Open CV

vr basics contd.. (srikrishna sadula)

contents

Breif

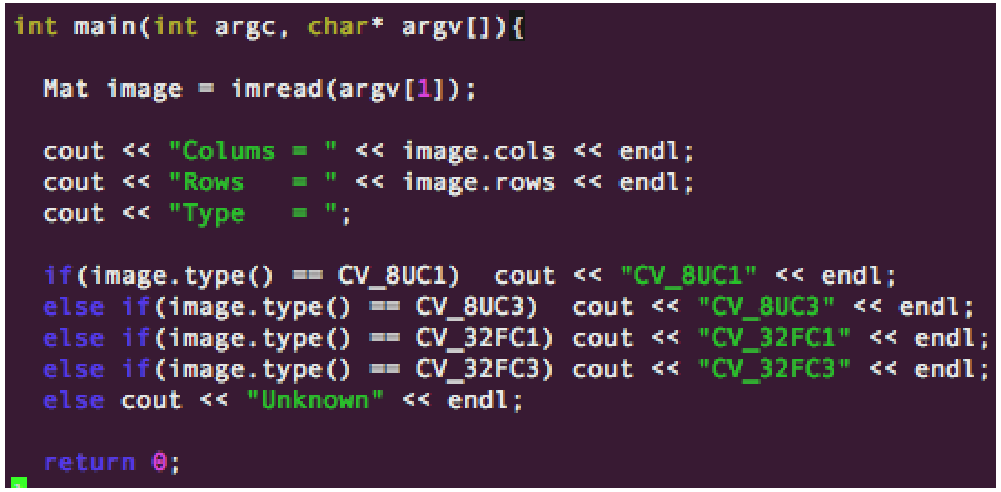
- Open CV is an Image processing library created by Intel and maitained by Willow Garage
- Available for C, C++, python, Java, C#
- Open source and free
- Easy to use and install

Basic opencv structures

2D point object - (x,y)
point.dot<point> computes dot product
point.inside<rect> returns if point is inside
Math operators: +, +=, -, -=, *, *=, ==, !=

size
-2D size structure (int width, int height)
point.area() - returns (width * height)

rect
2D rectangle structure
int x, y, width, height
point.tl - returns top left point
point.br - returns bottom right

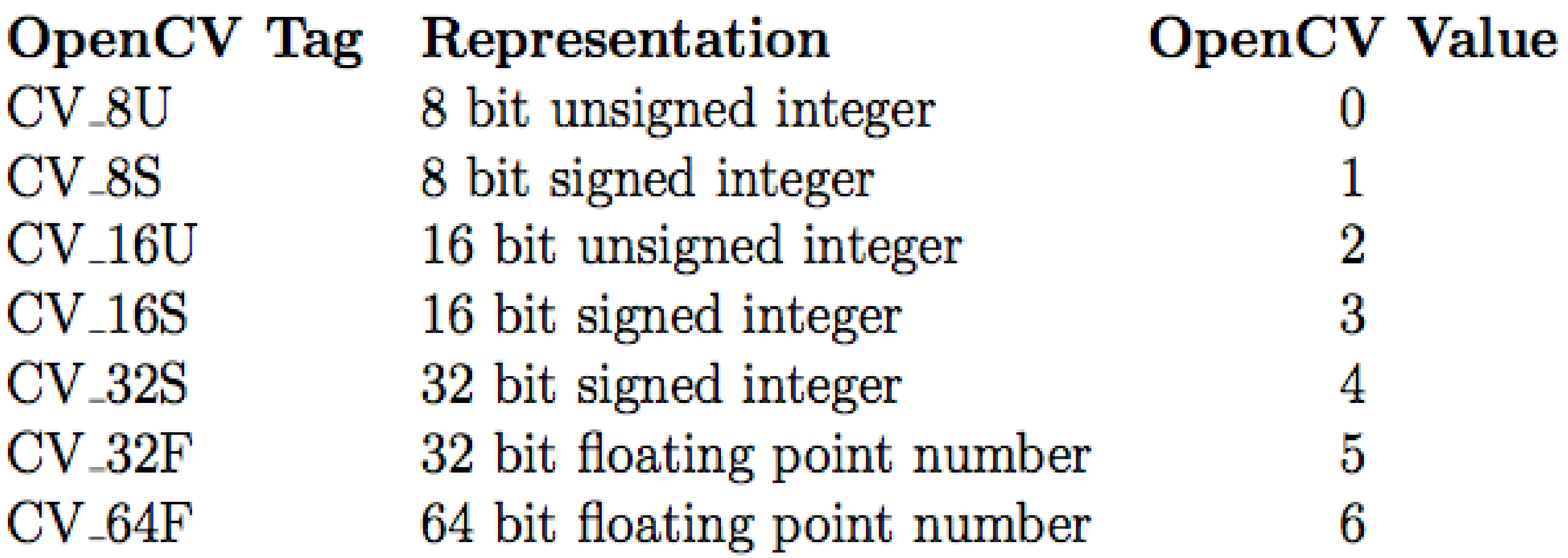


```
int main(int argc, char* argv[]){

  Mat image = imread(argv[1]);

  cout << "Colums = " << image.cols << endl;
  cout << "Rows   = " << image.rows << endl;
  cout << "Type   = ";

  if(image.type() == CV_8UC1)  cout << "CV_8UC1" << endl;
  else if(image.type() == CV_8UC3)  cout << "CV_8UC3" << endl;
  else if(image.type() == CV_32FC1) cout << "CV_32FC1" << endl;
  else if(image.type() == CV_32FC3) cout << "CV_32FC3" << endl;
  else cout << "Unknown" << endl;

  return 0;
```

Mat
Its an object which stores an image components
rows, cols, length & width
channels: grayscale, , rgb
Mat.at, Mat.channels, mat.clone, create, cross, depth, dot
Iterator usage(begin, end)

Image Types

- The TYPE is a very important aspect of OpenCV
- Represented as CV_<Datatype>C<# Channels>
- Example Datatypes/ Depths

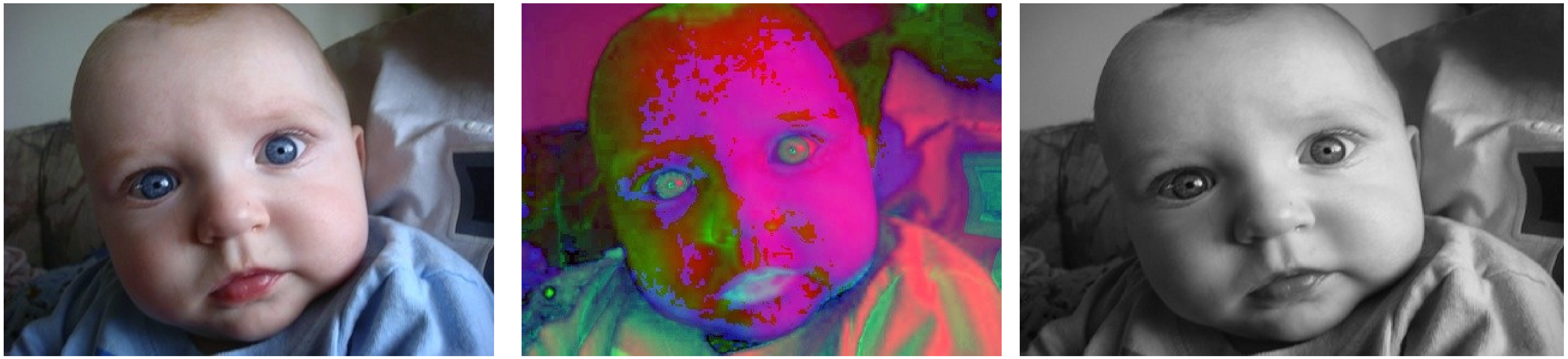


OpenCV Tag	**Representation**	**OpenCV Value**
CV_8U	8 bit unsigned integer	0
CV_8S	8 bit signed integer	1
CV_16U	16 bit unsigned integer	2
CV_16S	16 bit signed integer	3
CV_32S	32 bit signed integer	4
CV_32F	32 bit floating point number	5
CV_64F	64 bit floating point number	6

pixel types

- PixelTypes shows how the image is represented in data
  - BGR - The default color of imread(). Normal 3 channel color
  - HSV - Hue is color, Saturation is amount, Value is lightness. 3 channels
  - GRAYSCALE - Gray values, Single channel

- OpenCV requires that images be in BGR or Grayscale in order to be shown or saved. Otherwise, undesirable effects may appear.

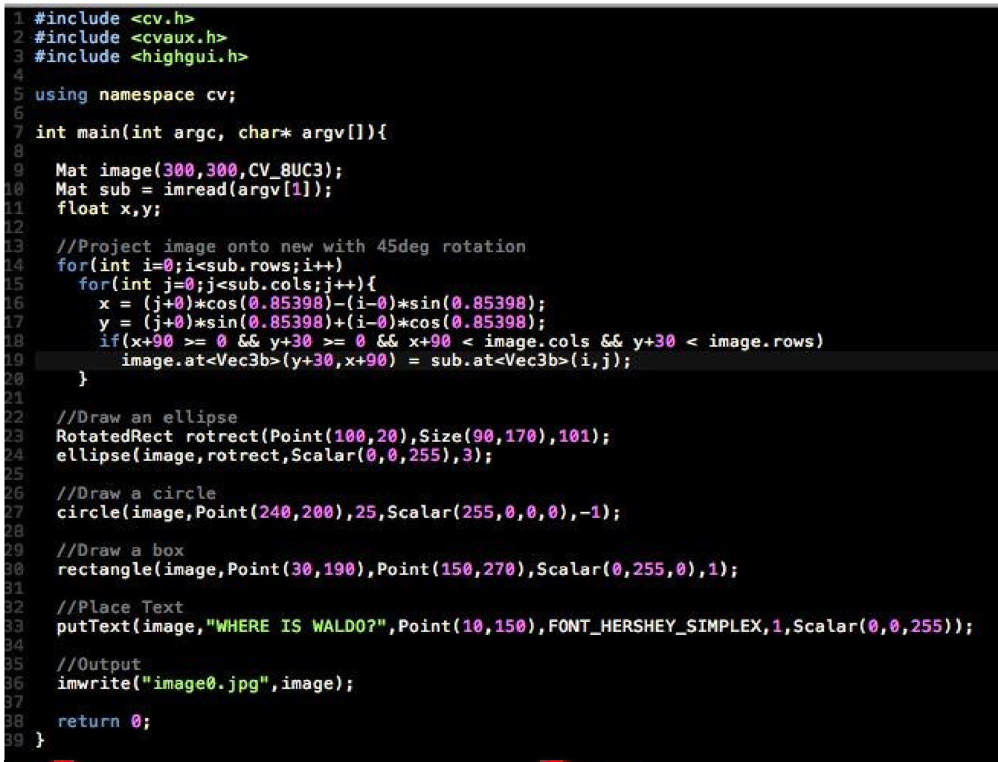


example code

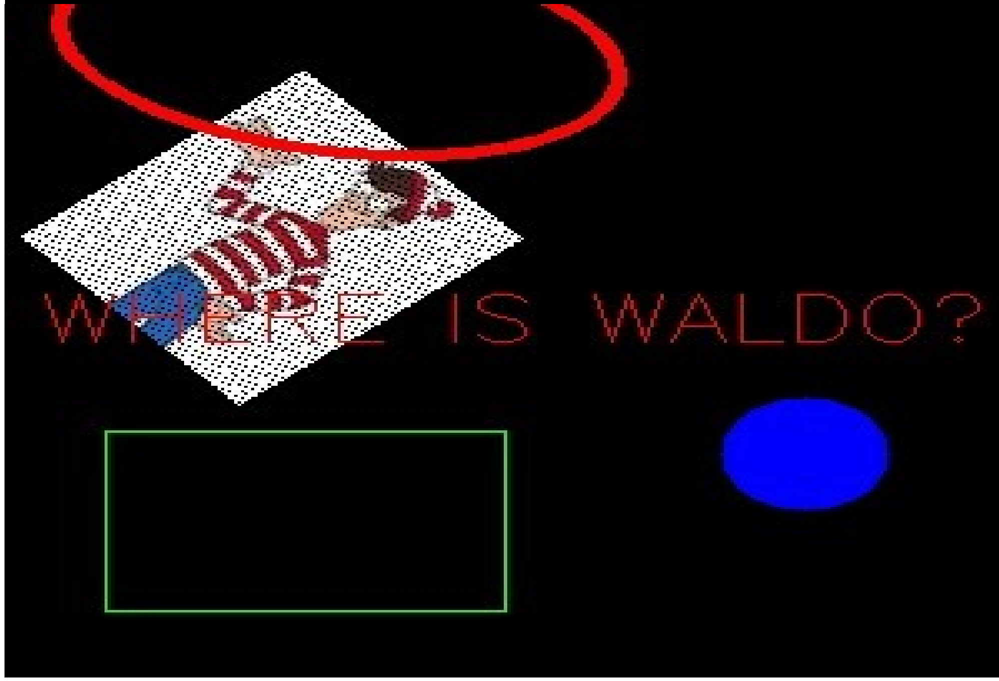
```
//Loads image and displays
//call by ./a.out image.jpg
//
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>

using namespace cv;

int main(int argc, char* argv[ ]){
   Mat image = imread(argv[1]);

   namedWindow(“Sample Window”);
   imshow(“Sample Window”,image);
   waitKey(0);
   return 0;
}
```

Drawing ??

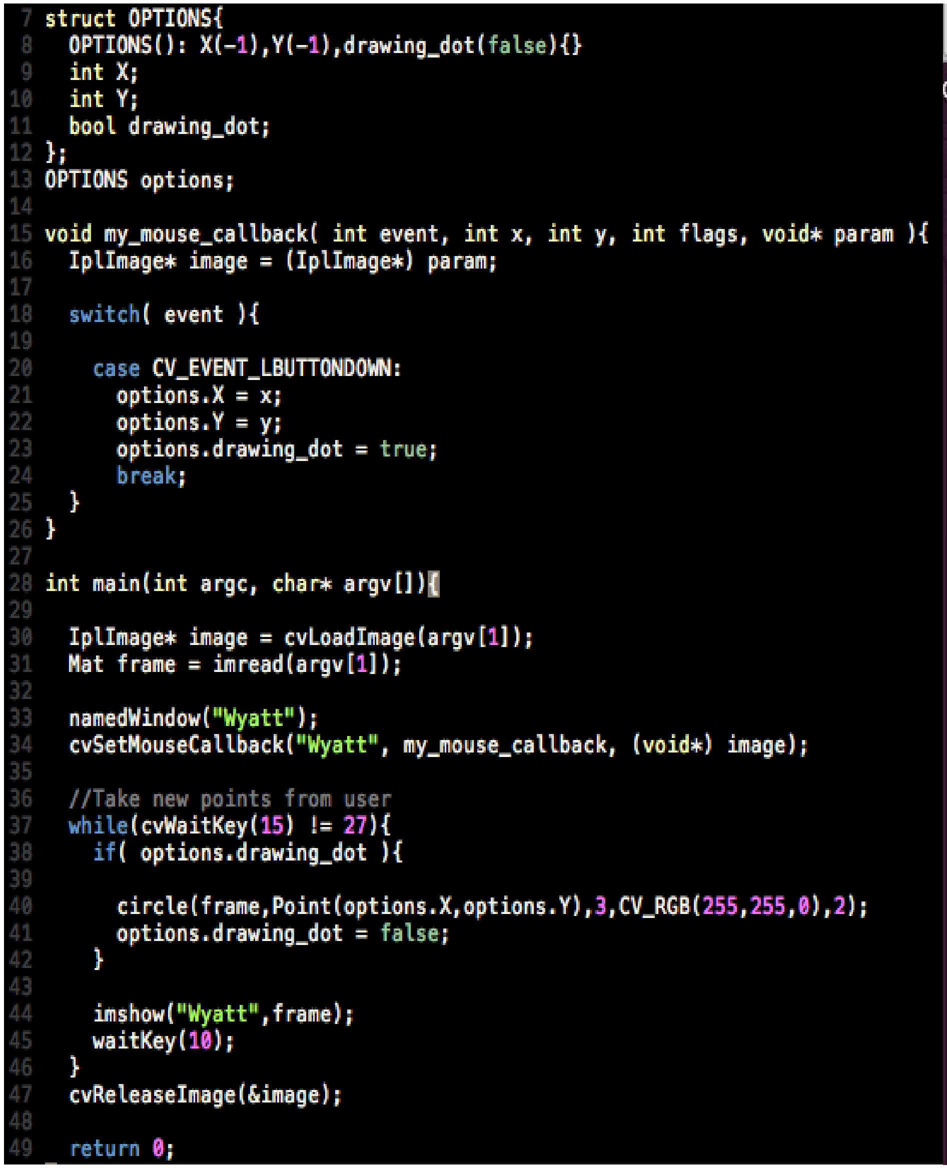
- Sometimes it is necessary to draw stuff onto the image. Instead of using complicated functions, why not just call a simple function?
- Here are some simple examples...
- void **circle**(image, Point(x,y),int rad, CV_BGR(b,g,r), int thickness=1)
- void **ellipse**(image, RotatedRect box, CV_BGR(b,g,r), int thickness=1)
- void **line**(image, Point(x,y), Point(x,y), CV_BGR(b,g,r), int thickness= 1)
- void **rectangle**(img, Point(x,y), Point(x,y), CV_BGR(b,g,r), int thickness)
  - NOTE: negative thickness will fill in the rectangle



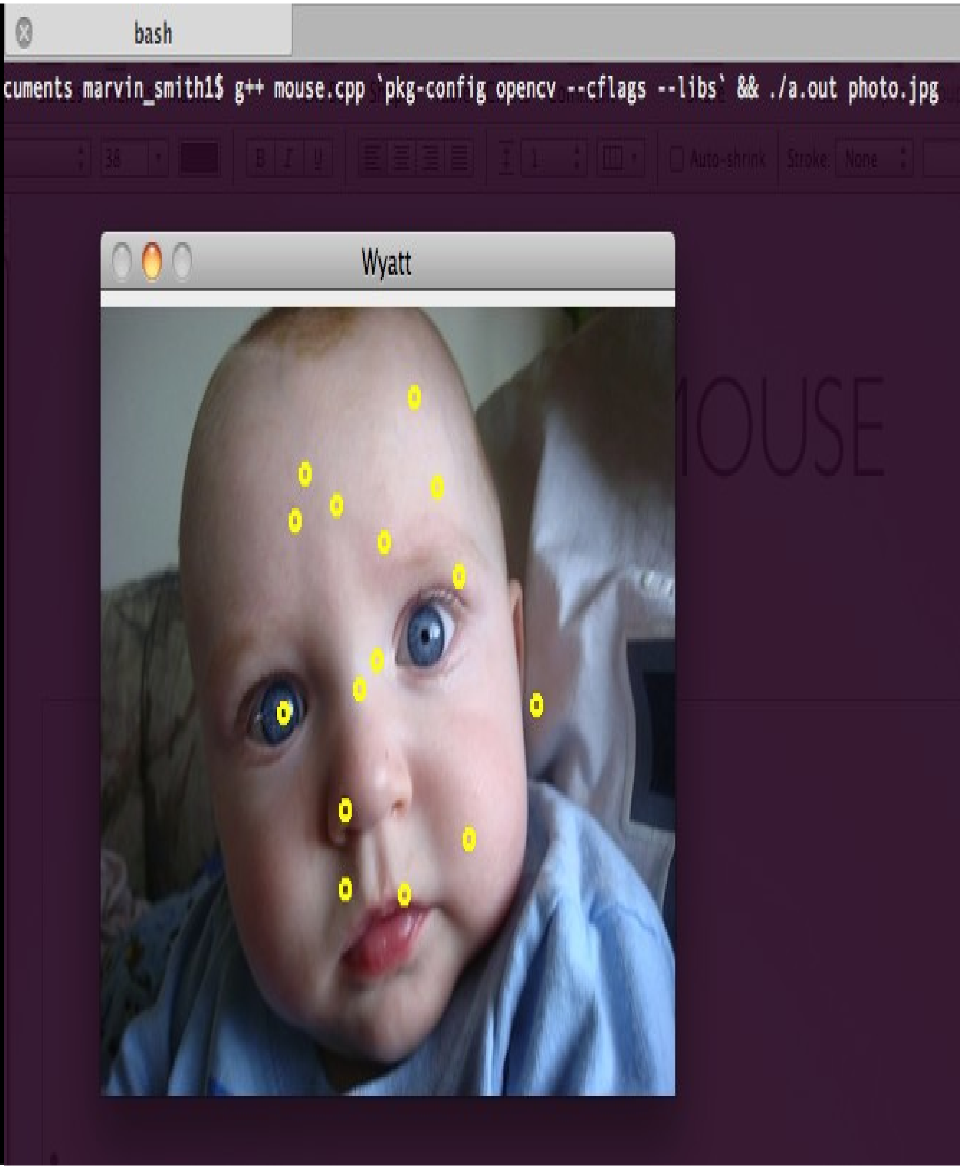
```
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>

using namespace cv;

int main(int argc, char* argv[]){

  Mat image(300,300,CV_8UC3);
  Mat sub = imread(argv[1]);
  float x,y;

  //Project image onto new with 45deg rotation
  for(int i=0;i<sub.rows;i++)
    for(int j=0;j<sub.cols;j++){
      x = (j+0)*cos(0.85398)-(i-0)*sin(0.85398);
      y = (j+0)*sin(0.85398)+(i-0)*cos(0.85398);
      if(x+90 >= 0 && y+30 >= 0 && x+90 < image.cols && y+30 < image.rows)
        image.at<Vec3b>(y+30,x+90) = sub.at<Vec3b>(i,j);
    }

  //Draw an ellipse
  RotatedRect rotrect(Point(100,20),Size(90,170),101);
  ellipse(image,rotrect,Scalar(0,0,255),3);

  //Draw a circle
  circle(image,Point(240,200),25,Scalar(255,0,0,0),-1);

  //Draw a box
  rectangle(image,Point(30,190),Point(150,270),Scalar(0,255,0),1);

  //Place Text
  putText(image,"WHERE IS WALDO?",Point(10,150),FONT_HERSHEY_SIMPLEX,1,Scalar(0,0,255));

  //Output
  imwrite("image0.jpg",image);

  return 0;
}
```



Using the mouse



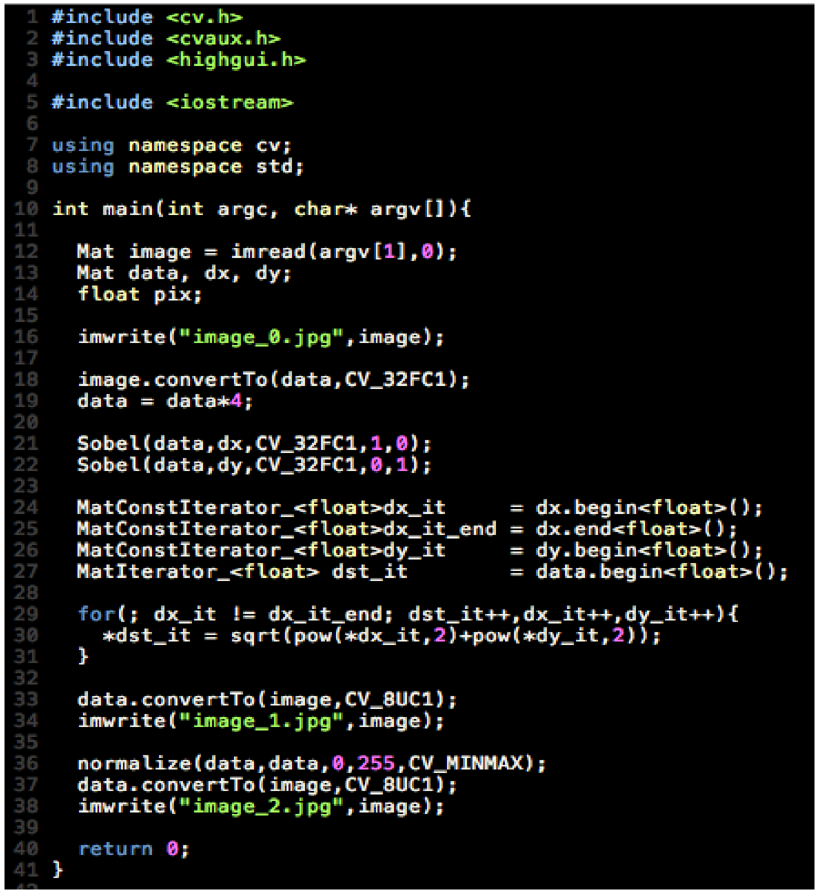
```
struct OPTIONS{
  OPTIONS(): X(-1),Y(-1),drawing_dot(false){}
  int X;
  int Y;
  bool drawing_dot;
};
OPTIONS options;

void my_mouse_callback( int event, int x, int y, int flags, void* param ){
  IplImage* image = (IplImage*) param;

  switch( event ){

    case CV_EVENT_LBUTTONDOWN:
      options.X = x;
      options.Y = y;
      options.drawing_dot = true;
      break;
  }
}

int main(int argc, char* argv[]){

  IplImage* image = cvLoadImage(argv[1]);
  Mat frame = imread(argv[1]);

  namedWindow("Wyatt");
  cvSetMouseCallback("Wyatt", my_mouse_callback, (void*) image);

  //Take new points from user
  while(cvWaitKey(15) != 27){
    if( options.drawing_dot ){

      circle(frame,Point(options.X,options.Y),3,CV_RGB(255,255,0),2);
      options.drawing_dot = false;
    }

    imshow("Wyatt",frame);
    waitKey(10);
  }
  cvReleaseImage(&image);

  return 0;
```



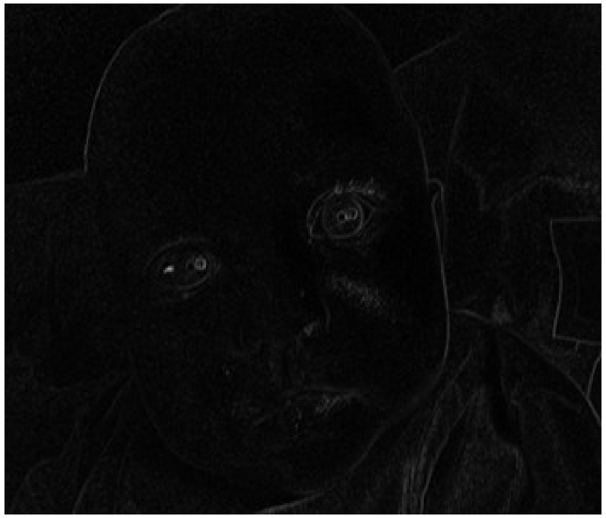
```
cuments marvin_smith1$ g++ mouse.cpp `pkg-config opencv --cflags --libs` && ./a.out photo.jpg
```

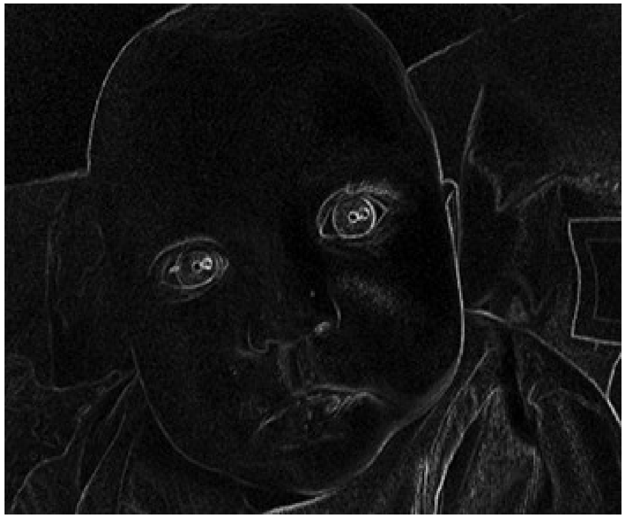
image normalization

- **normalize**(imagein, imageout, low, high, method);
- Image normalization is the process of stretching the range of an image from [a, b] to [c, d].
- This is incredibly important for visualization because if the image is beyond [0,255] it will cause truncation or unsightly effects.



```
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>

#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char* argv[]){

  Mat image = imread(argv[1],0);
  Mat data, dx, dy;
  float pix;

  imwrite("image_0.jpg",image);

  image.convertTo(data,CV_32FC1);
  data = data*4;

  Sobel(data,dx,CV_32FC1,1,0);
  Sobel(data,dy,CV_32FC1,0,1);

  MatConstIterator_<float>dx_it     = dx.begin<float>();
  MatConstIterator_<float>dx_it_end = dx.end<float>();
  MatConstIterator_<float>dy_it     = dy.begin<float>();
  MatIterator_<float> dst_it        = data.begin<float>();

  for(; dx_it != dx_it_end; dst_it++,dx_it++,dy_it++){
    *dst_it = sqrt(pow(*dx_it,2)+pow(*dy_it,2));
  }

  data.convertTo(image,CV_8UC1);
  imwrite("image_1.jpg",image);

  normalize(data,data,0,255,CV_MINMAX);
  data.convertTo(image,CV_8UC1);
  imwrite("image_2.jpg",image);

  return 0;
}
```





thresholding

- threshold(image, image, thresh, maxVal, CODE);
- CODE - this is the method of thresholding. Different actions will be taken depending on this code.

- **THRESH_BINARY**

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{maxVal} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_BINARY_INV**

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{maxVal} & \text{otherwise} \end{cases}$$

- **THRESH_TRUNC**

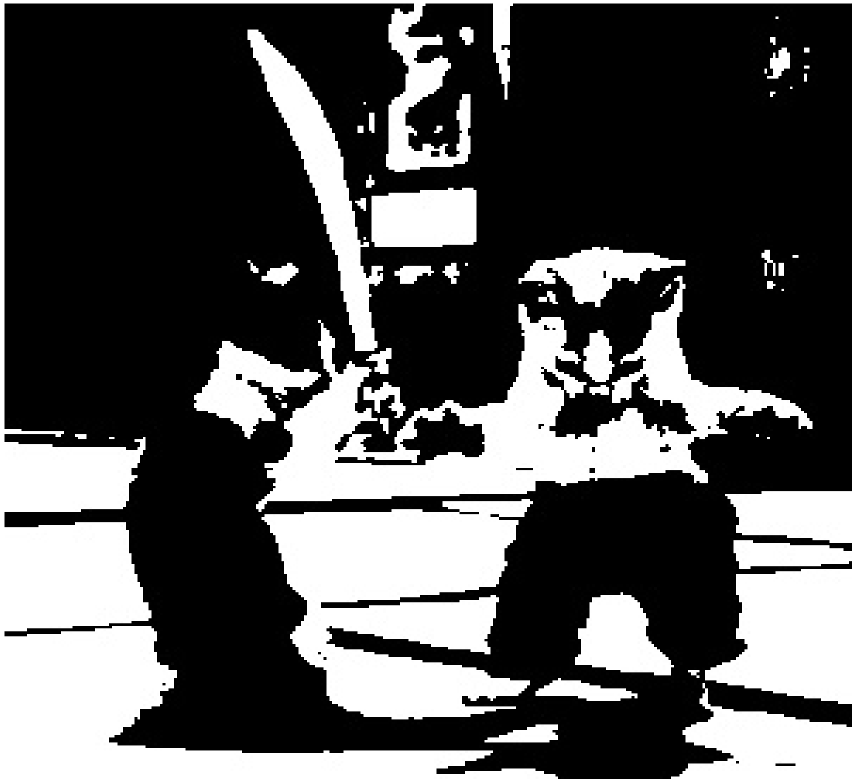
$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{threshold} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO**

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{src}(x,y) & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO_INV**

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases}$$



Edge detection

Building the library

- cmake
- face detection program
- Integrating with Unity

Thanks

- Q & A