Intro to Moments for Objects Recognition

Objectives: Develop a technical to detect different shaped, different color, different size objects

Image Acquisition

Preprocessing (1): 2D Gaussian Convolution

Features Extraction 1

....

Features Extraction i

Moments

Feature Vector V(v1,...,vk)

Feature Vector Learning

Decision Making

Pattern Recognition For Binary Images

The tool box for pattern recognition for binary images

1. Size
2. Moments

   $\overline{x}$

   $\overline{y}$

   $\overline{x^k}$

   $\overline{y^k}$ etc.
3. Perimeter
4. Orientation
5. Compositions of the above

   Perimeter and moments: vector
6. Invariant operators

   size invariant

   orientation invariant

   illumination invariant

Note: Starting from binary images, extended to color images

Biologically inspired techniques

Rule 1. Proximity

Rule 2. Similarity

Rule 3. Closure

Rule 4. Good continuation

Rule 5. Symmetry

Rule 6. Simplicity

Note: 'Proximity' usage for clean up binary image and remove noise, as well as growing boundary points per 'good continuation' rule to form a better edge map.

Note: Similarity defines a interesting question, how to describe one object is similar, or somewhat similar to others, neural network and fuzzy logic may help.

Ground rule: signature of a image, tools including 3 invariant characteristics

Harry Li, PhD. SJSU

Binary Image Processing

Extended to color images

Size

- 1 Perimeter — $\int_I f(x,y)dl$ …(1)
- Area — $\iint_I f(x,y)dxdy$ …(2)
  - Noise Removal — 2 Flood fill/Threshold
  - Size classification — Flood fill/Threshold

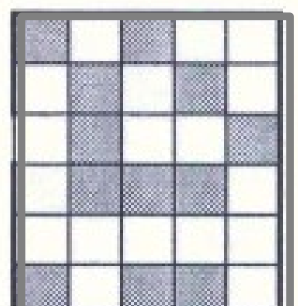
Shape

- 3 Enlongation axis
- Center
  - 4 Moment

$$\overline{x^k} = \frac{\iint_I x^k f(x,y)dxdy}{\iint_I f(x,y)dxdy} \quad (3)$$

$$\overline{y^k} = \frac{\iint_I y^k f(x,y)dxdy}{\iint_I f(x,y)dxdy} \quad (4)$$

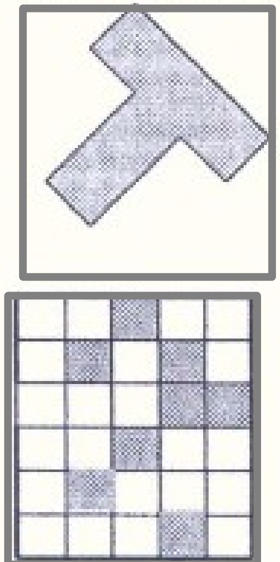
Example:



5 Skeletonization of 2D Binary Images

Example On Simple Pattern Recognition

Given two binary images, derived from two objects, T and O, design a technique to identify them



Example: Computation of
(1) Area (size);
(2) X-bar;
(3) Y-bar;
(4) Orientation, theta angle
(5) Perimeter of an object

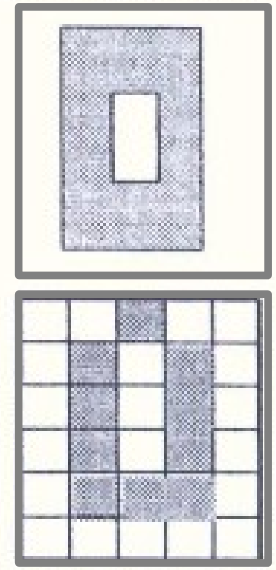


Fig1(a),(b)

Fig2(a),(b)

Good continuation or noise? What to do with this noise?

Feature Vector		Size	X-bar	Y-bar	Orientation	Perimeter	
V_1(v1,..., v5)	T	v11	v12	v13	v14	v15	From Fig1(b)
V_2(v1,..., v5)	L	v21	v22	v23	v24	v25	From Fig2(b)

Intro Feature Characterization

Example: Fill out this table based on the characteristics of each feature

	Perimeter v1	Area v2	x_bar v3	y_bar v4	Theta v5	Moments v6-vi	Hu-Moments v(i+1)-vk
Illumination invariant							
Scale invariant			Y	Y	Y		Y
Orientation Invariant							Y

Perimeter:

$$P = \int_{\Omega} f(x,y)\,dl$$

Or,

$$P = \sum_{k_1=1}^{N} \sum_{k_2=1}^{M} B'(x,y)$$

Where B'(x,y) from object whose neighboring pixels belong to background

x_bar:

$$\bar{x} \triangleq \frac{\iint_{\Omega} x\,B(x,y)\,dx\,dy}{\iint_{\Omega} B(x,y)\,dx\,dy}$$

y_bar can be defined similarly

Moments:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y)\,dx\,dy$$

Central moments:

And

$$\mu_{pq} \triangleq \frac{\iint_{\Omega} (x-\bar{x})^p (y-\bar{y})^q B(x,y)\,dx\,dy}{A}$$

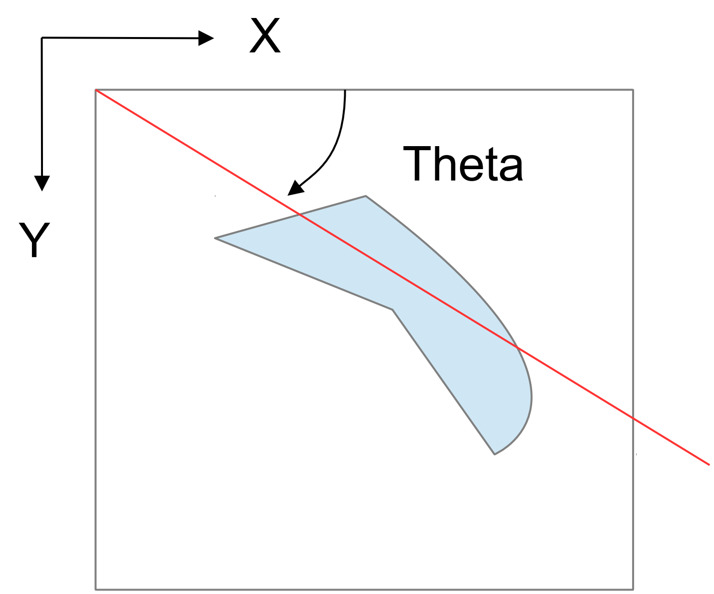
Orientation Computation

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

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

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

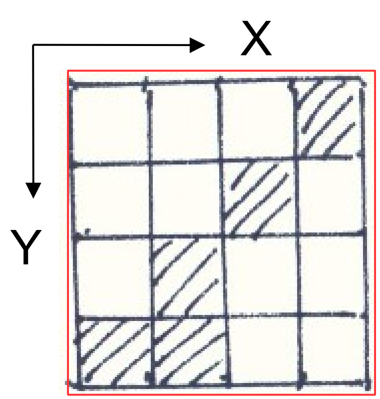
$$c = \iint_{\Omega} (y-\bar{y})^2 B(x,y)\, dx\, dy \quad \cdots (4)$$



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)

Raw Moments

The "raw moment" of order (p + q) for image f(x,y) is defined as:

$$M_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)\, dx\, dy \qquad (1)$$

For the discrete function, we have:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y) \qquad (2)$$

Note: image I(x,y) can be binary image or gray scale images. But we start the discussion from the binary images first.

We can treat image intensity as its probability density function

$$\sum_x \sum_y I(x,y) \qquad (3)$$

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

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)
```

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)
```

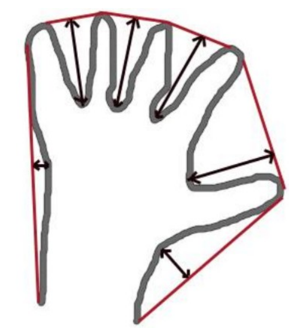
4. Contour Approximation

```
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

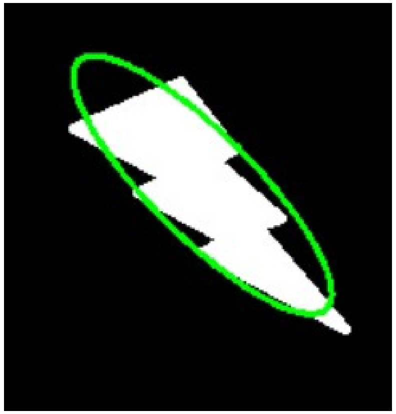
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

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)
```

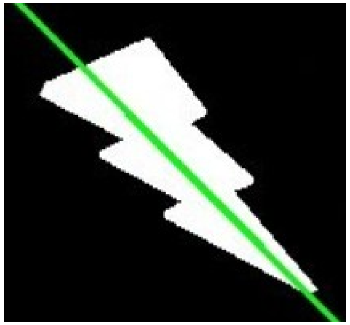
9. Fitting an Ellipse

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



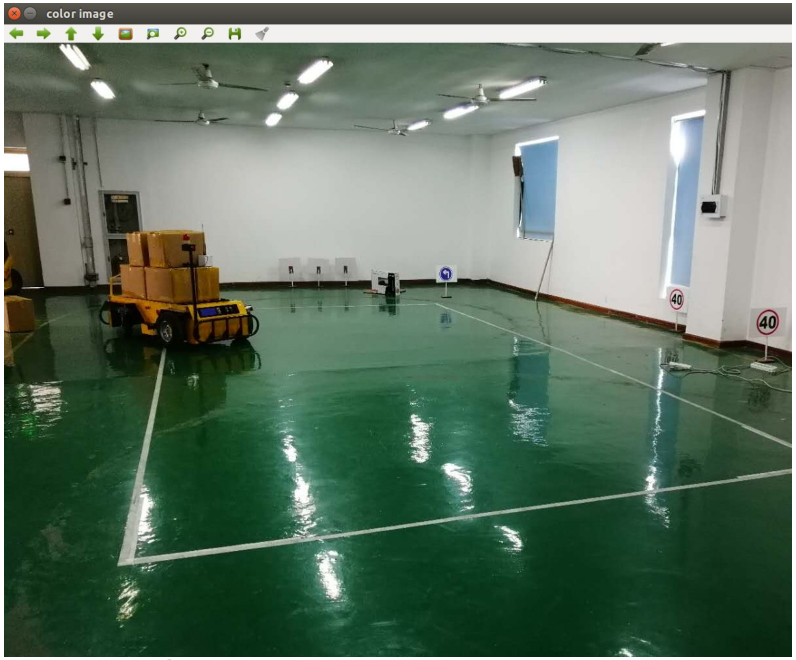
10. Fitting a Line

```
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)
```

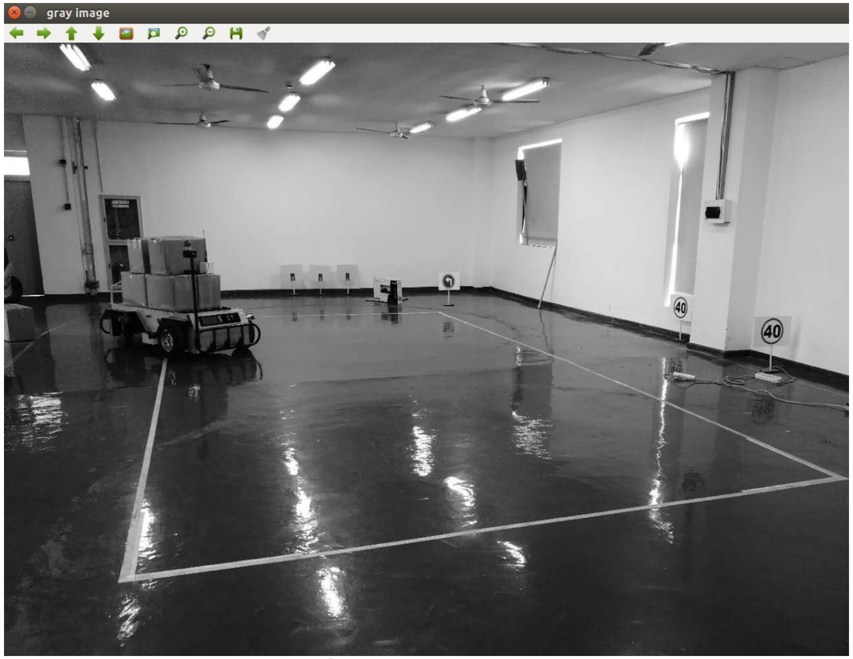


Separation of Floor Track Example With Practical Challenge

Team Homework Separation of Floor Track

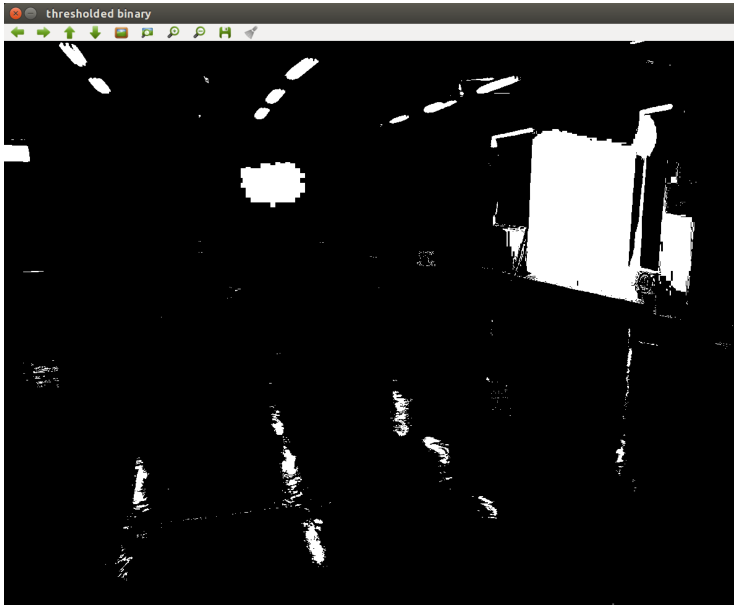


Original image

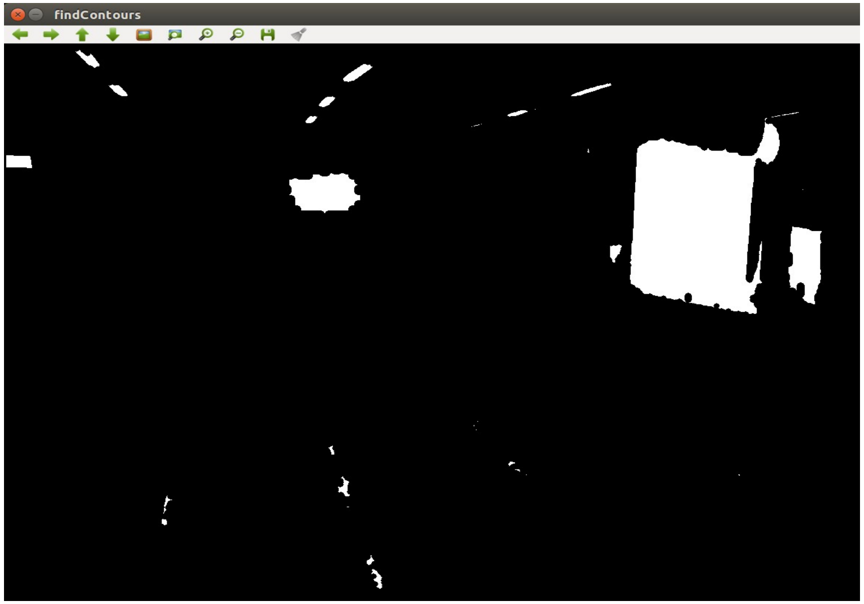


Gray-image

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



thresholdbinary

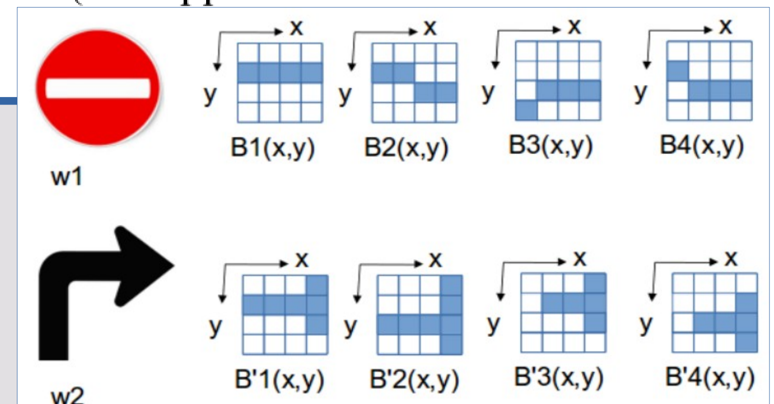


findcontour

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	(9/4)/0	(8/5)/(2/5)	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

Harry Li, Ph.D