11/11

*FFT & Filtering:

*

$$X[u] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{x2\pi un}{N}}$$

- $u = u_0 + u_1\frac{N}{2}$, $0 \le u_0 < N/2$, $u_1 = 0$ or 1

$$X\left[u_0+u_1\frac{N}{2}\right] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{x2\pi(u_0+u_1N/2)n}{N}}$$

$$= \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{x2\pi u_0 n}{N}} \cdot e^{-j\frac{x2\pi u_1 n}{2}}$$

$$= \sum_{n=0}^{N-1} x[n]\, \left(e^{-j\pi}\right)^{u_1 n} \cdot e^{-j\frac{x2\pi u_0 n}{N}}$$

$$= \sum_{n=0}^{N-1} x[n]\, (-1)^{u_1 n} \cdot e^{-j\frac{x2\pi u_0 n}{N}}$$

Now consider odd & even terms seperately

$$X\left[u_0+u_1\frac{N}{2}\right] = \sum_{n=0}^{N/2-1} x[2n]\,(-1)^{2u_1 n}\, e^{-j\frac{x2\pi u_0 x2n}{N/2}} + \sum_{n=0}^{N/2-1} x[2n+1](-1)^{u_1(2n+1)}\, e^{-j\frac{x2\pi u_0(2n+1)}{N}}$$

$$= \sum_{n=0}^{N/2-1} x[2n]\, e^{-j\frac{x2\pi u_0 n}{N/2}}$$

$$e^{-j\frac{x2\pi u_0}{N}} \cdot (-1)^{u_1} \sum_{n=0}^{N/2-1} x[2n+1]\, e^{-j\frac{x2\pi u_0 n}{N/2}}$$

$$= \sum_{n=0}^{N/2-1} x[2n]\, e^{-j\frac{x2\pi u_0 n}{N/2}}$$

$$+ (-1)^{u_1} e^{-j\frac{x2\pi u_0}{N}} \sum_{n=0}^{N/2-1} x[2n+1] \cdot e^{-j\frac{x2\pi u_0 n}{N/2}}$$

When $u_1 = 0$,

$$X[u_0] = X_{even} + e^{-j\frac{2\pi u_0}{N}} \cdot X_{odd}$$

$u_1 = 1$

$$X\left[u_0 + \frac{N}{2}\right] = X_{even} - e^{-j\frac{2\pi u_0}{N}} \cdot X_{odd}$$

- We have reduced $O(N^2)$ to $O(N \log N)$

Correlation & Convolution:

- Convolution → Flip filters
- Correlation → No flipping

---

*Convolution Theorem:

$g = x * h$ } Convolved

$F(G) = F(x) \cdot F(h)$ } Fourier transforms multiplied

$g = F^{-1}(F(G)) = F^{-1}(F(x) \cdot F(h))$

$\therefore \boxed{x * h = F^{-1}(F(x) \cdot F(h))}$

---

*Filters

* • Low pass filter  } Multiply with F.T.
Convolution theorem (Creates artifacts) → Due to the sinc like nature & we multiply this with image.

* • Butterworth low pass (No artifacts) ⟹ Similar to low pass.
- Gaussian low pass ⟹ (No artifacts)
- High pass filter = 1 − low pass filter. } Any of the above 3

*Laplacian in frequency domain:

$$F(\nabla^2 f(x,y)) = -(u^2+v^2)\,\mathcal{F}(u,v)$$ → Image

∴ The Laplacian filter in the frequency domain,

$$H(u,v) = -(u^2+v^2)$$

(∵ $F\left[\frac{d^n f(x)}{dx^n}\right] = (j\omega)^n F(\omega)$)

- We can use this in the frequency domain & then move back to the spatial domain.

---

*Notch Reject Filter: (Notch pass filter)

- In the fourier domain, apply a line (notch) as the filter & then regenerate image from new fourier.

(Example in slides)

12

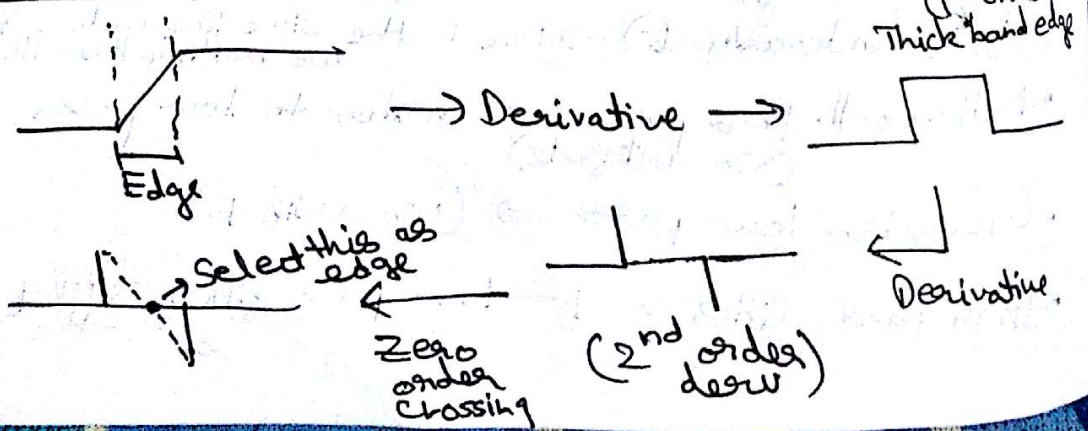
Edges & Hough Transform:

- Edge Models → Sharp Change (Amp)
  Gradual Change
  Peak up & down.

Detection:

*

- → Derivative →



Note: 1st order derivatives give thick edges
2nd order derivatives give double edge response.
⇒ Edge transition from light to dark or opposite → From sign of 2nd order derv.

———x———

⇒ Having noise ruins this method, so a low pass filter could be used. *

———x———

Image Gradient:

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \rightarrow [-1 \;\; 1], \quad \rightarrow \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$M(x,y) = \sqrt{g_x^2 + g_y^2}$ } Gradient magnitude

$\alpha(x,y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$ } Gradient angle

- Gradient direction ⊥ Edge direction.

⇒ Gradient from one direction is not the same as the other, so maybe an average would be better.

$f'_L(x) = f(x-1) - f(x)$
$f'_R(x) = f(x) - f(x+1)$, Average $= \frac{f(x-1) - f(x+1)}{2}$

∴ We use $[-1 \;\; 0 \;\; 1]$ ← Due to this.

———x———

Note: Separable filters → If it can be split into smaller filters.

———x———

* Sobel Filter:

- Averaging & then Derivative.
- The filter can be represented as a sum of two filters.

$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, $G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

$\Rightarrow |G| = |G_x| + |G_y|$

- We use both these filters to get the sobel edge.

We get G_x, G_y as averages & then derivative

$G_x = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix}$

Prewitt

* ~~Sobel~~ Filter

- Derivative after averages on neighborhood is done. (Since without average noise is an issue)

$H_x = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} 0.5 & 0 & -0.5 \end{bmatrix} = \frac{1}{6}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$, $H_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$

Note: Some sort of averaging is always done so that the noise is reduced.

* (Gaussian filtering)

** - $\frac{d}{dx}(f * g) = f * \left(\frac{d}{dx} g\right)$ } Just convolve with derivative of gaussian

---

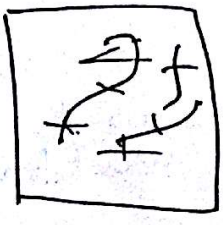
* Scale Space:

- Store multiple edge images with various σs for averaging.

$$G_\sigma(x,y) = e^{\frac{x^2+y^2}{2\sigma^2}}$$

---

* Edge Selection:



- If we want to extract the main curves out separately.

$\Rightarrow$ We start at a point and then

if neighbours gradient direction is similar to current & magnitude is similar, then we proceed (similar to flood fill with constraints)

---

1) Marr Hildreth Edge Detector:

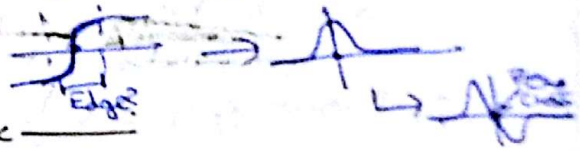
- Laplacian of a 2D gaussian ⇒  (mexican hat)

⇒ $I = \nabla^2(f * G)$, $G(x,y) = e^{\frac{-x^2+y^2}{2\sigma^2}}$ (In 3D)

$= f * (\nabla^2 G)$

↓ Laplacian of gaussian.

⇒ $\nabla^2 G = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial^2 y} = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{\frac{x^2+y^2}{2\sigma^2}}$

⇒ Find zero crossings & mark as edges 

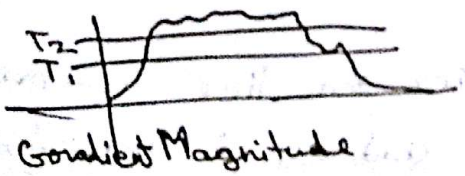
---

2) Canny Edge Detector: (Most commonly used)

* • Smooth image → With gaussian.
  • Compute gradient magnitude & gradient angle images.

⇒ Apply non-maxima suppression.
  ↳ Divide angle into 4 regions
  ↳ Consider adjacent if similar angle & similar magnitude.

⇒ Double thresholding & connectivity analysis to link edges



- Below T_1, suppress
- Between T_1 & T_2, use if pixel connected to one above T_2 else suppress etc.

• Many optimizations present.

---

* HOG Features: (Originally used for, SVM with human detection)

* (Histogram of oriented gradients)

⟹ Apply filters & compute gradients.

- Divide image into blocks, get all gradient magnitudes & angles For each block, iterate over pixels

$g = [10 \quad 12 \quad 15 \quad \ldots\ldots]$ } Number of pixels in block = size

$\theta = [30° \quad 90° \quad 25° \quad 10° \ldots\ldots]$

⟹ Now he just makes 9 bins for angles, and makes a histogram ⟹ [35 44 23 18 ....] (sum of magnitudes) 9 elements

- Now for each block we have a histogram

⟹ Just concatenate them all and we have our final HOG feature vector.

---

* Hough Transform:

* • Find most dominant lines in the image.

⟹ We could consider each point in the image & transform the space.

↳ That is on an edge. (Canny output)

* • We assume it lies on line $y = mx + c$

$c = y - mx$ ⟹ ∴ We try to find c, m in transformed space of (c, m) ⟹ Fix each m & get corresponding c, plot these and we will get lines and they intersect at points. (1 line for each pixel)

⟹ The points where most lines intersect, we consider those (c, m) as dominant lines in the original image.

* One problem with this is that m can vary from $-\infty$ to ∞.

⟹ So we would move to r, θ space.

* $\rho = x\cos\theta + y\sin\theta$

- Again iterating over each pixel on edge, we fix θ for a certain division, $0°, 10°, 20° \ldots, 350°$. We then calculate ρ for each and then plot. This time we dont get straight lines however.

⟹ We get curves which intersect in (ρ, θ) space. The points of maximal intersection, consider these lines as dominant in the original image.

---

*Note: Hough Transform is somewhat like, considers all possible lines through a point, for each point on edge. We then see which of these repeat the most & we consider these as dominant.

13

Note: Structure from motion → Only using SIFT for rotation of the camera

Note: Face landmark detection → Motion capture & transfer to animated characters.

- Scale, Viewpoint, Lighting & Occlusion
  ↳ Features should be independent of these.

* Corner Detection:

- Edges or plain/empty patches are not good for representing the features of an image since there can be many places where they can be matched.

∴ Corners are the most interesting

$$\max_{x,y} \sum_{w} (I(x,y) - I(x+\Delta x, y+\Delta y))^2$$

→ We want to maximize this.

*1) Harris corner detection: (Slides for intuition)

$$A(x,y) = \sum_{w} (I(x,y) - I(x+\Delta x, y+\Delta y))^2 \quad \rightarrow (1)$$

- We want $A(x,y)$ to be maximum for interest points.

- Taylor expansion,

$$F(x_0+\Delta x) \approx F(x_0) + F'(x_0)\Delta x + \frac{1}{2!} F''(x_0)\Delta x^2$$

$$\therefore F(x_0+\Delta x, y_0+\Delta y) \approx F(x_0,y_0) + \Delta x F_x(x_0,y_0) + \Delta y F_y(x_0,y_0)$$

⟹ Applying this to (1),

$$A(x,y) = \sum_{w} (I(x,y) - I(x,y) - \Delta x I_x(x,y) - \Delta y I_y(x,y))^2$$

$$= \sum_{w} I_x^2 \Delta x^2 + I_y^2 \Delta y^2 + I_x \Delta x I_y \Delta y$$

$$= \sum_{w} \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Autocorrelation matrix

$$A(x,y) = [\Delta x \;\; \Delta y] \begin{bmatrix} \sum_{\omega} I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

↑ auto correlation matrix

∴ The auto correlation matrix captures the structure of the local neighbourhood of a point (x,y).

⇒ They concluded that if both the eigen values are large → corner, one of the eigen values of the autocorrelation matrix is large → edge & if none are large → flat surface.

* We define a term R,

$$R = \det(M) - \alpha(\text{trace}(M))^2$$

$$R = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2) \quad \} \text{ Eigen values}$$

⇒ If $R > 0$ } Corner
$|R|$ = Small } Flat
$R < 0$ } Edge

Note: Algorithm → Stepwise in slides. } Harris Corner Detection

---

Note: At times we get too many points, so we apply non-maximal suppression.

↳ Corner response is highest, nearby are also high, so we try to ignore these since we consider the corner anyways.

⇒ This is rotation invariant since on rotation, eigen values don't change so we will end up marking the correct points. Additive intensity → Invariant
Multiplicative intensity → Not Invariant

* Scaling is not invariant, since on scaling up corners start becoming edges. So Harris might stop working.

⇒ Harris is not scale invariant.

———x———

*2) SIFT: (interest point detection)

- Consider laplacian of gaussian for various σs.
- Convolve image with that, then construct a scale space.

⇒ Now a point is a feature point if it is the maximum in a 9×3 region.
(9 = Pixels around selected point)
(3 = Image above, current & below in scale space)

- It was also considered that laplacian of a gaussian can be estimated by the difference of the gaussians.
- ⇒ This entire process was done considering different sizes of the image → Octaves.

———x———
———x———

*SIFT feature descriptor: (Similar to HOG)

(128 length vector describing a point)

- Take a window around interest point.
- Divide into a 4×4 grid. For each part of the grid, the gradient direction move them into 8 directions (360°/8)

Gradient direction } Split & add weights to 1 & 2 directions

- Now you would have $4 \times 4 \times 8 = 128$ length for a single point.

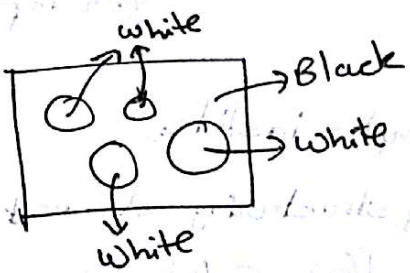
* ⟹ This is not rotation invariant, we solve this by computing gradient directions as a relative term (wrt overall gradient across chosen window).
  - ↳ So instead of just fixing the numbers 1 to 8 on direction, we can rotate it so that 1 gets aligned to the overall gradient of window.

Note: Color SIFT } So its discriminative to color.

14

* Morphological Operations:

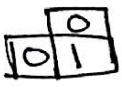
1) Binary images, num of connected components:



- So there are 4 connected components (White foreground)

a) Flood fill & find out all 4 individual components. → Recursion is slow.

b) Iterate over pixels, We also store a map
- Current pixel = 1.
- Left & Top = Already processed are both 0s.
- We add a new connected component.

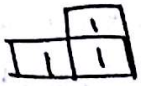
i) 

ii) 

- We say current pixel also belongs to component of top pixel.

iii) 

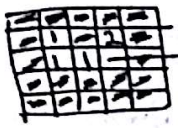
- Current pixel belongs to component of left pixel.

iv) 

- we can assign either left component or top component.

* ↳ If the components are different then we add a mapping to our map saying component-top = component-left

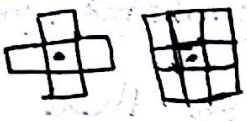
⟹ Once this is done, we iterate over image converting all equivalent components to a single component, using the map.

ex:  → This can be either 1 or 2, we add a mapping 1 ⟺ 2

———— x ————

* Structuring Elements:

- Small sets/subimages with a center used to probe an image by running it over the image.

ex:  etc. (Consider b/w images for implementation)

*1) Erosion: (Examples in slides)

* - Erosion of set A by structuring element B (set)

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

$$A \ominus B = \{z | (B)_z \cap A^c = \phi\}$$

⇒ Select a pixel only if all SE aligned pixels are 1 when centered over current pixel.

(All SE '1' pixels must lie on '1' pixels of image)

↓ To select pixel 'x' center SE at 'x'.

2) Dilation : (Examples in slides)

* [redacted]

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \phi\}$$

↑ Reflection of SE 'B'.

⟹ If atleast one SE [redacted] aligned pixel is 1 then we select the pixel

(Any one SE '1' pixel must lie on '1' pixels of image)

↓ To select pixel 'x' center SE at 'x'.

⟹ Properties: $A \oplus B = B \oplus A$ (Commutative)

$A \oplus (B \oplus C) = (A \oplus B) \oplus C$ (Associative)

Proofs: $A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \phi\}$

$= \{z \mid \hat{B} \cap (A)_z \neq \phi\}$

$= \{z \mid B \cap (\hat{A})_z \neq \phi\} = B \oplus A$

⟹ [redacted] $= A^c \oplus \hat{B}$ (Duality)

Proofs: [redacted] $= \{z \mid (B)_z \subseteq A\}$

$A^c \oplus \hat{B} = \{z \mid (B)_z \cap A^c \neq \phi\}$

$(A \ominus B)^c = \{z \mid (B)_z \cap A^c = \phi\}^c$

$= \{z \mid (B)_z \cap A^c \neq \phi\}$

$= A^c \oplus \hat{B}$

———x———

*3) Opening: (Example in slides) (Removes islands)

$$A \circ B = (A \ominus B) \oplus B$$

- Smoothens contours, eliminates thin protrusions, slight noisy errors etc.

4) Closing: (Example in slides) (Removes holes)

$$A \bullet B = (A \oplus B) \ominus B$$

- Remove narrow holes, small erroneous holes etc

⟹ Used in fingerprint detection (recognition)

---

*Morphological Algorithms:

1) Boundary Extraction:

- Difference between image & eroded image gives you the boundary.

*2) Hit & Miss Transform: (Example in slides)

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

- One SE fires completely & the other doesn't then we choose it.

⟹ So if either both SEs hit or both fail, we don't select it but if erosion with B_1 is a success on A & B_2 is a failure on A, then we select those.

3) Region Filling:

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad , \quad B = SE =$$ 

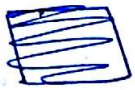
- Start with a point inside region with setting it to 1 & rest to 0.
- Dilate with B
- Take intersection of complement with A (So we don't cross boundary).

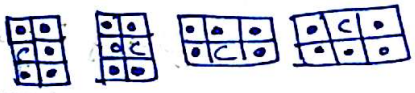
Repeat 4 till no changes occur.

4) Convex Hull:

a) Keep iterating over image & pixels which are empty with 3 or more neighbours that are full, we fill in the pixel. (Stop iteration if no changes)

b) Morphological approach:

- Consider 4 SE's 

 } Apply these multiple times on image.

⇒ Take union of 4 generated images to get the convex hull.

⇒ Take Hit or Miss Transform with each SE.

5) Skeleton:

- Set of all points which are equidistant from 2 closest points on object boundary

 →  } Skeleton.

———x———

*Gray Scale Morphology:

*1) Erosion: $f \ominus b = \min_{(s,t) \in b} \{f(x+s, y+t)\}$

*2) Dilation: $f \oplus b = \max_{(s,t) \in b} \{f(x-s, y-t)\}$

⇒ Similar to binary but the values are continuous. Min & max operations.

———x———

*Morphological Gradient:

- Dilation − Erosion ⇒ Boundary (some form of gradient)

———x———

Applications:

*1) Top Hat Transform:

$T_{hat}(f) = f - (f \circ B)$ (Only islands left out ↓ smaller than B)

(SE B should be a little larger than the components we want to preserve/ focus on rej

- ex: Small noise in the image / illumination changes are captured by $f \circ B$. We now subtract these from the original image so that the illumination across the image is constant.

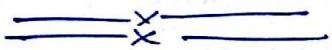
⇒ We can now do image segmentation using a decent threshold.

2) Granulometry: (example in slides)

- Use different sizes of SEs so that objects of different sizes are captured.

⇒ ex: sorting coins of different sizes in the image.

⇒ Objects of size smaller than SE get removed.



16/1

* Shape / Boundary Descriptors:

*• Boundary Following: (Boundary Detection) (example in slides)

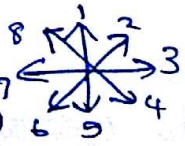
⇒ Start at the uppermost, leftmost point.

- Mark that pixel as b_0. The pixel to its left is b_1 (Background)

⇒ Now starting from b_1 go clockwise around b_0 storing last seen background as b_1 & if we see foreground set as b_0. Repeat process marking all seen b_0's. This will give us a single line boundary.

*1) Chain Codes: (Boundary descriptors)

(Not rotation invariant & not scale invariant)



- Consider 8 directions.
- Starting from some pixel, to go to the next clockwise pixel, select & store the direction in which we move.

⇒ This gives you a string ⇒ Chain code (This is cyclic)

- Consider lexicographically smallest order since chain code is cyclic → this is applied in shape numbers

* ⇒ To make it rotation invariant, just store the difference between every 2 points in the chain code assuming we move in cyclic order.

ex: Chain code : 3 5 4

Rotation invariant. 2 7 } We can use this code to describe the shape
(Difference)

2) Shape Number: (example in slides)

Rotation Invariant & but not scale invariant.

- Lexicographically smallest difference, in chain code order

~~Given same size of chain code you want, we discretize image into blocks of the required size.~~

⇒ Now you just get the chain code from this box of h x w where you set h & w, so you can kind of fix chain lengths & then get differences & then lexico ordering.

*3) Signature:

- 1D signature of 2D shape

⇒ Kind of hard with non convex shapes.

- Starting from center, at each degree (θ) from 1 to 360 store distance of boundary from center.

⇒ We can make this scale ~~& rotation~~ invariant, by dividing by maximum or something.

*4) Fourier Descriptors: (Examples in slides)

* • Consider each point on boundary (x, y) as a complex number $x + jy$.

⇒ Take fourier transform of all these, then we can take inverse fourier to regenerate the image.

• A great advantage of this is we can consider only a small fraction of fourier coefficients while regenerating but we still capture the broad shape.

⇒ We can also estimate rotation / scale etc between two images using these fourier descriptors.

* ↳ Low order coefficients can be used to compare the shapes

• Few rotation, translation, scaling, starting point variations etc would give transformed fourier descriptors

* ↳ Slides for examples

17//

* Region Descriptors:

• Compactness $= C = \frac{P^2 \rightarrow \text{Perimeter}^2}{\text{Area}}$ //

*

• Circularity ratio $= R_c = \frac{4\pi \times \text{Area}}{P^2} = \frac{\text{Area}}{P^2/4\pi}$ //

↓ For a circle its equal to 1.

⇒ Normalized area, Mean/Median intensity, max intensity etc could also describe a region.

*1) Topological descriptors: (Region descriptors)

- Holes in the ~~img~~ image H
- C = Connected components.
- * Euler Number = C − H = (Vertices − Edges + Faces)

—x—

ex: How to differentiate 0, 1, 8, 6, 2 ?

↳ Use euler number
↳ Then for clashes use convex hull
(Area of original ≈ Area of convex hull)

—x—

2) Texture:

*a) Statistical:

- Consider intensity histogram (Normalized) of image

$f \rightarrow h_f$

- * Mean intensity $= \sum_{i=0}^{L-1} i \cdot h(i)$

Variance $= \sum_{i=0}^{L-1} (i-M)^2 \cdot h(i)$

Skewness $= \sum_{i=0}^{L-1} (i-M)^3 \cdot h(i)$

↳ If symmetric around mean, skewness is 0, else +ve or −ve

Entropy $= \sum_{i=0}^{L-1} -h(i) \log(h(i))$

- $R = 1 - \frac{1}{1+\sigma}$ → Standard deviation
- Uniformity $= \Sigma(p(i))^2$

xs) ⇒ The issue with these comparisions is that
* histograms dont take care of spatial pixel relations (How x pixel occurred with y pixel etc).

xs) i) * • Co-occurrence Matrix (Gr): (Slides for example)
*
↳ LxL matrix if image has L intensity values
↳ Gr(i,j) = Number of times intensity j occurs on the adjacent right of intensity i.
↳ Called position operator.

ii)
- Normalize Gr by dividing by N = sum(Gr)
↳ Once we have Gr, we can calculate various things on rows or columns individually. → like variance, mean etc.

* • This coocurrence matrix is not illumination independant, so we move on to others.

ii) * Local Binary Pattern: (we could use any size of neighbours → Not just 3x3.)
*
- For all 8 neighbours of pixel, mark as 1 if its greater than pixel, else 0.
⇒ So this would give us an 8 bit representation
↳ An integer.

* • Consider patches of nxn pixels in the image. Calculate LBP for patches, then compute histogram of LBP for each patch.

- Now these histograms could be used to compare textures.

——— x ———

Note: In videos for action recognition, consider a 3x3x3 volume across time as well and compute LBP of 3x3 in normal spatial, then 3x3 horizontal & 3x3 vertical across time frames as well.

- Use these features to make an SVM learn

——— x ———

b) Spectral:

$f(x,y) \leftrightarrow F(u,v)$

$P(u,v) = |F(u,v)|^2$ } Power spectrum.

? * $P(r) = 2\sum_{\theta=0}^{\pi} P(r,\theta)$ ← (Since its symmetric)

* $P(\theta) = \sum_{r=0}^{L/2} P(r,\theta)$ & } Directionality of texture

↑ 360 element vectors } Can use these as features.

* i) Moment Invariants: (Fitting Ellipses)

- Hu's moment invariants
  - ↳ Combines multiple moments to get a lot of invariance (Translation, Rotation, etc)

——— x ———
——— x ———

Image Segmentation:

• Semantic segmentation → Assign labels to segments.

*Gestalt Principles of grouping: (example in slides)
(How humans segment things)

- Proximity
- Similarity
- Closure
- Continuation
- Common Fate — Moving in same direction etc
- Symmetry
- Parallelism
- Familiarity

—— × ——

*1) Two Class Segmentation:
(Foreground vs Background → Thresholding)

a) Global Thresholding:

*

$$g(x,y) = \begin{cases} 1, & f(x,y) > T \\ 0, & f(x,y) \leq T \end{cases}$$

- T is the same for all pixels in the image.

* i) Search for T:

- Start with a random selection of T from the histogram of the image.
- Then for each segment calculate the mean. And set $T = \frac{m_1 + m_2}{2}$

⇒ Repeat the process until convergence.

(This is a very basic algorithm)

* ii) Otsu's Algorithm:

* • Maximizes between class variance (The same as minimizing within class variance)

$$\Rightarrow \sigma_B^2(k) = P_1(k)(m_1(k) - m_G)^2 + P_2(k)(m_2(k) - m_G)^2 \rightarrow (1)$$

$$\left(P_1(k) = \sum_{i=0}^{k} p_i\right\} \text{for first half of histogram}$$

$$\left(P_2(k) = \sum_{i=k+1}^{n} p_i\right\} \text{For second half)}$$

- $m_1(k) = \frac{\text{Mean of first half}}{P_1(k)}$

$$m_2(k) = \frac{\text{Mean of second half}}{P_2(k)}$$

m_G = Overall Mean.

For thresholding the image $\Rightarrow$ We just try for all k & then choose a k which maximizes (1) (Otsu's could be extended for multiple thresholds)

$\Rightarrow$ If the image is noisy these segmentations don't work well, so we apply a blur filter first (Gaussian etc) and then perform the thresholding. *

* b) Local Thresholding: (Variable) (Adaptive)

* i) Useful if background varies in illumination as well. (example in slides)

- Divide image into subimages & apply Otsu's for each part & concatenate them.

⇒ If our regions are too small then
* we need to decide if we want to perform thresholding or just give it a constant label.
↳ Consider edges in region etc, based on that decide if you give it a background label, foreground label or perform Otsu's.

* ii) Per pixel variable thresholding:
*
- For each pixel, calculate mean $m(x,y)$ & variance $\sigma^2(x,y)$.
↳ around local neighborhood

$$g(x,y) = \begin{cases} 1, & f(x,y) > T_{xy} \\ 0, & f(x,y) \leq T_{xy} \end{cases}$$

* - $T_{xy} = a\sigma_{xy} + b m_{xy}$
or $a\sigma_{xy} + b m_G$ ↳ Global mean of image

—x—

Note: Adaptive thresholding algorithm → PR 2010 (Pai)
↓ Implement ↓ paper

—x—

(Slides example)

* Note: Barcode detection from image → Use case.
** 1) Apply per pixel variable thresholding.
2) Find all connected components.
3) For each connected component find moments (Fitting ellipses). Then since bar code stripes have large eccentricity, keep all components with large eccentricity. (Circles have 0 eccentricity)
4) We might still have some text left.
5) Make a histogram of major axis length, and choose the largest few peaks.
6) Then make a histogram of orientation & choose largest few peaks.

—x—

*2) Region Growing:

- Select a point in image as seed point.
- Then grow it into a region based on a few parameters / thresholds.

* 3) Region Splitting & Merging:

- For each region if σ (Variance) $> a$ & $0 < m < b$ (Mean), split into 4 regions & recurse. Break otherwise.

4) Clustering: (example in slides)

- A lot of algorithms exist
  - ⇒ k-means clustering
  - ⇒ Agglomerative clustering
  - ⇒ Spectral clustering

* i) k-means

- Choose random k cluster centers.
- Assign each datapoint to clusters.
- Take mean of each cluster & set the cluster center to mean.
- Repeat steps 2–3 multiple times. (till convergence)

- Apply clustering considering 3D (rgb) space for each pixel as a datapoint.

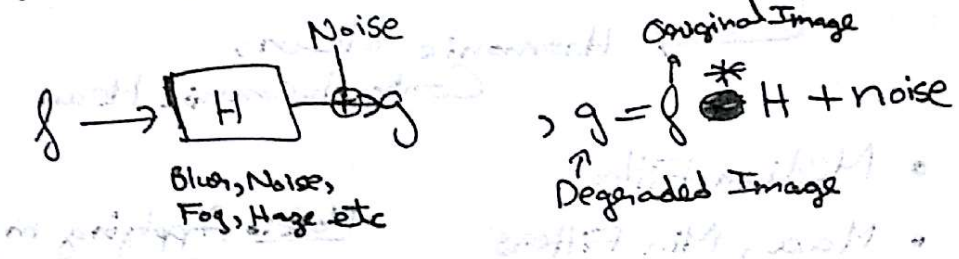
? Could also use (x, y, grayscale) ⇒ Spatial coherence could be used to merge similar nearby clusters.

Note: Mean shift; Graph cut → Advanced segmentation algorithms.

19

Image Restoration:

Note: Lightfield Cameras
- Motion Blur restoration
- Fog Removal



⇒ If we know → H, g → Recovery
- g → Blind recovery
- g, H partial → Semi blind recovery.
- f, g → System Identification

* 1) Degradation due to noise: (Graphs in slides)
(No filter)

i) Gaussian Noise

ii) Rayleigh Noise

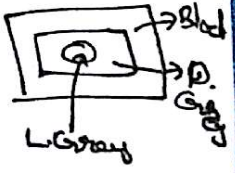
iii) Erlang (Gamma) Noise

iv) Exponential Noise.

v) Uniform Noise

vi) Impulse Noise → (Salt & Pepper etc)

* • We can estimate noise also by considering a patch in the image which should be single
⇒ To estimate noise models take a photo of a specific image →



* ↳ Then calculate the histograms & due to noise we will get many peaks instead of just 3.
↳ Estimate type of noise based on histogram

2) Restoration from Noise:

- Mean Filters
  - → Arithmetic Mean
  - → Geometric Mean
  - → Harmonic Mean, Contraharmonic Mean
- Median Filter
- Max, Min Filters
- Mid point Filter
  - → (Max in region + Min in region)/2
- Alpha Trimmed filter
  - → In neighbourhood take AM but ignore top 25% & bottom 25% values (robust to outliers)

Note:
- Applying mean filters multiple times, image converges.
- Applying median filter multiple times, does it converge?

i) Adaptive Mean Filtering:

- Identify parameters of noise by considering patches in the image which you know are originally of the same color. Get its histograms & estimate noise parameters.

⇒ Or we could use the noise detection method using the specific image.

$$\hat{f}(x,y) = g(x,y)$$

ii) Notch pass/reject

- For noise that occurs in a repetitive pattern.
- Take fourier image an apply notchs, & take inverse fourier.

———x———

*3) Estimation of degradation function: (No noise)

- Calculate the impulse response & use that to estimate the filter.
- To get impulse response in images, set one pixel to white & pass through the system and the output is the impulse response of the system. → This is the filter H.
- We create a mathematical model which we use to identify filter H.

Take photo where one small area is white

* i) Uniform Motion Blurring:

ii) Atmospheric Turbulance:

———— x ————

*4) Degradation with filter & noise:

* • Even if we know H, we cannot retrieve the original image from the degraded image.

$F = ?$, $G = F * H + N$

⇒ Directly dividing by H would give us,

$$\frac{G}{H} = F + \underbrace{\frac{N}{H}}$$

This might amplify our noise as well if value of H<1.

• We try to find another filter W, such that $GW = \hat{F}$ } Minimize error.

* i) Weiner Filter: (Examples in slides)

• Assume we have original image & degraded image, then we try to find 'W' when $GW = \hat{F}$

• We minimize error b/w F & $\hat{F}$

- Suppose we don't know the actual image, we can estimate its power spectrum to be a value K.

$$\Rightarrow K = \frac{|N(u,v)|^2}{|G_r(u,v)|^2}$$

$|N(u,v)|^2$ → Noise (estimated from degraded image) → Using the similar patch etc in the image

$|G_r(u,v)|^2$ → Degraded image

- Nowadays, we use this K in a lot of practical applications.

———x———

20

* Morphing:

- Simplest way to morph → Adjust alpha (Cross dissolve) → Good if images are aligned.
  - ↳ Average of images. (We want average of objects)

1) Align images & then do cross dissolve.
   - ↳ Doesn't work well on isolated objects etc. (Example in slides) (different) → Need warp.

*2) Local Warp & Cross Dissolve: → Colour

* - Manually compute feature correspondences on both images.

⇒ Based on these point to point correspondances we move to triangle mesh correspondances.

———x———

* i) Triangulation:

- Given n points, form triangles which don't intersect each other & cover the area of the convex hull.
- Brute force → $O(n^3)$ → Wont give best result.

→ Delaunay triangulation

↳ Triangulation T_1 is better than T_2 if minimum angle in $T_1 > T_2$.

↳ Any circle around edge of Δ'lg formed does not contain another point. → Few exceptions.

—— x ——

⟹ After triangulation of both images, at each timestep we can perform local warp. Colour is taken as a pixel (weighted) average after warp.

Steps:

1) At moving timesteps, calculate weighted average of each feature point & generate same triangulation on these points.

2) Warp img1 locally to new triangulation & warp img2 locally to new triangulation.

3) Take the weighted average of img1 & img2 for the current structure. → Colour would be a weighted average based on t. (Cross dissolve).

(Detailed steps in slides)

—— x ——

—— x ——

Note: Affine template matching

—— x ——

* Template Matching:

- Scale invariance can be achieved with pyramids
- Rotation → Ignore (Affine template matching does this)

1) SSD: (Doesn't work very well)

- Squared sum distance.
- This doesn't work well with illumination changes.

⇒ We want to try to be illumination invariant.

2) Correlation: (Measure of similarity)

* g = Template to match
  f = Image to search in

- Direct correlation doesn't work well. Correlation is somewhat like projection, so we want everything to be normalized.

⇒ $\text{corr}(f, g)$ → Needs to be extended.

⇒ $\text{corr}(f - \tilde{f}, g)$

↳ Move to Normalized cross correlation

⇒ Areas where template correlation > 0.75 → get locally maximal correlation points.

(examples in slides)

———x———