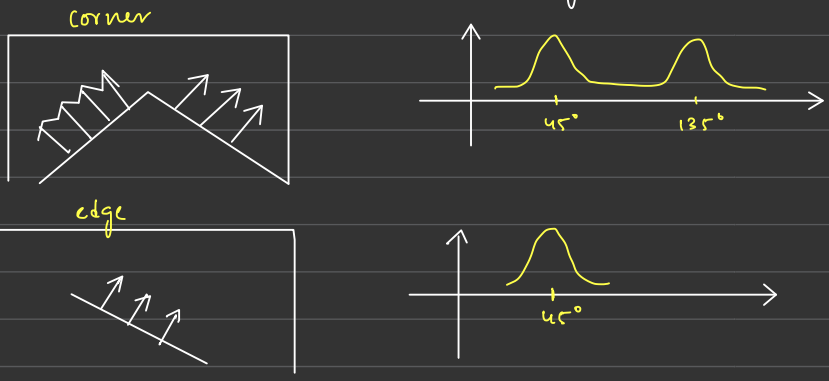
Assignment 2

A20442409

Review questions:

(1) Corner detection:

(a) The basic principle of corner detection is to find a window such that there are two directions for gradients as shown in the below image.



* corner has more than one direction in orientation histogram.
* principle direction can be assessed based on the number of peaks in the orientation histogram.

(b) First find the direction v such that projection of $\{g_i\}$ onto v is minimised.

$$E(v) = \sum_i (g_i \cdot v)^2 = \sum_i (g_i^T v)(g_i^T v)$$

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

additional directions minimize projection such that being orthogonal to previous directions.

(c) $C = \sum_i g_i g_i^T$

$$* \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad * \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$* \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad * \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$* \begin{bmatrix} 0 \\ 2 \end{bmatrix} \begin{bmatrix} 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix} \qquad * \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

$$* \begin{bmatrix} 0 \\ 3 \end{bmatrix} \begin{bmatrix} 0 & 3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 9 \end{bmatrix} \qquad * \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 3 & 9 \end{bmatrix}$$

$$* \begin{bmatrix} 0 \\ 4 \end{bmatrix} \begin{bmatrix} 0 & 4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 16 \end{bmatrix}$$

$$C = \begin{bmatrix} 0+4 & 0+1+2+3 \\ 0+1+2+3 & 1+4+9+16+1+4+9 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 6 & 44 \end{bmatrix}$$

(d) $E(v) = v^T C v \qquad C \equiv \sum_i g_i g_i^T$

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

$\Rightarrow \quad \nabla E(v) = 0$

$2Cv = 0 \Rightarrow$ Solution is eigenvector belonging to smallest eigenvalue

(C: 2×2)

eigen value = variance in corresponding principal direction.

(e)

1. compute $\lambda_1 \cdot \lambda_2$ for all windows
2. select windows with $\lambda_1 \cdot \lambda_2 > \tau$ and sort it in decending order.
3. Select the top of the list as corner and delete all other corners in its neighbourhood from the list
4. Stop once deleting x % of the points as corners.

* overlepping windows are needed to account for corners between windows.
* the analysis need to be done at multiple scales.
  ↳ (pyramids).

(f) The process followed to perform Harris corner detection are:

1. Compute correlation matrix C for windows
2. Compute cornerness measure

hyperparameter

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

$\det(c) = \lambda_1 \cdot \lambda_2$; $tr^2(c) = (\lambda_1 + \lambda_2)^2$; $tr \Rightarrow$ trace.

3. Detect corners where $C(c)$ is high

$k \in [0, 0.5]$ hyperparameter eg: $0.04 - 0.15$

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

if $k = 0.5$ $C(c)$ detects edges.

$$\begin{aligned} C(c) &= \det(c) - k \, tr^2(c) \\ &= \lambda_1 \lambda_2 - k (\lambda_1 + \lambda_2)^2 \\ &= (1 - 2k) \lambda_1 \lambda_2 - k (\lambda_1^2 + \lambda_2^2) \end{aligned}$$

$(1 - 2k) \lambda_1 \lambda_2$: $= 0$ if $k = 0.5$ → corner detection

$k (\lambda_1^2 + \lambda_2^2)$: $= 0$ if $k = 0$ → edge detection.

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

p^* → location of corner

C^{-1} → correlation matrix inverse

Better localization is achieved only when we evaluate the below condition

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

$$\rightarrow \nabla E(p) = 0.$$

(h)

1. Split each path into cells (possibly overlapping)
2. Create orientation histogram in each cell (using edge or gradient directions, possibly weighted by distance from centre & gradient magnitude
3. Concatinate orientation histograms.

Requirements for good characterization of feature points are:

1. Translation invariance → local window
2. Rotation invariance → histograms
3. Scale invariance → pyramids
4. Illumination invariance → gradients

(i)

* Use weighted sum to create orientation histograms in cells, then concatinate
* Align histogram based on dominant direction (rotation invariance)

② Line detection:

(a) There are two main problems when we talk about using slope and y intercept as the parameters while performing Hough transformation.
First, we do not know or practically compute a definitive number for the range of slope as the slope can vary from $-\infty$ to ∞
Second, we can not represent a line with slope $= \infty$ on the parameter space.

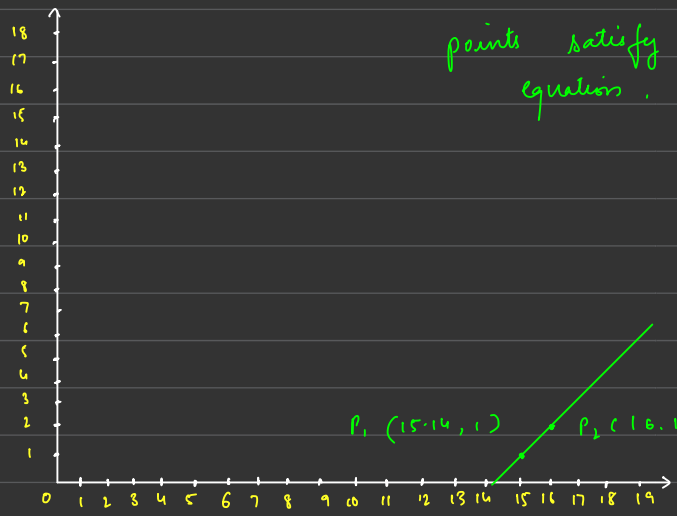
(b) we have $(\cos\theta)\, x + (\sin\theta)\, y = d$

substitute $\theta = -45°$ and $d = 10$.

$$\frac{1}{\sqrt{2}}\, x - \frac{1}{\sqrt{2}}\, y = 10$$

$$x - y = 10\sqrt{2}$$

$$\Rightarrow \quad y - x + 10\sqrt{2} = 0$$



(c) While using the polar coordinate system each vote represents a bin of space in the parameter space. For a line we get a 2D bin.

(d) Plotting different curves on the parameter space pertaining to a respective points yeilds in intersection of curves at one particular point. The x and y coordinates of this intersection (can be θ and d in case of polar coordinate system) gives the actual parameters of the line detected.

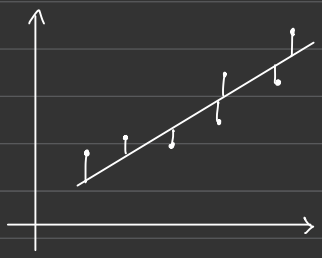
(e) Larger bins are more efficient but provide less localization. Larger bins are more efficient because we calculate less number of points and hence we get a less accurate but fast result.

(f) The fundamental problem of using slope and y intercept as the parameters in the parameter space is that we cannot find a definitive representation of the range of slope and to define vertical lines.
Knowing the normal at each voting point allows us to represent the parameters using polar coordinates thus voting in the parameter space is improved. Range for $\theta \in [0, 180^\circ]$ and $d \in [-\sqrt{n^2+m^2}, \sqrt{n^2+m^2}]$.

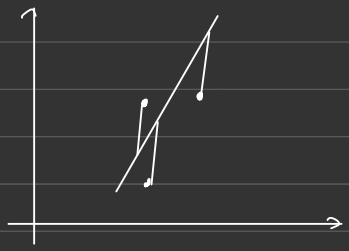
(g) While using Hough transformation for circles, number of dimensions of the parameter space = 3.

(3) Model fitting 1

(a)



works just fine in case of m not close to 90°



If m is close to 90° measuring the distance from the line to the point serves better.

Lines with slope close to 90° cannot be fit accurately.

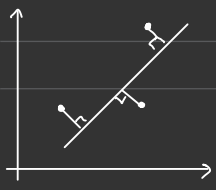
(b) Given normal $n\,(1, 2) \Rightarrow n_x = \frac{1}{\sqrt{5}} \quad n_y = \frac{2}{\sqrt{5}}$

line : $n_x x + n_y y - d = 0$

$$l^T = \begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \\ -2 \end{bmatrix}$$

(c) * We use line equations in homogeneous coordinates.

* point p is on the line l if

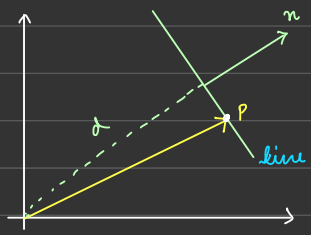


$l^T p = 0$

$(a,b,c)^T$ $(x,y,1)$

we have the line eqn $ax + by + c = 0$

(a,b): normal vector; c: negative distance from origin



for p to be on the line

$p \cdot n = d$

$n_x x + n_y y - d = 0$

$ax + by + c = 0$

If $n = (a,b)$ is not normalized c is only proportional to the distance.

(d) $S = \sum p_i p_i^T$ $(0,1)$ $(1,3)$ $(2,6)$

3×1 1×3

$$= \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & m \end{bmatrix}$$

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

(e) $(x^2, xy, y^2, x, y, 1)$

$l^T \rho = 0$

(a, b, c, d, e, f) model parameters.

The constraint is $b^2 - 4ac < 0$

(f) * when x_i is on the ellipse : $l^T x_i = 0$

* when x_i is off the ellipse : $l^T x_i = q_i$

provides a measure for distance from the ellipse

algebraic distance $q_i = l^T x_i \simeq \frac{d_i}{d_i + r_i}$

here $d_1 \simeq d_2$ but $q_1 > q_2$ because $(r_2 > r_1)$

Points close to the short axis affect the fitting more.

(g) This is the required equation.

$d(p, f) = |p - x^*| = \frac{|f(p)|}{|\nabla f(x^*)|}$ ← algebraic distance.

↑ geometric distance

The additional complexities involved are:

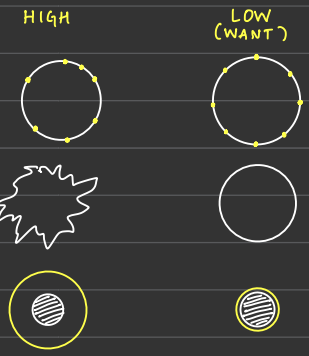
① computing $\nabla f(x^*)$ is expensive.

② Even after approximating $\nabla f(x^*)$ with $\nabla f(p)$ the obtained solution is not explicit due to which we might need to use iterative numerical solution.

(h)

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

$\alpha(s)$, $\beta(s)$, $\gamma(s)$ are coefficients of the different energy terms.



Continuity energy : $E_{cont} = \left[\frac{d\phi}{ds} \right]^2$

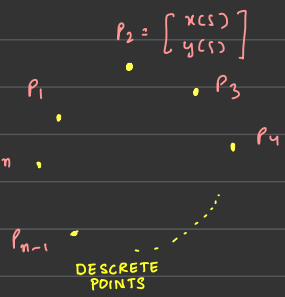
Curvature energy : $E_{curv} = \left[\frac{d^2\phi}{ds^2} \right]^2$

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

↑ Maximizing the sum of gradients

(i)

* Descrete case : $\phi(s) \longrightarrow \{p_i\}_{i=1}^{n}$ curve that traces the object



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

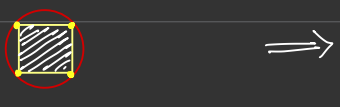
velocity

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

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

acceleration.

(j) To allow for piecewise curves set $\beta_i = 0$ to points with high curvature, i.e., when $|p_{i+1} - 2p_i + p_{i-1}| > \tau$



4. Model fitting 2

(a)

$$C = \sum_{2\times 1} g_i \underset{1\times 2}{g_i^T} = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix}$$

$g_1 (1,2)$ $g_2 (3,4)$

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

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

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

slope $m = \dfrac{20-10}{20-10} = 1$

$\therefore$ The line equation should be

$y = x + c$

since points $(10,10)$ and $(20,20)$ passes through the line

$\Rightarrow c = 0$

$\therefore$ line eqn : $y = x$.

implicit line eqn is $x - y + 0 = 0$.

$n_x x + n_y y - d = 0$

$n_x = 1 \quad n_y = 1$

normalised normals : $(n_x, n_y) = (1, -1)$

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

$x + 2y + 3 = 0$ is the line

for $x = 2$ $\quad 2 + 2y + 3 = 0 \Rightarrow y = -5/2 = -2.5$

(d) $d = x \cos\theta + y \sin\theta$. for $(x, y) = (1, 1)$ $\theta = 0°$

$d = \cos 0° + \sin 0° = 1$

Hence a vote will be casted in the parameter space at $d = 1$.

(e) $S = \sum_i p_i p_i^T$ = correlation matrix of points

(p_i: 3×1, p_i^T: 1×3)

$$= \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & m \end{bmatrix}$$

$p_1 = (1, 2)$ $\quad p_2 = (3, 4)$

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

(f) $d(p, f) = |p - x^*| = \dfrac{|f(p)|}{|\nabla f(x^*)|} =$ (geometric distance)

given $f(p) = 1$ and $\nabla f(x^*) = 2$

$$d(p, f) = \frac{1}{2} = 0.5.$$

(g) $d(p, f) = |p - x^*| = \dfrac{|f(p)|}{|\nabla f(x^*)|} = \dfrac{|f(p)|}{|\nabla f(p)|}$ ← algebraic distance. (geometric distance)

given $f(p) = 1$ and $\nabla f(p) = 2$

$$d(p, f) = \frac{1}{2} = 0.5.$$

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

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

velocity

Substitute $i = 2$.

$E_{cont} = (P_3 - P_2)^2 = |(3-2)^2 + (4-3)^2| = 2$

* $E_{curv} = \left[\frac{d^2\phi}{ds^2}\right]^2 = |(P_{i+1} - P_i) - (P_i - P_{i-1})|^2$

acceleration.

$= |(P_3 - P_2) - (P_2 - P_1)|^2$

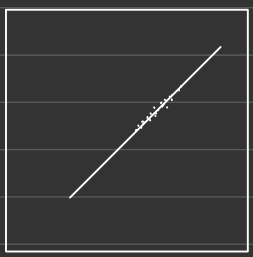
$= \left|\sqrt{(3-2)^2 + (4-3)^2} - \sqrt{(2-1)^2 + (3-2)^2}\right|^2$

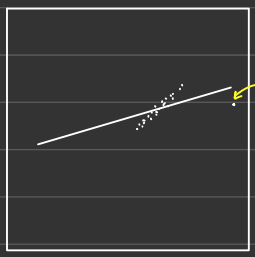
$= 0$

(i) $\beta = 0$ for a point with high curvature.

5. Robust Estimation.

(a)





An outlier is a data point that deviates the model to fit accurately. Outliers influence the way the model is fit and hence pose a problem in model fitting.

(b) Rhobust estimation :

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

ρ_σ performs better than the standard d^2 objective fn.

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

Mean square error fitting (MSE)

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

(c) Geman - McClure fn :

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

→ large σ → include more points
→ small σ → include fewer points
→ variable estimation :
Start with large σ, decrease as converging.

(d) $$\rho_\sigma(1) = \frac{1^2}{1^2 + 1^2} = \frac{1}{2} = 0.5$$

(e) ① Draw a large set of points uniformly at random
② Select initial value at σ
③ Fit model → $\theta^{(i)}$
④ Compute $\sigma^{(i)}$ using median distance of points
⑤ Continue while objective is decreasing.

The number of points drawn at each attempt should be

small due to the reason that considering a small subset of points increases the probability of inliers.

(f) Parameters:

n = number of points drawn at each evaluation.
d = min # points needed to estimate model
k = # trails
t = distance threshold to identify inliers.

$$k = \frac{\log(1-p)}{\log(1-w^n)} \qquad w = \frac{\text{\# inliers}}{\text{\# points}}$$

(g) Let us consider we draw 3 points at a time.

$$k = \frac{\log(1-0.99)}{\log(1-0.9^3)} = \frac{-2}{-0.5170} = 3.527.$$

We need to perform 4 experiments.

6. Segmentation and recognition.

(a) The objective of image segmentation are:

- Separate object(s) from background
- Find contours of the object
- label each pixel in the image with class label.

(b) **Agglomerative segmentation:**

* Start with each pixel in a seperate cluster.
* Merge clusters with small distance.
* Repeat while clusters are not satisfatory.

Divisive segmentation:

* Start with all pixels in one cluster.
* Split clusters to produce large distance between them.
* Repeat while clusters are not satisfatory.

(c)

K-means algorithm:

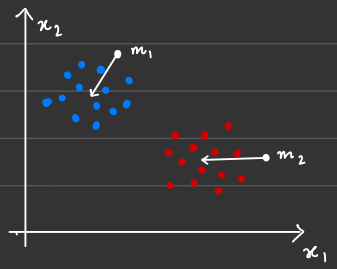
* Select K
* Select initial guess of means: $m_1, m_2, \dots m_K$
* Repeat while m_j change:

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

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

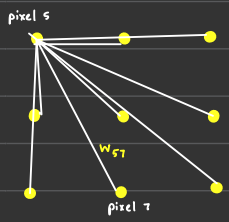
$$m_j = \frac{\sum_{i \in S_j} f_i}{\# S_j} \qquad \text{recompute means}$$

* Simoultanious solution of two problems:
  - → Find cluster centers
  - → Assign points to cluster.
* Hold one parameter fixed and change other to minimize error.
* Coordinate descent on $E(\{l_i\}, \{m_j\}) = \sum \| f_i - m_{e_i} \|^2$



(d)

Graph cuts :

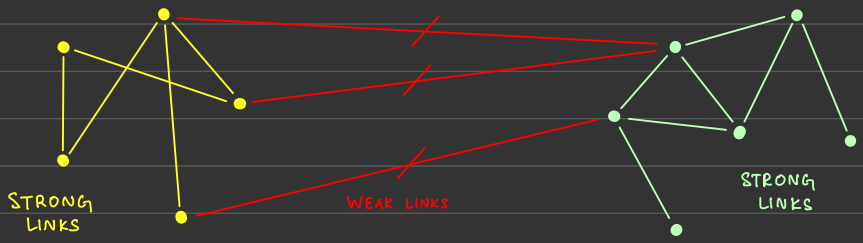


→ Represent image using a graph
→ pixels are nodes
→ Each pixel is connected to all other pixels
→ The weight on the link between nodes p and q represents the similarity between them. (intensity + location)

$$w_{pq} = \exp\left(-\| f(p) - f(q) \|\right)$$

if f(p) is similar to f(q) ⇒ $w_{pq} = 1$
if f(p) is not similar to f(q) ⇒ $w_{pq} = 0$

→ Delete links to create segments.
→ Remove low similarity links.
→ Nodes in each cluster should mark strong links.



→ Possible graph cuts remove links to create disconnected sub graphs
→ The cost of a cut is the sum of links removed.

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

→ Minimum cut is the cut with lowest cost.
→ Minimizing the cost of a cut will yeild cuts of single nodes

→ A normalized cut assigns cost taking into account the size of produced clusters.

$$N_{cut}(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

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

Normalized cuts are more effective compared to graph cut since we emphasize on the size of produced clusters.

(e) Given $n = 100$

Similarity matrix :

$$\begin{bmatrix} w_{11} & \cdots & w_{1100} \\ \vdots & & \vdots \\ w_{1001} & \cdots & w_{100100} \end{bmatrix}$$

Weighted degree matrix :

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

Laplacian matrix : $L = D - W$

$$\begin{bmatrix} d_1 & -w_{12} & \cdots & -w_{199} & -w_{1100} \\ -w_{21} & \ddots & & & -w_{2100} \\ \vdots & & \ddots & & \vdots \\ -w_{991} & & & \ddots & -w_{99100} \\ -w_{1001} & -w_{1002} & \cdots & -w_{10099} & d_{100} \end{bmatrix}$$

(f)

→ The weighted degree d_i of a node is the sum of all edges connected to it:

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

($w_{i,j}$: similarity of node i to j)

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix}$$

← Similarity matrix

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

← weighted degree of each node.

→ Define a cut

$x = \{1, -1\}$

$x(i) = 1 \iff i \in A$ ← CLUSTER (1)

$x(i) = -1 \iff i \in B$ ← CLUSTER (2)

$$N_{cut}(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(A, B)}{vol(B)}$$

$$N_{cut}(A, B) = \frac{\sum_{x_i > 0, x_j < 0} (-w_{ij} x_i x_j)}{\sum_{x_i > 0} d_i} + \frac{\sum_{x_i > 0, x_j < 0} (-w_{ij} x_i x_j)}{\sum_{x_j > 0} d_j}$$

(g) The optimization problem that needs to be solved is as follows:

$$\begin{cases} \min_x N_{cut}(x) \cong \min_y \dfrac{y^T (D - W) y}{y^T D y} \\ \text{such that } y^T D \mathbf{1} = 0 \end{cases}$$

$\mathbf{1} = [1 \ \cdots \ 1]$

(h) We need to solve the below eq^n to obtain the solution.

$\Rightarrow (D - W)\, y = \lambda D y$

→ The first eigen vector (belonging to $\lambda = 0$) is $1 \cong [1, \ldots, 1]$

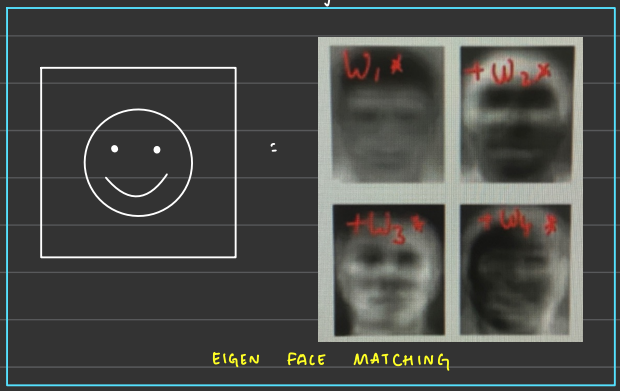
→ The second smallest eigen vector is the solution.

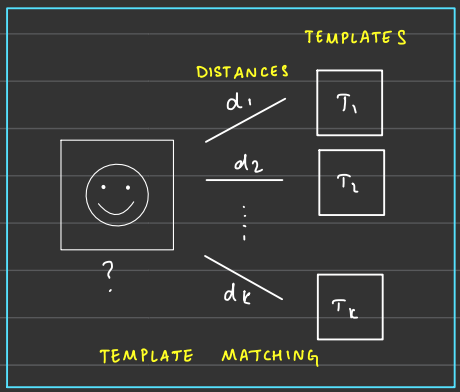
(i)

* Eigen faces approach :

→ Map image (eg: 100×100) to lower dimentional vectors (eg: 64) using $P < A$

→ Measure similarity to lower dimentional space.





The drawback of eigenfaces is that this method has a lack of discriminant power. This is because eigenfaces do not take class information into account. Eigenfaces is also scale sensitive.

This method requires uniform background and not robust when it deals with extreme variations in pose as well as in expression and disguise.

(j)

* Bag of words: Detect points of interest.
  → Extract features (eg: SIFT or HOG)
  → Cluster features to create a codebook (dictionaries)
  → compute a distribution of code words in each class
  → classify using distribution of code words.

Disadvantages:

* Bag of words leads to high dimentional feature vector due to large size of vocabulary
* Bag of words do not leverage co-occurance statistics between words. In other words this model assumes all words are independent.
* It leads to high sparse vectors as there is non zero values in dimentions corresponding to words that occur in the sentence.