Name : Yash Manharbhai Patel

Student ID : A20451170

Course : CS512 – Computer Vision

Semester : Fall20

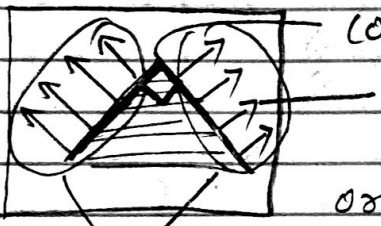
CS-512 - Assignment-2

Ans-1) a)

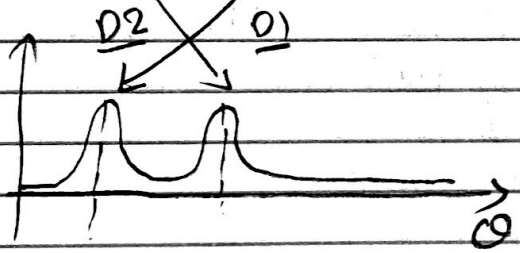
for corner detection we use orientation histogram.

orientation histogram shows frequency of vectors at angle of gradient vector (θ)

when we use windows in an image we get corner like below

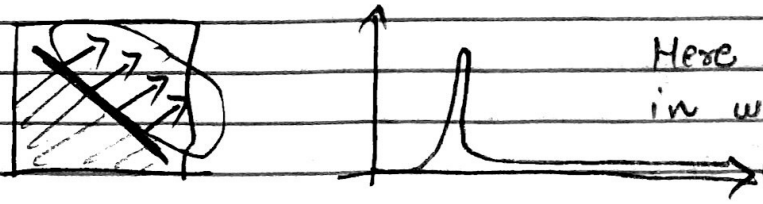


orientation histogram as below



Here we have two principal directions
It can be more than one

In the image gradient vectors on the same edge have same directions. So, frequency of gradient vector at same angle is higer. So, we get two peeks here.



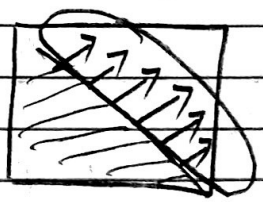
Here we have only edge in window. So, we get only one peek

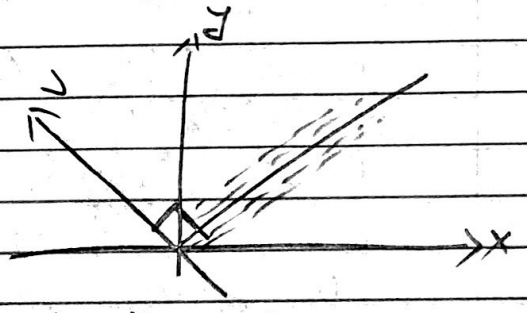
below is how number of principal directions assessed:

a)- find correlation matrix of gradients in local window
- find eigenvalues of correlation matrix
- detect corner in window if eigenvalues are sufficiently large

b) In PCA we find direction vector V such that projection of all gradient vectors in local window onto V is minimized.

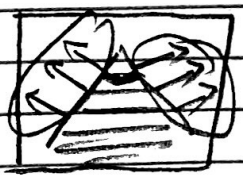
in edge

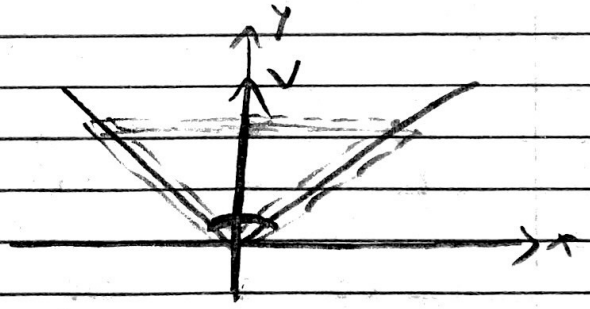




In edge all gradient vectors are almost in same direction so we choose V at orthogonal direction.

in corner





$$E(V) = \sum_i (g_i \cdot V)^2 = V^T \left(\sum_i g_i g_i^T \right) V = V^T C V$$

$$C = \sum_i g_i g_i^T = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i y_i & \sum y_i^2 \end{bmatrix} = D^T D$$

$$D = \begin{bmatrix} g_1^T \\ \vdots \\ g_M^T \end{bmatrix}$$

c) gradient vectors :

$\{(0,0), (0,1), (0,2), (0,3), (0,4), (1,0), (1,1), (1,2), (1,3)\}$

$$C = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 6 & 44 \end{bmatrix}$$

d) To detect corner using PCA we find a vector V on which projection of all gradient vectors are minimized.

$E(v) = v^T C v$, C = correlation matrix

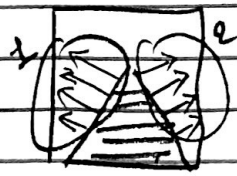
$\nabla E(v) = 0$

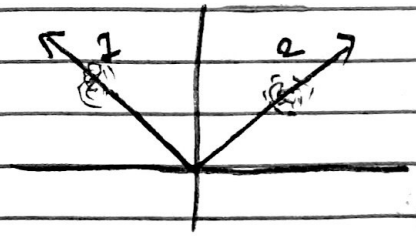
$2Cv = 0$

$Cv = 0$

to find vector V we find eigenvalues of correlation matrix C.
here we get two eigenvalues λ_1, λ_2

In Corner





for corners we get both λ_1, λ_2 large.

In given window it's a corner

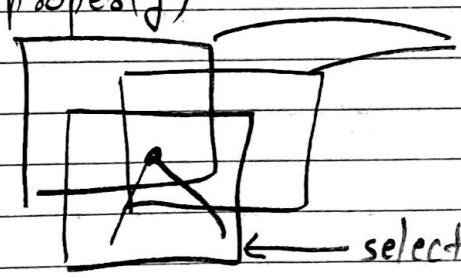
if $\lambda_1 \cdot \lambda_2 > \tau$ ← Threshold

eigen value is variance in principal direction.
V is eigenvector belonging to smallest eigenvalue

e) In non-maximum suppression,

- we first find eigenvalues λ_1, λ_2 for each window in an image
- from all these windows we select windows with $\lambda_1 \cdot \lambda_2 > \Sigma$ and then sort these selected windows in decreasing order of $\lambda_1 \cdot \lambda_2$
- select top window from the list and remove other windows from list which lies in neighborhood of selected window.

(Here top selected window define corner properly)



A window is in neighborhood if distance between centers of two windows is less than window size

- stop once x % of corner points are detected

we use overlapping window to find corners between windows. So, analysis needs to be done at multiple scale.

f) Harris corner detection doesn't find eigen values λ_1, λ_2 of correlation matrix C

It computes cornerness measure.

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

Here, $\det(C) = \lambda_1 \cdot \lambda_2$

$tr(C) = \lambda_1 + \lambda_2$

$$C(C) = (\lambda_1 \cdot \lambda_2) - k(\lambda_1 + \lambda_2)^2 = \underbrace{(1-2k)\lambda_1\cdot\lambda_2}_{\text{corner}} - \underbrace{k(\lambda_1^2+\lambda_2^2)}_{\text{edge}}$$

Insted of finding λ_1, λ_2, It just find $\det(C)$ and $tr(C)$ of 2x2 correlation matrix which is fast.

$k \in [0, 0.5]$ is a parameter.

If $k = 0$ $C(C)$ detects corner

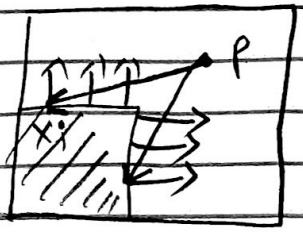
If $k = 0.5$ $C(C)$ detects edge.

It helps to detect both corner and edge controlled by parameter k.

g) To determine P is the corner.

we connect each point x_i in a window to P and project the gradient at x_i onto $(x_i - P)$

The best P will minimize all projections.



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

$$= \sum_i \underbrace{(x_i - p)^T}_{1\times 2} \underbrace{\left(\nabla I(x_i) \nabla I(x_i)^T \right)}_{2\times 2} \underbrace{(x_i - p)}_{2\times 1}$$

To find best p.

$p^* = \underset{p}{\operatorname{argmin}} E(p)$, take derivative with respect to p to find min value of p

$\nabla E(p) = 0$

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

$$\underset{2\times 1}{} = \underbrace{C^{-1}}_{2\times 2} \underbrace{\left(\sum_i \nabla I(x_i) \nabla I(x_i)^T x_i \right)}_{2\times 1}$$

p^* is a point of corner

C = correlation matrix

Condition to exiest

we took windows where $\lambda_1 \cdot \lambda_2 > 2$ for corner λ_1, λ_2 both are non-zero and positive
so, C must be non-singular

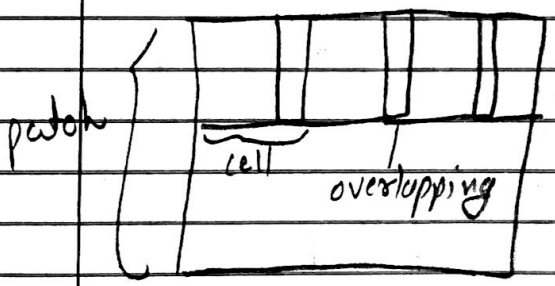
h) In HOG,

- split each patch into possibly overlapping cells
- create orientation histogram for each cell in a patch

for orientation histogram we can use edge or gradient directions.

to find more characteristic feature points we can possibly weighted this histogram by distance from center or gradient magnitude

concate orientation histograms of each cell in a patch



now match this orientation histogram of a single patch in another image's orientation histogram of all patches.

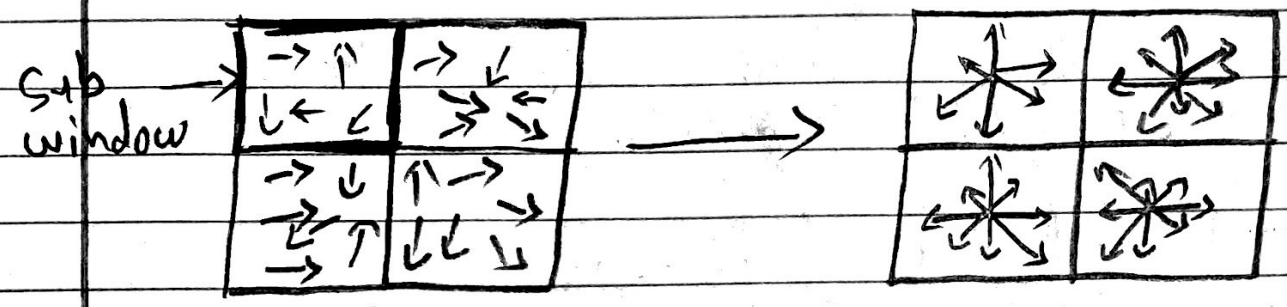
This is how feature point characterization finds one feature from one image to another image. where feature can be translated, rotated, scaled or illuminated.

The requirements from a good characterization of feature points

- translation invariance
- rotation invariance
- scale invariance
- illumination invariance

i) In SIFT

create subwindows in a window
find orientation histogram for each subwindow
concat each orientation histogram of subwindow in a window



align histogram of subwindows based on dominant direction (rotation invariance)
also use weighted sum to create histogram (high weight when near to center of a window and vise a versa)

Ans-2| a) In hough transform

a line eqⁿ : $y = ax + b$

a is a slope and b is y-intercept

The problem here is to define a range for (a) slope parameter
vertical lines have infinite slope ($a = \infty$)

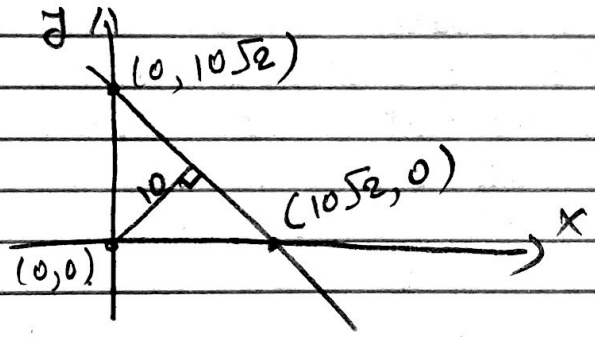
so, representing vertical lines is a problem in hough transform

b) a = slope = 45°
d = 10

a = ? , b = ? , c = ? in $ax + by + c = 0$

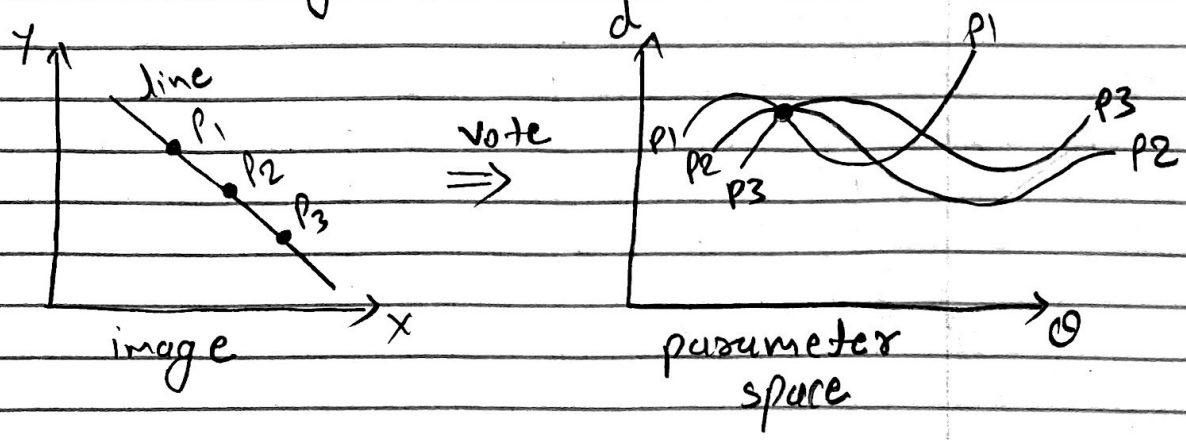
$a = \cos 45 = 1/\sqrt{2} = 0.53$
$b = \sin 45 = 1/\sqrt{2} = 0.53$
$c = -d = -10$

line : $\frac{1}{\sqrt{2}} x + \frac{1}{\sqrt{2}} y - 10 = 0$



c) polar representation of lines

$ax + by + c = 0$
$x \cos\theta + y \sin\theta = d$



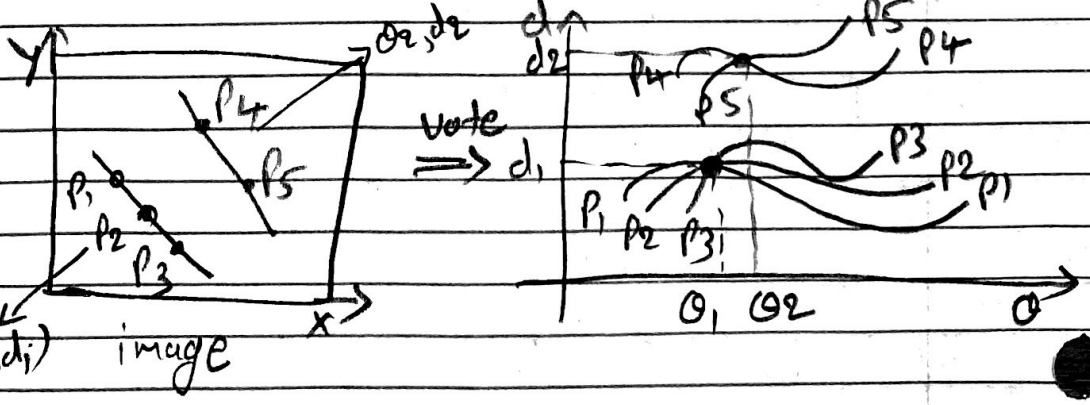
for each point (x,y) :
take θ from 0 to 360 :
get distance d at each θ

Here we will get curve in parameter space for each point in an image

d) If in image there are n points on a line. then in parameter space we get n curves intersecting at point (θ,d) which are the parameters of that line.

→ To create parameter space, for each point (x,y) with each θ_i normal vote for $d_i = x\cos\theta_i + y\sin\theta_i$

when we look at the parameter space, we can see that some curves in parameter space for some points in image are intersecting. So points associated with curves that are intersecting are on the same line where intersecting point (θ,d) are line's parameter



e) In parameter space

- If bins size is large then hough transform will be more efficient in terms of time and space but it provides less localization.

large bins means less number of samples

- If bins size is small, it means large number of samples, and then hough transform will be less efficient in terms of time and space to process these samples but it provides more and accurate localization.

f) Explicit line equation with parameters (θ, d)

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

$$(n_x, n_y) = (\cos\theta, \sin\theta)$$

If we know the normal then for each point we just need to find parameter d. So, for each point parameter θ is fixed.

This is time and space efficient. So, now we don't need to find all d for each θ for each point.

To find points on a line we just need to

compare d of points having same θ.

In parameter space we won't get curve for each point. for each point we get a point in parameter space.

g) Circle equation,

$$x' = x_c + r\cos\theta \qquad \text{or} \qquad (x'-x_c)^2 + (y'-y_c)^2 = r^2$$
$$y' = y_c + r\cos\theta$$

(x', y') Point on a circle
(x_c, y_c) center of a circle
r is radius of a circle.
θ is an angle, $\theta \in [0, 360]$

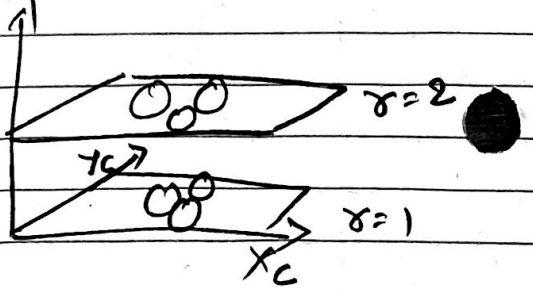
circle's parameters: x_c, y_c, r

Three parameters for circle.

So, the dimensions of the parameter space will be three in Hough transform

Given (x', y') vote in each r plane using

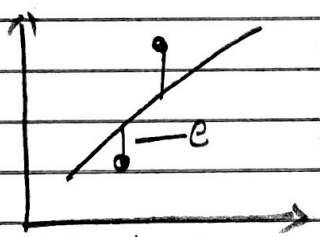
$x_c = x' - r\cos\theta$, $\theta \in [0, 360]$
$y_c = y' - r\cos\theta$



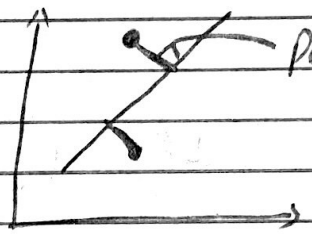
Ans-3) a) The objective function for line $y = ax + b$ fitting is

$$E(a,b) = \sum_i \underbrace{(d_i - (ax_i + b))}_{e_i}^2$$

But this objective is not good because it uses algebraic distance.



Algebraic distance



geomatric distance

In Algebraic, as slope increase verticle distance is not reliable because of the infinite slop of verticle lines cannot fit.
So here, geomatric distance should be used.

b) $(n_x, n_y) = (1, 2) = (a, b)$

$d = 2$

$c = -d = -2$

$$l = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

c) $l^T P$ is the objective function of point $P = (x, y, 1)$ from line $l = (a, b, c)$

Here algebric error can be neglizibe of being geomatric.

Explicit line Equation : $l^T P = 0$

objective function : $\mathcal{E}(l) = \sum_i (l^T P_i)^2$

$$\mathcal{E}(l) = \sum_i l^T P_i P_i^T l = l^T \left(\sum_i P_i P_i^T\right) l = l^T S l$$

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

$$\nabla \mathcal{E}(l) = 0$$
$$Sl = 0$$

l is eigenvector of S belonging to 0 eigenvalue.

$$S = \sum P_i P_i^T = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & n \end{bmatrix}$$

$S = D^T D$ where $D = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & & \\ x_n & y_n & 1 \end{bmatrix}$

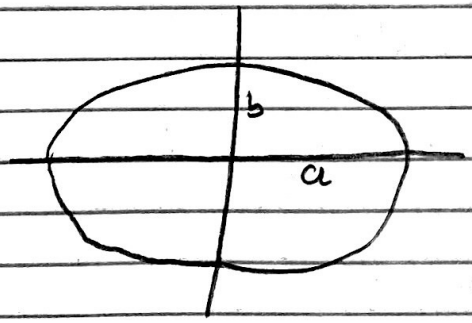
d) Points : $\{(0,1), (1,3), (2,6)\}$

$$S = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i y_i & \Sigma x_i \\ \Sigma x_i y_i & \Sigma y_i^2 & \Sigma y_i \\ \Sigma x_i & \Sigma y_i & n \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 15 & 3 \\ 15 & 46 & 10 \\ 3 & 10 & 3 \end{bmatrix}$$

e) for conic curves, Explicit Equation

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



Implicit Equation :

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

If $b^2 - 4ac < 0$, we can fit the model on elipse

f) Implicit equation:

$$a^T p = 0$$

$$a^T = (a, b, c, d, e, f)$$
$$p = (x^2, xy, y^2, x, y, 1)$$

$$p_i = (x_i, y_i)$$

Ellipse fitting:
minimize objective function

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

$$S = \sum_{i} p_i p_i^T$$

$$a_* = \underset{a}{\operatorname{argmin}}\, E(a) \quad \text{subject to } b^2 - 4ac < 0$$

This objective function use algebraic distance

$$b^2 - 4ac < 0 \iff a^T S a = -1$$

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

Modified $\mathcal{E}(\lambda)$ objective function:

$$\mathcal{E}(\lambda) = \lambda^T S \lambda + \lambda(\lambda^T C \lambda + 1)$$

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

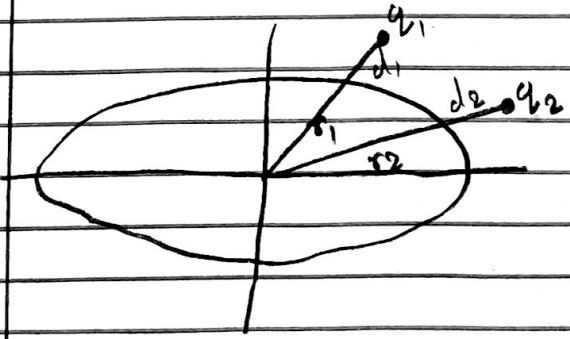
$$\nabla \mathcal{E}(\lambda) = 0$$
$$\lambda = \lambda_i S^{-1} C \lambda$$

λ^* is eigenvector of $S^{-1}C$ belong to negative eigenvalue

when x_i is on ellipse: $\lambda^T x_i = 0$
when x_i is off ellipse: $q_i = \lambda^T x_i$

q_i = a distance from the ellipse

algebraic distance $= q_i = \lambda^T x_i \approx \dfrac{d_i}{d_i + r_i}$



$d_1 \approx d_2$
$q_1 > q_2$, $r_2 > r_1$

points close to the short axis of ellipse affect more to the fitting.

g) geomatric distance for ellipse.

objective function,

$$\mathcal{E}(P) = \sum_i \frac{|f(P_i, \lambda)|}{|\nabla f(P_i, \lambda)|} =$$

$$f(P_i, \lambda) = \lambda^T P_i$$

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

there is no explicit solution to $\nabla \mathcal{E}(\lambda) = 0$

only iterative numerical solution is there.

h)

$$\mathcal{E}(\phi(s)) = \int_{\phi(s)} \underbrace{\alpha(s)\mathcal{E}_{cont} + \beta(s)\mathcal{E}_{curve}}_{\text{internal Energy}} + \underbrace{\gamma(s)\mathcal{E}_{img}}_{\text{external Energy}} ds$$

$$\mathcal{E}_{cont} = \left|\frac{d\phi}{ds}\right|^2 = |P_{i+1} - P_i|^2 \quad \text{continuity energy}$$

$$\mathcal{E}_{curv} = \left|\frac{d^2\phi}{ds^2}\right|^2 = |P_{i+1} - 2P_i + P_{i-1}|^2 \quad \text{curvature energy}$$

$$\mathcal{E}_{img} = -|\nabla I|^2 = \text{image energy.}$$

for discrete, $\phi(s) \longrightarrow \{P_i\}_{i=1}^{n}$

- E_{cont} is for the curve to be continuous. should be low. because we want equal interval
- E_{curve} for curve to be smooth. and it should be low
- E_{img} is for gradient to be on edge.

$\alpha(s)$, $\beta(s)$, $\gamma(s)$ are user selected parameters

$\{P_i\}$ unknown model parameters

d is a computed parameter.

$$E(\{P_i\}) = \sum_{i=1}^{n} \alpha_i (|P_{i+1} - P_i| - d)^2 + \sum_{i=1}^{n} \beta_i |P_{i+1} - 2P_i + P_{i-1}|^2 - \sum_{i=1}^{n} \gamma_i |\nabla I(P_i)|^2$$

d is average distance b/w points to prevent shrinking.

i) continuity of discreate curve.

$$E_{cont} = \left| \frac{d\phi}{ds} \right|^2 \approx |P_{i+1} - P_i)^2$$

: first order derivative

continuity is the difference with next neighbor on curve.

Curvature of discrete curve :

$$E_{curve} = \left| \frac{d^2\phi}{ds^2} \right|^2 = \left| P_{i+1} - 2P_i + P_{i-1} \right|^2$$

second order derivative

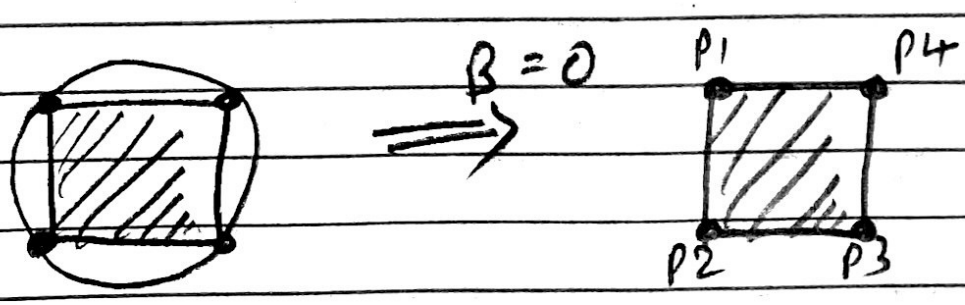
difference of two consecutive first order derivative.

i)

continuity of active contours may be relaxed by taking $\beta_i = 0$ at corners to allow discontinuity.

to allow tight fitting for curves, set $\beta_i = 0$ to point with high curvature. when,

$$|P_{i+1} - 2P_i + P_{i+1}| > \tau$$



Ans-4] a) gradient vectors (1,2), (3,4)

$$C = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix}$$

$$C = \begin{bmatrix} 10 & 14 \\ 14 & 20 \end{bmatrix}$$

b) points (10, 10), (20, 20)

implicit line equation : ?

$y = ax + b$

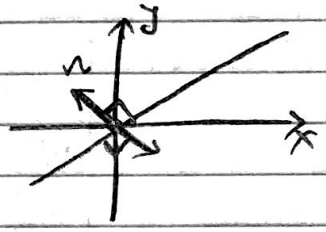
$a = \frac{y_2 - y_1}{x_2 - x_1} = \frac{20-10}{20-10} = 1$

$y = x + b$

take point (10, 10)

$10 = 10 + b$

$b = 0$



Explicit line eq^n : $y = x$

normal : $(-1, 1) = \vec{n}$

normalized normal : $\hat{n} = \frac{\vec{n}}{|\vec{n}|} = (-1/\sqrt{2}, 1/\sqrt{2})$

implicit line eq^n :

$(n_x, n_y) \cdot (x, y) = d$ or $n_x X + n_y Y - d = 0$

$(n_x, n_y) = (-1/\sqrt{2}, 1/\sqrt{2})$ or $(1/\sqrt{2}, -1/\sqrt{2})$

$d = 0$, line passes through origin

Eq^n : $\frac{-1}{\sqrt{2}} X + \frac{1}{\sqrt{2}} Y - 0 = 0 \Rightarrow \frac{1}{\sqrt{2}} x = \frac{1}{\sqrt{2}} Y$

$\Rightarrow X = Y$

c) $(a, b, c) = (1, 2, 3)$

$(x, ?) = (2, ?)$

line : $ax + by + c = 0$

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

take $x = 2$,

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

$y = -5/2$

d) Points $(1,1)$, $\theta = 0$

vote : $d = n_x x + n_y y = x\cos\theta + y\sin\theta$

$= \cos\theta + \sin\theta$

$= 1 + 0$

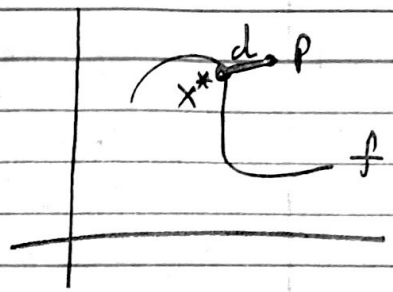
$d = 1$

e) Points $(1,2), (3,4)$, $n = 2$

$$S = \sum_i p_i p_i^T = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & n \end{bmatrix} = D^T D$$

$$D = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 4 & 1 \end{bmatrix} \Rightarrow S = \begin{bmatrix} 10 & 14 & 4 \\ 14 & 20 & 6 \\ 4 & 6 & 2 \end{bmatrix}$$

f) implicit curve f
a point P



$|f(P)| = 1$
$|Df(x^*)| = 2$
$d(f, P) = ?$
(geometric)

$$d(f, P) = |P - x^*| = \frac{|f(P)|}{|Df(x^*)|} = \frac{1}{2}$$

g) implicit curve f
a point P

$|f(P)| = 1$
$|Df(P)| = 2$
$d(f, P) = ?$
(algebraic)

$$d(f, P) = \frac{|f(P)|}{|Df(P)|} = \frac{1}{2}$$

h) $P_1 = (1, 2)$, $P_2 = (2, 3)$, $P_3 = (3, 4)$

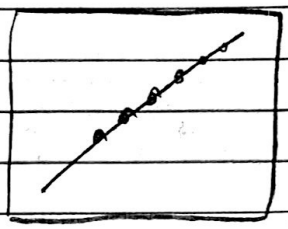
$$E_{cont}(P_2) = |P_3 - P_2|^2 = |(1, 1)|^2 = \sqrt{2}^2 = 2$$

$$E_{curv}(P_2) = |P_3 - 2P_2 + P_1|^2 = |(0, 0)|^2 = 0$$

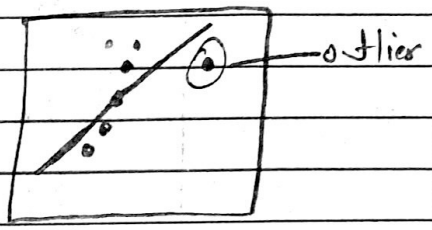
i) Given a point with high curvature (e.g. corner) the coefficient β_i has to be set to zero for that location to guarantee tight fitting of an active contour.

Ans-5| a) outliers are the data points which are different from the other points in the dataset

- These outliers can be incorrect or be noise in dataset. It affects model fitting



correct.



incorrect

- If dataset has no outliers then model fitting will be correct.
but if one outlier is added then model fitting solution will be destroied
- Model fitting is very sensitive to outliers.
- Main problem is when we minimize objective error function which is calculated using sum of square of distance, distance for outliers might be large and due to these outliers solution moves towards the outliers and It fits a wrong model.

b) Robust estimation use M-estimators

MSE : Mean Square Error

$$E(\theta) = \sum_i d^2(x_i, \theta) \quad , \quad d^2 = (\perp^T x_i)^2 \text{ for line fitting}$$

$E(\theta) = \sum \rho_\sigma(d(x_i, \theta))$; objective function

Here, we use standard deviation as a parameter.

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

In standard least square error we use $\rho_\sigma(x) = x^2$ which is MSE
but in M-estimator we use ρ_σ function instead of square function.

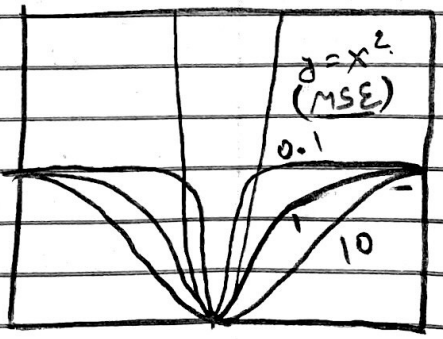
c) Geman-McClure estimator

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

$\rho_\sigma(x) = \frac{x^2}{x^2 + \sigma^2}$, where σ is a bandwidth parameter.

$x >> \sigma$, $\rho_\sigma(x) = 1$

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



as σ gets large, valley becomes wider and it also includes outliers and vice a versa.

advantage is that is distance becomes larger than σ. Model consider it as outlier and stop giving more weights to its error and takes maximum 1.

but in MSE it gives more weight as distance (error) increase.

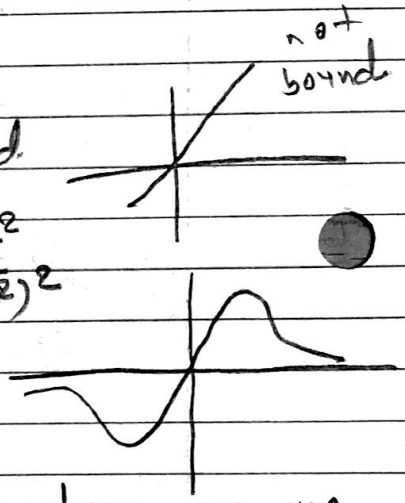
large $\sigma \Rightarrow$ include more points as inlier.
Small $\sigma \Rightarrow$ include less points as inlier.

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

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

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

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



In geman-McClure estimator we are putting boundaries or limit over the effect of errors

variable estimation:
start with large σ and decrease as converging

adjust σ in each iteration.

$$\sigma^{(n)} = 1.5 \text{ median}\left\{ d(x_i, \theta_i^{(n-1)}) \right\}$$

$\sigma^{(n)}$: estimate at step n

$\theta_i^{(n-1)}$: parameters of $(n-1)^{th}$ step

d) $x = 1, \quad \sigma = 1$

$$\rho_\sigma(x) = \frac{x^2}{x^2+\sigma^2} \Rightarrow \rho_{\sigma=1}(x=1) = \frac{1^2}{1^2+1^2} = \frac{1}{2}$$

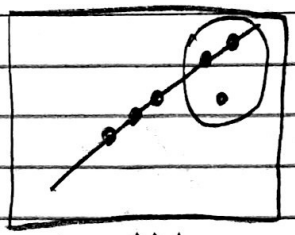
e) - RANSAC performs multiple experiments and choose best results

Parameter :
n = # points drawn at each evaluation
d = min # points needed to estimate model
K = # trials
t = distance threshold to identify inliers

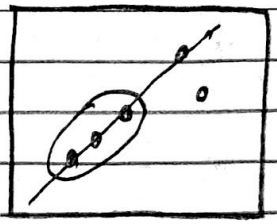
Algorithm :
- Repeat K times :
  - Draw n points uniformly at random (with replacement)
  - fit a model to points
  - find inliers in entire set [distance $< t$]
  - Recompute model (if atleast d inliers)
  - update parameters (K, t)

options to choose best solution :
- Largest consensus set ← preferable
- Smallest error

The number of points drawn at each attempt should be small in hope that atleast one set will not have outliers



set with outlier



set without outlier.

f) Parameters:

n = # points drawn at each evaluation
d = min # points needed to estimate model
K = # trials
t = distance threshold to identify inliers

estimate # trials (K):

P: with probability of P at least one experiment does not have outliers
w: probability that a point is an inlier. (initialy $w=0.5$)

Probability that all K experiments failed

$$(1-P) = (1-w^n)^K$$
$$\log(1-P) = K \log(1-w^n)$$

$$K = \frac{\log(1-P)}{\log(1-w^n)}$$, large P → large K
small w → large K

$w = \frac{\#\text{inliers}}{\#\text{points}}$ update w, K every iteration but set upper bound for K.

g) $p = 0.99$, $K = ?$
$w = 0.9$

$$K = \frac{\log(1-p)}{\log(1-w^n)} \quad , \quad n = ?$$

assume $n = 5$,

$$K = \frac{\log(1-0.99)}{\log(1-0.9^5)} = \frac{\log(0.01)}{\log(1-0.59)} = \frac{\log(0.01)}{\log(0.41)}$$

$$= \frac{-2}{-0.387} = 5.168 \approx 5$$

Ans - 6)

a) - separate objects from background (foreground)
- find contours of objects
- semantic image segmentation: label each pixel in the image with class label.

b) Agglomerative approach:
- start with each pixel in a separate cluster.
- Merge clusters with small distance.

Divisive approach:
- start with each pixel in one cluster
- split clusters to produce large distance between them.

c) K-Mean Algorithm :

- Select K random points in space
- select initial guess of means.
  $m_1, m_2, \ldots, m_K$
- Repeat while m_j change :

$$l_i = \underset{j \in [1,K]}{\operatorname{argmin}} \| f_i - m_j \|^2 \quad \text{for each pixel}$$

assign cluster

$$S_j = \{ i \mid l_i = j \}$$

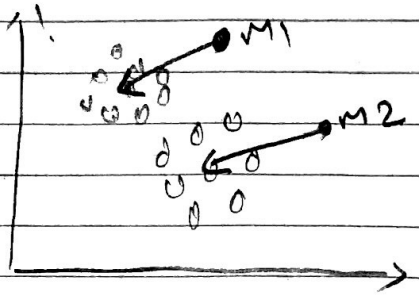
$$m_j = \frac{\sum_{i \in S_j} f_i}{\# S_j}$$

recompute means

$\Rightarrow$ Simultaneous Solution of two problem

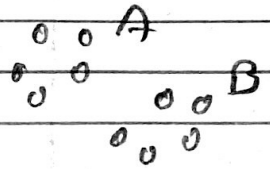
- find cluster centers
- assign points to clusters

Here one parameter fixed and chang other to minimize error.



$\Rightarrow$ coordinate descent on $E(\{l_i\}, \{m_j\}) = \sum \| f_i - m_{l_i} \|^2$

d) Graph Cut:
- possible graph cuts remove links to create disconnected sub-graphs
- The cost of a cut is the sum of links being removed



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

Minimum cut: cut with lowest cost

Normalized cut:
- A normalized cut assigns cost taking into account the size of produced clusters

$$Ncut(A,B) = \frac{Cut(A,B)}{Vol(A)} + \frac{Cut(A,B)}{Vol(B)}$$

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

⇒ Normalized cut is necessary because during minimizing the cost of a cut will yield cuts of single nodes

e) Similarity matrix

$$W = \begin{bmatrix} w_{1,1} & \cdots & w_{1,100} \\ \vdots & & \vdots \\ w_{100,1} & \cdots & w_{100,100} \end{bmatrix}$$

The weighted degree matrix,

$$d_i = \sum_{j=1}^{100} w_{i,j} \quad \longleftarrow \text{Similarity of node } i \text{ to } j$$

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

weighted degree of each node
diagonal matrix

The laplacian matrix:

$$L = D - W \quad \Rightarrow \text{Symmetric and Sum of every row and column in } L \text{ is } 0$$

D: weighted degree; W: adjacency similarity

$$L = \begin{bmatrix} d_1 - w_{1,1}, \ldots, -w_{1,100} \\ \vdots \\ -w_{100,1} \ldots d_{100} - w_{100,100} \end{bmatrix}$$

f) Defining a Cut.

$$x = \{1, -1\}^{\sim} \qquad x(i) = 1 \iff i \in A$$
$$x(i) = -1 \iff i \in B$$

$$Ncut(A,B) = \frac{Cut(A,B)}{Vol(A)} + \frac{Cut(A,B)}{Vol(B)}$$

$$= \frac{\sum_{\substack{x^p>0 \\ x_j<0}} (-\omega_{ij}\, x_i x_j)}{\sum_{x^i>0} d^i} + \frac{\sum_{\substack{x^i>0 \\ x_j<0}} (-\omega_{ij}\, x_i x_j)}{\sum_{x_j^i<0} d_j}$$

g) $$k = \frac{\sum_{x^p>0} d^i}{\sum_i d^i} \qquad b = \frac{k}{1-k} = \frac{\sum_{x^i>0} d^i}{\sum_{x_j<0} d_j}$$

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

convert the problem

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

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

$1 = [1, \ldots, 1]$

h) when y is discrete, it's hard to solve for normalized cuts

So, move to a continuous domain

Solve: $\min_y \frac{y^T(D-\omega)y}{y^T D y}$, s.t. $y^T D 1 = 0$

Rayleigh quotient

$\Rightarrow (D-W)\vec{y} = \lambda D\vec{y}$

The second smallest eigenvalue and a coorsponding eigenvector is the solution.

Because, smallest eigenvalue is zero and coorsponding eigenvector is $[1, 1, \ldots, 1]$

i) Eigen faces approach:

- map image (e.g. 100x100 dimensions) to lower dimensional vectors (e.g. 64) using PCA.
- measure similarity to templates in lower dimensional space

The limitation of this approach is that it's less sensitive to small variations.

j) Beg-of-words:
- extract features (e.g. SIFT or HOG)
- cluster features to create a codebook (dictionary)
- Compute a distribution of code words in each class.
- classify using distribution of code words

The limitation is that classifier looks at diff. local neighborhood to make classification but doesn't look at the relationship between them.