Tutorial on:
Digital Topology, Geometry and Applications

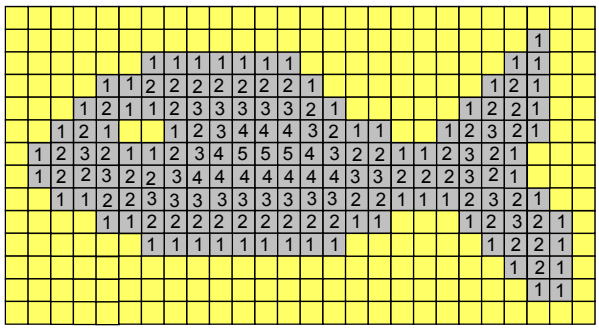
1st talk: Distance Transforms

Gunilla Borgefors

Centre for Image Analysis
Uppsala University

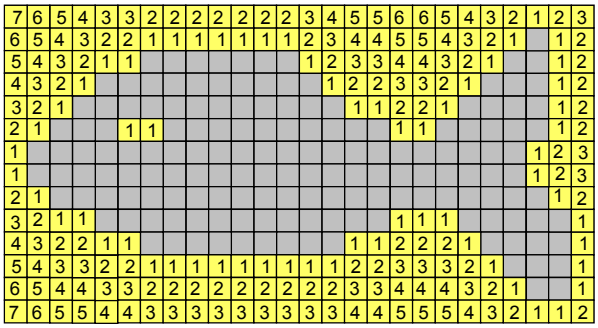
Gunilla.Borgefors@it.uu.se

Distance into shape



Each pixel gets a value that is the distance to the nearest background pixel in the used metric.

Distance from shape



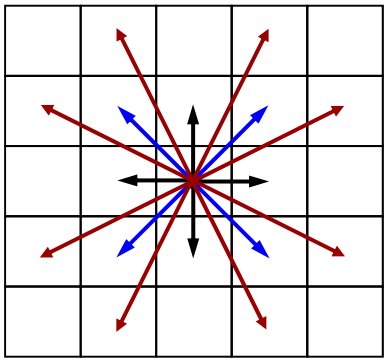
A distance transform imposes a structure on an object / background that can be used for manipulating, recognizing, and analysing the image

Contents

The distance between two points in Z^n is defined as the length of the shortest path connecting them in an appropriate graph.

Depends on:

1. Neighbourhood relation
2. Definition of path length



edge-neighbour steps

point-neighbour steps

knight-neighbour steps

3 x 3 neighbourhood mask

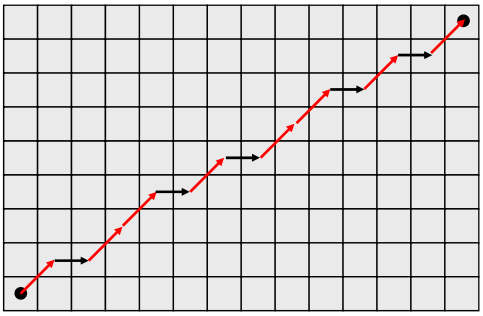
+b	+a	+b
+a	+0	+a
+b	+a	+b

3x3 weighted DT

the weights **a** and **b** are the local distances between the pixels

9

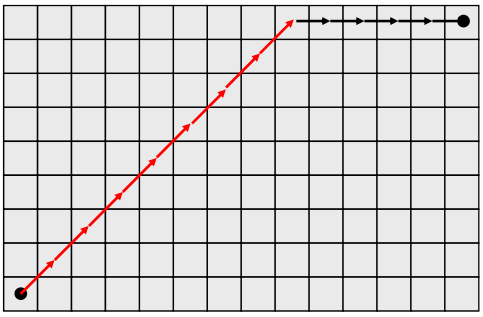
Path between two pixels

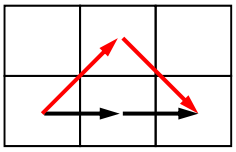


10

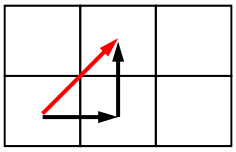
Path – alternative order, same distance

It's not about simulating a straight line – it is about measuring a distance!





2a < 2b

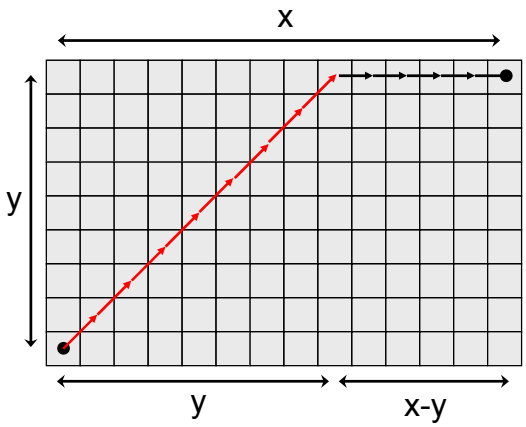


1b < 2a

A 3 x 3 weighted DT is

REGULAR if **a** < **b** < 2**a**

SEMI-REGULAR if **a** ≤ **b** ≤ 2**a**



path length =

b y + **a** (x-y) =

x **a** + y (**b** - **a**)

What are the optimal **a** and **b**?

Assume x ≤ y and an M x M image. Minimize rotation dependence.

Difference from Euclidean for max x = M is

$$\text{Diff}(y) = y\,(b - a) + aM - \sqrt{x^2 + y^2}, \quad 0 \le y \le M$$

Max of Diff(y) occurs for

y=0	$E1 = (a-1)\,M$
y : Diff'(y) = 0	$E2 = (a-\sqrt{1-(b-a)^2})\,M$
y = M	$E3 = (b-\sqrt{2})\,M$

min{E1, E2, E3} occurs for -E1 = E2 = -E3, so solve

$$1-a = a-\sqrt{1-(b-a)^2} = \sqrt{2}-b$$

Optimal weights

$$a_{opt} = \frac{1}{2} + \frac{1}{2}\sqrt{2\sqrt{2}-2} \approx 0.955$$

$$b_{opt} = \sqrt{2} - \frac{1}{2} + \frac{1}{2}\sqrt{2\sqrt{2}-2} \approx 1.369$$

$$\text{maxdiff} = \frac{1}{2} - \frac{1}{2}\sqrt{2\sqrt{2}-2} \approx 0.045\ M$$

Optimal weights

$$a_{opt} = \frac{1}{2} + \frac{1}{2}\sqrt{2\sqrt{2}-2} \approx 0.955$$

$$b_{opt} = \sqrt{2} - \frac{1}{2} + \frac{1}{2}\sqrt{2\sqrt{2}-2} \approx 1.369$$

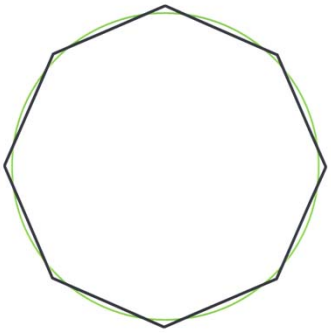
$$\text{maxdiff} = \frac{1}{2} - \frac{1}{2}\sqrt{2\sqrt{2}-2} \approx 0.045 \text{ M}$$

Note that a = 1, b = $\sqrt{2}$ give maxdiff = 0.090 M

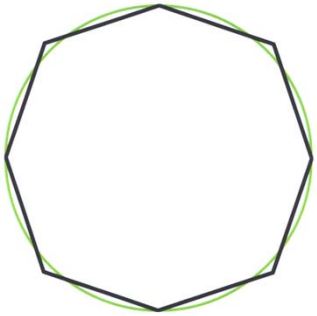
Integer approximations

a	b	maxdiff
opt	opt	0.064
2	3	0.134
3	**4**	0.081
8	11	0.073

Multiply the optimal **a**, **b** by a number and then round to the nearest integer.

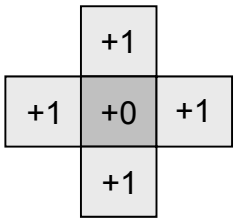


optimal <a, b> weights disc



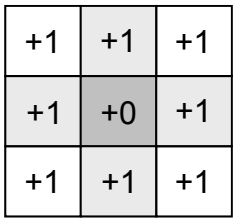
<3, 4> disc

Path-generated distance transforms



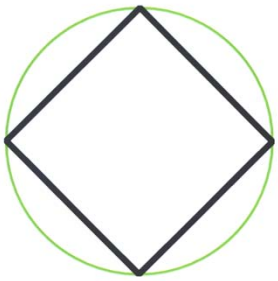
City block / Manhattan DT

D^4

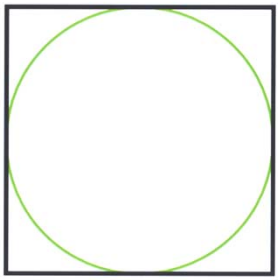


Chessboard DT

D^8



City block disc



Chessboard disc

5 x 5 neighbourhood mask

	+c		+c	
+c	+b	+a	+b	+c
	+a	+0	+a	
+c	+b	+a	+b	+c
	+c		+c	

adding **knight**-neighbour step **c**

Optimal a, b, c are complex expressions, approximately

$\mathbf{a}_{opt} \approx 0.986$

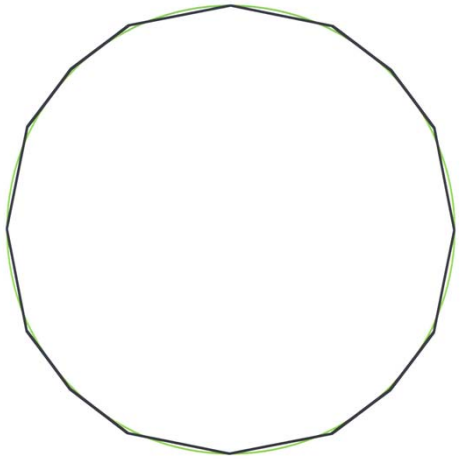
$\mathbf{b}_{opt} \approx [1.400, 1.422]$

$\mathbf{c}_{opt} \approx 2.208$

maxdiff ≈ 0.014 M

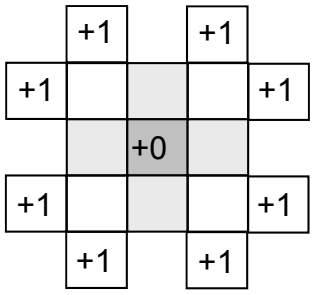
Integer approximation

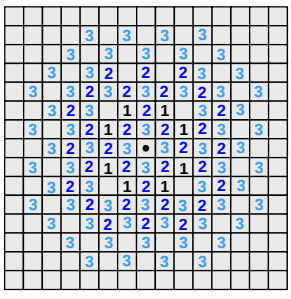
a	b	c	maxdiff
opt	opt	opt	0.0141
5	**7**	**11**	0.0202



<5, 7, 11> disc

Path-generated Knight-distance





this generates a (strange) metric

Euclidean distance transform

+(1,1)	+(0,1)	+(1,1)
+(1,0)	+(0,0)	+(1,0)
+(1,1)	+(0,1)	+(1,1)

A vector in each pixel, counting x-steps and y-steps separately.

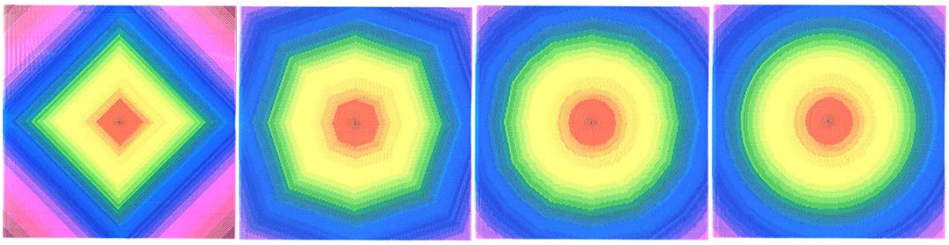
Distance value = $\sqrt{x^2 + y^2}$,
but usually $(x^2 + y^2)$ is used

Useful 2D distance transforms

	a	**b**	**c**	maxdiff
City block	1	-	-	58.6 %
3 - 4	3	4	-	8.1 %
5-7-11	5	7	11	2.0 %
Euclidean	(1,0)	(1,1)	-	0* %

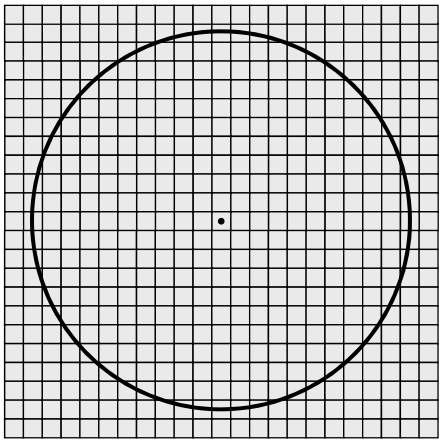
* If computed correctly (which is not trivial!)

Useful 2D distance transforms



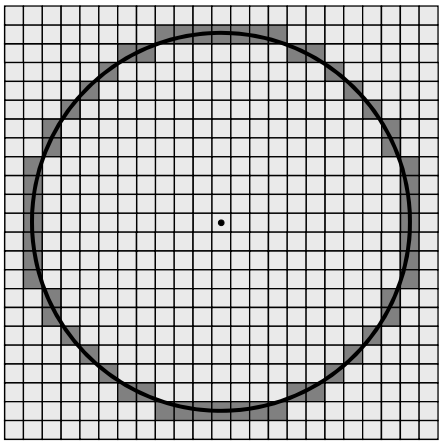
city block <3,4> <5, 7, 11> EDT

Distance transform from single point



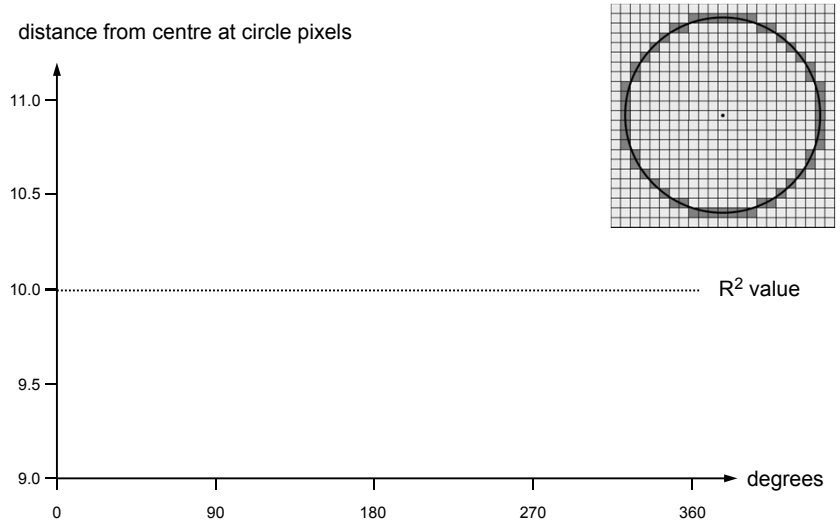
R^2 circle, radius = 10

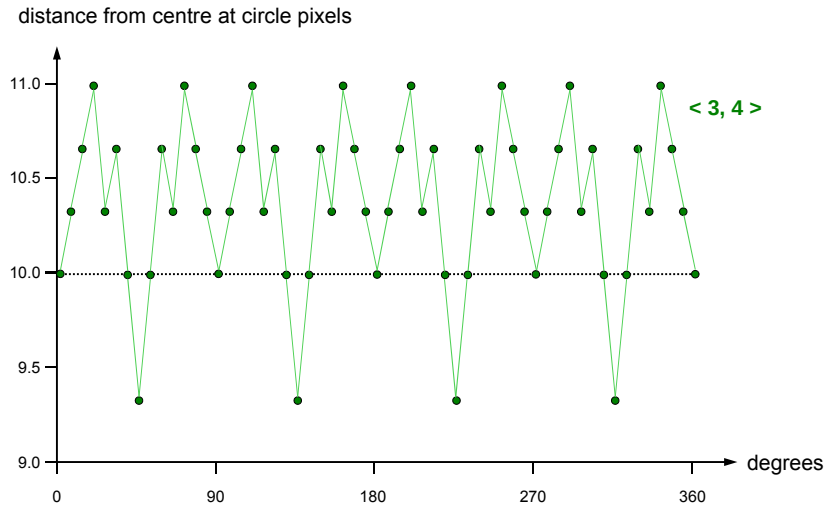
29

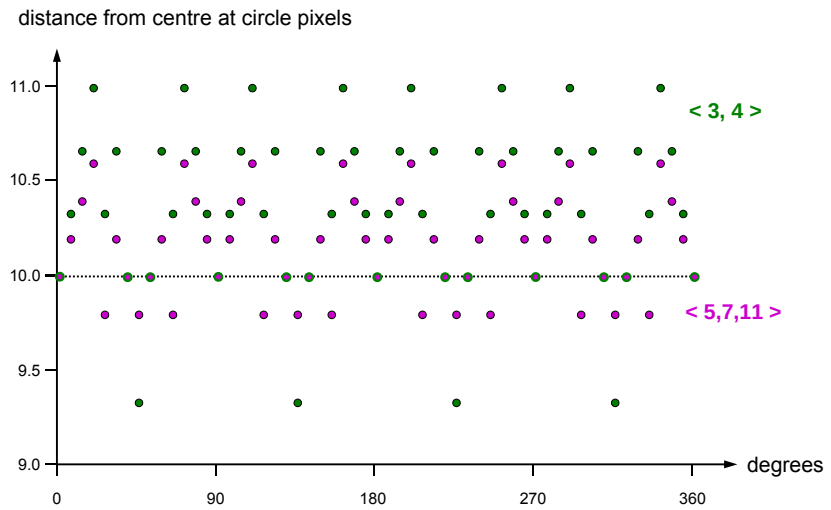


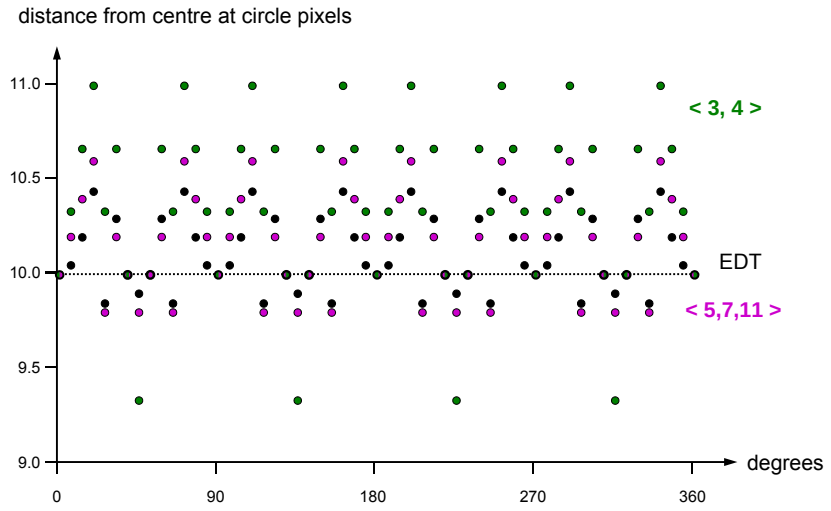
Z^2 circle, radius = 10

30



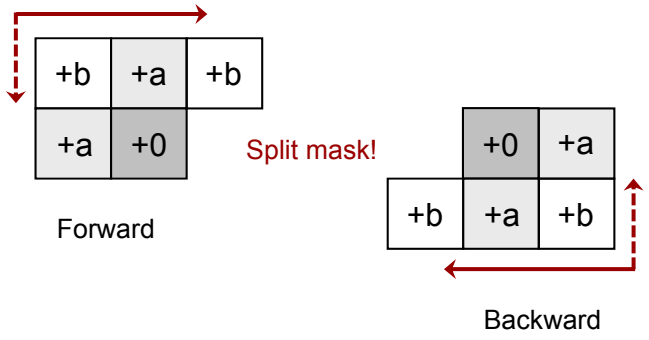

distance from centre at circle pixels
11.0
10.5
10.0
9.5
9.0
< 3, 4 >
0
90
180
270
360
degrees


distance from centre at circle pixels
11.0
10.5
10.0
9.5
9.0
< 3, 4 >
< 5,7,11 >
0
90
180
270
360
degrees



IS the irregular and EDT really that much better than the regular 5-7-11 DT in the discrete grid?

Computation by chamfering

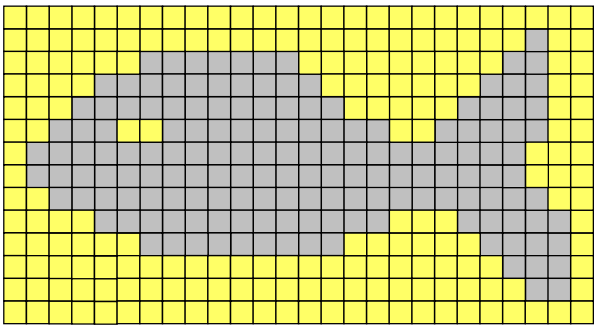


Raster scan the masks over the image. For each pixel, add the value in the mask to the pixel under it and put the minimal value as the new value of the pixel.

35

<3,4> computation by chamfering

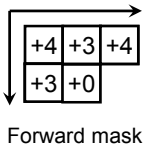
Initialisation



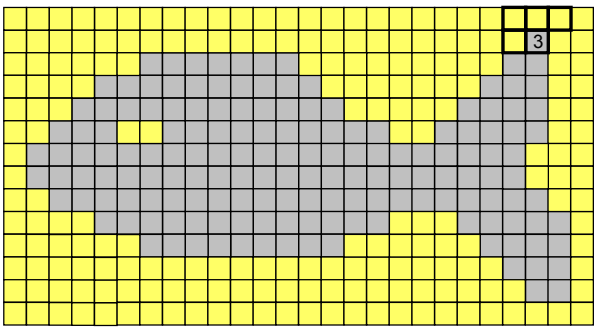
36

<3,4> computation by chamfering

Forward pass – first change

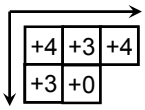


Forward mask

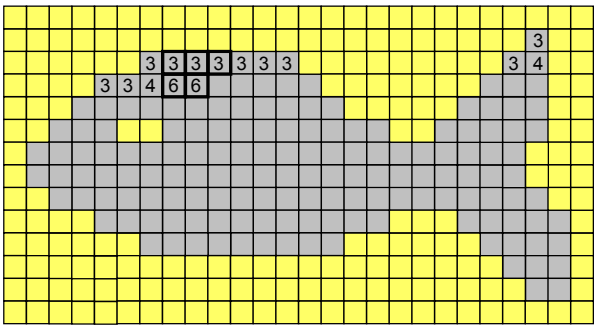


<3,4> computation by chamfering

Forward pass – ongoing

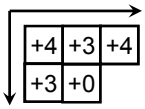


Forward mask

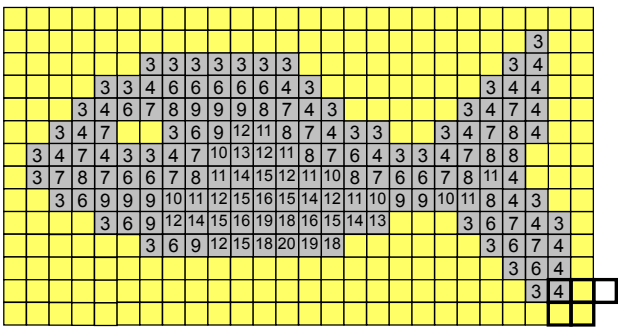


<3,4> computation by chamfering

Forward pass – finished



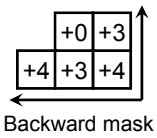
Forward mask

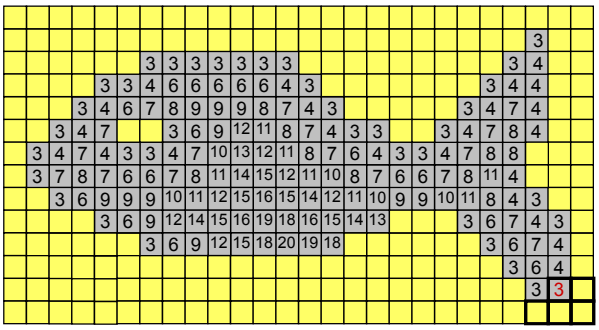


Distances are now measured from left and from above.

<3,4> computation by chamfering

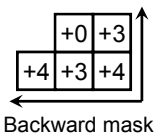
Backward pass – first change

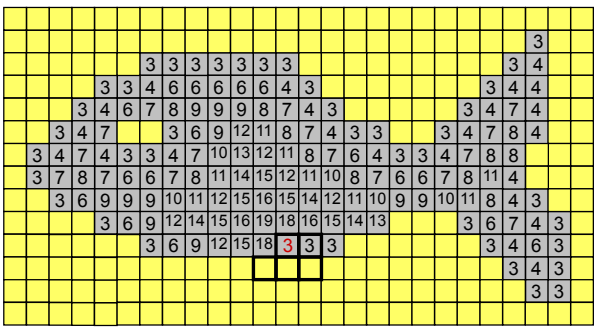




<3,4> computation by chamfering

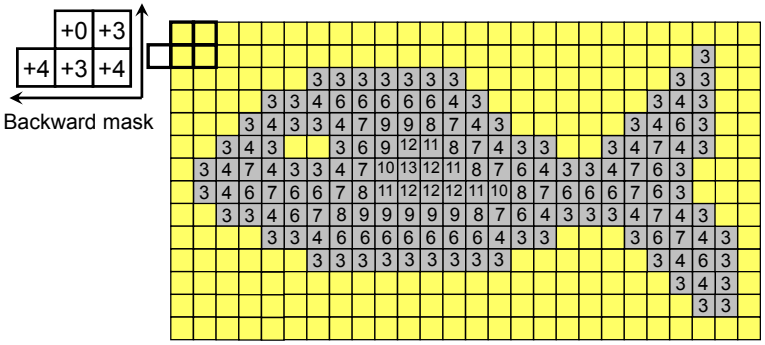
Backward pass – ongoing





<3,4> computation by chamfering

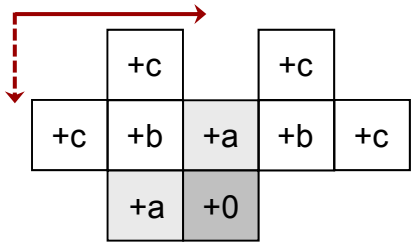
Backward pass – finished



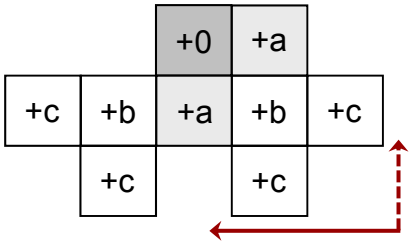
Final result

Distances are now measured also from right and from bottom.

Chamfering: 5x5 masks



Forward

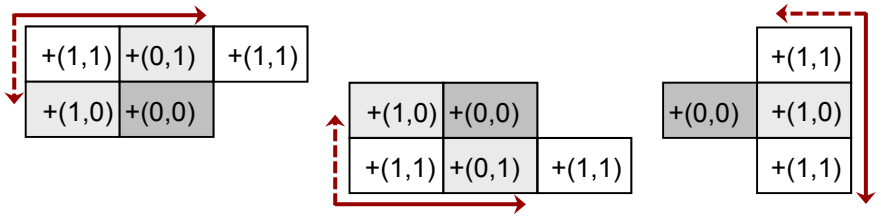


Backward

Only two passes necessary also in this case.

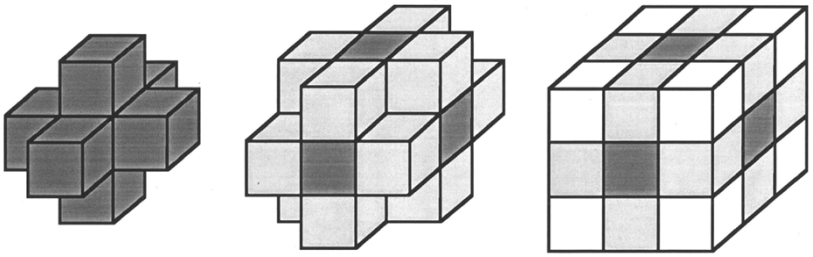
Chamfering: Euclidean distance

...can be computed by three raster scans…



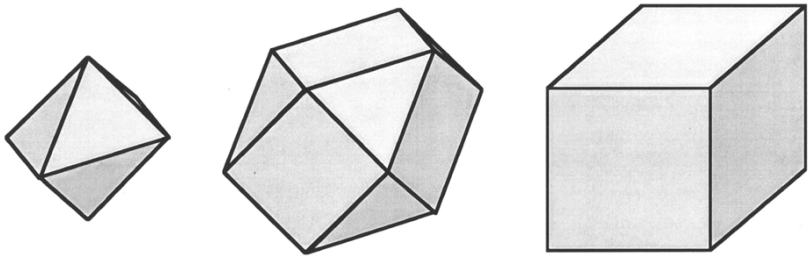
…but the result is not quite correct in all pixels and the result is not a metric. Use more complex, but correct algorithms!

3D distance transforms



area neighbours (6) + edge neighbours (18) + point neighbours (26)

3D distance transforms - path-generates spheres



area neighbours D^6

+ edge neighbours D^{18}

+ point neighbours D^{26}

Do not use D^{18} – it has some weird properties.

3 x 3 x 3 neighbourhood

+c	+b	+c
+b	+a	+b
+c	+b	+c

+b	+a	+b
+a	+0	+a
+b	+a	+b

+c	+b	+c
+b	+a	+b
+c	+b	+c

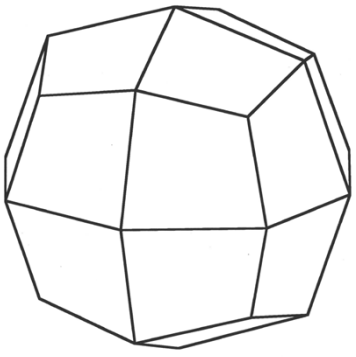
A weighted DT is semiregular if

$a \le b \le 2a$
$b \le c \le 3b/2$

Useful 3D distance transforms

	a	b	c	maxdiff
D^6	1	-	-	126.8 %
3-4-5	3	4	5	11.8 %
optimal	0.926	1.341	1.658	7.4 %
Euclidean	(1,0,0)	(1,1,0)	(1,1,1)	0* %

* If computed correctly (which is not trivial!)



<3, 4, 5> sphere = icositetrahedron

Natural garnet crystal

Reverse distance transform

Start from seed points with radius values.

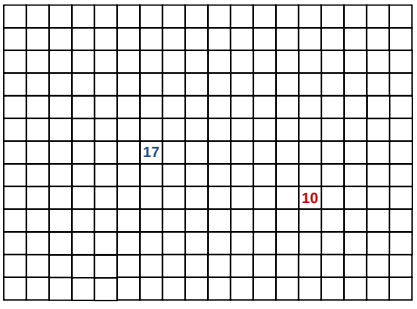
Subtract local distance from neighbours in mask

Maximize the values

works for city block and weighted distances.

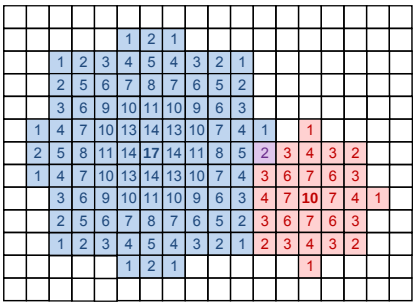
Computing the reverse Euclidean DT is quite complex.

Reverse distance transform



Reverse <3,4> from seed points **17** and **10**.

Reverse distance transform



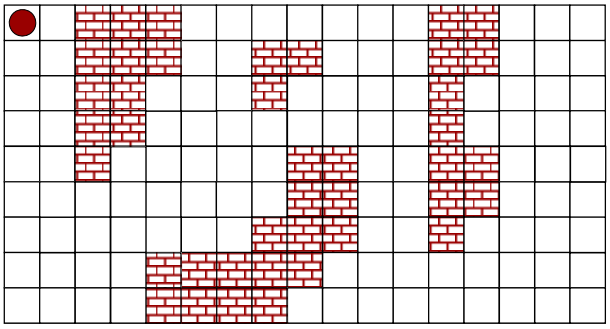
Reverse <3,4> from seed point 17.

DT values are not the same. But the **shape** is the correct one.

Constrained distance transform

start

constraint

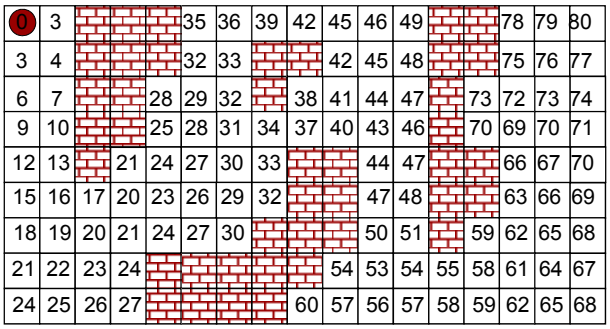


Compute DT from start using parallel algorithm

Constrained distance transform

start

constraint

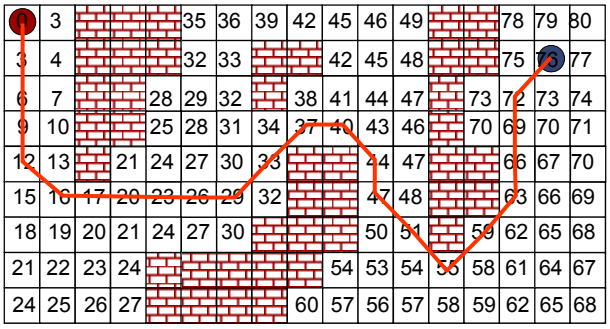


Constrained distance transform

start

goal

constraint



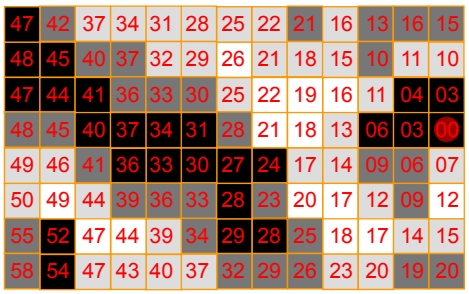
Shortest path: follow gradient from goal to start

Geodesic distance transform

2D: Consider a grey-level image as a landscape and find the shortest path though it.
Problem: How high is a grey-level step?

<3-4> DT
4 grey-levels of height 2

start



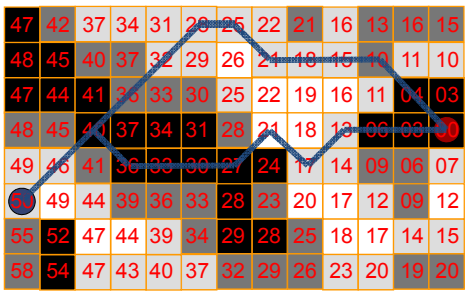
Geodesic distance transform

Shortest path(s) in the geodesic DT

<3-4> DT
4 grey-levels of height 2

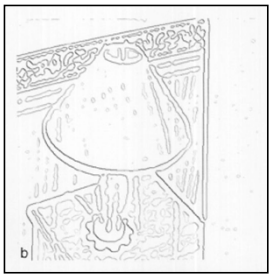
start

goal

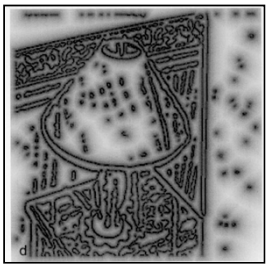


Salience distance transform

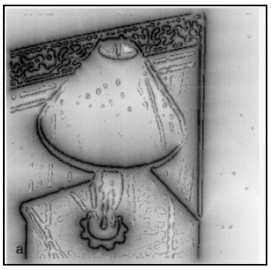
Spread other information than grey-level together with the distance:
Edge strength – Edge length – Curvature – etc.



Gradient map



DT map from thresholded edges



Salience DT map

Stolen from Paul Rosin and Geoff West

A few key references

Borgefors, G. (1986). Distance Transformations in Digital Images, *Computer Vision, Graphics and Image Processing* 34, pp. 344–371.

Borgefors, G. (1996). On Digital Distance Transforms in Three Dimensions, *Computer Vision and Image Understanding* 64, pp. 368–376.

Borgefors, G. (2003). Weighted Digital Distance Transforms in Four Dimensions, *Discrete Applied Mathematics* 125, pp. 161–176.

Coeurjolly, D. (2003). *d*-Dimensional Reverse Euclidean Distance Transformation and Euclidean Medial Axis Extraction in Optimal Time, *LNCS 2886*, Springer-Verlag, pp. 327–337.

Das, P.P. and Chatterji, B.N. (1988). Knight's Distance in Digital Geometry, *Pattern Recognition Letters* 7, pp. 215–226.

Ikonen, L. and Toivanen, P. (2007). Distance and nearest neighbour transforms on grey-level surfaces, *Pattern Recognition Letters* 28, pp. 604–612.

A few more key references

Piper, J. and Granum, E. (1987). Computing Distance Transformations in Convex and Non-Convex Domains, *Pattern Recognition* 20, pp. 599–615.

Rosenfeld, A. and Pfaltz, J.L. (1968). Distance Functions on Digital Pictures, *Pattern Recognition* 1, pp. 33–61.

Ragnemalm, I. (1993). The Euclidean Distance Transform in Abritrary Dimensions, *Pattern Recognition Letters* 14, pp. 883–888.

Rosin, P.R. and West, G.A.W. (1995). Salience Distance Transforms, *Graphical Models and Image Processing* 57, pp. 483–521.

Sintorn, I.M. and Borgefors, G. (2004). Weighted Distance Transforms for Volume Images Digitized in Elongated Voxel Grids, *Pattern Recognition Letters* 25, pp. 571—580.

Ye, Q.Z. (1988). The Signed Euclidean Distance Transform and Its Applications, In: *Proc. 9th ICPR*, Rome, Italy, pp. 495–499.

THE END