The Eccentricity Transform on CUDA

Andrej Varchola

The Eccentricity Transform: Definition

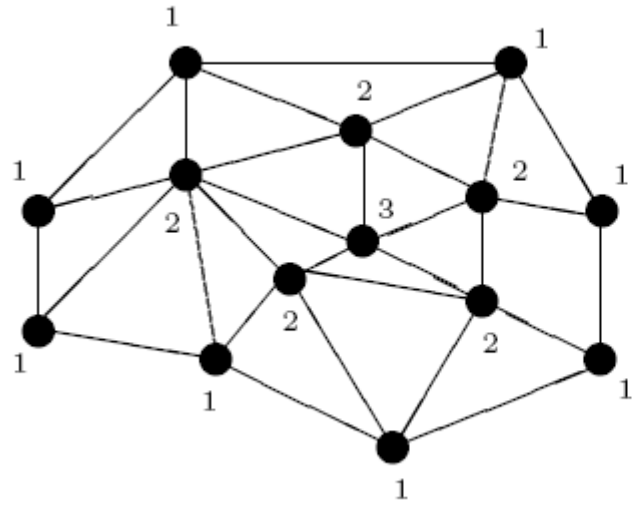
"The eccentricity transform assigns to each point in the binary image the shortest distance to the point farthest away from it. [1]"

$$ecc(p) = \max\{\lambda(\pi(p,q)) \mid \forall q \in B\}$$

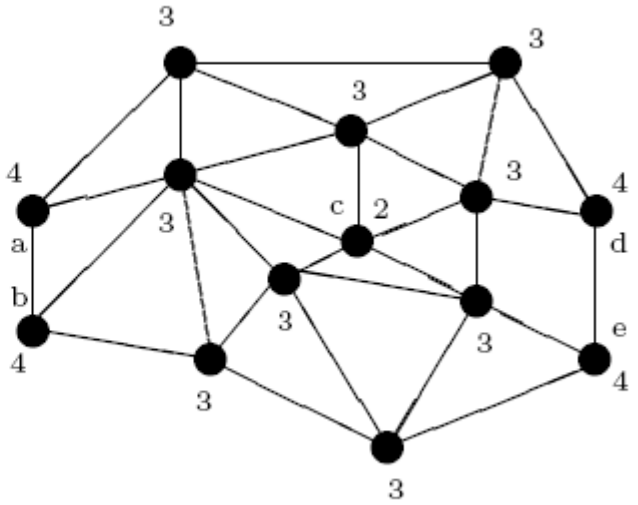
1. Kropatsch et. al, The Eccentricity Transform (of a Digital Shape), 2006

The Distance Transform

"The distance transform assigns to each point in the binary image a value of a distance to the closest point on its border. [1]"

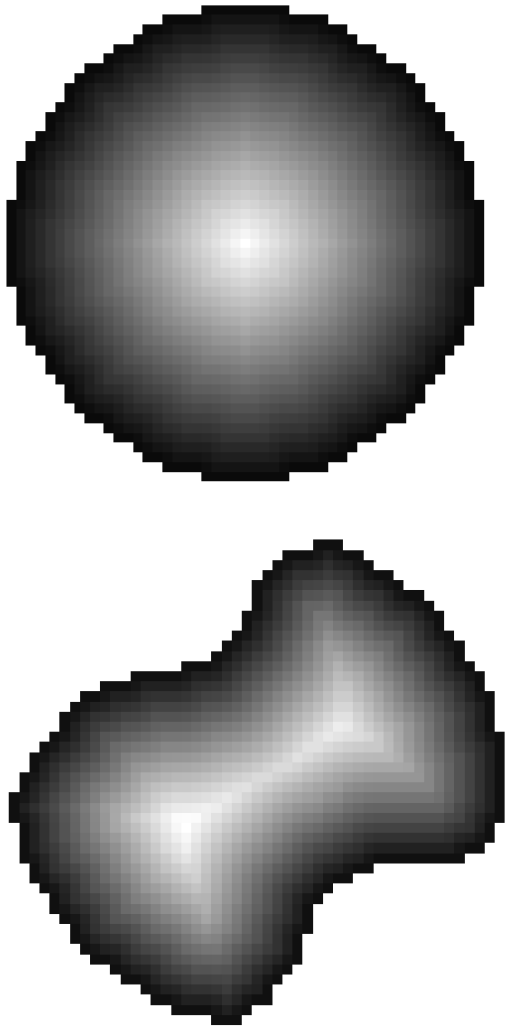


Distance transform of a graph

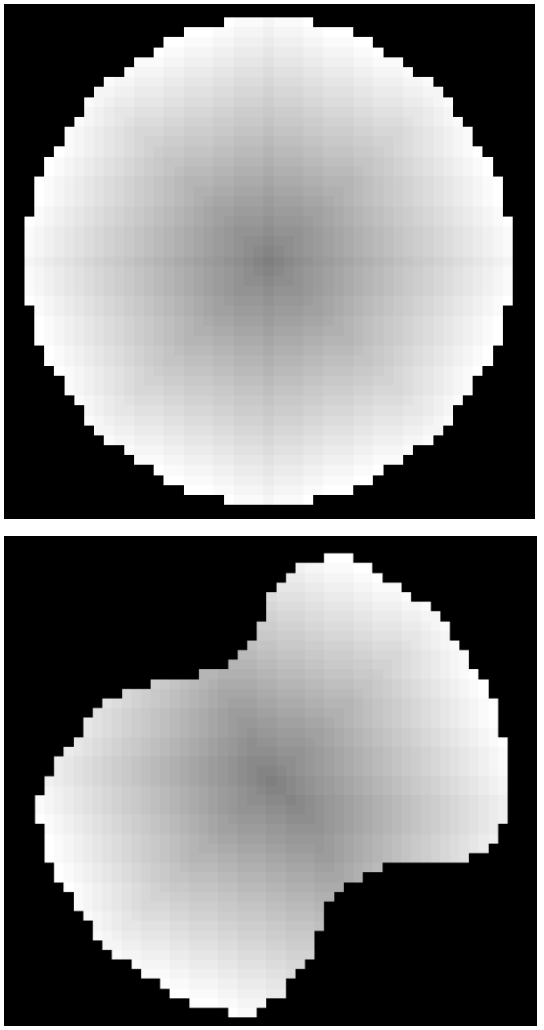


Eccentricity transform of a graph

The Eccentricity Transform of a Binary Image



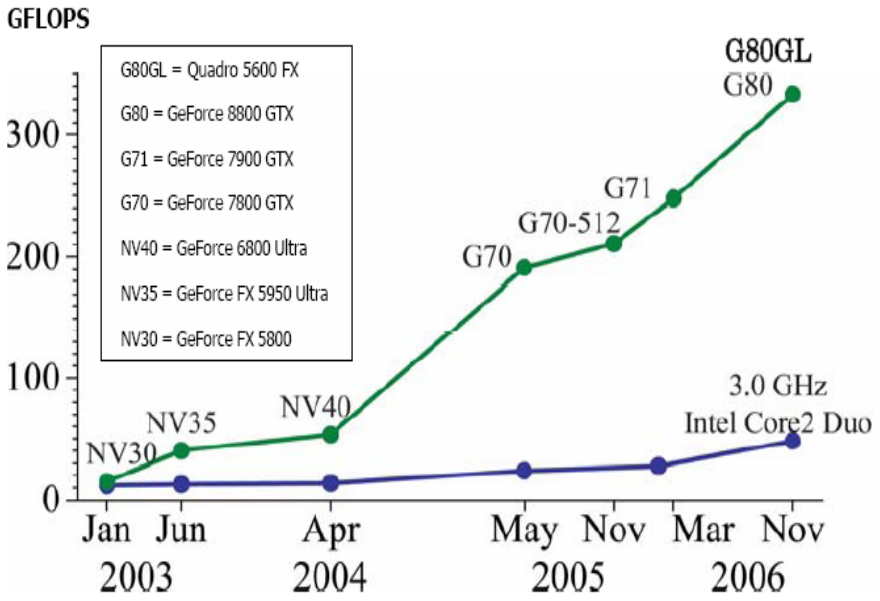
Distance transform



Eccentricity transform

CUDA: Compute Unified Device Architecture

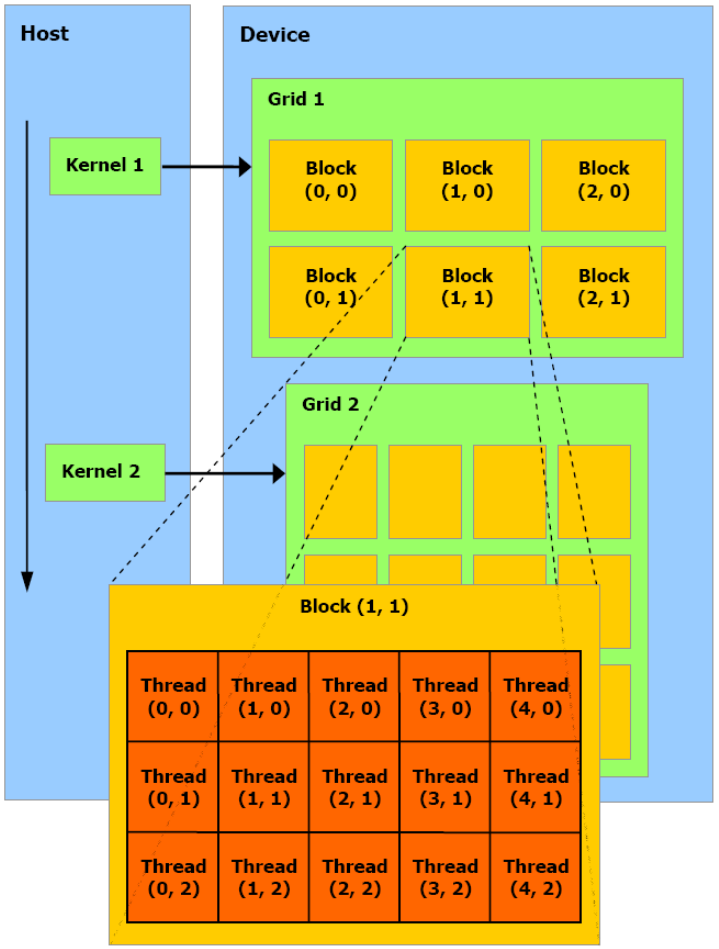
- GPU is specialized for data processing (rather than data caching and flow control)
- G80 has 16 multiprocessors w/ 8 processors each
- At any given cycle is same instruction executed on different data- SIMD



Floating-Point operations per second for CPU and GPU

CUDA: Compute Unified Device Architecture

- Collection of *threads* running in parallel
- Each thread executes single instruction set – *kernel*
- Each thread is given unique *ID*
- If number of threads is more then a warp size, they are time shared *internally*



The Eccentricity Transform via DF

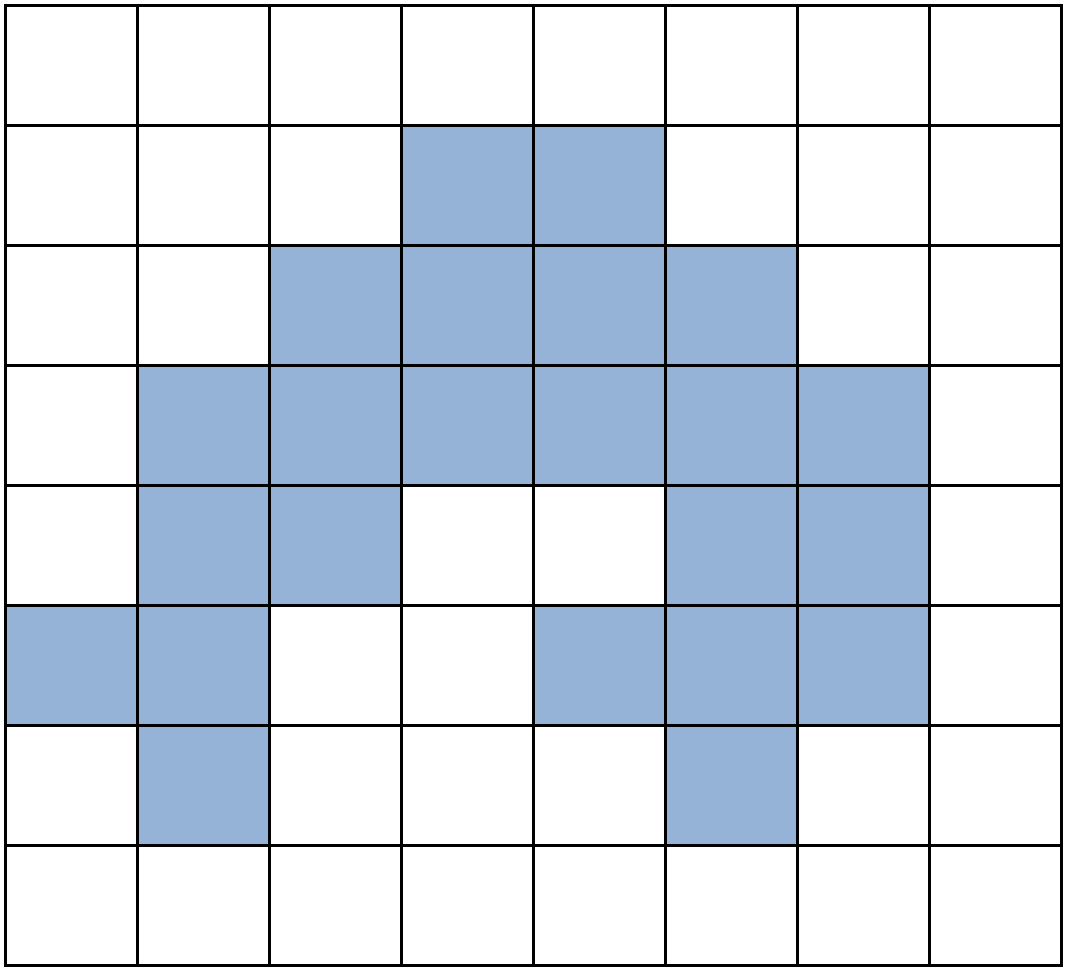
1: for pivo in pivoxels:
2: ecc(pivo) = max(df(pivo))

DF Algorithm on CUDA: 01

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

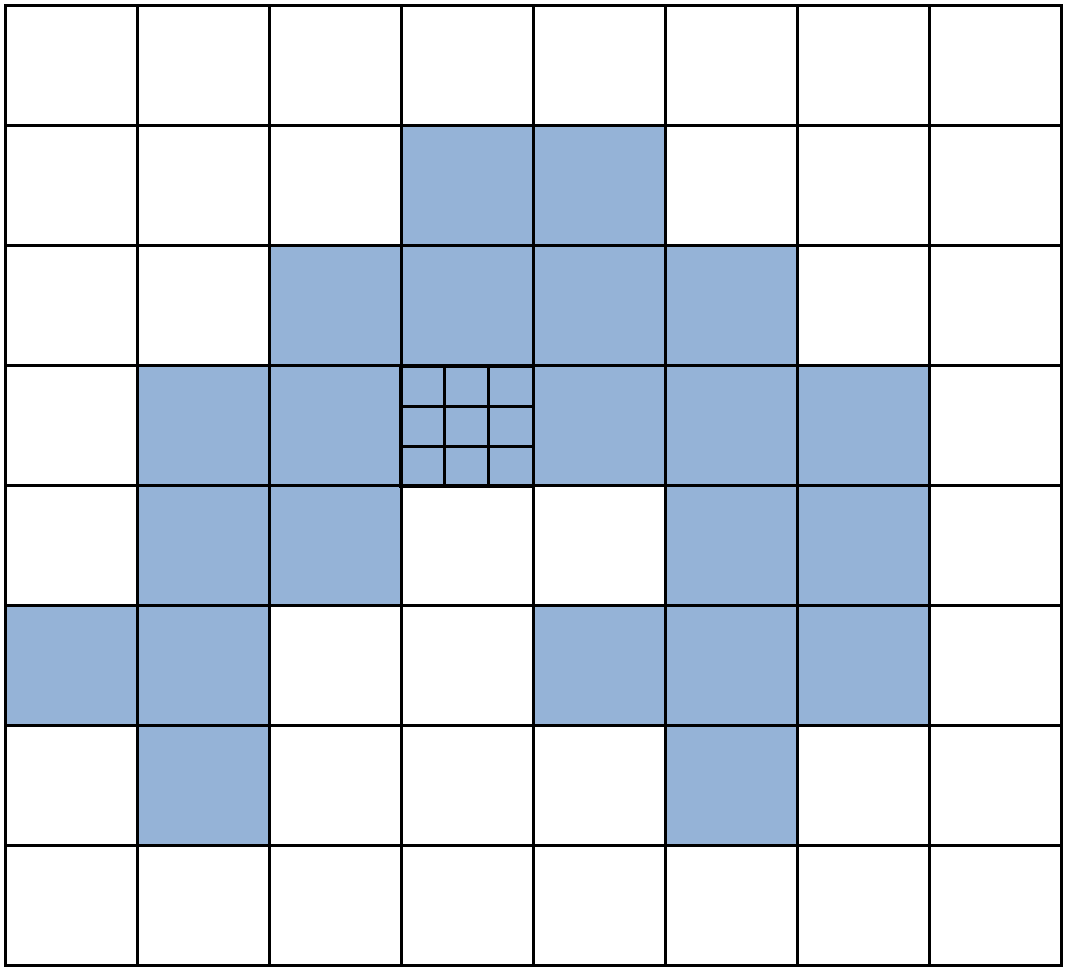


DF Algorithm on CUDA: 02

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



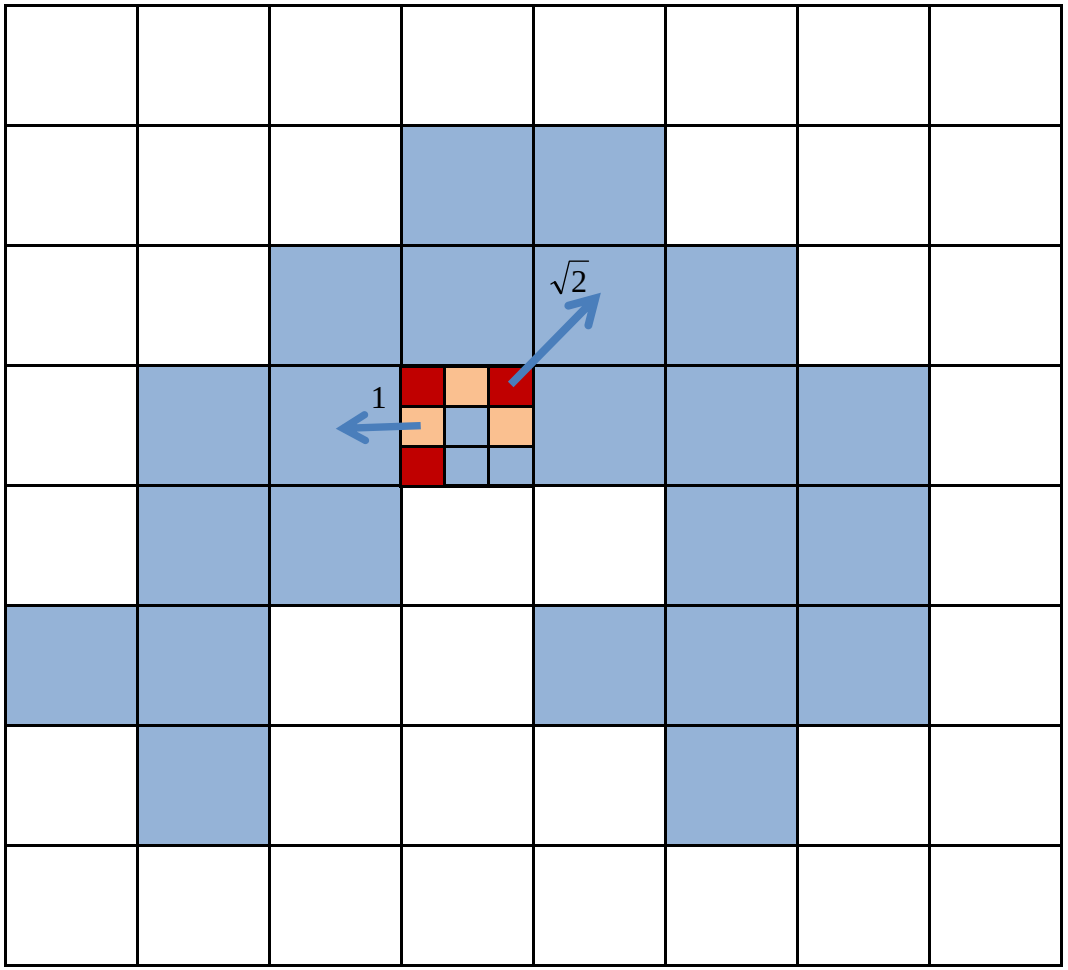
Euclidean distance metric -> 8 neighbours

DF Algorithm on CUDA: 02

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



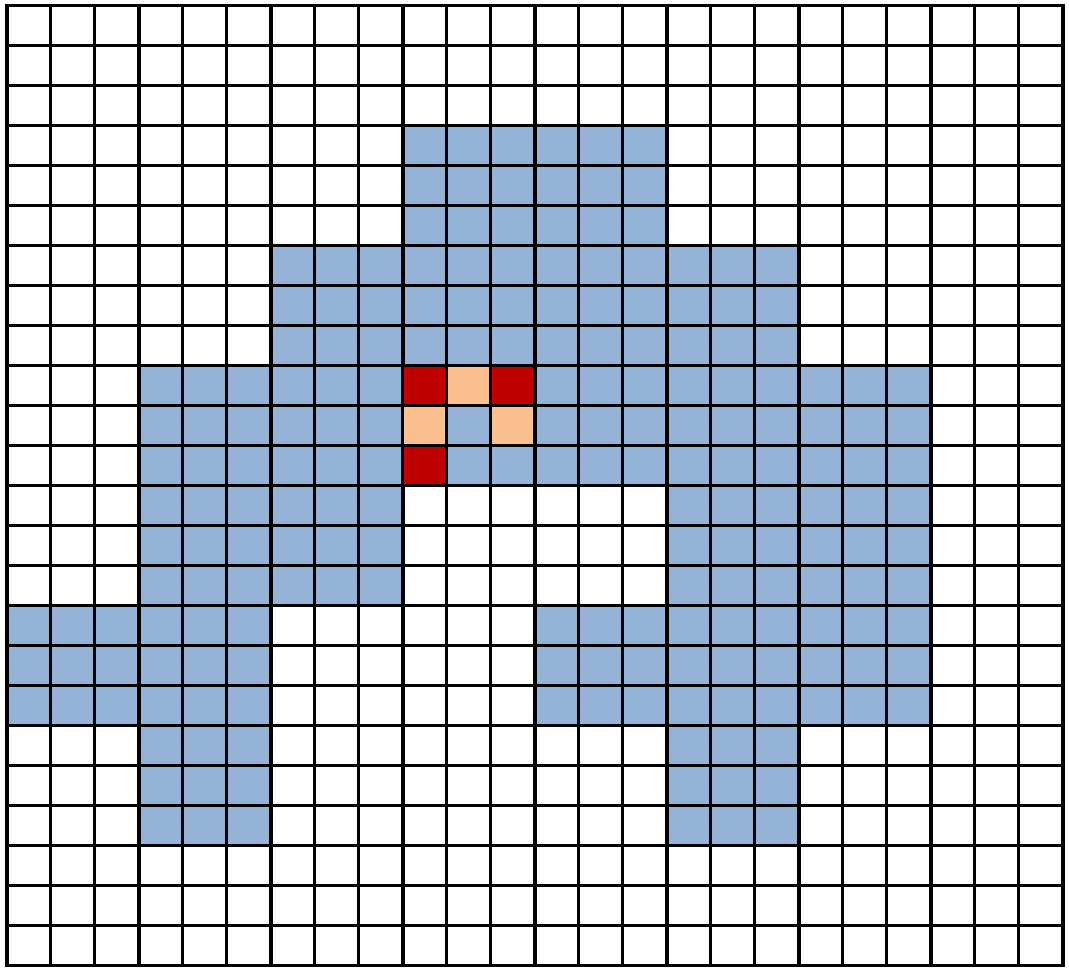
Weight is 1 to direct neighbour and √2 to diagonal

DF Algorithm on CUDA: 02

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

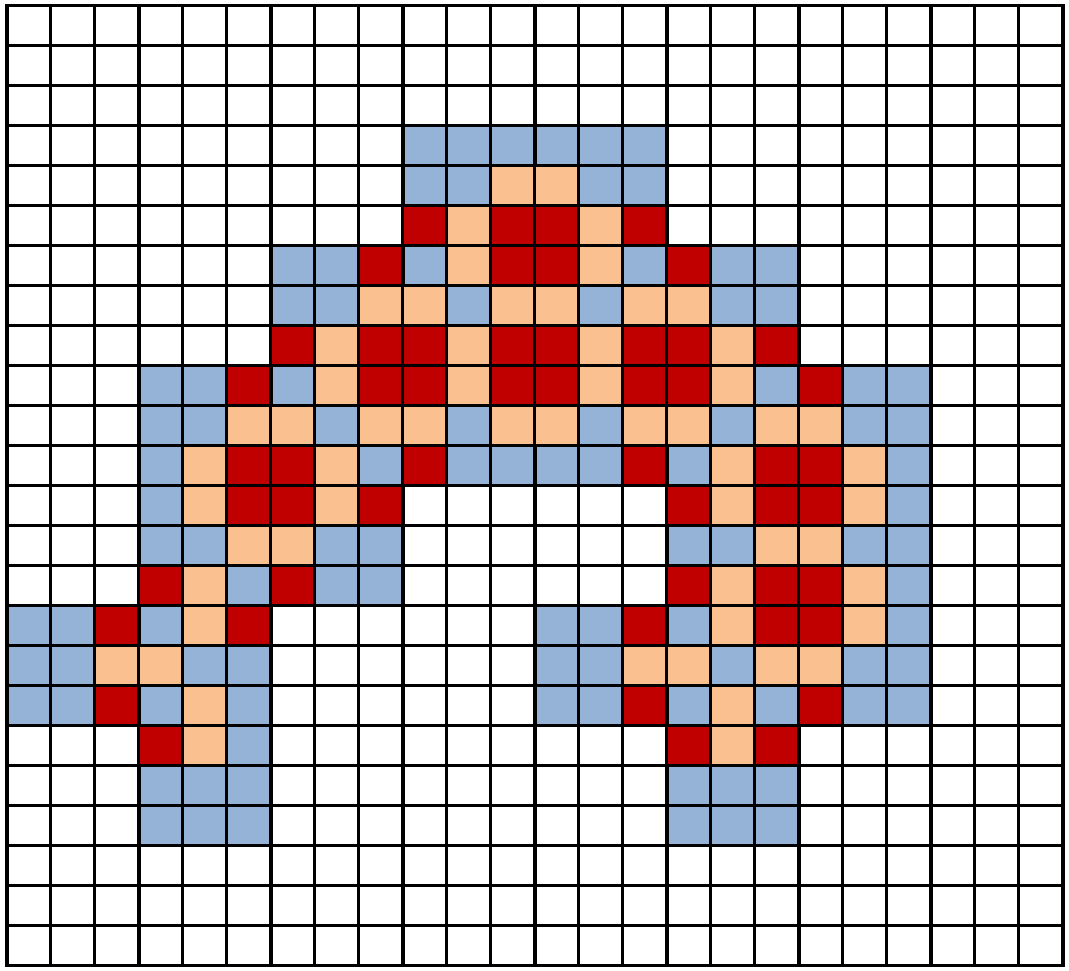


DF Algorithm on CUDA: 02

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 03

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 04

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 05

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 06

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 07

```
01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13:    k1(P,W,M,C,U)
14:    finished = false
15:    k2(P,W,M,C,U,finished)
16: while finished
```

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 08

```
01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13:    k1(P,W,M,C,U)
14:    finished = false
15:    k2(P,W,M,C,U,finished)
16: while finished
```

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 09

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**1**	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 10

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	**0**	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 11

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

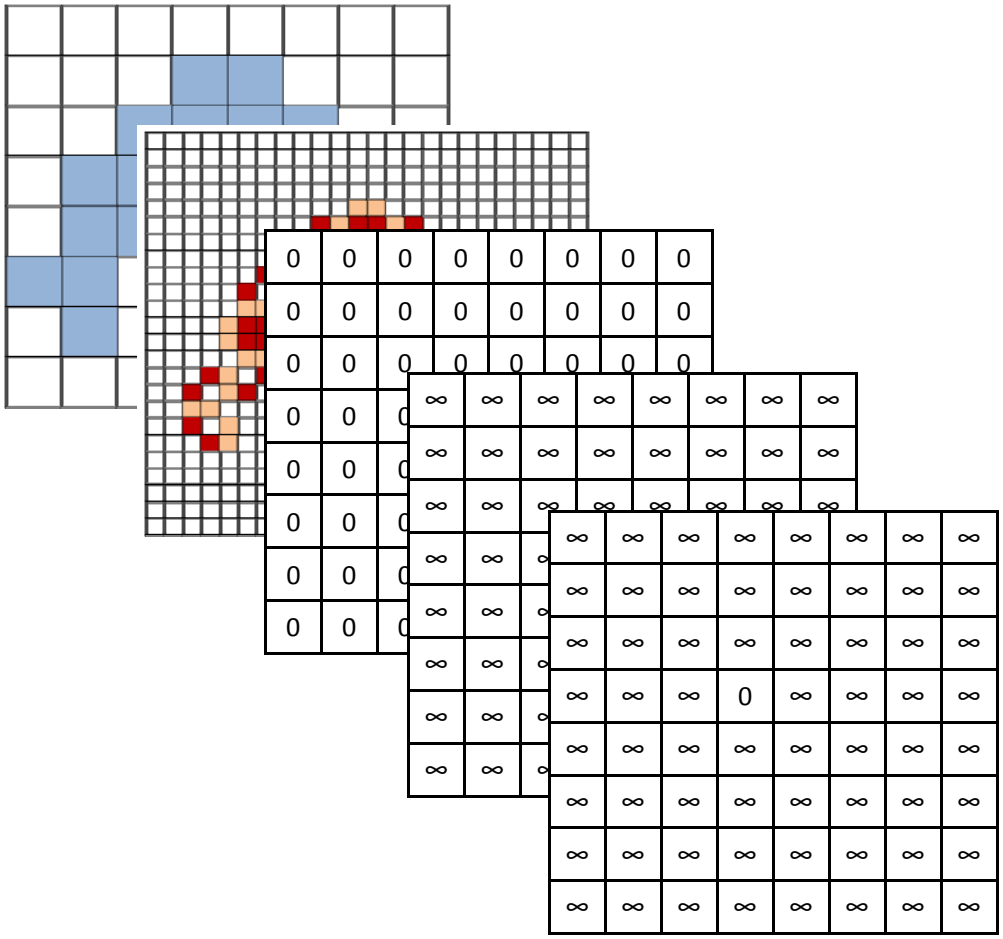
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	**0**	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 12

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 13

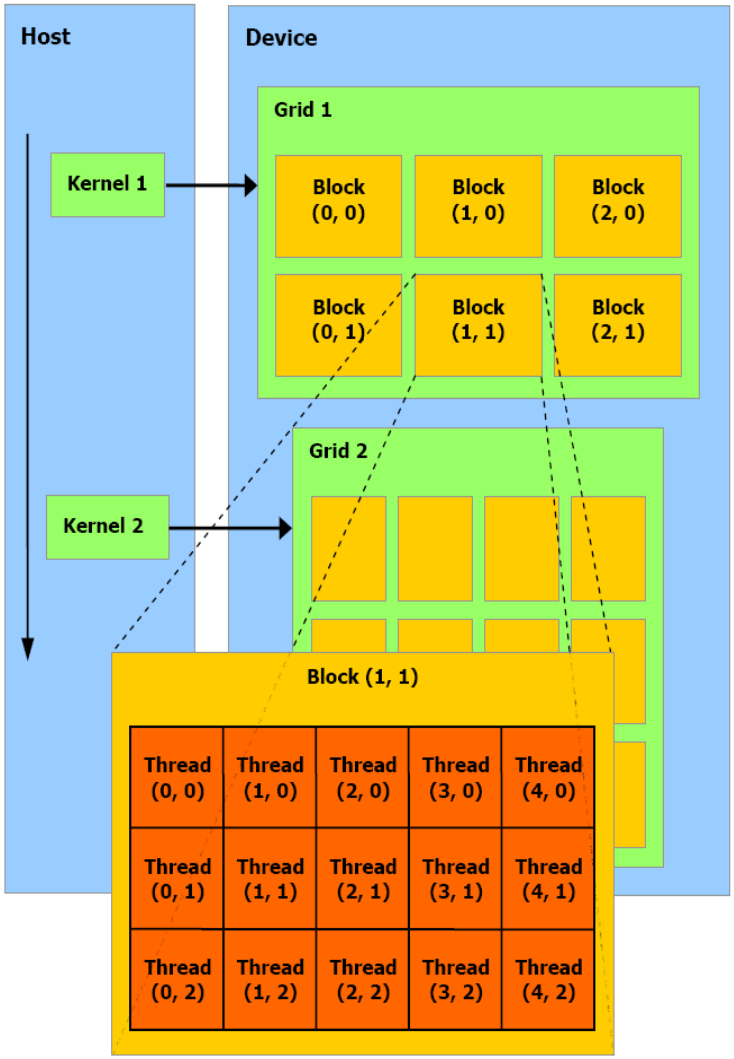
```
01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13:    k1(P,W,M,C,U)
14:    finished = false
15:    k2(P,W,M,C,U,finished)
16: while finished
```

k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()

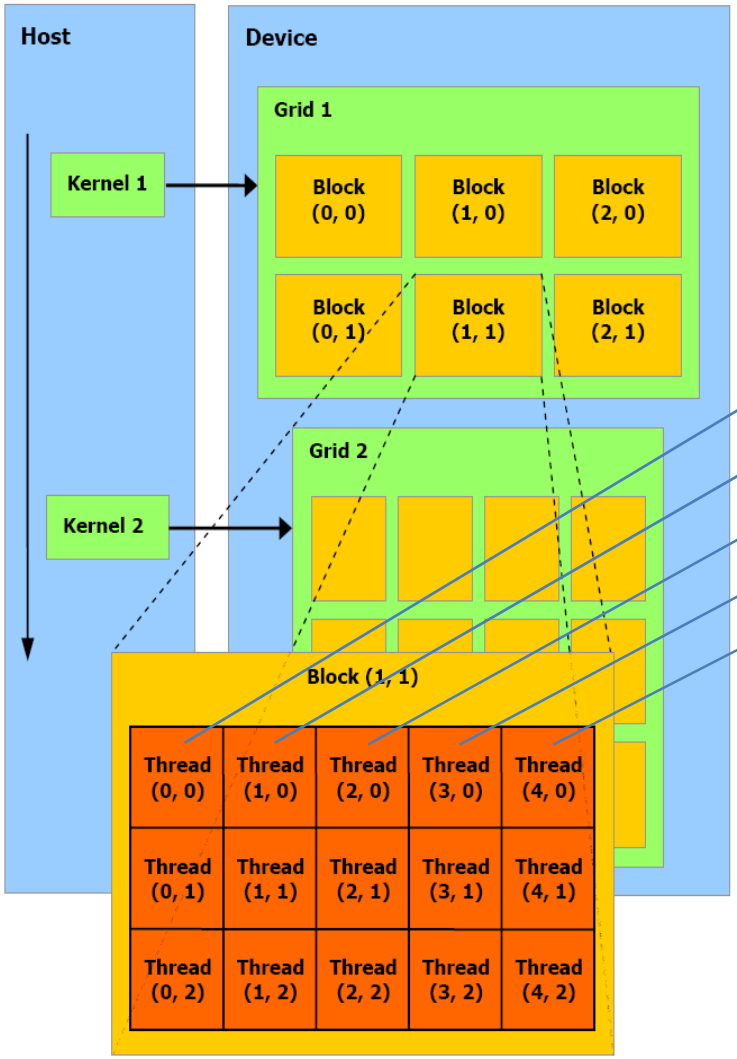
DF Algorithm on CUDA: 13



CUDA programming model

k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()

DF Algorithm on CUDA: 13



CUDA programming model

k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()

DF Algorithm on CUDA: 13

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()

DF Algorithm on CUDA: 13.1

```
01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U
```

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

```
12: do:
13:    k1(P,W,M,C,U)
14:    finished = false
15:    k2(P,W,M,C,U,finished)
16: while finished
```

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

DF Algorithm on CUDA: 13.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

M[0]	M[1]	M[2]	M[3]	M[4]	M[5]	M[6]	M[7]
M[8]	M[9]	M[10]	M[11]	M[12]	M[13]	M[14]	M[15]
M[16]	M[17]	M[18]	M[19]	M[20]	M[21]	M[22]	M[23]
M[24]	M[25]	M[26]	M[27]	M[28]	M[29]	M[30]	M[31]
M[32]	M[33]	M[34]	M[35]	M[36]	M[37]	M[38]	M[39]
M[40]	M[41]	M[42]	M[43]	M[44]	M[45]	M[46]	M[47]
M[48]	M[49]	M[50]	M[51]	M[52]	M[53]	M[54]	M[55]
M[56]	M[57]	M[58]	M[59]	M[60]	M[61]	M[62]	M[63]

DF Algorithm on CUDA: 13.2

```
01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U
```

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

```
12: do:
13:   k1(P,W,M,C,U)
14:   finished = false
15:   k2(P,W,M,C,U,finished)
16: while finished
```

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 13.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	1	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

DF Algorithm on CUDA: 13.3

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

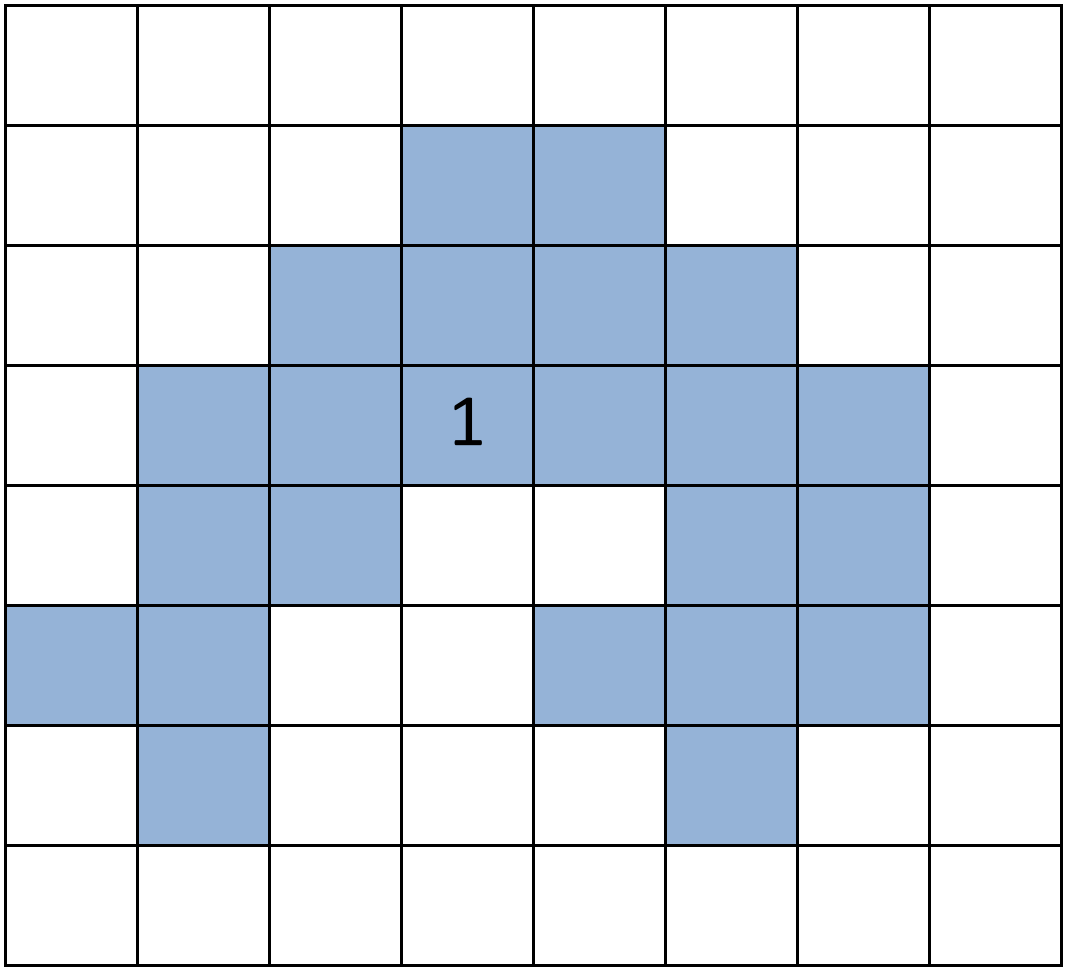
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**0**	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 13.4

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

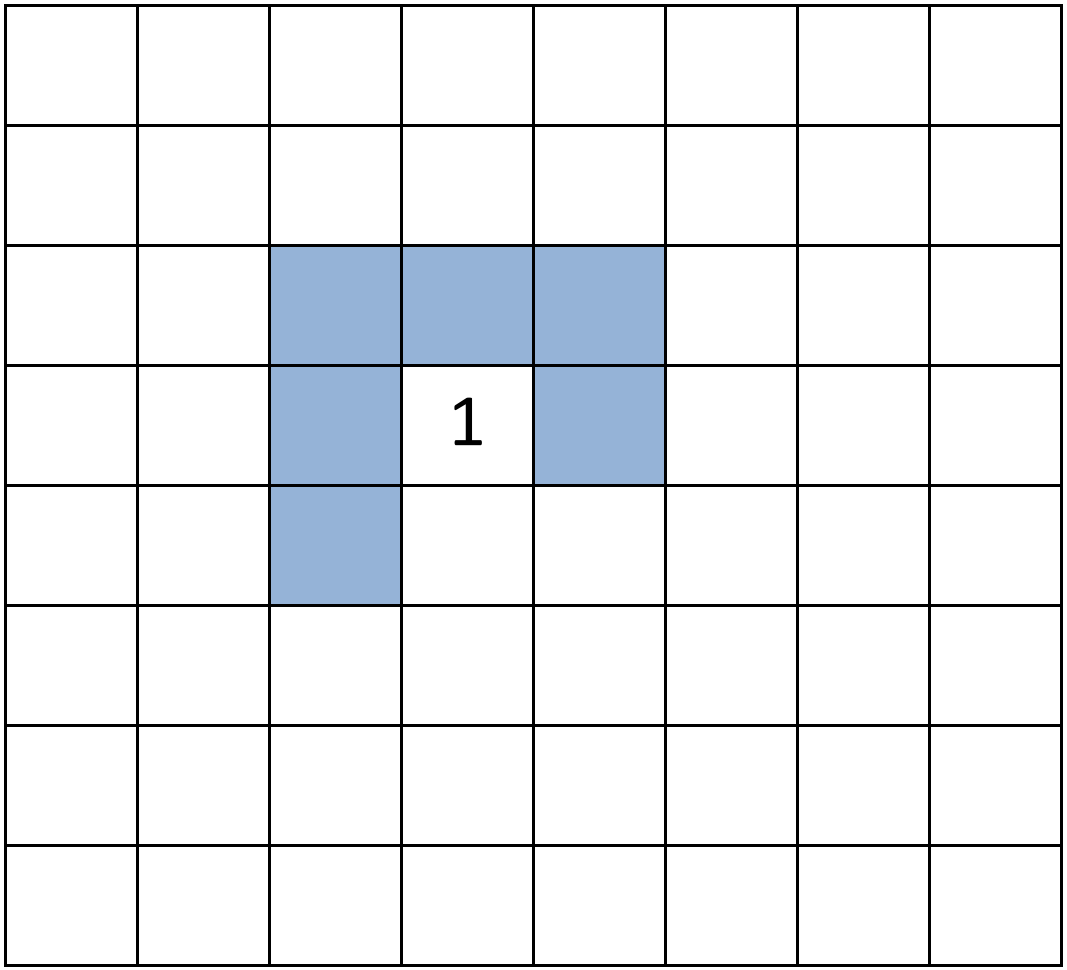


DF Algorithm on CUDA: 13.4

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 13.4.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

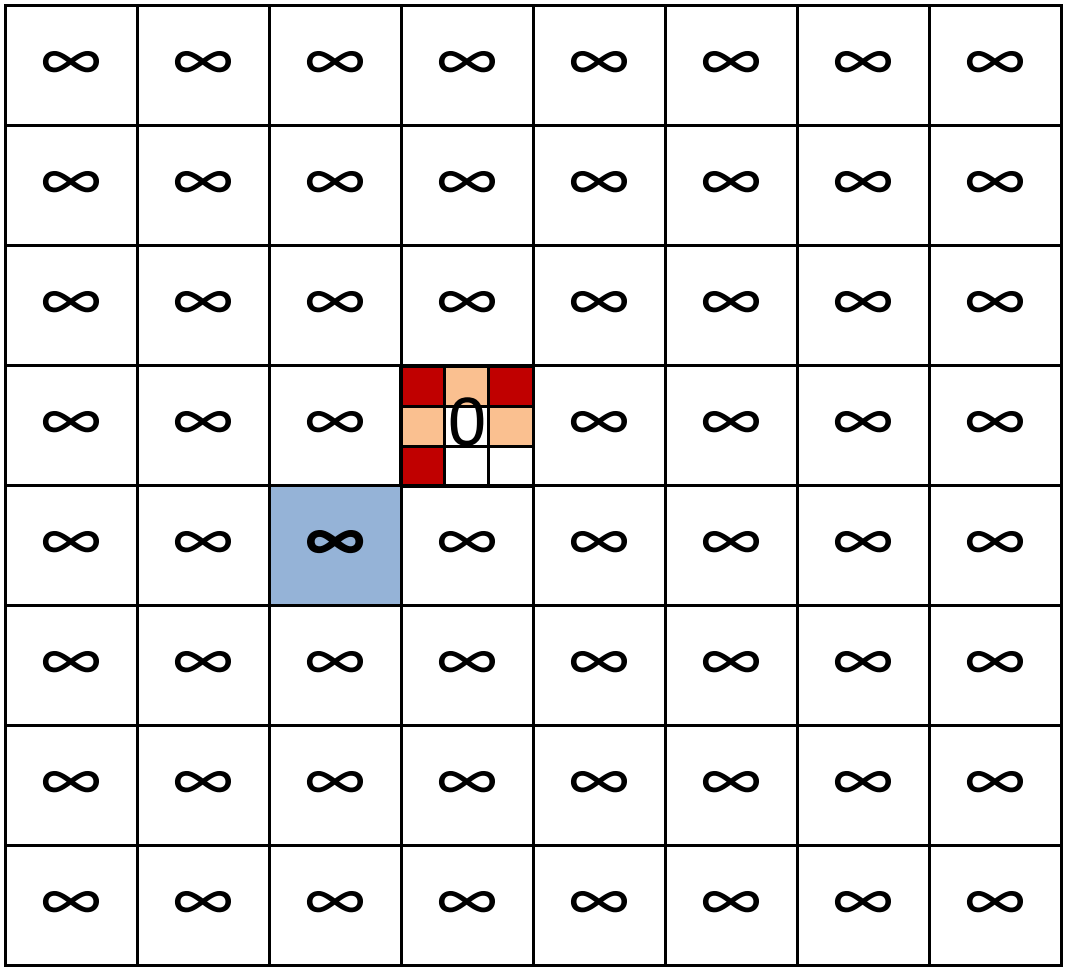
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 13.5.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 13.6.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	∞	∞	∞	∞
∞	∞	141	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 13.6.6

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

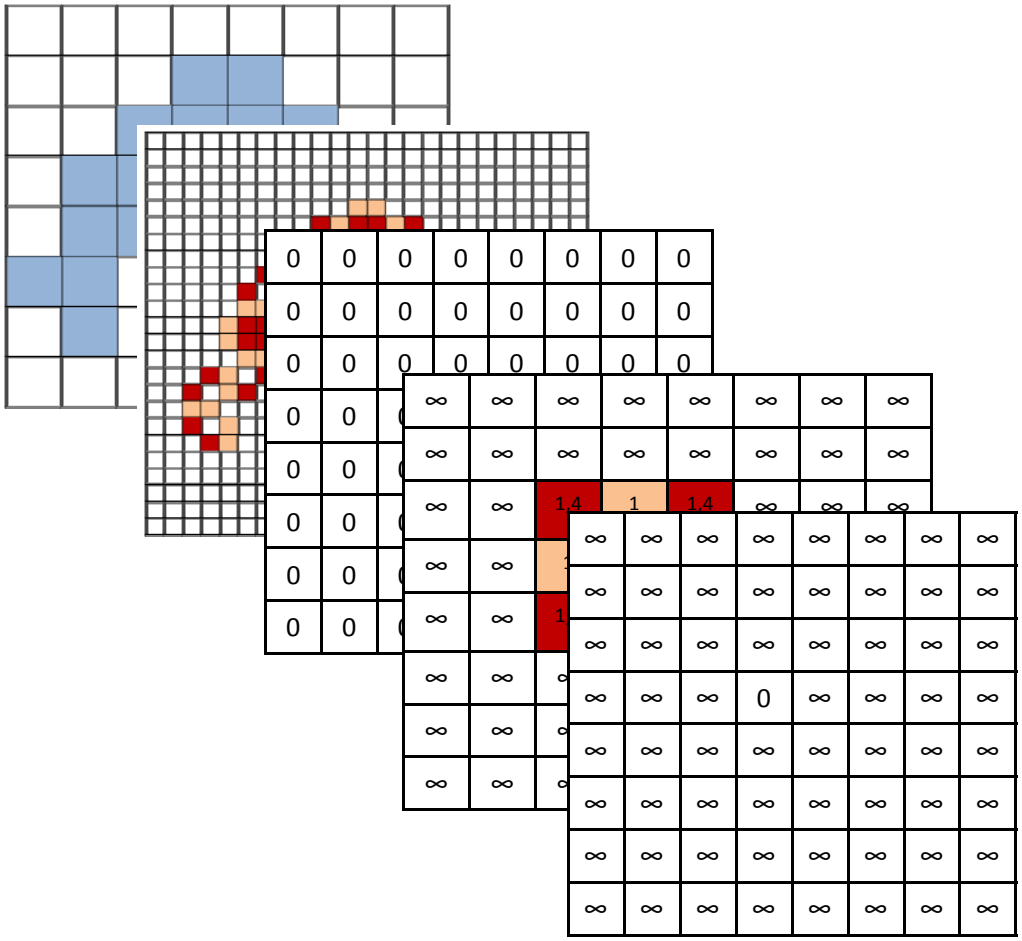
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	1,41	1	1,41	∞	∞	∞
∞	∞	1	0	1	∞	∞	∞
∞	∞	1,41	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 14

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 15

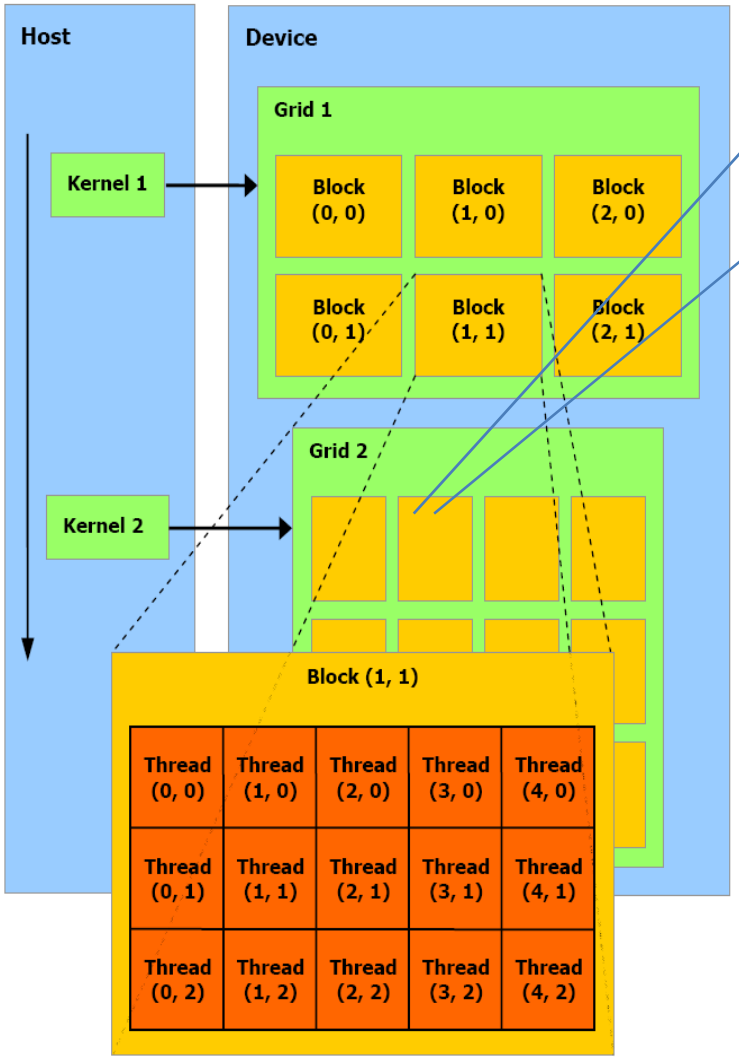
```
01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13:    k1(P,W,M,C,U)
14:    finished = false
15:    k2(P,W,M,C,U,finished)
16: while finished
```

k1()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()

DF Algorithm on CUDA: 15



CUDA programming model

k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()

DF Algorithm on CUDA: 15

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

```
15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3:     C[tid] = U[tid]
15.4:     M[tid] = true
15.5:     finished = true
15.6: U[tid] = C[tid]
```

```
12: do:
13:     k1(P,W,M,C,U)
14:     finished = false
15:     k2(P,W,M,C,U,finished)
16: while finished
```

k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()

DF Algorithm on CUDA: 15.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

DF Algorithm on CUDA: 15.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]
C[8]	C[9]	C[10]	C[11]	C[12]	C[13]	C[14]	C[15]
C[16]	C[17]	C[18]	C[19]	C[20]	C[21]	C[22]	C[23]
C[24]	C[25]	C[26]	C[27]	C[28]	C[29]	C[30]	C[31]
C[32]	C[33]	C[34]	C[35]	C[36]	C[37]	C[38]	C[39]
C[40]	C[41]	C[42]	C[43]	C[44]	C[45]	C[46]	C[47]
C[48]	C[49]	C[50]	C[51]	C[52]	C[53]	C[54]	C[55]
C[56]	C[57]	C[58]	C[59]	C[60]	C[61]	C[62]	C[63]

DF Algorithm on CUDA: 15.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 15.3

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	1,41	1	1,41	∞	∞	∞
∞	∞	1	0	1	∞	∞	∞
∞	∞	1,41	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 15.4

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	**1**	**1**	**1**	0	0	0
0	0	**1**	0	**1**	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 15.5

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	**1**	**1**	**1**	0	0	0
0	0	**1**	0	**1**	0	0	0
0	0	**1**	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 15.6

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

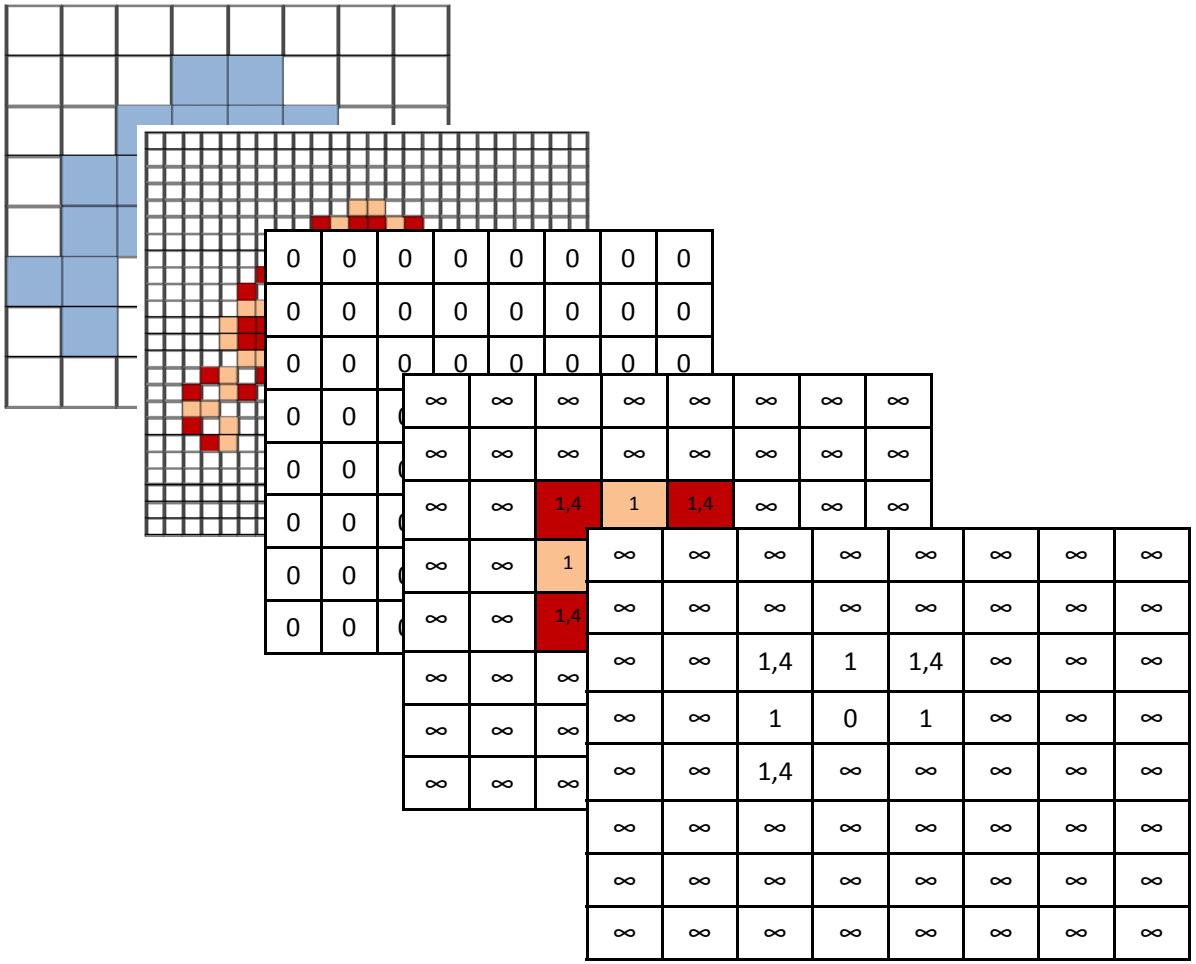
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	1,41	1	1,41	∞	∞	∞
∞	∞	1	0	1	∞	∞	∞
∞	∞	1,41	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 16

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 13.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()
k1()	k1()	k1()	k1()	k1()	k1()	k1()	k1()

DF Algorithm on CUDA: 13.2.1

```
01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U
```

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

```
12: do:
13:    k1(P,W,M,C,U)
14:    finished = false
15:    k2(P,W,M,C,U,finished)
16: while finished
```

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

DF Algorithm on CUDA: 13.2.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

M[0]	M[1]	M[2]	M[3]	M[4]	M[5]	M[6]	M[7]
M[8]	M[9]	M[10]	M[11]	M[12]	M[13]	M[14]	M[15]
M[16]	M[17]	M[18]	M[19]	M[20]	M[21]	M[22]	M[23]
M[24]	M[25]	M[26]	M[27]	M[28]	M[29]	M[30]	M[31]
M[32]	M[33]	M[34]	M[35]	M[36]	M[37]	M[38]	M[39]
M[40]	M[41]	M[42]	M[43]	M[44]	M[45]	M[46]	M[47]
M[48]	M[49]	M[50]	M[51]	M[52]	M[53]	M[54]	M[55]
M[56]	M[57]	M[58]	M[59]	M[60]	M[61]	M[62]	M[63]

DF Algorithm on CUDA: 13.2.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0
0	0	1	0	1	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 13.2.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	1	1	1	-	-	-
-	-	1	-	1	-	-	-
-	-	1	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

DF Algorithm on CUDA: 13.2.3

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

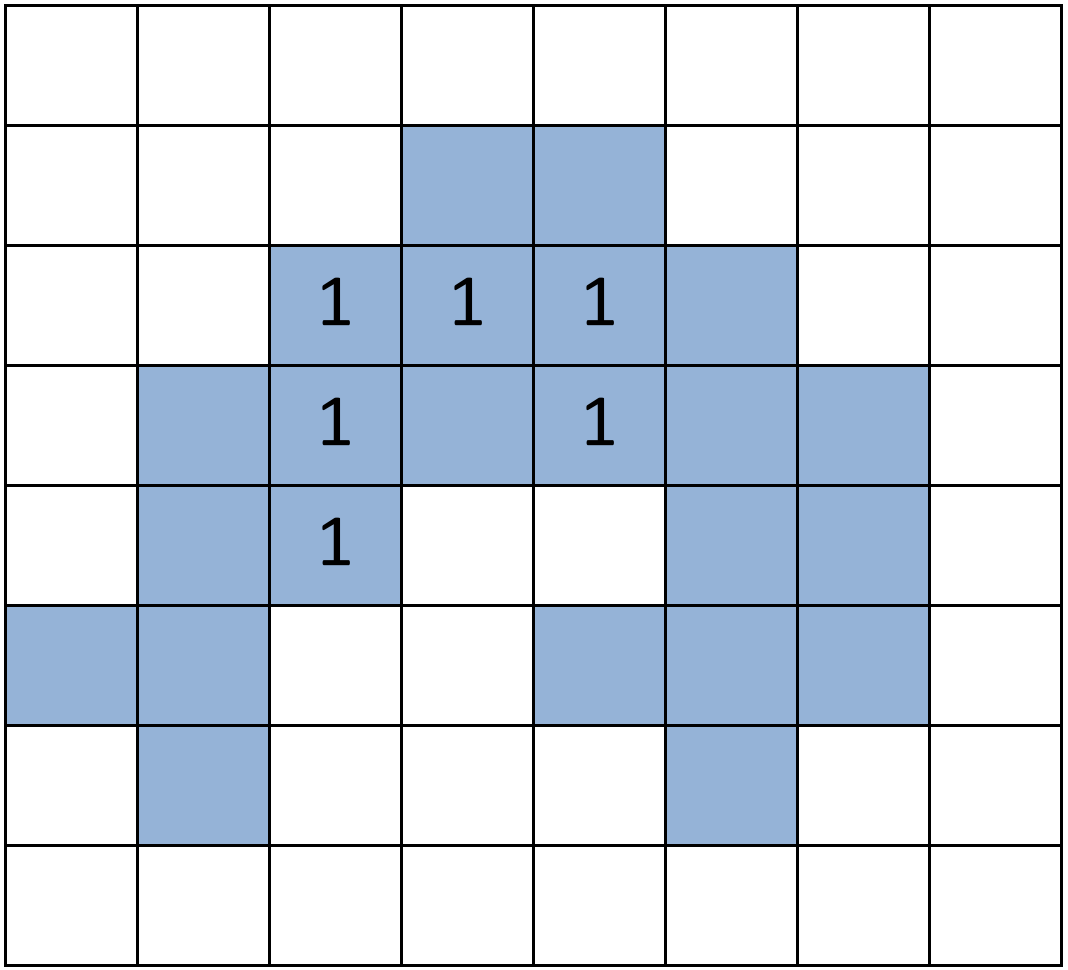
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	**0**	**0**	**0**	0	0	0
0	0	**0**	0	**0**	0	0	0
0	0	**0**	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 13.2.4

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

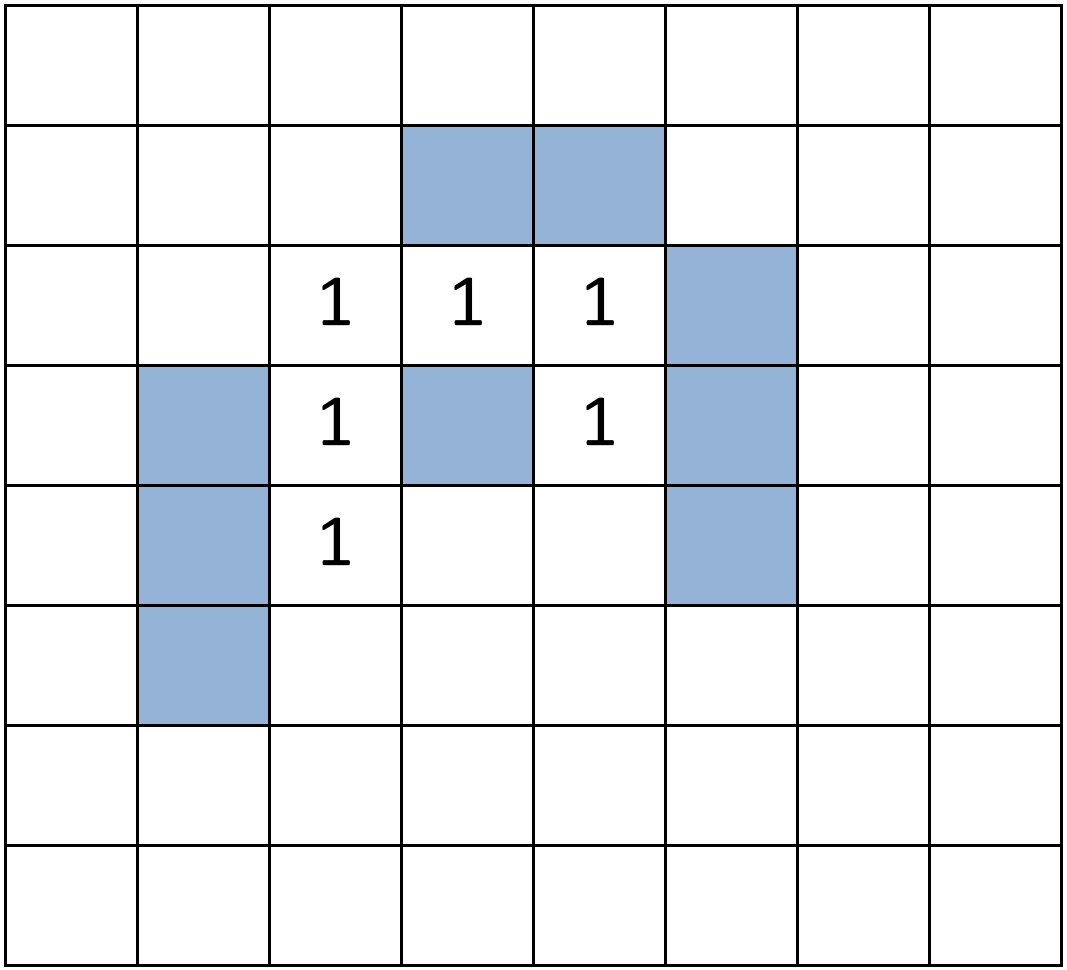


DF Algorithm on CUDA: 13.2.4

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 13.2.4.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

13.1: tid = getThreadID()
13.2: if M[tid]:
13.3: M[tid] = false
13.4: for nid in tid.neighbors():
13.5: if U[nid] > C[tid] + W[nid]:
13.6: U[nid] = C[tid] + W[nid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	1.41	1	1.41	∞	∞	∞
∞	∞	1	0	1	∞	∞	∞
∞	∞	1.41	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 13.2.6

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

```
13.1: tid = getThreadID()
13.2: if M[tid]:
13.3:   M[tid] = false
13.4:   for nid in tid.neighbors():
13.5:     if U[nid] > C[tid] + W[nid]:
13.6:       U[nid] = C[tid] + W[nid]
```

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

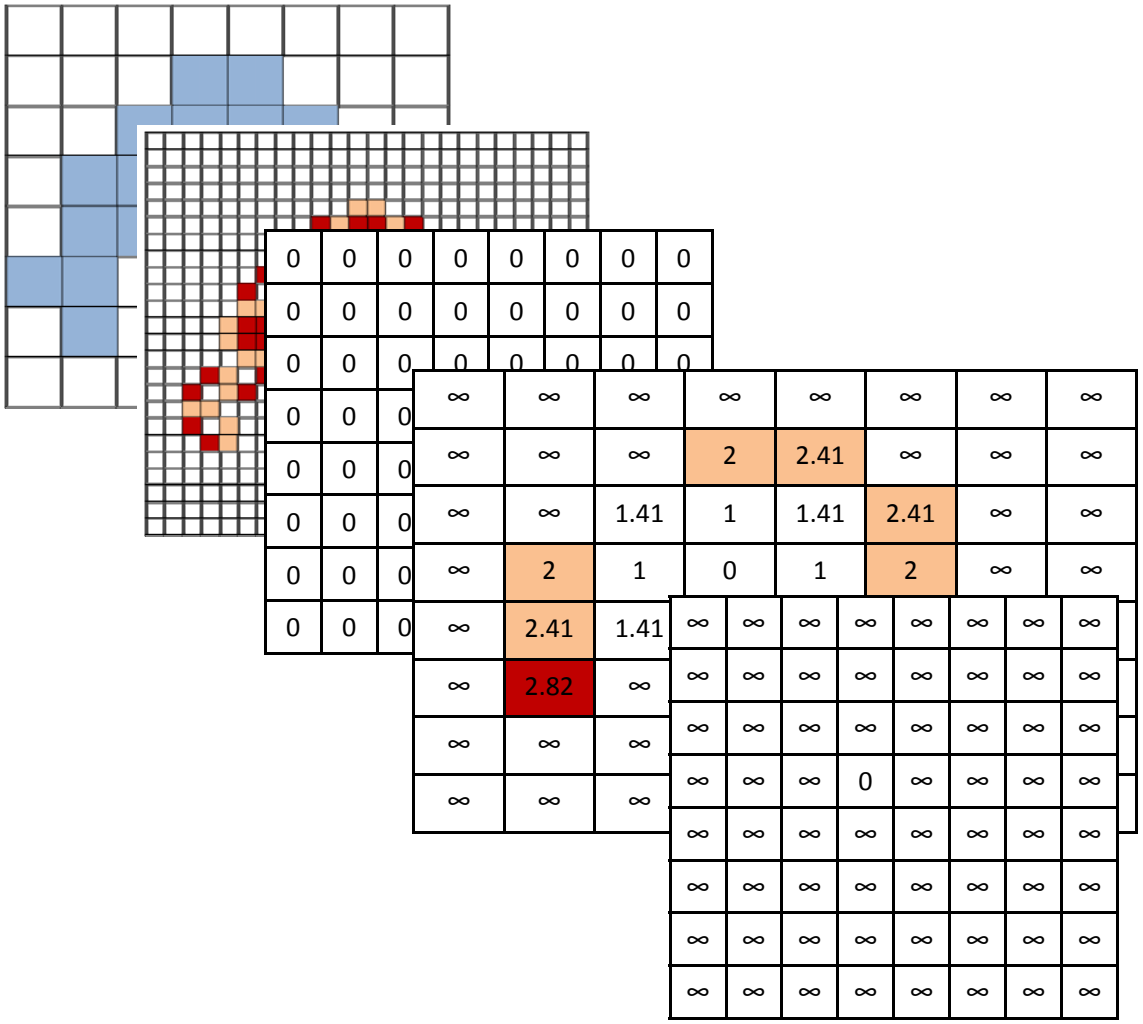
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	2	2.41	∞	∞	∞
∞	∞	1.41	1	1.41	2.41	∞	∞
∞	2	1	0	1	2	∞	∞
∞	2.41	1.41	∞	∞	2.41	∞	∞
∞	2.82	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 14.2

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished



DF Algorithm on CUDA: 15.2.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

```
15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3:     C[tid] = U[tid]
15.4:     M[tid] = true
15.5:     finished = true
15.6: U[tid] = C[tid]
```

```
12: do:
13:     k1(P,W,M,C,U)
14:     finished = false
15:     k2(P,W,M,C,U,finished)
16: while finished
```

k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()
k2()	k2()	k2()	k2()	k2()	k2()	k2()	k2()

DF Algorithm on CUDA: 15.2.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

DF Algorithm on CUDA: 15.2.1

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]
C[8]	C[9]	C[10]	C[11]	C[12]	C[13]	C[14]	C[15]
C[16]	C[17]	C[18]	C[19]	C[20]	C[21]	C[22]	C[23]
C[24]	C[25]	C[26]	C[27]	C[28]	C[29]	C[30]	C[31]
C[32]	C[33]	C[34]	C[35]	C[36]	C[37]	C[38]	C[39]
C[40]	C[41]	C[42]	C[43]	C[44]	C[45]	C[46]	C[47]
C[48]	C[49]	C[50]	C[51]	C[52]	C[53]	C[54]	C[55]
C[56]	C[57]	C[58]	C[59]	C[60]	C[61]	C[62]	C[63]

DF Algorithm on CUDA: 15.2.3

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	**1,41**	1	**1,41**	∞	∞	∞
∞	∞	1	0	1	∞	∞	∞
∞	∞	**1,41**	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 15.2.3

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	2	2.41	∞	∞	∞
∞	∞	1.41	1	1.41	2.41	∞	∞
∞	2	1	0	1	2	∞	∞
∞	2.41	1.41	∞	∞	2.41	∞	∞
∞	2.82	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: 15.2.4

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	**1**	**1**	0	0	0
0	0	0	0	0	**1**	0	0
0	**1**	0	0	0	**1**	0	0
0	**1**	0	0	0	**1**	0	0
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 15.2.5

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

0	0	0	0	0	0	0	0
0	0	0	**1**	**1**	0	0	0
0	0	0	0	0	**1**	0	0
0	**1**	0	0	0	**1**	0	0
0	**1**	0	0	0	**1**	0	0
0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DF Algorithm on CUDA: 15.2.6

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

15.1: tid=getThreadID()
15.2: if C[tid] > U[tid]:
15.3: C[tid] = U[tid]
15.4: M[tid] = true
15.5: finished = true
15.6: U[tid] = C[tid]

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	2	2.41	∞	∞	∞
∞	∞	1.41	1	1.41	2.41	∞	∞
∞	2	1	0	1	2	∞	∞
∞	2.41	1.41	∞	∞	2.41	∞	∞
∞	2.82	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

DF Algorithm on CUDA: Result

01: Pixel field ~ P
02: Weight field ~ W
03: Mask field ~ M
04: Cost field ~ C
05: Updating cost field ~ U

06: M[:] = 0
07: C[:] = Inf.
08: U[:] = Inf.
09: M[S] = 1
10: C[S] = 0
11: U[S] = 0

12: do:
13: k1(P,W,M,C,U)
14: finished = false
15: k2(P,W,M,C,U,finished)
16: while finished

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	2	2.41	∞	∞	∞
∞	∞	1.41	1	1.41	2.41	∞	∞
∞	2	1	0	1	2	3	∞
∞	2.41	1.41	∞	∞	2.41	3.41	∞
3.82	2.82	∞	∞	3.82	3.41	3.82	∞
∞	3.82	∞	∞	∞	4.41	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

The Eccentricity Transform via DF

1: for pivo in pivoxels:
2: ecc(pivo) = **max(df(pivo))**

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	2	2.41	∞	∞	∞
∞	∞	1.41	1	1.41	2.41	∞	∞
∞	2	1	0	1	2	3	∞
∞	2.41	1.41	∞	∞	2.41	3.41	∞
3.82	2.82	∞	∞	3.82	3.41	3.82	∞
∞	3.82	∞	∞	∞	**4.41**	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

The Eccentricity Transform via DF

1: for pivo in pivoxels:

2: ecc(pivo) = max(df(pivo))

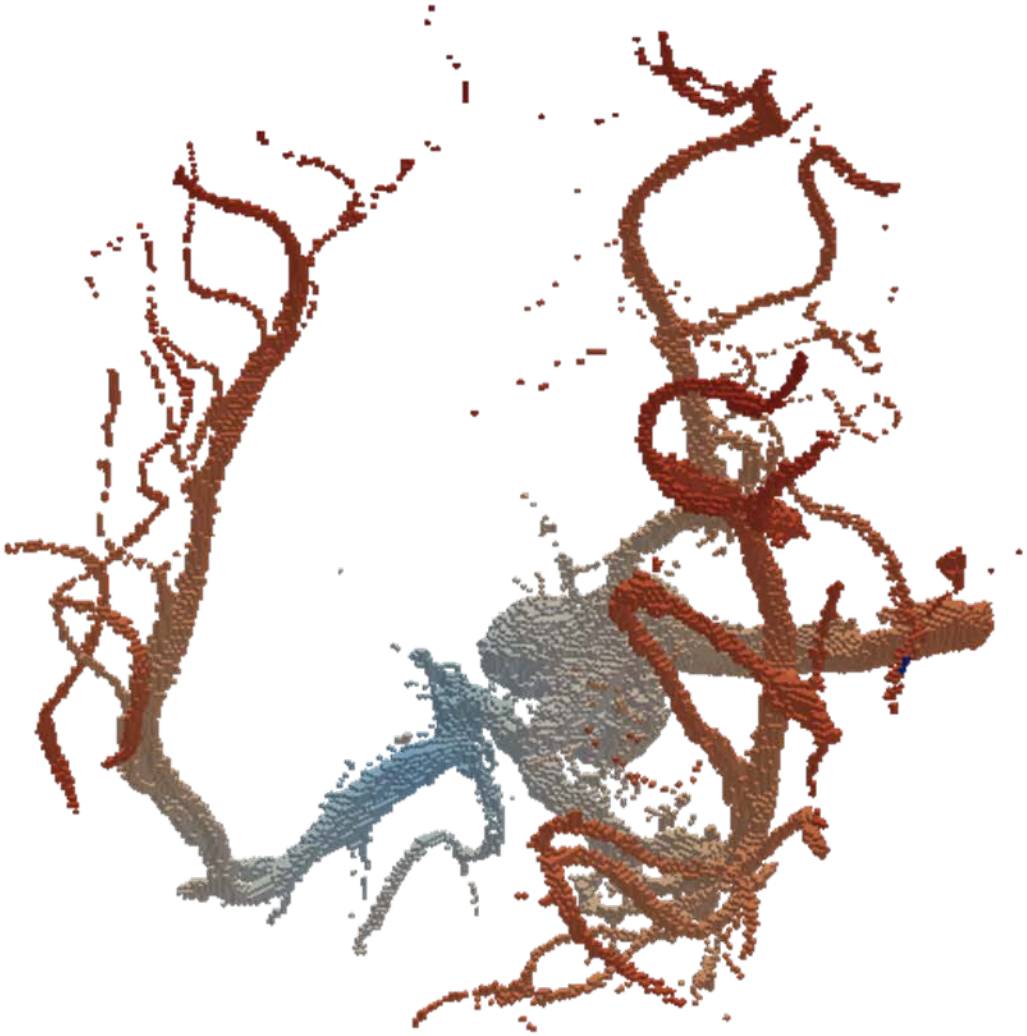
∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	0	∞	∞	∞
∞	∞	0	0	0	0	∞	∞
∞	0	0	**4.41**	0	0	0	∞
∞	0	0	∞	∞	0	0	∞
0	0	∞	∞	0	0	0	∞
∞	0	∞	∞	∞	**0**	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

The Eccentricity Transform: Result

1: for pivo in pivoxels:
2: ecc(pivo) = max(df(pivo))

∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	5.8	6.2	∞	∞	∞
∞	∞	5.8	4.8	5.2	6.2	∞	∞
∞	6.4	5.4	4.4	4.8	5.8	6.8	∞
∞	6.8	5.8	∞	∞	6.2	7.2	∞
8.2	7.2	∞	∞	7.6	7.2	7.6	∞
∞	8.2	∞	∞	∞	**8.2**	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞

The Eccentricity Transform in 3D: Result



Optimizations of ET

- Compute distance field of any given point (non-extremal) and find local maxima
- Compute DF only from those local maximas
- Maximum at each (pi/vo)xel should be value of ET

Conclusion

- Eccentricity transform (ET) can be adapted to CUDA
- Optimizations of ET possible based eccentric points
- ET has global character
- ET is robust against noise