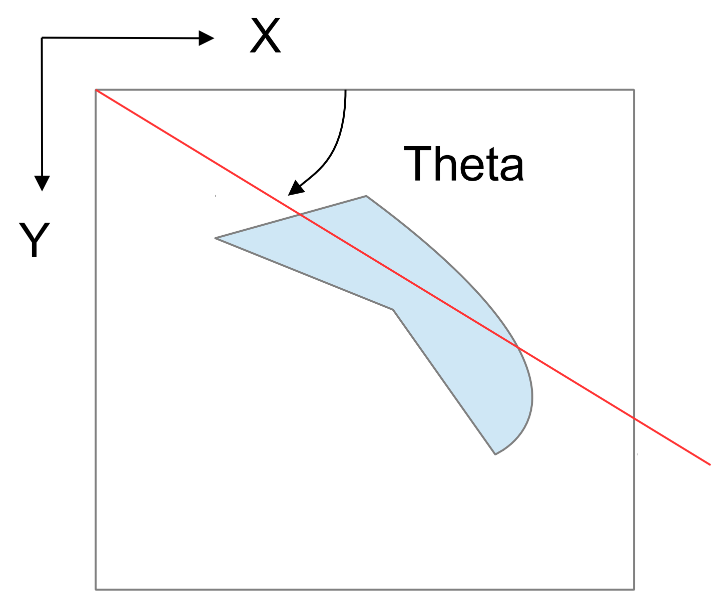
Orientation Computation

$$\tan 2\phi \triangleq \frac{b}{a-c}$$

$$a = \iint_{\Omega} (x-\bar{x})^2 B(x,y)\,dx\,dy \quad \cdots (2)$$

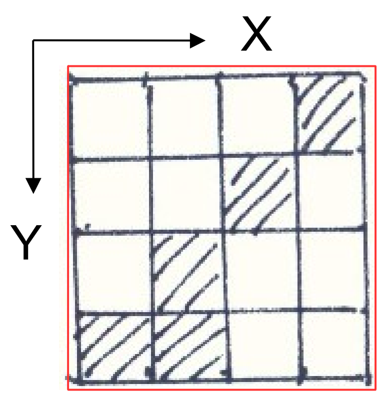
$$b = \iint_{\Omega} 2(x-\bar{x})(y-\bar{y}) B(x,y)\,dx\,dy \quad \cdots (3)$$

$$c = \iint_{\Omega} (y-\bar{y})^2 B(x,y)\,dx\,dy \quad \cdots (4)$$



Reference: Robot Vision, by BPK, Horn, Chapter 3, pp. 46-64

Example: See my handout



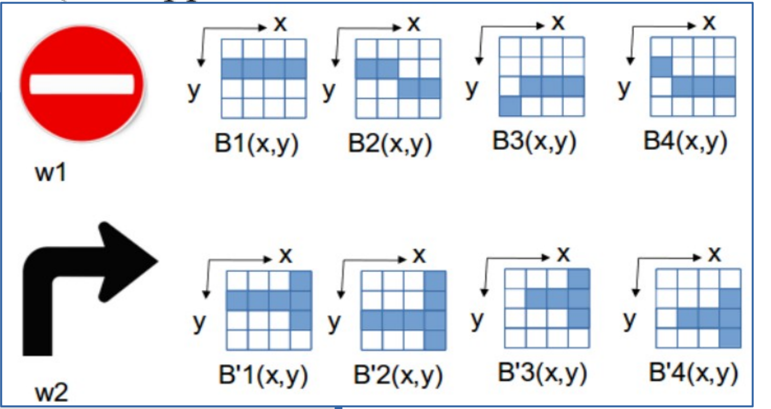
a = 7
b = -8
c = 6
Theta = -41.4375

Note: my hand calculation use integer, when have access to computer, use Float!
(x_bar = 2.8 changed to 3, and y_bar = 2.4 changed to 2)

Computation of Moments

QUESTION 3 (15 Points) Given two traffic signs and their binarized images taken from different conditions as shown in the following figure, design a machine learning technique by answering the following questions:

5.1 (5 pts) Based on given 2 classes of image, find moments m_01, m_10 for each of the image, and form feature vector space with your computation result (see Appendix for m_pq definition if needed).



	W_{11}	W_{12}	W_{13}	W_{14}	W_{21}	W_{22}	W_{23}	W_{24}
u_y/m_{01}	2/0	2/0	3/0	2/0	2/0	(19/4)/0	(8/5)/(7/5)	3/0
u_x/m_{10}	2.5/0	2.5/0	2.5/0	2.5/0	(18/6)/0	(22/7)/(-9/7)	(17/5)/0	(17/5)/0

$m_{10,23} = (4-17/5)\times 2 + (2-\frac{17}{5}) + (3-\frac{17}{5}) + (4-\frac{17}{5})$

$= \frac{6}{5} + (-\frac{7}{5}) + (-\frac{2}{5}) + \frac{3}{5} = 0$

$m_{10,24} = (4-17/5)\times 2 + (2-\frac{17}{5}) + (3-\frac{17}{5}) + (4-\frac{17}{5}) = 0$

For Class 2, $W_{2j}, j=1,2,3,4$.

$u_{x21} = [4+(1+2+3+4)+4]\frac{1}{A} = 18/6$

$u_{x22} = [4\times 3+(1+2+3+4)]\frac{1}{A} = 22/7$

$u_{x23} = [4\times 2+(2+3+4)]\frac{1}{A} = 17/5.$

$u_{x24} = [4\times 2+(2+3+4)]\frac{1}{A} = 17/5$

$m_{10,21} = [(4-18/6)+(1-\frac{18}{6})+(2-\frac{18}{6})+(3-\frac{18}{6})+(4-\frac{18}{6})+(4-\frac{18}{6})]\frac{1}{A}$

$= [\frac{6}{6} - \frac{12}{6} - \frac{6}{6} + 0 + \frac{6}{6} + \frac{6}{6}]\frac{1}{6} = 0$

$m_{10,22} = [(4-22/7)\times 3 + (1-\frac{22}{7}) + (2-\frac{22}{7}) + (3-\frac{22}{7}) + (4-\frac{22}{7})]\frac{1}{A}$

$= (\frac{45}{7} - \frac{15}{7} - \frac{8}{7} - \frac{1}{7} + \frac{6}{7})\frac{1}{A} = (-\frac{63}{7})\frac{1}{A} = (-9)\frac{1}{7}$

(2) PART II. K-mean Cluster Algorithm.
Use (u_x, u_y) to form feature vectors.
Then, apply OpenCV. K-mean

Python Example For Moments

First, let's find contours, by openCV.org definition, “Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.”

Note: In OpenCV, object to be found should be white and background should be black when applying contour finding function.

cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

The arguments: the 1st is source image, 2nd is contour retrieval mode, 3rd is contour approximation method. And it outputs the contours and hierarchy. contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.

```
im = cv2.imread('test.jpg')
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray,127,255,0)
im2, contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

Contours

actly what a contour is. A contour is a list of points that represent, in one way or another, a curve in an image. This representation can be different depending on the circumstance at hand. There are many ways to represent a curve. Contours are represented in OpenCV by sequences in which every entry in the sequence encodes information about the location of the next point on the curve. We will dig into the details of such

Reference: Learning OpenCV, pp. 250

The function cvFindContours() computes contours from binary images. It can take images created by cvCanny(), which have edge pixels in them, or images created by functions like cvThreshold() or cvAdaptiveThreshold(), in which the edges are implicit as boundaries between positive and negative regions.*

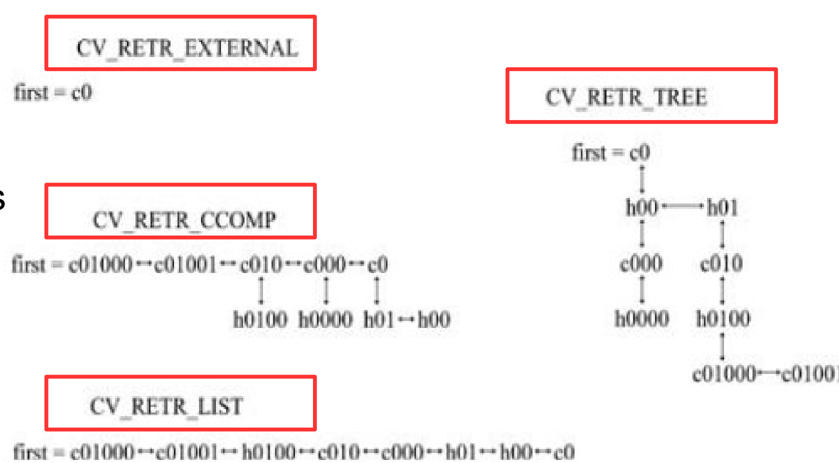
Contours Mode Variable

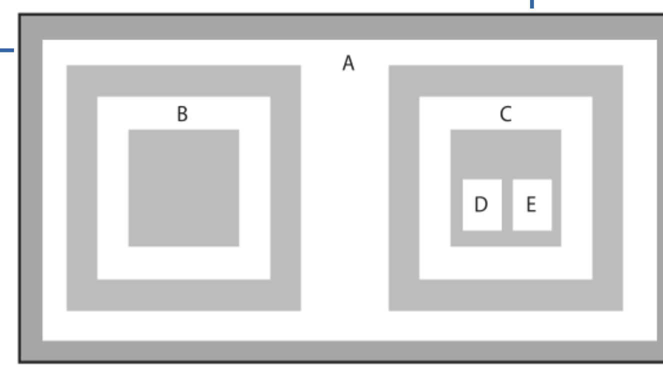
findContours(image_canny, contours, hierarchy,
RETR_CCOMP, CHAIN_APPROX_SIMPLE);

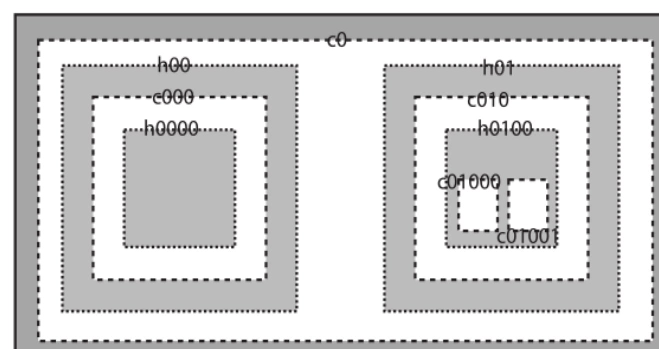
The mode variable can be set to any of four options: CV_RETR_EXTERNAL, CV_RETR_LIST, CV_RETR_CCOMP, or CV_RETR_TREE. The value of mode indicates to cvFindContours() exactly what contours we would like found and how we would like the result presented to us. In particular, the manner in which the tree node variables (h_prev, h_next, v_prev, and v_next) are used to "hook up" the found contours is determined by the value of mode. In Figure 8-3, the resulting topologies are shown for all four possible values of mode. In every case, the structures can be thought of as "levels" which are related by the "horizontal" links (h_next and h_prev), and those levels are separated from one another by the "vertical" links (v_next and v_prev).

Retrieves only the extreme outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.

CV_RETR_CCOMP
Retrieves all the contours into two-level hierarchy, top-level for external boundaries and the 2nd level for the holes.







retrieves all contours without any hierarchical relationships.

http://opencvexamples.blogspot.com/2013/09/find-contour.html

Contours Mode Variable

http://opencvexamples.blogspot.com/2013/09/find-contour.html

```
void drawContours(
        InputOutputArray image,
        InputArrayOfArrays contours,
        int contourIdx,
        const Scalar& color,
        int thickness=1,
        int lineType=8,
        InputArray hierarchy=noArray(),
        int maxLevel=INT_MAX,
        Point offset=Point() )
```

hierarchy – Output vector, containing contour topology. It has as many elements as the number of contours. For each i-th contour contours[i] , the elements hierarchy[i][0] , hiearchy[i][1] , hiearchy[i][2] , and hiearchy[i][3] are set to 0-based indices in contours of the next and previous contours at the same hierarchical level, the first child contour and the parent contour, respectively. If for the contour i there are no next, previous, parent, or nested contours, the corresponding elements of hierarchy[i] will be negative.

contourIdx – Parameter indicating a contour to draw. If it is negative, all the contours are drawn.

maxLevel – Maximal level for drawn contours. If it is 0, only the specified contour is drawn. If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available.

Canny Edge Detector

https://en.wikipedia.org/wiki/Canny_edge_detector

For (2K+1) by (2K+1) Gaussian kernel:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right);$$

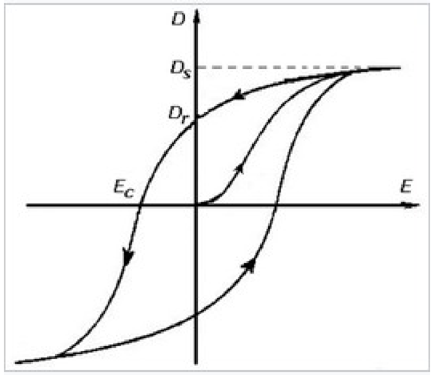
Canny edge detection algorithm in 5 steps:
1. Apply Gaussian filter to remove noise;
2. Compute intensity gradients;
3. Apply non-maximum suppression to get rid of spurious response to edge detection;
4. Apply double thresholds to potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing weak and not connected edges.

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x),$$

Edge gradient

Hysteresis is the dependence of the state of a system on its history. For example, a magnet may have more than one possible magnetic moment in a given magnetic field, depending on its past. Plots of a single component of the moment often form a loop or hysteresis curve, different values of one variable depending on the direction of change of another variable.



https://en.wikipedia.org/wiki/Hysteresis

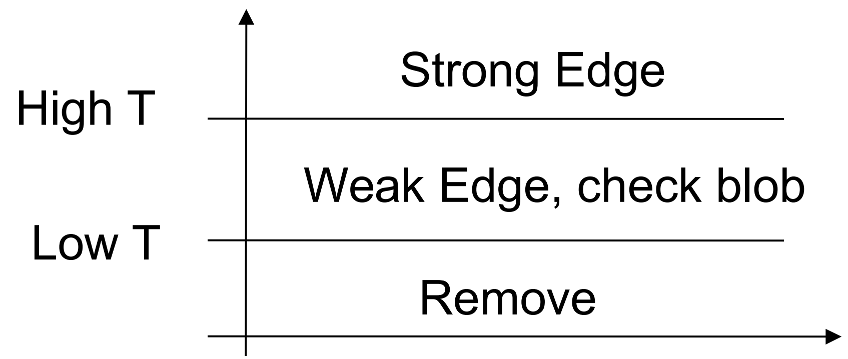
The edge direction is rounded to one of 4 angles (0°, 45°, 90° and 135°). An edge direction will be set to a specific angle values, for instance θ in [0°, 22.5°] or [157.5°, 180°] maps to 0°.

Harry Li, Ph.D.

Double Threshold And Blob

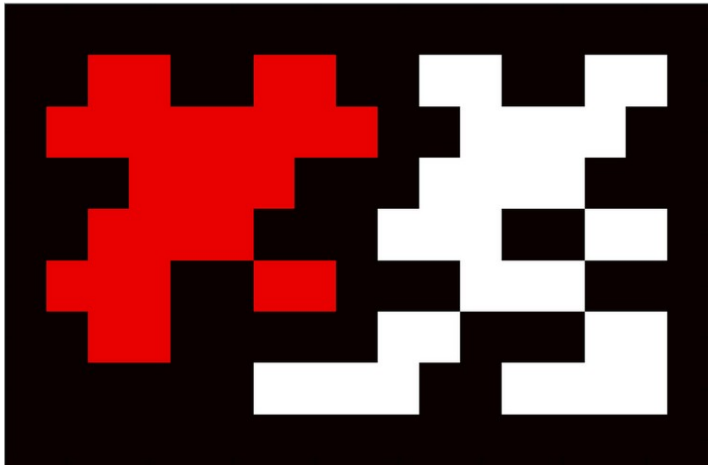
https://en.wikipedia.org/wiki/Canny_edge_detector

Double threshold: some edge pixels are caused by noise and color variation, e.g., spurious responses, so remove those with weak gradient and keep those with a high gradient by selecting high and low threshold values. If an edge pixel’s gradient value is higher than the high threshold value, it is marked as a strong edge pixel. If an edge pixel’s gradient value is smaller than the high threshold but higher than the low threshold, it is marked as a weak edge pixel. If an edge pixel's value is smaller than the low threshold value, it will be suppressed.



Blob algorithm: similar to flood-fill algorithm, based on 4-connected or 8-connected neighbors

https://en.wikipedia.org/wiki/Connected-component_labeling

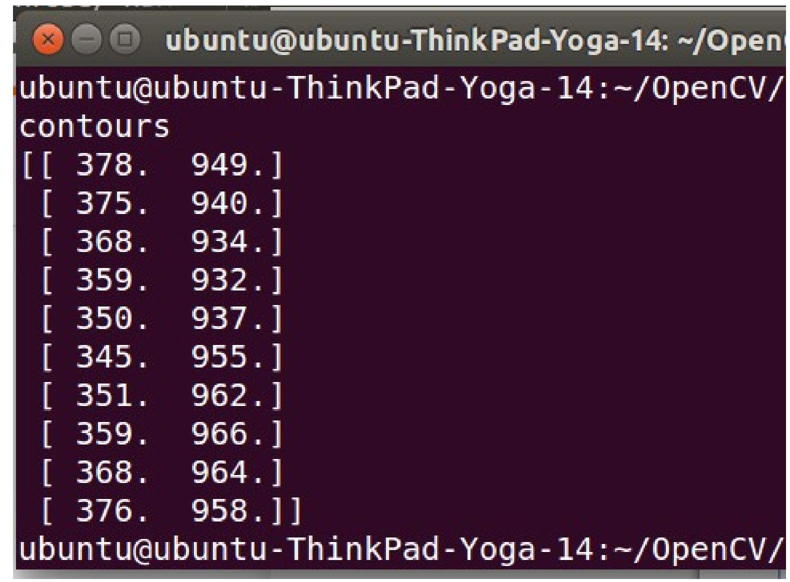


Harry Li, Ph.D.

Contours Data Type In Python

https://stackoverflow.com/questions/20928944/create-contour-from-scratch-in-python-opencv-cv2

```
#------------------------------------#
# program: contour-test.py     #
# tested by: HL                     #
import cv2, numpy
contour = numpy.array( [
(378, 949), (375, 940), (368, 934),
(359, 932), (350, 937), (345, 955),
(351, 962), (359, 966), (368, 964),
(376, 958) ], numpy.float32 )
cv2.isContourConvex(contour)
print ('contours')
print (contour)
```



Compute Contours Features

https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

1. Moments

```
import cv2
import numpy as np

img = cv2.imread('star.jpg',0)
ret,thresh = cv2.threshold(img,127,255,0)
contours,hierarchy = cv2.findContours(thresh, 1, 2)

cnt = contours[0]
M = cv2.moments(cnt)
print M
```

2. Contour Area

```
area = cv2.contourArea(cnt)
```

3. Contour Perimeter

```
perimeter = cv2.arcLength(cnt,True)
```

4. Contour Approximation

```
epsilon = 0.1*cv2.arcLength(cnt,True)
approx = cv2.approxPolyDP(cnt,epsilon,True)
```

5. Convex Hull

Convexity defects

checks a curve for convexity defects and corrects it

```
hull = cv2.convexHull(cnt)
```

6. Checking Convexity

```
k = cv2.isContourConvex(cnt)
```

7.a. Straight Bounding Rectangle

```
x,y,w,h = cv2.boundingRect(cnt)
cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

7.b. Rotated Rectangle

```
rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.int0(box)
cv2.drawContours(img,[box],0,(0,0,255),2)
```

Compute Contours Features

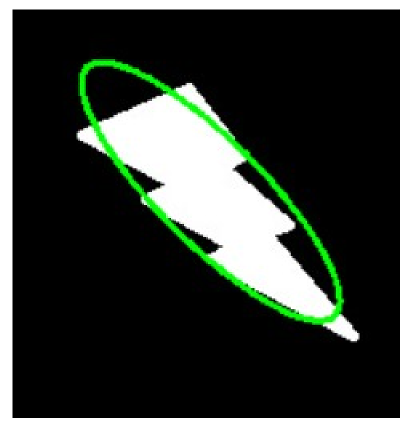
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

8. Minimum Enclosing Circle

```
(x,y),radius = cv2.minEnclosingCircle(cnt)
center = (int(x),int(y))
radius = int(radius)
cv2.circle(img,center,radius,(0,255,0),2)
```

9. Fitting an Ellipse

```
ellipse = cv2.fitEllipse(cnt)
cv2.ellipse(img,ellipse,(0,255,0),2)
```



http://nicky.vanforeest.com/misc/fitEllipse/fitEllipse.html

10. Fitting a Line

```
rows,cols = img.shape[:2]
[vx,vy,x,y] = cv2.fitLine(cnt, cv2.DIST_L2,0,0.01,0.01)
lefty = int((-x*vy/vx) + y)
righty = int(((cols-x)*vy/vx)+y)
cv2.line(img,(cols-1,righty),(0,lefty),(0,255,0),2)
```

From Contour Find Shapes

https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

4. Contour Approximation

```
1 epsilon = 0.1*cv2.arcLength(cnt,True)
2 approx = cv2.approxPolyDP(cnt,epsilon,True)
```

5. Convex Hull Convexity defects

checks a curve for convexity defects and corrects it

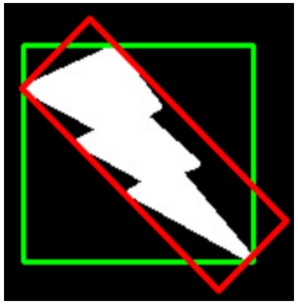
```
hull = cv2.convexHull(cnt)
```

7.a. Straight Bounding Rectangle

```
1 x,y,w,h = cv2.boundingRect(cnt)
2 cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

7.b. Rotated Rectangle

```
1 rect = cv2.minAreaRect(cnt)
2 box = cv2.boxPoints(rect)
3 box = np.int0(box)
4 cv2.drawContours(img,[box],0,(0,0,255),2)
```

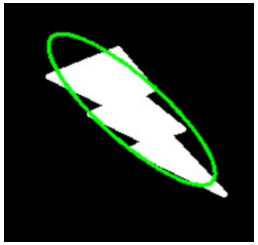


8. Minimum Enclosing Circle

```
1 (x,y),radius = cv2.minEnclosingCircle(cnt)
2 center = (int(x),int(y))
3 radius = int(radius)
4 cv2.circle(img,center,radius,(0,255,0),2)
```

9. Fitting an Ellipse

```
1 ellipse = cv2.fitEllipse(cnt)
2 cv2.ellipse(img,ellipse,(0,255,0),2)
```



Harry Li, Ph.D

Contour-Shapes Properties

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html

11. Aspect Ratio

$$Aspect\ Ratio = \frac{Width}{Height}$$

```
x,y,w,h = cv2.boundingRect(cnt)
aspect_ratio = float(w)/h
```

12. Extent

$$Extent = \frac{Object\ Area}{Bounding\ Rectangle\ Area}$$

```
area = cv2.contourArea(cnt)
x,y,w,h = cv2.boundingRect(cnt)
rect_area = w*h
extent = float(area)/rect_area
```

13. Solidity

$$Solidity = \frac{Contour\ Area}{Convex\ Hull\ Area}$$

```
area = cv2.contourArea(cnt)
hull = cv2.convexHull(cnt)
hull_area = cv2.contourArea(hull)
solidity = float(area)/hull_area
```

14. Equivalent Diameter

$$Equivalent\ Diameter = \sqrt{\frac{4 \times Contour\ Area}{\pi}}$$

```
area = cv2.contourArea(cnt)
equi_diameter = np.sqrt(4*area/np.pi)
```

15. Orientation

Following method also gives the Major Axis and Minor Axis lengths.

```
(x,y),(MA,ma),angle = cv2.fitEllipse(cnt)
```

Contour Mask And Pixel Points

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html

```
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(imgray,mask = mask)
```

16. Mask and Pixel Points

All the points comprises that object (contour)

```
mask = np.zeros(imgray.shape,np.uint8)
cv2.drawContours(mask,[cnt],0,255,-1)
pixelpoints = np.transpose(np.nonzero(mask))
#pixelpoints = cv2.findNonZero(mask)
```

Above, "two methods, one using Numpy functions, next one using OpenCV function (last commented line) are given to do the same. Results are also same, but with a slight difference. Numpy gives coordinates in (row, column) format, while OpenCV gives coordinates in (x,y) format. So basically the answers will be interchanged. Note that, row = x and column = y."

17. Maximum Value, Minimum Value and their locations

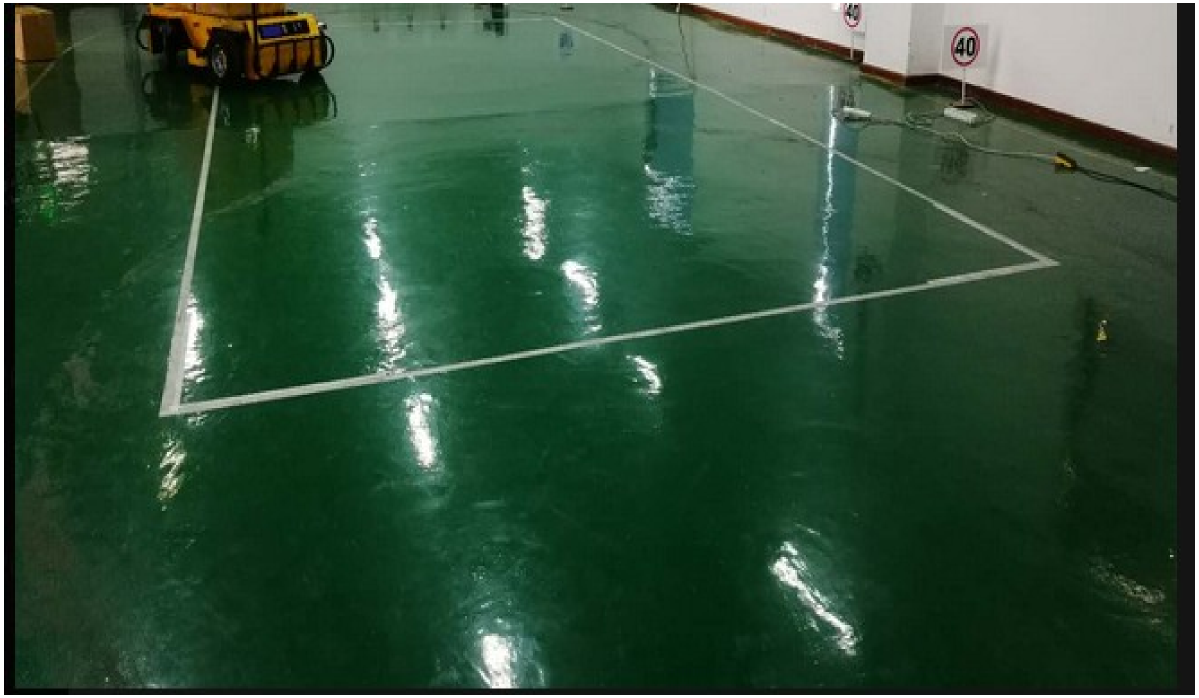
18. Mean Color or Mean Intensity

```
mean_val = cv2.mean(im,mask = mask)
```

19. Extreme Points

```
leftmost = tuple(cnt[cnt[:,:,0].argmin()][0])
rightmost = tuple(cnt[cnt[:,:,0].argmax()][0])
topmost = tuple(cnt[cnt[:,:,1].argmin()][0])
bottommost = tuple(cnt[cnt[:,:,1].argmax()][0])
```

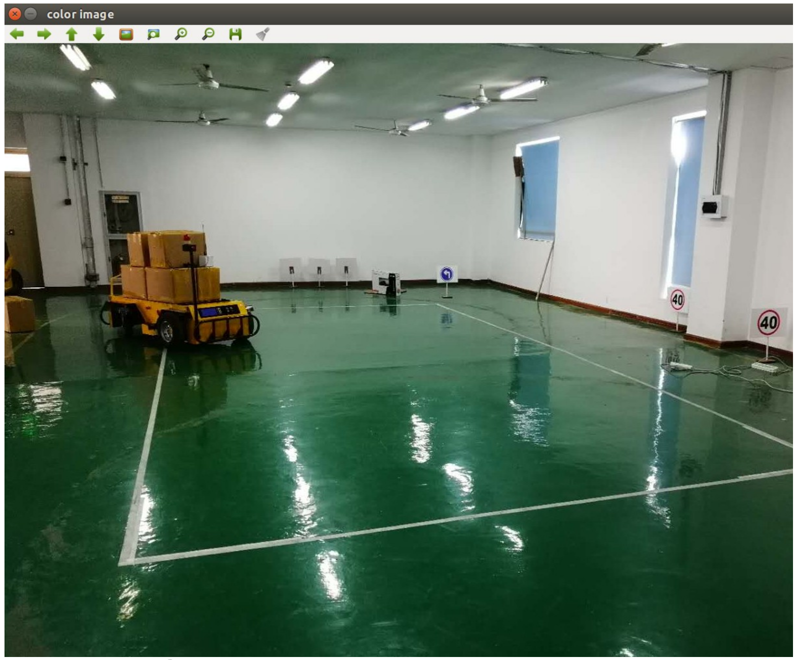
Example Separation of Floor Track



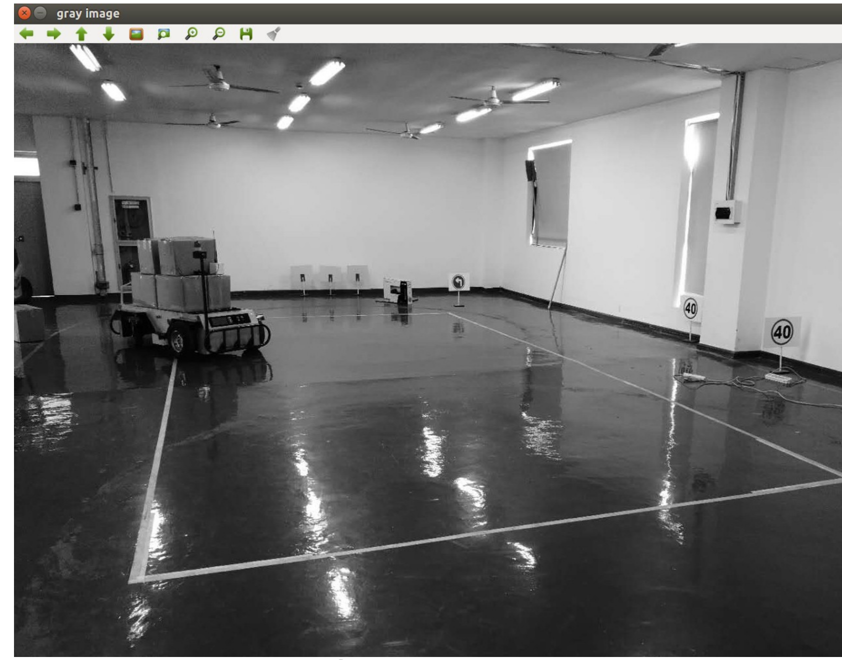
From Shapes && Colors Find ROI And Remove Reflections

	Ceiling Lights (class w1)	Window Lights (class w2)
Shape	Rectangles x1	Rectangles x1
	Ellipses x2	Ellipses x2
	Circles x3	Circles x3
Location	Anywhere x4 smaller part image x5	Anywhere x4 smaller part image x5
Color	white x6	white x6
Repeated Pattern	maybe x7	maybe x7

Team Homework Separation of Floor Track



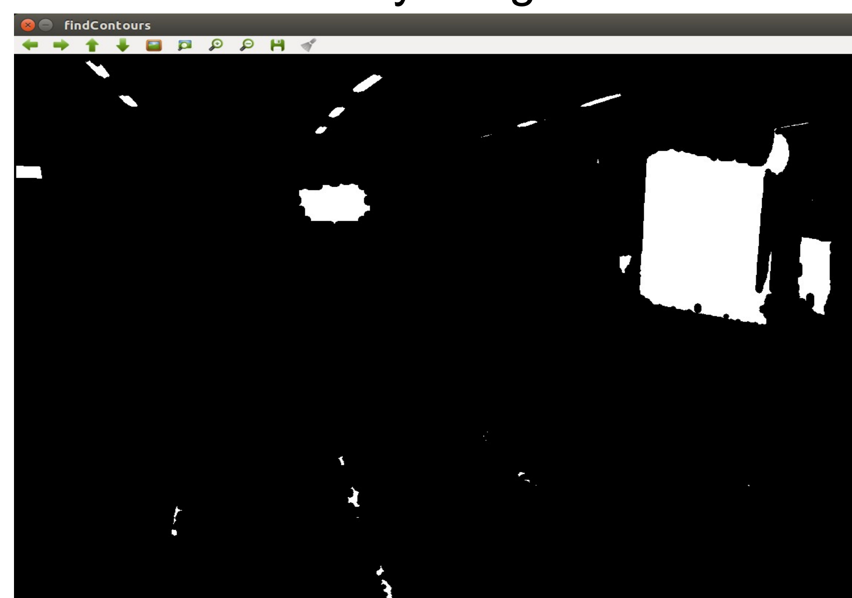
Original image



Gray-image

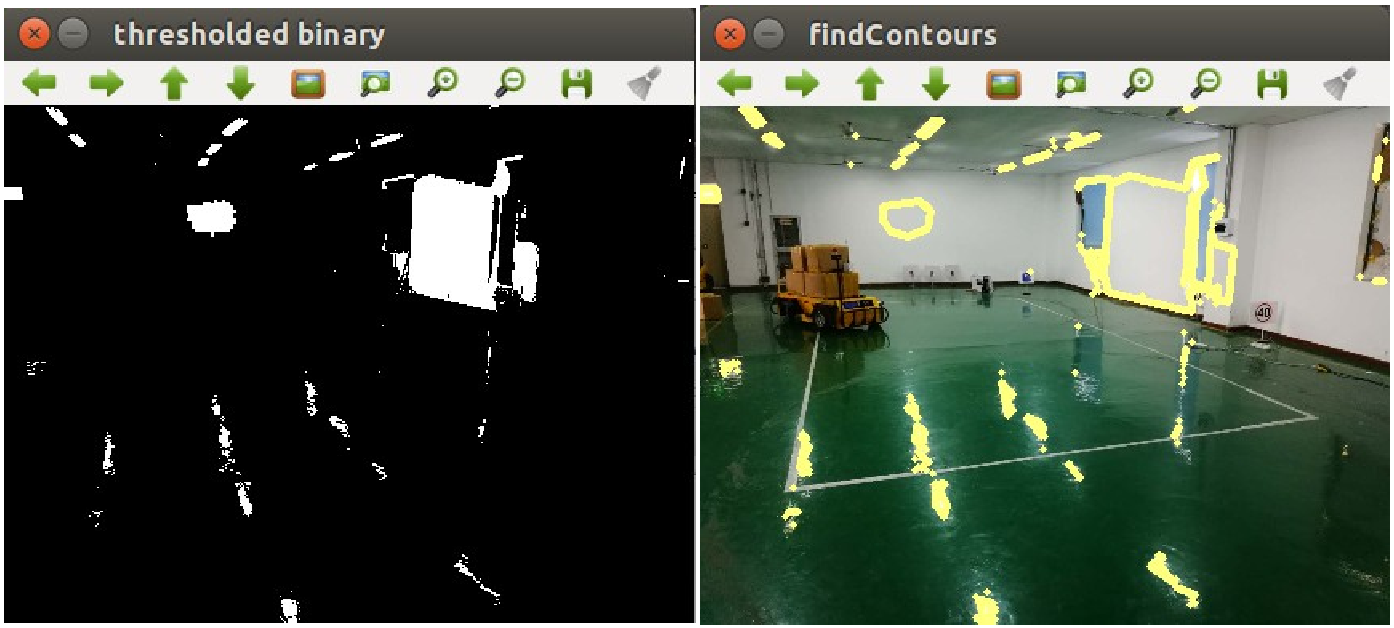
Difference = Original – high intensity thresholded image
>> image with removal of high intensity region

thresholdbinary



findcontour

Reflection Removal Based On Threshold



Reflection Removal Based On Adaptive Threshold

```
img_blr4 = cv2.GaussianBlur(img, (21,21), 36, 47)
```

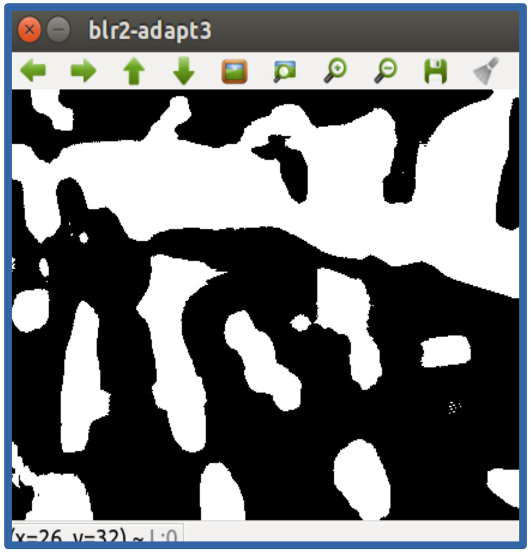
```
img_blr4_gray = cv2.cvtColor(img_blr4, cv2.COLOR_BGR2GRAY)
```

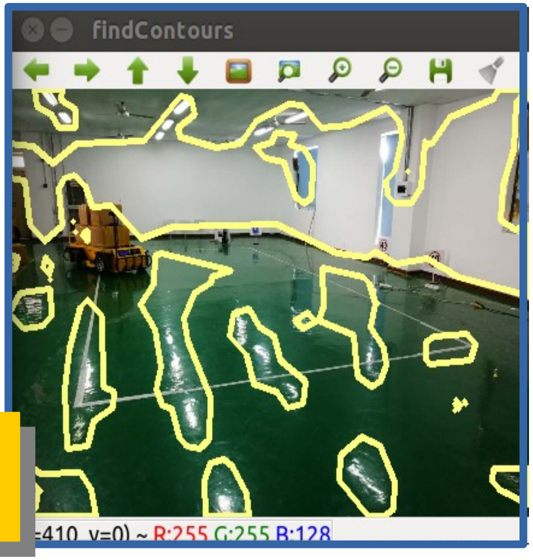
```
thresh3 = cv2.adaptiveThreshold(img_blr4_gray,255,\
          cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
          cv2.THRESH_BINARY,233,0)
```

```
cv2.imshow('blr4-adapt3',thresh3)
```

```
_,contours,hierarchy = cv2.findContours(thresh3, \
        cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = [cv2.approxPolyDP(cnt, 3, True) for cnt in contours]
```

```
cv2.drawContours(img, contours, -1, (128,255,255),3)
```



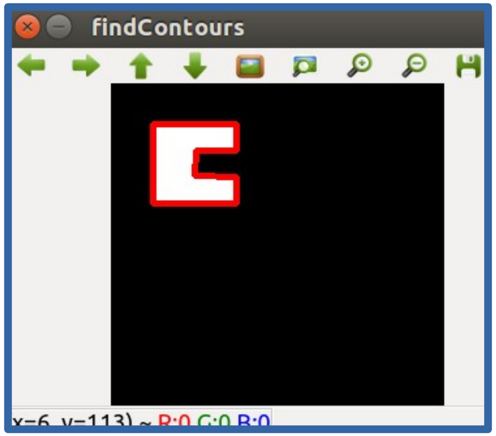


Is the track being removed as well?

Harry Li, Ph.D

Contour Attributes

```
_,contours,hierarchy = cv2.findContours(thresh, /
                cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = [cv2.approxPolyDP(cnt, 3, True) for cnt in contours]

cv2.drawContours(img, contours, -1, (0,0,255),3)
```



Inference Engine

1

Table 1. Attribute Table

	Ceiling Lights (class w1)	Window Lights (class w2)
Shape	Rectangles x1	Rectangles x1
	Ellipses x2	Ellipses x2
	Circles x3	Circles x3
Location	Anywhere x4 smaller part image x5	Anywhere x4 smaller part image x5
Color	white x6	white x6
Repeated Pattern	maybe x7	maybe x7

3

So decision function

$$\boxed{f(X) = x1\ x5\ x6 + x2\ x6 + x3\ x6 + x5\ x6 \quad \ldots (1)}$$

C/c++ implementation of the inference engine (switching function)

4

Table 2. Identification Table

2

	x1 rect	x2 elli	x3 cir	x4 loc	x5 sml	x6 wht	x7 rep	f(X)
x1 x5 x6	1	D	D	D	1	1	D	1
x2 x6	D	1	D	D	D	1	D	1
x3 x6	D	D	1	D	D	1	D	1
x5 x6	D	D	D	D	1	1	D	1

Define primary implicant, removal of any of its column will result in the mis-identification of f(X)

No: C/C++ Inference Engine

```
#include<stdio.h>
int And(int a, int b);
int Or(int a, int b);
int Not(int a);
void main()
{
 ///where main body of code will go
}
int And(int a, int b)
{
 int output;
 if(a==0 && b==0)
  output=0;
  if(a==1 && b==0)
  output=0;
 if(a==0 && b==1)
  output=0;
 if(a==1 && b==1)
  output=1;
 return (output);
}
```

Simplify it 1. as boolean; 2. logically as &&

```
int Or(int a, int b)
{
 int output;
 if(a==0 && b==0)
  output=0;
  if(a==1 && b==0)
  output=1;
 if(a==0 && b==1)
  output=1;
 if(a==1 && b==1)
  output=1;
 return (output);
}
int Not(int a)
{
int output;
 if(a==0 )
  output=1;
 if(a==1 )
  output=0;
  return (output);
}
```

In fact C/C++ support all the boolean logic operators, so build inference engine should be straight forward

Simplify it 1. as boolean;

```
int And(int a, int b)
{
    return a && b;
}
```

```
return Not(And(a, b));
```

Build NAND, NOR, XOR etc

C/C++ Bitwise Operators

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

```
// C Program to demonstrate the working of logical operators
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;
    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) equals to %d \n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) equals to %d \n", result);

    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) equals to %d \n", result);

    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) equals to %d \n", result);

    result = !(a != b);
    printf("!(a == b) equals to %d \n", result);

    result = !(a == b);
    printf("!(a == b) equals to %d \n", result);
    return 0;
}
```

C/C++ Inference Engine

```
//-----Inference Engine to find reflection spots---//
//-----April 7, 2018, by HL, version 0x0.1; --------//
 #include <stdio.h>
#include <stdbool.h>
#define dimension 100
bool    x[dimension], f_identification;
int     item;

int main()
{
    printf("Inference Engine to identify reflections \n");
    printf("x1 rectangle? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[1] = true;
    if (item == 0) x[1] = false;

    printf("x2 ellips? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[2] = true;
    if (item == 0) x[2] = false;

    printf("x3 circle? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[3] = true;
    if (item == 0) x[3] = false;

    printf("x4 location? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[4] = true;
    if (item == 0) x[4] = false;
```

```
printf("x5 small size? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[5] = true;
    if (item == 0) x[5] = false;

    printf("x6 white color? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[6] = true;
    if (item == 0) x[6] = false;

    printf("x7 repetative? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[7] = true;
    if (item == 0) x[7] = false;

    f_identification = (x[1] && x[5] && x[6])
                     || (x[2] && x[6])
                     || (x[3] && x[6])
                     || (x[5] && x[6]);

    if (f_identification){
    printf("The object is reflection\n");}
    else {
    printf("The object is not reflection\n");}

    return 0;
}
```

OpenCV Contours For Shapes

Table 3 (based on Table 2) openCV functions

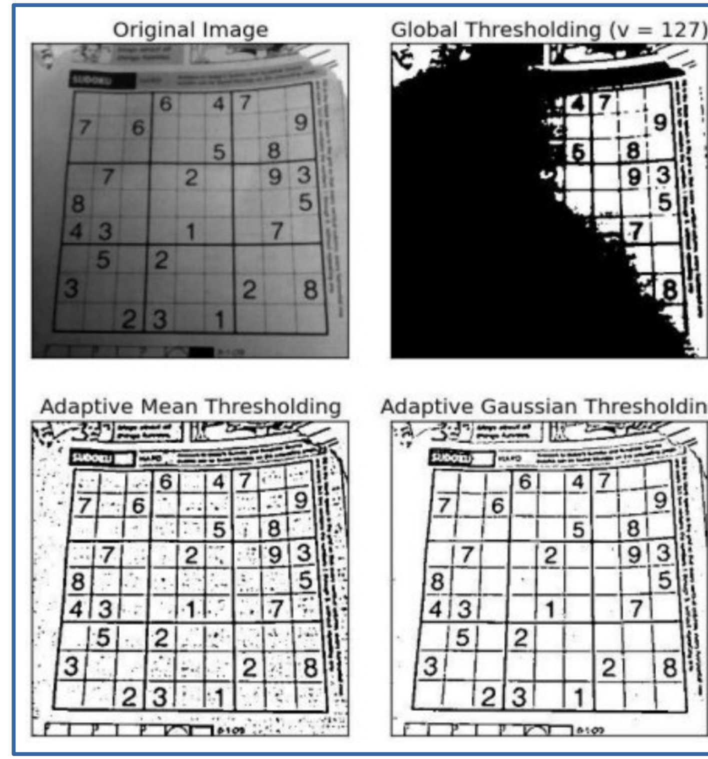
	x1 rect	x2 elli	x3 cir	x4 loc	x5 sml	x6 wht	x7 rep
x1 x5 x6	1	D	D	D	1	1	D
x2 x6	D	1	D	D	D	1	D
x3 x6	D	D	1	D	D	1	D
x5 x6	D	D	D	D	1	1	D

Rectangle detection (size, location and color, as well as total number);
ellips detection (size, location and color, as well as total number);
Circle detection (size, location and color, as well as total number);

Adaptive Threshold

https://docs.opencv.org/3.3.0/d7/d4d/tutorial_py_thresholding.html

```
thresh2 = cv2.adaptiveThreshold(img_gray,255,\
          cv2.ADAPTIVE_THRESH_MEAN_C,\
          cv2.THRESH_BINARY,33,0)
thresh3 = cv2.adaptiveThreshold(img_gray,255,\
          cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
          cv2.THRESH_BINARY,33,0)
```



cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) → dst

src – Source 8-bit single-channel image.
dst – Destination image of the same size and the same type as src .
maxValue – Non-zero value assigned to the pixels for which the condition is satisfied.
adaptiveMethod – ADAPTIVE_THRESH_MEAN_C or ADAPTIVE_THRESH_GAUSSIAN_C .
thresholdType – THRESH_BINARY or THRESH_BINARY_INV .
blockSize – Size of a pixel neighborhood 3, 5, 7, and so on.
C – Constant subtracted from the mean or weighted mean, positive may be zero or negative.

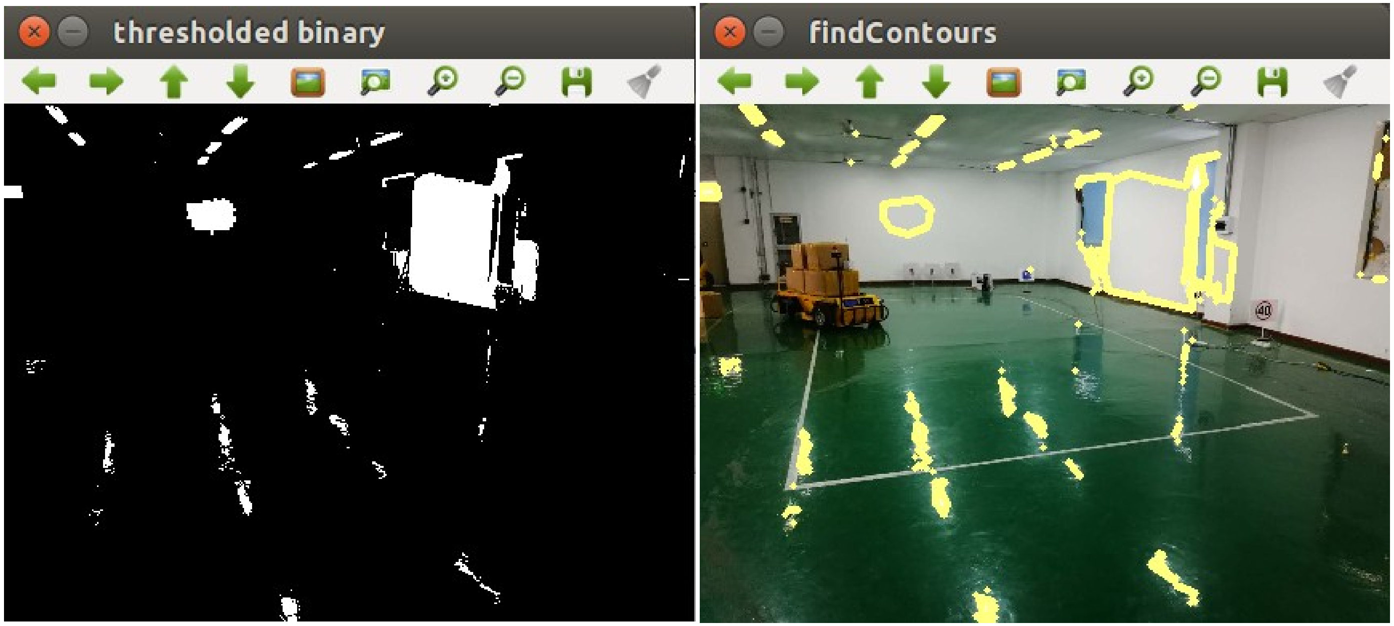
cv:: void adaptiveThreshold( )

```
void adaptiveThreshold(
        InputArray src,
        OutputArray dst,
        double maxValue,
        int adaptiveMethod,
        int thresholdType,
        int blockSize, double
        C)
```

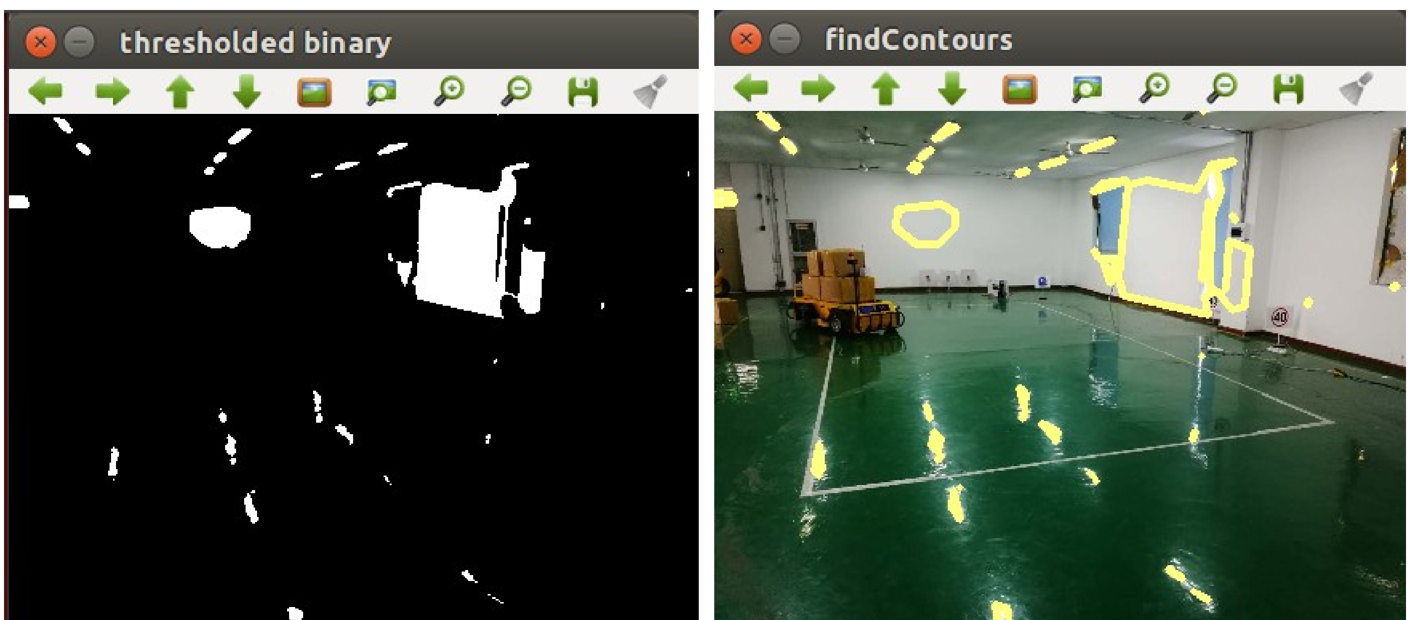
Parameters:

src – Source 8-bit single-channel image.
dst – Destination image of the same size and the same type as src .
maxValue – value assigned to pixels for condition satisfied.
adaptiveMethod –
ADAPTIVE_THRESH_MEAN_C or
ADAPTIVE_THRESH_GAUSSIAN_C .
thresholdType – THRESH_BINARY or THRESH_BINARY_INV .
blockSize – kernel sise, 3, 5, 7, and so on.
C – Constant subtracted from the mean or weighted mean. Normally, positive, can be zero or negative as well.

With Or W/O Gaussian Blur Binrization+Contour



Binrization+Contour



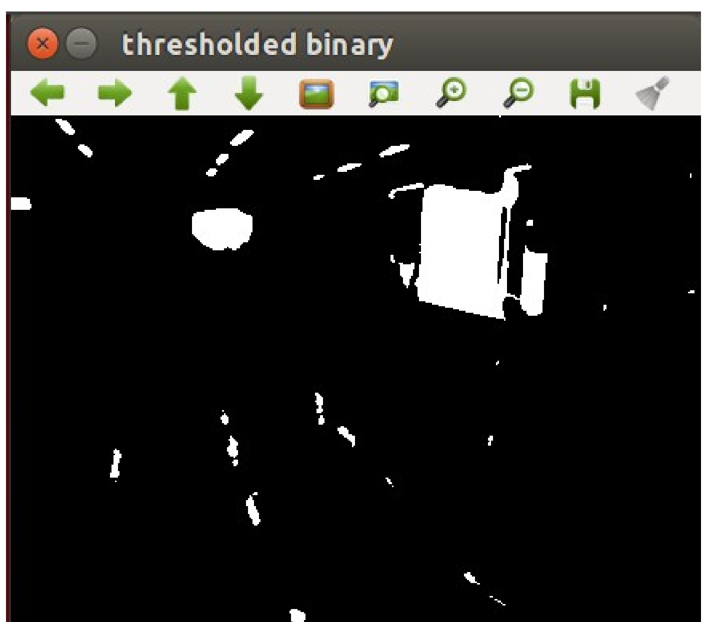
GaussianBlur+
Binrization+Con
tour

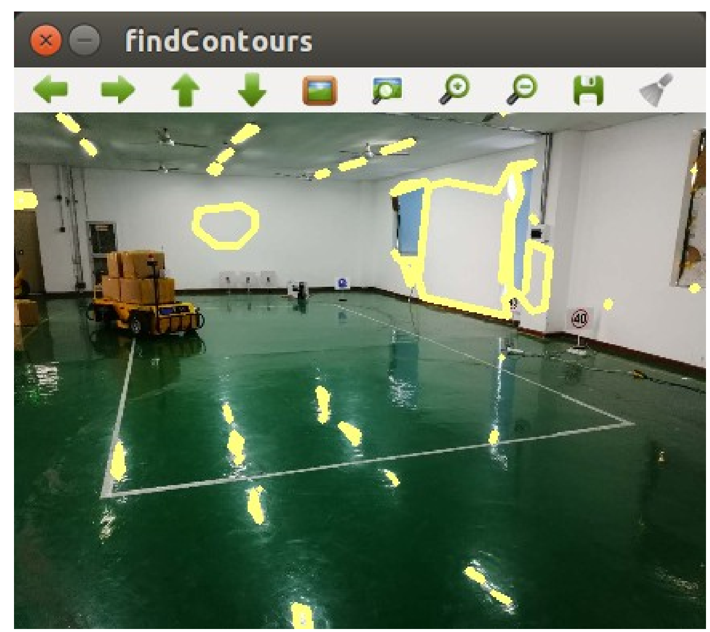
```
img_blr0 = cv2.GaussianBlur(img, (3, 3), 2, 3)
img_gray = cv2.cvtColor(img_blr0, cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(img_gray,200,255,0)
```

Surgical Removal

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_core/py_basic_ops/py_basic_ops.html

```
>>> px = img[100,100]
>>> print px
[157 166 200]

# accessing only blue pixel
>>> blue = img[100,100,0]
>>> print blue
157
```





cv::inRange( ) Thresholding Colour Images

https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#inrange

```
C++: void
inRange(
    InputArray src,
    InputArray lowerb,
    InputArray upperb,
    OutputArray dst)
```

src – first input array.
lowerb – inclusive lower boundary array or scalar.
upperb – inclusive upper boundary array or scalar.
dst – output array, CV_8U type.

$$\mathtt{dst}(\mathrm{I}) = \mathtt{lowerb}(\mathrm{I})_0 \leq \mathtt{src}(\mathrm{I})_0 \leq \mathtt{upperb}(\mathrm{I})_0 \wedge \mathtt{lowerb}(\mathrm{I})_1 \leq \mathtt{src}(\mathrm{I})_1 \leq \mathtt{upperb}(\mathrm{I})_1$$

Finding Lane Lines with Colour Thresholds
https://medium.com/@tjosh.owoyemi/finding-lane-lines-with-colour-thresholds-beb542e0d839
Joshua OwoyemiSelf-driving Car Engineer, PhD Candidate in Computer Vision and Robot Manipulation, Sharing technology insights.

My ColorPicker.cpp

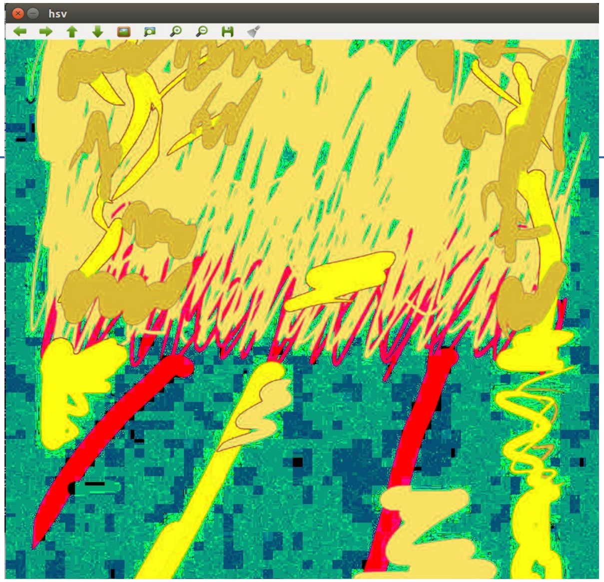
```
//-------------------------------------------------------------*
// program: colorPicker.cpp;  Coded by: HL on line *
// soure.                                                      *
// purpose: hsv color picking                                  *
// last update: April 28, 2018.                                *
//-------------------------------------------------------------*
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace cv;
using namespace std;

Mat im_hsv;
void pick_color(int e, int x, int y, int s, void *)
{
    if (e==1) // left mouse down
    {
        Vec3b p = im_hsv.at<Vec3b>(y, x); //pixel value
        cerr << int(p[0]) << " " << int(p[1]) << " " << int(p[2]) << endl;
    }
}
```

```
int main( int argc, char** argv )
{
    namedWindow("hsv");
    setMouseCallback("hsv", pick_color);

    if (argc<2) return -1;
    Mat im_bgr = imread(argv[1]);
    if (im_bgr.empty()) return -2;

    cvtColor(im_bgr, im_hsv, COLOR_BGR2HSV);
    imshow("hsv", im_hsv);
    waitKey();

    return 0;
}
```

```
ubuntu@ubuntu-ThinkPad-Yoga-14: ~/Do
ts/source/cpp$ ./main art-road1.jpg
init done
opengl support available
24 247 255
100 226 249
17 255 255
```

~/Documents/SJSU/CMPE297/CMPE297VideoAnalytics/lec/lec5-binary-image/lec5-2-Contours-Moments/source/cpp$ ./main art-road1.jpg

Right click to pick pixel color



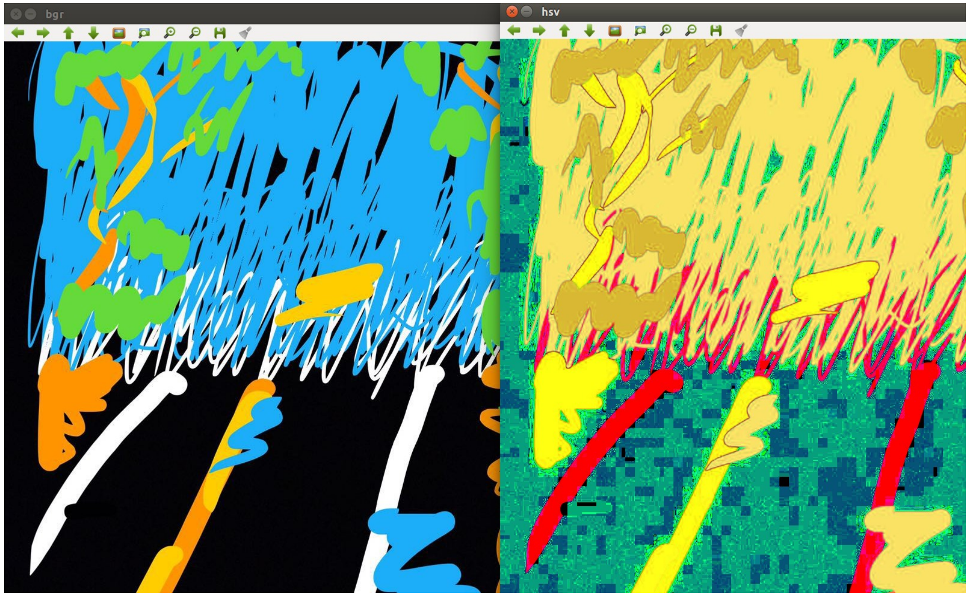
Display image in hsv space

My ColorPicker.py

/Documents/SJSU/CMPE297/CMPE297VideoAnalytics/lec/lec5-binary-image/lec5-2-Contours-Moments/source/py$

opencv hsv color picker

How to define the “lower” and “upper” range of a color?
http://answers.opencv.org/question/134248/how-to-define-the-lower-and-upper-range-of-a-color/




```
#------------------------------------------------------*
# program: colorPicker.py;                             *
# reference code: see Harry Li's PPT for               *
#          original source;                            *
# date: April 28, 2018;  status: tested;               *
#------------------------------------------------------*
import cv2
import numpy as np

image_hsv = None   # global
pixel = (20,60,80) # some default

# mouse callback function
def pick_color(event,x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDOWN:
        pixel = image_hsv[y,x]
```

C++: void log(InputArray src, OutputArray dst)

https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#inrange

https://www.learnopencv.com/high-dynamic-range-hdr-imaging-using-opencv-cpp-python/

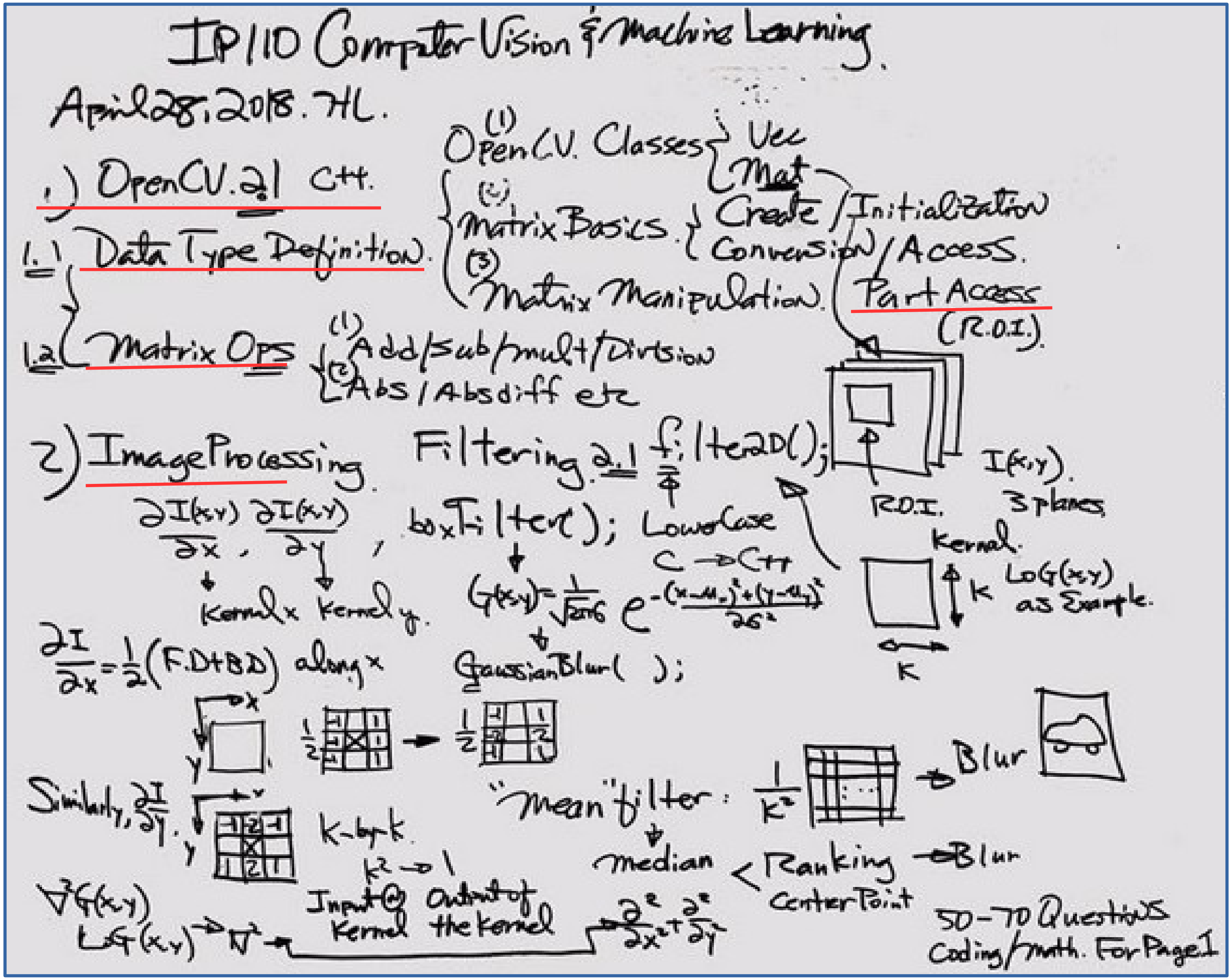
High dynamic imaging

The function log calculates the natural logarithm of the absolute value of every element of the input array

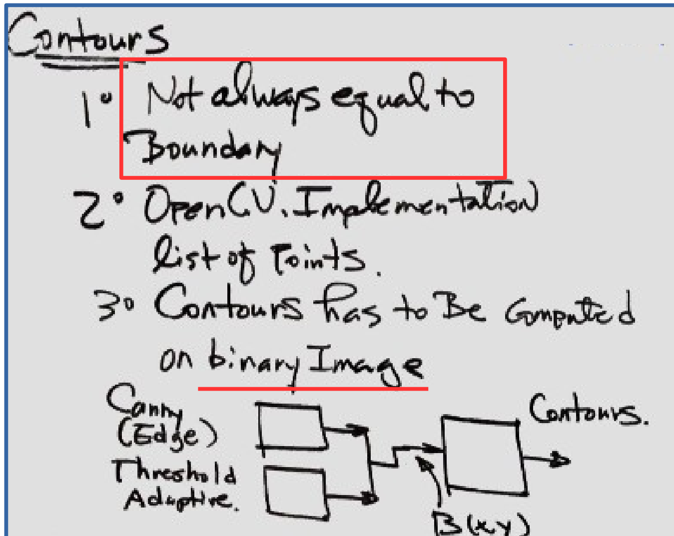
$$\texttt{dst}(I) = \begin{cases} \log|\texttt{src}(I)| & \text{if } \texttt{src}(I) \neq 0 \\ \texttt{C} & \text{otherwise} \end{cases}$$

where C is a large negative number (about -700 in the current implementation). The maximum relative error is about 7e-6 for single-precision input and less than 1e-10 for double-precision input. Special values (NaN, Inf) are not handled.

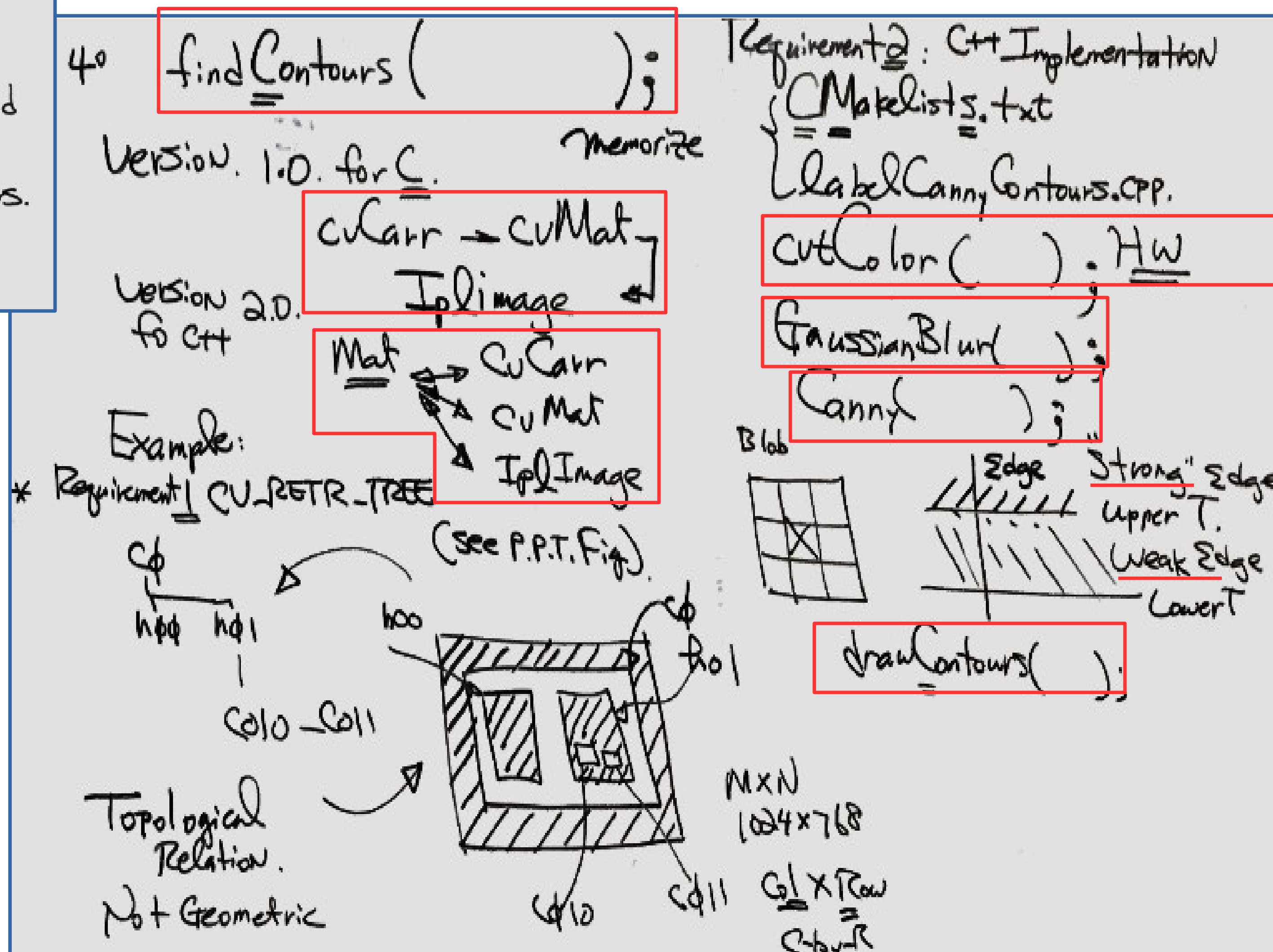
OpenCV Version 2.1 C++ Overview



Contours Trees



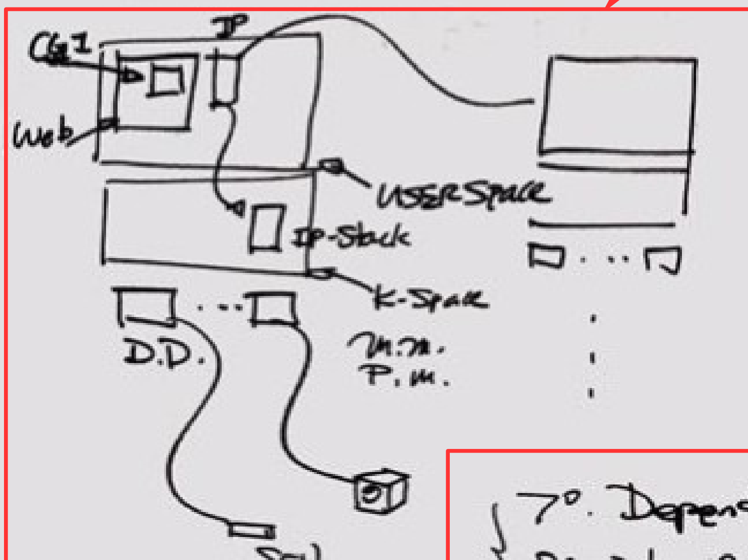
Topological relationship of each contour and mapping from a given contour image to tree structure



findContour( ); only works on binary image, one of the 3 images, Canny, Threshold and adaptiveThreshold

Full Stack Embedded Software Developer

Full Stack Embedded Software (Doc) Developer



1. Device deriver layer; 2. Kernel Space layer; 3. User Space (app) layer; 4. Networking layer; 5. Database and enterprise layer interface

11 Common hiring reference checking questions to prepare while working on your project

Reference checking:

1° How Long Have you been working/Knowing this person?

2° What is the working Relationship. Tech Supervisor/Co-worker/...

3° Problem Solver?

4° Take any initiatives?

5° Team player?

6° Good Communicator?

7° Dependable?

8° Handle Stress?

9° Handle Criticism?

10° Detail Oriented?

11° Given a chance, to hire this person again?

12° Is there Anything else do you want to tell us about?