CS4733 Class Notes, Computer Vision

Sources for online computer vision tutorials and demos - http://www.dai.ed.ac.uk/HIPR2 and Computer Vision resources online - http://www.dai.ed.ac.uk/CVonline

1 Vision Sensing

- The fundamental relationship in imaging a surface is: $I(X, Y) = F(R, G, L)$, where I=intensity of the image at pixel (X,Y), R=Reflectance of the surface, G=Geometry of the surface, and L=Lighting
- Given the image intensities I, we would like to recover the surfaces we have imaged (i.e. depth and orientation at each point on the surface). There are are 2 main problems in inverting this equation:
- Mapping is projection from 3-D to 2-D, which means the inverse is multi-valued (each visible point projects to a unique image point, but each image point "back projects" to a line in space.
- The effects of $R,\ G,\ L$ on intensity of the image are coupled. They can not be easily separated out.
- To make vision systems work, we need to add constraints. Without constraints, vision problem is too hard and too ill-posed to solve

2 Machine Vision

- Why is it machine vision so hard when we can "see" with so little conscious effort?
  - Matrix is not a retina; variable resoltuion in retina
  - Biological systems use active vision. High leve of coordination between eye movements and procesing
  - Biological vision is robust to lighting changes, surface reflectance changes, color changes, resolution changes
- Robot Vision systems are characterized by:
  - Images tend to be binary, not gray scale
  - Resolution reduced to enable real-time processing
  - Lighting is controlled
  - Objects usually in known position and orientation
  - 2-D methods prevail; 3-D methods typically require more computation

- The process of acquiring an image, processing it and understanding its content (i.e. perception) can be thought of as a “Signals to Symbols” paradigm.
- Low-level: image acquisition, noise reduction, enhancement, edge detection.
- Middle-level: Segmentation and region labeling. Surface recovery - depth and orientation (2 1/2-D sketch). Analysis of texture, motion, color, shading etc.
- High-level: Labeling of images with 3-D components, object recognition, functional analysis

LEVELS OF MACHINE VISION

LOW	MIDDLE	HIGH
Digitization	Shape From Methods	Scene Understanding
Compression	-texture	3-D Preception
Enhancement	-motion	Object Recognition
Morphology	-shading	Model Building
Features	-stereo	
edges, corners	Segmentation	
	2 $\frac{1}{2}$-D Sketch	

- The hardest problem in using machine vision is getting the low and high levels integrated.

3 Low Level Vision

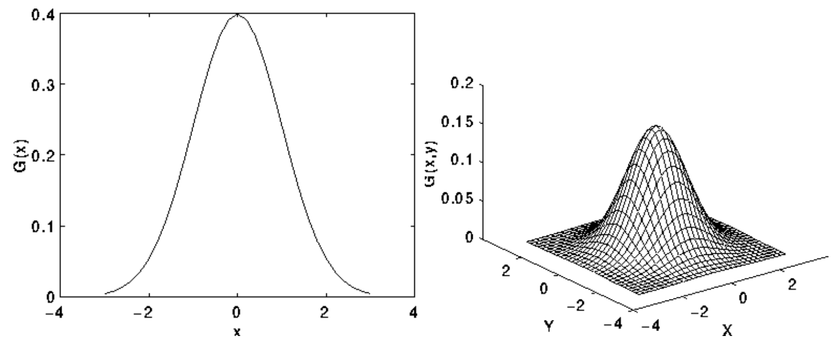
We first acquire images digitally. An Image is a continuous signal that is sampled at discrete spacings called pixels. Each pixel is typically quantized to 8 bits of resolution mononchrome (256 gray levels) or 24 bits for color (8 bits each for the 3 color channels Red, Blue and Green).

Low-lvel vision is a series of weak methods to understand simple scenes. Many common low-level processes use the following idea:

- For each image, construct a new filtered image.
- The filtered image will consist of a weighted sum of the pixels surrounding each pixel in the image. Every pixel gets combined *locally* with the same set of weights.

3.1 Filtering

- Images are subject to noise. Common filters include median filter to reduce spike noise, averaging and Gaussian smoothing filters to remove high frequency components. Filtering can be done in the spatial domain with convolutions or in the frequency domain using Fourier techniques.



$\frac{1}{273}$

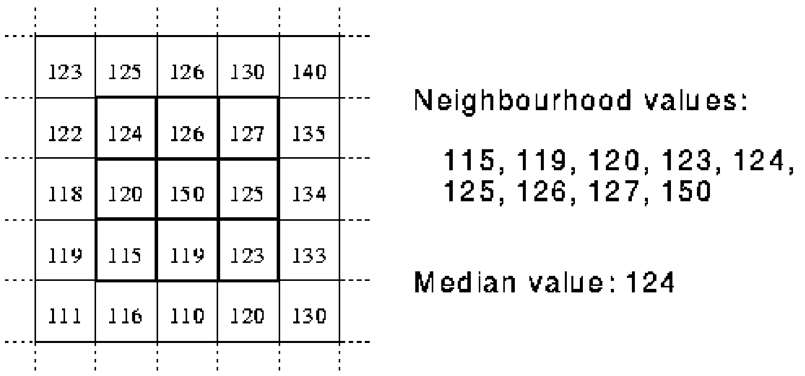
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Figure 1: Gaussian filter. Left: 1-D Gaussian with mean=0 and $\sigma = 1$. Middle: 2-D Gaussian with mean=0 and $\sigma = 1$. Right: $5x5$ convolution mask for Gaussian smoothing with mean=0 and $\sigma = 1$

- *Mean Averaging Filter:* This filter just averages the pixel values in a neighborhood around a pixel. Neighborhood sizes are variable, depending upon the spatial extent of the filter needed. Common sizes are 3x3, 5x5, 7x7 etc. A 3x3 mean filter uses the following set of local weights:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

- *Gaussian Smoothing Filter:* Another smoothing filter is the Gaussian filter, which uses a neighborhood that approximates the fall-off of a Gaussian centered on the pixel of interest. This filter has larger weights for the central pixels and nearest neighbors rather than the mean filter which treats all pixels in the neighborhood with equal weights. See figure 1 above.



123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Figure 2: Median filter. Noisy pixel in center (150) is removed by median of its neighborhood.

- Median Filter: This filter is used to remove outlier noise values in a region. It is based upon *order statistics*, and is a non-linear filter. In this filter, pixels in a neighborhood are sorted by value, and the *median* value of the pixel's in the neighborhood is taken to be the filter's response. If the pixel being processed is an outlier, it will be replaced by the median value. This filter is useful for "shot" or "salt-and-pepper" noise. See figure 2.

3.2 Enhancement

Often, most of the image values will be centered within a limited range of the full 256 gray levels of an image. *Contrast stretching* performs a linear remapping from the gray level range (I_{low}, I_{high}) to $(0, 255)$, effectively "stretching" the contrast in the image. See figure 3. Before the stretching can be performed it is necessary to specify the upper and lower pixel value limits over which the image is to be normalized. Often these limits will just be the minimum and maximum pixel values in the image. For example for 8-bit graylevel images the lower and upper limits might be 0 and 255. Call the lower and the upper limits a and b respectively.

The simplest sort of normalization then scans the image to find the lowest and highest pixel values currently present in the image. Call these c and d. Then each pixel P is scaled using the following function: $P_{out} = (P_{in} - c)(\frac{b-a}{d-c}) + a$

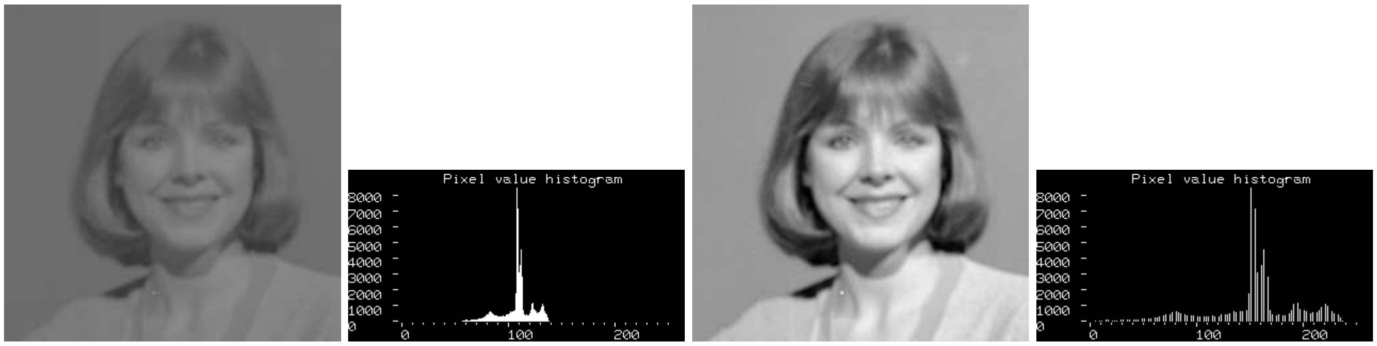


Figure 3: Contrast stretching. Original image and histogram and stretched image and histogram.

Histogram equalization is used to change the response over the entire range of gray values. Often, it is used to create a *uniform* histogram that has all gray values used at the same frequency. This may or may not be useful: large homogeneous regions can get remapped into many gray levels, introducing texture(see figure 4). If an image has R rows and C columns, and there are N gray levels $z_1, z_2, z_3, \ldots, z_n$ total (e.g. 256) then uniform histogram equalization requires each gray value to occur $q = \frac{R \times C}{N}$ times. Using the original histogram, we define $H_{in}[i]$ as the number of pixels in the original image having gray level z_i. The first gray level threshold t_1 is found by advancing i in the input image histogram until q pixels are used. All input image pixels with gray level $< t_1$ will be mapped to gray level z_1 in the output image:

$$\sum_{i=1}^{t_1 - 1} H_{in}[i] \leq q < \sum_{i=1}^{t_1} H_{in}[i]$$

This means that t_1 is the smallest gray level such that the original histogram contains no more thatn q pixels with lower gray values. The kth threshold t_k is defined by continuing the iteration:

$$\sum_{i=1}^{t_k - 1} H_{in}[i] \leq k \cdot q < \sum_{i=1}^{t_k} H_{in}[i]$$

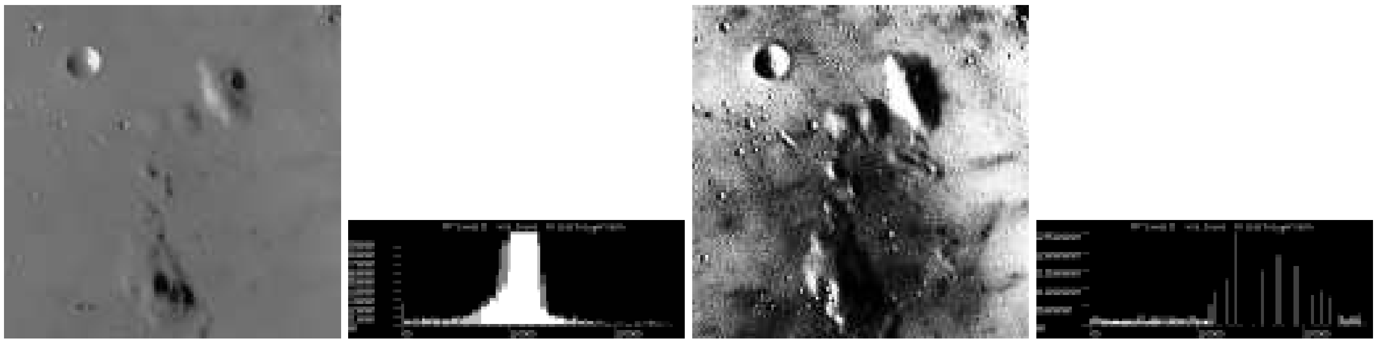


Figure 4: Histogram Equalization. Original image and histogram and equalized image and histogram. See http://www.dai.ed.ac.uk/HIPR2/histeq.htm.

3.3 Edge Detection

Find the gradients at each pixel in the image using a gradient operator. Common edge detection masks look for a derivative of the image intensity values in a certain direction. Derivatives are found by differencing the intensity values. The simplest edge detector masks are:

$$VericalOrientedEdge: \begin{bmatrix} -1 & 1 \end{bmatrix} \quad HorizontalOrientedEdge: \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Each edge detector esentially generates a gradient in the X and Y directions, G_x, G_y. We can calculate the gradient magnitude of the filter's response as:

$$\|G\| \;=\; \sqrt{G_x^2 + G_y^2} \;\; or \;\; \||G\|| \;=\; |G_x| + |G_y|$$

and the edge's orientation (direction) will be $\theta \;=\; atan2(G_y, G_x)$.

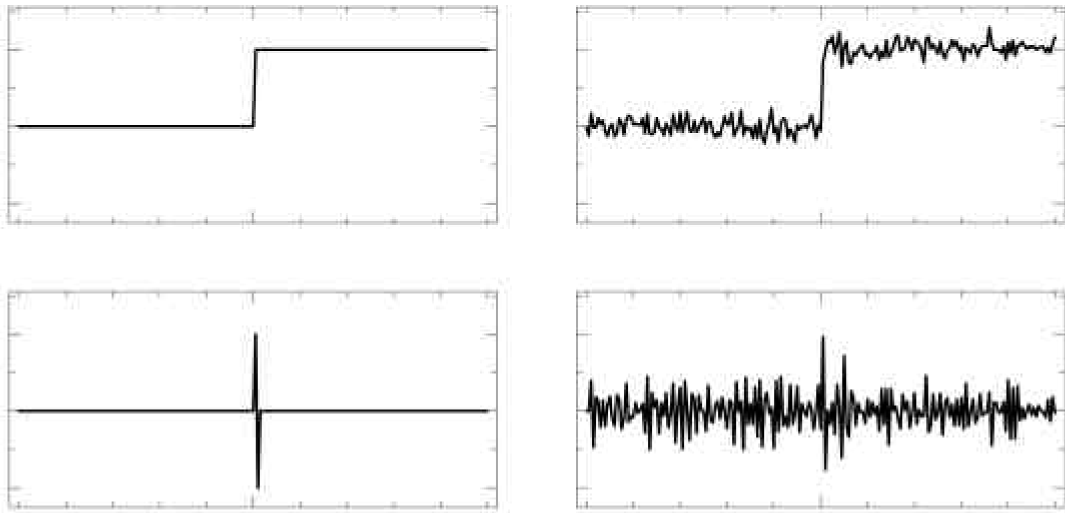
More sophisticated masks include the Sobel Operators:

$$Vertical: \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad Horizontal: \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Zero Crossing of Laplacian of Gaussian

- Identification of features that are stable and match well
- Laplacian of intensity image $$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$
- Convolution with P: $$L = P \otimes I$$ $$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$
- Step / Edge Detection in Noisy Image
  - *filtering through Gaussian smoothing*

$$\begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}$$



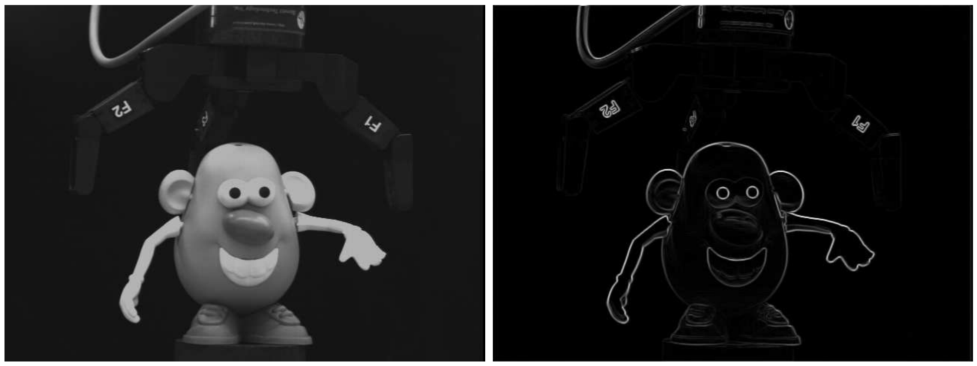


Figure 5: Edge Detection on image. Edge color signifies edge magnitude (brighter == larger magnitude.

4 Middle Level Vision

Middle level vision tries to move beyond the pixel level to larger abstractions including shape and geometry.

4.1 Region Labeling: Recursive Region Growing

Recursive region growing is a simple method. Starting from a binary image, it scans the image for any foreground pixels (not black). For each foreground pixel, it labels that pixel with a unique label, “grows” the pixel by coloring any of its non-black 4-neighbors with this unique color label, and pushing these pixels on a queue. The queue is then processed until empty. All 4-connected pixels in the region will be labeled consistently. Recursive method can be slow however, and may need large memory for recursive calls.

4.2 Region Labeling: Blob Coloring

This algorithm uses 2 passes. The first pass labels each pixel and the second pass merges the labels into a consistent labeling.

Let the initial color, $k = init_{color}$, and choose a color_increment to change the color each time a new blob is found. Scan the image from left to right and top to bottom. Assign colors to each non-zero pixel in pass 1. In pass2, we merge the regions whose colors are equivalent. To maintain the equivalence table between merged colors, we can use a standard disjoint set Union-Find data structure.

```
If $I(x_C) = 0$ then continue
else begin

        if$I(x_U) = 1$ and $I(x_L) = 0$
        then color $(x_C)$: = color $(x_U)$
        if$I(x_L) = 1$ and $I(x_U) = 0$
        then color $(x_C)$: = color $(x_L)$
        if$I(x_L) = 1$ and $I(x_U) = 1$
        then begin /* two colors are equivalent. */
            color $(x_C)$: = color $(x_L)$
            color $(x_L)$ is equivalent to color $(x_U)$
            end
        if$I(x_L) = 0$ and $I(x_U) = 0$ /* new color */
        then color $(x_C)$: = k; k: = k + color_increment

end
```

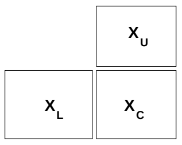


Figure 6: Image topology of x_u, x_c, x_l for region growing

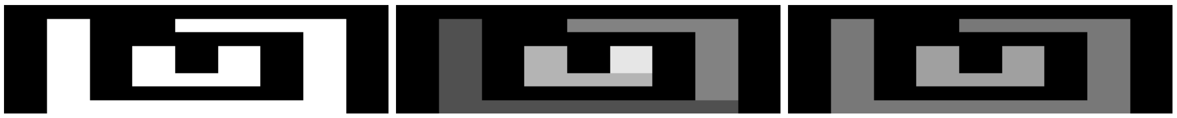


Figure 7: Blob coloring. Left: original binary image. Middle: blob color assignment after first pass. Right: Blob color assignment after merging colors.

Below are 3 ascii images, showing the original test pattern in figure 6, the first pass results, and the final image after region labels are merged. The initial color=80 and the color increment is 50.

```
0    0    0    0    0    0    0    0    0
0    1    0    0    1    1    1    1    0
0    1    0    0    0    0    0    1    0
0    1    0    1    0    1    0    1    0
0    1    0    1    0    1    0    1    0
0    1    0    1    1    1    0    1    0
0    1    0    0    0    0    0    1    0
0    1    1    1    1    1    1    1    0

0    0    0    0    0    0    0    0    0
0   80    0    0  130  130  130  130    0
0   80    0    0    0    0    0  130    0
0   80    0  180    0  230    0  130    0
0   80    0  180    0  230    0  130    0
0   80    0  180  180  180    0  130    0
0   80    0    0    0    0    0  130    0
0   80   80   80   80   80   80   80    0

0    0    0    0    0    0    0    0    0
0  130    0    0  130  130  130  130    0
0  130    0    0    0    0    0  130    0
0  130    0  230    0  230    0  130    0
0  130    0  230    0  230    0  130    0
0  130    0  230  230  230    0  130    0
0  130    0    0    0    0    0  130    0
0  130  130  130  130  130  130  130    0
```

5 Simple Shape Matching

- Template Matching: Simple matching of masks (templates) that contain object's image structure
- Object is represented as a region of pixels. Region is compared against all other positions in the image.
- Measure is absolute value of difference between template pixels and image pixels - zero means exact match. Find minimum response for template operator and this is best match
- Problems: Translation, Rotation, Scaling, Lighting changes between image and template
- Translation is handled by applying template everywhere in image
- Rotation handled by using a set of templates oriented every few degrees. Increases cost

- Scaling is more difficult. Can scale templates but not easily. Not clear how many scales to use.
- Lighting changes can be alleviated by using normalized correlation. Use correlation operator and scale template responses by average intensities of image and template.
- Method of Moments: Use statistical properties of object to match.

$$Continuous: \; M_{ij} = \int\int x^i \, y^j \, f(x,y) dxdy \, ; \;\; Discrete: \; M_{ij} = \sum\sum x^i \, y^j \, f(x,y)$$

- If we assume $f(x,y)$ is a mass function that calculates object mass at each point of the object x, y, then these are the moments of inertia from physics.
- If we further assume $f(x,y)$ is binary valued (1= object present in image, 0= no object at x, y) then we can use these moments as shape descriptors
- M_{00} is simply the area of the object in the image. Counts the pixels that contain the object.
- We can calculate the *centroid* of the object. This is equivalent to the point where an object of uniform mass balances. The mass is equally distributed in all directions.

$$X_c \;=\frac{M_{10}}{M_{00}} \;\; . \;\; Y_c \;=\frac{M_{01}}{M_{00}}$$

- By translating the object coordinates by X_c, Y_c, we can move the object to a known coordinate system. These are *central moments*. Creates translational invariance in moment computation.
- Rotational Invariance can be achieved by finding princiapl axis of object. This is the axis of the moment of least inertia. We can always align an object's principal axis with $X \, Y$ or Z axis.
- Scaling invariance is posible using *normalized moments* which scales by an area measure.
- Higher order moments can be used as unique shape descriptors for an object. Problem: simple scalar measures like this are not robust.

5.1 Finding the Principal Axis

Assume a point set centered on the origin: $(x - x_c, y - y_c)$, where the centroid of the points is (x_c, y_c). To find the principal axis we want to find the rotation angle that will align the axis of minimum intertia with the X axis:

We rotate the points by $-\theta$ to align the dataset with the x axis:

$$ROT(Z, -\theta) \;\; \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \; ; \; \Rightarrow \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \; = \; \begin{bmatrix} xcos\theta - ysin\theta \\ -xsin\theta + ycos\theta \end{bmatrix}$$

Matching using Correlation

- Find locations in an image that are similar to a ***template***
- Filter = template

3	8	3

⇨ test it against all image locations

$I:$

3	2	4	1	3	8	4	0	3	8	7	7

- Similarity measure: Sum of Squared Differences (SSD)

$$\sum_{i=-N}^{N}\left(F(i)-I(x+i)\right)^2=\sum_{i=-N}^{i=N}\left(F(i)\right)^2+\sum_{i=-N}^{i=N}\left(I(x+i)\right)^2-2\underbrace{\sum_{i=-N}^{i=N}\left(F(i)I(x+i)\right)}_{\textbf{Correlation}}$$

$J:$

26	37	21	50	54	1	50	65	59	16	42	17

- Similarity measure: Correlation?

$J:$

30	37	41	29	51	85	56	21	48	86	101	77

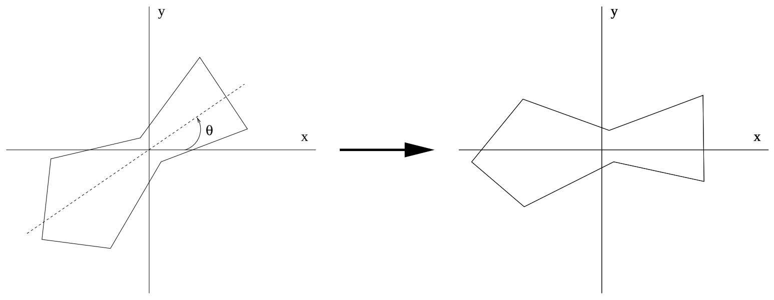


Figure 8: Left: Principal Axis of a 2D object whose centriod is at the origin. Right: rotated object so principal axis is aligned with X axis.

So we can calculate the moments of order 2 for a rotated point set by: $\sum\sum(-xsin\theta + ycos\theta)^2$

These are the moments of order 2 about the X axis for the rotated point set. The term $(-xsin\theta + ycos\theta)$ is the vertical distance from the X axis (i.e. the Y coordinate value) of the transformed point set.

Now, find the value of θ that minimizes that measure. We do this by differentiating with respect to θ, and setting the resulting measure equal to zero:

$$\sum\sum 2(-xsin\theta + ycos\theta)(-xcos\theta - ysin\theta) \ = 0$$

$$2\sum\sum(x^2 sin\theta cos\theta + xysin^2\theta - xycos^2\theta - y^2 sin\theta cos\theta) \ = 0$$

$$2sin\theta cos\theta \sum\sum x^2 + 2(sin^2\theta - cos^2\theta)\sum\sum xy - 2cos\theta sin\theta \sum\sum y^2 \ = 0$$

Using the definition of discrete moments M_{ij}:

$$2sin\theta cos\theta \overline{M_{20}} + 2(sin^2\theta - cos^2\theta)\overline{M_{11}} - 2cos\theta sin\theta \overline{M_{02}} \ = 0$$

where $\overline{M_{ij}}$ refers to *Central Moments*, moments where the centroid is translated to the origin.

Since $sin2\theta \ = \ 2sin\theta cos\theta$ and $cos2\theta \ = \ cos^2\theta - sin^2\theta$, we can substitute to get:

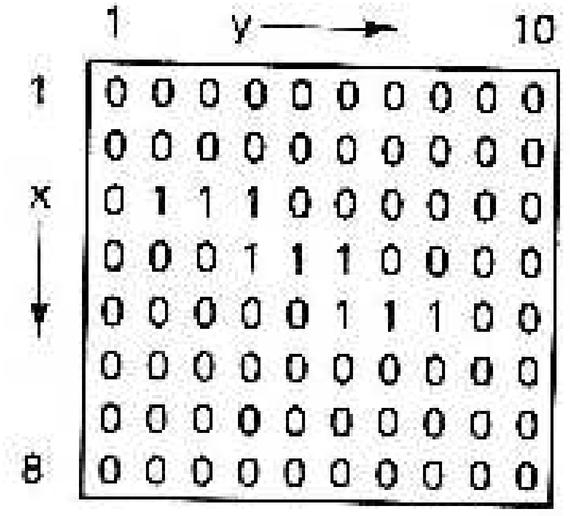
$$sin2\theta \overline{M_{20}} - 2cos2\theta \overline{M_{11}} - sin2\theta \overline{M_{02}} \ = 0$$

and

$$\frac{sin2\theta}{cos2\theta} \ = \ \frac{2\overline{M_{11}}}{\overline{M_{20}} - \overline{M_{02}}}$$

The principal angle is: $2\theta = atan2(2\overline{M_{11}}, \overline{M_{20}} - \overline{M_{02}})$

5.2 Example: Moments



	1			y →						10
1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
x ↓	0	1	1	1	0	0	0	0	0	0
	0	0	0	1	1	1	0	0	0	0
	0	0	0	0	0	1	1	1	0	0
	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0

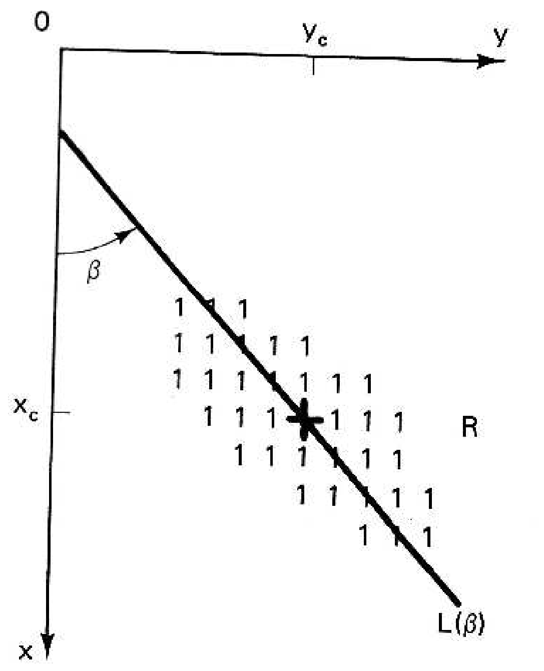


Figure 9: Left: Image used in example below. Right: Idea of Principal angle computation: rotate blob by $-\beta$ so its coincident with X axis

From the image above, we have a region R denoted by values = 1. We can calculate the discrete moments for the region as:

$$M_{ij} = \sum \sum x^i \, y^j \, f(x, y)$$

and $m_{00} = 9$, $m_{01} = 45$, $m_{10} = 36$, $m_{11} = 192$, $m_{02} = 255$, $m_{20} = 150$.

We can create *central moments* by finding the centroid and translating the region so that the origin is the centroid, (x_c, y_c) :

$$Area \;=\; m_{00} \;=\; 9 \quad ; \quad x_c \;=\; \frac{m_{10}}{m_{00}} \;=\; 4 \quad ; \quad y_c \;=\; \frac{m_{01}}{m_{00}} \;=\; 5$$

Finally, the principal angle for the image on the left is computed as $\beta = \frac{atan2(2\overline{M_{11}}, \overline{M_{20}} - \overline{M_{02}}}{2}$:

$$\beta \;=\; \frac{atan2(24, -24)}{2} \;=\; \frac{135}{2} \;=\; 67.5^\circ$$