactoe: situation



Open CV demonstration

Detect colors

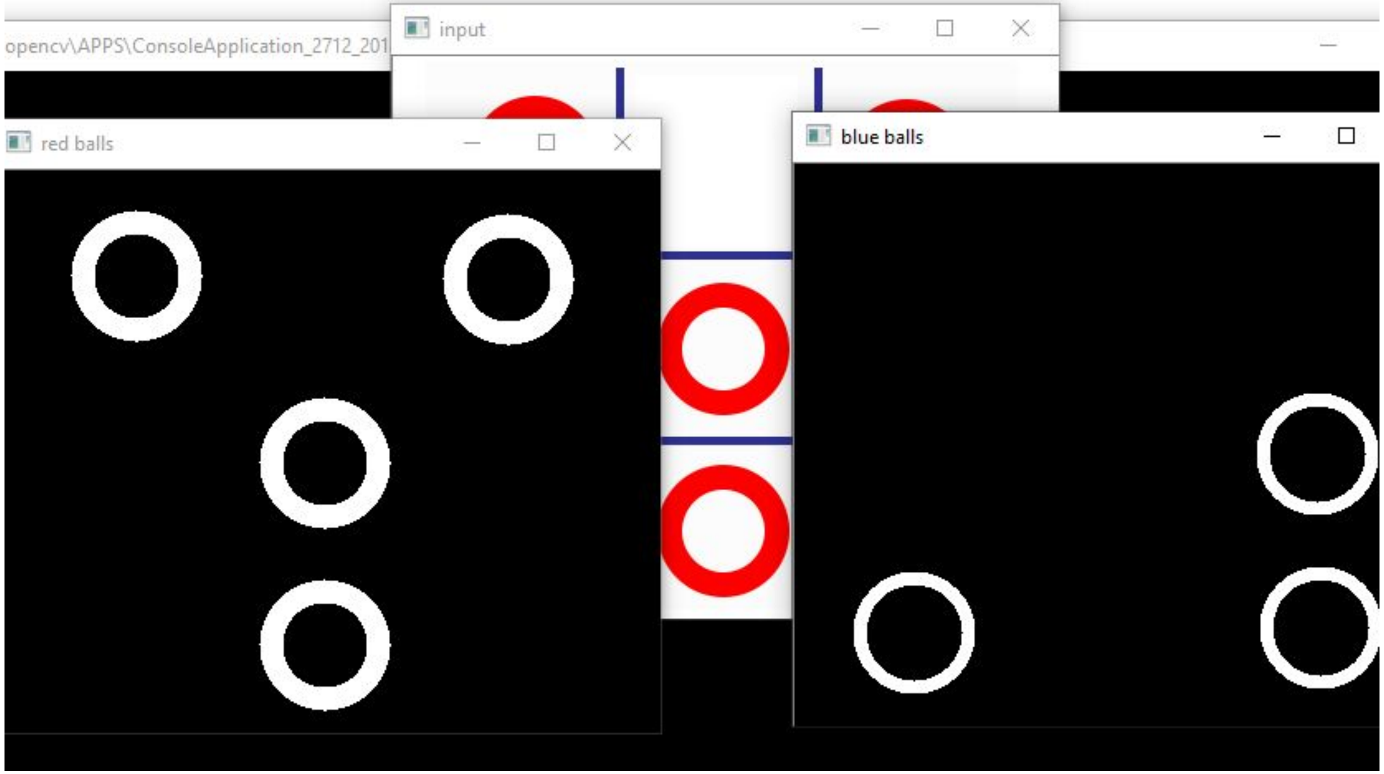
Tictactoe: situation



Open CV demonstration

Detect colors

Tictactoe: situation

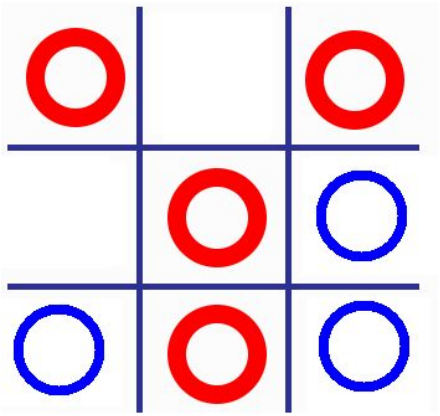


OpenCV functions can detect red and blua balls

Open CV demonstration

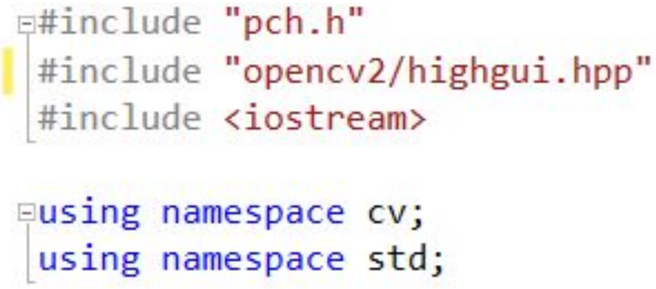
Detect colors

Tictactoe: situation



Codes

These headerfiles are added to Visual Studio console project

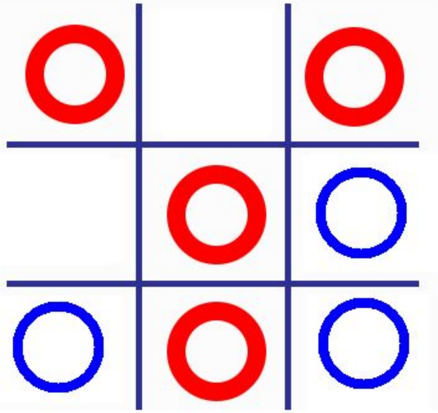


```
#include "pch.h"
#include "opencv2/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;
```

Open CV demonstration

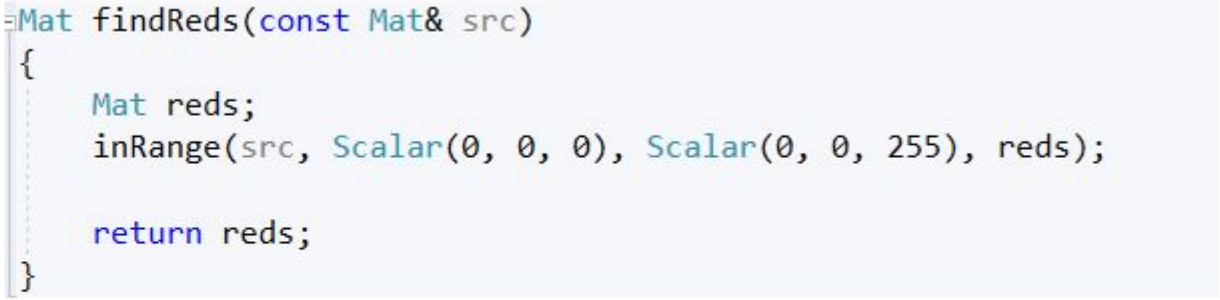
Detect colors

Tictactoe: situation

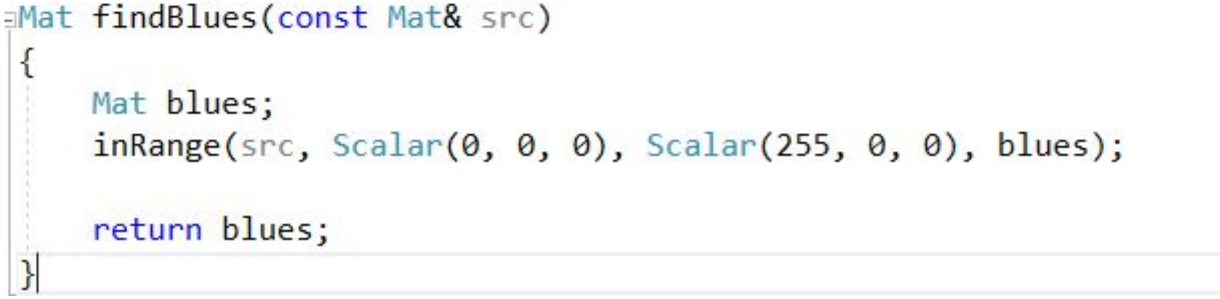


Codes

Here are the important functions that detect balls



```
Mat findReds(const Mat& src)
{
    Mat reds;
    inRange(src, Scalar(0, 0, 0), Scalar(0, 0, 255), reds);

    return reds;
}
```

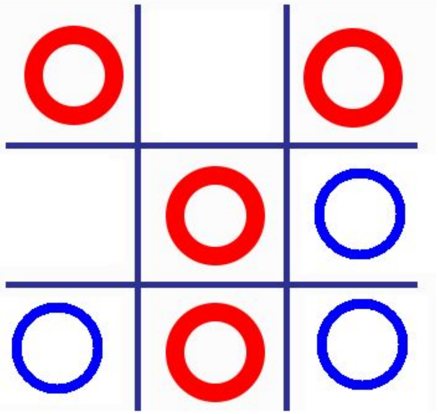


```
Mat findBlues(const Mat& src)
{
    Mat blues;
    inRange(src, Scalar(0, 0, 0), Scalar(255, 0, 0), blues);

    return blues;
}
```

Open CV demonstration

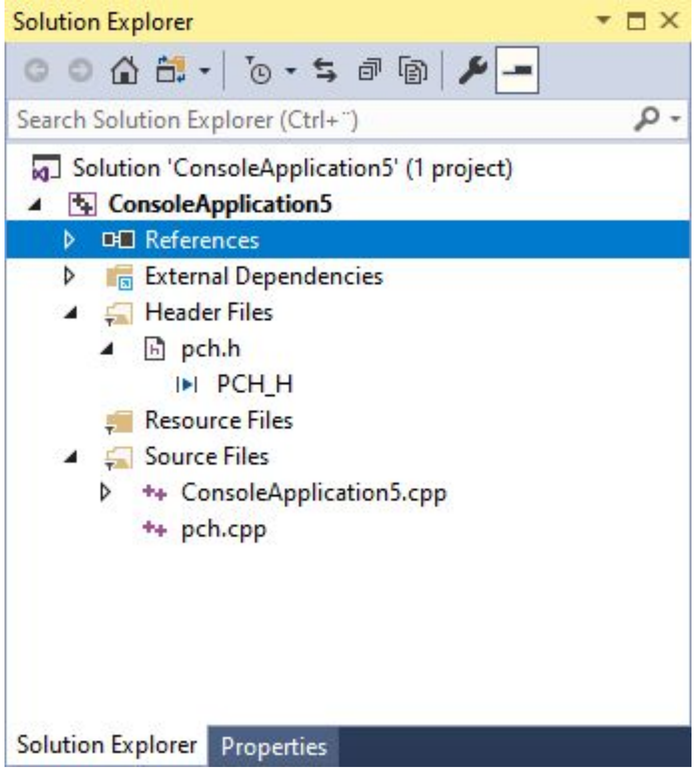
Detect colors

Tictactoe: situation



Codes

Included pch files



Solution 'ConsoleApplication5' (1 project)
- **ConsoleApplication5**
  - References
  - External Dependencies
  - Header Files
    - pch.h
      - PCH_H
  - Resource Files
  - Source Files
    - ConsoleApplication5.cpp
    - pch.cpp

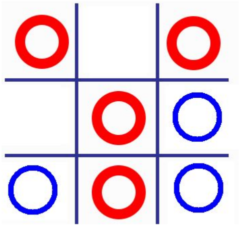
Open CV demonstration

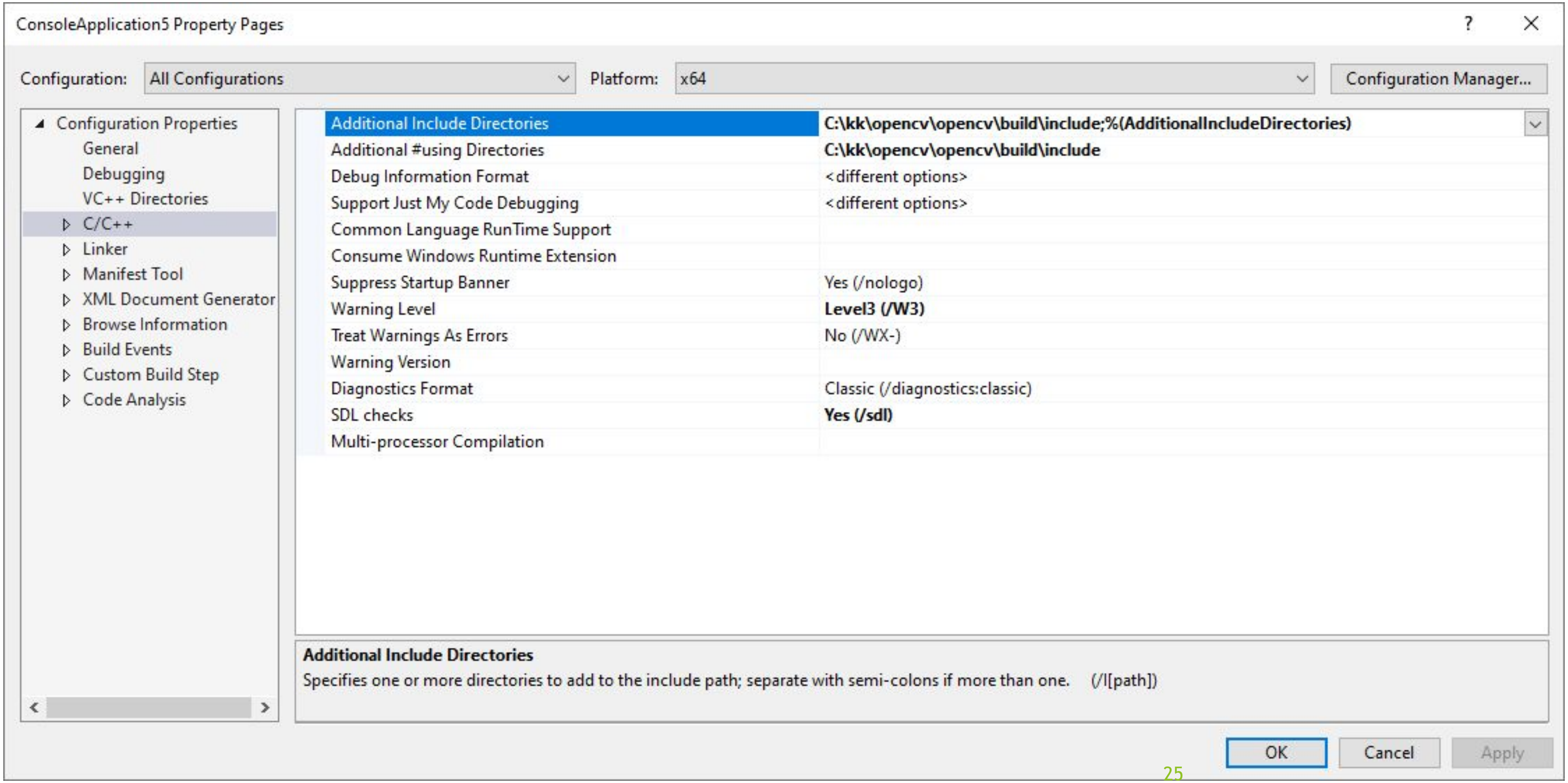
Detect colors

Tictactoe: situation

Codes

Additional directories needed for opencv:





Configuration: All Configurations

Platform: x64

Additional Include Directories	**C:\kk\opencv\opencv\build\include;%(AdditionalIncludeDirectories)**
Additional #using Directories	**C:\kk\opencv\opencv\build\include**
Debug Information Format	<different options>
Support Just My Code Debugging	<different options>
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Classic (/diagnostics:classic)
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	

Additional Include Directories

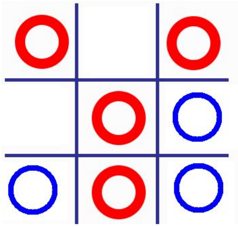
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

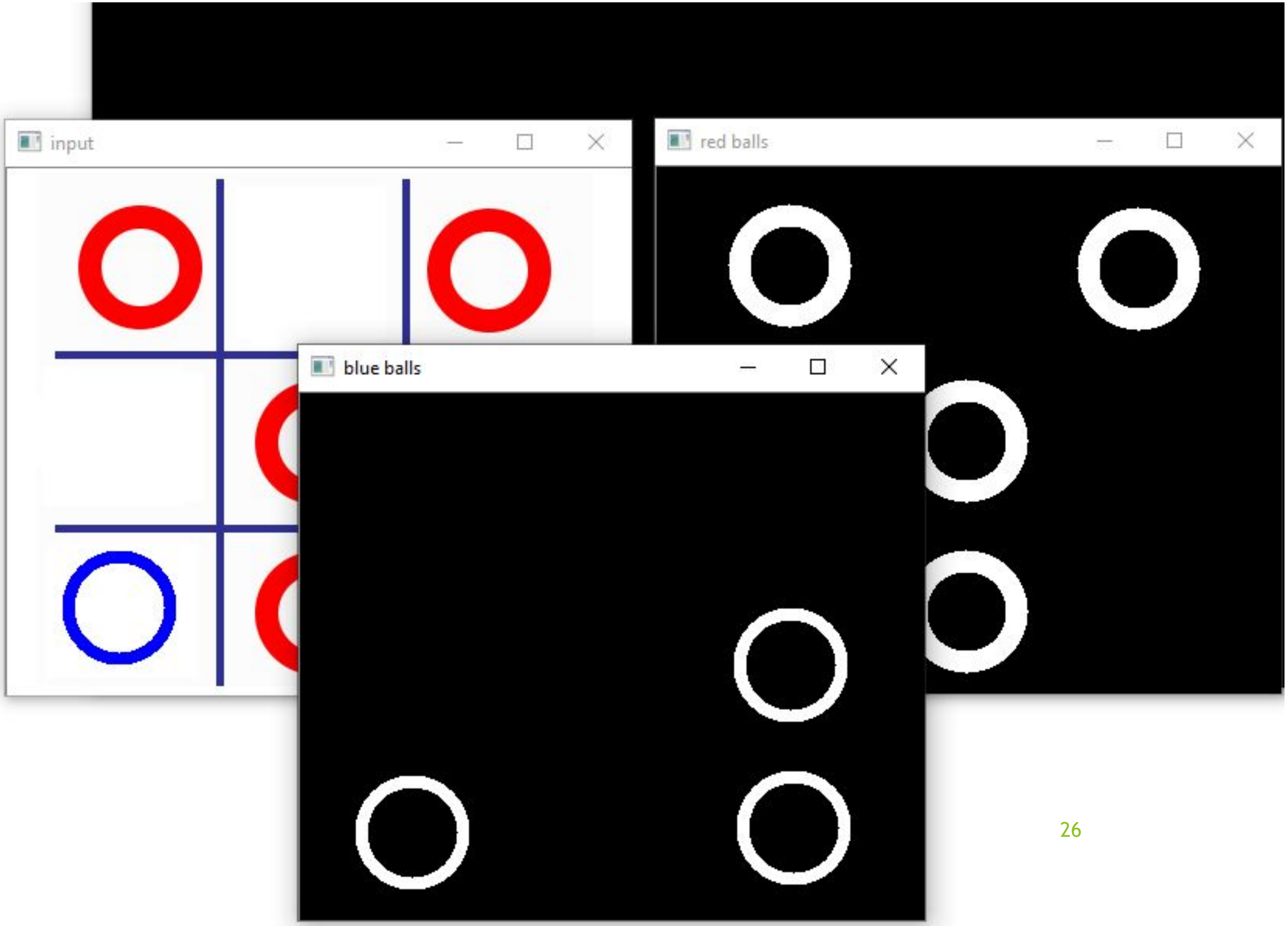
Open CV demonstration

Test run

Detect colors

Tictactoe: situation





Open CV demonstration

Detect faces and expressions

Sources:
OpenCV documentation

The goal is to test
Karan’s code…

https://www.paulekman.com/about/paul-ekman/

https://www.rcciit.org/students_projects/projects/it/2018/GR8.pdf

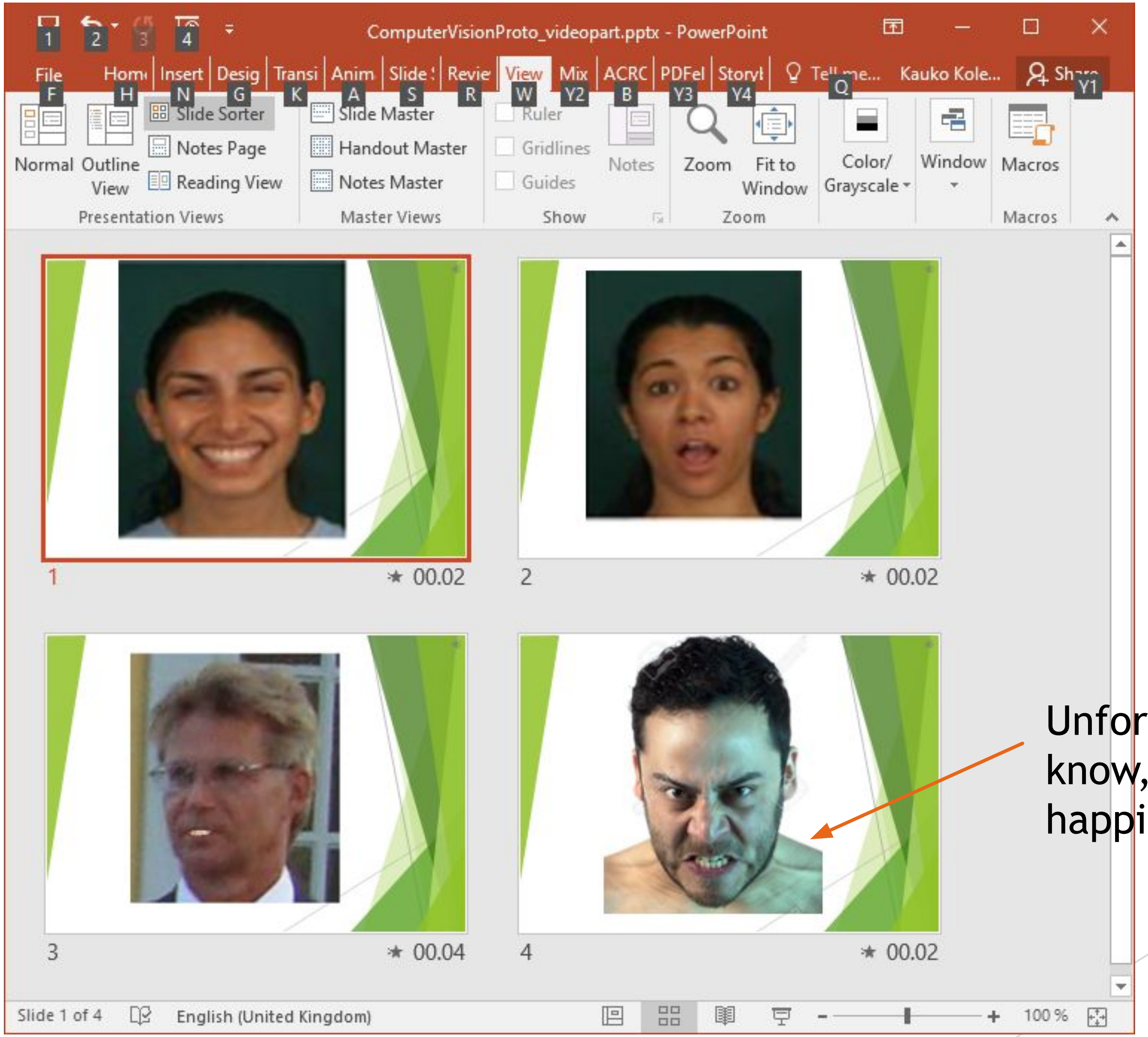
Main code that is tested in this presentation comes from
https://medium.com/swlh/emotion-detection-using-opencv-and-keras-771260bbd7f7

Thanks to Karan Sethi

Kakelino’s Code School

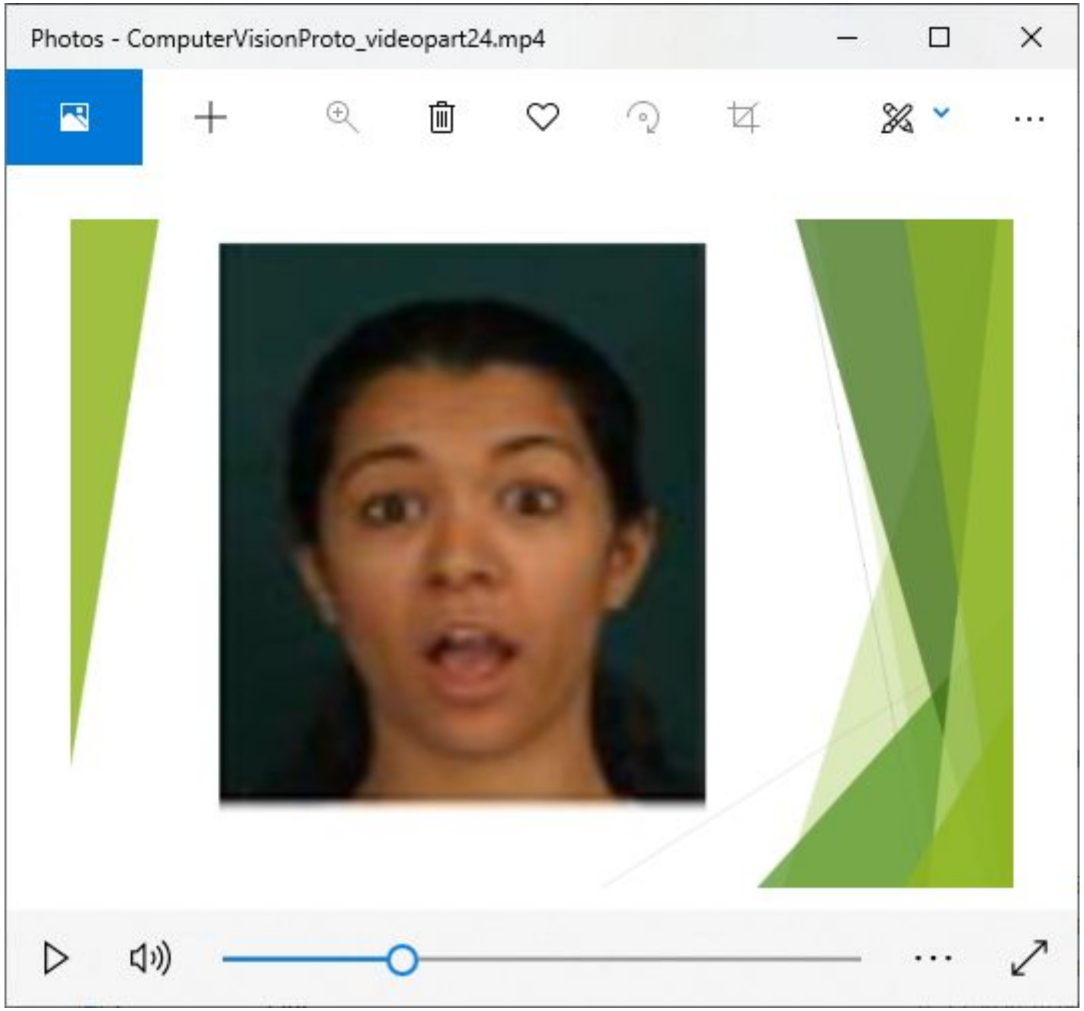
Here is the powerpoint document that is used as an image collection: it is exported to a video instead of using a life video (laptop cam)

Kakelino’s Code School



Unfortunately source not know, hopefully person is happier if we meet later…

Kakelino’s Code School



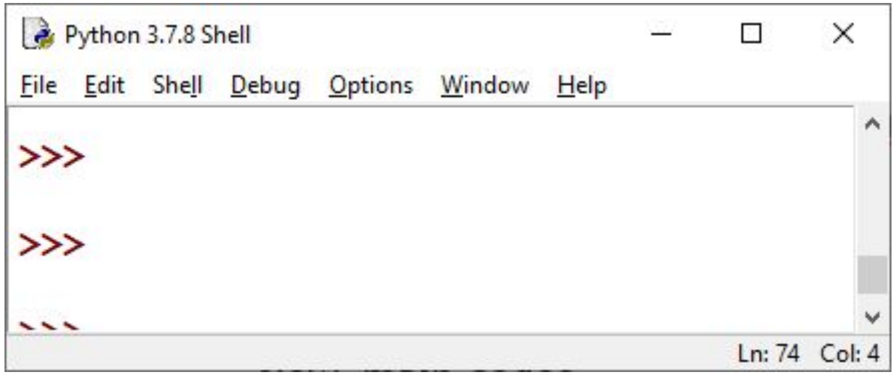
Photos - ComputerVisionProto_videopart24.mp4

Open CV demonstration

Detect faces and expressions

Now, main codes…

We use python and now the version was Python 3.7.8…



```
>>>

>>>
```

Open CV demonstration

Detect faces and expressions

Now, main codes…

To python environment you have to install tensorflow, keras and cv2.
Also pillow is needed…

Sometimes it is better to upgrade instead of trying to install moduless

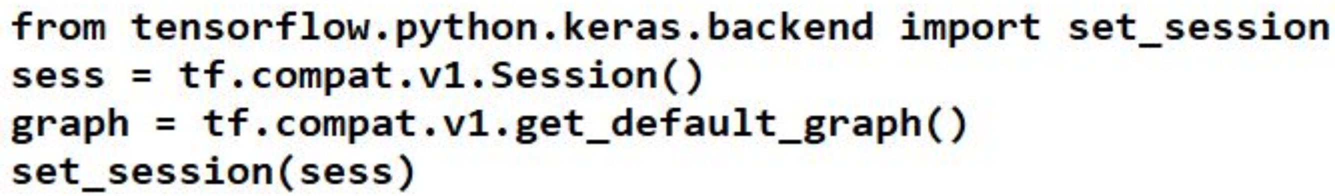
Tool/command pip is used for installing…

Open CV demonstration

Detect faces and expressions

Now, main codes…

Some adjustments had to be done because of my newer mdoule versions…

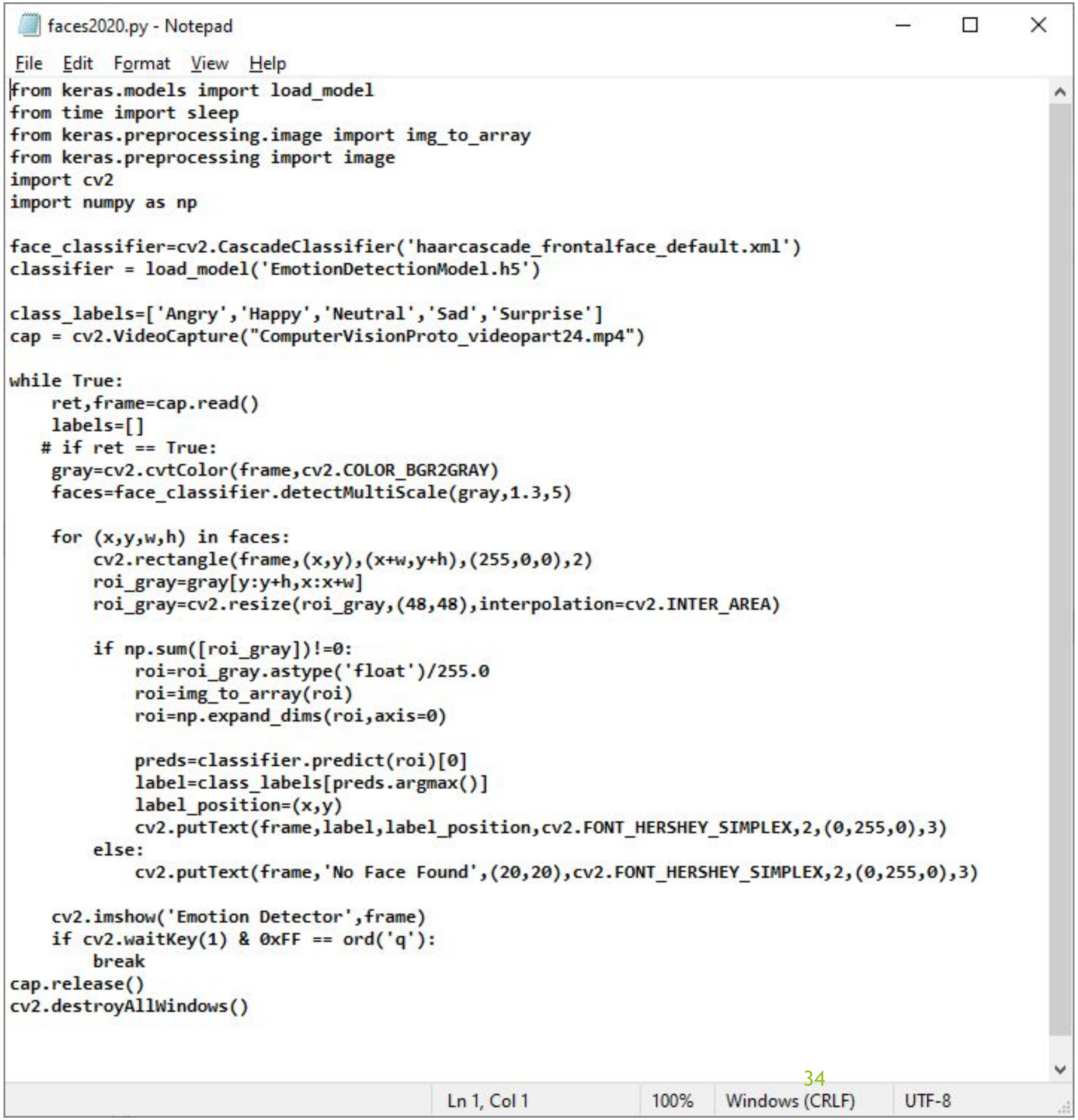


```
from tensorflow.python.keras.backend import set_session
sess = tf.compat.v1.Session()
graph = tf.compat.v1.get_default_graph()
set_session(sess)
```

Open CV demonstration

Now, main codes…

Detect faces and expressions



```
from keras.models import load_model
from time import sleep
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np

face_classifier=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
classifier = load_model('EmotionDetectionModel.h5')

class_labels=['Angry','Happy','Neutral','Sad','Surprise']
cap = cv2.VideoCapture("ComputerVisionProto_videopart24.mp4")

while True:
    ret,frame=cap.read()
    labels=[]
   # if ret == True:
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces=face_classifier.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray=gray[y:y+h,x:x+w]
        roi_gray=cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)

        if np.sum([roi_gray])!=0:
            roi=roi_gray.astype('float')/255.0
            roi=img_to_array(roi)
            roi=np.expand_dims(roi,axis=0)

            preds=classifier.predict(roi)[0]
            label=class_labels[preds.argmax()]
            label_position=(x,y)
            cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
        else:
            cv2.putText(frame,'No Face Found',(20,20),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)

    cv2.imshow('Emotion Detector',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Open CV demonstration

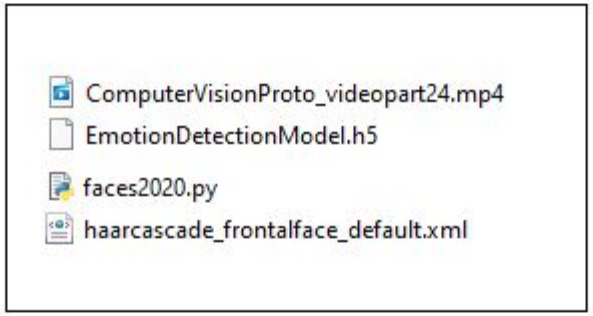
Detect faces and expressions

Now, main codes…

All needed files are now in the same folder

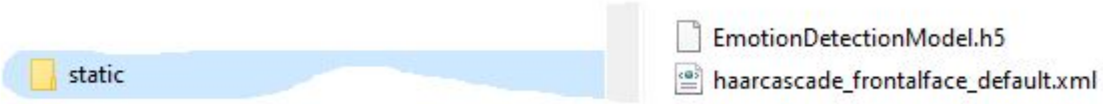
Code from Karan Sethi Thank you!

- code is published in github



ComputerVisionProto_videopart24.mp4
EmotionDetectionModel.h5
faces2020.py
haarcascade_frontalface_default.xml

Training and validation data is in static subfolder



static

EmotionDetectionModel.h5
haarcascade_frontalface_default.xml

Open CV demonstration

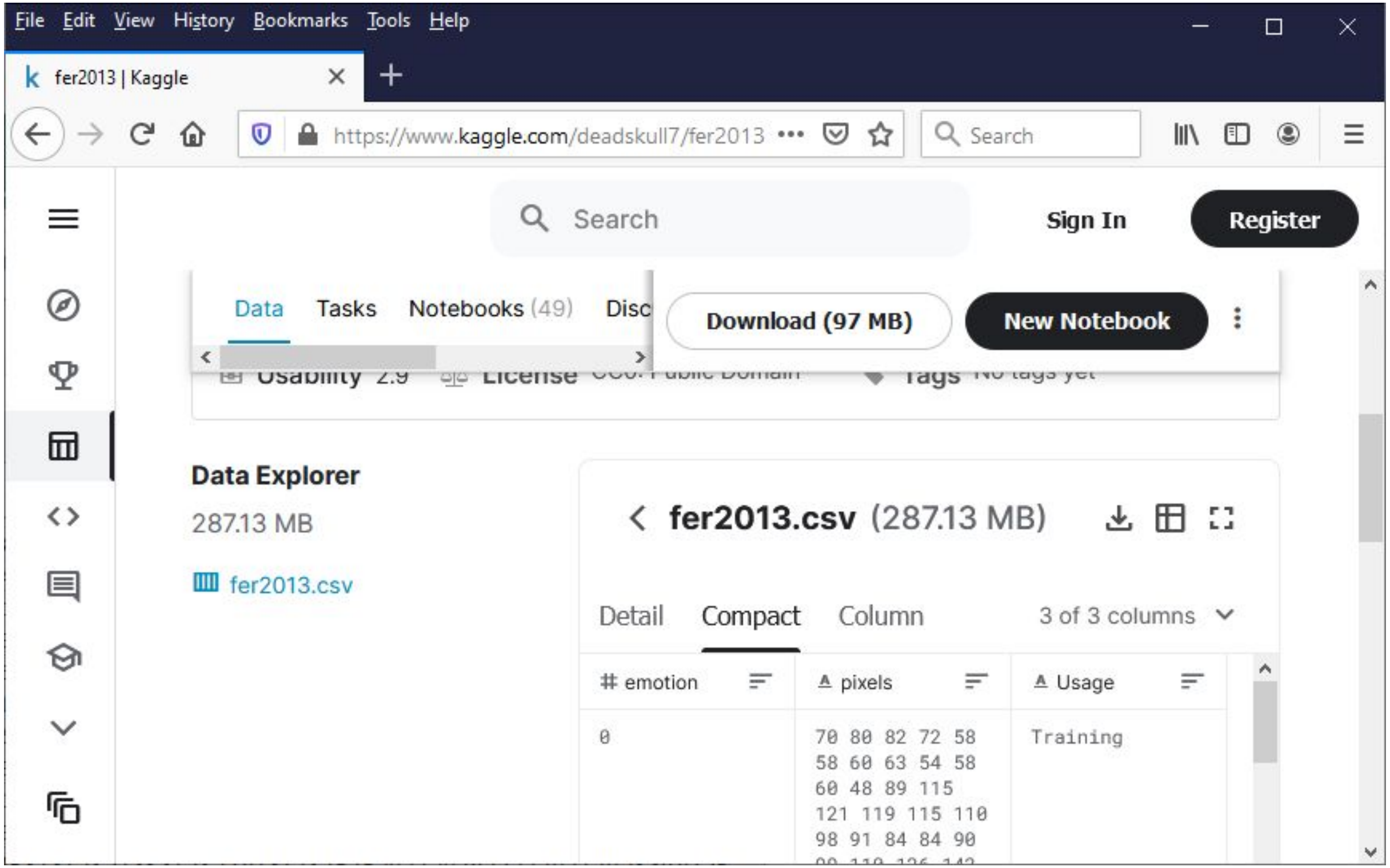
Detect faces and expressions

Now, main codes…

Expression database is fer2012.csv

Code from Karan Sethi
Thank you!

- code is published in github



Data Explorer

287.13 MB

fer2013.csv

fer2013.csv (287.13 MB)

Detail | Compact | Column | 3 of 3 columns

# emotion	pixels	Usage
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 91 84 84 90 99 110 126 143	Training

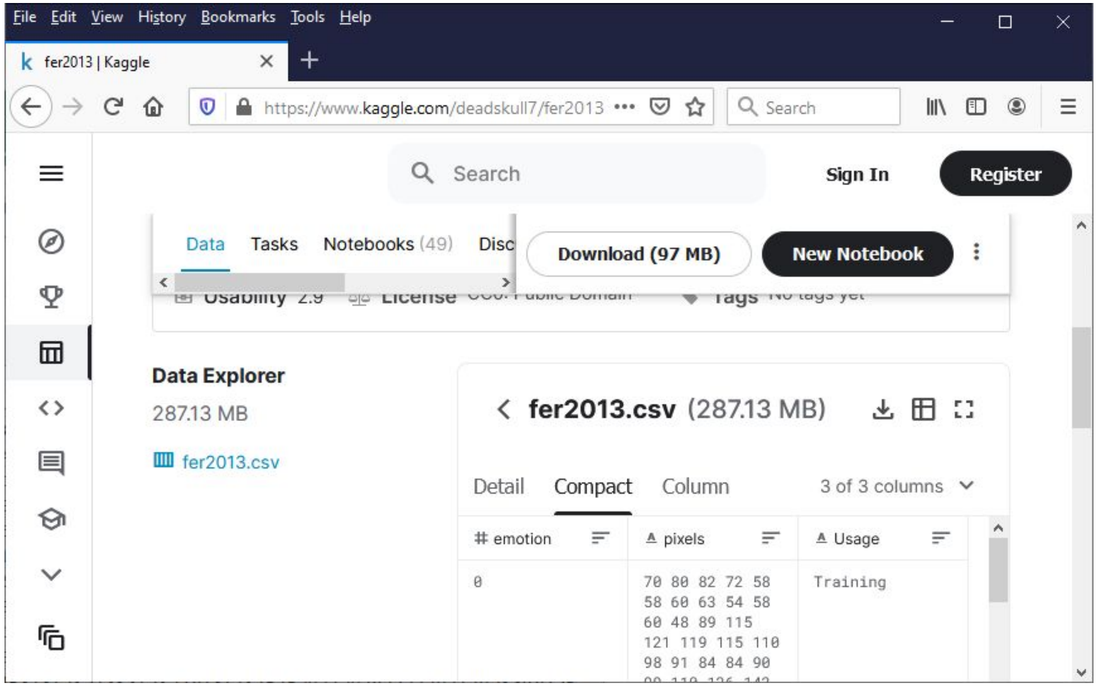
Open CV demonstration

Detect faces and expressions

Now, main codes…

Code from
Karan Sethi
Thank you!
- code is published in github

Expression database is fer2012.csv



# emotion	pixels	Usage
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 91 84 84 90	Training

It is used to get training and validation data.

Open CV demonstration

Detect faces and expressions

Now, main codes…

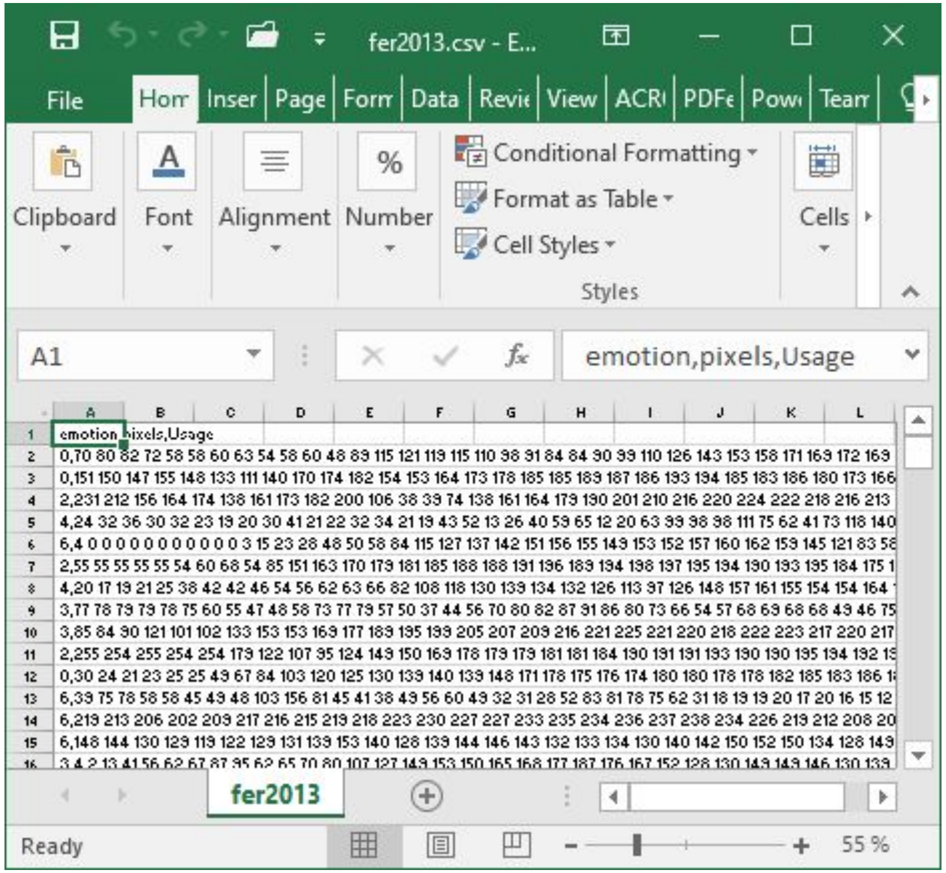
Code from
Karan Sethi
Thank you!

- code is published in github

Expression database is fer2012.csv

It is used to get training and validation data.

Sample in Excel



A1 · emotion,pixels,Usage

```
   A       B       C       D       E       F       G       H       I       J       K       L
1  emotion,pixels,Usage
2  0,70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 91 84 84 90 99 110 126 143 153 158 171 169 172 169
3  0,151 150 147 155 148 133 111 140 170 174 182 154 153 164 173 178 185 185 189 187 186 193 194 185 183 186 180 173 166
4  2,231 212 156 164 174 138 161 173 182 200 106 38 39 74 138 161 164 179 190 201 210 216 220 224 222 218 216 213
5  4,24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 19 43 52 13 26 40 59 65 12 20 63 99 98 98 111 75 62 41 73 118 140
6  6,4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84 115 127 137 142 151 156 155 149 153 152 157 160 162 159 145 121 83 58
7  2,55 55 55 55 55 54 60 68 54 85 151 163 170 179 181 185 188 188 191 196 189 194 198 197 195 194 190 193 195 184 175 1
8  4,20 17 19 21 25 38 42 42 46 54 56 62 63 66 82 108 118 130 139 134 132 126 113 97 126 148 157 161 155 154 154 164
9  3,77 78 79 79 78 75 60 55 47 48 58 73 77 79 57 50 37 44 56 70 80 82 87 91 86 80 73 66 54 57 68 69 68 68 49 46 75
10 3,85 84 90 121 101 102 133 153 153 169 177 189 195 199 205 207 209 216 221 225 221 220 218 222 223 217 220 217
11 2,255 254 255 254 254 179 122 107 95 124 149 150 169 178 179 179 181 181 184 190 191 191 193 190 190 195 194 192 1
12 0,30 24 21 23 25 25 49 67 84 103 120 125 130 139 140 139 148 171 178 175 176 174 180 180 178 178 182 185 183 186 1
13 6,39 75 78 58 58 45 49 48 103 156 81 45 41 38 49 56 60 49 32 31 28 52 83 81 78 75 62 31 18 19 19 20 17 20 16 15 12
14 6,219 213 206 202 209 217 216 215 219 218 223 230 227 227 233 235 234 236 237 238 234 226 219 212 208 20
15 6,148 144 130 129 119 122 129 131 139 153 140 128 139 144 146 143 132 133 134 130 140 142 150 152 150 134 128 149
16 3,4 2 13 41 56 62 67 87 95 62 65 70 80 107 127 149 153 150 165 168 177 187 176 167 152 128 130 149 149 146 130 139
```

Open CV demonstration

Detect faces and expressions

Now, main codes…

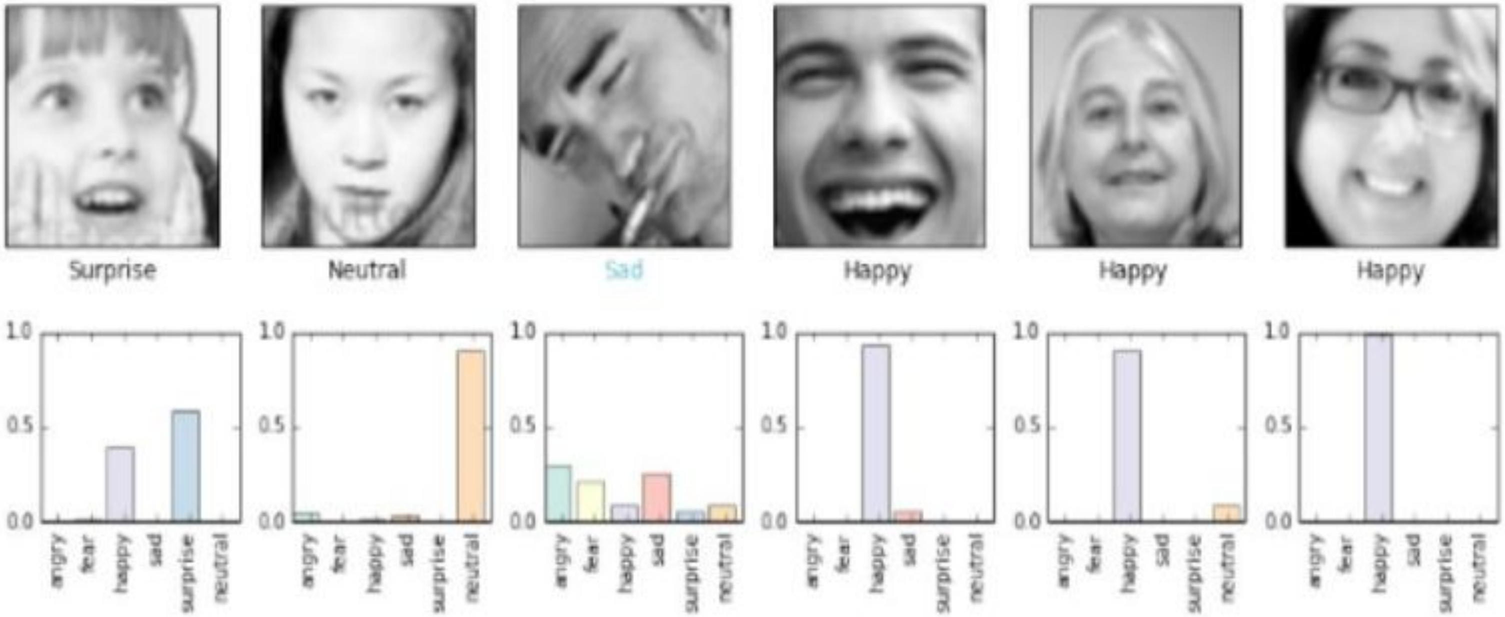
Expression database is fer2012.csv

It is used to get training and validation data.

Code from Karan Sethi Thank you!

- code is published in github

Example of model validation



Open CV demonstration

Detect faces and expressions

There are several steps to go trough when setting up face and expression detecting system….

Good instructions can be found from Internet, Opencv documents and published codes..

Good sources were given also at the beginning of this presentation

Open CV demonstration

Detect faces and expressions

There are several steps to go trough when setting up face and expression detecting system….

Good instructions can be found from Internet, Opencv documents and published codes..

Good sources were given also at the beginning of this presentation

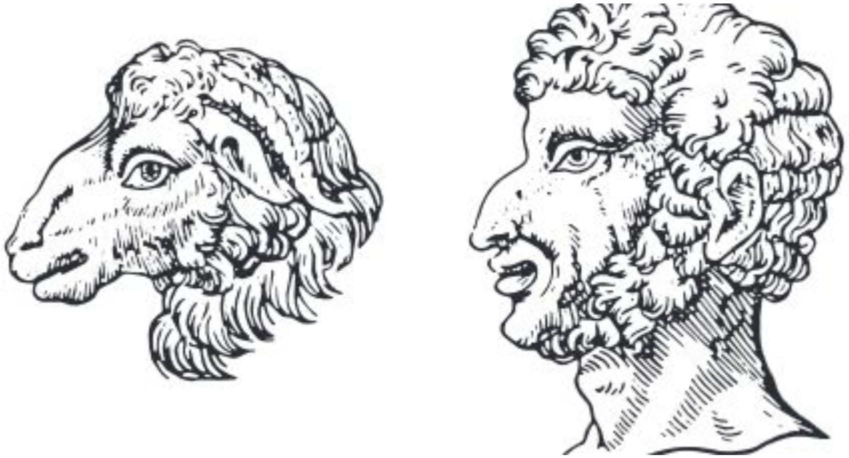
Now the main goal. We wanted to test if our code can detect expressions from our own video!

Open CV demonstration

Detect faces and expressions

Here is a life presentation

Now the main goal. We wanted to test if our code can detect expressions from our own video!



Small joke: these are too difficult to detect or separate (sourse John Liggings)

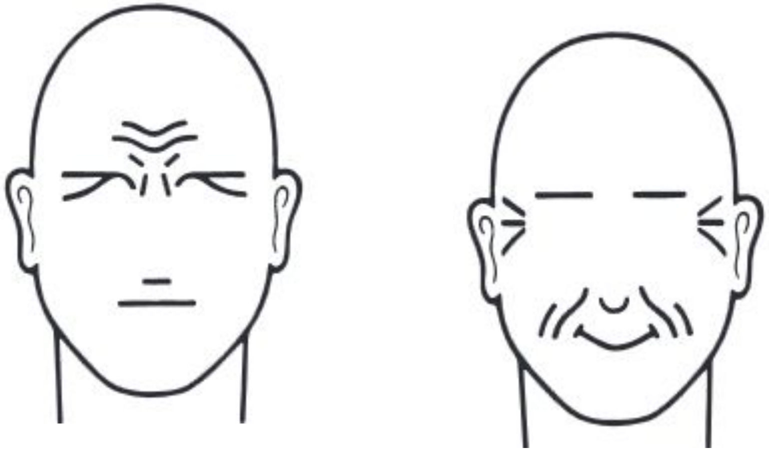
Open CV demonstration

Detect faces and expressions

Now the main goal.
We wanted to test if
our code can detect
expressions from our
own video!

Here is a life presentation

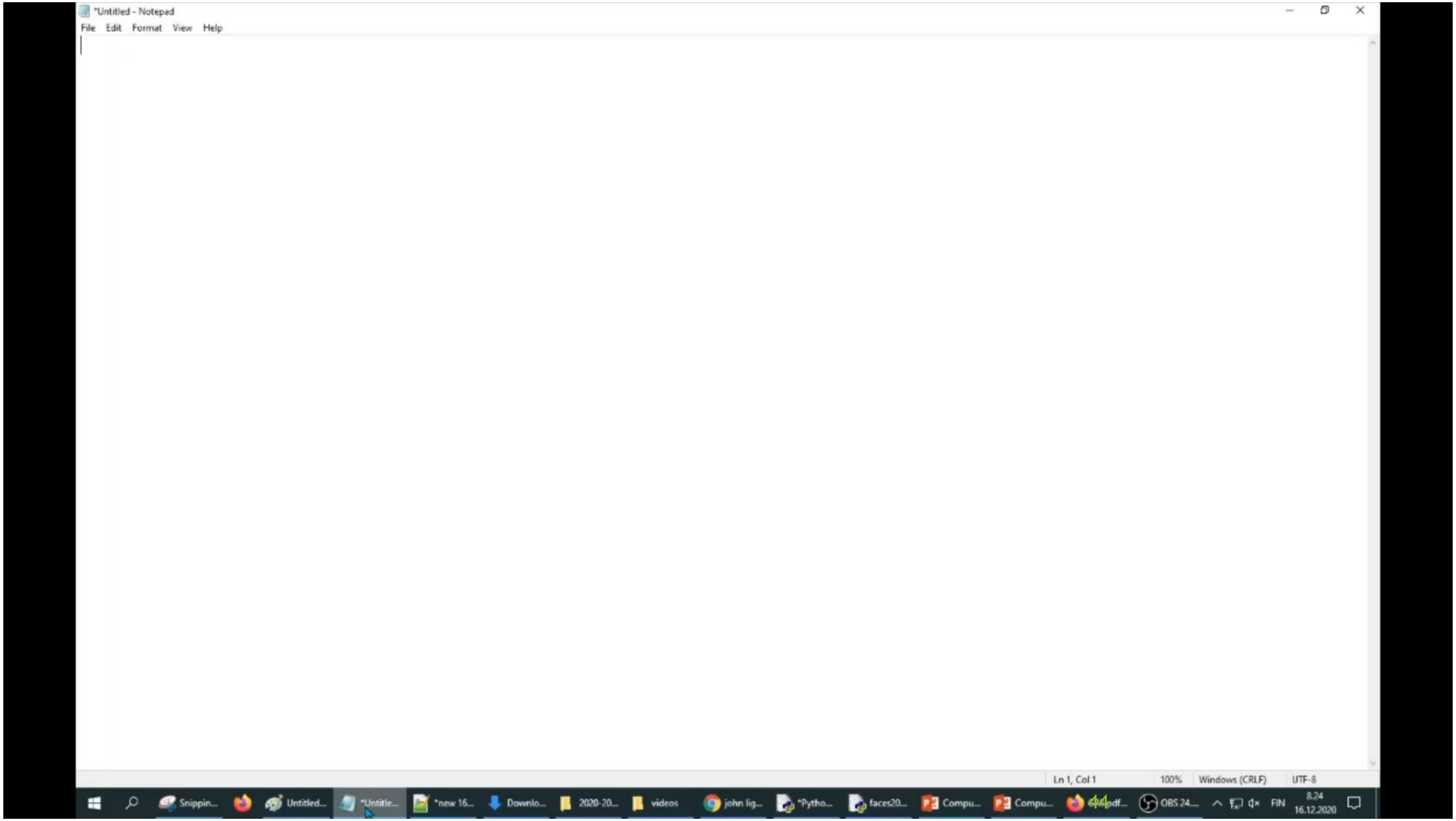
What about these?

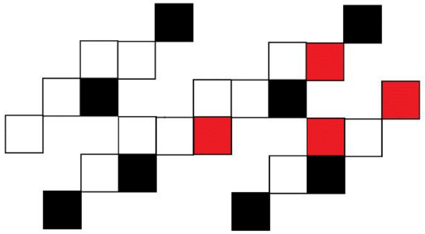


Open CV demonstration

Detect faces and expressions

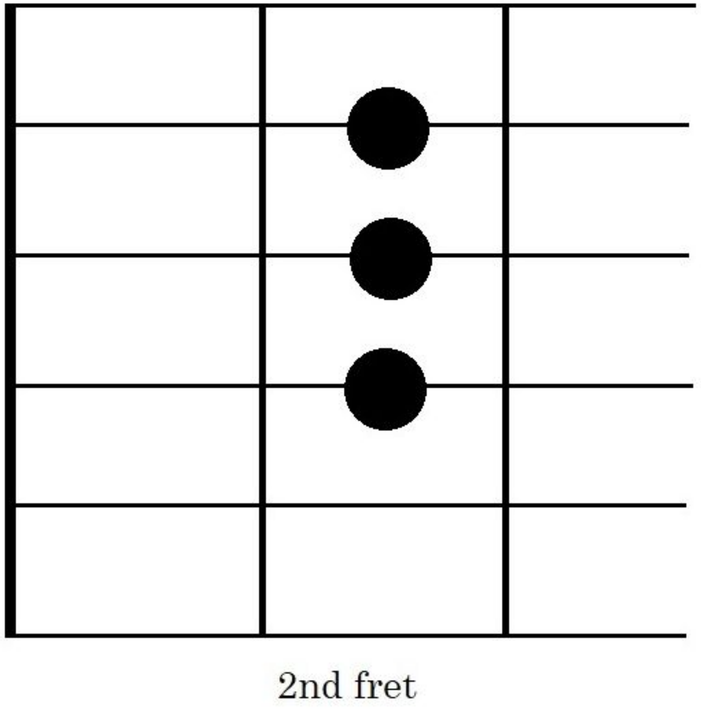
Here is a life presentation





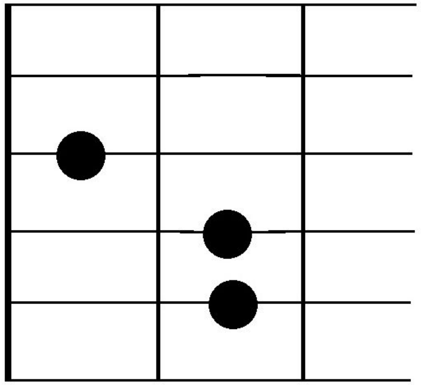
Open CV demonstration

Detecting chords

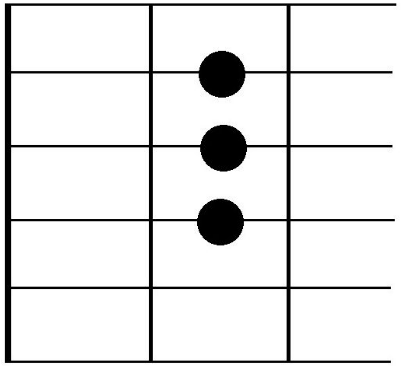


Open CV demonstration

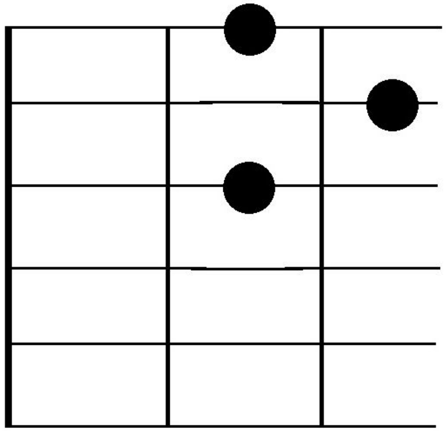
Detecting chords



2nd fret



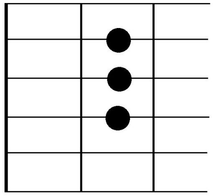
2nd fret



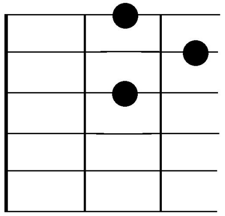
2nd fret

Open CV demonstration

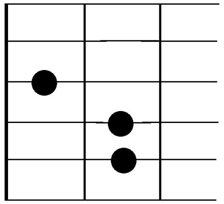
Detecting chords



2nd fret



2nd fret

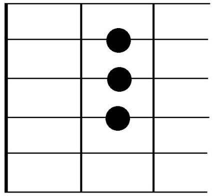


2nd fret

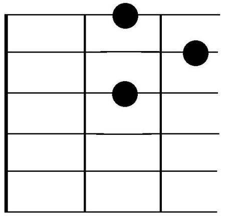
Open CV demonstration

Detecting chords

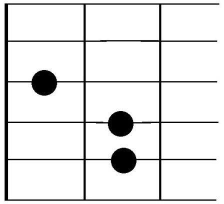
We have only 3 chords here, they are played with a guitar …



2nd fret



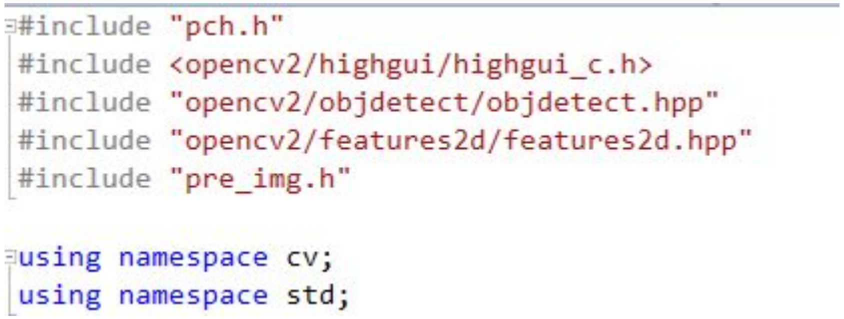
2nd fret



2nd fret

Codes
Source is opencv

We have a Visual Studio C++ Console project.
These header files are added first

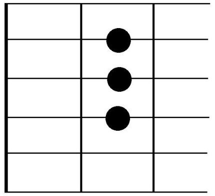


```
#include "pch.h"
#include <opencv2/highgui/highgui_c.h>
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/features2d/features2d.hpp"
#include "pre_img.h"

using namespace cv;
using namespace std;
```

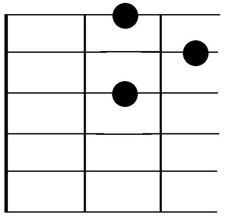
Open CV demonstration

Detecting chords

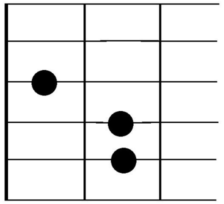
We have only 3 chords here, they are played with a guitar …



2nd fret



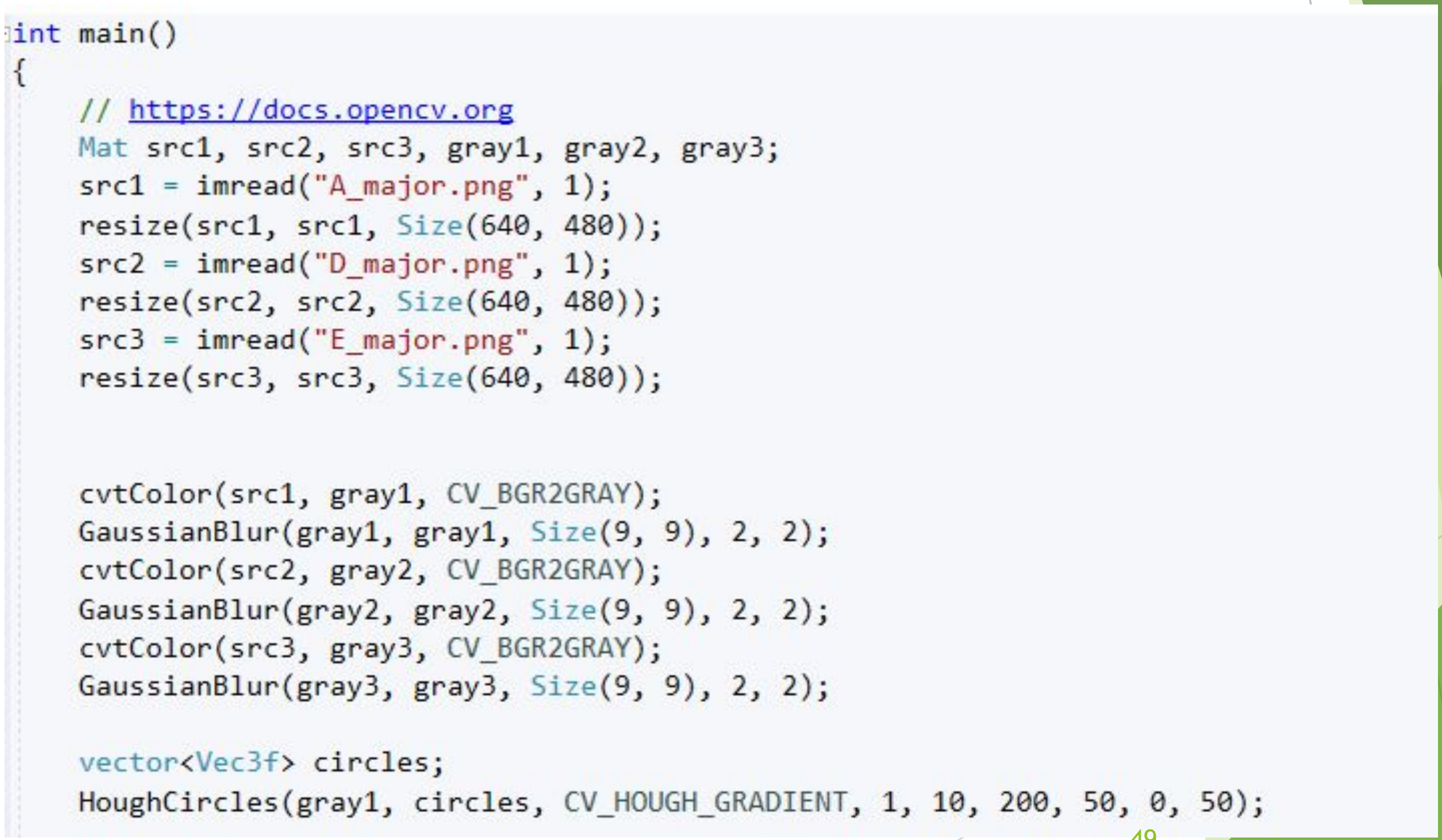
2nd fret



2nd fret

Codes
Source is opencv

Function main()
A)

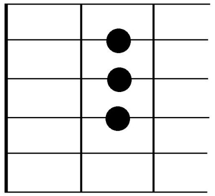


```
int main()
{
    // https://docs.opencv.org
    Mat src1, src2, src3, gray1, gray2, gray3;
    src1 = imread("A_major.png", 1);
    resize(src1, src1, Size(640, 480));
    src2 = imread("D_major.png", 1);
    resize(src2, src2, Size(640, 480));
    src3 = imread("E_major.png", 1);
    resize(src3, src3, Size(640, 480));


    cvtColor(src1, gray1, CV_BGR2GRAY);
    GaussianBlur(gray1, gray1, Size(9, 9), 2, 2);
    cvtColor(src2, gray2, CV_BGR2GRAY);
    GaussianBlur(gray2, gray2, Size(9, 9), 2, 2);
    cvtColor(src3, gray3, CV_BGR2GRAY);
    GaussianBlur(gray3, gray3, Size(9, 9), 2, 2);

    vector<Vec3f> circles;
    HoughCircles(gray1, circles, CV_HOUGH_GRADIENT, 1, 10, 200, 50, 0, 50);
```

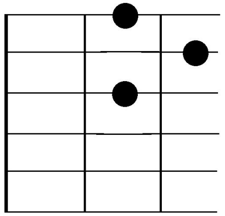
Open CV demonstration

Detecting chords

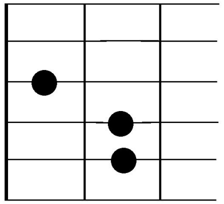
We have only 3 chords here, they are played with a guitar …



2nd fret



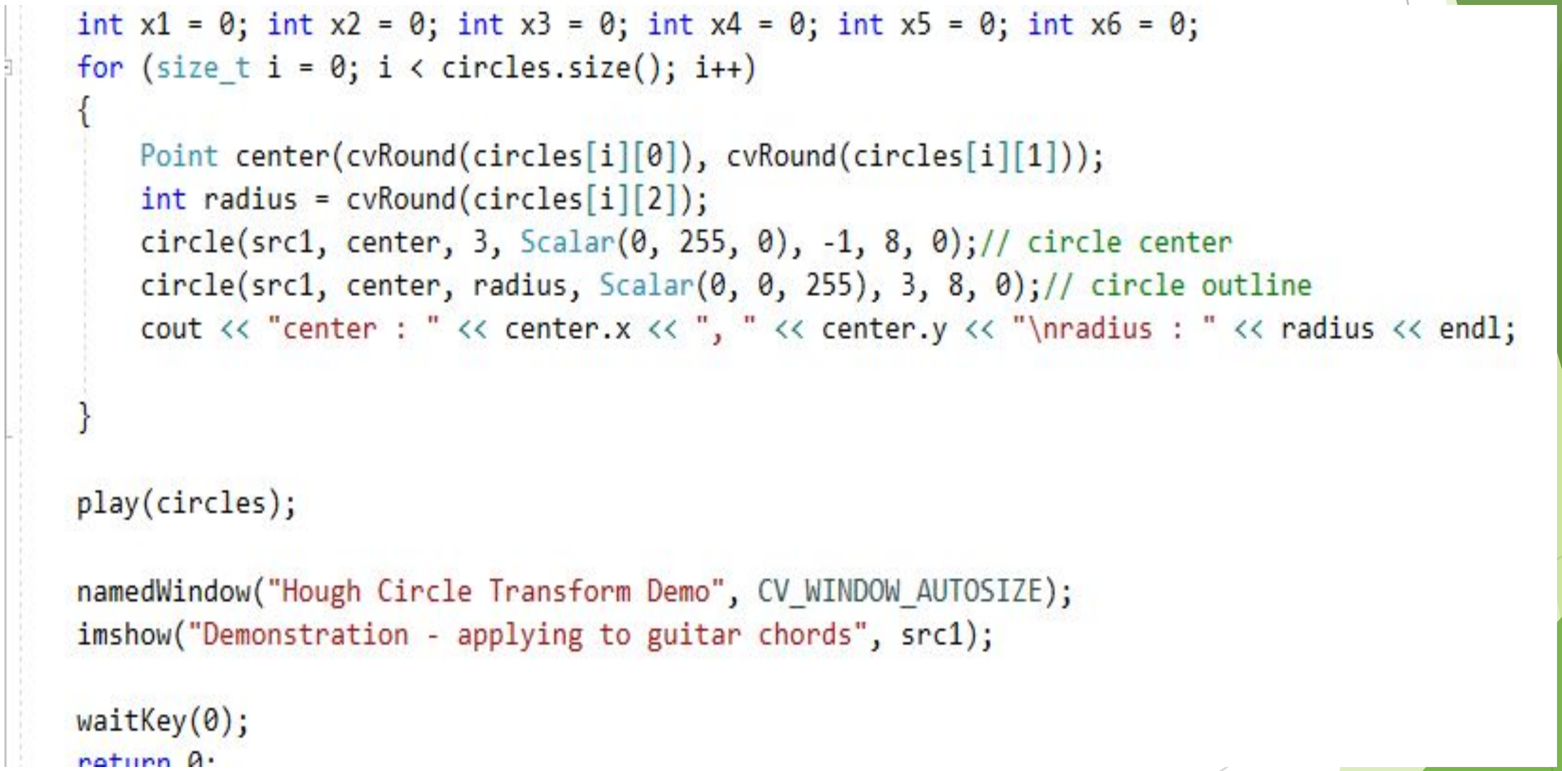
2nd fret



2nd fret

Codes
Source is opencv

Function main()
B)



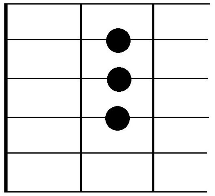
```
int x1 = 0; int x2 = 0; int x3 = 0; int x4 = 0; int x5 = 0; int x6 = 0;
for (size_t i = 0; i < circles.size(); i++)
{
    Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
    int radius = cvRound(circles[i][2]);
    circle(src1, center, 3, Scalar(0, 255, 0), -1, 8, 0);// circle center
    circle(src1, center, radius, Scalar(0, 0, 255), 3, 8, 0);// circle outline
    cout << "center : " << center.x << ", " << center.y << "\nradius : " << radius << endl;

}

play(circles);

namedWindow("Hough Circle Transform Demo", CV_WINDOW_AUTOSIZE);
imshow("Demonstration - applying to guitar chords", src1);

waitKey(0);
return 0;
```

Open CV demonstration

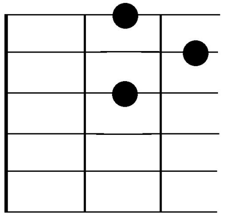
Codes
Source is opencv

Detecting chords

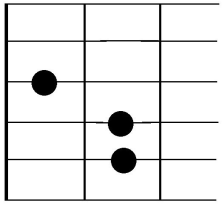
We have only 3 chords here, they are played with a guitar …



2nd fret

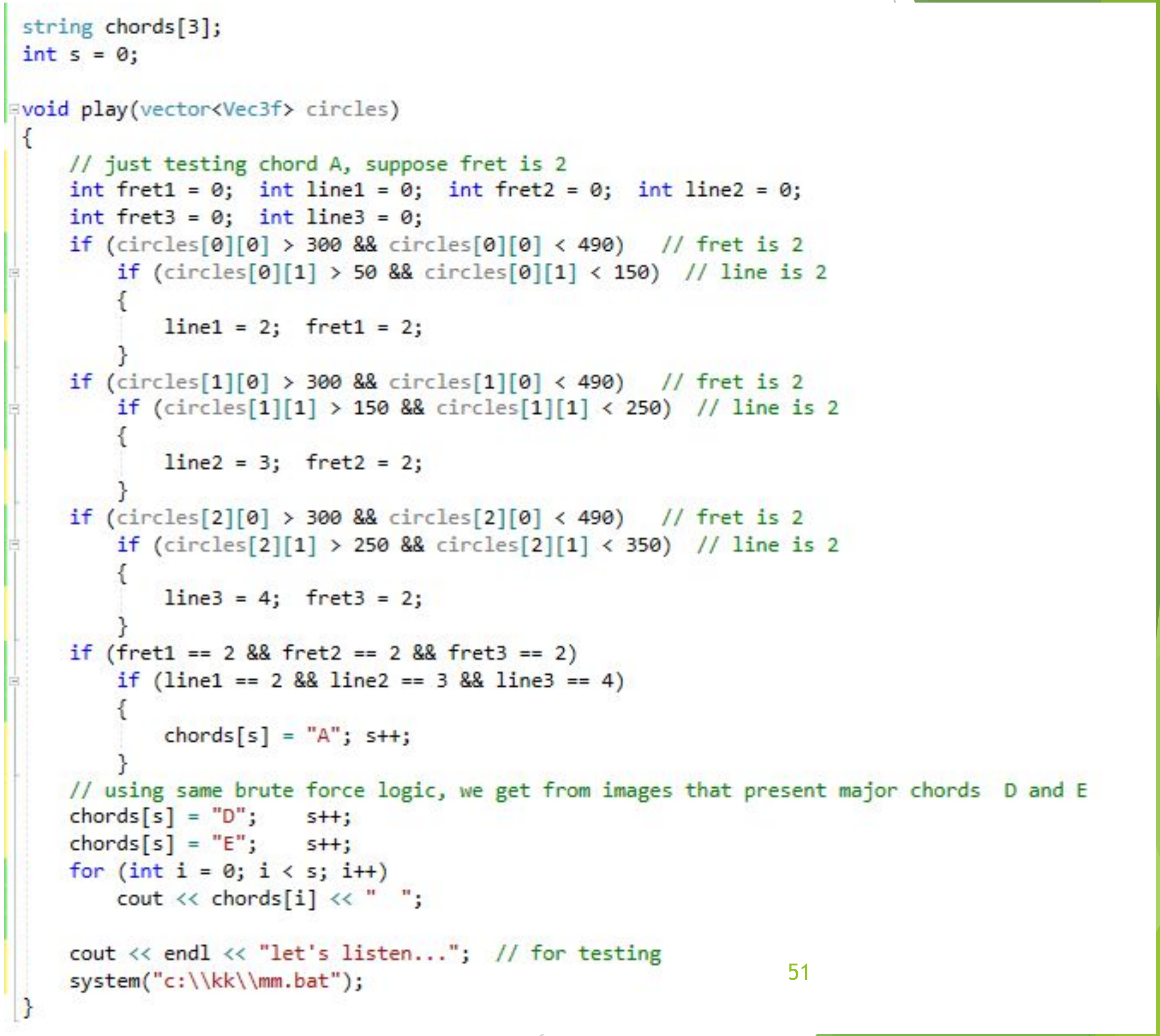


2nd fret



2nd fret

Function play()



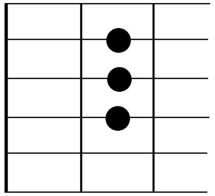
```
string chords[3];
int s = 0;

void play(vector<Vec3f> circles)
{
    // just testing chord A, suppose fret is 2
    int fret1 = 0;  int line1 = 0;  int fret2 = 0;  int line2 = 0;
    int fret3 = 0;  int line3 = 0;
    if (circles[0][0] > 300 && circles[0][0] < 490)   // fret is 2
        if (circles[0][1] > 50 && circles[0][1] < 150)  // line is 2
        {
            line1 = 2;  fret1 = 2;
        }
    if (circles[1][0] > 300 && circles[1][0] < 490)   // fret is 2
        if (circles[1][1] > 150 && circles[1][1] < 250)  // line is 2
        {
            line2 = 3;  fret2 = 2;
        }
    if (circles[2][0] > 300 && circles[2][0] < 490)   // fret is 2
        if (circles[2][1] > 250 && circles[2][1] < 350)  // line is 2
        {
            line3 = 4;  fret3 = 2;
        }
    if (fret1 == 2 && fret2 == 2 && fret3 == 2)
        if (line1 == 2 && line2 == 3 && line3 == 4)
        {
            chords[s] = "A"; s++;
        }
    // using same brute force logic, we get from images that present major chords  D and E
    chords[s] = "D";    s++;
    chords[s] = "E";    s++;
    for (int i = 0; i < s; i++)
        cout << chords[i] << "  ";

    cout << endl << "let's listen...";  // for testing
    system("c:\\kk\\mm.bat");
}
```

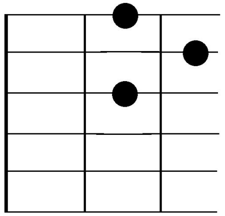
Open CV demonstration

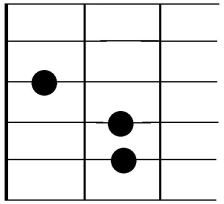
Codes
Source is opencv

Detecting chords

We have only 3 guitar chords here



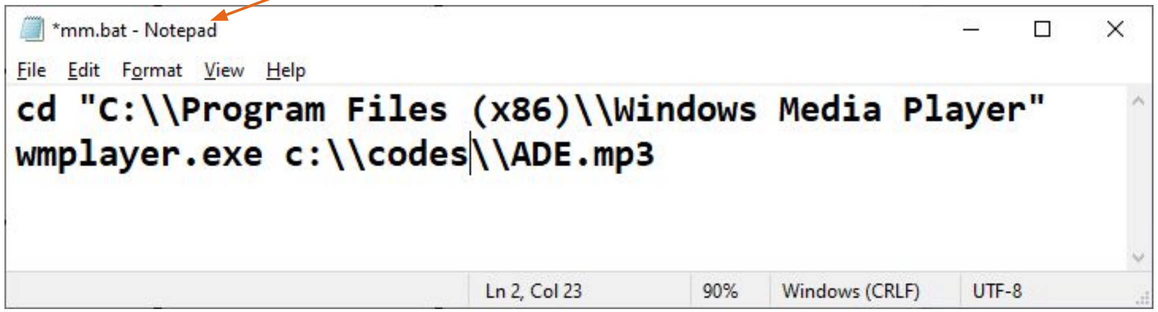




In this proto, chords are played with Windows Media Player

We use just a batch-file to open that player ☺️

```
system("c:\\codes\\mm.bat");
```



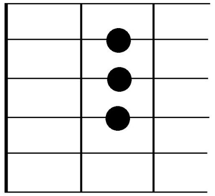
```
cd "C:\\Program Files (x86)\\Windows Media Player"
wmplayer.exe c:\\codes\\ADE.mp3
```

Open CV demonstration

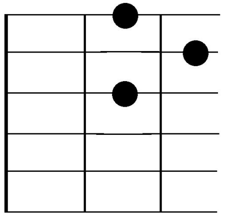
Codes
Source is opencv

Detecting chords

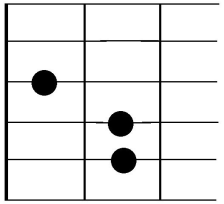
Test run



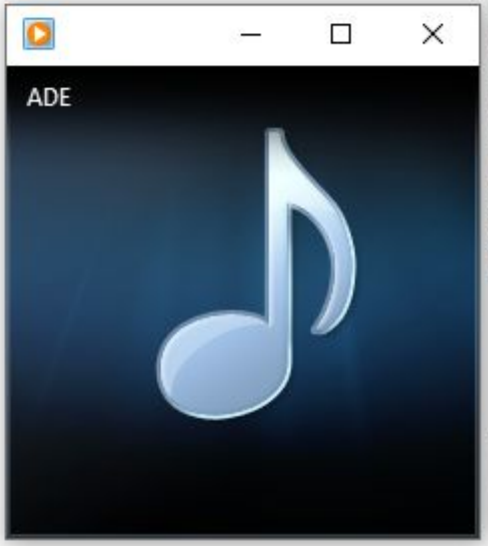
2nd fret

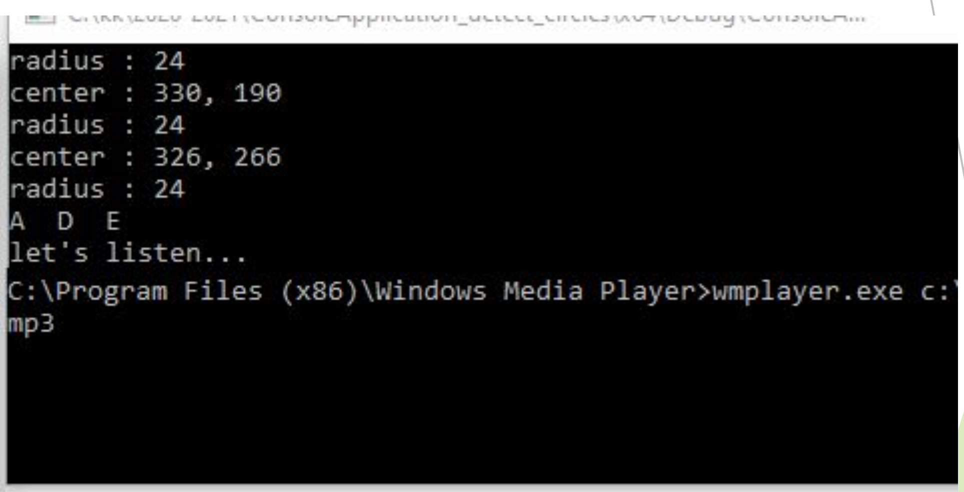


2nd fret



2nd fret





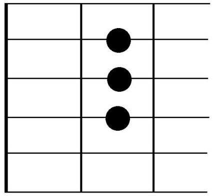
```
C:\kk\2020-2021\ConsoleApplication_detect_circles\x64\Debug\ConsoleA...
radius : 24
center : 330, 190
radius : 24
center : 326, 266
radius : 24
A  D  E
let's listen...
C:\Program Files (x86)\Windows Media Player>wmplayer.exe c:
mp3
```

Open CV demonstration

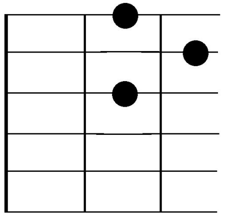
Codes
Source is opencv

Detecting chords

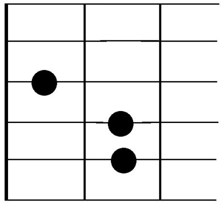
Test run



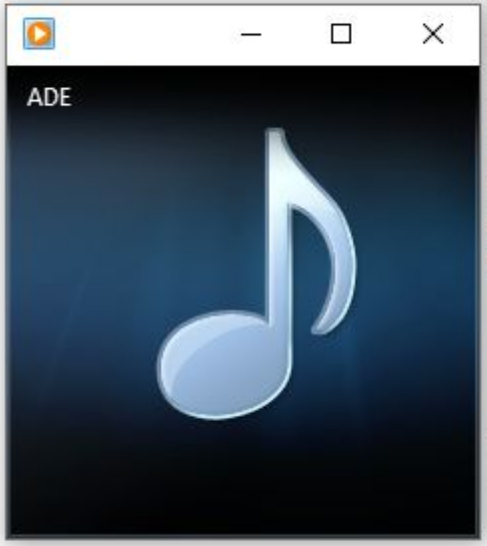
2nd fret

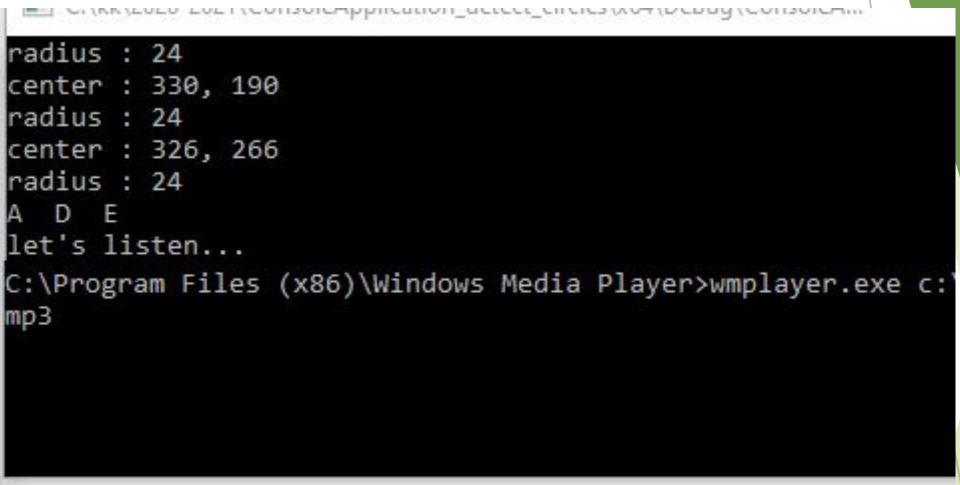


2nd fret



2nd fret

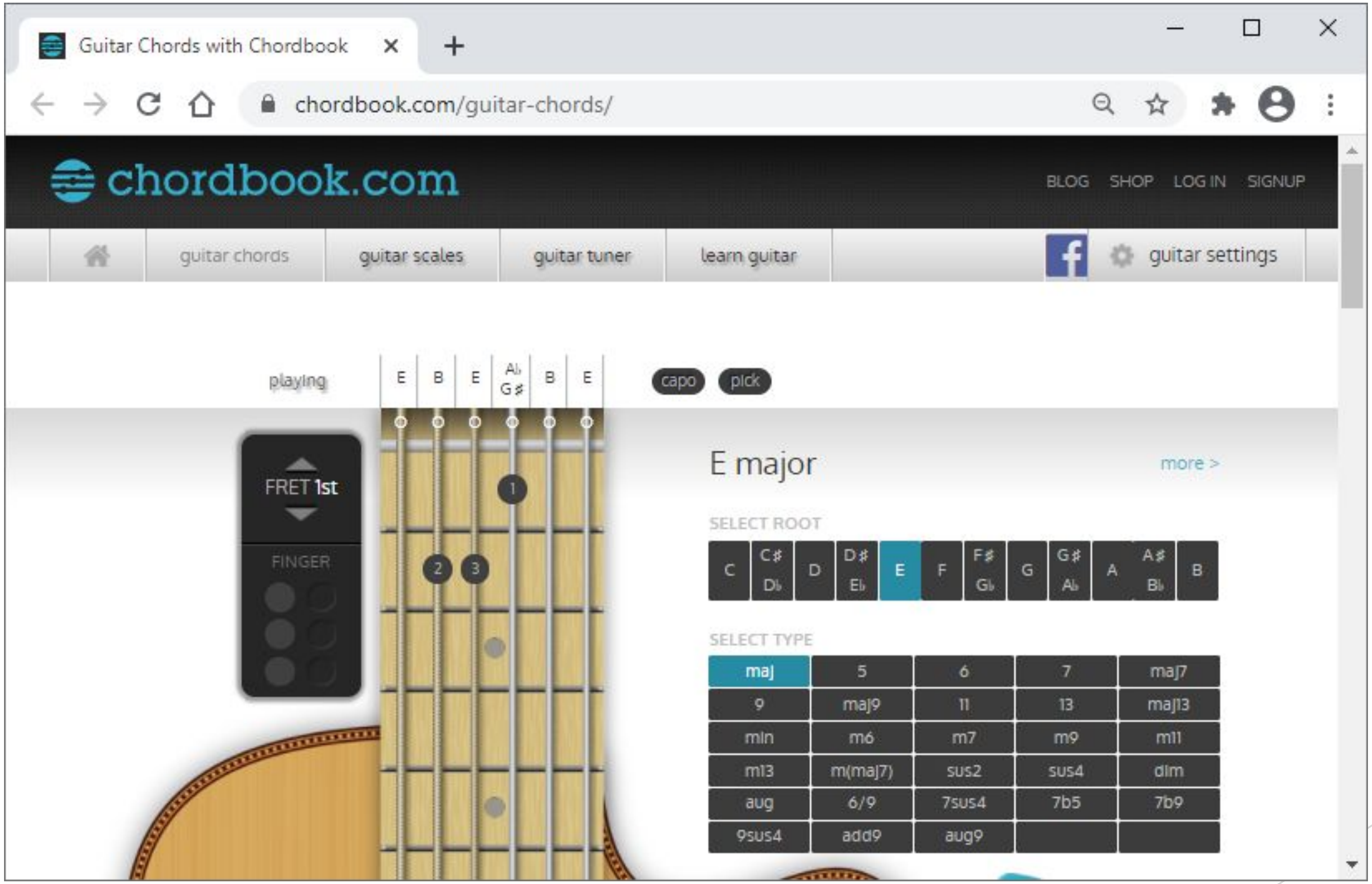



C:\kk\2020-2021\ConsoleApplication_detect_circles\x64\Debug\ConsoleA...

```
radius : 24
center : 330, 190
radius : 24
center : 326, 266
radius : 24
A  D  E
let's listen...
C:\Program Files (x86)\Windows Media Player>wmplayer.exe c:\
mp3
```

Listen:

To easily get guitar chords were some tools used



E major

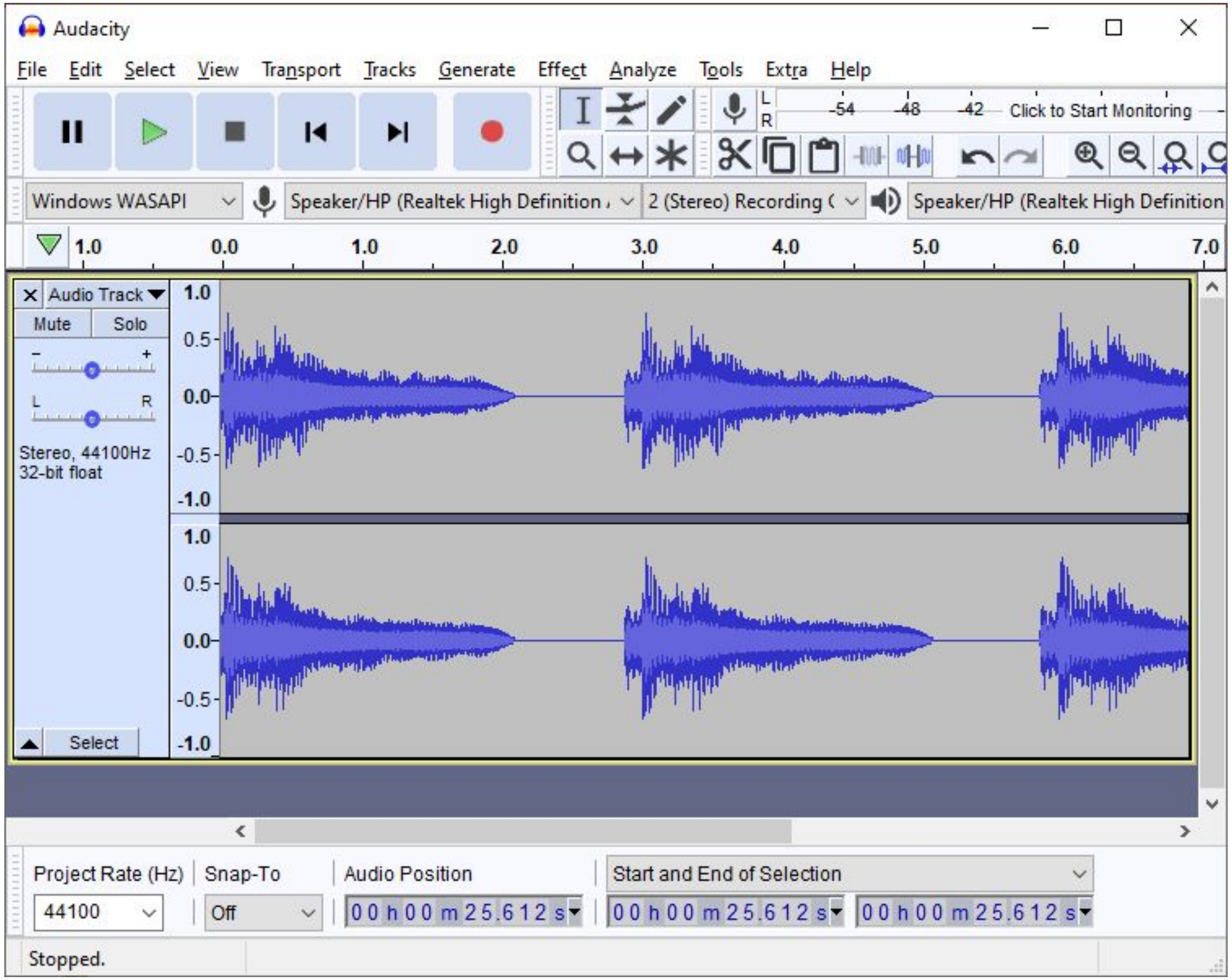
SELECT ROOT

C	C♯ D♭	D	D♯ E♭	E	F	F♯ G♭	G	G♯ A♭	A	A♯ B♭	B

SELECT TYPE

maj	5	6	7	maj7
9	maj9	11	13	maj13
min	m6	m7	m9	m11
m13	m(maj7)	sus2	sus4	dim
aug	6/9	7sus4	7b5	7b9
9sus4	add9	aug9		

To easily get guitar chords were some tools used



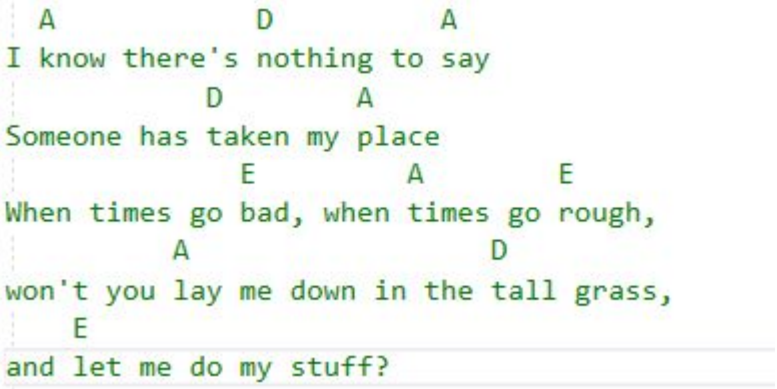
Stereo, 44100Hz
32-bit float

Project Rate (Hz): 44100
Snap-To: Off
Audio Position: 00 h 00 m 25.612 s
Start and End of Selection: 00 h 00 m 25.612 s 00 h 00 m 25.612 s

Stopped.

To easily get guitar chords were some tools used…

Sample here: A D and E chords were used:
Second Hand News



```
  A            D          A
I know there's nothing to say
            D        A
Someone has taken my place
              E         A        E
When times go bad, when times go rough,
          A                  D
won't you lay me down in the tall grass,
    E
and let me do my stuff?
```

Listen a sample