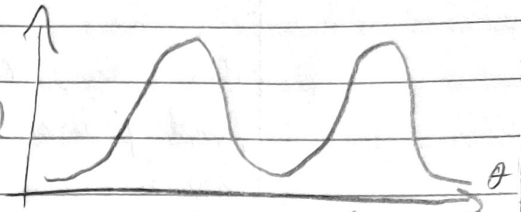
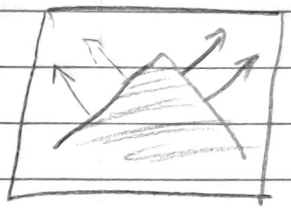
CS-512: ASSIGNMENT-2

- AMAN AGRAWAL (A20446346)

(1) a) In case of a corner, there is more than 1 direction present in the local neighborhood of the point of interest. As a result, >1 peaks are observed in the orientation histogram obtained from the image gradients around the corner

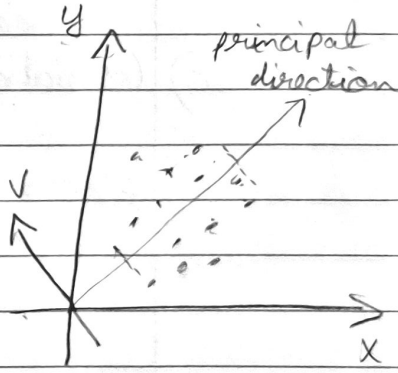


Orientation histogram



Thus, studying the orientation histogram, around the corner region, gives us the number of principal directions i.e. equal to the number of peaks observed in the histogram

b) Let $\{g_i\} = \{x_i, y_i\}$ be a set of image gradients, shown in the plot $\Rightarrow$



Thus, the principal direction is the one where the variance of the points is large. But, in 2 dimensions, the second principal direction, V has to be perpendicular to the first principal direction. As a result, find direction V s.t. projection of $\{g_i\}$ onto V is minimized

$$\therefore E(V) = \sum_i (g_i \cdot V)^2$$ (Squaring to avoid +ve & -ve projections to cancel out)

$$= \sum_i (g_i^T V)(g_i^T V)$$

$$= \sum_i (v^T g_i)(g_i^T v) = \sum_i v^T (g_i g_i^T) v$$

$$= v^T \left(\sum_i g_i g_i^T\right) v = v^T C v$$

This way, PCA (Principal Component Analysis) is used to find principal directions of gradient orientations in a local patch.

$$E(v) = v^T C v$$

$$v^* = \underset{v}{\operatorname{argmin}}\, E(v) \Rightarrow \nabla E(v) = 0$$

$$\Rightarrow 2Cv = 0$$

$\Rightarrow$ solution is eigenvector belonging to the smallest eigenvalue

Since eigenvalue = variance in corresponding principal direction.

c) Correlation matrix $C = \sum_i g_i g_i^T$

$$\Rightarrow C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 9 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 16 \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} + \begin{bmatrix} 1 & 3 \\ 3 & 9 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 6 \\ 6 & 44 \end{bmatrix}$$

d) To detect corners, the product of eigenvalues of the gradient correlation matrix should be greater than a threshold value i.e., $\lambda_1 \cdot \lambda_2 > \tau$

e) Suppose, a corner in an image exists in multiple overlapping windows. Thus, to avoid detection of corner at multiple points within the window, we use non-maximum suppression as follows:-
- compute $\lambda_1 \cdot \lambda_2$ for all nodes
- select windows with $\lambda_1 \cdot \lambda_2 > \tau$ & sort in decreasing order
- select the top of the list as corner, and delete all other corners in its neighborhood from that list.
- stop once x% of the points detected as corners.

f) The algorithm of Harris corner detection is:-
- compute correlation matrix C for window
- compute cornerness measure

$$C(c) = \det(c) - K\, tr^2(c)$$

- Detect corner where $C(c)$ is high.

where $K \in [0, 0.5]$ is selected by user

if $K = 0$ $C(c)$ detects corners

$K = 0.5$ $C(c)$ detects edges

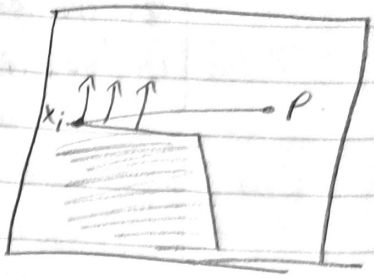
As it can be seen, in Harris corner detection we don't calculate eigenvalues. However, in reality,

$$\det(c) = \lambda_1 \cdot \lambda_2$$

$$tr^2(c) = (\lambda_1 + \lambda_2)^2$$

Thus, this method avoids calculating the eigenvalues.

g) Given that there is a corner in a window, to compute better localization of a corner :-



- connect each point x_i to p & project the gradient at x_i onto $(x_i - p)$
- The best 'p' will minimize the sum of all the projections

$$\therefore E(p) = \sum_i \left(\nabla I(x_i) \cdot (x_i - p)\right)^2$$

$$= \sum_i (x_i - p)^T \left(\nabla I(x_i) \cdot \nabla I(x_i)^T\right) (x_i - p)$$

$$p^* = \underset{p}{\operatorname{argmin}}\ E(p)$$

$$\Rightarrow \nabla E(p) = 0$$

$$-2 \sum_i \left(\nabla I(x_i)\, \nabla I(x_i)^T\right)(x_i - p^*) = 0 .$$

$$\Rightarrow \underbrace{\sum \nabla I(x_i) \nabla I(x_i)^T}_{C}\ p^* = \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

$$\Rightarrow p^* = C^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

In order for p^* to exist, C^{-1} should exist, i.e. C should be non-singular. But, if C was singular it would have a zero eigenvalue. This is not possible, since we already detected a corner in the window and $\lambda_1 \cdot \lambda_2 > \tau$. Thus, C must be non-singular.

b) A good characterization of feature points satisfies the following desired properties:

- translation invariance, → solved using smaller window
- rotation invariance → solved using histograms
- scale invariance → solved by forming image pyramids
- illumination invariance → solved through gradients

Feature points can be characterized using HOG through following algorithm:-

- split each patch of the window into cells (possibly overlapping)
- create orientation histogram in each cell, using edge or gradient directions (possibly weighted by distance from center or gradient magnitude)
- concatenate orientation histograms

i) SIFT or Scale Invariant Feature Transform works similar to HOG except:-
- it uses weighted sum to create orientation histograms in cells & then concatenate
- align histograms based on the dominant direction. This also leads to rotation invariance.

② a) When using the Hough transform, we represent a point in co-ordinate space to a line in parameter space, which tells the possible values of slope & y-intercept of a line passing through the point in the co-ordinate space. However, in order to represent this, the possible values of slope and y-intercepts were plotted on the parameter space. But, this poses a problem. The values of slope can range between $(-\infty, \infty)$ In order to fully represent all the values in the parameter space, the slope axis has to be infinitely long. This is clearly not feasible, and hence Hough transform is difficult to perform.

b) Any point (x, y) on the line satisfies

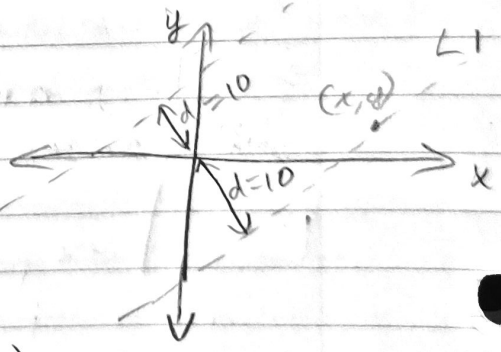
$$(x, y) \cdot (n_x, n_y) = d$$

For a line with slope 45°,

$$(n_x, n_y) = (\pm K, \pm K)$$

K = constant

Unit normal $= \left(\frac{\pm 1}{\sqrt{2}}, \frac{\pm 1}{\sqrt{2}}\right)$.



$\Rightarrow$ Eqn of line: $\mp \frac{x}{\sqrt{2}} \pm \frac{y}{\sqrt{2}} = 10$

$\therefore$ The 2 line equations are:-

$-x + y = 10\sqrt{2} \quad \rightarrow L2$

$x - y = 10\sqrt{2} \quad \rightarrow L1$

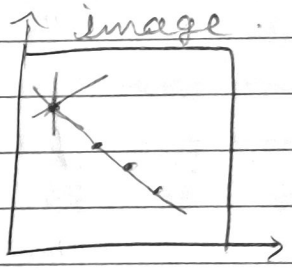
Both L1 & L2 satisfy the explicit line eqn $\Rightarrow ax + by + c = 0$

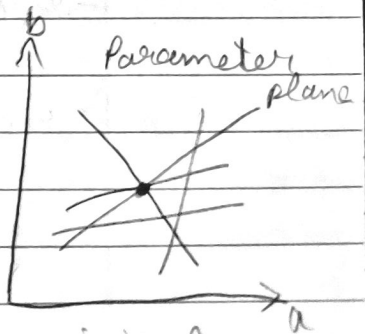
where $a = \pm \frac{1}{\sqrt{2}} \quad b = \mp \frac{1}{\sqrt{2}}, \quad c = -10$

c) When using polar representation of lines, the vote of each point in the image looks like a sinusoidal curve in the parameter plane. This is because, the distance d is represented as combination of two sinusoidal functions in polar coordinates

$\Rightarrow d = x\cos\theta + y\sin\theta$

d) Each point in an image represents a line in the parameter plane. This line can be considered as a vote for the right set of parameters 'a' & 'b' of $y = ax + b$ in the image, which passes through that point.





Taking the vote count from all the points, we find the point in the parameter plane, where maximum lines intersect. This point (a, b) is taken for original line representation: $y = ax + b$

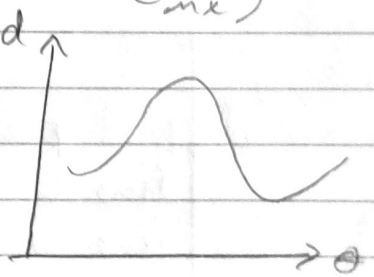
c) The trade off regarding the bin size in the parameter plane -
- Larger bins are more efficient, but provide less localization. On the other hand;
- Smaller bins are more accurate in terms of localization, but are less efficient, as it takes more time computationally to accurately pinpoint the right bin

d) The slope-intercept form of line equation is impossible to represent fully in the parameter plane, as the slope of line lies in the unbounded region :- $(-\infty, \infty)$
Also, it is difficult to represent vertical lines in this plane
However, if normal at each point is known, the line passing through this point and distance (d) from origin can be used to represent the normal form of line equation :-

$$(n_x, n_y) \cdot (x, y) = d$$

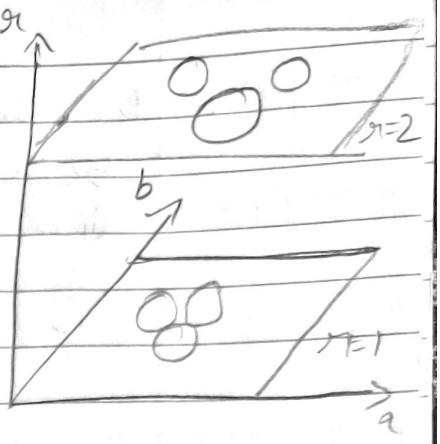
or $d = x\cos\theta + y\sin\theta$ where $\theta = \tan^{-1}\left(\frac{n_y}{n_x}\right)$

Thus, a point in this line can be represented in the parameter plane as a sinusoidal curve ⇒



This allows for :-
- representation of vertical lines with $\theta = 0$
- the parameters θ & d are constrained. Thus, better representation of parameter plane is observed.

(2) There are 3 dimensions of parameter space, when using Hough transform for circles



⇒ Circle equation :-

$$(x-a)^2 + (y-b)^2 = r^2$$

or

$$x = a + r\cos\theta$$
$$y = b + r\sin\theta$$

⇒ There are 3 parameters a, b, r.

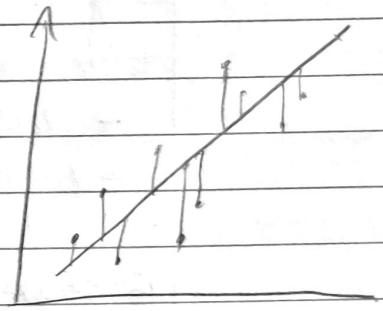
Thus, given (x, y) vote in each r plane using

$$a = x - r\cos\theta$$
$$b = y - r\sin\theta \quad \text{where } \theta \in [0, 360^\circ]$$

∴ Each point (x, y) casts a circle vote in each r plane.

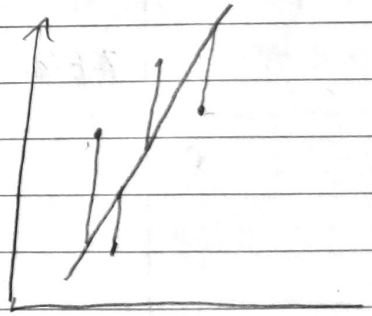
(3) a) When fitting a line $y = ax + b$, we minimize the objective fxn

$$E(a, b) = \sum_i (y_i - (ax_i + b))^2$$



This is basically the sum of squares of vertical distance between the point and the predicted line

This works fine until the slope of line increases. At this point the objective fxn $E(a, b)$ is not a good measure, as the vertical distance isn't a good indicative as the measure of error



(3) b) $p \cdot \hat{n} = d$

$\Rightarrow \frac{1}{\sqrt{1^2+2^2}}(x,y) \cdot (1,2) = 2$

$\Rightarrow (x,y) \cdot (1,2) = 2\sqrt{5}$

$\Rightarrow x + 2y = 2\sqrt{5} \Rightarrow x + 2y - 2\sqrt{5} = 0$

$\therefore l = \begin{bmatrix} 1 \\ 2 \\ -2\sqrt{5} \end{bmatrix}$

c) Given $\{x_i, y_i\}_{i=1}^{m}$ find the parameters of the explicit line equation $y = ax + b$

a = slope

b = y-intercept

In order to find a, b

- define the objective fxn. $E(a,b) = \sum_i (y_i - (ax_i + b))^2$
- Find $a^*, b^* = \underset{a,b}{\operatorname{argmin}} E(a,b)$

To solve this, put $\nabla E(a,b) = 0 \Rightarrow a^*, b^*$

$$E(a,b) = \sum_i (y_i - (ax_i + b))^2$$

$$\nabla E(a,b) = \begin{bmatrix} \partial E/\partial a \\ \partial E/\partial b \end{bmatrix} = \begin{bmatrix} \sum 2(y_i - ax_i - b)(x_i) \\ \sum 2(y_i - ax_i - b)(-1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & \sum 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum x_i y_i \\ \sum y_i \end{bmatrix}$$

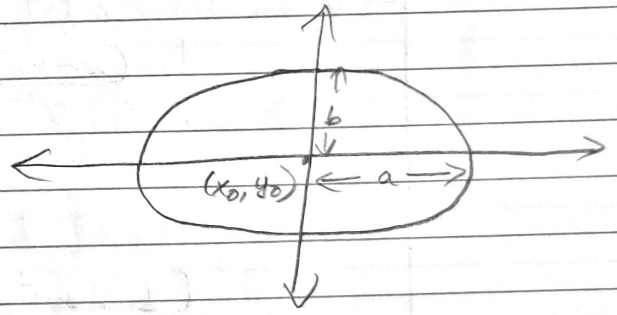
$$\Rightarrow \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i \\ \Sigma x_i & m \end{bmatrix}^{-1} \begin{bmatrix} \Sigma x_i y_i \\ \Sigma y_i \end{bmatrix}$$

d) The 3x3 matrix required is $\sum_i p_i p_i^T$

$$\Rightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 6 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & 6 & 1 \end{bmatrix}$$

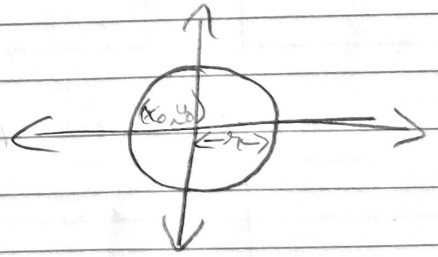
$$\Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 3 & 1 \\ 3 & 9 & 3 \\ 1 & 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 12 & 2 \\ 12 & 36 & 6 \\ 2 & 6 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 15 & 3 \\ 15 & 46 & 10 \\ 3 & 10 & 3 \end{bmatrix}$$

e) The explicit equation of an ellipse aligned to the axes, and centered at (x_0, y_0) is:-



$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1$$

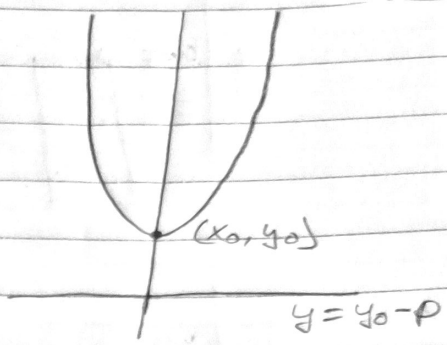
The explicit equation of a circle of radius 'r' and centered at (x_0, y_0) is:-



$$(x-x_0)^2 + (y-y_0)^2 = r^2.$$

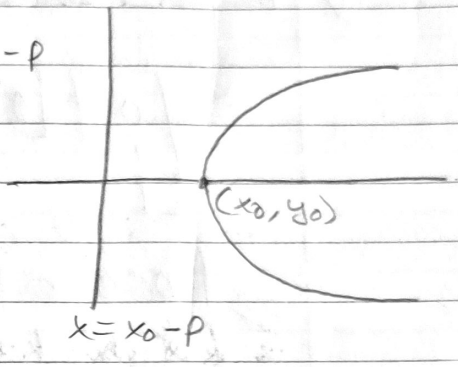
Explicit equation of a parabola with vertex (x_0, y_0) and directrix $y = y_0 - p$

$$\Rightarrow Y = \frac{1}{4p}(x - x_0)^2 + y_0$$



Explicit equation of a parabola with vertex (x_0, y_0) and directrix $x = x_0 - p$

$$\Rightarrow x = \frac{1}{4p}(y - y_0)^2 + x_0$$

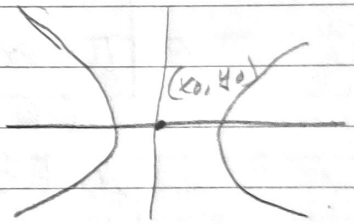


Explicit equation of a hyperbola, centred at (x_0, y_0) and

- with horizontal transverse axis

$$\frac{(x - x_0)^2}{a^2} - \frac{(y - y_0)^2}{b^2} = 1$$

$\Rightarrow$



- with vertical transverse axis

$$\frac{(y - y_0)^2}{a^2} - \frac{(x - x_0)^2}{b^2} = 1$$

Implicit equation of a conic curve:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

To guarantee an ellipse, $b^2 - 4ac < 0$

f) For an Implicit ellipse equation :- $l^T p = 0$

where l = model parameters = (a, b, c, d, e, f)

$p = (x^2, xy, y^2, x, y, 1)$

We need to find parameters l by minimizing the algebraic distance :-

$$E(l) = \sum_{i=1}^{m} (l^T p_i)^2 = l^T S l$$

where $S = \sum_{i=1}^{m} p_i p_i^T$

Then $l^* = \underset{l}{\operatorname{argmin}}\, E(l)$

To fit an ellipse we have to ensure that $b^2 - 4ac < 0$.

$\Rightarrow b^2 - 4ac < 0 \Rightarrow l^T C l = -1$

where

$$C = \begin{bmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus, modify the objective function as :-

$$E(l) = l^T S l + \lambda (l^T C l + 1)$$

$$l^* = \underset{l}{\operatorname{argmin}}\, E(l)$$

$\Rightarrow \nabla E(l) = 0$

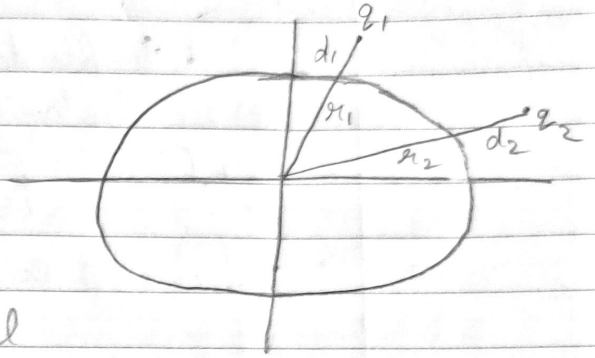
$\Rightarrow S l = \lambda C l \Rightarrow l = \lambda S^{-1} C l$

$\therefore l^*$ is the eigenvector of $S^{-1}C$ belonging to the negative eigenvalue

Since we used the algebraic distance, $\ell^T x_i$ this means:-
- when x_i is on the ellipse: $\ell^T x_i = 0$
- when x_i is off the ellipse: $q_i = \ell^T x_i$ provides a measure for distance from the ellipse

This can be approximated as:

$$q_i = \ell^T x_i \approx \frac{d_i}{d_i + r_i}$$



Thus, if $d_1 \approx d_2$, then still

$q_1 > q_2$ because $r_2 > r_1$

Thus, points closer to the short axis more affect the fitting process, as the algorithm finds these points at a greater distance to ellipse than the points closer to the long axis. Therefore the model tries to fit these points more aggressively by putting higher penalties to these points.

g) In order to fit an ellipse using geometric distance for objective function:-

$$E(\ell) = \sum_i \frac{|f(p_i, \ell)|}{|\nabla f(p_i, \ell)|}$$

where $f(p_i, \ell) = \ell^T p_i$

$$\ell^* = \underset{\ell}{\operatorname{argmin}}\, E(\ell) \quad \text{s.t. } b^2 - 4ac < 0$$

$$p_i = (x_i^2, x_i y_i, y_i^2, x_i, y_i, 1)$$
$$\ell = (a, b, c, d, e, f)$$

But, this introduces a complication when trying to detect the parameters by minimizing the function $E(\ell)$.

The function $E(\ell)$ is not a linear function. Hence, there is no explicit solution to $\nabla E(\ell) = 0$. It can be solved only using iterative methods such as Gradient descent, etc.

b) The objective function to minimize for active contour

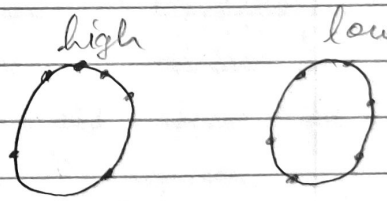
Define error functional :-

$$E[\phi(s)] = \int_{\phi(s)} \underbrace{(\alpha(s) E_{cont} + \beta(s) E_{curv}}_{\text{internal energy}} + \underbrace{\gamma(s) E_{img}}_{\text{external energy}}) ds$$

where :-

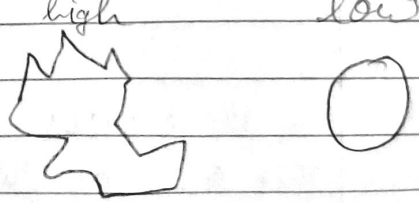
- $\phi(s)$ = parametric curve to represent contour.
- $\alpha(s)$, $\beta(s)$ & $\gamma(s)$ are coefficients of different energy terms.
- Continuity energy (E_{cont}) defines the continuous property of the contour

$$E_{cont} = \left| \frac{d\phi}{ds} \right|^2$$



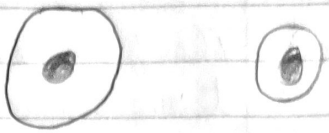
- Curvature energy (E_{curv}) defines the curvature property of the contour

$$E_{curv} = \left| \frac{d^2\phi}{ds^2} \right|$$



- Image Energy (E_{img}) defines how close the contour is to the edge of the object in the image

$$E_{img} = -|\nabla I|^2$$



i) In discreet case, the parametric curve $\phi(s)$ is represented as a set of discreet points

$$\Rightarrow \quad \phi(s) \rightarrow \{p_i\}_{i=1}^{n}$$

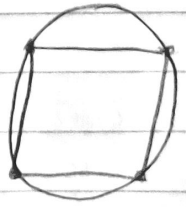
The continuity (E_{cont}) and curvature (E_{curv}) are estimated as:-

$$E_{cont} = \left|\frac{d\phi}{ds}\right| = |p_{i+1} - p_i|^2$$

$$E_{curv} = \left|\frac{d^2\phi}{ds^2}\right| = |(p_{i+1} - p_i) - (p_i - p_{i-1})|^2$$

$$= |p_{i+1} - 2p_i + p_{i-1}|^2$$

ii) In certain cases, we allow high curvatures on contours to obtain a better fit. Example: corners of a square



In these cases we set β_i = coefficient of $E_{curv} = 0$
This relaxes the continuity of the curve, as it makes

the curve piecewise continuous by allowing discontinuity at the corners where $p_i = 0$ if curvature at p_i is greater than threshold τ

i.e. when $|p_{i+1} - 2p_i + p_{i-1}| > \tau$

(4) a) Correlation matrix $C = \sum_i m_i m_i^T$

$$C = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} + \begin{bmatrix} 9 & 12 \\ 12 & 16 \end{bmatrix} = \begin{bmatrix} 10 & 14 \\ 14 & 20 \end{bmatrix}$$

b) $$\frac{20-10}{20-10} = \frac{y-10}{x-10}$$

$$\Rightarrow x-10 = y-10 \Rightarrow x-y=0$$

Normalized normal to this line $\Rightarrow \left(\frac{-1}{\sqrt{2}}, \frac{+1}{\sqrt{2}}\right)$

c) Line equation

$$\Rightarrow \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

$$\Rightarrow x + 2y + 3 = 0$$

when $x = 2 \Rightarrow y = -2.5$

the curve piecewise continuous by allowing discontinuity at the corners where $\beta_i = 0$ if curvature at p_i is greater than threshold τ

i.e. when $|p_{i+1} - 2p_i + p_{i-1}| > \tau$

(4) a) Correlation matrix $C = \sum_i n_i n_i^T$

$$C = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} + \begin{bmatrix} 9 & 12 \\ 12 & 16 \end{bmatrix} = \begin{bmatrix} 10 & 14 \\ 14 & 20 \end{bmatrix}$$

b) $$\frac{20-10}{20-10} = \frac{y-10}{x-10}$$

$$\Rightarrow x-10 = y-10 \Rightarrow x-y=0$$

Normalized normal to this line $\Rightarrow \left(\frac{-1}{\sqrt{2}}, \frac{+1}{\sqrt{2}}\right)$

c) Line equation

$$\Rightarrow \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

$$\Rightarrow x + 2y + 3 = 0$$

when $x = 2 \Rightarrow y = -2.5$

e) Matrix formed when performing line fitting

$$\Rightarrow \sum_i p_i p_i^T$$

$$\Rightarrow \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} \begin{bmatrix} 3 & 4 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 3 & 6 & 3 \end{bmatrix} + \begin{bmatrix} 9 & 12 & 3 \\ 12 & 16 & 4 \\ 3 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 14 & 4 \\ 14 & 20 & 6 \\ 6 & 10 & 4 \end{bmatrix}$$

f) $D(f,p) \approx \dfrac{|f(p)|}{|\nabla f(p)|} = \dfrac{1}{2}$

g) Approx. algebraic distance $= |f(p)| = 1$

h) $E_{cont} = |p_{i+1} - p_i|^2 = |(3,4) - (2,3)|^2$

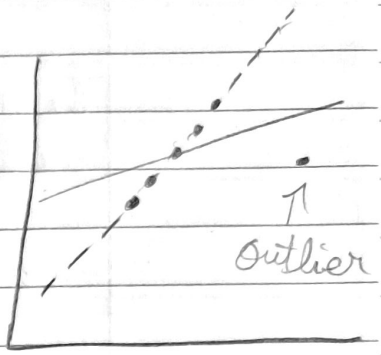
$= |(1,1)|^2 = 1$

$E_{curv} = |p_{i+1} - 2p_i + p_{i-1}|^2 = |(3,4) - 2(2,3) + (1,2)|^2$

$= |(0,0)|^2 = 0$

i) At points with high curvature (e.g. corner), the coefficient β is set to 0, for that location to guarantee tight fitting of an active contour.

(5) a) Outliers are points in an image that do not follow the pattern like most other points. Usually, they are the noise in the image.



However, in model fitting, they can significantly vary the results and deviate the learnt model from the more accurate one, in the process of minimizing the objective function. Ex, as shown in the line-fitting model, due to the presence of an outlier, the line fit by the model (solid line) is quite different from the more accurate fit (dotted line). This fit could have been obtained if only the outlier point was absent.

b) When using Mean Squared Error (MSE), the error term is defined as :-

$$E(\theta) = \sum_i d^2(x_i, \theta)$$

But, this is sensitive to outliers due to the squared penalty

For a more robust estimation, the error-term is defined as :-

$$E(\theta) = \Sigma \rho_\sigma(d(x_i, \theta))$$

NOTE) MSE is a special case where $\rho_\sigma(x) = x^2$

c) Geman-McClure function for robust estimation:-

$$\rho_\sigma(x) = \frac{x^2}{x^2+\sigma^2}$$

The main advantage of the function is:-

- when $x >> \sigma$, then $\rho_\sigma(x) = 1$

- and when $x << \sigma$, then $\rho_\sigma(x) = \frac{x^2}{\sigma^2}$

Thus, when x is big, the error term gives the output 1 instead of high output given by MSE. Also, when x is small, the Geman-McClure function behaves like a parabola. This is particularly advantageous since it is not sensitive to outliers.

The effect of σ on curve fitting:-

- large σ $\Rightarrow$ include more points
- small σ $\Rightarrow$ include fewer points

Thus, while estimating in an iterative manner, start with a higher value of σ. And as the model starts to converge, reduce the value of σ.

$$E(\theta) = \sum_i \rho_\sigma(d(x_i, \theta))$$

$$\nabla E(\theta) = \sum \frac{\partial}{\partial d} \rho_\sigma(d) \frac{\partial}{\partial \theta} d(\theta)$$

For $\rho_\sigma(d) = d^2 \Rightarrow \frac{\partial}{\partial d} \rho_\sigma(d) = 2d$

For $\rho_\sigma(d) = \frac{d^2}{d^2+\sigma^2} \Rightarrow \frac{\partial}{\partial d} \rho_\sigma(d) = \frac{2d\sigma^2}{(d^2+\sigma^2)^2}$.

As a result:-

- MSE has error gradient of $2d$ which is very large
- Geman-McClure has error gradient of

$$\frac{2d\sigma}{(\sigma^2+d^2)^2} \approx 0$$

Thus, it is less sensitive to the outliers

d) $$f_{\sigma=1}(x=1) = \frac{1^2}{1^2+1^2} = \frac{1}{2}$$

e) The principle behind RANSAC algorithm is:-

- perform multiple experiments by taking a random sample of points
- choose the best results
- the number of points drawn for each sample should be small so as to reduce the number of outliers present in each sample. This also increases the probability of having a sample with no outlier present in one of the experiments.

f) Parameters of the RANSAC algorithm are:-

n = no. of points drawn at each evaluation
d = minimum no. of points needed to estimate model
K = no. of trials
t = distance threshold to identify inliers. i.e. a sample point in an experiment is considered an inlier if distance of the point from the model is less than t.

In order to estimate the no. of trials (K) use:-

p: with probability of p at least one experiment does not have an outlier or at least one experiment succeeds

w: probability that a point is an inlier

initially $w = 0.5$, but in subsequent iterations:

$$w = \frac{\#\text{ inliers}}{\#\text{ points}}$$

∴ probability that all K experiments fail:-

$$(1-p) = (1-w^m)^K$$

$$\Rightarrow \log(1-p) = K \log(1-w^m)$$

$$\Rightarrow K = \frac{\log(1-p)}{\log(1-w^m)}$$

Thus for large p and small w, we get large values of K

g) $$K = \frac{\log(1-p)}{\log(1-w^m)} = \frac{\log(1-0.99)}{\log(1-(0.9)^m)}$$

$$= \frac{\log(0.01)}{\log(1-(0.9)^m)}$$

(6) a) The objectives of image segmentation are:-
- separate object/foreground from background
- find contours of objects
- semantic image segmentation; i.e. to label each pixel in the image with class label

b) Agglomerative Segmentation
- start with each pixel in a separate cluster
- merge clusters with small distance
- repeat while clusters are not satisfactory

Divisive segmentation
- start with all pixels in one cluster
- split clusters to produce large distance between them
- repeat while clusters are not satisfactory.

c) The K-means algorithm for clustering pixels with spatial context is:-
- select K
- select initial guess of means: $m_1, m_2 \ldots m_K$
- Repeat while m_j change;
  - $\rightarrow l_i = \underset{j \in [1..K]}{\operatorname{argmin}} \|f_i - m_j\|^2$ for each pixel assign labels
  - $\rightarrow S_j = \{i \mid l_i = j\}$ update clusters
  - $\rightarrow m_j = \sum_{i \in S_j} f_i / \#S_j$ recompute means

d) In graph cuts,
- cost of a cut is the sum of the links being removed:

$$cut(A,B) = \sum_{p \in A, q \in B} w_{pq}$$

- And the minimum cut is the cut with the lowest cost.
- But minimizing the cost of the cut, reduces to cuts resulting in unbalanced subgraphs or single nodes

Thus, in normalized cuts,
- it assigns the cost taking into account the size of clusters produced as a result of the cut

$$- \quad N_{cut}(A,B) = \frac{cut(a,b)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

$$vol(A) = \sum_{p \in A, q \in A \cup B} w_{pq}$$

e) Similarity matrix (W):

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,100} \\ w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,100} \\ \vdots & & & & \vdots \\ w_{100,1} & w_{100,2} & w_{100,3} & \cdots & w_{100,100} \end{bmatrix}$$

Weighted degree matrix (D):

$$\begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_{100} \end{bmatrix}$$

where $d_i = \sum_{j=1}^{n} w_{i,j}$

laplacian matrix (D-W) :

$$\begin{bmatrix} d_1 - w_{1,1} & -w_{1,2} & \cdots & -w_{1,100} \\ -w_{2,1} & d_2 - w_{2,2} & \cdots & -w_{2,100} \\ \vdots & \vdots & & \vdots \\ -w_{100,1} & -w_{100,2} & \cdots & d_{100} - w_{100,100} \end{bmatrix}$$

where

W is the adjacency matrix, and
D is the degree matrix of the graph

e) A cut can be specified using a vector (x) whose length is the number of nodes if the elements in the vector belong to $\{+1, -1\}$

- If $x(i) = 1 \iff i \in$ cluster A
- If $x(i) = -1 \iff i \in$ cluster B

g) To determine a minimum normalized cut, solve the optimization problem:

$$\min_x N_{cut}(x) \cong \min_y \frac{y^T (D-W) y}{y^T D y}$$

s.t. $y^T D 1 = 0$

$1 \equiv [1 \cdots 1]$

where $y = (1+x) - b(1-x)$

and $b = \dfrac{\sum_{x_i > 0} d_i}{\sum_{x_i < 0} d_i}$

h) To find minimum normalized cut, using continuous variables :-

- solve, $\min_y \frac{y^T(D-W)y}{y^TDy}$ s.t. $y^TD1 = 0$

Rayleigh Quotient

- this solution exists $(D-W)y = \lambda Dy$

- The first eigenvector (belonging to $\lambda = 0$) is $[1 \cdots\cdots 1]$

- Then second smallest eigenvector is the solution

i) The eigenfaces approach for object recognition is :-

- map image to lower dimensional vectors (e.g. 64) using PCA

- measure similarity to templates in lower dimension space.

One limitation with this approach is that it is less sensitive to smaller variations in the image. Thus, it may fail to differentiate objects that might resemble each other

j) For object recognition using bag-of-words approach:-
- extract image features such as SIFT, HOG, etc.
- cluster features to create a codebook (dictionary) using a clustering algorithm such as K-means
- compute a distribution of code words in each class
- classify using distribution of code words

One drawback of this approach is that, even if the image window consists only a part of the object, the algorithm will identify the window as the whole object. For example, if the image window consists of an eye only, the BOW approach might still classify it as an image of a face.