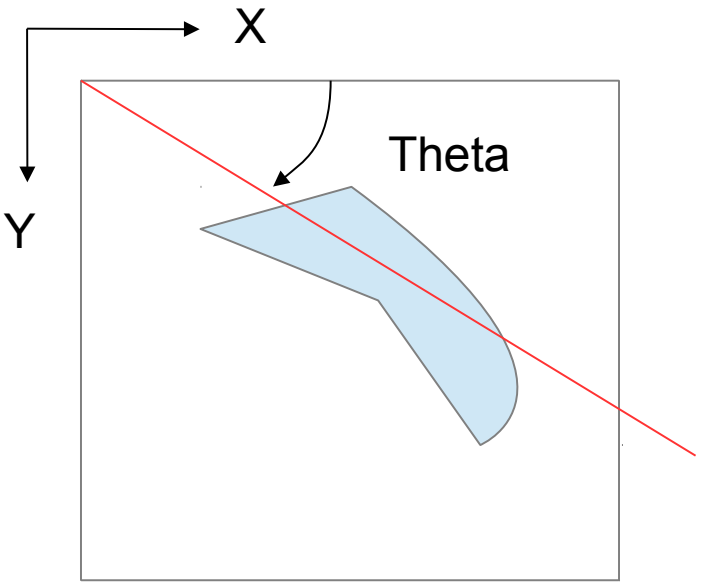
Orientation Computation

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

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

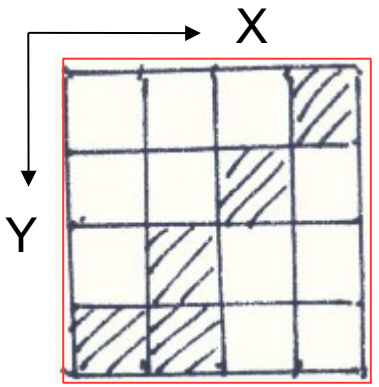
$$b = \iint_{\Omega} 2(x-\bar{x})(y-\bar{y}) B(x,y)\, dx dy \quad \cdots (3)$$

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



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

Example: See my handout



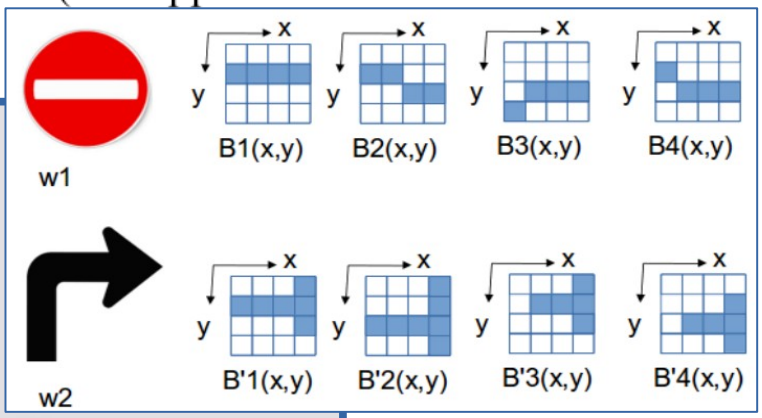
a = 7
b = -8
c = 6
Theta = -41.4375

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% 2% 3% 2% 2/0 (9/4)/0 (8/5)/(7/5) 3/0

u_x/m_{10} 2.5% 2.5% 2.5% 2.5% (18/6)/0 (22/7)/(-9/7) (17/5)/0 (17/5) 0

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

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

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

$+(4-17/5)=0$

(2) Part II. K-mean Cluster Algorithm.

Use (u_x, u_y) to form feature vectors.

Then, apply OpenCV. k-mean

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

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

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

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

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

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

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

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

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

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)

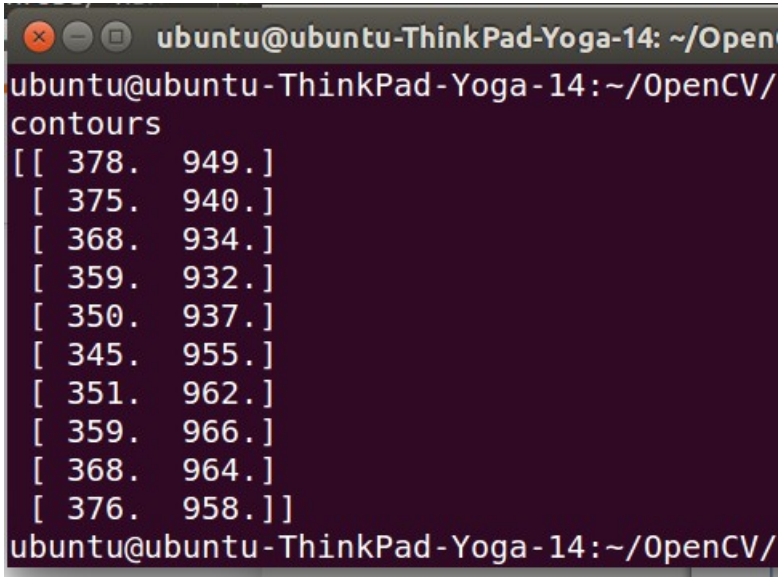
The arguments: the 1st is source image, 2nd is contour retrieval mode, 3rd is contour approximation method. And it outputs the contours and hierarchy. contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.

```
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 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

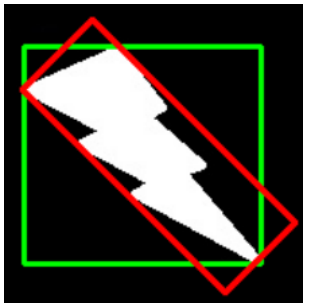
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

1. Moments

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread('star.jpg',0)
5 ret,thresh = cv2.threshold(img,127,255,0)
6 contours,hierarchy = cv2.findContours(thresh, 1, 2)
7
8 cnt = contours[0]
9 M = cv2.moments(cnt)
10 print M
```

2. Contour Area

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



3. Contour Perimeter

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

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)
```

6. Checking Convexity

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

Compute Contours Features

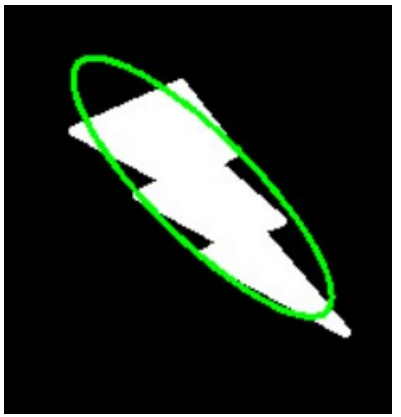
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

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)
```



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

10. Fitting a Line

```
1 rows,cols = img.shape[:2]
2 [vx,vy,x,y] = cv2.fitLine(cnt, cv2.DIST_L2,0,0.01,0.01)
3 lefty = int((-x*vy/vx) + y)
4 righty = int(((cols-x)*vy/vx)+y)
5 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

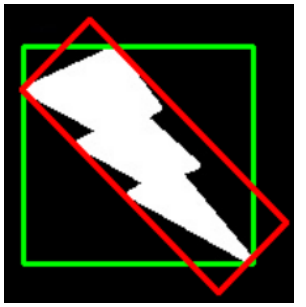
```
hull = cv2.convexHull(cnt)
```

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)
```

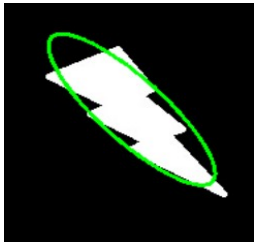


8. Minimum Enclosing Circle

```
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

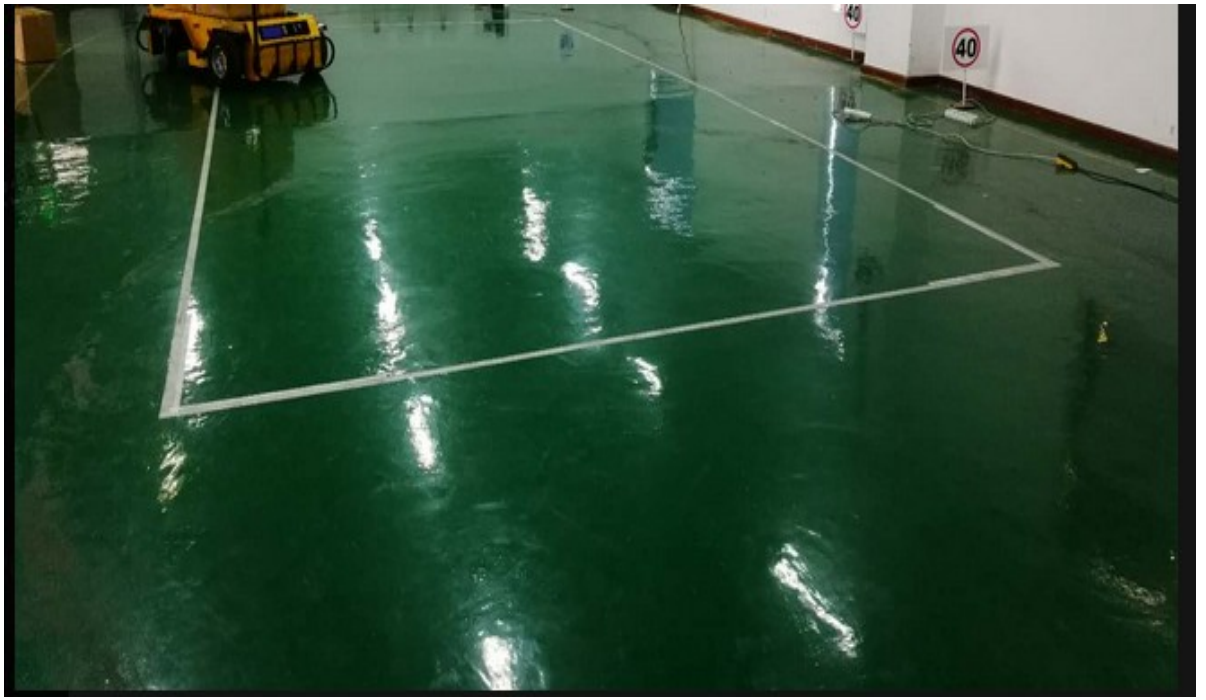
18. Mean Color or Mean Intensity

```
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])
```

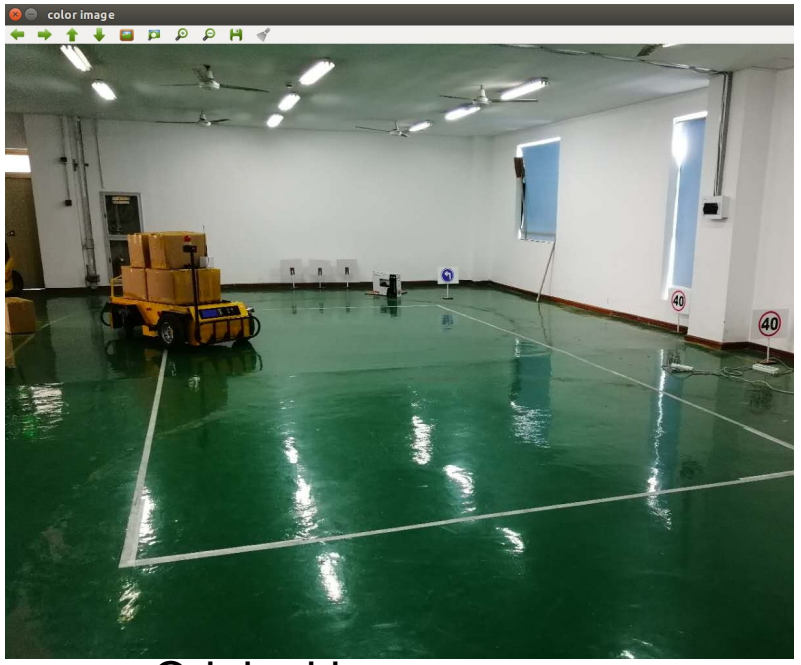
Example Separation of Floor Track



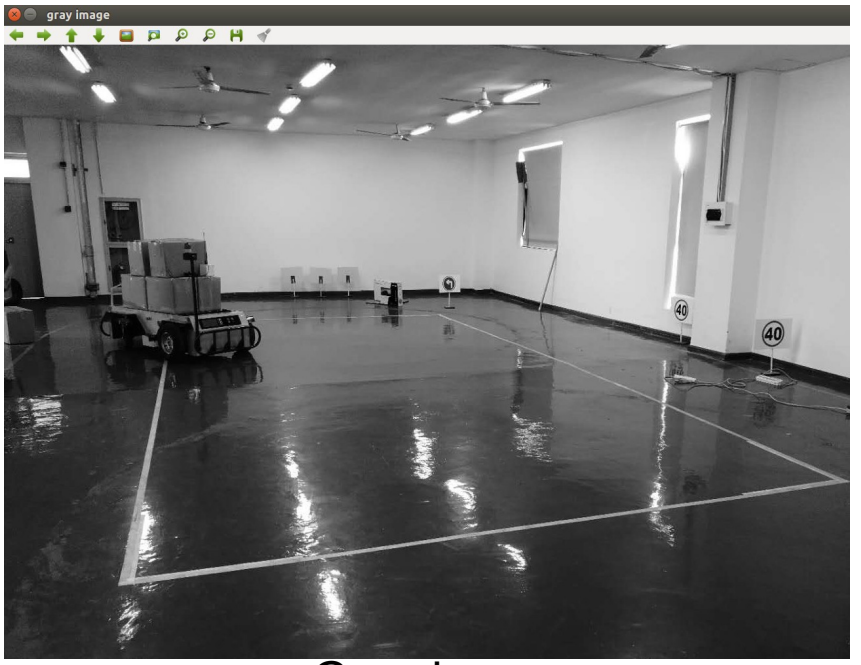
From Shapes && Colors Find ROI And Remove Reflections

	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



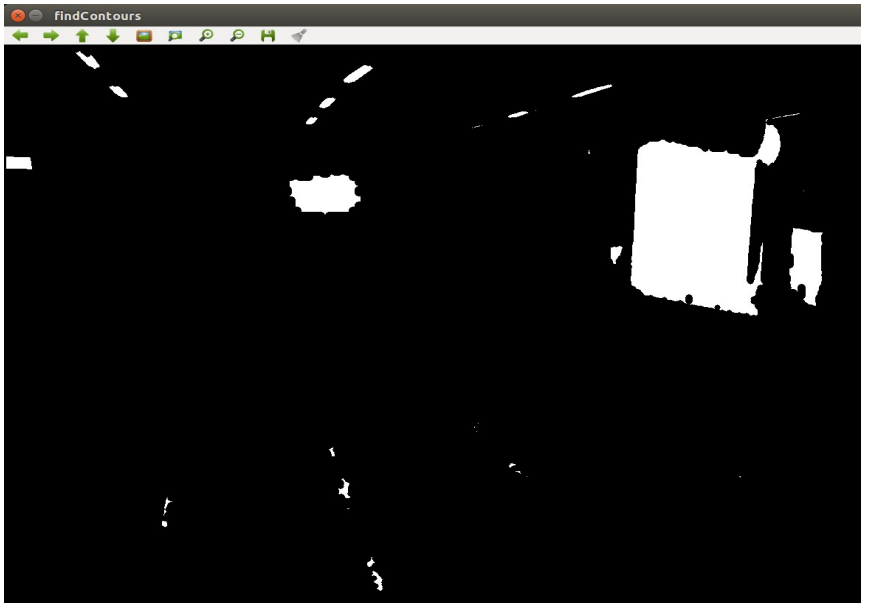
Original image



Gray-image

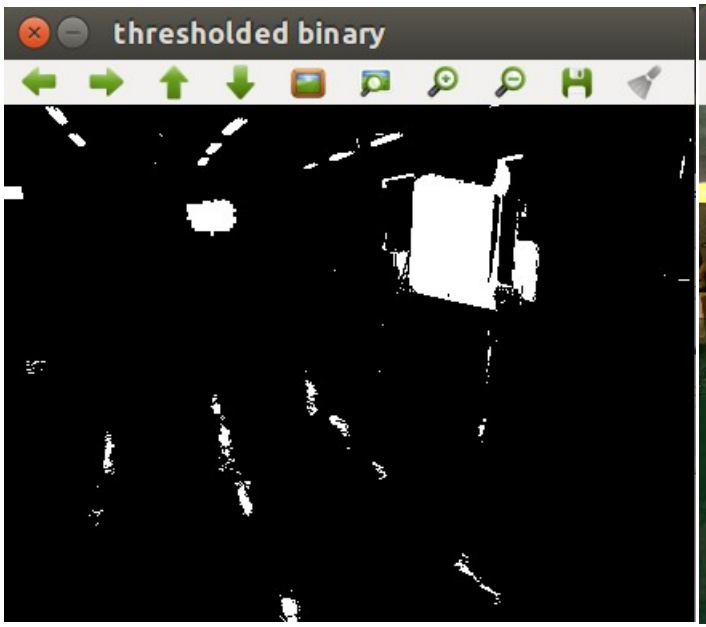
Difference = Original – high intensity thresholded image
>> image with removal of high intensity region

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)
```



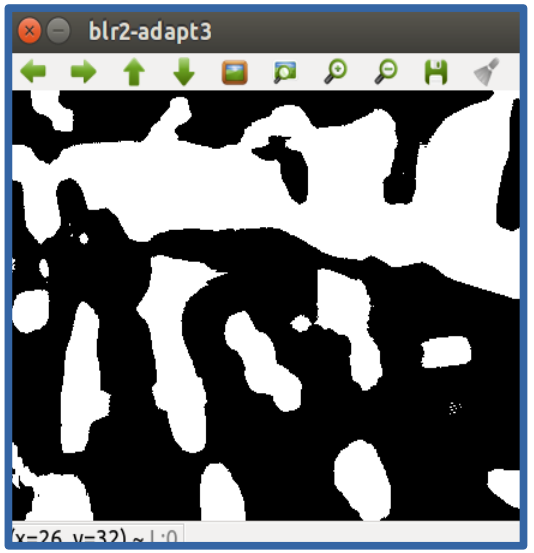
```
cv2.imshow('blr4-adapt3',thresh3)
```

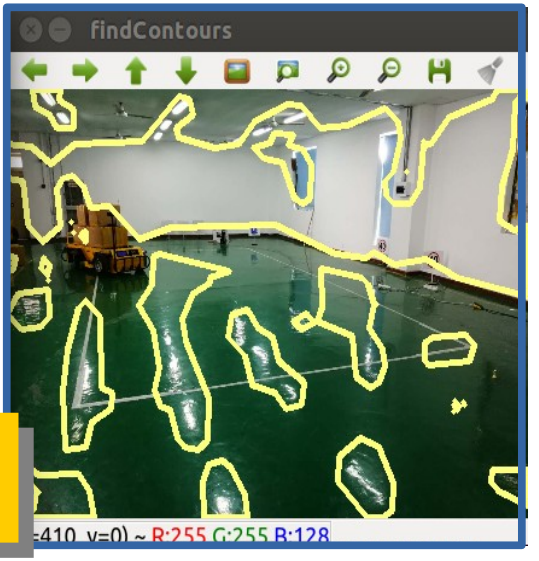


```
_,contours,hierarchy = cv2.findContours(thresh3, \
        cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = [cv2.approxPolyDP(cnt, 3, True) for cnt in contours]
```



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

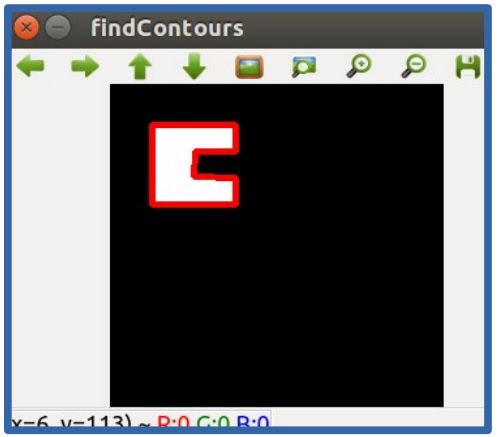




Is the track being removed as well?

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

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

4

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

2

Table 2. Identification Table

	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

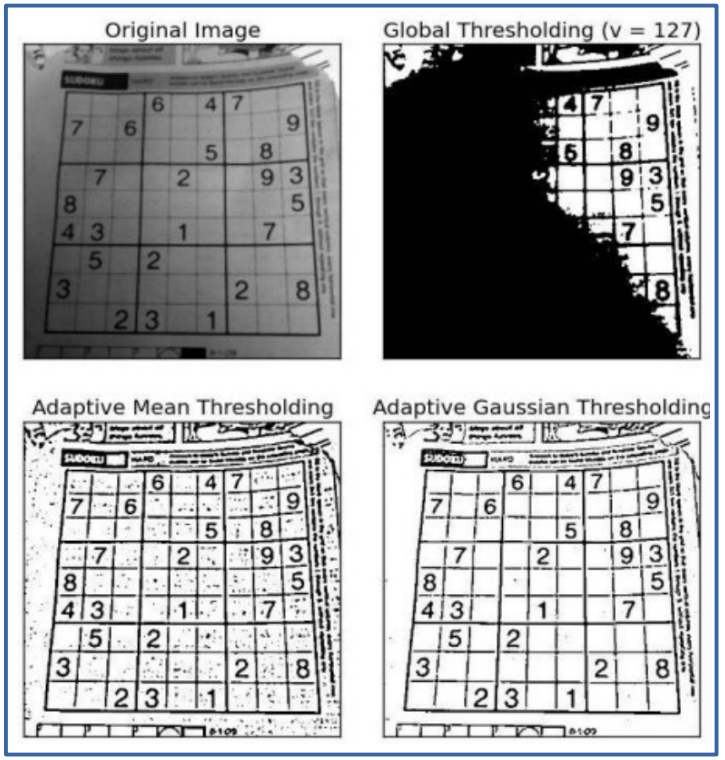
	x1 rect	x2 elli	x3 cir	x4 loc	x5 sml	x6 wht	x7 rep
x1 x5 x6	1	D	D	D	1	1	D
x2 x6	D	1	D	D	D	1	D
x3 x6	D	D	1	D	D	1	D
x5 x6	D	D	D	D	1	1	D

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

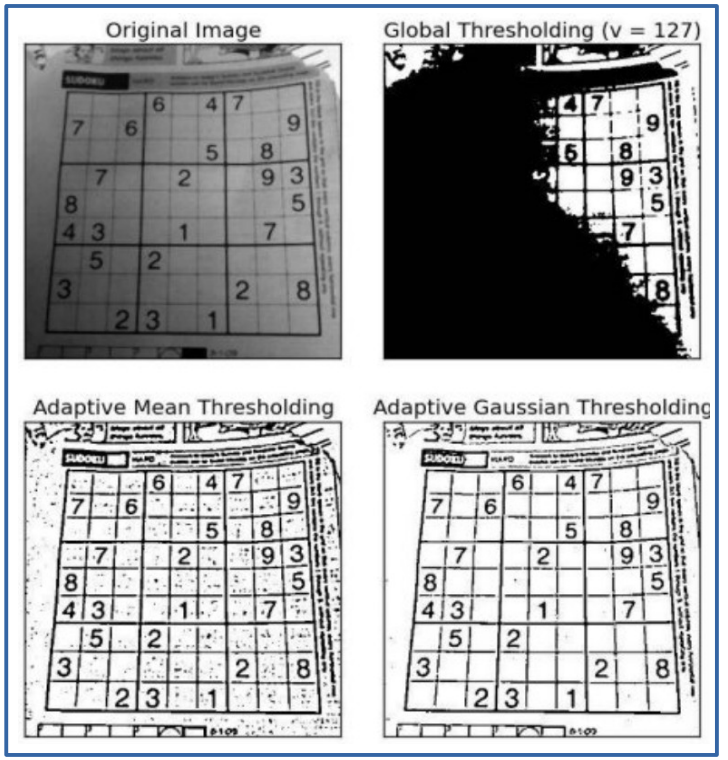
src – Source 8-bit single-channel image.
dst – Destination image of the same size and the same type as src .
maxValue – Non-zero value assigned to the pixels for which the condition is satisfied.
adaptiveMethod – ADAPTIVE_THRESH_MEAN_C or ADAPTIVE_THRESH_GAUSSIAN_C .
thresholdType – THRESH_BINARY or 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.

Adaptive Threshold

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

How Otsu's Binarization Works?

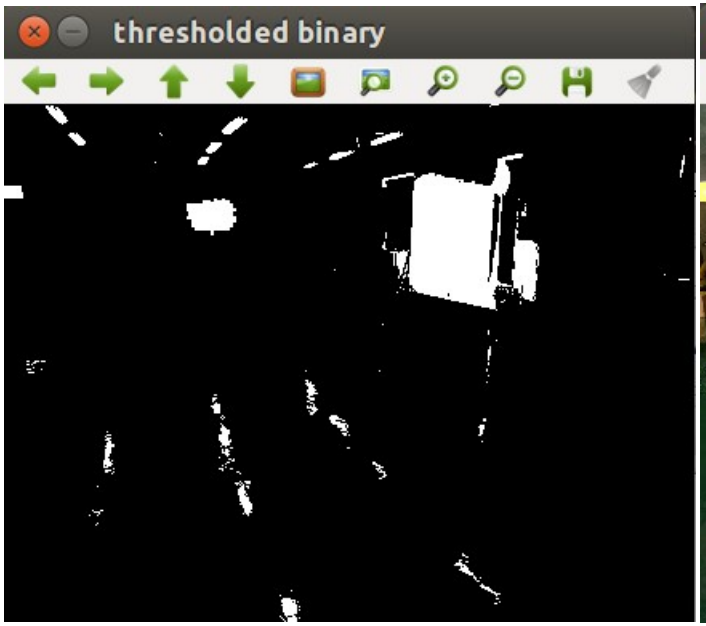
```
thresh2 = cv2.adaptiveThreshold(img_gray,255,\
          cv2.ADAPTIVE_THRESH_MEAN_C,\
          cv2.THRESH_BINARY,33,0)
thresh3 = cv2.adaptiveThreshold(img_gray,255,\
          cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
          cv2.THRESH_BINARY,33,0)
```



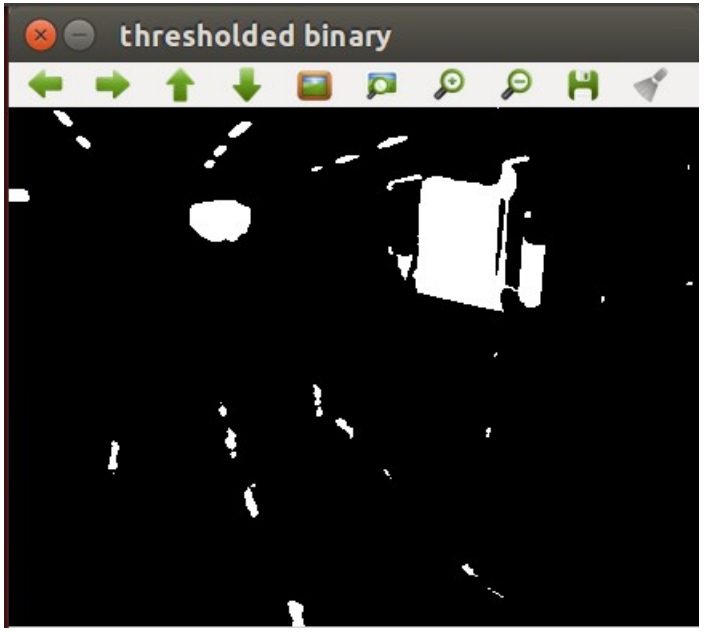
High Light Removal

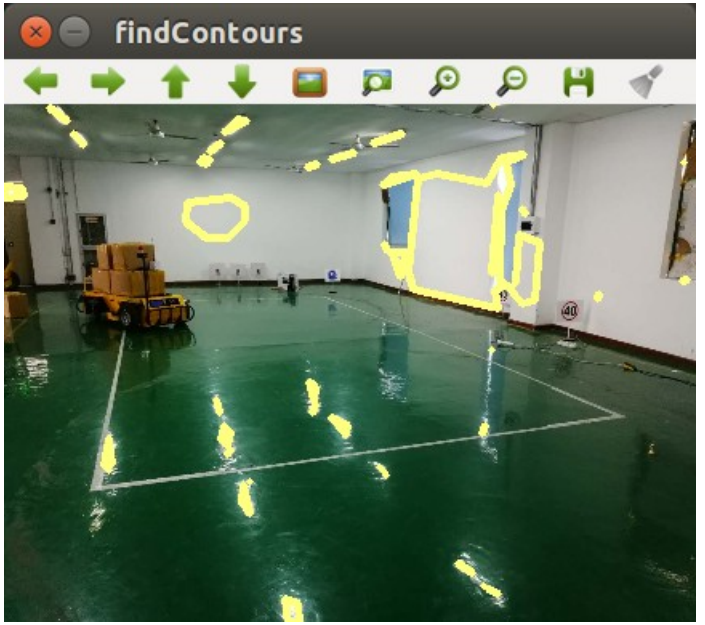
```
img = cv2.imread('1track.jpg')  #'0' to read color as gray scale

img_blr1 = cv2.GaussianBlur(img, (5, 5), 3, 3) #5 by 5 kernel, sigma 3 for x and y
img_blr2 = cv2.GaussianBlur(img, (7, 7), 3, 7)
img_blr3 = cv2.GaussianBlur(img, (9, 9), 3, 11)
img_blr4 = cv2.GaussianBlur(img, (21,21), 6, 17)

cv2.imshow('gaussian 1',img_blr1)
cv2.imshow('gaussian 2',img_blr2)
cv2.imshow('gaussian 3',img_blr3)
cv2.imshow('gaussian 4',img_blr4)

#img_diff
img_diff1 = img - img_blr1
img_diff2 = img – img_blr2
img_diff3 = img - img_blr3
img_diff4 = img – img_blr4

cv2.imshow('diff1', img_diff1)
cv2.imshow('diff2', img_diff2)
cv2.imshow('diff3', img_diff3)
cv2.imshow('diff3', img_diff4)
```

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
```