EX.:

5 iulie 2017

(5.) Compute the filtered img. by applying a treshold average filter (3x3), T = 20.

8 biti/pixel 4x4

50	10	250	125
20	30	75	100
5	100	50	15
100	10	30	160

⇓

50	10	250	125
20	68	75	100
5	40	50	15
100	10	30	160

[30] → $Av = \frac{50+10+250+75+50+100+5+20+50}{9} = 67,77$

$|30 - 67,77| = 37,7 > 20$

[75] → $Av = 75,5$

$|75 - 75,5| < 20$

[100] → $Av = 40$

$|100 - 40| > 20$

[50] → $Av = 65$

$|50 - 65| < 20$

(6.) Det. the useful width of a 2D convolutional Gaussian filter and give the formulas for computing its kernel, given $\sigma = 1.16$

(7.) Canny edge detection alg.

Criteria: 1. Optimal detection: the prob. of marking a real edge point should be high and the prob. of marking a false edge point should be low. (maximize SNR)

2. Correct localization: the points marked as belonging to an edge should be as close as possible to the real edge points

3. Single response to one edge

? ⑧ Img. w/ circles → alg. for automatically finding the approx diameter of each circular object

1) Filter the img. s.t. all small irrelevant points dissappear (ex. treshold average filter)
2) An alg. for edge detection, exctraction and closing
3) When closing occurs compute length of circle $2\pi R \Rightarrow$ diameter $= \frac{l_{circle}}{\pi}$

31 aug. 2015

⑤ Methods for SNR computation from digital images.

→ Signal & noise variances: $\sigma_s^2 = \langle | s(i,j) - \langle s(i,j) \rangle |^2 \rangle$

$\sigma_n^2 = \langle | n(i,j) |^2 \rangle$

$SNR = \frac{\sigma_s}{\sigma_n}$

$\sigma_f^2 = \sigma_s^2 + \sigma_n^2 \Rightarrow SNR = \sqrt{\frac{\sigma_f^2}{\sigma_n^2} - 1}$

→ To calculate w/ 1 image: need 2 of σ_n, σ_s or σ_f

↳ from image $\Rightarrow \sigma_f$

↳ if img. has regions w/ no signal (sky/water) → estimate σ_n (just noise)

→ To calculate w/ 2 images of the same scene:

$f(i,j) = s(i,j) + n(i,j)$

$g(i,j) = s(i,j) + n(i,j)$

$\langle n(i,j) \rangle = \langle n(i,j) \rangle = 0$ → the noise uncorellated $\langle n(i,j)\, n(i,j) \rangle = 0$

the noise uncorellated w/ the signal

$\langle s(i,j)\, n(i,j) \rangle = \langle s(i,j)\, n(i,j) \rangle = 0$

$\rho = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2} \Rightarrow SNR = \sqrt{\frac{\rho}{1-\rho}}$

$SNR > 20$ → little variable noise

$SNR \simeq 10$ → some noise

$SNR \simeq 4$ → noise

$SNR \simeq 2,1$ → image??

(6) Linear digital filtering: Fourier space filtering vs. real space filtering

Filtering { image space: by convolution (directly processing the input pixel array)
Fourier space: by multiplication: $F(u,v) \times H(u,v) = Y(u,v)$
(Fourier transform, process, inverse Fourier)

Linear => a single output = convolution
$g(i,j) = h(i,j) \odot f(i,j)$

a) Fourier space: apply Fourier on f & h, multiply, apply inverse Fourier
→ $g(i,j) = \mathcal{F}^{-1}\{F(k,l)\,H(k,l)\}$
→ floating point format
→ computational cost is not dependent on filter function $h(k,l)$
→ cannot be extended to non-linear operations
→ exs.: high-pass, low-pass, band-pass

b) Real space:
→ direct approximation of shift and multiply
→ $g(i,j) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}-1} \sum_{n=-\frac{M}{2}}^{\frac{M}{2}-1} h(m,n)\, f(i-m, j-n)$
↳ filter function of size $M \times M$
→ integer format
→ computational cost directly related to the filter size
(3x3 filter => 9 multipliers & 9 adds)
→ exs.: averaging, differentiating

? (7) 8 bits/pixel 4x4; compute the filtered img. by applying a convolution w/ a Laplacian filter (3x3).

50	10	250	20
20	40	25	100
5	100	50	5
200	10	25	150

$\frac{1}{8} *$

0	1	0
1	-4	1
0	1	0

⇓

50	10	250	20
20	-5	340	100
5	-295	-45	5
200	10	25	150

1: 100 - 200 + 10 - 20 = 130 - 200 = -70
2: 155 - 160 = -5
3: 440 - 100 = 340
200 - 155 = -45

1: 155 - 160 = -5
2 440 - 100 = 340
3: 105 - 400 = -295
4: 155 - 200 = -45

(8) Det. the useful width of a 2D convolutional LoG filter ($LoG = \frac{x^2+y^2-2\sigma^2}{\sigma^4} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$) and give the formulas for computing its kernel elements given $\sigma = \frac{1}{2\sqrt{2}}$

↳ σ = std. deviation ; determines scale of filter

↳ width: $w = \sigma \cdot 2\sqrt{2}$

↳ useful width: $\Delta = 3w$

↳ compute kernel: $G(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}$

(x_0, y_0) - coordinates of central row and column of kernel

sub. B

(5.) SNR = ? ; variance of the img. pixels = 100, standard dev. of noise = $2\sqrt{5}$; img. quality by analyzing SNR?

$\sigma_g^2 = 100$ ↳ variance

$\sigma_n = 2\sqrt{5} \Rightarrow \sigma_n^2 = 20$ ↳ standard deviation

$\Rightarrow SNR = \sqrt{\frac{\sigma_g^2}{\sigma_n^2} - 1} = \sqrt{\frac{100}{20} - 1} = \sqrt{4} = 2$

SNR = 2 ⇒ img. severely degraded

(6.) High pass filters in freq. domain

→ attenuates (or removes) low spatial freq. while allowing high spatial freq. to pass

→ effect: enhancing the edges in images

→ note that it also enhances noise

→ types:
- ideal: $H(k,l) = \begin{cases} 0, & \text{for } k^2+l^2 \leq w_0^2 \\ 1, & \text{else} \end{cases}$, creates ringing
- smooth cut-off: $H(k,l) = 1 - \exp\left(-\frac{w}{w_0}\right)^2$, ringing is minimized

7. a) Real space linear filters:
↳ finite range of filtering mask
↳ type: real space averaging

0	1	0
1	1	1
0	1	0

1	1	1
1	1	1
1	1	1

real space differentiation: x diff → enhances vertical edges; y diff → horizontal edges

b) 8 bits/pixel; compute filtered img. w/ mean filter

50	10	250	50
20	30	25	100
5	100	50	15
100	10	20	100

→ median

	30	50	
	25	70	

mean

50	10	250	50
20	60	70	100
5	40	50	15
100	10	20	100

1. $\frac{50+10+250+25+50+100+5+20+30}{9} = 60$
2. $\frac{10+250+50+100+15+50+100+30+25}{9} = 70$
3. $\frac{360}{9} = 40$
4. $\frac{450}{9} = 50$

5, 10, 20, 25, 30, 50, 50, 100, 250
10, 15, 25, 30, 50, 50, 100, 100, 250
5, 10, 20, 20, 25, 30, 50, 100, 100
10, 15, 20, 25, 30, 50, 100, 100, 100

8. a) Edge extraction based on 1st derivatives
↳ we can detect an edge by a simple treshold of: $\underbrace{\left|\frac{\partial f(x)}{\partial x}\right| > T \Rightarrow \text{Edge}}_{1D}$

2D: $\frac{\partial f(x,y)}{\partial x}$ → vertical edge

$\frac{\partial f(x,y)}{\partial y}$ → horizontal edge

↳ for all directions: $\frac{\partial f(x,y)}{\partial x} \cdot \cos\Theta + \frac{\partial f(x,y)}{\partial y} \cdot \sin\Theta$ (Θ - direction)

→ treshold to obtain edges: $|\nabla f(x,y)| > T \Rightarrow$ Edge
$|\nabla f(x,y)| < T \Rightarrow$ no Edge

↳ for digital implement., convolution of img w/ 2 Kernels:

$$A = \frac{\partial f(i,j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \circ f(i,j)$$

$$B = \frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \circ f(i,j)$$

$$|\nabla f(i,j)| = \sqrt{A^2 + B^2}$$

↳ gradient = $|A| + |B|$ → faster approximation

6) Compute a 5×5 mixed convolution kernel (Gaussian filter 3×3 + Sobel derivative kernel 3×3) that can be used for directly computing the 1st derivative of a smoothed img. on horizontal direction

Sobel: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ & $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

↳ vertical ↳ horizontal

Gaussian: low pass, high pass etc.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -3 & -3 & 0 & 3 & 3 \\ -4 & -4 & 0 & 4 & 4 \\ -3 & -3 & 0 & 3 & 3 \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix} \rightarrow \text{vertical}$$

$$\Rightarrow \text{horizontal} \rightarrow \begin{bmatrix} -1 & -3 & -4 & -3 & -1 \\ -1 & -3 & -4 & -3 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 4 & 3 & 1 \\ 1 & 3 & 4 & 3 & 1 \end{bmatrix}$$

sub. A

5) 2 img. f, g on the same scene, taken at different time, are affected by independent additive noise. SNR = ?, knowing that the normalized correlation between f and g is $r = \frac{100}{101}$; what can you say about the quality of img.?

$$SNR = \sqrt{\frac{r}{1-r}} = \sqrt{\frac{100}{101} \cdot \frac{1}{101 - \frac{100}{101}}} = \sqrt{\frac{100}{101} \cdot \frac{101}{1}} = \sqrt{100} = 10$$

$\Rightarrow SNR = 10 \Rightarrow$ some visible noise

(6) Low pass filters in frequency domain

→ low spatial freq. pass unattenuated, high ones are attenuated or completely blocked

→ can be used to reduce the effect of random noise (which is associated w/ high spatial freq.), but this results in removal of img. info.

→ applied as a multiplication in Fourier space, appears as convolution in real space

→ types: a) ideal : $H(k,l) = \begin{cases} 1, & \text{for } k^2+l^2 \le w_0^2 \\ 0, & \text{else} \end{cases}$ ⇒ ringing

b) smooth cut-off : $H(k,l) = \exp\left(-\frac{w}{w_0}\right)^2$

c) Butterworth : $H(k,l) = \frac{1}{1+\left(\frac{w}{w_0}\right)^n}$

(7.) a) Real space non-linear filters:

↳ in real space convolution filtering is defined by shift and multiply

↳ the diff. from linear filtering involves the replacement of summations by the specified non-linear operator

↳ the characteristics of the filtering operations are now determined by both the mask $h(i,j)$ and the operator

↳ types: a) Shrink and Expand } combined: remove all bright regions smaller than the filter size
op = min op = max

b) Treshold average filters:

↳ do average of neighboring pixels; if center pixels differ from average by more than treshold, replace it w/ average

c) Median filter: replace center pixel w/ median (middle value in the ordered array of 9 pixels)

d) Homomorphic filter:

b) Median filter (8 bits/pixel 4x4 pixels)

50	10	250	50
20	30	25	100
5	100	50	15
100	10	20	100

5, 10, 20, 25, (30) 50, 50, 100, 250

10, 15, 25, 30, (50), 50, 100, 100, 250

5, 10, 20, 20, (25), 30, 50, 100, 100

10, 15, 20, 25, (30), 50, 100, 100, 100

↳

⇒

(8) a) Edge extraction based on 2nd order derivatives
↳ by computing the Laplacian: $\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$
↳ it can be implemented by a single convolution of:

$$\nabla^2 f(x,y) = \frac{1}{8} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \circ f(i,j)$$

↳ it is recommended to smooth the img. before applying Laplacian
↳ problems: thin edges, closed loops, noise

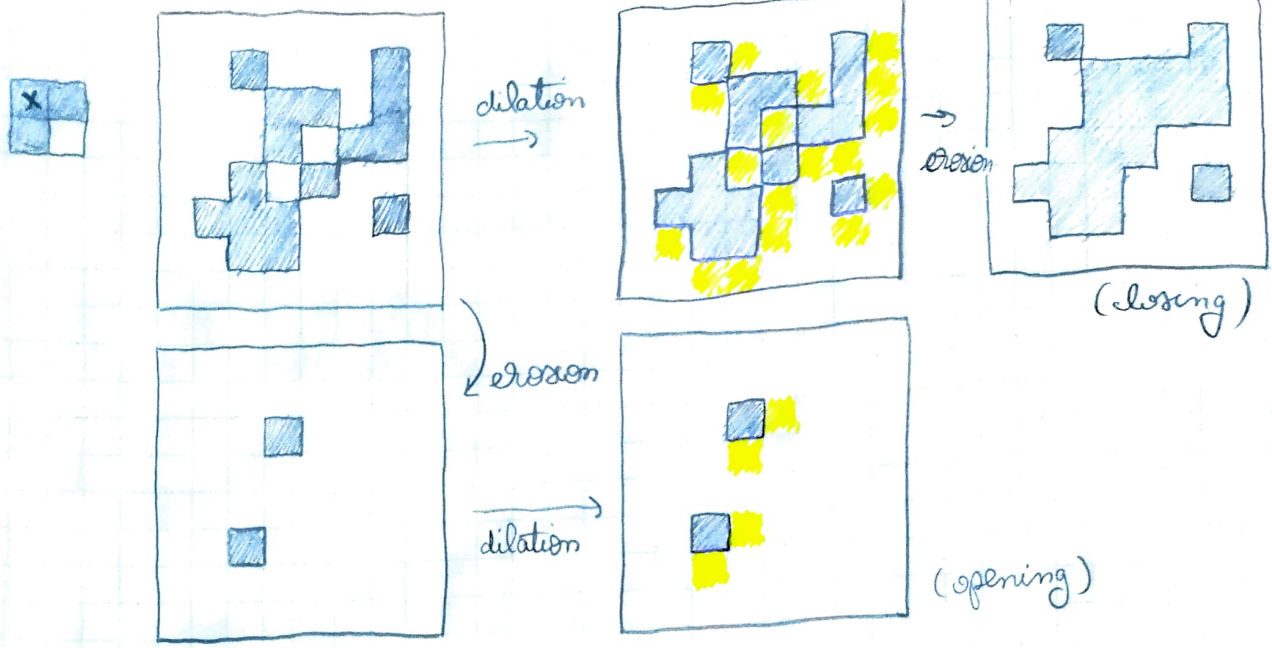
b) Compute a 5x5 mixed convolution kernel (Mean filter 3x3 + Laplace 3x3) that can be used for directly computing the 2nd derivative of a smoothed img.

Mean filter: $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Laplace filter: $\frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

$$\frac{1}{72}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & -2 & -1 & -2 & 1 \\ 1 & -1 & 0 & -1 & 1 \\ 1 & -2 & -1 & -2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

(*)

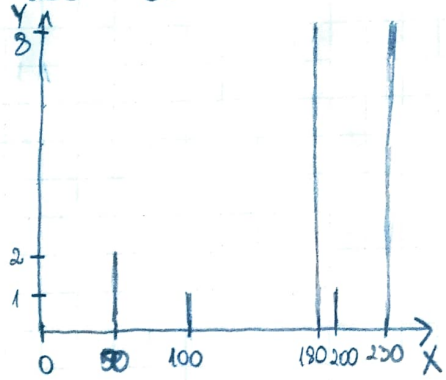


(*) 4 x 5

180	230	50	230	230
100	180	180	180	230
230	180	180	180	180
230	230	200	230	50

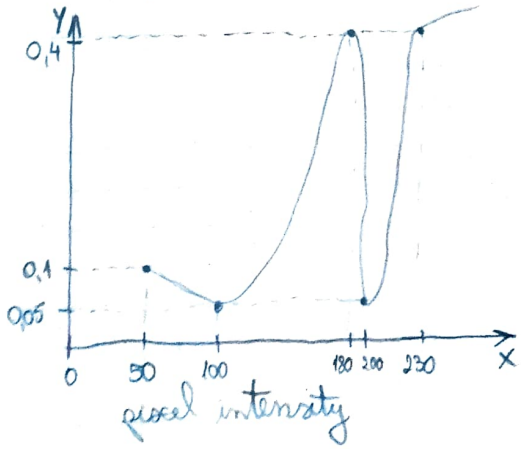
1+1+2+8+8

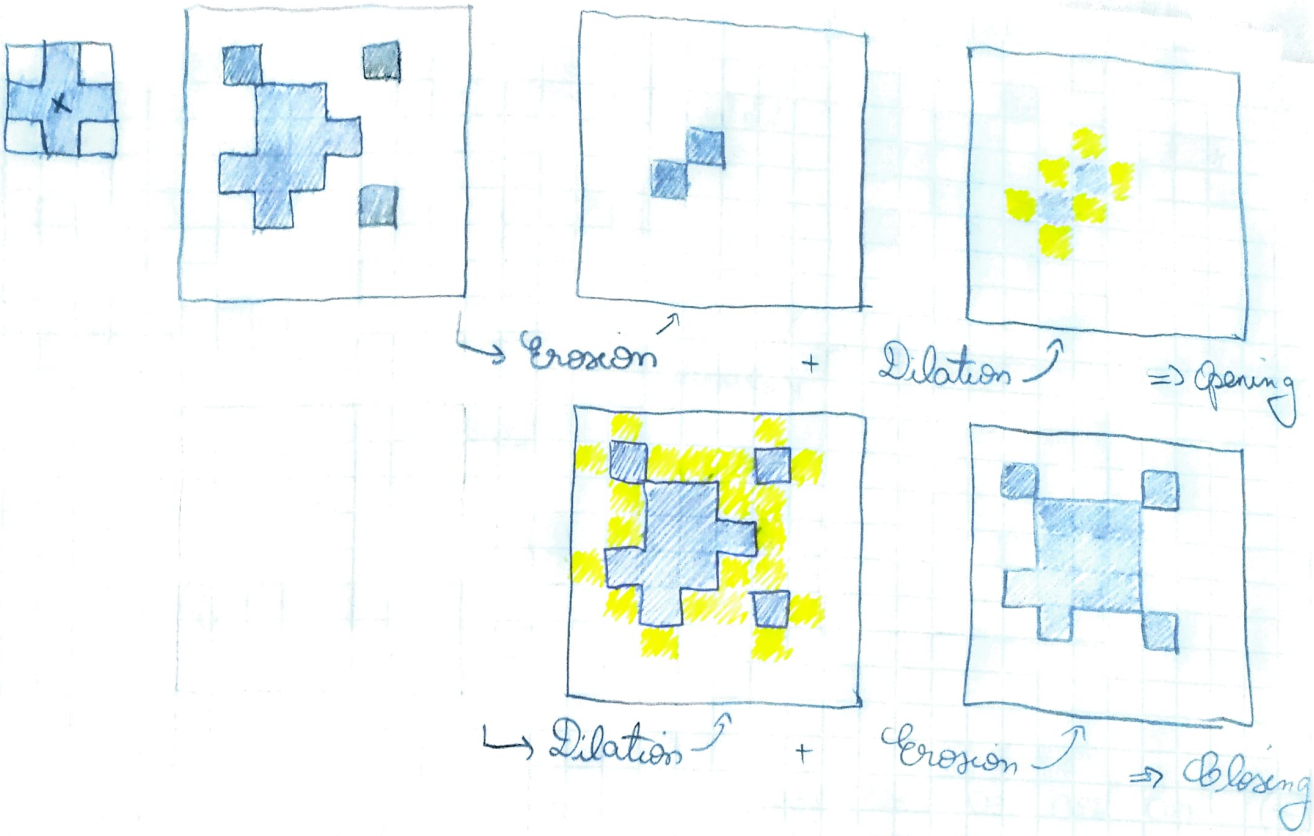
a)



$PDF = \frac{h(g)}{M}$, $M = 4 * 5 = 20$

$\frac{1}{20} = 0,05$ $\frac{2}{20} = 0,1$ $\frac{8}{20} = 0,4$



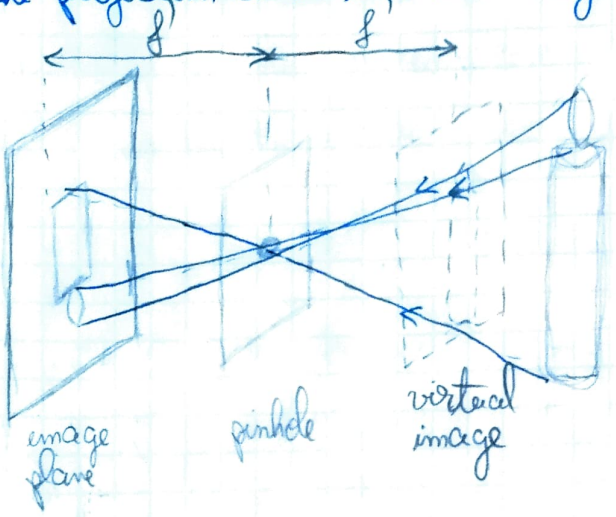

x
↳ Erosion
+
Dilation
⇒ Opening
↳ Dilation
+
Erosion
⇒ Closing

REZOLVĂRI

1. Camera models (pin hole, weak perspective, thin lens)

a) Pin hole (perspective camera model):

↳ most common
↳ each point in the object space is projected by a straight line through the projection center into the image plane



$$\begin{cases} x = f \cdot \frac{x_c}{z_c} \\ y = f \cdot \frac{y_c}{z_c} \end{cases}$$

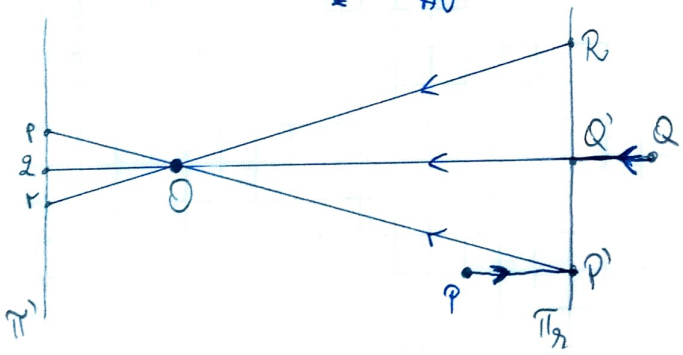
f - focal length

x_c, y_c, z_c - coord. of scene point

b) Weak perspective:

↳ closer objects appear bigger in projection and vice-versa
↳ the relative distance along the optical axis, between 2 scene points is much smaller than the average distance z_{AV} between the camera and these points

$$\delta_z < z_{AV}/20$$



$$\begin{cases} x' = f' \frac{x}{z} \approx \frac{f'}{z_{AV}} \cdot x \\ y' = f' \frac{y}{z} \approx \frac{f'}{z_{AV}} \cdot y \end{cases}$$

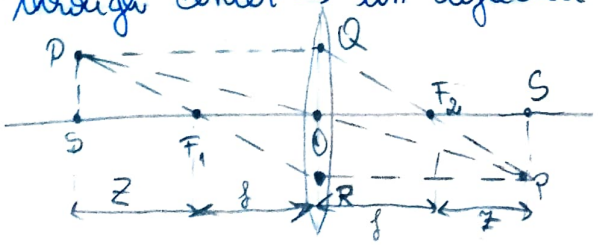
$x' = -m \cdot x$

$y' = -m \cdot y$

$m = -\frac{f'}{z_{AV}}$ → magnification

c) Thin lens camera model:

Any ray entering the lens
↳ parallel ⇒ through focus
↳ through focus ⇒ parallel
↳ through center ⇒ un-deflected



$$z \cdot Z = f^2$$

$$\frac{1}{Z+f} + \frac{1}{z+f} = \frac{1}{f}$$

$$f = \frac{R}{2(n-1)}$$

2. a) Pinhole camera model w/ the following propr.: focal length $f = 35$ px; img. resolution 1920×1080. Compute the principal point (u_0, v_0) knowing that $P(X, Y, Z) \to P(0\text{m}, 5\text{m}, 25\text{m})$ is projected into $p(u, v) \to p(1220\text{px}, 660\text{px})$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} f_x \cdot x_m + u_0 \\ f_y \cdot y_m + v_0 \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} u = f_x \cdot x_m + u_0 \\ v = f_y \cdot y_m + v_0 \end{cases} \Rightarrow \begin{cases} u_0 = 1220 - 800 \cdot \frac{x_c}{z_c} = 1220 - 800 \cdot \frac{10}{25} \\ v_0 = 660 - 800 \cdot \frac{y_c}{z_c} = 660 - 800 \cdot \frac{5}{25} \end{cases} \Rightarrow \begin{cases} u_0 = 900 \\ v_0 = 500 \end{cases}$$

b) $f = 60$ px; img. res 320×240 px; $P(X, Y, Z) \to P(X, Y, 20\text{ m})$

$p(u, v) \to p(310, 150)$

(u_0, v_0) - in the center of img. plane $\to (160, 120)$

$X, Y = ?$

$(u, v) \to (0, 0)$ top left corner

$(X, Y, Z) \to (0, 0, 0)$ origin $\to$ lens center

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} f_x \cdot x_m + u_0 \\ f_y \cdot y_m + v_0 \\ 1 \end{bmatrix}$$

$$\begin{pmatrix} Y \\ X \end{pmatrix} = f \begin{pmatrix} x_m \\ y_m \end{pmatrix} = f \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix} = f \begin{pmatrix} X/20 \\ Y/20 \end{pmatrix}$$

$$\Rightarrow \begin{cases} u = f_x \cdot x_m + u_0 \\ v = f_y \cdot y_m + v_0 \end{cases} \Rightarrow \begin{cases} 310 = 60 \cdot x_m + 160 \\ 150 = 60 \cdot y_m + 120 \end{cases} \Rightarrow \begin{cases} x_m = \frac{310 - 160}{60} = \frac{5}{2} \\ y_m = \frac{150 - 120}{60} = \frac{1}{2} \end{cases}$$

$f_x = f_y = f = 60$

$$\begin{cases} x_m = X/Z \\ y_m = Y/Z \end{cases} \Rightarrow \begin{cases} X = x_m \cdot Z = \frac{5}{2} \cdot 20 = 50 \\ Y = y_m \cdot Z = \frac{1}{2} \cdot 20 = 10 \end{cases} \Rightarrow P(50, 10, 20)$$

3. Physical camera parameters

a) Intrinsic parameters = internal camera geometrical and optical charact.:
→ focal length (f)
→ effective pixel size
→ principal point (u_0, v_0)
→ distortion coefficients of the lens: radial (K_1, K_2) and tangential (p_1, p_2)

b) Extrinsic parameters = the 3D position and orientation of the camera frame relative to a certain world coordinate syst.:
→ rotation vector $R = [R_x, R_y, R_z]^T$
→ translation vector $T = [T_x, T_y, T_z]^T$

4. Modeling the lens distortions

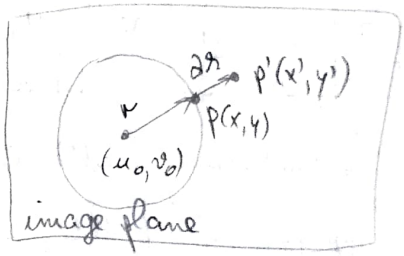
a) Radial lens distortion

↳ causes the actual image point to be displayed radially in the image plane

$$\begin{bmatrix} \partial x^r \\ \partial y^r \end{bmatrix} = \begin{bmatrix} x \cdot (K_1 \cdot r^2 + K_2 \cdot r^4 + \dots) \\ y \cdot (K_1 \cdot r^2 + K_2 \cdot r^4 + \dots) \end{bmatrix}$$

$r = x^2 + y^2$

$K_2, K_2, \dots$ - radial distortion coefficients



b) Tangential distortion

↳ appears if the centers of curvature of the lenses' surfaces are not strictly collinear

$$\begin{bmatrix} \partial x^t \\ \partial y^t \end{bmatrix} = \begin{bmatrix} 2p_1 \cdot xy + p_2 (r^2 + 2x^2) \\ p_1 (r^2 + 2y^2) + 2p_2 \cdot xy \end{bmatrix}$$

p_1, p_2 - tangential distortion coeff.

↳ transf. from metric units to img. coordinates

$p[x, y]^T \Rightarrow [u, v]^T$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} D_u \cdot (x + \partial x^r + \partial x^t) \\ D_v \cdot (y + \partial y^r + \partial y^t) \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

↳ the projection equations become non-linear!

Sol.: perform distortion correction on image and afterwards linear projection

5. Projection matrise

$$P = A \cdot [R_{wc} \mid T_{wc}]$$

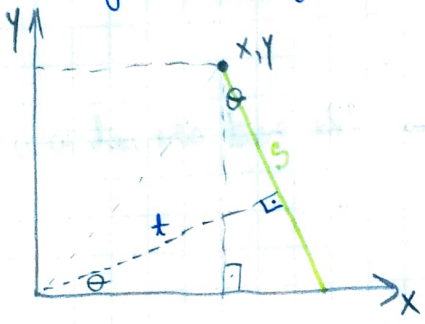
↳ the projection eq. of a 3D world point $[X_w, Y_w, Z_w]$ in normalized coord.:

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = P \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \qquad s = z_s \rightarrow \text{scaling factor}$$

↳ obtaining the 2D image coord. from normalized coord.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{x_s}{z_s} \\ \frac{y_s}{z_s} \end{bmatrix}$$

Projection – geom. propr.(?)



$$p_\theta(t) = \int b(t\cos\theta - s\sin\theta,\ t\sin\theta + s\cos\theta)\, ds$$

Vertical proj. ($\theta = 0$): $v(x) = \int b(x,y)\, dy$

Horizontal proj ($\theta = \pi/2$): $h(y) = \int b(x,y)\, dx$

Area: $\int v(x)\, dx = \int h(y)\, dy$

Center of the area: $\bar{x}A = \int x v(x)\, dx$

$\bar{y}A = \int y h(y)\, dy$

6. Tresholding

1) Global tresholding

→ $f(x,y)$ - source img., $b(x,y)$ - binary img.

$$b(x,y) = \begin{cases} 1, & f(x,y) \le T \\ 0, & \text{otherwise} \end{cases}$$

• Obj. $\in [T_1, T_2]$

$$b(x,y) = \begin{cases} 1, & T_1 \le f(x,y) \le T_2 \\ 0, & \text{otherwise} \end{cases}$$

• Obj. $\in Z$, Z is a set of intensity values

$$b(x,y) = \begin{cases} 1, & f(x,y) \in Z \\ 0, & \text{otherwise} \end{cases}$$

2) Semitresholding

→ pixels $\in$ a given treshold range retain their values; the rest are set to 0

$$b(x,y) = \begin{cases} f(x,y), & k_1 \le f(x,y) \le k_2 \\ 0, & \text{otherwise} \end{cases}$$

→ regions of high values can be isolated using:

$$b(x,y) = \begin{cases} f(x,y), & f(x,y) \ge k \\ 0, & \text{otherwise} \end{cases}$$

→ reg. of low values -II-:

$$b(x,y) = \begin{cases} f(x,y), & f(x,y) \le k \\ 0, & \text{otherwise} \end{cases}$$

3) Multilevel tresholding

→ allows for segmentation of pixels into multiple classes

→ $f(x,y)$ - source img., $K_1, \dots, K_m$ - treshold levels satisfying $K_1 < K_2 < \dots < K_m$

⇒ $m+1$ intervals assoc. w/ values $V_1, \dots, V_{m+1}$

$$b(x,y) = \begin{cases} V_1, & f(x,y) < K \\ V_i, & K_{i-1} \le f(x,y) < K_i \\ V_{m+1}, & K_m \le f(x,y) \end{cases}$$

4) Variable tresholding

→ allows different treshold levels to different regions of an img.

→ $f(x,y)$ - source img., $d(x,y)$ - local treshold value

$$b(x,y) = \begin{cases} 1, & f(x,y) \ge d(x,y) \\ 0, & f(x,y) < d(x,y) \end{cases}$$

5) Treshold selection using mean and standard deviation

→ the mean of $f(x,y)$: $\mu = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} f(i,j)$

→ the standard deviation is: $\sigma = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [f(i,j) - \mu]^2}$

→ the treshold level T: $T = k_1 \mu + k_2 \sigma$

↳ for low-res. img. $k_1 = k_2 = 1$

↳ for high-res. img. $k_1 = 1$ or 1.5, $k_2 = 2$

6) Treshold selection by best approximation with a two level image

→ img. f_{ij}, binary img. $g_{ij} = \begin{cases} a, & f_{ij} < t \\ b, & f_{ij} \geq t \end{cases}$

↳ a and b are constants chosen to minimize the distance on the corresp. intervals

7. Run-Length Coding

2 approaches
- the start positions and length of runs of 1s for each row are used
- lengths of runs, starting w/ the length of the 0 run

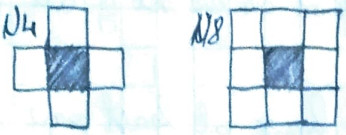
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	1	1	0	0	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Start and length of 1 runs: (1,3) (7,2) (12, 4) (17, 2) (20, 3)
(5, 13) (19,4)
(1,3) (17,6)

Length of 0 and 1 runs: 0,3,3, 2,3,4, 1,2,1,3
4,13,1,4
0,3,13,6

8. Binary Labeling Algorithms

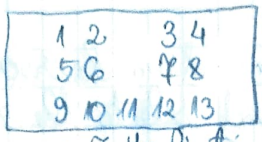
→ neighbours:
↳ 4-neighbours or d-neighbours if they share a common side
↳ i-neigh. if they share a common corner
↳ 8-neigh. if they share at least a corner



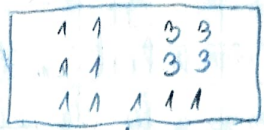
8.1 Sequential labeling alg.

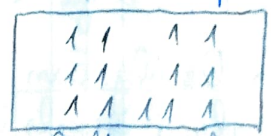
Exc:


1. Initial img.



2. Initialization

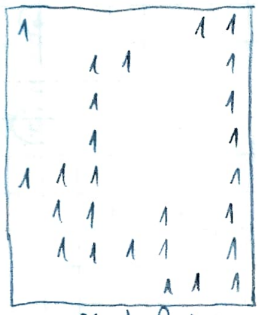

3. Top-down & left-right label propagation

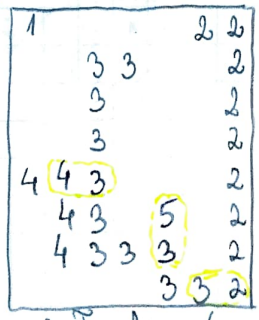

4. Bottom-up & right-left label propagation

→ Alg.:
1. Initialization step → all pixels labeled "1" will get a unique label
2. Repeat
   top-down & left-right label propagation
   bottom-up & right-left label propagation
   until no changes occur

8.2 Classical alg.

Exc:


1. Initial img.



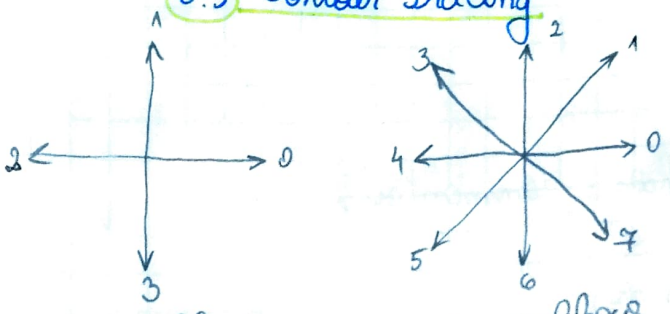
2. Top down (pass 1)

EQTABLE: (4,3), (3,5), (3,2) ...

→ 2 passes through the img, but requires a large global table for equivalences

1) 1st pass: label propagation
   ↳ find equivalences and enter them in the table (the smaller label propagates)
   ↳ each eq. class is assigned a unique label (the min label in the class)
2) 2nd pass: performs a translation, assign to each pixel the label of the eq. class of it's 1st pass label

8.3 Contour Tracing



→ N-neighbour (chain-code)

4-neighbour 8-neighbour

→ contour / i-contour of a connected set of pixels R is defined as the set of all pixels in R which have at least one d-neighbour not in R

→ d-contour of R is the set of all pixels in R, which have at least one neighbour not in R

Single contour tracing

→ the alg. can be described in terms of an observer who walks counterclockwise along pixels belonging to the set and selects the rightmost pixel available

→ it is over when the current pixel is the same as the initial pixel

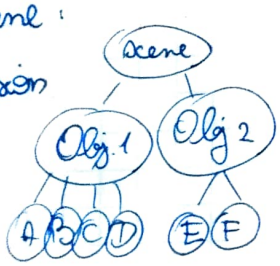
→ must be combined w/ a search alg. for locating holes in the interior of the region

→ must be applied once for each hole in a region

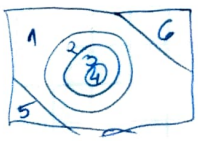
Traversal of all the contours of an img.

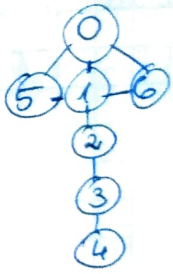
→ description of scene:

a) Tree of inclusion



b) Adjacency graph





109 31/133

→

CG

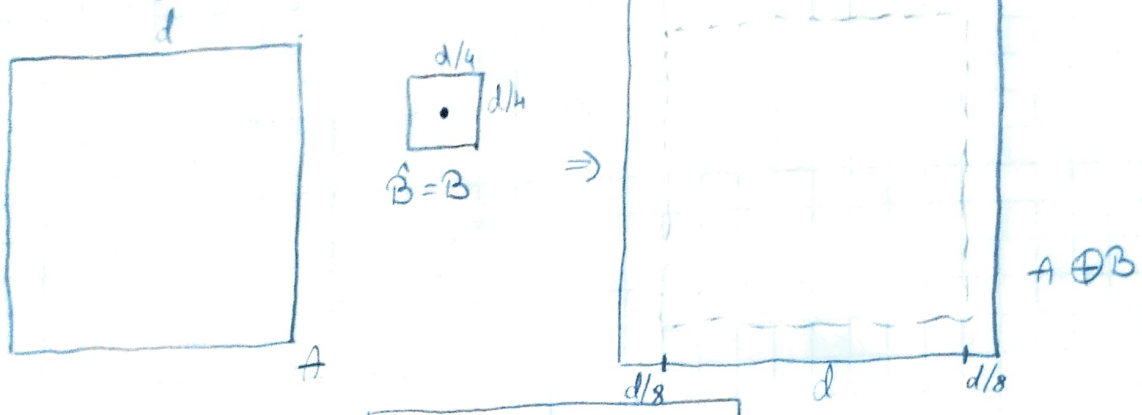
9. Morphologic Image Processing

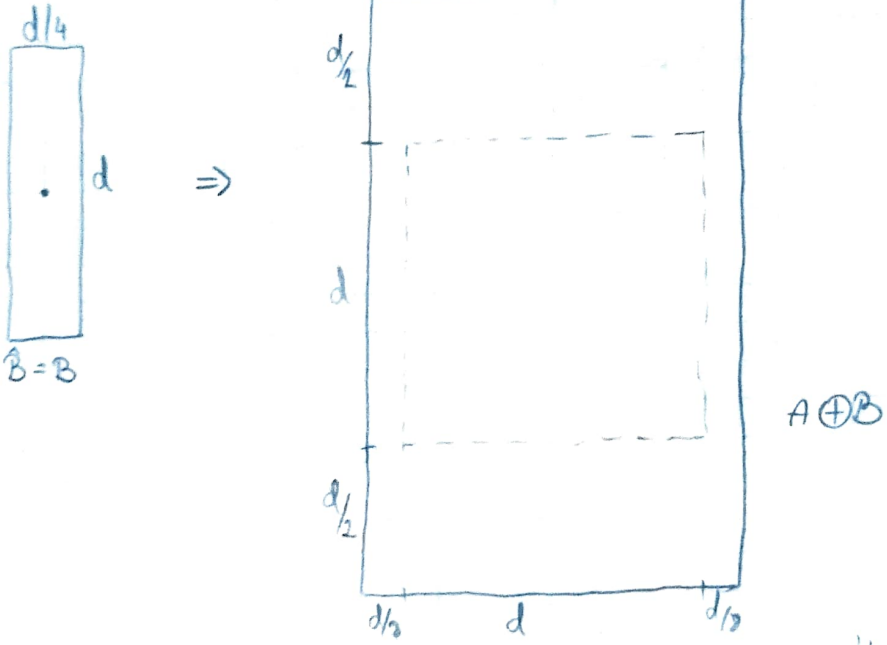
→ Concepts: subset, union, intersection, complement, difference, reflection, translation

9.1 Dilation

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

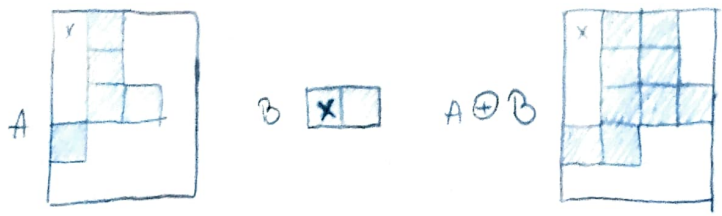
Exc. 1:





b) $A \oplus B = \{z \in \mathbb{Z}^2 \mid z = a + b, \exists a \in A, b \in B\}$ ← don't care values?

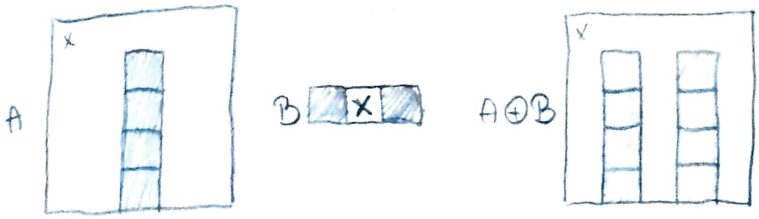
Exc. 2:



$A = \{(0,1), (1,1), (2,1), (2,2), (3,0)\}$

$B = \{(0,0), (0,1)\}$

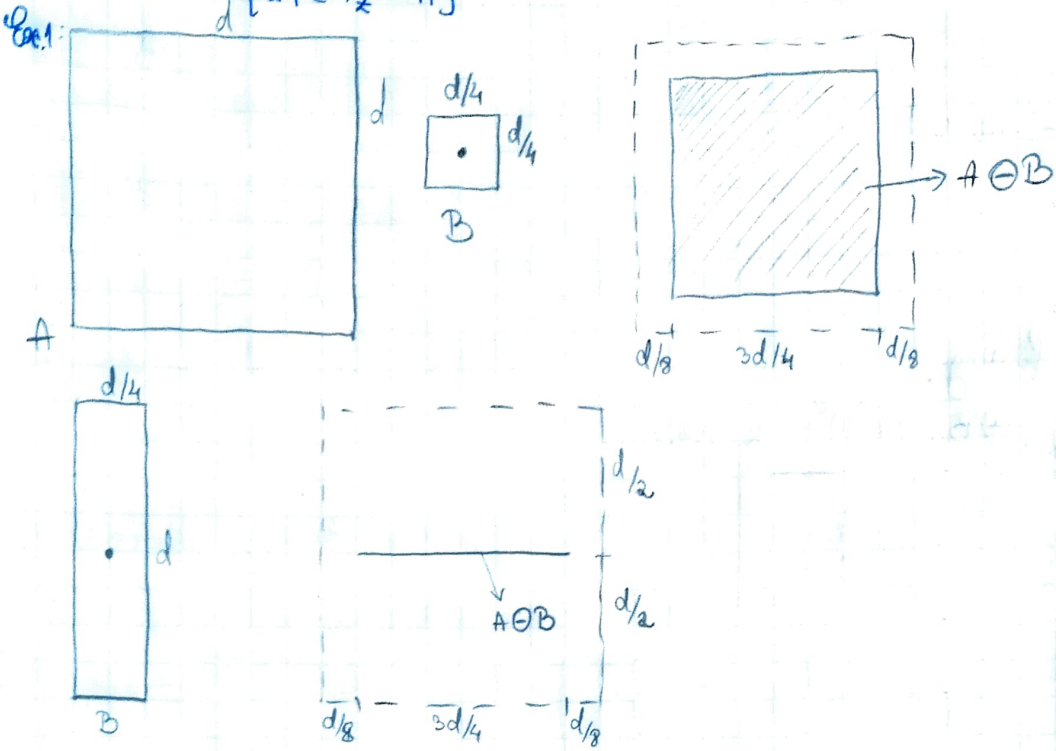
$A \oplus B = \{(0,1), (1,1), (2,1), (2,2), (3,0), (0,2), (1,2), (2,2), (2,3), (3,1)\}$



9.2 Erosion

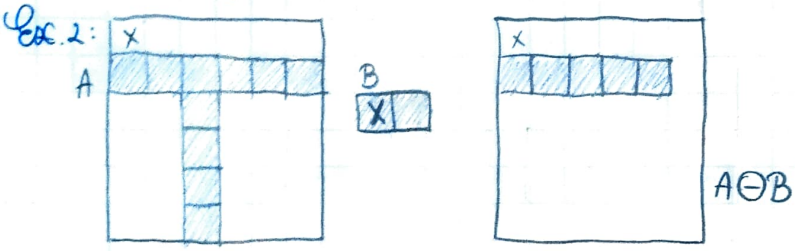
a) $A \ominus B = \{z \mid (\hat{B})_z \subseteq A\}$

Exc.1:



b) $A \ominus B = \{x \in \mathbb{Z}^2 \mid x + b \in A, \forall b \in B\}$

Exc.2:



9.3 Opening and Closing

opening: $A \circ B = (A \ominus B) \oplus B$ → smoothes the contour of an object

closing: $A \bullet B = (A \oplus B) \ominus B$ → fills gaps in the contour, eliminates small holes

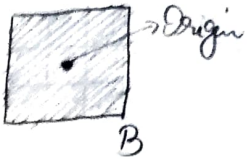
9.4 Hit-or-Miss transform

$$A \circledast B = (A \ominus J) \cap (A^c \ominus K)$$

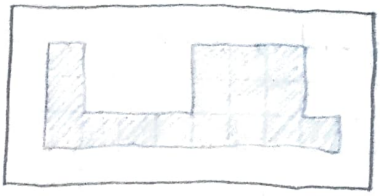
9.5 Boundary extraction

$\beta(A) = A - (A \ominus B)$

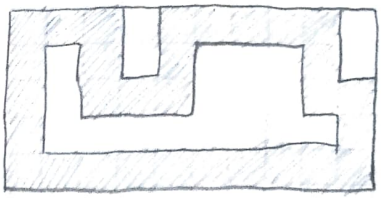

A



B

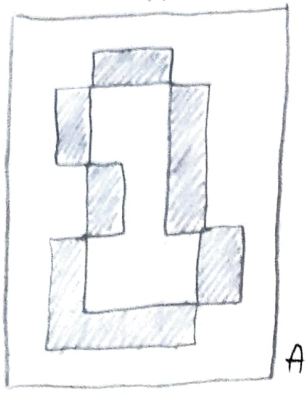

$A \ominus B$

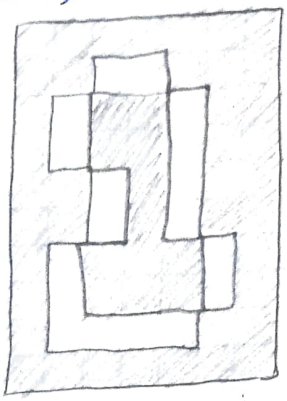
$\beta(A)$

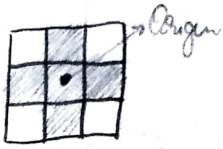


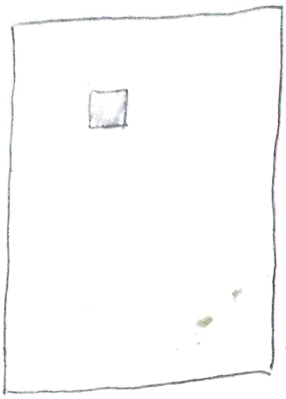
9.6 Region filling

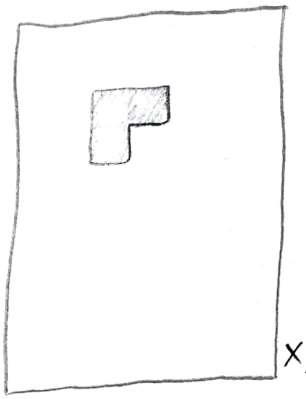
$X_k = (X_{k-1} \oplus B) \cap A^c \quad , k = 1, 2, 3, \ldots$

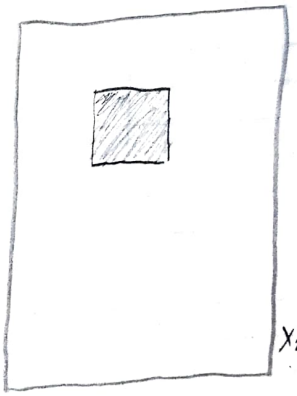

A

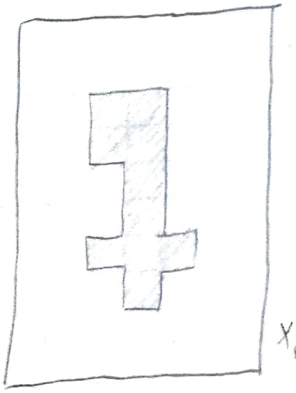

A^c

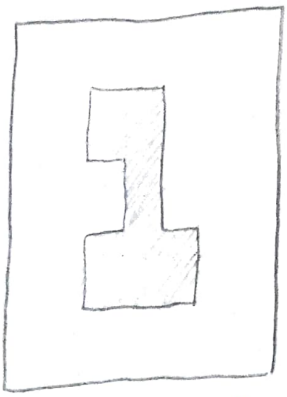


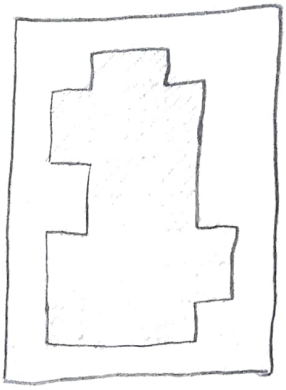

X_0


X_1


X_2


X_6


X_7


$X_7 \cup A$

9.7 Extracting connected components

9.8 Thinning $A \otimes B = A - (A \circledast B) = A \cap (A \circledast B)^c$

C7

10. Grayscale img. processing

10.1. Statistical prop.

→ mean: $\mu = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) = \langle f(i,j) \rangle$

→ variance: $\sigma^2 = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (f(i,j) - \mu)^2 = \langle | f(i,j) - \mu |^2 \rangle$

$= \langle |f(i,j)|^2 \rangle - \langle f(i,j) \rangle^2$

→ Probability distribution func.:

$P(f)$ = prob. that pixel value $\leq f$

$\Rightarrow \mu = \frac{1}{N^2} \sum_{f=0}^{f_{max}} f \cdot h(f)$

$\sigma^2 = \frac{1}{N^2} \sum_{f=0}^{f_{max}} (f - \mu)^2 \cdot h(f)$

→ Probability density func. (PDF):

PDF: $p(f) = d P(f) / df$

$p(f) = \frac{h(f)}{N^2}$ (normalized histogram)

$h(f)$ = gray level histogram of f

10.2. Grayscale img. segmentation: Basic global tresholding alg.

< computes automatically treshold T
< can be applied on images with bimodal histograms

10.3. Gamma correction

$g(\text{output}) = T(f(\text{input}))$

$g_{out} = c \cdot f_{in}^{\gamma}$

10.4. Information

- source $\xrightarrow{\text{message}}$ recipient $\rightarrow$ to not already have information

10.5. Entropy = average information of the image

10.6. Energy = how many gray-levels are distributed

10.7. Histogram equalization: AIM = to distribute pixels equally across available grey level range

$f \in [0 \dots L-1] \Rightarrow r \in [0 \dots 1]$ → normalized grayscale levels

$s = T(r) \in [0 \dots 1] \Rightarrow g \in [0 \dots L-1]$ → transformation function

- T features: a) single valued and monotonically increasing $\Rightarrow \exists r = T^{*}(s)$
  b) $0 \le T(r) \le 1$

- $p_r(r)$, $p_s(s)$ - probability density functions of input & output

(1): $p_s(s) = \frac{dP(s)}{ds} = p_r(r)\frac{dr}{ds}$

(2): $s = T(r) = \int_0^r p_r(w)\,dw$

10.8. Cumulative density function (CDF):

(2): $s = T(r) = \int_0^r p_r(w)\,dw$

(3): $\frac{ds}{dr} = \frac{dT(r)}{dr} = \frac{d}{dr}\left[\int_0^r p_r(w)dw\right] = p_r(r)$ (prin Leibniz)

(1)+(3) $\Rightarrow p_S(s) = p_r(r)\frac{dr}{ds} = p_r(r)\frac{1}{p_r(r)} = 1$ } Histogram equalization

- uniform PDF
- independent from $p_r(r)$

10.3 Histogram eq alg.:

$p_r(r_k) = \frac{n_k}{n}$, $k = 0 \dots L$

$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}$, $k = 0 \dots L-1$

⇒ re-map the gray-scale values of the output img.: $r_k \rightarrow s_k$

$g_k = \text{round}(s_k(L-1))$

⇒ re-scale the gray-scale values of the output img.: $s_k \rightarrow g_k$

10.10 Histogram specification / matching

C8 **11. Convolution**

$g(x,y) = h(x,y) * f(x,y)$

$f * g = g * f$ → commutativity

$f * (g * h) = (f * g) * h$ → associativity

$f * (g + h) = (f * g) + (f * h)$ → distributivity

$f * \delta = \delta * f = f$ → identity element

$a(f * g) = (af) * g = f * (ag)$ → associativity w/ scalar multiplication

$D(f * g) = Df * g = f * Dg$ → differentiation rule

$F(f * g) = kF(f)F(g)$ → convolution theorem