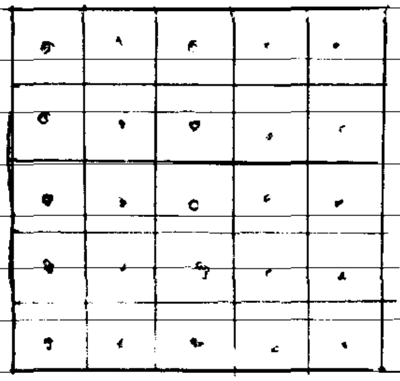
Great! P = 9.7 T = 9.9 Final = 10

Afdeling Wiskunde en Informatica R.U.G.

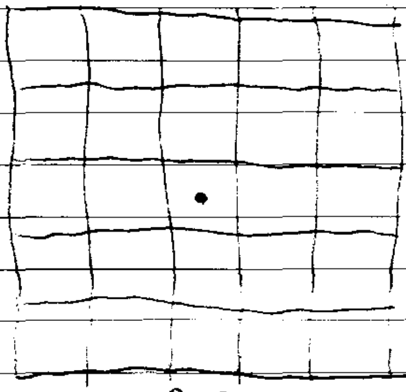
Naam: André Emerencia	Studentnummer: 1283436	Bladnr.: 1/3
Adres: Eslaantraan 390	Studierichting: Inf	Tentamen:
Postcode en 9741 MC		Datum:
Woonplaats: Groningen	Jaar van eerste inschrijving: 2001	Naam docent:

Problem 1a



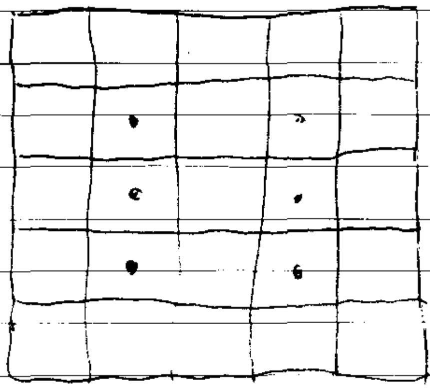
$X \oplus A$

The dilation of X by A fills the entire image.

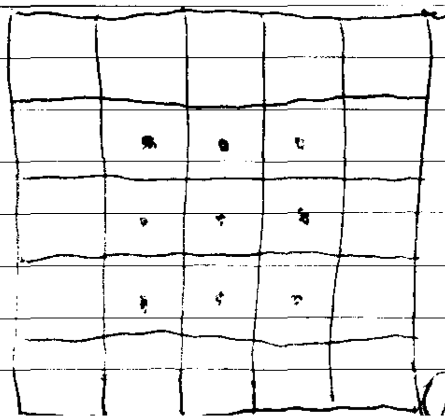


$X \ominus A$

Erosion leaves only one point in this case.



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



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

1b

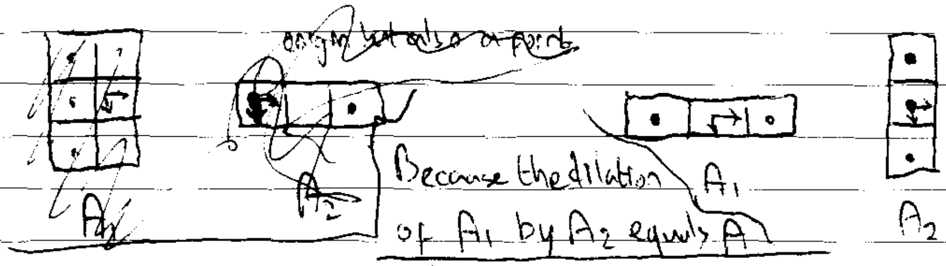
$(X \ominus A_1) \ominus A_2$ can be rewritten because of the iteration property:

$X \ominus (A_1 \oplus A_2)$

1

So we see that $A = A_1 \oplus A_2$

So we can ~~write~~ take for A_1 and A_2 for example:



c

Because the ~~size of structuring~~ ~~number of black pixels in the structuring element~~ A is

size of structuring element $A = 9$

~~for~~ for each pixel in X ~~needs to be~~ is checked 9 times (in brute force)

0.4

A_1 has ~~three black pixels and A_2 has 3~~ size 3, and A_2 has size 3 as well, so $3+3=6$ ~~so 3+3 = 6~~ ops per pixel of X

So $(X \ominus A_1) \ominus A_2$ needs $\frac{6}{9} = \frac{2}{3}$ of the pixel operations needed to compute $X \ominus A$.

actually only the foreground pixels need be included

2.4

2a

northerly:

1	2	1
0	0	0
-1	-2	-1

north-west:

2	1	0
1	0	-1
0	-1	-2

1

northeast:

0	1	2
-1	0	1
-2	-1	0

here propagating the property that the middle row weighs heavier than ~~the sides~~ things on either side.

b

The simple gradient filter kernel will be more sensitive to noise because it only looks at the pixel values in a single row. ~~The Sobel kernel, by looking at multiple rows.~~

0.5

By taking the values of multiple rows, the Sobel operator, in general, processes a sort of smoothing, making it less sensitive

Problem 2c

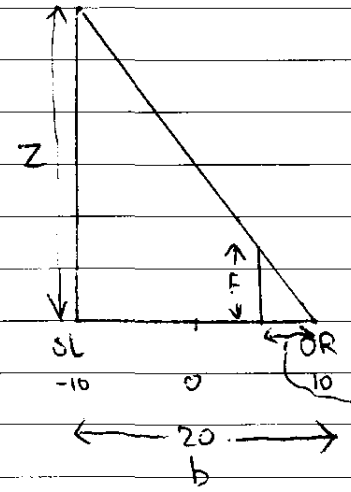
0.5

Another way to remove noise is by first smoothing the image with a gaussian kernel ~~applying median filtering~~. ~~Potential problems: computing the median takes more time, special hardware might not be available.~~

Problems: edges and noise gets smoothed, so detail is lost.

3.1

We can see the feature in the x,z plane below:



Because the feature shows in the origin in the left picture, we know that its true ~~x~~ x = OL(x) = -10 and its y = OL(y) = 0

To get ~~size~~ depth z we calculate, similar triangles:

$$\frac{2}{f} = \frac{20}{z}$$

$$\frac{22}{20} = 20$$

$$22 = 400$$

$$z = 200$$

So the (x,y,z) position of the object is (-10, 0, 200)

- The distance b between O_L and O_R
- Distance f to the image planes.
- The disparity $p = u_L - u_R$

Formulas for perspective projections $P_L = (u_L, v_L)$ and $P_R = (u_R, v_R)$ given as:

$$\begin{pmatrix} u_L \\ v_L \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x + \frac{b}{2} \\ y \end{pmatrix}$$

$$\begin{pmatrix} u_R \\ v_R \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x - \frac{b}{2} \\ y \end{pmatrix}$$

Solving for z gives:

$$z = \frac{f \cdot b}{\underbrace{u_L - u_R}_{p}}$$

So $z \cdot p = b \cdot f$

Accuracies d_f, d_b, d_p lead (to the first order) to accuracy d_z:

$$z\,dp + p\,dz = b\,df + f\,db$$

or

$$\frac{dz}{z} = \frac{db}{b} + \frac{df}{f} - \frac{dp}{p}$$

Assuming errors db, df, dp uncorrelated with variances $\sigma_b, \sigma_f, \sigma_p$ respectively, result in variance σ_z:

$$\left(\frac{\sigma_z}{z}\right)^2 = \left(\frac{\sigma_b}{b}\right)^2 + \left(\frac{\sigma_f}{f}\right)^2 + \left(\frac{\sigma_p}{p}\right)^2$$

With perfect calibration and perfect orientation, the relative precision of z reduces to:

$$\frac{\sigma_z}{z} = \frac{\sigma_p}{p} = \frac{z \cdot \sigma_p}{f \cdot b}$$

Conclusion: Relative precision of z is inversely proportional to the disparity p. Nearby points can be determined with higher accuracy. And it is proportional to f and b, increasing these gives better (smaller) precision.

Afdeling Wiskunde en Informatica R.U.G.

Naam: Arvid Emerencia	Studentnummer: 1283936	Bladnr.: 3/3
Adres:	Studierichting:	Tentamen:
Postcode en		Datum:
Woonplaats:	Jaar van eerste inschrijving:	Naam docent:

Problem 4a

(i) thresholding: thresholding is clearly region based, since it depends on properties of image regions (such as intensity), rather than on edges.

(ii) watershed: The watershed algorithm is also region based, since ~~pixels~~ regions grow ~~pixels~~ ~~toward~~ from local extrema and markers.

(iii) Snakes: The snakes algorithm is edge based, since it is ~~a~~ attracted to edges (large image gradients).

4b

(i) thresholding: + easy and fast to implement.
- harder to automate (what is the right threshold to set?)
- Doesn't work on very noisy images or images with high contrast texture

(ii) watershed: + Dynamic, allows you to set markers.
- harder to automate (where to place markers?)

(iii) snakes: + Very suited for deriving/filling up edges with missing points
+ Allow usage of a priori (higher) knowledge
− Harder to implement (lots of parameters to set)
− Very dependent on initial location/initialization

(2)