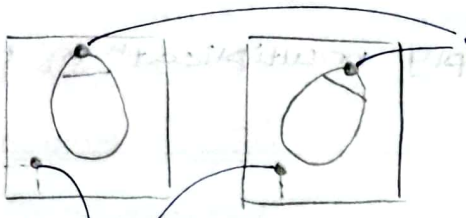
Topic-2:- Image Alignment:-

* In a typical greyscale image, intensity values range 0(black) - 255(white) in uint8
  0 - 1 in double.
  imshow() needs this, in matlab.

*



pixels in physical correspondance

pixls in digital corespondance.

* images I_1, I_2 are said to be aligned, if for every (x,y) in domain Ω the pixels are in physical correspondance.
  if not, say misaligned.
  we want to correct this relative motion. (or) relative motion.

→ **Motion models:-**

* lets denote coralinates (x_1, y_1) (in I_1) & (x_2, y_2) (in I_2). Physical corresponding points.

a) **translation:**

$$x_2 = x_1 + t_x$$
$$y_2 = y_1 + t_x$$

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

b) **rotation abt (0,0) Anticlock:-**

$$x_2 = x_1\cos\theta - y_1\sin\theta$$
$$y_2 = y_1\cos\theta + x_1\sin\theta$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

c) **rotation about (x_c, y_c):-**

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -s\theta & x_c \\ s\theta & c\theta & y_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 - x_c \\ y_1 - y_c \\ 1 \end{pmatrix}$$

e) **Scaling abt origin:-**

$$x_2 = c_x x_1$$
$$y_2 = c_y y_1$$

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

d) **Affine transformation:-**

* rotation, Scaling, shearing besides translation.

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

Assumption:-

the 2x2 matrix is not rank-deficient.
otherwise it'll map 2D to 1D.
or a point.

f) **Shearing:-**

vertical: $x_2 = x_1$
$y_2 = s_v x_1 + y_1$

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

horizontal:
$y_2 = y_1$; $x_2 = x_1 + s_h y_1$

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

* This 2D affine motion model : 6 degrees of freedom.
  - Only accounts for in-plane motion. Not appropiate for 3D models.
  - There exist more complex 3D models. We'll stick to this.
  - composition of multiple motions is simply multiplicat^{n} of their matrices.

→ Alignment with control points:-

* We figure out pairs of physically corresponding points - manually or automatically
  - no. of control points $\geq$ $\frac{\text{degrees of freedom}}{2}$
* Solve for unknown params (the Affine matrix) using least squares framework. i.e. pseudo-inverse method.

$$\begin{bmatrix} x_{21} & x_{22} & \dots & x_{2m} \\ y_{21} & y_{22} & \dots & y_{2m} \\ 1 & 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ y_{11} & y_{12} & \dots & y_{1m} \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

* There are methods for finding matching control points from both images.
  SIFT technique in CV.
  Scale-invariant feature transform. Don't know...

→ Alignment with mean square error:-

* MSSD - mean sum of squared deviation.

$$MSSD = \frac{1}{N} \sum_{x,y \in \Omega} \left(I_1(x,y) - I_2(x,y) \right)^2$$

N = no. of pixels in field of view.

after affine transf, we'll have some empty regions. those are out of FOV.

* now; motion parameters

$$T^* = \text{argmin}_T \, MSSD \left(I_1(v), I_2(Tv) \right)$$

How to do this argmin?

- There are many ways; the simplest (& least efficient) is a brute force search.

  θ: -45:1:45
  tx: -30:1:30 (in case of simple rotation & translat^{n})
  ty: -30:1:30

- Apply motion to I_1, keep I_2 fixed. Alternatively to I_2, keeping I_1 fixed.
- pick the param values that give the minimum MSSD.

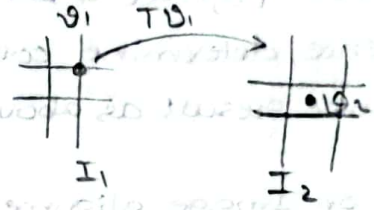
→ Image warping:-

I) Forward warping:- (→ Bakwas.) we have an Image I_1 & a transformat^n T. Gotta find final warped image I_2.

* Apply transfm T to every $v_1 = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ in I_1 to yeild new position v_2. & store the $I_1(v_1)$ at v_2.

- Handling required if v_2 is not integer point] highly likely. or not in FOV.

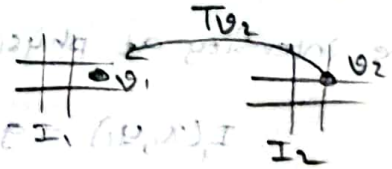
Issues:-
- Can leave holes in final image.
- Can allot multiple values to same pixel.
- Bakwas warping.

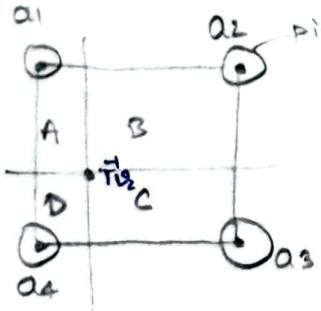


II) Reverse warping:-

* For every coordinate v_2 in final image I_2; do Tv_2 & copy the $I_1(v_1)$ from initial image.

- In case of non-integral reverse transformation, do interpolation.
- nearest neighbour interpolation or Bilinear interpolation.





a) nearest neighbour:-
use $I(a_4)$ since a_4 is closest.

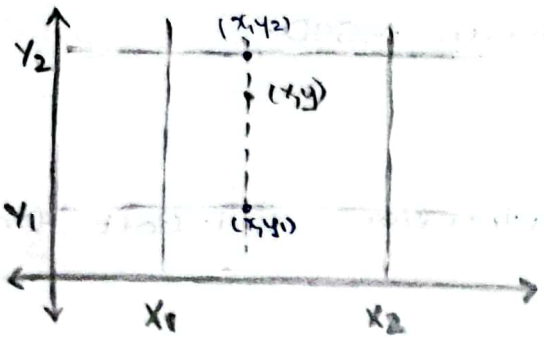
b) Bilinear interpolation:-
- weighted mean of all 4 pixel intensities.

the form $[a_0 + a_1x + a_2y + a_3xy]$ is bilinear.

we'll write $I = A \cdot a_3 + B \cdot a_4 + C \cdot a_1 + D \cdot a_2$

weights are areas. $\Sigma A = 1$.

→ Billinear interpolation in more detail:-



$$f(x,y_1) = \frac{x_2-x}{x_2-x_1} \cdot f(x_1,y_1) + \frac{x-x_1}{x_2-x_1} f(x_2,y_1)$$

$f(x,y_2) =$ ----

$$f(x,y) = \frac{y_2-y}{y_2-y_1} \cdot f(x,y_1) + \frac{y-y_1}{y_2-y_1} \cdot f(x,y_2)$$

finally; we'll get the Areas weighted mean.

you understand this structure of billinearity na!

* bilinear funct^n $f(x,y) = a_0 + a_1x + a_2y + a_3xy$.

we know $f(\cdot,\cdot)$ at 4 corners.

Hence, determine coefficients.

Same result as above.

→ Further on Image alignment with MSSD:- [measures of image alignment]

* MSSD is reffered as "Image similarity metric".

- major assumption in this alignment technique is that "physically corresponding points have same intensity".

Intensity changes in images:-

* if the intensity at physically corresponding points were related like...

$$I_1(x_1,y_1) = g(I_2(x_2,y_2)) \quad \forall \; x_1y_1 \& x_2y_2 \text{ corresponding physically}$$

then eazy!

$$\text{transformed MSSD} = \frac{1}{N} \sum_{xy \in \Omega} \left[g(I_2(x,y)) - I_1(x,y) \right]$$

again, only points in FOV.

- But practically we wont know the function 'g' na.

* if we knew that relationship is linear but we dunno the exact coeffcents:

$$I_1(x_1,y_1) = a \cdot I_2(x_2,y_2) + b \qquad \forall\ x_1,y_1,x_2,y_2$$

($I_1(x_1,y_1)$ and $I_2(x_2,y_2)$ → physically corresponding points)

then calculate corelation coefficant. (😊). gotta be maximum for aligned images.

$$NCC = \left| \frac{\sum_{(x,y)\in\Omega} (I_1(x,y)-\bar{I}_1)(I_2(x,y)-\bar{I}_2)}{\sqrt{\Sigma(I_1(x,y)-\bar{I}_1)^2 \cdot \Sigma(I_2(x,y)-\bar{I}_2)^2}} \right|$$

NCC = normalized corelation coefficant.

* so, now $T^* = \underset{T}{\operatorname{argmax}}\ NCC(I_1(v), I_2(Tv))$ $\quad T = \begin{pmatrix} c_0 & -s_0 & 0 \\ s_0 & c_0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad v = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

* In most practical scenarios, we dont even know if there exists such a functional relationship.

  -Lets use image histograms.

→ Image histograms:-

normalized histogram:- sums up to 1

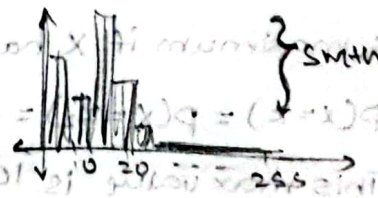
* say intensity lies in $[0, L-1]$.

- $p(r_k) = \frac{n_k}{H \cdot W}$; where n_k is num. of pxls with $I = k$.

- sometimes instead of discrete values, we use intervals as bins.

$$p([r_k^{min}, r_k^{max}]) = \frac{n_k}{H \cdot W}$$

- sum of all these $p([\cdot,\cdot])$ is 1.



smthing like this.

→ Joint image histogram:-

* joint image histogram is a function of the form $p(r_{k_1}, r_{k_2})$, where r_{k_1} and r_{k_2} represent intensity bins from the two images I_1, I_2.

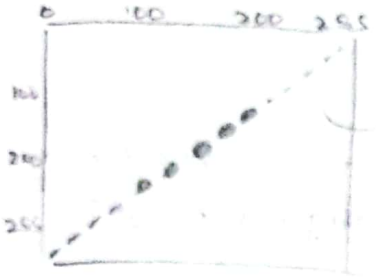
$p(r_{k_1}, r_{k_2}) = \frac{n_k}{H \cdot W}$ where n_k represent the no. of pixels which lie in bin r_{k_1} in I_1 & in bin r_{k_2} in I_2.

its not $p(r_{k_1}) \cdot p(r_{k_2})$! NOTE!

dont confuse.

<L_ image alignment> pg 40, 41. Joint histograms for aligned images and unaligned images.

* for aligned images, Joint Hist. is sharp.



is the joint histogram for negatives.

reveal the relationship b/w intensity of phy. corresponds pts.

* Observeation:-

Registered Images have sharp histograms.
(aligned)

non-aligned ones have dispersed histograms.

- measure of this dispersion is... ENTROPY.

→ Entropy:-

* consider a discrete RV X. with pmf as p(x).

* The entropy of X is the measure of uncertainity in X:
(which should have nothing to do with magnitude of X)

$$H(X) = -\sum_{x} p(X=x) \cdot \log_2[p(X=x)]$$

- first introduced by Claude E. Shannon.
- father of information technology.

$H(X) \geq 0$.

* Entropy is maximum if X has a discrete uniform distribut^{n}.

i.e. $p(X=x_1) = p(X=x_2) = \cdots$

This max value is $\log_2(|D_X|)$
= set of discrete values.

* Entropy is 0 (no uncertainity) if normalized histogram of X is a Kronecker delta-function.

* Joint entropy for two RV X,Y is:-

$$H(X,Y) = -\sum_{x \in Dx} \sum_{y \in DY} P(X=x, Y=y) \cdot \log_2[P(X=x, Y=y)]$$

- max joint entropy = $\log_2 |Dx| \cdot |DY|$
- min joint entropy = 0.

* Minimizing joint entropy is one method of aligning two images with different intensity profiles.

$$T^* = \text{argmin}_T H(I_1(v), I_2(Tv))$$

~ Finally, the components of an Image alignment Algorithm are:-

1) Choice of metric to optimize (Joint entropy $H(\cdot,\cdot)$, MSSD)
2) Choice of motion model (Affine, transt+rot)
3) Choice of interpolation to generate transformed img
4) Choice of optimization algo. (here, we did bruteforce)

Applications:-

1) template matching
2) Image panaromas

What we learnt:-

1) Affine motion modul
2) Forward & reverse warping
3) Field of view during alignment.
4) Measures of Img alignment: MSSD, NCC, Joint entropy.

What we didn't learn:-

- complicated motion models:- higher degree polynomials, non-rigid models (Eg: motion of heartbeat, face expressions)
- efficient techniques for optimizing the measure of alignment.

we did bruteforce.

Topic-3: Image Enhancement

→ Intensity transformations:-

$J(x,y) = T(I(x,y))$

could be $(I(x,y))^2$ or $\frac{I(x,y)+I(x,y-1)}{2}$.

* Transform image intensities so that resulting image is more visually suited.

Subjective perception by end user.

* Some popular image transformations are

- Image negatives
- Logarithm & Power Law transformations
- Contrast stretching
- Bit-plane slicing
- Histogram equalization
- Histogram Specification.

---

1. Image negatives:-

* if $I(\cdot,\cdot) \in [0, L-1]$ then $S(r) = L-1-r$

where S = output I
r = input I.

- This produces photographic negative.

---

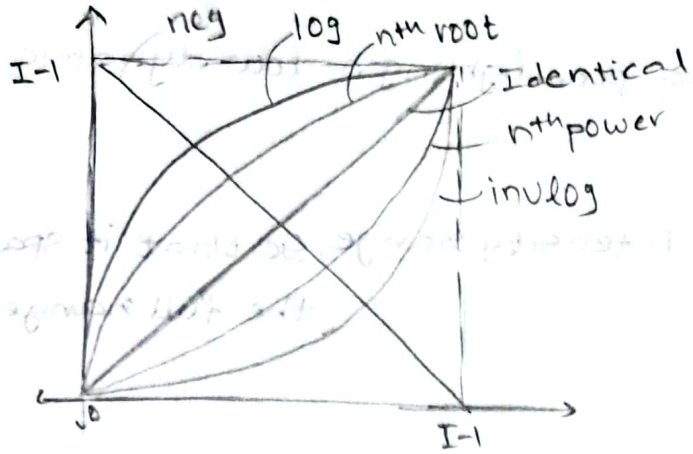
2. Logarithamic transformation:-

* Has the form $S(r) = C \cdot \log(1+r)$.

* This maps a narrow range of low intensity values to a wider range & maps a range of high intensity values to a narrow range.

* A logarithamic transformation compresses the intensity range of images whose initial intensity range is very wide.

- When we do fourier transformations in class, we will make heavy use of $\log(1+r)$ transformation.



Intensity transformation.

3. Power-law (Gamma) Transformation:-

* transformation of the form $S(r) = cr^{\gamma}$

$c > 0$ is a constant
$\gamma > 0$ is a constant

* if $0 < \gamma < 1$; a narrow range of low intensity values get mapped to a wider range & a range of high intensity values gets mapped to a narrower range.

(this is similar to log transformation)

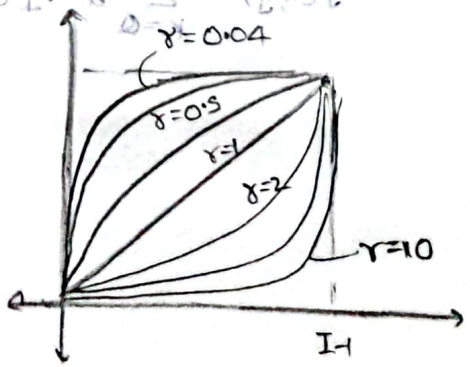
* Gamma-correction:

(in a cathode ray tube)

- the input intensity to a output voltage follows gamma law with γ b/w 1.8 to 2.5.
- So we preprocess image as $S(r) = \left(\frac{r}{c}\right)^{1/\gamma}$.

∴ finally we have $c \cdot \left[\left(\frac{r}{c}\right)^{1/\gamma}\right]^{\gamma}$

$= r$ ✓



<L3 pg 14> demo.

4) Contrast Streching:-

* low contrast images occur due to poor lighting or low dynamic range of camera sensor.

* contrast stretching expands the intensity range, so that it spans the full range.

$$S(r) = (L-1)\frac{(r-r_{min})}{r_{max}-r_{min}}.$$

Hence $r_{min} \rightarrow 0$

$r_{max} \rightarrow L-1$

---

5) Bit plane Slicing:-

* Intensity is a 8 bit value. Hence an image is a sum of 8 single-bit planes.

* From example, the first 4 planes (most significant planes) carry most of the image information.

∴ if we decide to use from plane 0 to 0'

$$I(x,y) = \sum_{K=0}^{0'} 2^{K-1} \cdot I(x,y,K)$$

→ 6. Histogram equalisation:- <see slides for illustrations>

* A method to improve image contrast.

* see examples <L3 pg 26,27>

* Seeks to apply an intensity transformation to the pixel intensities of a low-contrast image to convert it into one with nearly uniform

* consider $S = T(R)$

where R is random variable standing for intensity in original image. $R \in [0, L-1]$.

S is transformed RV.

- let values acquired by S, R be s, r.
- we are intrested in a T which is
  - Strictly monotonically increasing
  - $0 \le r \le L-1$ then $0 \le S \le L-1$.

* if R has pdf $P_R(r)$ then S:

$$P_S(s) = P_S(T(r)) = \frac{P_R(r)}{|T'(r)|}$$

---

* lets consider the following transformation:

$$S = T(r) = (L-1) \cdot \underbrace{\int_0^r P_R(w)\,dw}_{\text{cummulative prob till } r}$$

then $P_S(s) = \frac{1}{L-1}$ } what we needed! uniform!

* Hence; to perform this procedure; replace intensity r by cummulative probability till r multiplied by (L-1).

<illustrations in slides>

* in discrete RV case.

$$s_k = T(r_k) = L-1 \cdot \sum_{j=0}^{k} P_r(r_j)$$

∴ every pixel with intensity r_k is replaced by s_k.

<see lec slides pg38>

* discrete histogram after equalisation is not exactly uniform... due to rounding off of intensity values. (Quantisation)

→ 7. Histogram specification/matching:- <see slides for illustrations>

* Apply transformation, so that resultant has a prespecified histogram. (equalization is a special case).

* $P_R(r)$ is old image
$P_Z(z)$ is new image

$$s = T(r) = L-1 \int_0^r P_R(w)dw$$

consider RV z & transformation G s.t.

$$G(z) = (L-1) \int_0^z P_Z(v)dv$$

NOW

$$s = G(z) = T(r)$$

$$z = G^{-1}(s) = G^{-1}(T(r))$$

* Now the real deal is "choosing final histogram"! upto user judgement.

* <L3 pg 46,47>

Why Hist. Eqⁿ is not always right.

Cause, naturally our img must have had more darkness.

∴ HE will cause a washed out effect.

(Adaptive)

→ Local histogram equalization :-

* HE/HS so far are global methods.

This approach maynot enhance the details in small areas of image.

- for every (x,y) take a neighbourhood KxK &
  apply HE/HS on this neighbourhood & update value only at (x,y).
  like this for every pixel.

Illustrations pg: 56-58.

→ Spatial filters:-

→ Local spatial filters:-

* change pixel values based on some weighted combination of pixels from a small (often rectangular) neighbourhood around pixel.

$$g(x,y) = \sum_{i=-a}^{a} \sum_{j=-a}^{a} w(i,j) f(x+i, y+j)$$

w(i,j) — filter mask.

- Applied for each pixel in input image.
- Output consists a new 'filtered' image
- filter characteristics : weights w(i,j)
  size of neighbourhood.

* if operation on image pixels linear → linear filter.
  else → non linear filter.

* usually need zero padding to border areas.

→ corelation and convolution:-

* corelation:- moving a filter mask over the image & compute SOP.
  convolution:- similar to corelation; except that the mask rotated by 180° before moving over the image.

* image is [0 0 0 0 1 0 0 0]] unit impulse.
  w is [1 2 3 4] or Kronecker-delta fun.

---

corelation is [0 0 5 4 3 2 1 0].
convolution is [0 0 1 2 3 4 5 0].

* $(w \otimes f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x+s, y+t)$
  Correlat^{n}.

$(w * f)(x,y) = \sum_{-a}^{a} \sum_{-b}^{b} w(s,t) \cdot f(x-s, y-t)$
Convolution.

* convolution is commutative.

pad w,f till infinity with 0's.

$$(w*f)(x,y) = \sum_{-a}^{a}\sum_{-b}^{b} w(s,t)\, f(x-s, y-t) = \sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty} w(s,t)\, f(x-s, x-t)$$

$$= \sum_{-\infty}^{\infty}\sum_{+\infty}^{\infty} w(x-s, y-t)\, f(s,t)$$

- corelation ain't commutative.

* convolution is associative.

"proofs are much easier when we study fourier transforms."

$$((x*y)*z)(t) = (x*(y*z))(t)$$

- corelation is not!

→ Genesis of convolution:-

* system: input $x(t)$

output $y(t) = T(x(t))$

↳ transformation executed by system.

T satisfies two conditions:-

Linearity ⟶ $T[ax_1(t) + x_2(t)] = aT[x_1(t)] + T[x_2(t)]$

time-invariance.

↳ if $y(t) = T[x(t)]$

then $y(t-s) = T[x(t-s)]$

*

this is $\delta(t)$
unit impulse

output for unit impulse → impulse response $h(t)$

$$h(t) = T[\delta(t)]$$

* Response of a linear time invariant [LTI] system to any input x is convolution of signal with its impulse response.

$h(t) = T[\delta(t)]$

we express

$$x(t) = \sum_{u=-\infty}^{\infty} \delta(t-u)x(u).$$ Sifting property of δ.

$$T[x(t)] = T\left[\sum_{-\infty}^{\infty} \delta(t-u)\cdot x(u)\right]$$

$$= \sum_{u=-\infty}^{\infty} x(u)\cdot h(t-u)$$

$$= (h * x)(t)$$

hmmm... genesis of convolution.

→ genesis of corelation:-

* Gives similarity b/w signal w & signal f.

$$(w \otimes f)(x) = \sum_{t=-b}^{b} w(t)\cdot f(x+t)$$

→ Image noise:- could also have impulse noise. slides <pg 79, L3>.

* Gaussian noise:

$$f(x,y) = f_c(x,y) + \eta, \quad \eta \sim \mathcal{N}(0,\sigma^2)$$

$$\frac{e^{-(x-\mu)^2/2\sigma^2}}{\sigma\sqrt{2\pi}}$$

- thermal noise in camera.

See <pg 79 slides>

i) mean filter:-

$$g(x,y) = \frac{1}{(2a+1)^2} \sum_{i=-a}^{a} \sum_{j=-a}^{a} f(x+i, y+i)$$

- smoothing filter (amount of blur ∝ width of mask)
- image noise gets attenuated.
- images edges get blurred too 😮.

e) weighted mean filter:

Gaussian weights ∝ exponential weights.

$$g(x,y) = \frac{\sum_{i=-a}^{a} \sum_{j=-a}^{a} f(x+j, y+i)\cdot e^{-(i^2+j^2)/2\sigma^2}}{\sum_i \sum_j e^{\frac{-(i^2+j^2)}{2\sigma^2}}}$$

decay param.

3) median filter:-

* when there are wild outliers such as impulse noise, salt & pepper noise mean filter gives poor results.

  data transmission errors in old records.

* Median is more robust to large outliers in a dataset.

- for closely related datasets; if large outliers!

  mean changes drastically, the median remains same almost as long as these drastic changes affect less than half the no. of elements in the array.

  <L3, Pg 88-92 for illustrations>

* Since mean filter is linear & position invariant; it can be implemented as convolution.

  where as median filter can't be.

* Convolution has 2 cases: choices

  zero-boundary : zero padding done.

  cropped-boundary: don't convolve at end points.

HW2

Q3) consider a clean image $I(x,y)$.

lets say pixel intensity is I. Pdf of I is $p_I(x)$.

lets say, due to gaussian noise, η is added.

$$\therefore p_\eta(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}$$

given $p_I(x)$ & $p_\eta(x)$ & the fact that random variables I, η are independent:

$X = I + \eta$. (here X is r.v. denoting the pixel intensity after noise addition)

$$P_x(t) = Pr(X \le t) = \int_{y=-\infty}^{t} \int_{K=-\infty}^{\infty} p_{I\eta}(K, y-K)\, dK\, dy$$

cdf for X

$$= \int_{-\infty}^{t} \int_{-\infty}^{\infty} p_I(K) \cdot p_\eta(y-K)\, dK\, dy \quad (\because I, \eta \text{ are independent})$$

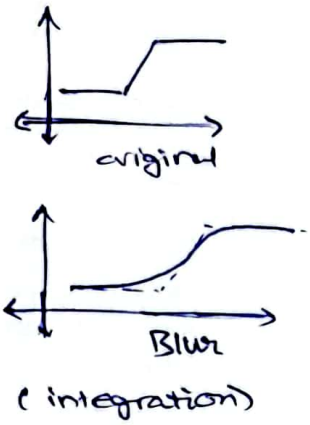
$$p_x(t) = \frac{\partial}{\partial t} P_x(t) = \int_{-\infty}^{\infty} p_I(K) \cdot p_\eta(t-K)\, dK$$

this is the convolution operation.

Hence

$$\text{Pdf}_{\text{Noisying img}}(x) = [\text{Pdf}_{\text{cleaning}} * p_\eta](x) \cdot \text{ where } p_\eta(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

→ Sharping filters:-

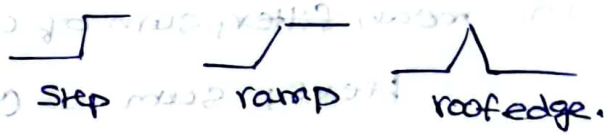


Aim:- To add back local derivative magnitudes to low contrast image.

* 1-D image f(x). is intensities. Digital derivative operators.

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$



* Along a ramp:

1st der is non zero

2nd der is zero; except at begin and end.

* 2nd derivative is more desirable for image sharpening.

(gives just 2 pixels; where as 1st der gives a thick line)

* 2nd der changes sign at Stepedge. zero crossing property.

→ Laplacian:-

* images are 2D. What type of derivatives?

* Isotropic operators; whose output doesn't depend upon the direction of discontinuity in image intensity.

→ rotationally invariant 2nd der. operator. Laplacian....

* $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ } rotationally symmetric operator.

$= f(x+1,y) + f(x,y+1) - 4f(x,y)$
$+ f(x-1,y) + f(x,y-1)$

adding 2nd derivatives along both diagonals.

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{ or } \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

* these masks are applied pointwise to entire image.

In mean filter, sum of coefficients = 1.

Here, sum = 0.

* de-emphazies regions with slow varying intensities.
Highlights discontinuities.

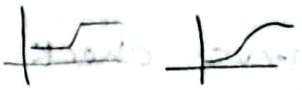
> Subtracting the laplacian from an image yeilds sharpening
> $$g(x,y) = f(x,y) - c\,\nabla^2 f(x,y); \; c>0$$

if mask is $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$ then $= f(x,y) + c\,\nabla^2 f(x,y)$.

∴ the convolution mask for sharpening is

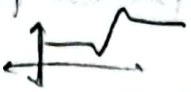
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - c\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -c & 0 \\ -c & 1+4c & -c \\ 0 & -c & 0 \end{pmatrix}$$

→ Unsharp masking:-



* f.

• Smooth the original using filtering:  add this to f.



$f_1 = f * g$.

• now subtract $f_2 = f - f * g$. f_2 denotes the sharp regions.

• add f_2 to f.

$$f_3 = f + K f_2$$

K=1 : unsharp masking

K>1 : high boost filtering.

→ Derivative filters & noise:-

* Derivative filters by themselves; exacerbate noise.

↳ use in conjunction with smoothing filters.

→ Sobel filters:-

→ x-smoothing. y-differentiating.

* $\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ → y-smoothing. x-differentiating.

X-Sobel filter I think.

Sum of coefficients = 0.

• now $\sqrt{g_x^2 + g_y^2}$ is Gradient map.

after applying sobel filter.

* note that $\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} (1 \;\; 2 \;\; 1)$ (seperable filters)

outer product or convolution again!

• convolution b/w

MxN image & K x L mask is O(MNKL)

But now, since seperable into (Kx1)*(1xL)

O(MN[L+K])

Associativity of convolution.

* For seperable filters; check rank = 1. of matrix.

- mean seperable filter = $\frac{1}{3}\begin{pmatrix}1\\1\\1\end{pmatrix} * \frac{1}{3}(1\ 1\ 1)$
- 2D gaussian mean filter

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

also seperable✓.

* Laplacian of Gaussians:
* using Gaussian smoothing & then laplacian sharpening.

we compute $\nabla^2(g_\sigma * f)$

equivalent to $(\nabla^2(g_\sigma)) * f$. Associativity.

pre compute this!

$$g_\sigma = \frac{1}{\sigma^2 2\pi} \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$\frac{\partial^2 g_\sigma}{\partial x^2} = g_\sigma \cdot \frac{1}{\sigma^2}\left(\frac{x^2}{\sigma^2} - 1\right)$$

$$\frac{\partial^2 g_\sigma}{\partial y^2} = g_\sigma \cdot \frac{1}{\sigma^2}\left(\frac{y^2}{\sigma^2} - 1\right)$$

$$LOG_\sigma(x,y) = \nabla^2(g_\sigma) = \frac{e^{-\frac{(x^2+y^2)}{2\sigma^2}}}{2\pi\sigma^4}\left(\frac{x^2+y^2}{\sigma^2} - 2\right)$$

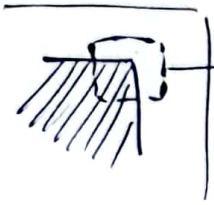
calculate coefficients beforehand.

Bilateral Filters: A smoothing filter. (not convolution) as not linear not space-invariant

* median filter may not always preserve edges.

→ we may need a non-linear filter.

> also assign weights to pixels; based on intensity value.



→ median filter won't preserve edges here.

↓ so we'll only care about relevant pixels.

$$BF[I]_p = \frac{1}{W_p} \cdot \sum_{q \in S} G_{\sigma_s}(\|p-q\|) \cdot G_{\sigma_r}(I_p - I_q) \cdot I_q$$

- $\frac{1}{W_p}$: normalizing factor.
- $G_{\sigma_s}(\|p-q\|)$: space weights
- $G_{\sigma_r}(I_p - I_q)$: Intensity weights

σ_s : as it increases, more smoothing.

σ_r : for modrat values : only I_q's close to I_p will affect

, for large values : Gaussian mean filter.

* **Cross-Bilateral:**

* noisy no-flash image
  +
  no noise- flash image (colors get distorted)

Cool!

So; apply bilater to no-flash image with [intersity weights corresponding to that of flash image.

just to know which pixels to actually bother, while doing mean.

Topic-7: Fourier Analysis

→ **Fourier series:** different from Fourier transformation

* Any periodic function can be accurately represented as a sum of sines & cosines of various frequencies.

$f(t)$ is periodic with period T:

$$f(t) = \sum_{n=-\infty}^{\infty} C_n \cdot e^{jn\cdot\frac{2\pi}{T}t} \quad \text{where} \quad C_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\cdot e^{-jn\frac{2\pi}{T}t}\,dt$$

discrete in frequency domain
continuous in time domain
for periodic signals

→ **Identity:** $\int_{-T/2}^{T/2} e^{j\frac{2\pi}{T}t\cdot n}\cdot e^{-j\frac{2\pi}{T}t\cdot m}\,dt = 1\cdot T$ if $n=m$

$= 0$ otherwise.

* Hence $e^{-j\frac{2\pi n}{T}t}$ act as orthogonal basis vectors... for $n\in\mathbb{Z}$

* only sine can't form a fourier series. We need both sine & cosine.

→ **Fourier transform:**

* Extension of fourier series, to non-periodic functions.

absolutely integrable or square integrable ones.

over $-\infty$ to ∞.

mathematically as.

$$F(f(t))(\mu) = \int_{-\infty}^{\infty} f(t)\cdot e^{-j2\pi\mu t}\,dt = F(\mu)$$

$F(f(t))(\mu)$: fourier coefficient transformation; μ: frequency; $F(\mu)$: fourier coefficient function.

Here

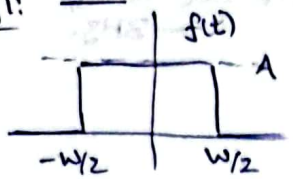
continuous frequency domain
continuous time domain
non periodic signals

Hence we would also have

$$f(t) = \int_{-\infty}^{\infty} F(\mu)\cdot e^{j2\pi\mu t}\,d\mu$$

* A signal and its fourier transform form a unique pair.

Eg1: Rect:-



rect.

$$F(\mu) = \int_{-\frac{W}{2}}^{W/2} A \cdot e^{-j2\pi\mu t}\, dt$$

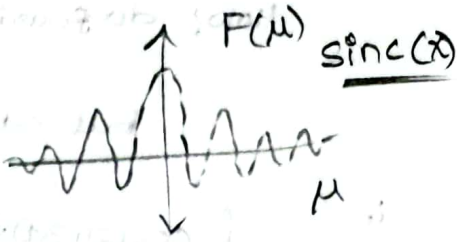
$$= A \cdot \frac{e^{-j2\pi\mu t}}{-j2\pi\mu}\Bigg]_{-W/2}^{W/2}$$

$$= \frac{-A}{j2\pi\mu} \times 2j\sin\left(-\frac{2\pi\mu W}{2}\right)$$

$$= \frac{A}{\pi\mu}\sin(\pi\mu W)$$

Luckily this is real. Coefficients

$$= AW \cdot \frac{\sin(\pi W\mu)}{\pi W\mu} = AW\,\mathrm{sinc}(W\mu)$$



Note:

time: freq:
rect → sinc
sinc → rect } tough complicated proof.

sinc() function = $\frac{\sin\pi x}{\pi x}$

* Kroneckor-delta vs

1850s

- discrete
- $\delta(0) = 1$
- sifting property yea. in Σ

1900s dirac-delta

- continuous
- $\delta(0) = \infty$
- Area under curve = 1
- gaussian, $\sigma \to 0$.
- sifting property yea. in $\int_{-\infty}^{+\infty}$.

$\delta(t)$ is dirac-delta

Sifting theorem:-

$$\int_{-\infty}^{\infty} f(t)\cdot\delta(t)\,dt = f(0)$$

$$\int_{-\infty}^{\infty} f(t)\cdot\delta(t-t_0)\,dt = f(t_0)$$

Eg2: dirac-delta function.

→ $\delta(t)$.

→ $F(\mu) = \int_{-\infty}^{\infty} \delta(t)\cdot e^{-j2\pi\mu t}\,dt = 1$

$$\int_{-\infty}^{\infty} 1\cdot e^{j2\pi\mu t}\,d\mu$$

$$\frac{e^{j2\pi\mu t}}{j2\pi t}\Bigg]_{-K}^{K} \quad t\to 0 \quad \frac{2j\sin(2\pi Kt)}{2j\pi t}$$

∴ for every frequency value; coefficient is 1.

$$F(\delta(t-t_0))(\mu) = e^{-j2\pi\mu t_0}$$

crazy!

now, not constant for all frequencies

complex fourier coefficients! common!.

Eg3:-

* $\cos(6\pi t)\cdot e^{-\pi t^2}$ has a 3Hz frequency in it.

How? do fourier transformation. getting a peak at 3Hz, -3Hz.

but at 5Hz, no peak.

in $\int_{-\infty}^{\infty} \cos(2\pi 3t)\cdot e^{-\pi t^2}\cdot e^{-j2\pi\mu t}\,dt$

3Hz damped

if $\mu = 3$; then real part is always positive.

Eg4: Sine & cosine signals.

1) $\sin(2\pi w t) = f(t) = \frac{1}{2j}\left(e^{j2\pi wt} - e^{-j2\pi wt}\right)$

$F(\mu) = \frac{1}{2j}\left(\delta(\mu - w) - \delta(\mu + w)\right)$

dirac deltas.

$\int_{-\infty}^{\infty} e^{j2\pi wt}\cdot e^{-j2\pi\mu t}\,dt = \delta(\mu - w)$

2) $f(t) = \cos(2\pi wt) = \frac{1}{2}\left(e^{j2\pi wt} + e^{-j2\pi wt}\right)$

$F(\mu) = \frac{1}{2}\left(\delta(\mu - w) + \delta(\mu + w)\right)$

* Real valued time domain, usually gives complex fourier coefficients.

∵ frequency values are real anyways.

Coefficients are complex.

→ Properties of Fourier Transform:

1) Linearity:

$$F(af+g)(\mu) = aF(f)(\mu) + gF(g)(\mu)$$

(a: complex valued scalar.)

- since integration is a linear operator.

2) shift invariance: doesn't hold.

Fourier shift theorem

$$F(f(t-t_0))(\mu) = e^{-j2\pi\mu t_0} F(f(t))(\mu)$$

not shift invariant.

$$\neq F(f(t))(\mu)$$

Hence;
- fourier transform can't be done via convolution...

* $e^{-j2\pi\mu t_0}$ ≡ phase factor.

variant:

$$F^{-1}(F(\mu-\mu_0))(t) = e^{j2\pi\mu_0 t} f(t)$$

phase factor.

* 3) convolution:- Super result

* in continuous setting

$$(f*g)(x) = \int_{s=-\infty}^{\infty} f(s)\cdot g(x-s)\,ds$$

convolution.

* "Fourier transformation of a convolution is product of individual fourier transformations"!

extremely important theorem in signal processing.

$$F((f*g)(t))(\mu) = F(f(t))(\mu)\cdot F(g(t))(\mu)$$

$$= \int_{-\infty}^{\infty} (f*g(t))\cdot e^{-j2\pi\mu t}\,dt$$

$$= \int_{t=-\infty}^{\infty}\int_{s=-\infty}^{\infty} f(s)\,g(t-s)\cdot e^{-2\pi\mu t}\,ds\,dt$$

$$= \int_{s} f(s)\cdot F(g(t))(\mu)\cdot e^{-2\pi\mu s}\,ds$$

$$= F(f(s))(\mu)\cdot F(g(t))(\mu)$$

Also,
"Fourier trans. of a product is convolution of fourier transformations".

Wah!

→ Convolution Theorem:-

* Fourier transform of convolution of two functions; is the product of their resp. fourier transformations at μ. (freq.)

* Also; Ft of product of two functions; is the convolution of their individual FTs at μ.

$$F\{(f(t)\cdot g(t))(\mu)\} = [F(f(t)) * F(g(t))](\mu)$$

$$F\{f * g(t)\}(\mu) = F(f(t))(\mu)\cdot F(g(t))(\mu)$$] proved last page

wah! So symmetric.

Proof by calculating Inv FT of RHS

$$F^{-1}(F_f * F_g(\mu)) = \int_{-\infty}^{\infty} F_f * F_g \cdot e^{j2\pi\mu t}\, d\mu$$

$$= \int_{\mu=-\infty}^{\infty} \int_{s=-\infty}^{\infty} F_f(s)\cdot F_g(\mu - s)\cdot e^{j2\pi\mu t}\, ds\, d\mu$$

$$= \int_{s=-\infty}^{\infty} F_f(s)\cdot e^{j2\pi s t}\cdot g(t)\, ds$$

$$= g(t)\cdot f(t)$$

Ta-da

By uniqueness,

we prove FT of product ≡ conv. of FT.

Property 4: Parseval's Theorem

hmmm...

* $$\int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} |F(\mu)|^2 d\mu$$

is this any useful??

hmmm...

$$\int_{-\infty}^{\infty} f(t)\,(f(t))^{*}\, dt$$

$$= \int_{t=-\infty}^{\infty} f(t)\cdot\left(\int_{-\infty}^{\infty} (F(\mu))^{*} e^{-j2\pi\mu t}\, d\mu\right) dt$$

$$\int_{-\infty}^{\infty} F(\mu)\cdot F(\mu)^{*}\, d\mu$$

$$= \int_{-\infty}^{\infty} |F(\mu)|^2 d\mu$$

→ Parseval's theorem:-

* $$\int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} |F(\mu)|^2 d\mu$$

- has interesting consequences in image compression.

* A variant; more general form is

$$\int_{-\infty}^{\infty} f(t) \cdot g^*(t)\, dt = \int_{-\infty}^{\infty} F(\mu) \cdot G^*(\mu)\, d\mu$$

(* : complex complement)

→ Differentiation Theorem:-

* $$F(f'(t))(\mu) = j2\pi\mu F(\mu)$$

Why so crazy crazy results!

$$\int_{-\infty}^{\infty} f'(t) e^{-j2\pi\mu t}\, dt$$ (1st term)

$$= \underbrace{e^{-j2\pi\mu t} f(t)\Big]_{-\infty}^{\infty}}_{0} + \int_{-\infty}^{\infty} (+j2\pi\mu) \cdot f(t) \cdot dt \cdot e^{-j2\pi\mu t}$$

$$= j2\pi\mu F(\mu)$$

* $$F(f''(t))(\mu) = -4\pi^2\mu^2 F(\mu)$$

hinting at something.... $F(f^n(t))(\mu) = \_\_\_\_$. hehe..

→ $$\boxed{F(f^n(t))(\mu) = (j2\pi\mu)^n F(\mu)}$$

cool.

→ Time limited and band limited:

* band limited if for every μ, $|\mu| > B$; $F(\mu) = 0$.

- Any signal stored in a digital computer is time limited.

Extend to image!

space-limited.

→ "non-zero time limited can't be band limited"

&

"non-zero band limited can't be time limited."

basically; due to the FT relation b/w them.

* we'll prove this!

1) say $g(t)$ is time limited

& for $|t| > \frac{W}{2}$; $g(t) = 0$

then $g(t) = g(t) \cdot \text{rect}(t; \frac{W}{2})$

$G(\mu)$ is its FT

then

convolution.

$$G(\mu) = G(\mu) * W\text{sinc}(W\mu)$$

now; we can try & check.

$G(\mu)$ can't be band limited.

2) Same proof as above.

$$G(\mu) = G(\mu) \cdot \text{rect}(\mu, W/2)$$

told you! fact!

$$g(t) = g(t) * B\text{sinc}(Bt)$$

* we already know $G(\mu) = G(\mu) * S(\mu)$

but now; for a time limited signal,

$$G(\mu) = (G * \text{sinc}(W\mu))(\mu)$$

such a frequency function SHOULD have time limited signal.

* A band limited signal is infinitely differentiable on time domain.

$$f(t) = \int_{-B}^{B} F(\mu) \cdot e^{j2\pi\mu t} \, d\mu$$

$$f'(t) = \int_{-B}^{B} F(\mu)(2\pi\mu j) \cdot e^{j2\pi\mu t} \, d\mu$$

cool man!
very cool.

→ FT in 2D:-

* $$F(\mu,\vartheta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) \cdot e^{-2j\pi(\mu x+\vartheta y)} \, dx\,dy$$

* $$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\vartheta) \cdot e^{j2\pi(\mu x+\vartheta y)} \, d\mu\,d\vartheta$$

* All mentioned theorems, hold in high dimension too!

→ Higher dimension FTs posses property of seperability

- First calculate FT wrt x, then wrt y.