# Spatial Domain Filtering

$g(x,y) = T[f(x,y)] \longrightarrow s = T(r)$

- $g(x,y)$: image filtered in spatial domain
- $T$: function
- $f(x,y)$: image
- $s$: intensity
- $r$: pixel

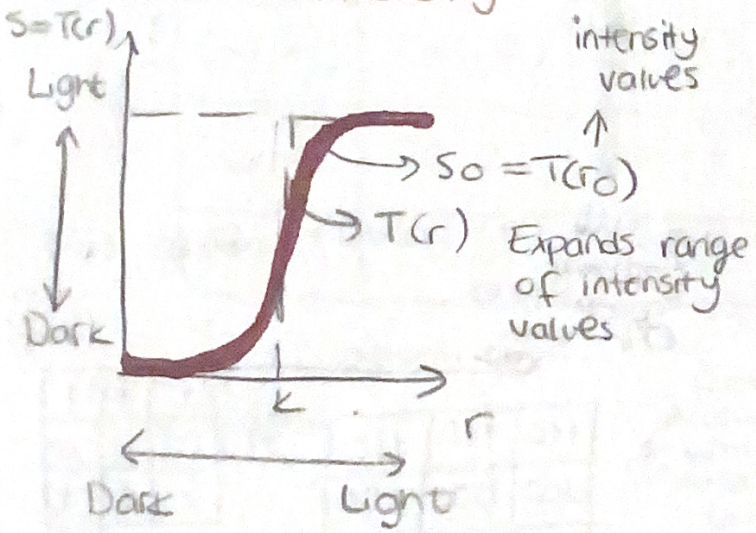
→ Single Pixel Operations → Negative Image
→ Contrast Streeching

→ Neighborhood Operations → Averaging Filter
→ Median Filtering

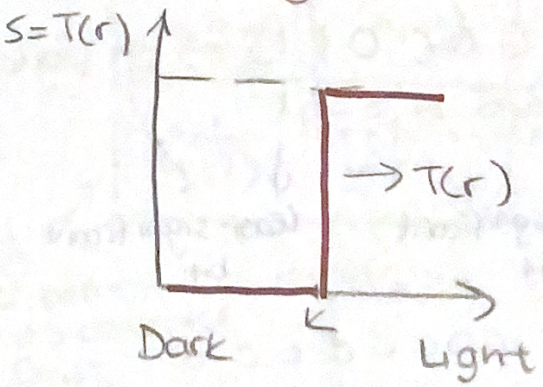
→ Geometric Spatial Functions → Scaling → $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$
→ Rotation → $\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$
→ Translation → $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$

## Intensity Transformations

### Contrast Stretching

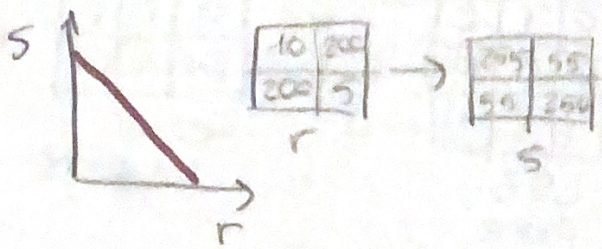



### Thresholding Function




### Image Negatives

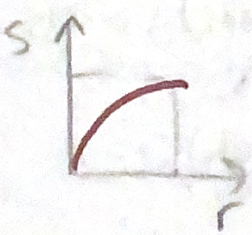
- $S = L-1-r$




### Log Transformation

- Increases the brightness
- $S = c\log(1+r)$

→ Expand the values of dark pixels & compress bright pixels




$[0, L/4] \rightarrow [0, 3L/4]$

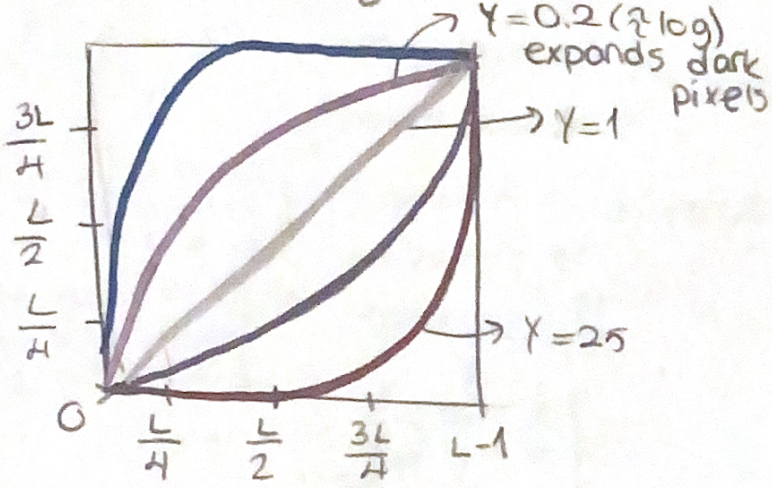
## Power Law (Gamma) Transform

$s = cr^{\gamma} \rightarrow c, \gamma > 0$

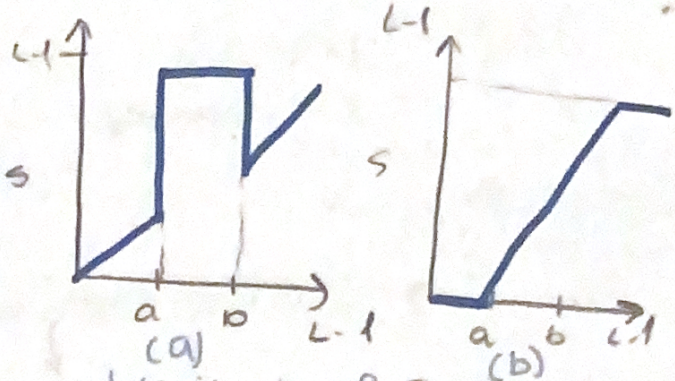
$0 < \gamma < 1 \rightarrow$ expands dark pixels

$\gamma > 1 \rightarrow$ compresses dark pixels

$c = 1 \rightarrow$ identity




## Piecewise Linear Transformation




→ Intensity Level Slicing

Highlight specific range of intensities in an image.

(a) Displays only one value in range of interest and rest is black (converts to binary)

(b) Brightens a range & leave the rest.

## Bit Plane Slicing

1 1 0 0 0 0 1 0 → Bit plane

8 7 6 5 4 3 2 1

most significant bit

least significant bit

0 → 0 0 0 0 0 0 0 0

intensity

255 → 1 1 1 1 1 1 1 1

MSB

All other intensities

Store less planes (with highest significance) to save memory.

Reconstruction → Multiply pixels of nth plane by $2^{n-1}$ (binary bit → decimal)

→ We can convert images to bit planes.

ex: 3 bit image

| | | |
|---|---|---|
| 110 | 111 | 110 |
| 000 | 000 | 000 |
| 001 | 101 | 101 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |

MSB plane

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

LSB plane

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

Center bit plane

# Spatial Filtering

## Linear Spatial Filtering

Sum of products between image $f$ & filter kernel $w$.

$$g(x,y) = w(-1,-1)\,f(x-1,y-1) + w(-1,0)\,f(x-1,y) + \dots + w(0,0)\,f(x,y) + \dots + w(1,1)\,f(x+1,y+1)$$

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

## Spatial Correlation & Convolution

Correlation: Moving center of a kernel over an image & computing sum of products at each location

Measurement between similarity of two signals

$(x,y)$ → any point in the image

$w$

| | | |
|---|---|---|
| $(-1,-1)$ | $(-1,0)$ | $(-1,1)$ |
| $(0,-1)$ | $(0,0)$ | $(0,1)$ |
| $(1,-1)$ | $(1,0)$ | $(1,1)$ |

↑ kernel coefficients

$f$

| | | |
|---|---|---|
| $(x-1, y-1)$ | $(x-1, y)$ | $(x-1, y+1)$ |
| $(x, y-1)$ | $(x, y)$ | $(x, y+1)$ |
| $(x+1, y-1)$ | $(x+1, y)$ | $(x+1, y+1)$ |

↑ pixel values under kernel when it's centered on $(x,y)$

Correlation: $(w \cdot f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$

Convolution: Measurement of effect of one signal on another

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

↑ flip the kernel on both sides

When kernel is symmetrical, correlation = convolution

# Filters

## Prewitt Filter

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \text{Vertical edges}$$

Convolving Prewitt will give you:

$$\nabla I = \frac{\nabla I}{\partial x} + \frac{\nabla I}{\partial y}$$

$\frac{\nabla I}{\partial x}$: horizontal change

$\frac{\nabla I}{\partial y}$: vertical change

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow \text{Horizontal edges}$$

## Sobel Filter

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Used in edge detection

## Smoothing

We can do $\longrightarrow$ $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 9$ which is brute force

Better solution $\longrightarrow$ Gaussian Smoothing

$$\begin{bmatrix} 8 & 9 & 8 \\ 9 & 10 & 9 \\ 8 & 9 & 8 \end{bmatrix}$$

Closer ones should be smoothed less

$$G(i,j) = \frac{1}{2\pi\sigma^2} e^{\frac{-(i^2+j^2)}{2\sigma^2}}$$

# Histogram Processing

Distribution of intensity values.

$h(r_k) = n_k$ for $k = 0, 1 \dots L-1$

- $r_k$: intensity level
- $n_k$: number of pixels in $f$ with intensity $r_k$

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$

- $p(r_k)$: estimates of probabilities of intensity levels
- $M$: row, $N$: column


# Histogram Equalization

Used to improve contrast.

Cumulative Distribution Func $\rightarrow \sum_{0}^{255} \frac{n_j}{n}$ $\quad P_r(v_k) = \frac{n_k}{n}$ ← probability density function

What we do $\rightarrow$ $s_i = cdf_i \cdot (L-1)$

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 5 | 6 | 1 |
| 1 | 6 | 5 | 1 |
| 1 | 1 | 1 | 1 |

$Pr_1 = 12/16 \quad cdf_1 = 12/16 \quad s_1 = 12/16 \cdot 6 = 4.5 \approx 4$

$Pr_5 = 2/16 \quad cdf_5 = 14/16 \quad s_5 = 14/16 \cdot 6 = 5.25 \approx 5$

$Pr_6 = 2/16 \quad cdf_5 = 16/16 \quad s_6 = 1 \cdot 6 = 6$

# Histogram Matching

Modifying images based on the contrast of another one.

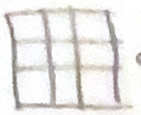
1. First equalize the histogram of both images. $((L-1) \sum_{j=0}^{k} Pr(r_j))$
2. Map each pixel of image A to B $\rightarrow G(z_q) = s_k$
3. Modify A according to B $\rightarrow z_q = G^{-1}(s_k)$

# Spatial Filtering

## Linear Spatial Filtering

Sum of products between image f & filter kernel w

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \dots + w(0,0)f(x,y) + \dots + w(1,1)f(x+1,y+1)$$

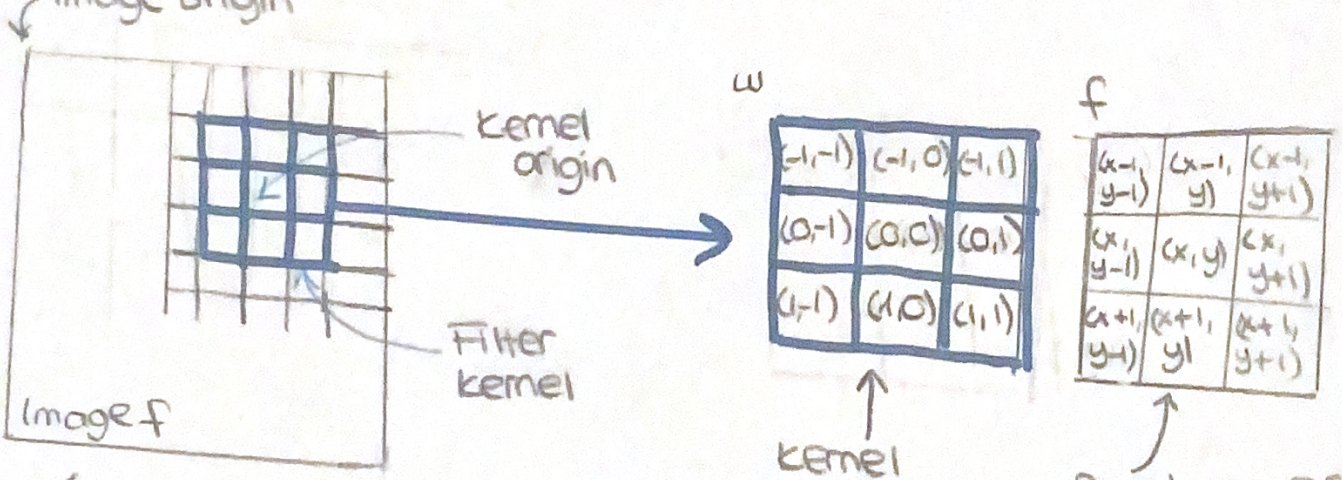


← on 3×3 kernel

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

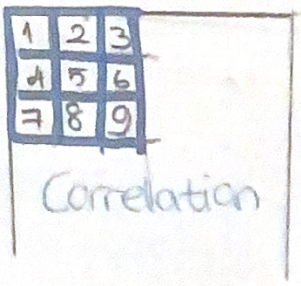
## Spatial Correlation & Convolution

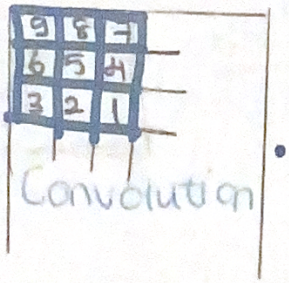
- Correlation: Moving center of a kernel over an image and computing sum of products at each location.
  It's a measurement of the similarity between 2 signals.




- Correlation Formula → $(w * f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$
- Convolution: Measurement of effect of one signal on the other signal.

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



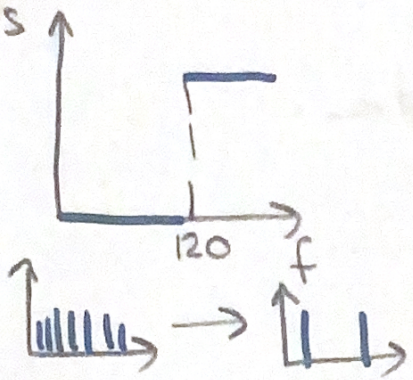




Kernel w: 1 2 3
4 5 6
7 8 9

- When kernel is symmetrical
  Correlation = convolution

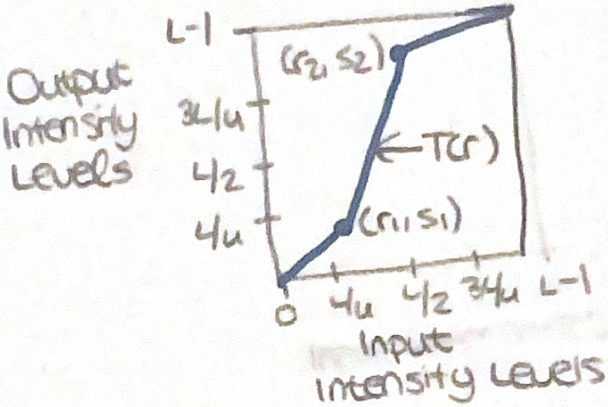
## → Image Binarization

Makes images B&W




## → Contrast Stretching

Expands range of intensity levels so that it comes to an ideal range.




---

# Histogram Processing

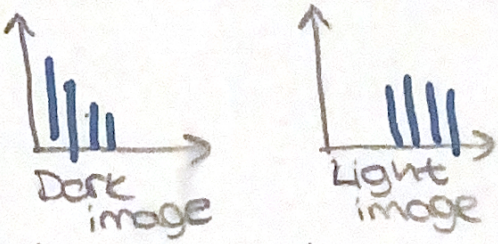
Distribution of intensity values.

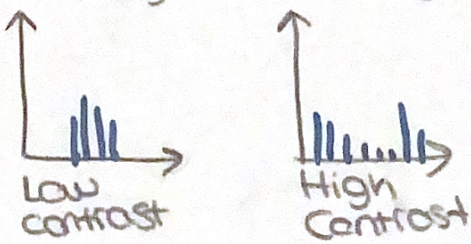
→ $h(r_k) = n_k$ for $k = 0, 1, \ldots L-1$

- $r_k$ → intensity level
- $n_k$ → number of pixels in f with intensity $r_k$

→ $p(r_k) = \frac{n(r_k)}{MN} = \frac{n_k}{MN}$

- $n(r_k)$ → raw
- $MN$ → column
- $p(r_k)$ → estimates of probs of intensity levels




## Histogram Equalization

Used to improve contrast.

Cumulative Distr. Func → $\sum_{0}^{255} \frac{n_j}{n}$ $\quad Pr(r_k) = \frac{n_k}{n}$ → probability density function

What we do → $s_i = cdf_i \cdot (L-1)$

ex)

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 5 | 6 | 1 |
| 1 | 6 | 5 | 1 |
| 1 | 1 | 1 | 1 |

$Pr_1 = 12/16$ $\quad cdf_1 = 12/16$ $\quad s_1 = \frac{12}{16} \cdot 255$

$Pr_5 = 2/16$ $\quad cdf_5 = 14/16$ $\quad s_2 = \frac{14}{16} \cdot 255$

$Pr_6 = 2/16$ $\quad cdf_6 = 16/16$ $\quad s_3 = \frac{16}{16} \cdot 255$

PDF → CDF → Equalization

# Spatial Operations

## Single Pixel Operations

Alter the intensity of its pixels individually.

$s = T(z)$ ← operation

## Neighborhood Operations

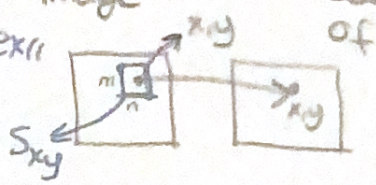
$S_{xy}$ → set of coordinates of a nbhd

ex// Compute the average value of the pixels in the input image with coordinates in the set $S_{xy}$ → Blurs

$$g(x,y) = \frac{1}{mn} \sum f(r,c)$$

$g$: generated image; $r$ & $c$: row & col coordinate of pixels in $S_{xy}$

ex//




## Geometric Transformations

Modify spatial arrangement of images. Consists of following

1. Spatial transformation of coordinates
2. Intensity interpolation that assigns intensity values to spatially transformed pixels

Transformation coordinates can be expressed as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

- $x', y'$: pixel coord. in transformed image
- $t_{11}, t_{12}, t_{21}, t_{22}$: these values determine the type of operation
- $x, y$: pixel coordinates in original image

ex// $(x', y') = (x/2, y/2)$ → shrinks the original image by half.

### Affine Transformations

- → Scaling
- → Translation
- → Rotation
- → Sheer

} geometric ops

ex// Identity → $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

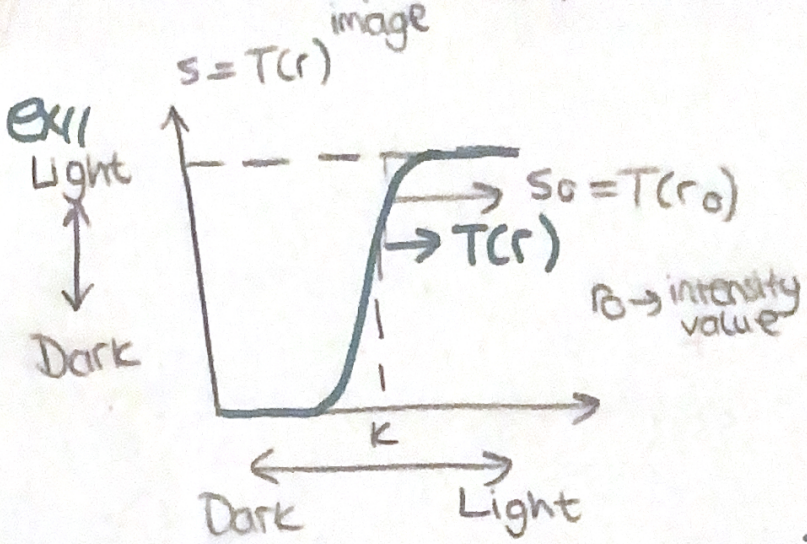
Scaling → $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

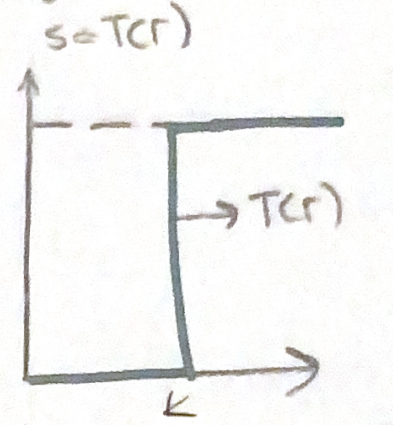
Rotation → $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

# Intensity Transformations & Spatial Filtering

$g(x,y) = T[f(x,y)]$

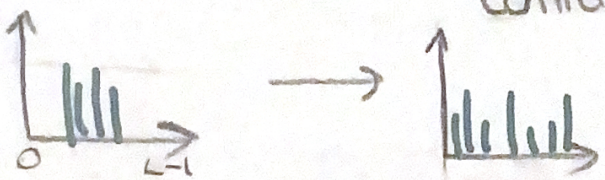
- $g(x,y)$ ↙ output image
- $T$ ↓ a func.
- $f(x,y)$ ↘ input image




**Contrast Stretching** ↘ Increasing Contrast

**Thresholding Function** ↙




## Basic Transformations

### → Image Negatives

$s = L - 1 - r$ (255)

| | |
|---|---|
| 10 | 200 |
| 200 | 5 |

f

Neg. →

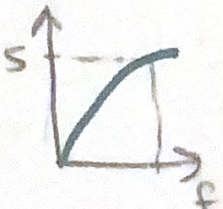
| | |
|---|---|
| 245 | 55 |
| 55 | 250 |

s

### → Log Transformation

Increases the brightness

$s = c \log(1+r)$

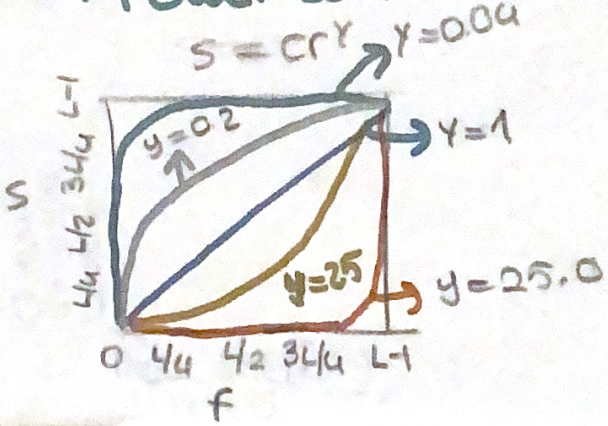



Expand the values of dark pixels in the image & compress higher level values.

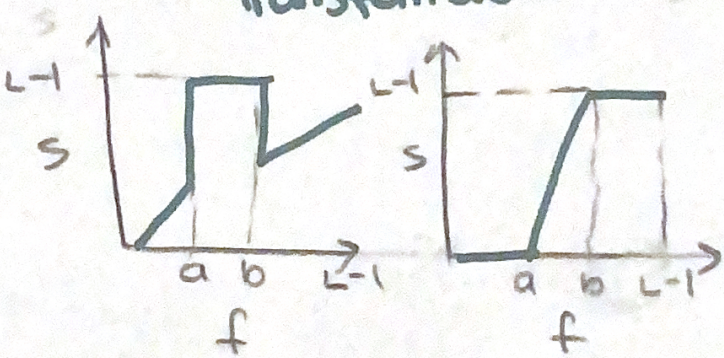
e.g. $[0, L/4] \rightarrow [0, 3L/4]$

### → Power Law (Gamma) Tr.

$s = cr^{\gamma}$




### → Piecewise Linear Transformation



# Histogram Matching

Modifying one image based on the contrast of another one.

1. First equalize the histogram of both images $((L-1)\sum_{j=0}^{k} Pr(r_j))$
2. Map each pixel of image A to B. $G(z_q) = s_k$
3. Modify A according to B. $z_q = G^{-1}(s_k)$

ex//

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 1 | 3 | 4 |
| 1 | 3 | 4 | 3 | 3 |
| 0 | 1 | 3 | 1 | 4 |
| 3 | 1 | 4 | 2 | 0 |
| 0 | 4 | 2 | 4 | 4 |

Image A

| Pixel Value | freq Histogram | Equalized Hist (cdf) |
|---|---|---|
| 0 | 4 | 4 |
| 1 | 5 | 9 |
| 2 | 3 | 12 |
| 3 | 6 | 18 |
| 4 | 7 | 25 |

| | | | | |
|---|---|---|---|---|
| 2 | 1 | 2 | 1 | 0 |
| 3 | 3 | 2 | 4 | 4 |
| 1 | 3 | 2 | 4 | 4 |
| 0 | 0 | 3 | 2 | 1 |
| 1 | 3 | 1 | 4 | 0 |

Image B (Target)

| | | |
|---|---|---|
| 0 | 4 | 4 |
| 1 | 6 | 10 |
| 2 | 5 | 15 |
| 3 | 5 | 20 |
| 4 | 5 | 25 |

Map each pixel in A based on it's equalized histogram, to the value of B.

| A | eq.A | | | |
|---|---|---|---|---|
| 0 | 4 | → look at what 4 corresponds in B (find the closest value) | 4 | 0 |
| 1 | 9 | | 10 | 1 |
| 2 | 12 | | 10 | 1 |
| 3 | 18 | | 15 | 2 |
| 4 | 25 | | 25 | 4 |

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 4 |
| 1 | 2 | 4 | 2 | 2 |
| 0 | 1 | 2 | 1 | 4 |
| 2 | 1 | 4 | 1 | 0 |
| 0 | 4 | 1 | 4 | 4 |

Modified A

* Changes:

0 → 0
1 → 1
2 → 1
3 → 2
4 → 4

# Fourier Transform

fftshift → 

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

→

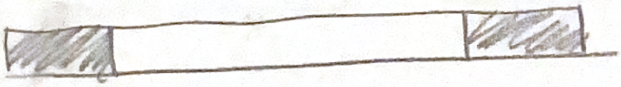
| | |
|---|---|
| 4 | 3 |
| 2 | 1 |

↓ we want to put zeros in the middle

↳ what does zero freq mean? zero change.
High freq → high change.

Smooth by removing high frequency. (details of the image)
If we want to remove details (smooth) we remove high frequencies.

5 4 3 2 0 1 2 4 9 → Remove frequencies > 3
↓ Multiply with [0 0 1 1 1 1 1 0 0]
0 0 3 2 0 1 2 0 0



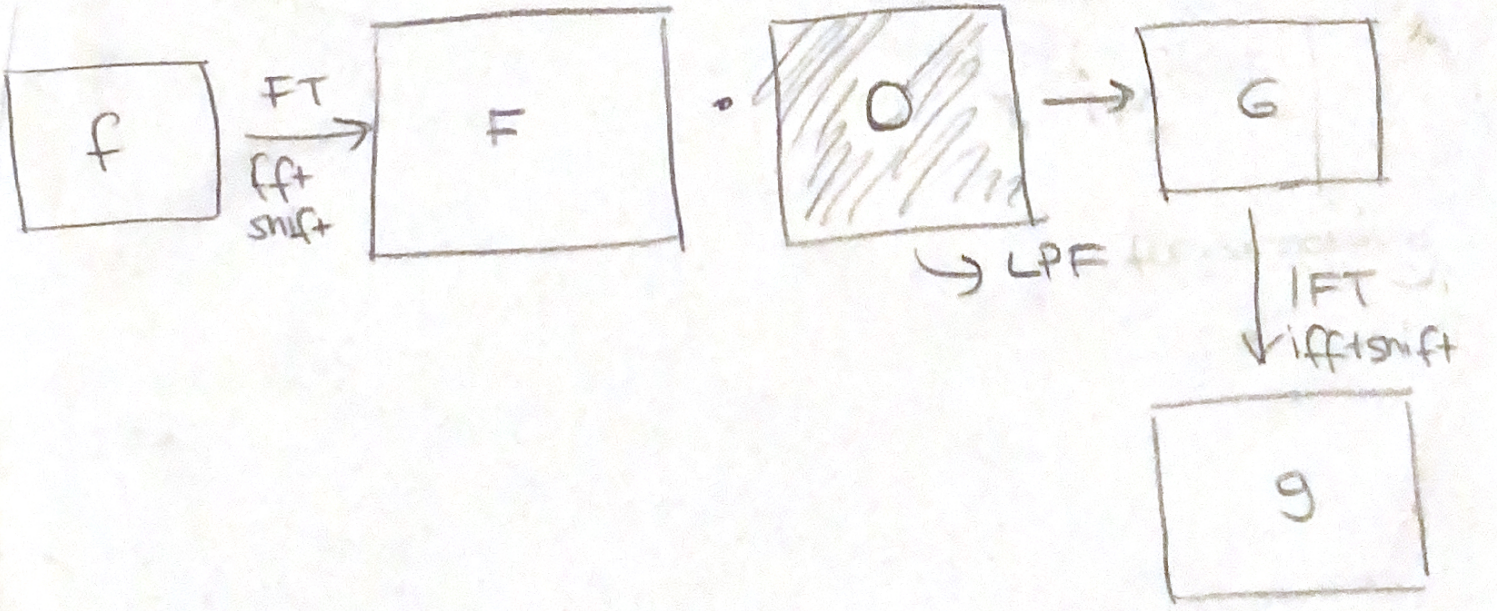
With convolution theorem $f * h \leftrightarrow F \cdot H$

spatial domain — freq domain

↳ with FT we can do filtering in freq domain (better for isolating frequencies)

$f \xrightarrow{FT} F \xrightarrow{F \cdot H} G \xrightarrow{IFT} g$

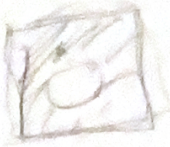
In 2D if we have



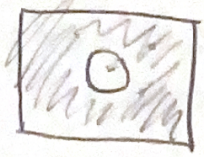
img $\xrightarrow{\text{fft2}}$ $\xrightarrow{\text{fftshift}}$ F $\longrightarrow$ F • H $\xrightarrow[\text{shift}]{\text{ifft}}$ $\xrightarrow{\text{ifft}}$ g
f

dot product

Smaller the white circle more blur (in LPF)



→ Ideal Filter

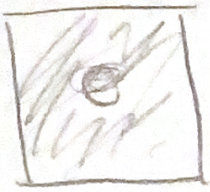


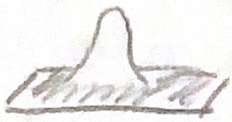

$H(u,v)=1$ if $D(u,v)<D_0$

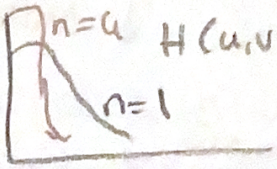
$H(u,v)=0$ if $D(u,v)>D_0$

$D(u,v)=\sqrt{(u-M/2)^2+(v-N/2)^2}$

→ Butterworth





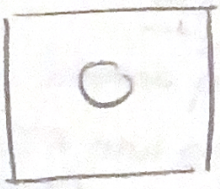


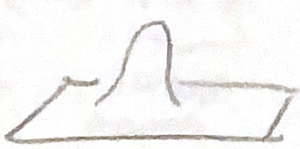

$$H(u,v)=\frac{1}{(1+D(u,v)/D_0)^{2n}}$$

kernel

order

Gauss

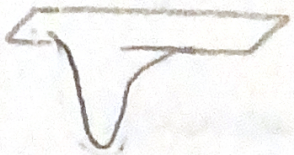




$$D(u,v)=\exp\left(\frac{-D(u,v)^2}{2D_0^2}\right)$$

$D_0$ ↓ cutoff freq

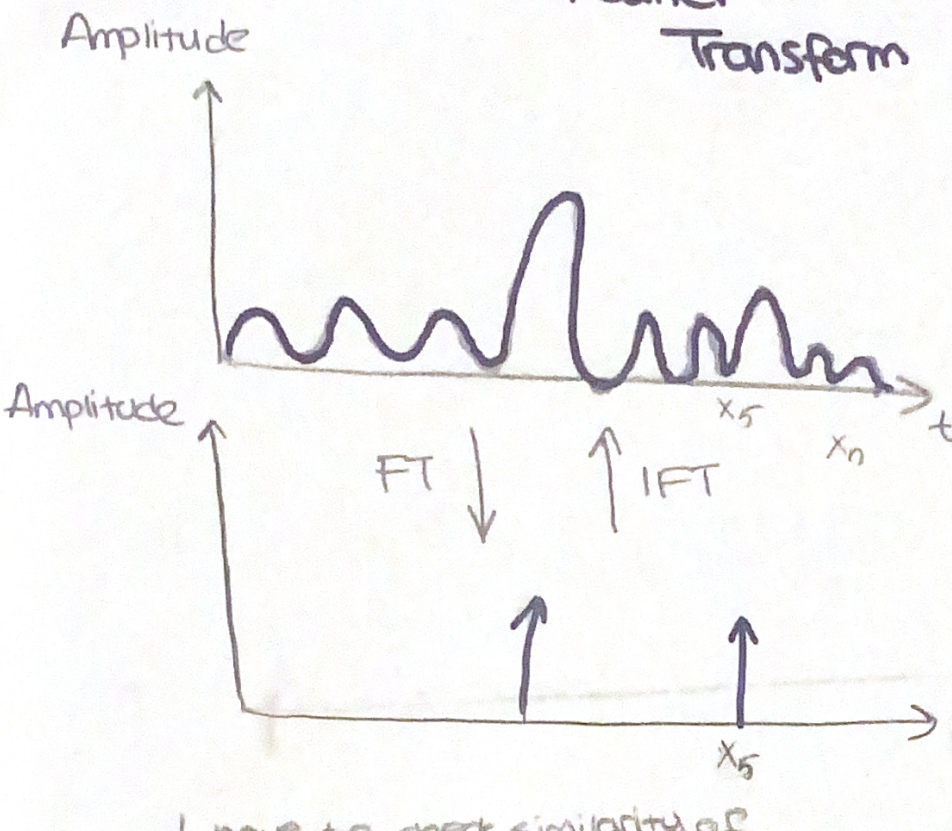
HPF = 1 − LPF →



$D_0$ ↗ HPF edge'leri kalır

$D_0$ ↙ LPF blur ↗

# Fourier Transform




$x_n : x_0, x_1, \dots x_{N-1}$

| $x_0$ | $x_1$ | ... | $x_{N-1}$ |
|---|---|---|---|

↓ find coordinates

- Euler formula

→ $e^{-ix} = \cos x - i \sin x$

Based on this we calculate bins.

$e^{-i 2\pi k n / N}$ } our sinusoidal coordinates (frequencies)

($N$ ↓ # bins)

I have to check similarity of every coordinate (for different $k$s) with my signal.


} Big similarity


} Small similarity

FT for one bin only → $X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n/N}$ ↗ size of signal

We need 2 for loops to calculate all $k$'s

```
* def myDFT(x_n):
      N = len(x_n)
      for k=0:N-1
          x_k = 0
          for n=0:N-1
              x_k = x_k + x_n[n+1] * exp(-2*i*pi*n*k/N)
                          ↑ element (signal)
          output[k+1] = x_k
```

We'll calculate this for:

| | n=0 | n=1 | n=2 | n=3 |
|---|---|---|---|---|
| k=0 | 00 | 01 | 02 | 03 |
| k=1 | 10 | 11 | 12 | 13 |
| k=2 | 20 | 21 | 22 | 23 |
| k=3 | 30 | 31 | 32 | 33 |

→ FFT reduces complexity from $n^2$ to $n \log n$.

Discrete Fourier Transform in 2D

1D → k n N x X

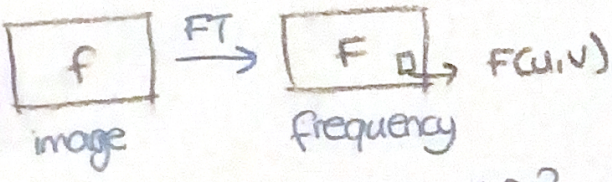
2D → u,v x,y M,N f F

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-i 2\pi \left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

- $F(u,v)$ → pixel of the frequency transformed image
- $f(x,y)$ → original signal

→ u & v are the frequencies

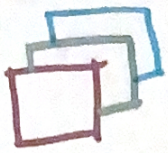
→ A frequency in an image corresponds to change in an image




How to calculate fourier in 2D?

```
def myDFT2D(f)
    for u=0:M-1
        for v=0:N-1
            for x=0:M-1
                for y=0:N-1
                output(u+1, v+1) = output(u+1,v+1) +
                    f(x+1, y+1) * exp(-2*i*pi*(x*u/M +
                    y*v/N)
```

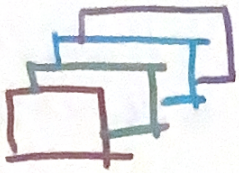
# COLORS



Color Image RGB

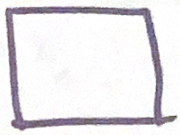
Each layer is 8 bits/px

3 layers = 24 px



RGBA Image with Alpha (transparency)

4 layers = 32 px



Gray Level Image

8 bits/px

Binary Image

1 bit/px

1 byte = 8 bits

1 MB = 1024 x 1024 byte

1 MB = 1024 kilobyte

ex// RGB Image → 8 bits/pixel = 1 byte

600×480

Size(MB) = $\frac{3 \text{ bytes} \times 600 \times 480}{1024} / 1024$

(3 ↗ channel; 1024 ↳ Byte→KB; 1024 ↳ KB→MB)

ex// Binary Image

600×400

1 bit/px

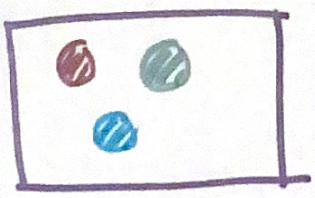
$\frac{600 \times 400 \times 1}{8} / 1024 / 1024$

(8 ↳ Bit→Byte; 1024 ↗ Byte→KB; 1024 ↗ KB→MB)

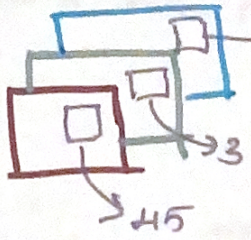
ex// HD Image (1920×1020)

RGB

→ 3 bytes × 1020 × 1080 / 1024 / 1024



Pixel

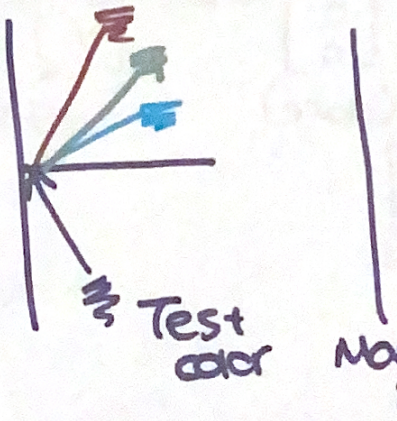



= we need a palette to answer!

390 → B

710 → R

## Colorimeter




Observer

UV VS IR

To define 3D perceptual space, observers match color of a given wavelenght → Lambda by mixing three other pure wavelenghts

phosphours of color TVs & other CRTs do not emit pure R, G or B light of a single wavelenght.

# CIE Chromasity Diagram

3D → 2D color space

$x + y + z = 1$

weights

$$x = \frac{x}{x+y+z} \qquad z = 1 - x - y$$

$$y = \frac{y}{x+y+z}$$

## Color Models

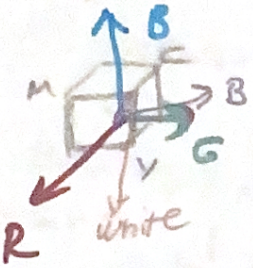
* RGB
  color monitors
  cameras
  Emitted light

* CMYK
  (cyan magenta yellow black)
  printers
  Reflected light

* HSI
  (Hue, saturation, illuminance)
  for human

## RGB Color Model

Based on a cartesian system




R: 0-255 1byte
G: 0-255 1byte
B: 0-255 1byte

Emitted light

ex

```
>> I = zeros(100,100,3)
>> I(:,:,1) = 255 → only red image
>> I(:,:,2) = 255 → made it yellow
>> I(:,:,3) = 255 → made it white
```

the color layer itself is white

## CMYK Color Model

Expresses reflected light

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

K = Black

if $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ (black) then $\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ black

if $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ white then $\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ white

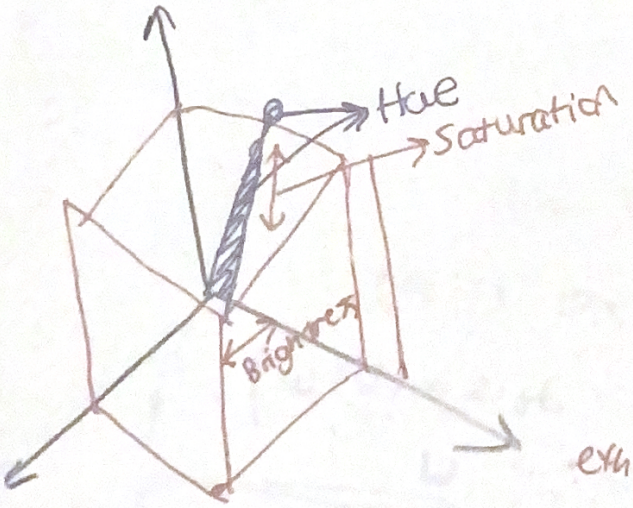
if $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ then $\begin{bmatrix} C \\ M \\ Y \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$

# HSI/HSV Color Model

Suitable for humans.

* Hue → Pure color (RGB CMYK)
* Saturation → Measure of purity, degree of which a pure color is dilated by white light
* Luminance / Value / Intensity → Brilliance of color

## HSI Geometric Representation




→ Choose color, then saturation then brightness.

```
Ihsv = rgb2hsv(I)
Ihsv(:,:,1) (hue)
  → colors, red is at the end
[m n c] = size(I)
ex for i=1:m
    for j=1:n
    if(Ihsv(i,j,1)<0.95
        I(i,j,1)=0;
```

ex1

```
if Ihsv(i,j,1)<0.90:
I(i,j,:) = (I(i,j,1)+I(i,j,2)+I(i,j,3))/3
```

→ convert to grayscale (everything but red)

ex1

```
Ihsv(:,:,3) = Ihsv(:,:,3)-0.5 → becomes darker
Inew = hsv2rgb(Ihsv)
```

(3 → brightness)

# Medical Imaging

## Medical Imaging Devices

Computed Tomography Scanner

⇉ Xray Scanner
⇉ Magnetic Resonance Impedance Scanner (MRI)

PET Scanner

Ultrasound devices

---

### X-Rays

- Short wavelength electromagnetic waves
- Wavelength is between $10^{-1}$–10 nm
- Xray photons have a lot of energy
- It is dangerous → can produce chemical damage on molecules
  ↳ knock particles out of atoms

High freq → high energy → heat

Image is produced based on absorption of x-rays to different tissues / bones of the body.

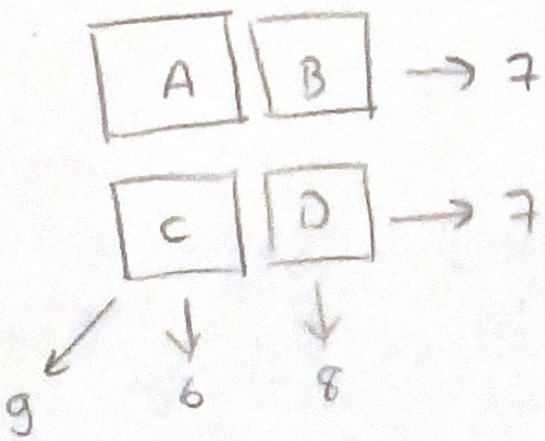
### Computed Tomography

Allows us to see what's inside of human body without dissecting it.

It allows us to produce 2D cross sections of an object using X-rays.

→ How the image is produced by multiple projections!

We have horizontal, vertical & diagonal sums of following image




A + B = 7
A + C = 6
...

Sums simulate x-ray absorptions of the image.

## MRI CT Scanner

→ In CT Scan (an X-ray) images are not clear because they're based on absorption (It is good for bones, sometimes tissue)

→ MRI is based on protons

→ Into the body they have arbitrary positions

→ When we apply a strong magnetic field, protons are aligned with magnetic field, when radiofrequency pulse is applied, some of them gets activated. When pulse ends, activation ends & relaxation time T1 is measured.

Because protons belong to $H_2O$ (hydrogen) it works better in tissues

## Ultrasound Devices

Subsound ← | → Ultrasound

→ Ultrasound is a sound we can't hear. (Audible → 20-18000 Hz)

It's greater than 20KHz. In medical imaging it's 1-15 MHz.

Lower Frequency ↓ Better penetration ↗ Less resolution (3 MHz)

Higher Frequency ↗ Worst penetration ↓ Higher resolution (12 MHz)

(we don't miss tissues

→ Development of fetus
→ Evaluation of blood flow } can be seen (using doppler effect)
→ Detection of tumors

# Medical Imaging

## Formats & Algorithms

### Dicom

→ Digital imaging communication in medicine (an image standard)
Has header & data part. Header → info about patient & scanner

### Hounsfield Unit

A unit describe the attenuation of X-rays inside the human body.

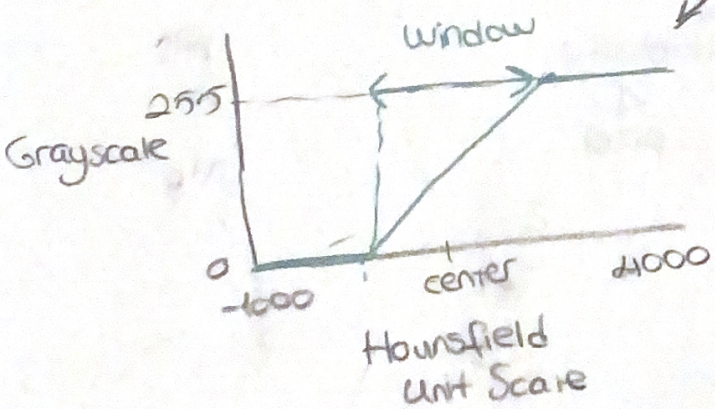
$$HU = \frac{\mu_x - \mu_{H_2O}}{\mu_{H_2O}} \times 1000$$

$\mu$ attenuation coefficient related to radiodensity of material → amount of transparency to the passage of x-rays through material.

$HU_{H_2O} = 0$ $\mu_{air} = 0 \rightarrow HU_{air} = -1000$

HU is used in CT scanners.
In DICOM we store data in a range larger than 0-255 how we can display a wider range of values → Map for region we would like to display

Window level function




### Volume Rendering

Technique for 3D visualization

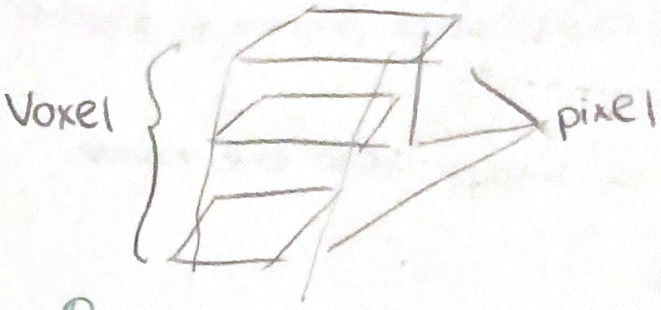
3D vit is hard due to → unclear borders between organs
→ unclear elements inside organs

Instead of trying to separate structure display all of them based on intensity values as 3D image

→ At beginning we have 2D CT scans.

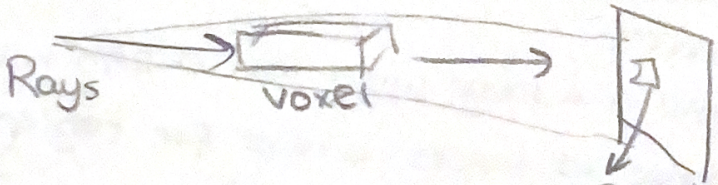
→ Then we define a voxel (volume element) → 3D pixel created from a pixel of an image belonging to a computed tomography scan.

We want opacity & color




## Ray Casting

- Popular method to visualize voxel models
- Ray goes to 3D grid & resulted pixels take into account all the opacities & color of voxels it has found.




(Bones are very opaque)

$$C_{out} = C_{in}(1 - a(x_i)) + c(x_i)\,a(x_i)$$

- $C_{out}$: outgoing intensity
- $C_{in}$: ingoing intensity
- $a(x_i)$: opacity of pixel
- $c(x_i)$: intensity of pixel