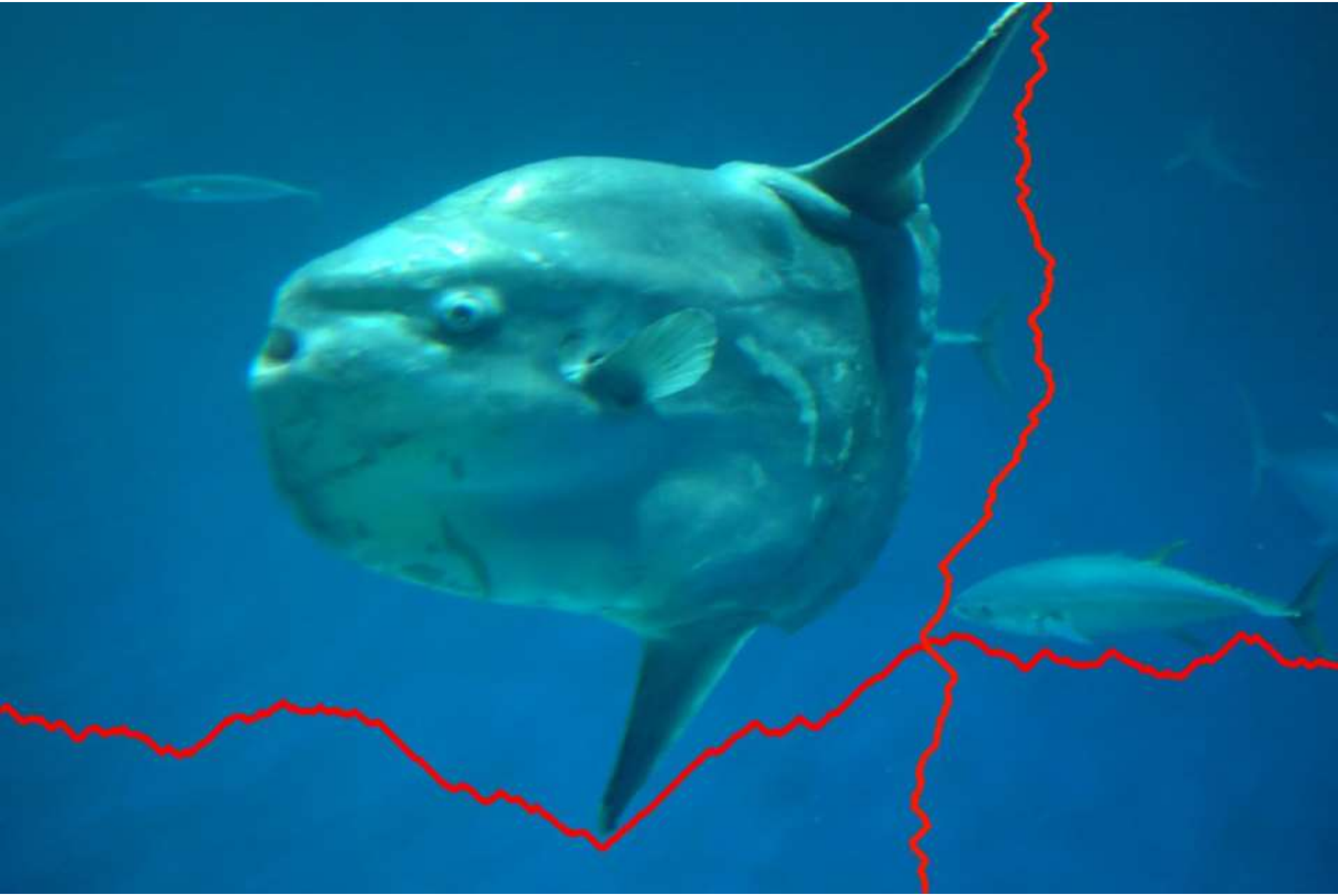
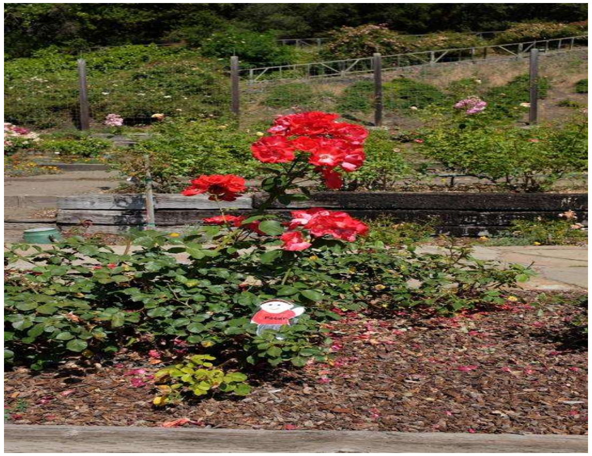


$820 \times 546 \times 3$

$420 \times 546 \times 3$



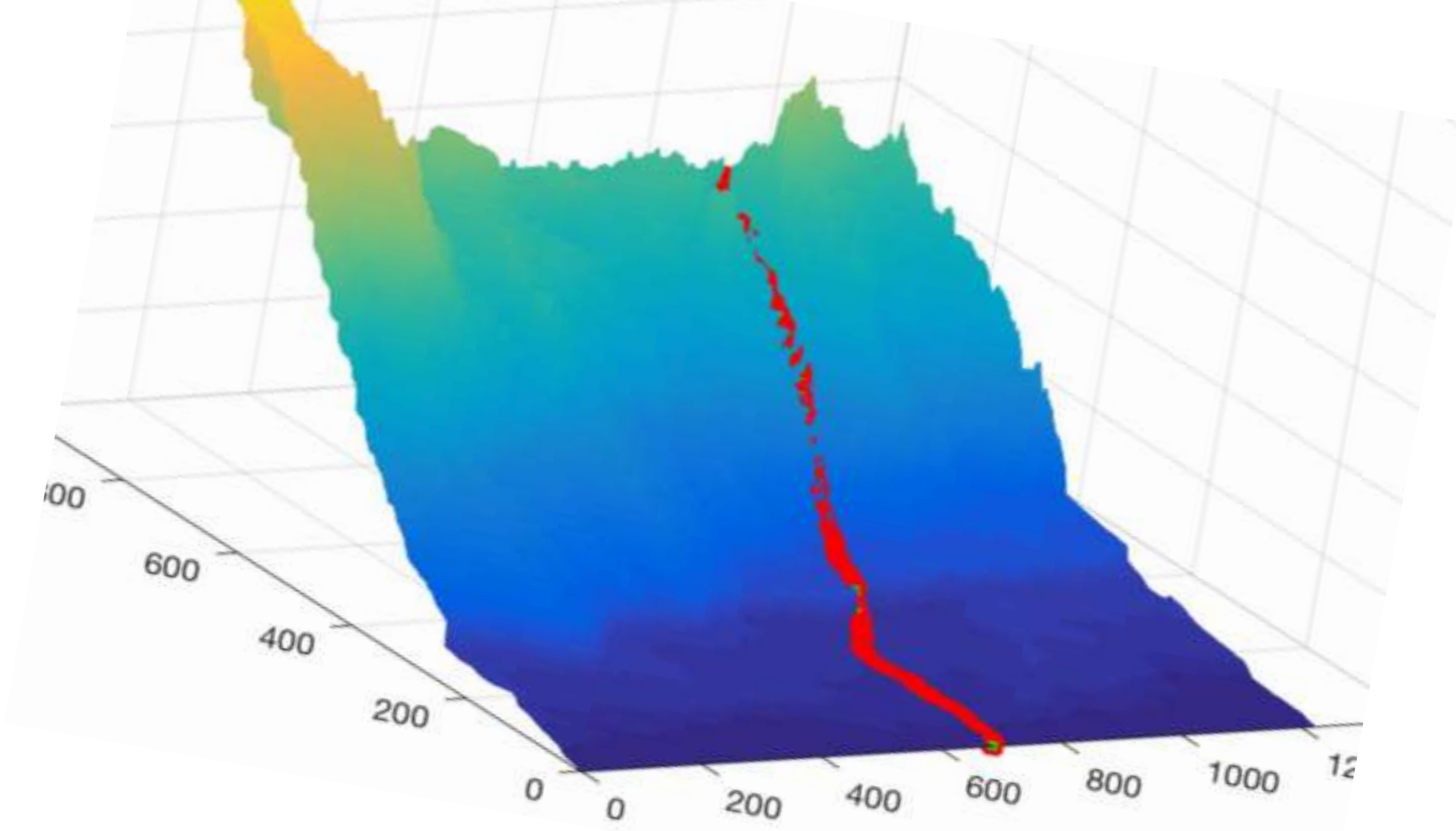
(a)

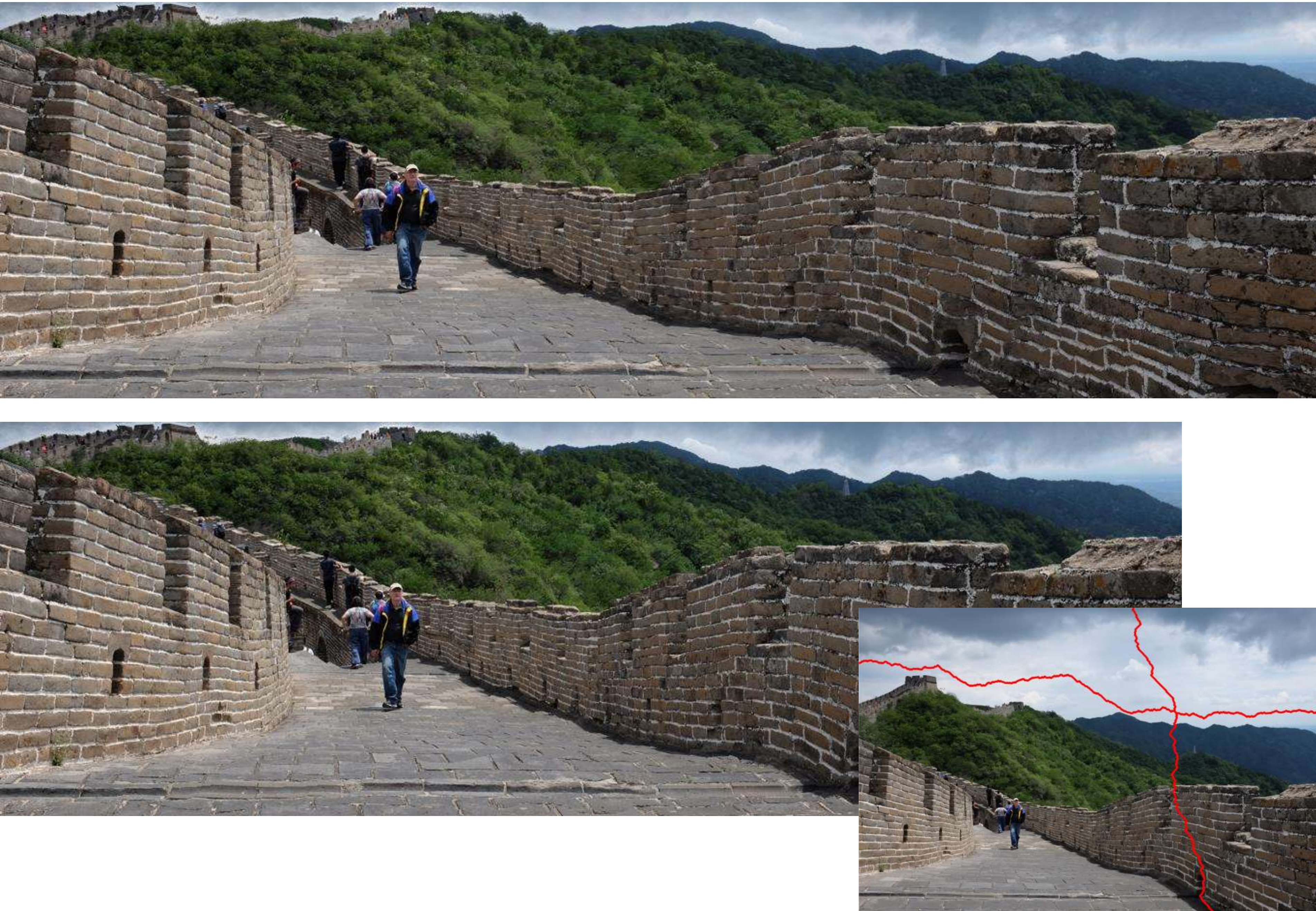
(b)

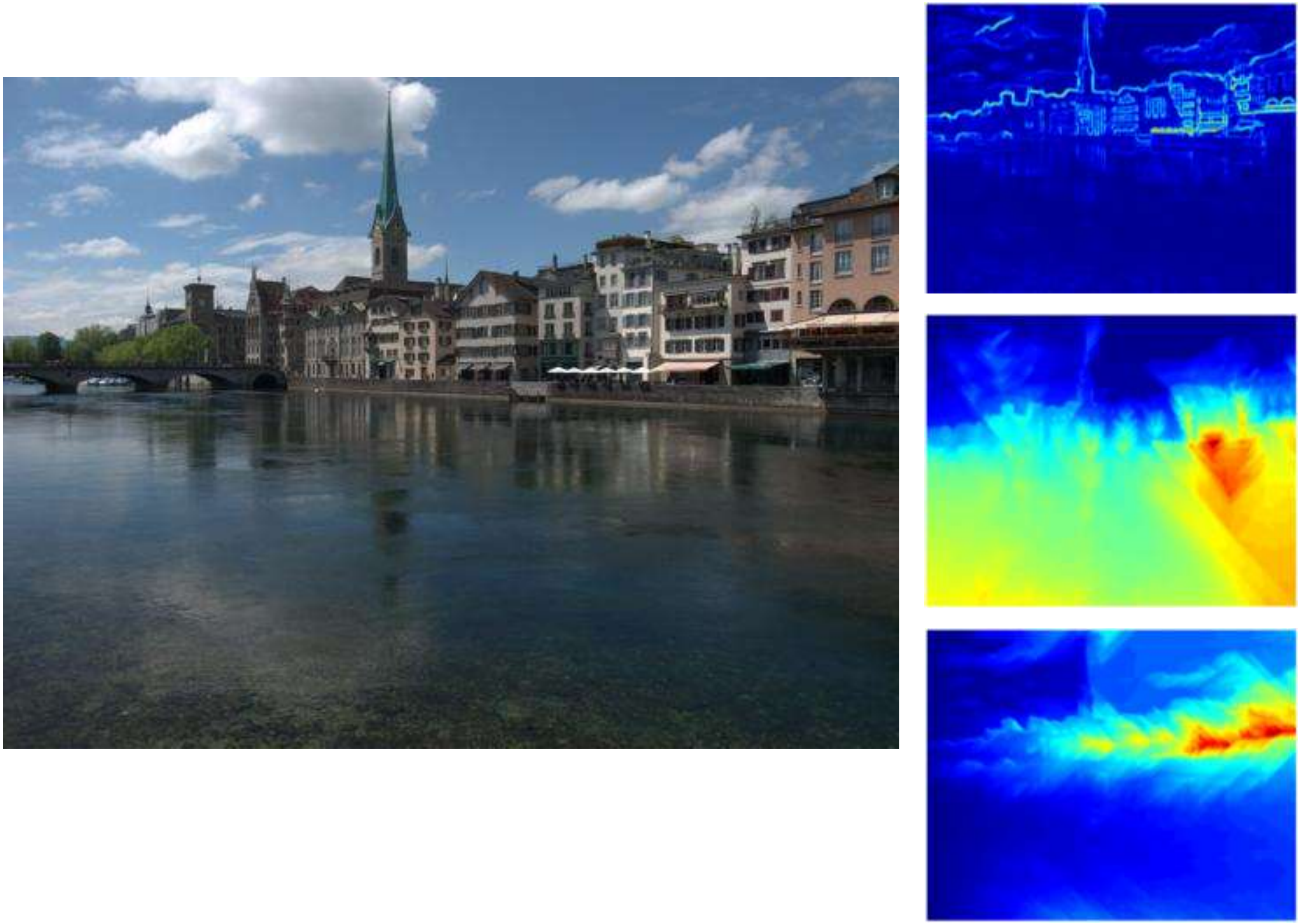
(c)

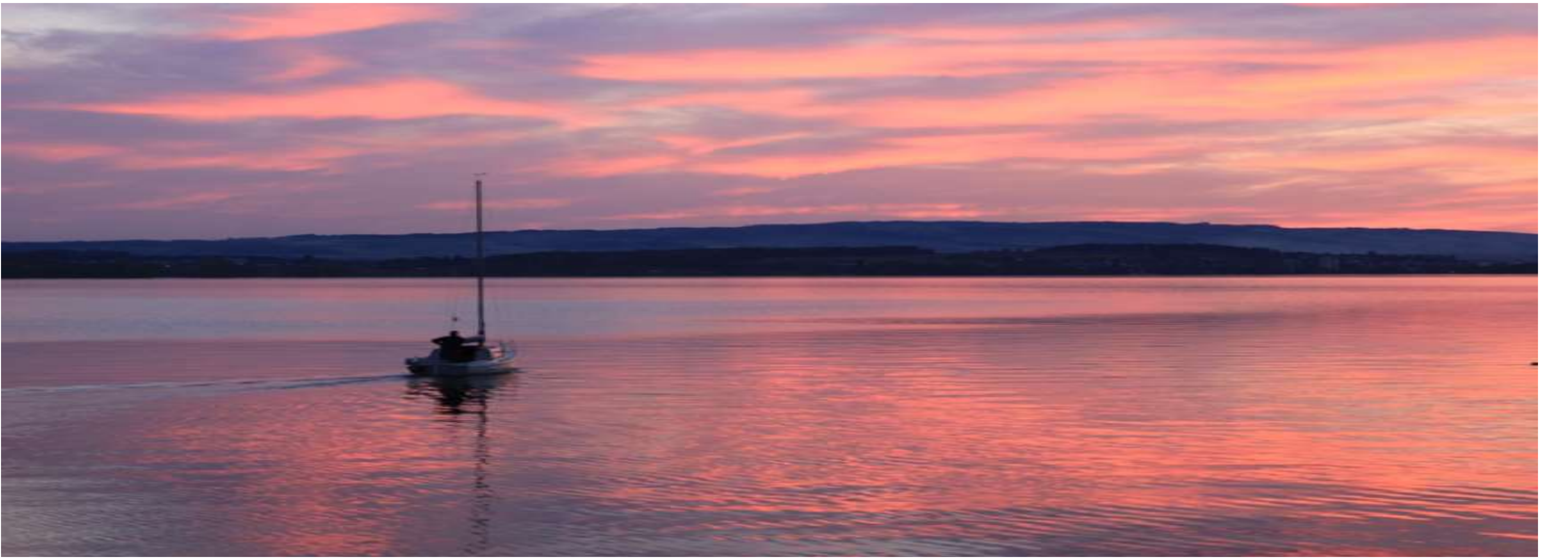
Guess

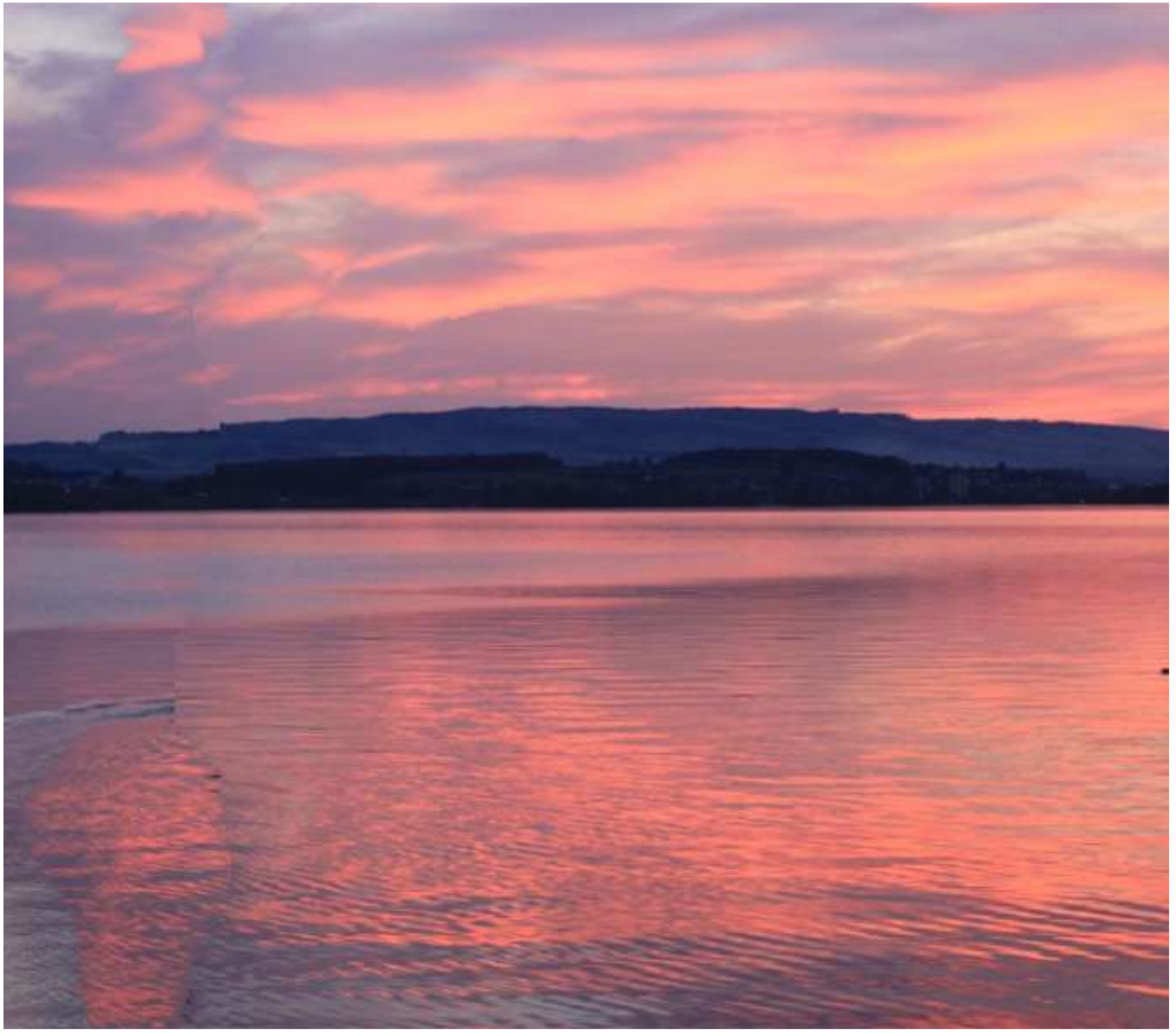
- We use "crop", "scaling" and "carving" for resizing the given image
- Guess which one is for carving?


600
400
200
0
0
200
400
600
800
1000

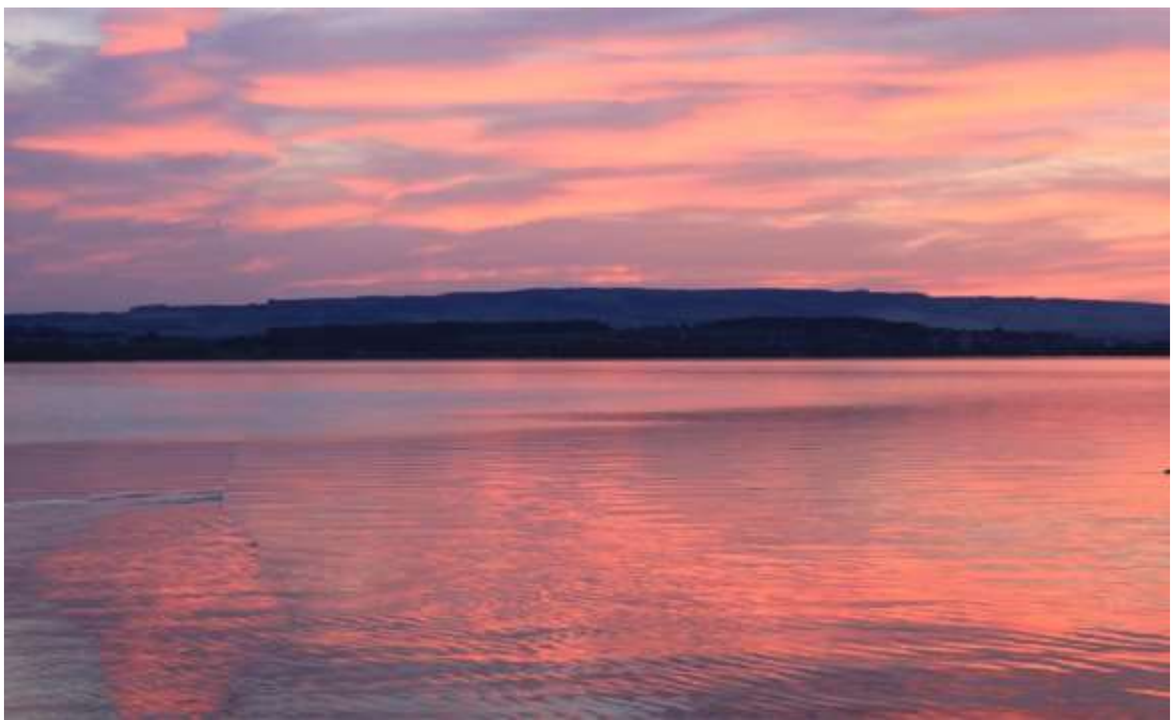








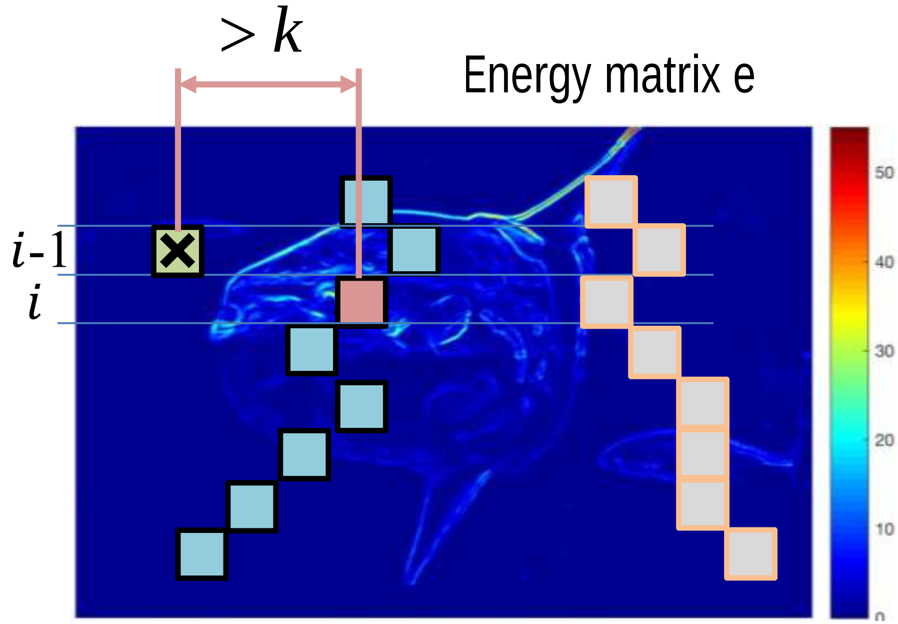
Expanded




Region to keep

M

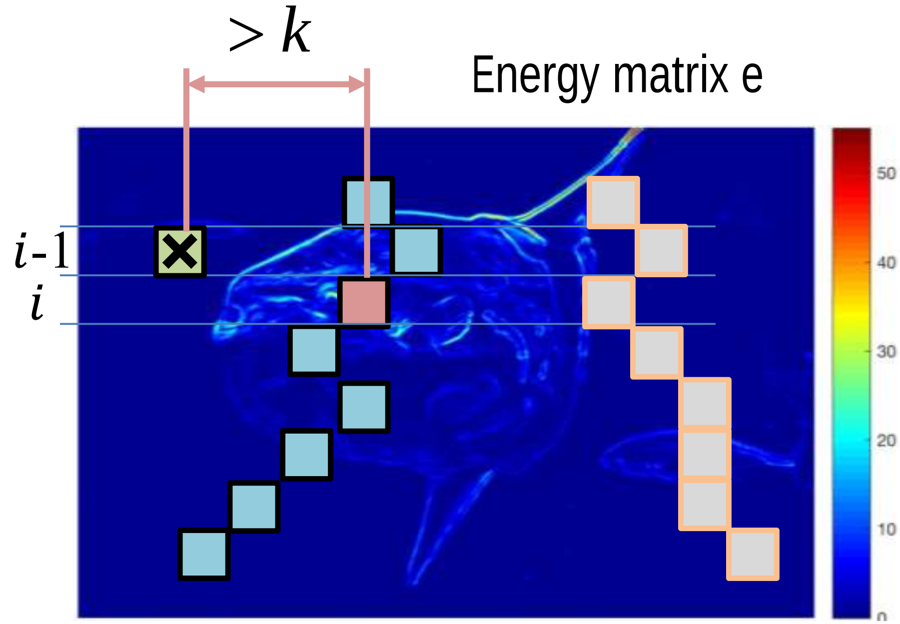
N



Seam: $S^y : \{(\mathrm{i}, \mathrm{y}(\mathrm{i})) \mid \mathrm{i} = 1,\ldots,\mathrm{M}\}$ s.t. $|\,\mathrm{y}(\mathrm{i}) - \mathrm{y}(\mathrm{i}-1)\,| \le \mathrm{k}$

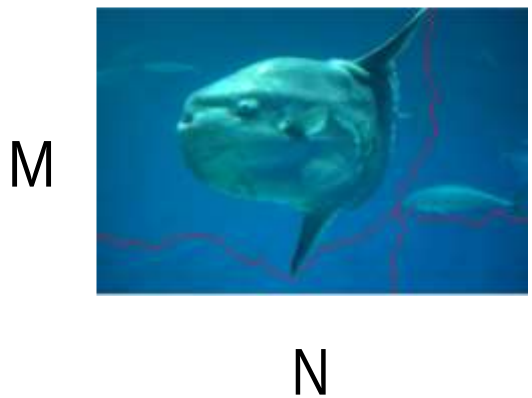
M

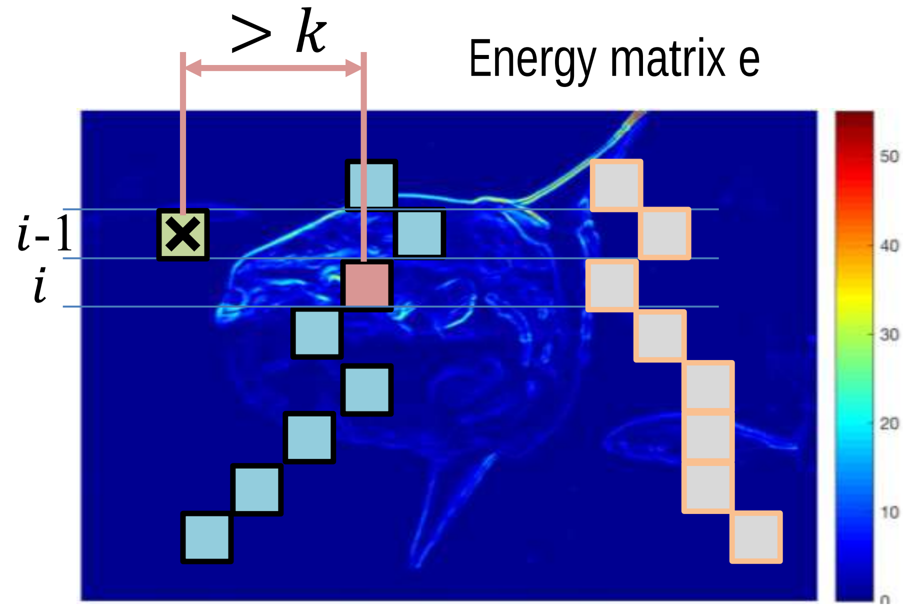
N



$$S^{y}:\{(\mathrm{i},\mathrm{y}(\mathrm{i}))\,|\,\mathrm{i}=1,\ldots,\mathrm{M}\}\quad \text{s.t.}\quad |\,\mathrm{y}(\mathrm{i})-\mathrm{y}(\mathrm{i}-1)\,|\leq \mathrm{k}$$

Seam Cost: $$\mathrm{E}(S^{y})=\sum_{i=1}^{\mathbf{M}} \mathrm{e}(S^{y}(i))$$

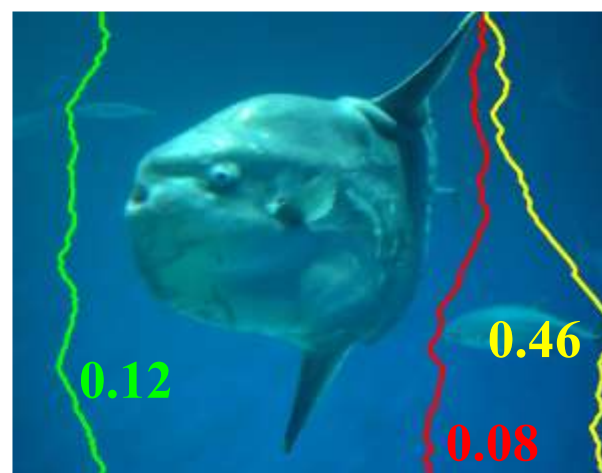




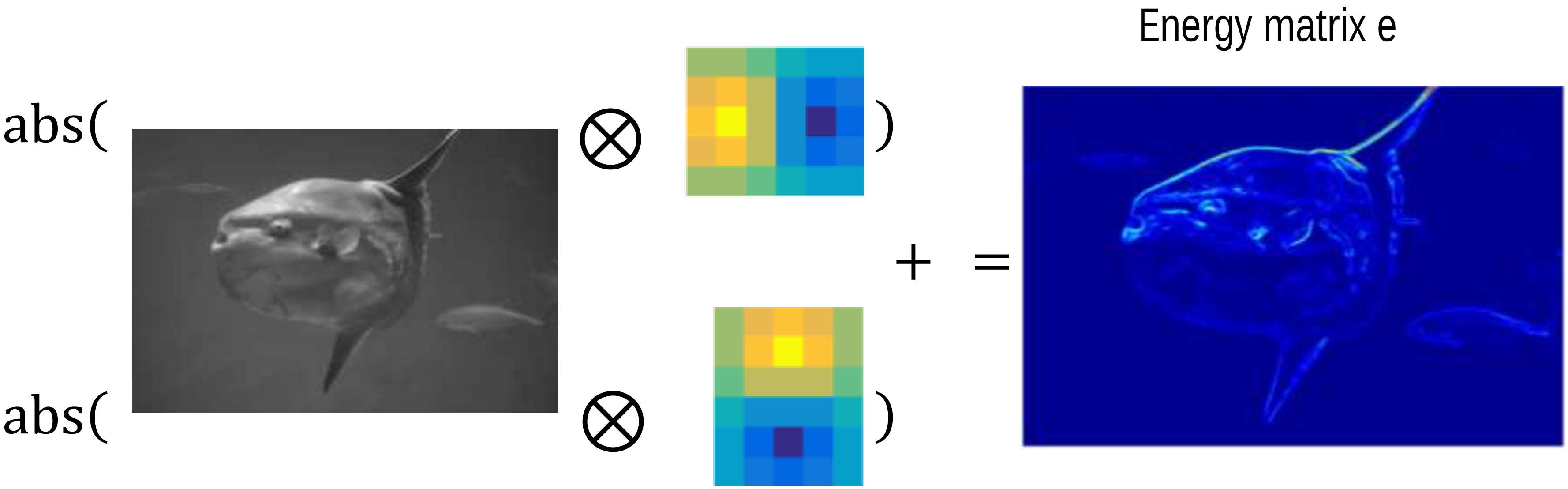
- Seam $S^y: \{(i, y(i)) | i = 1, \cdots, M\}$ $s.t.$ $|y(i) - y(i-1)| \leq k$
- **Seam Cost:**

$$\mathrm{E}(S^y) = \sum_{i=1}^{\mathbf{M}} \mathrm{e}(S^y(i))$$

- **Goal:** $S^* = \min \mathrm{E}(S^y)$



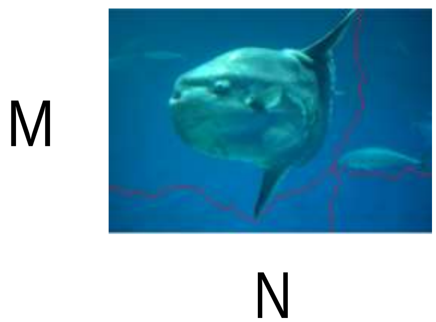
Where does the energy matrix come from?

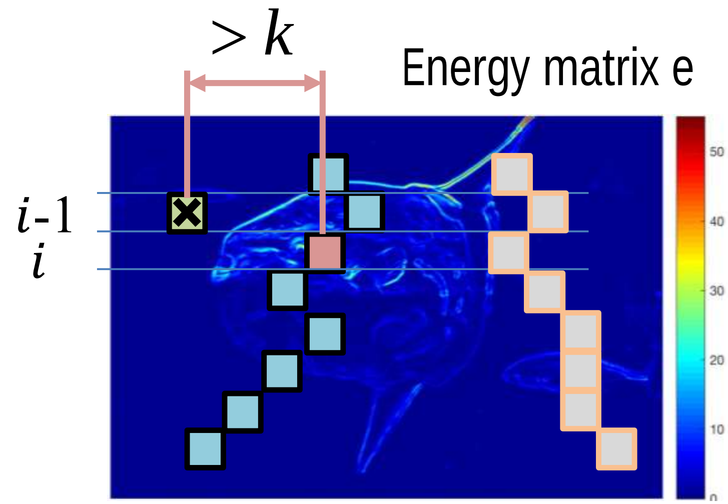


Energy matrix

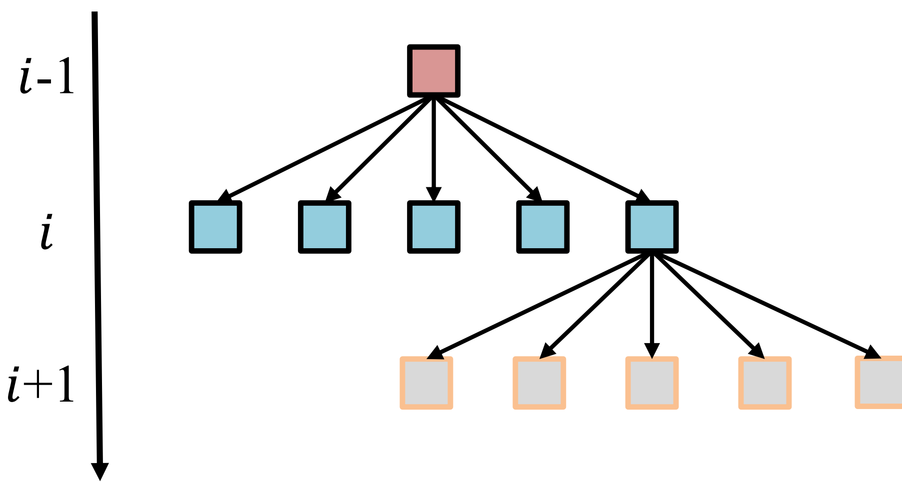
For example: L1 norm of the edge gradients for the energy function

How to find the best seam?





Idea 1: Brute force search

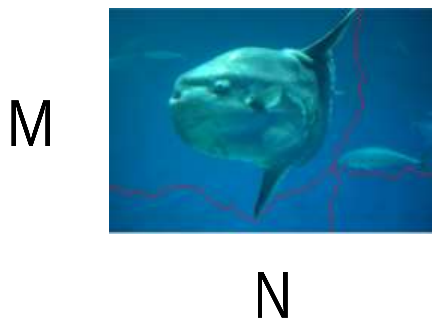


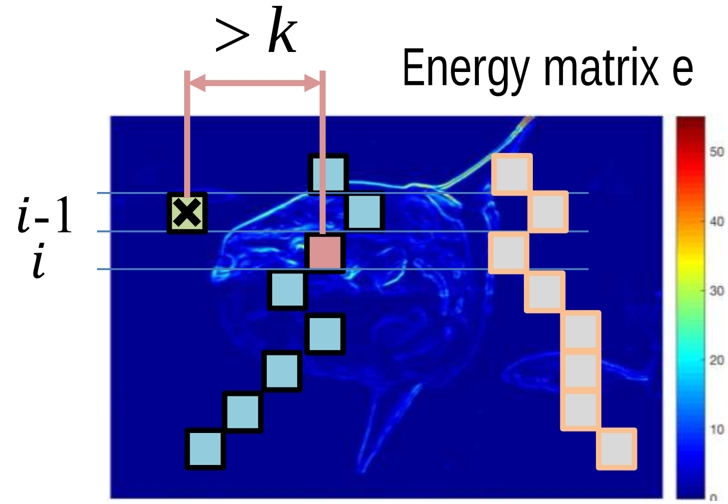
- 1st row: N
- 2nd row: $(2k+1)N$
- 3rd row: $(2k+1)^2 N$

....

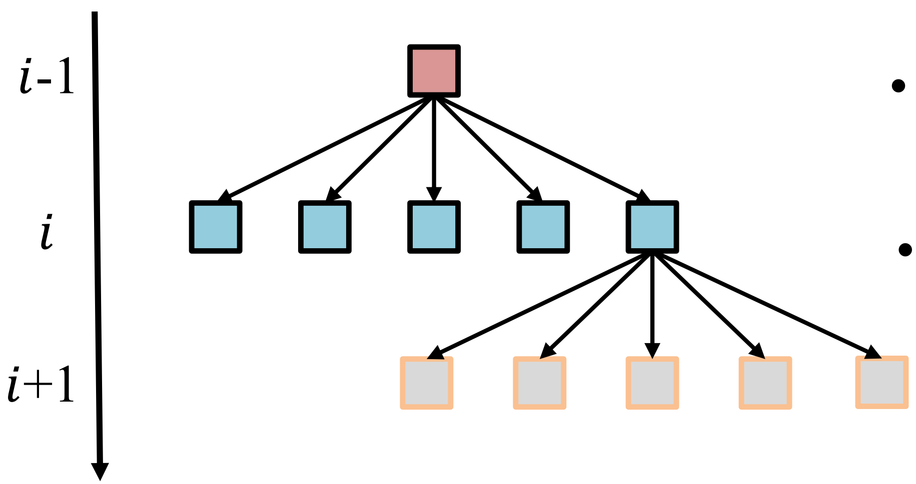
- Mth row: $(2k+1)^{(M-1)} N$

- How many possibilities for Seam S^y?





Idea 1: Brute force search



- $\mathbf{M}^{\text{th}}$ row: $(2k+1)^{(\mathbf{M}-1)}\mathrm{N}$
- Given $\mathbf{M} = 768, \mathbf{N} = 1024, k = 1$
- How many possibilities for Seam S^y?

1024×3^{767}

Too many possibilities!

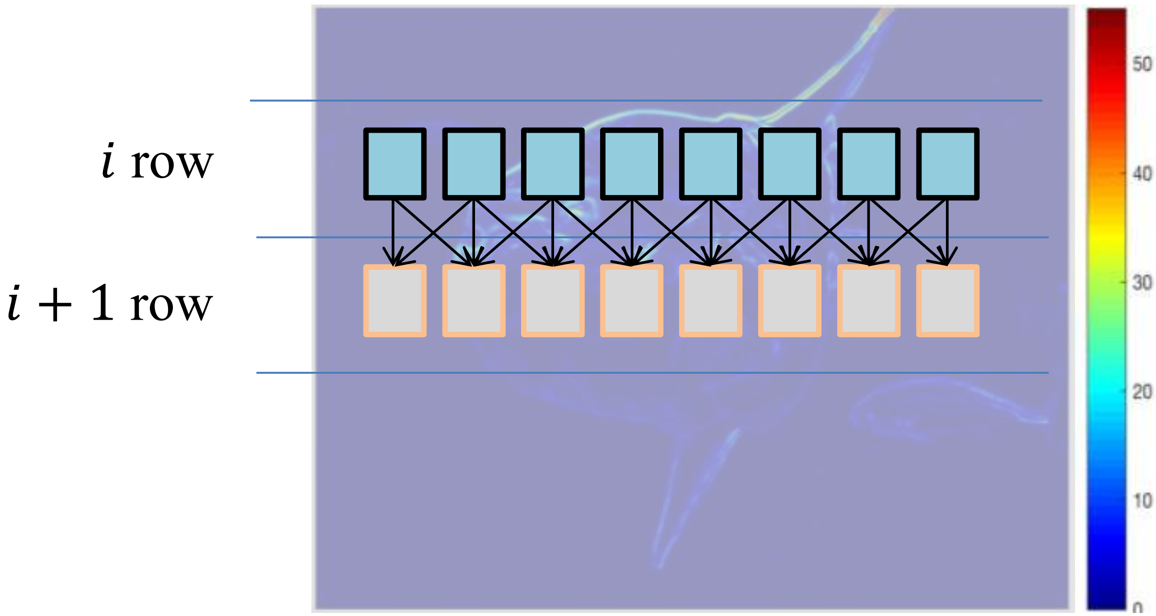
$$1024 \times 3^{767}$$

M

N

Idea 2: Find the shortest path from the first row to the last row on the energy map

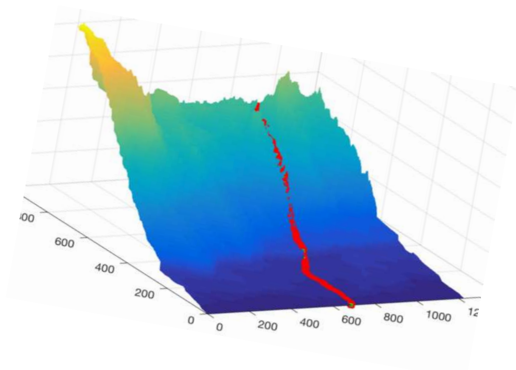
Shortest Path



Construct the directed graph **G** where

each pixel (node) is connected to the (2k+1) neighbors in the next row

Shortest Path



V(u)

- Create an `interior' set S, initialize with the first row
- Growing S with `least-resistant' step
- Construct a 'Value' matrix with value $V(u)$ encoding the shortest path cost to each node in S

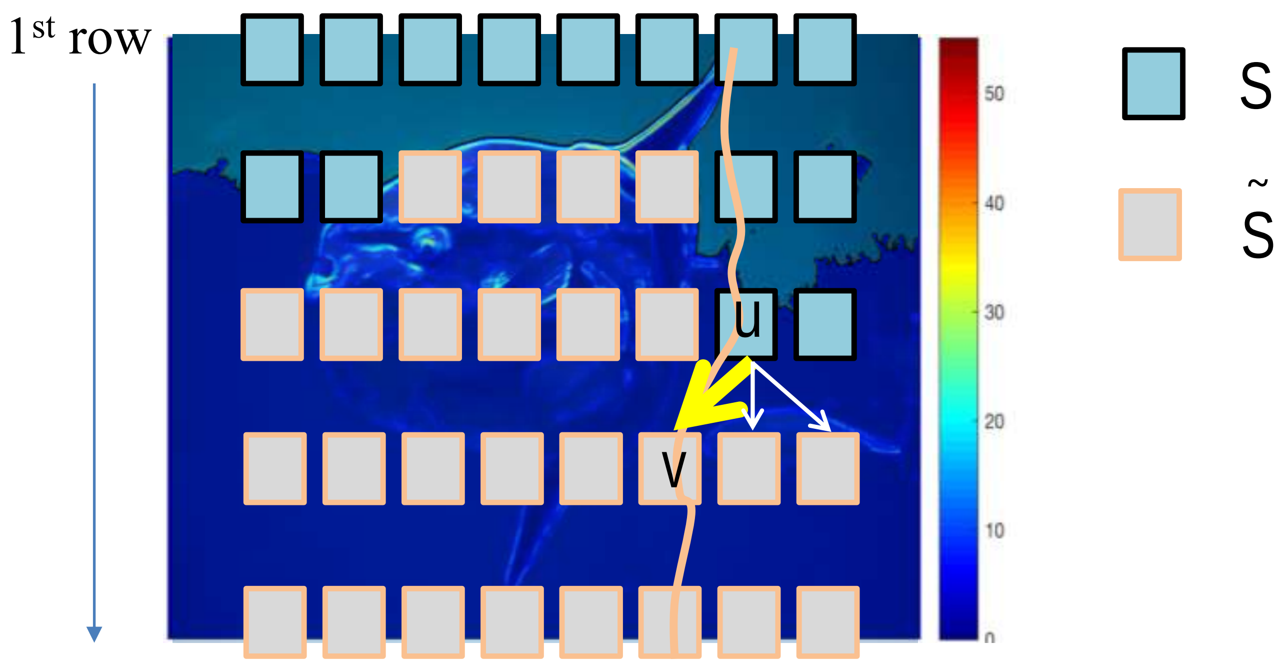
Shortest Path

- Initialize S to include the first row, and set the Value function

$$V(u) = e(u), u \in S$$

For the first row, the shortest path contains only itself

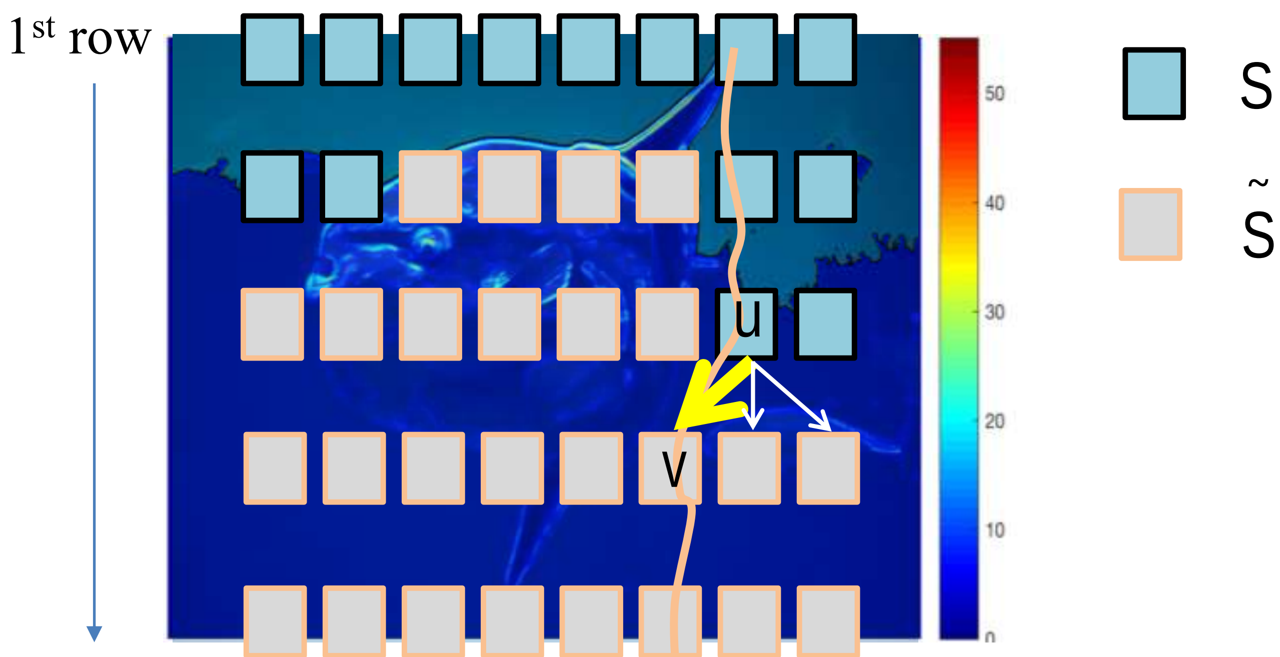
Shortest Path



Iterate: find the step expansion of `least resistant' from $S \rightarrow \tilde{S}$

$$v = \underset{u \in S, v \in \tilde{S}}{\operatorname{argmin}} [V(u) + e(v)] \quad \textit{where } (u \rightarrow v) \text{ is a graph edge}$$

Shortest Path

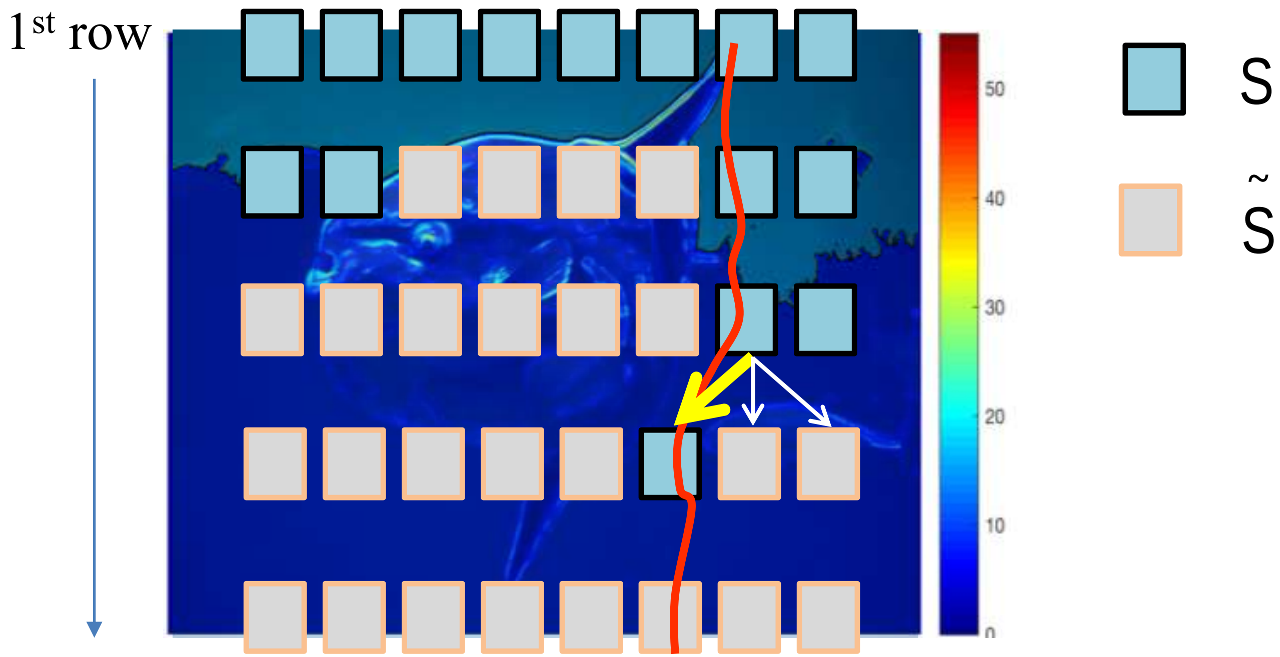


Iterate: find the step expansion of least resistant from $S \rightarrow \tilde{S}$

$$v = \underset{u \in S, v \in \tilde{S}}{\operatorname{argmin}} [V(u) + e(v)] \qquad \textit{where } (u \rightarrow v) \text{ is a graph edge}$$

Remember the path back from $(v \rightarrow u)$ $\qquad P(v) = u$

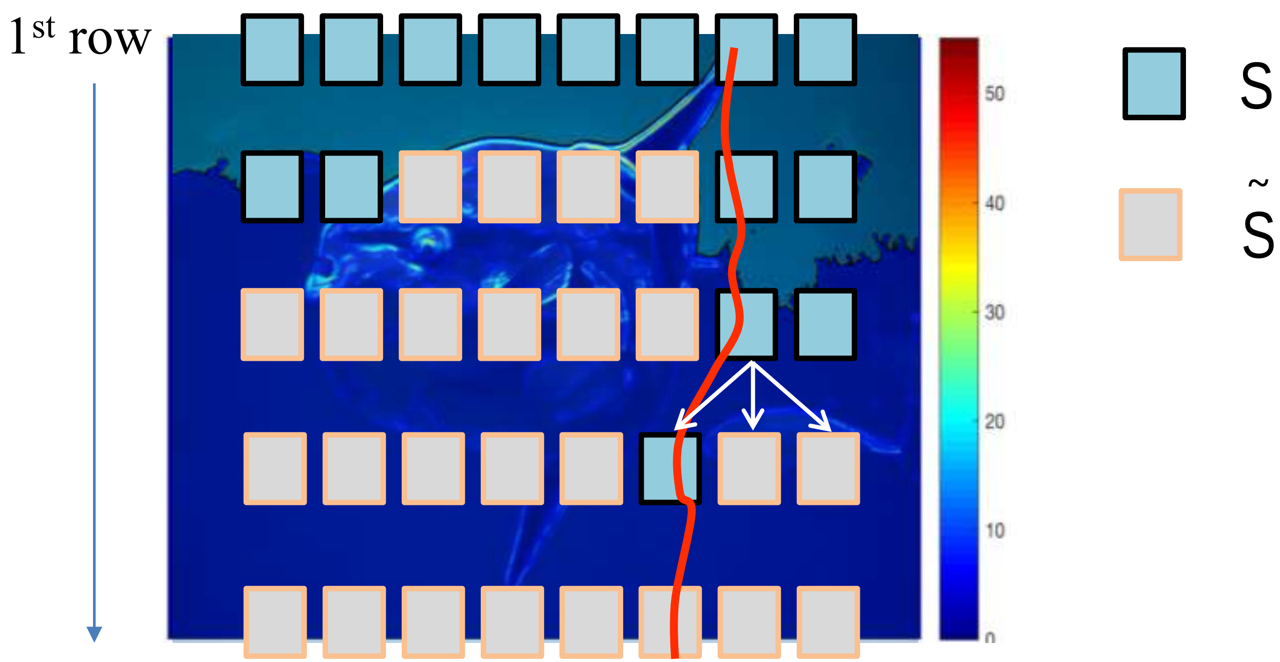
Shortest Path



- Updating the "interior" set: $S = S \cup \{v\}, \quad \tilde{S} = \tilde{S} - \{v\}.$
- Updating the Value function: $V(v) = \min_{v \in \tilde{S}} [V(u) + e(v)]$

$V(v)$: Is the **contingent cost** E of the shortest path connecting the pixel v to the first row

Shortest Path



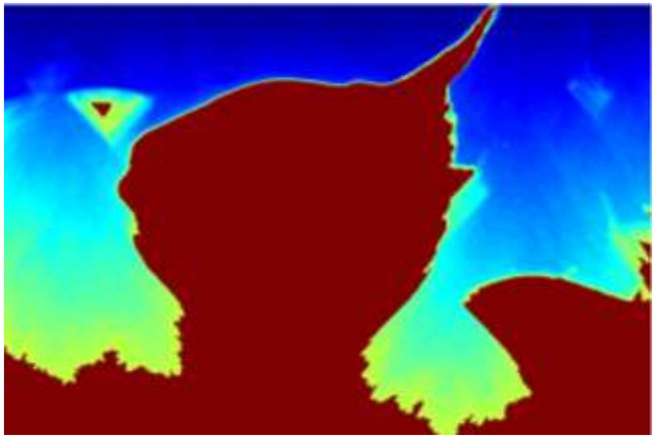
- Updating the "interior" set: $S = S \cup \{v\}, \quad \tilde{S} = \tilde{S} - \{v\}.$

- Updating the Value function: $V(v) = \min_{v \in \tilde{S}} [V(u) + e(v)]$

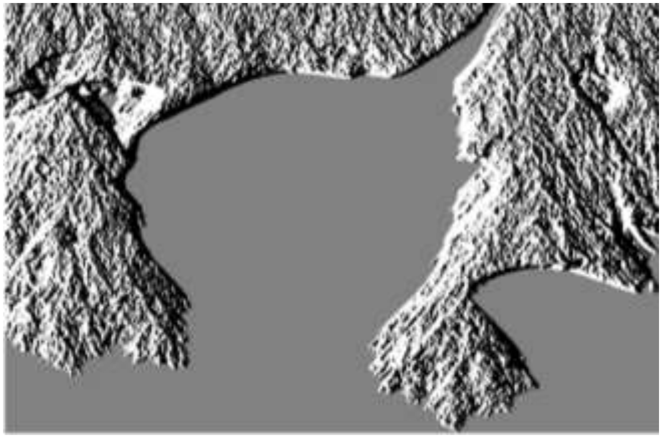
- Until reaching one of the pixels on the last row.

Energy Matrix e

Value function V(u)



Path Matrix P



Frontier growing row by row

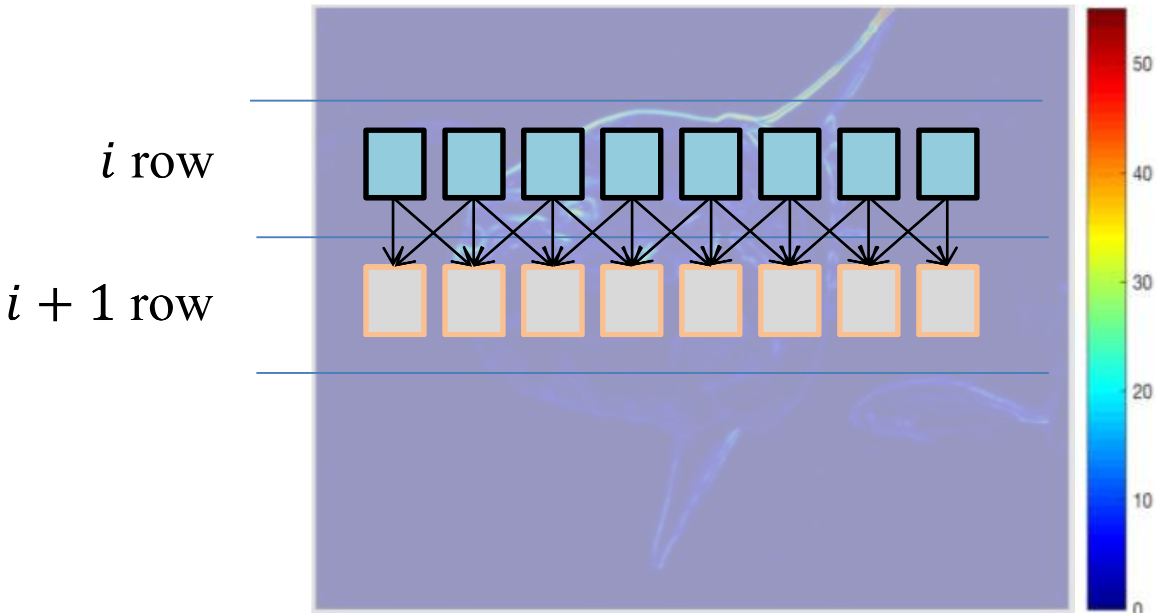
Dynamic Programming

M

N

Idea 3: Dynamic programming!

Dynamic Programming



Use the same graph structure

Propagate the frontier row by row to construct the Value Matrix

Dynamic Programming

Value Matrix V

Path Matrix P

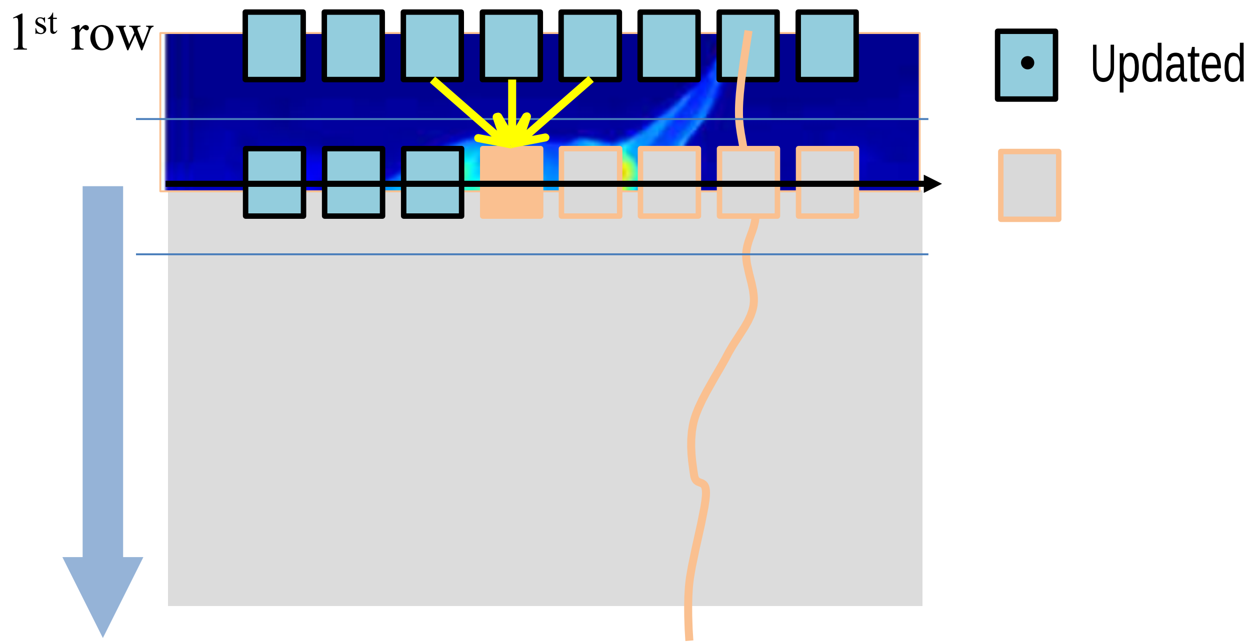
- Still start from first row, and initialize the Value function

$$V(1, j) = e(1, j)$$

- Set the Path function

$$P(1, j) = 0$$

Dynamic Programming

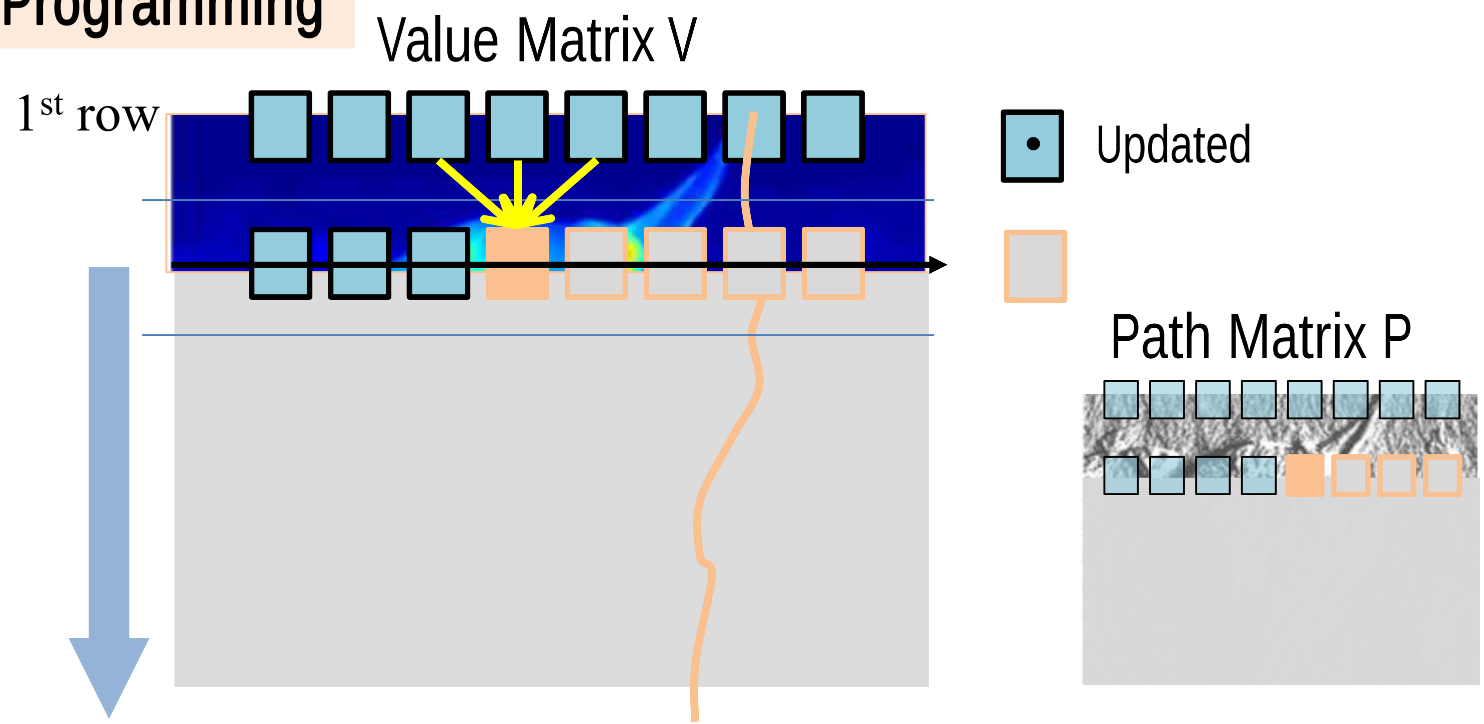


- Propagate to the second row, and update the value matrix

$$V(2,j) = e(2,j) + \min(V(1,j\text{-}k),\ldots,V(1,j),\ldots,V(1,j+k))$$

$V(2,j)$: is the **contingent cost** of the shortest path connecting the pixel **(2,j)** to the first row

Dynamic Programming



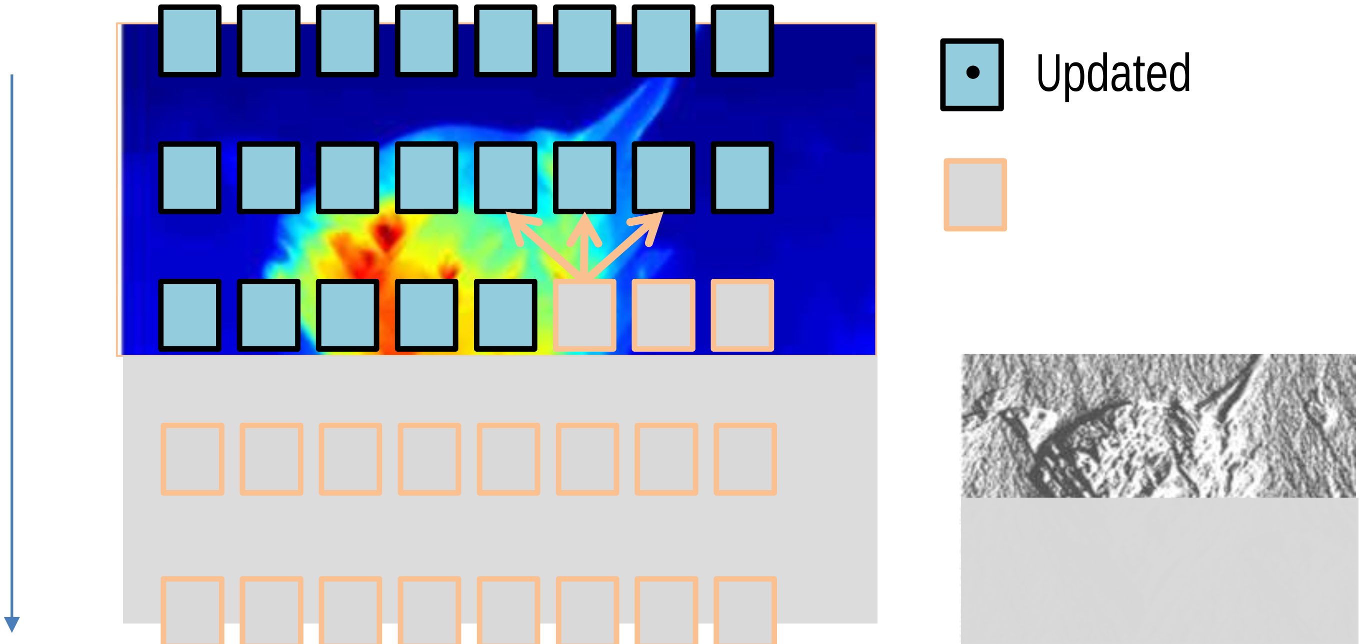
- Propagate to the second row, and update the value matrix

$$V(2,j) = e(2,j) + \min(V(1,j\text{-}k),\ldots,V(1,j),\ldots,V(1,j+k))$$

- Set the Path function

$$P(2,j) = \operatorname{argmin}(V(1,j\text{-}k),\ldots,V(1,j),\ldots,V(1,j+k))$$

Dynamic Programming



- Iteratively propagate to the **ith** row, and update the value matrix

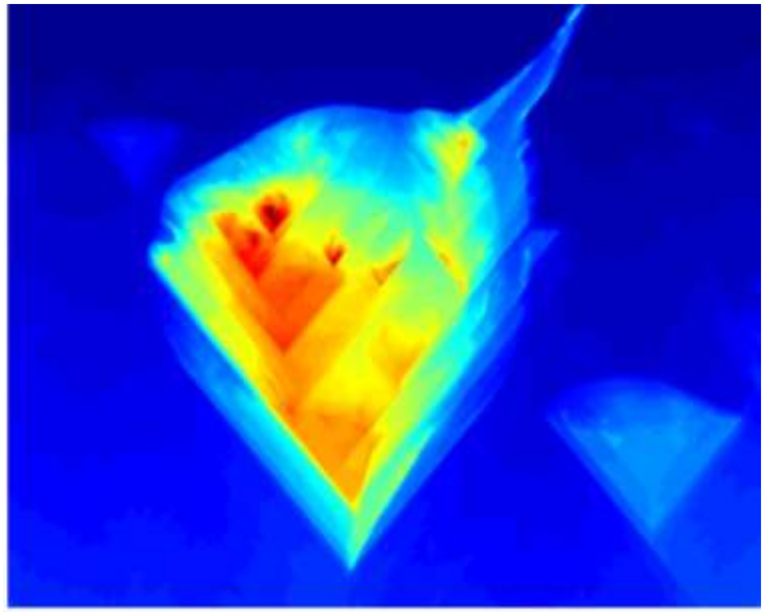
$$V(i,j) = e(i,j) + \min(V(i\text{-}1,j\text{-}k),\ldots,V(i\text{-}1,j),\ldots,V(i\text{-}1,j+k))$$

- Set the Path function

$$P(i,j) = \operatorname{argmin}(V(i\text{-}1,j\text{-}k),\ldots,V(i\text{-}1,j),\ldots,V(i\text{-}1,j+k))$$

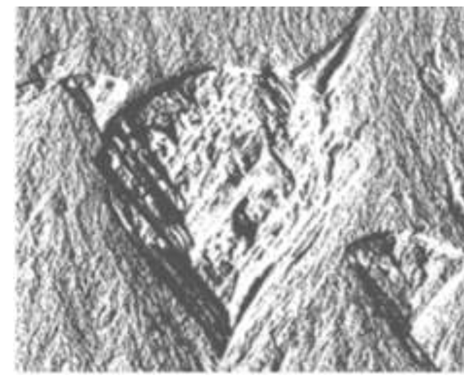
Dynamic Programming

Value Matrix V



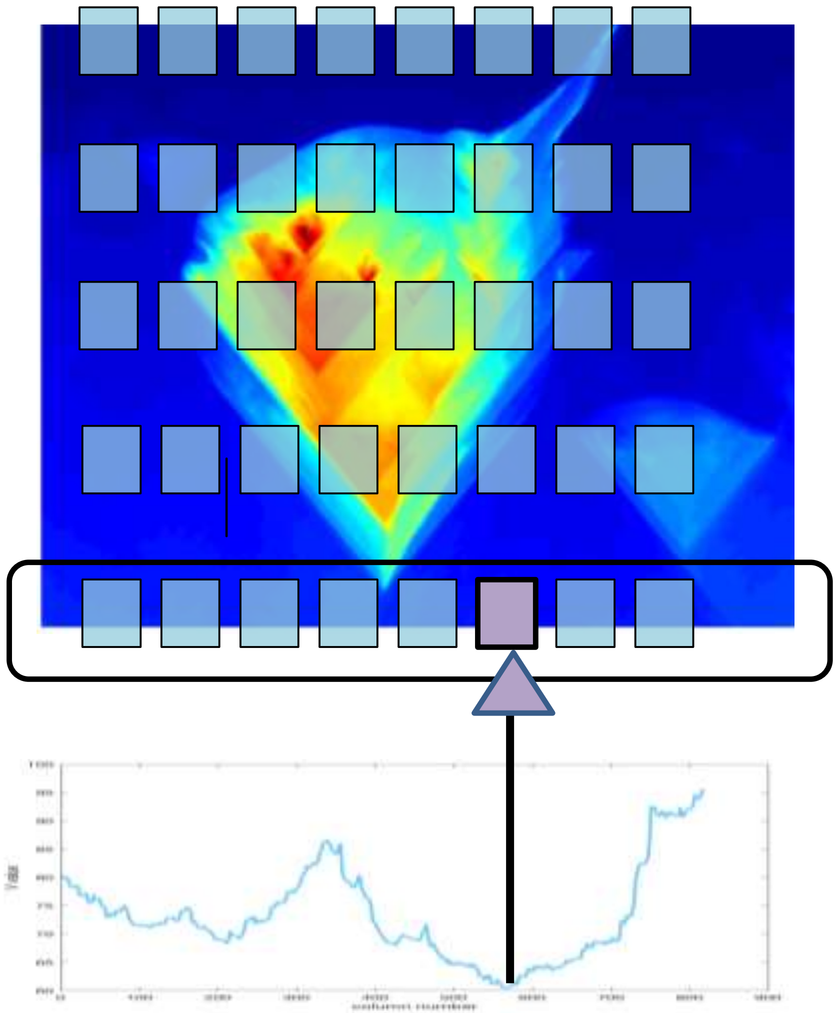
Energy Matrix e

Path Matrix P

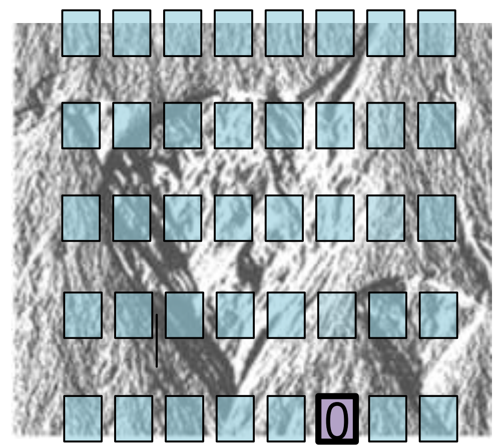


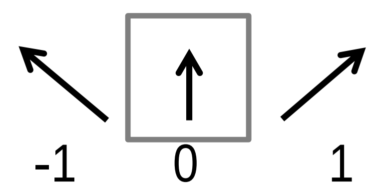
Dynamic Programming

Value Matrix V



Path Matrix P

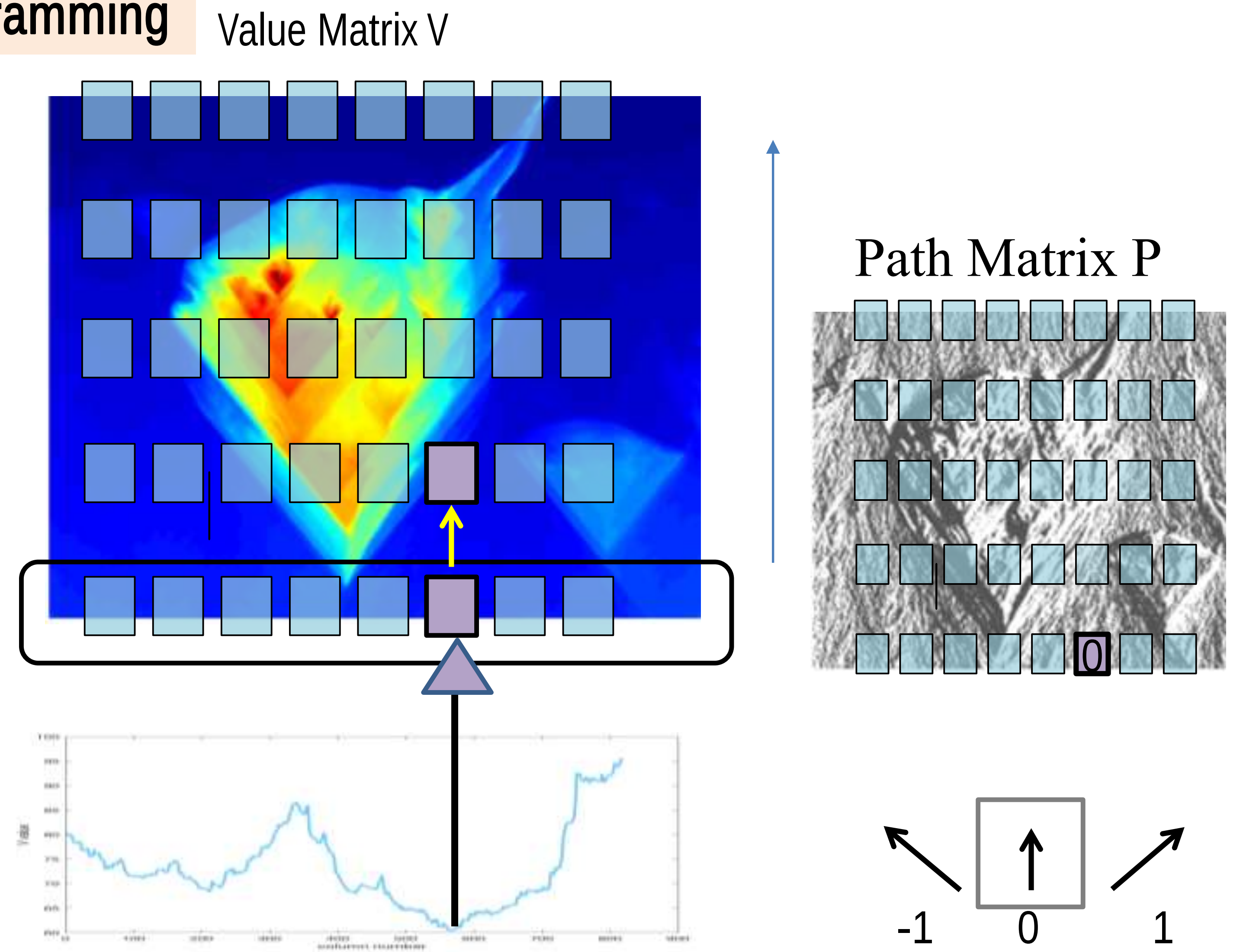




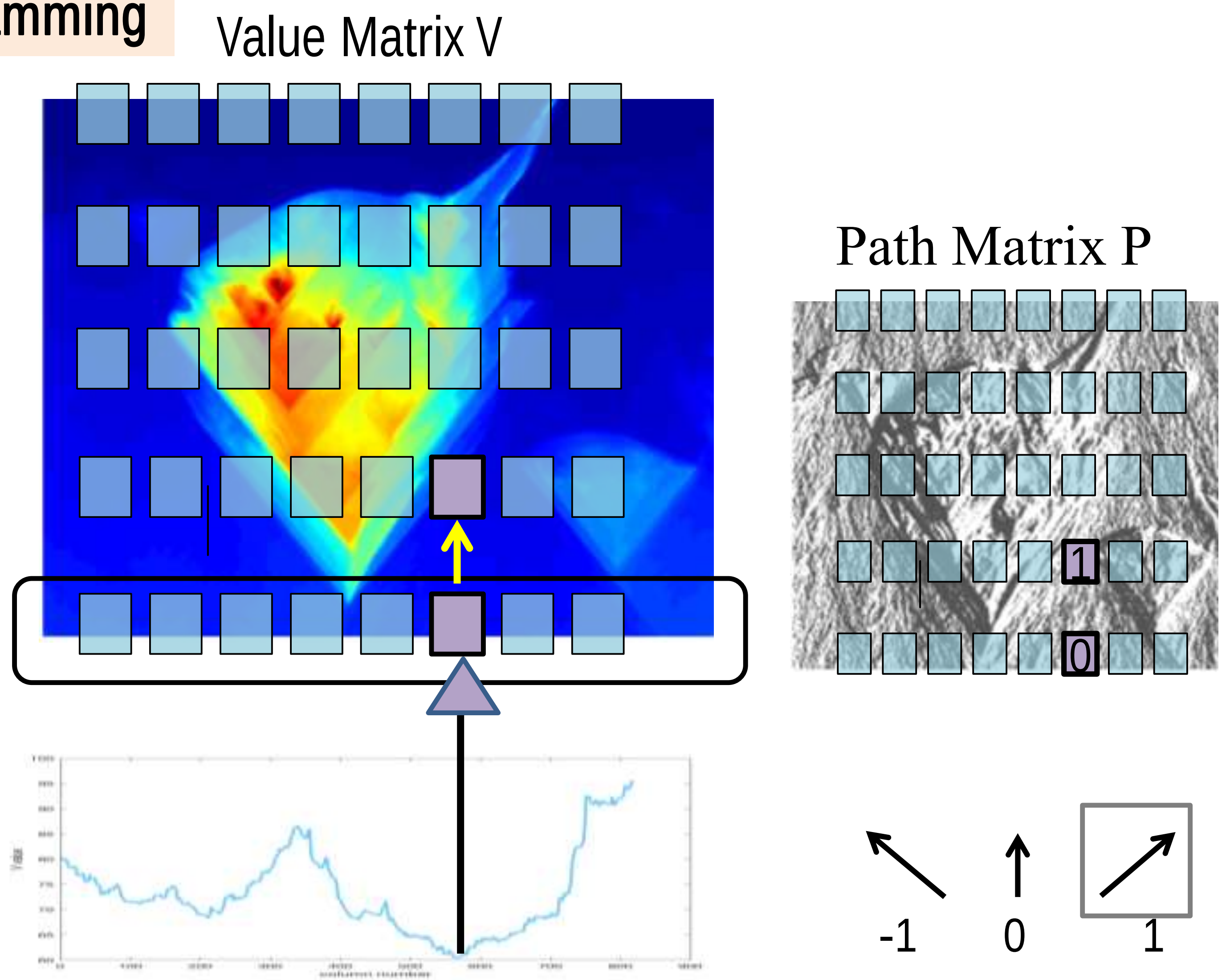
- Localize the minimum of the last row of

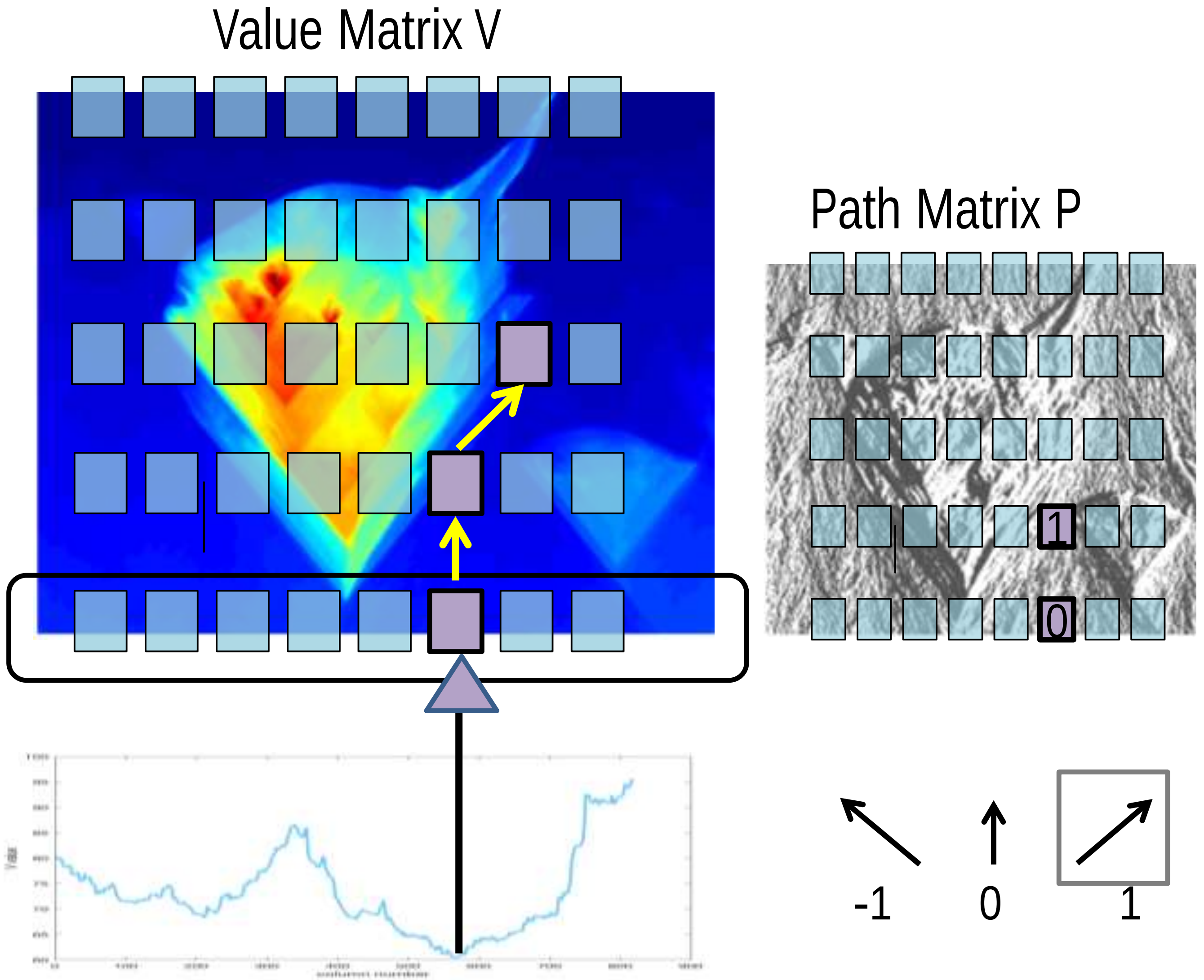
$$\operatorname{argmin} V(M,:)$$

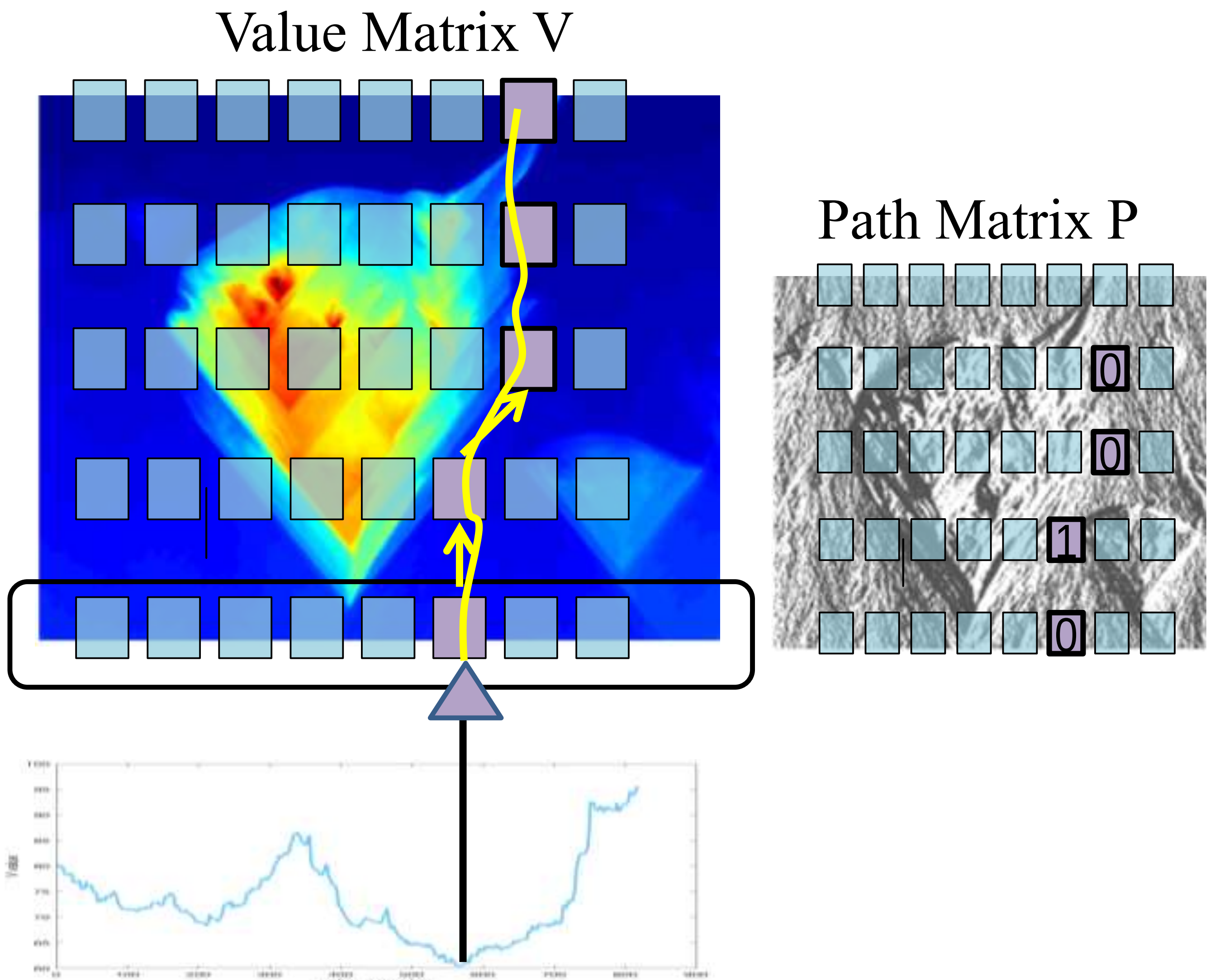
Dynamic Programming

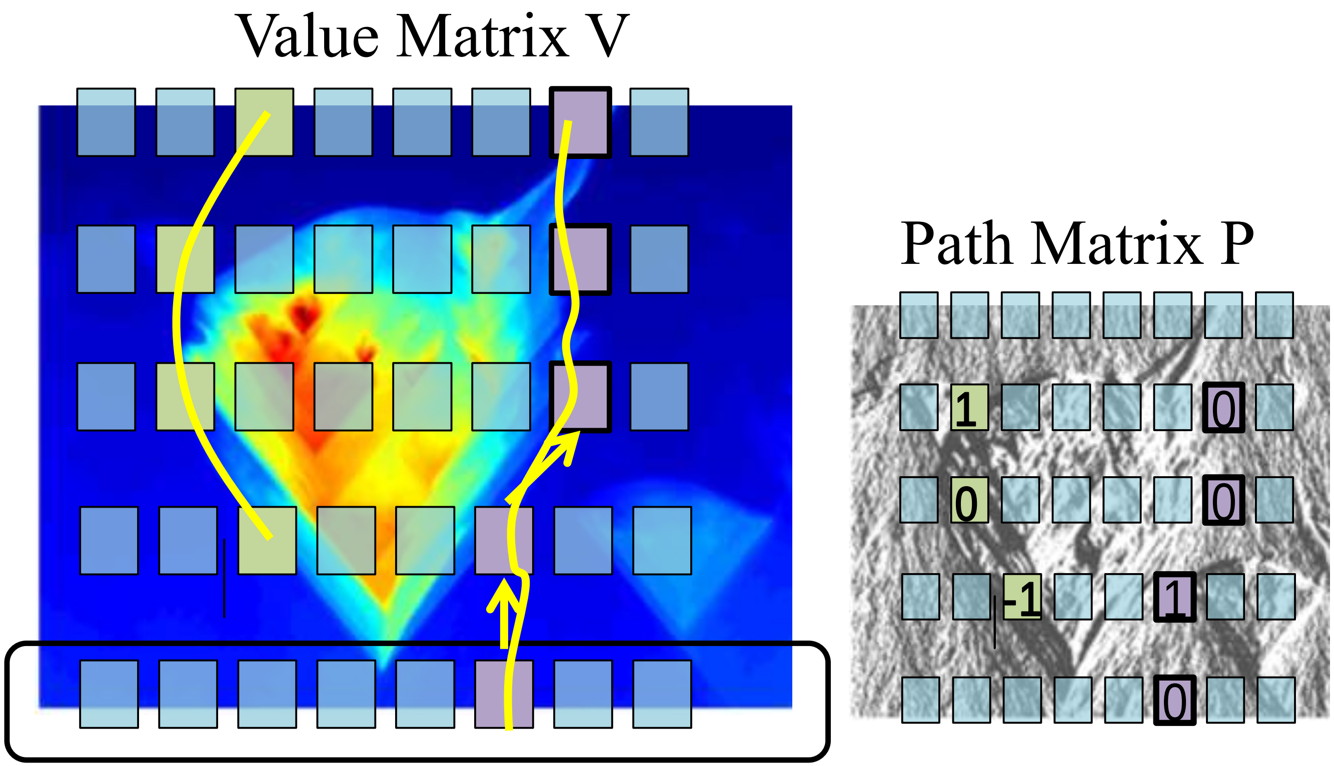


Dynamic Programming



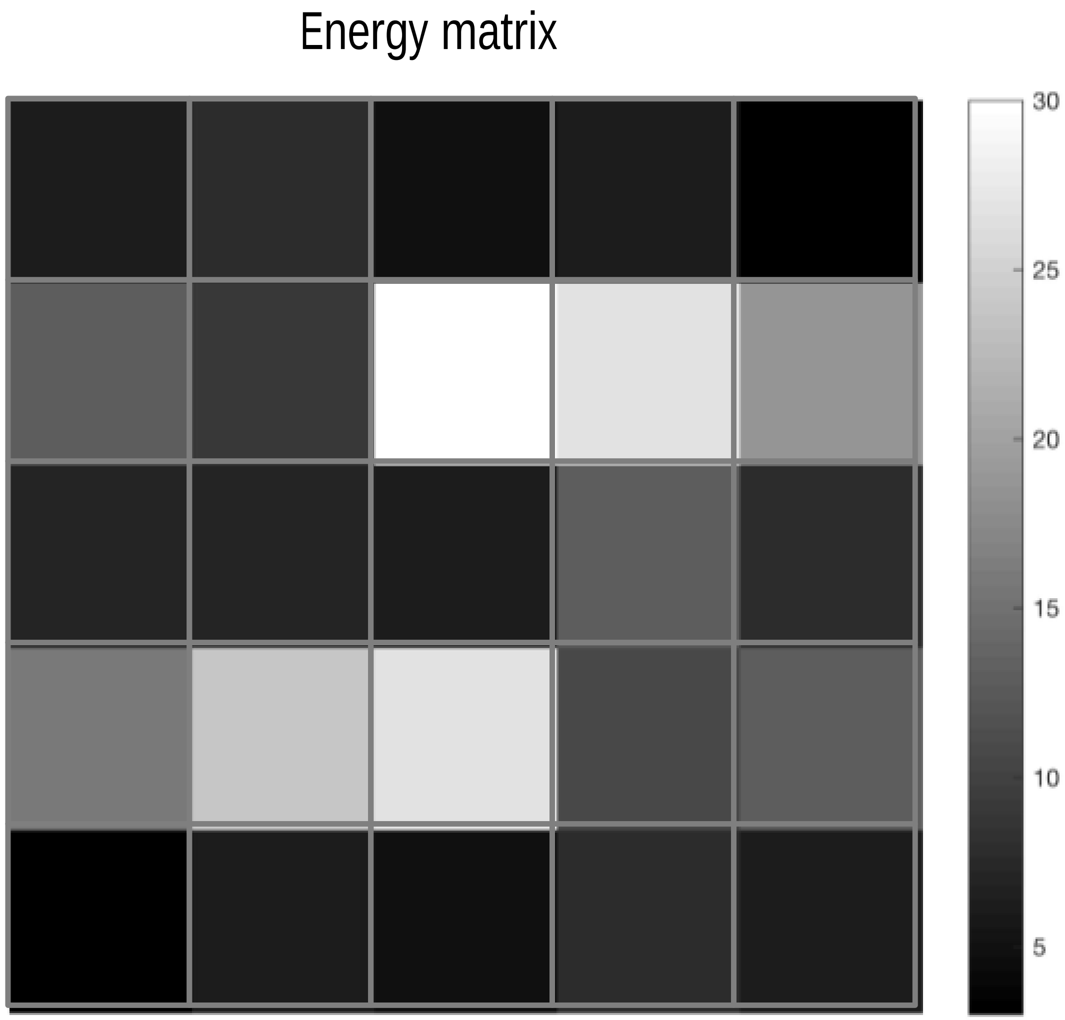

Value Matrix V
Path Matrix P
1
0
-1
0
1


Value Matrix V
Path Matrix P
0
0
1
0



- Since V is the contingency table, we can start from any pixel and find a shortest path to the first row!

Illustration for synthetic case


Energy matrix
30
25
20
15
10
5

Value matrix

5.5	8	4.5	6	3
18.5	13.5	34.5	30	22
20	20.5	19.5	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

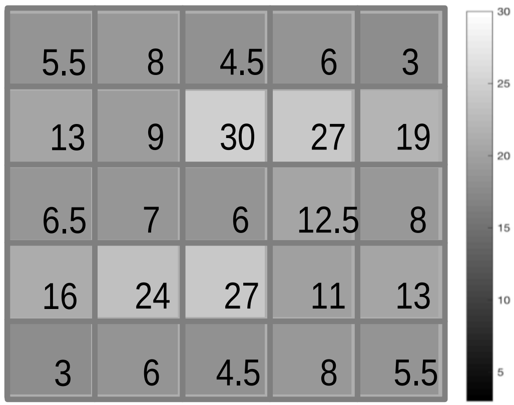
Energy matrix

5.5	8	4.5	6	3
13	9	30	27	19
6.5	7	6	12.5	8
16	24	27	11	13
3	6	4.5	8	5.5

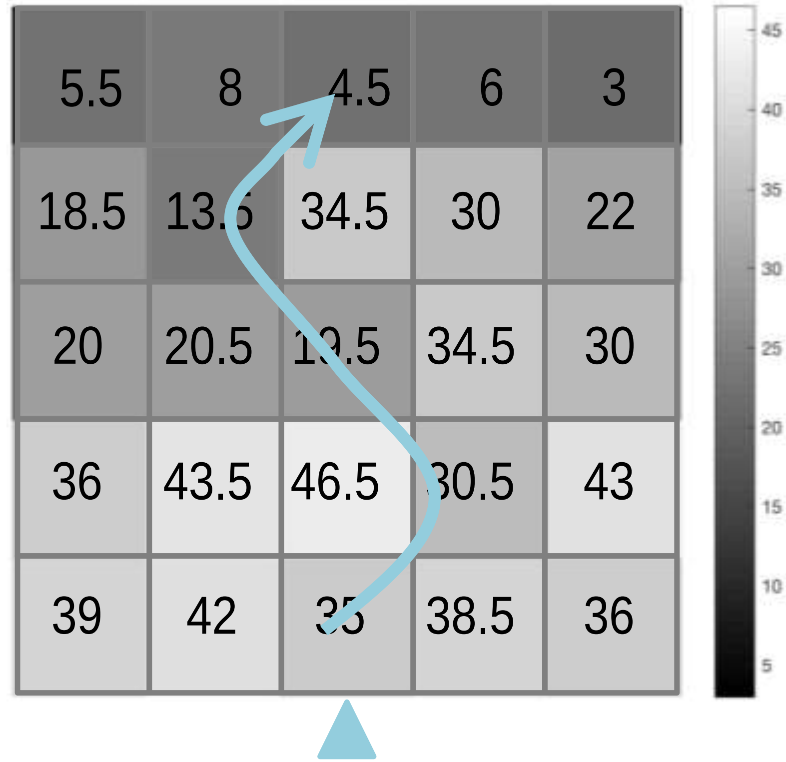
Goal:

- Construct *value and path matrix* from the ***energy matrix***
- ***Value matrix*** records the *energy of the shortest path* from the starting row to the current pixel

Energy matrix

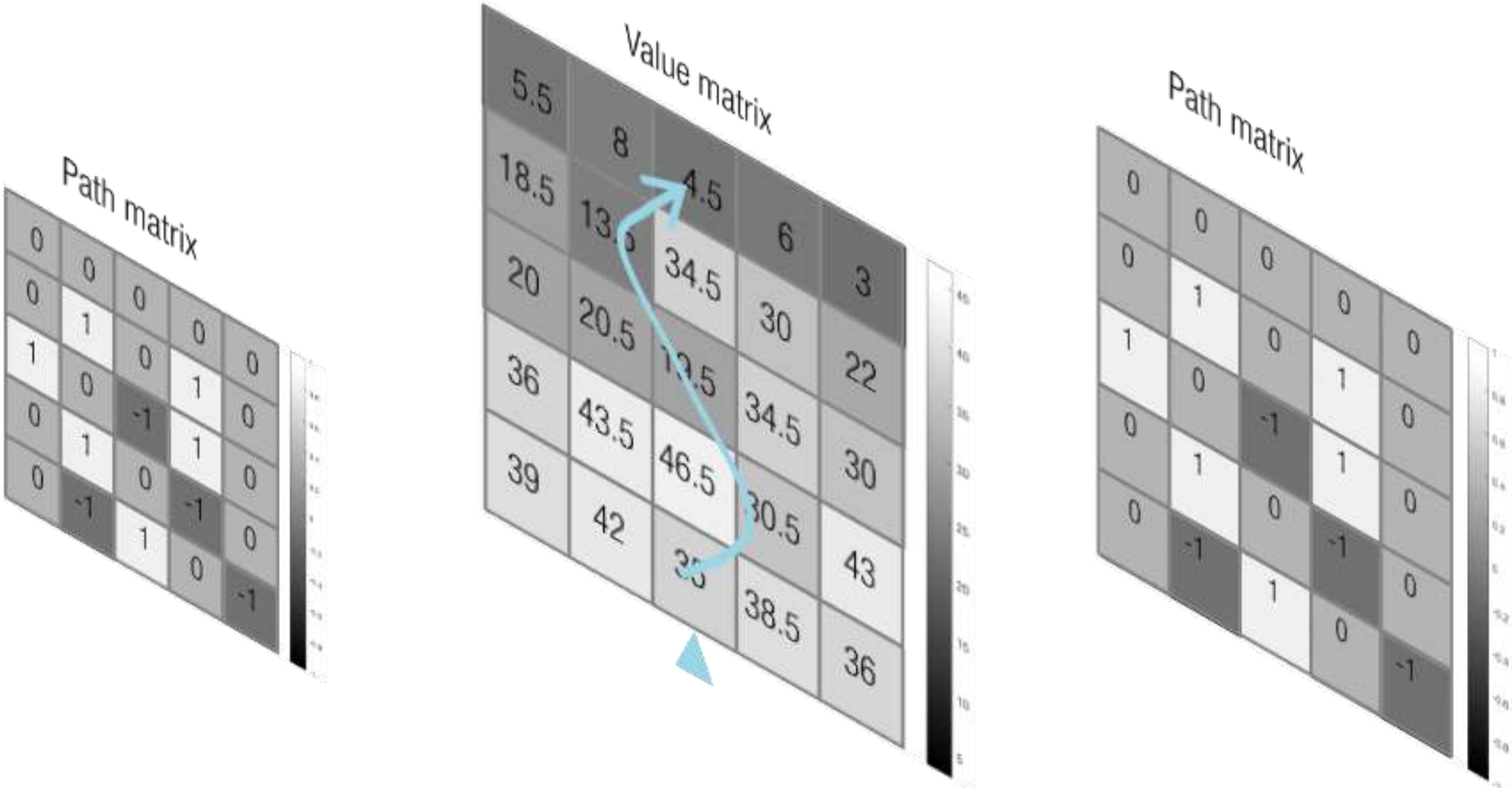


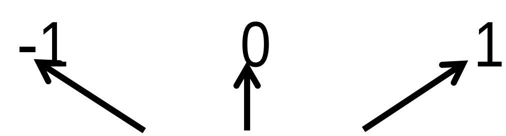
Value matrix



Goal 1:

- Construct *Value matrix* from the ***energy matrix***
- ***Value matrix*** records the ***energy of the shortest path*** from the starting row to the current pixel
- Property: every entry encodes the minimal shortest path to that node from starting row

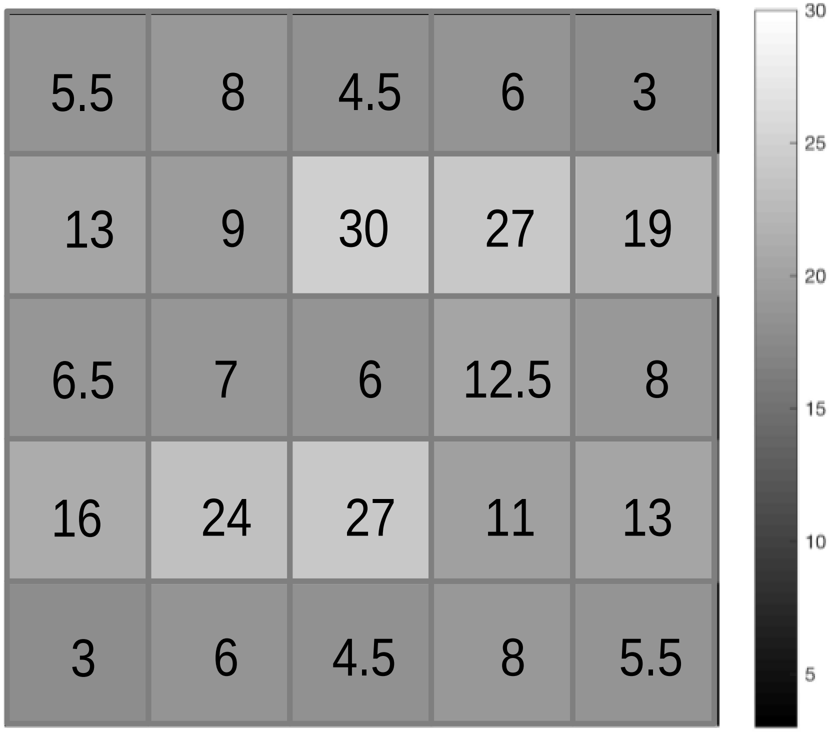




Goal 2:

- Construct ***Path matrix***
- The Path matrix records the immediate predecessor in the shortest path, for every node

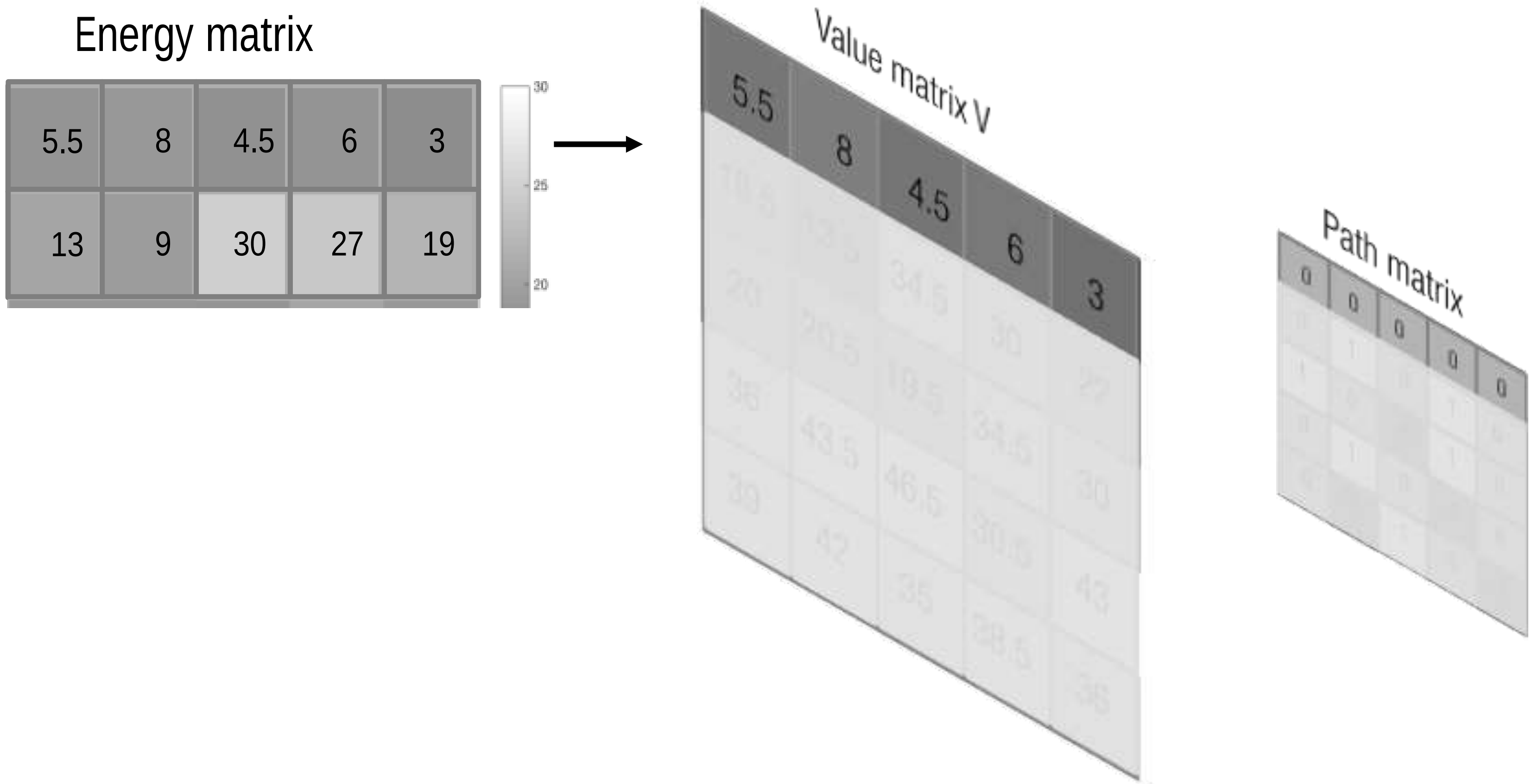
e: energy function



Energy function

- Energy function records the cost of a pixel.
- Typically it is the image gradient magnitude

How to generate the two matrices?



Step1: Initializing two matrices

- Set value and path matrix ***the same size*** as the energy matrix
- Initialize its ***first row of Value matrix*** with that of the energy matrix
- Initialize its ***first row of path matrix*** to zero

Value matrix V

5.5	8	4.5	6	3
18.5	13.5	34.5	30	22
20	20.5	19.5	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

Step2: Propagation

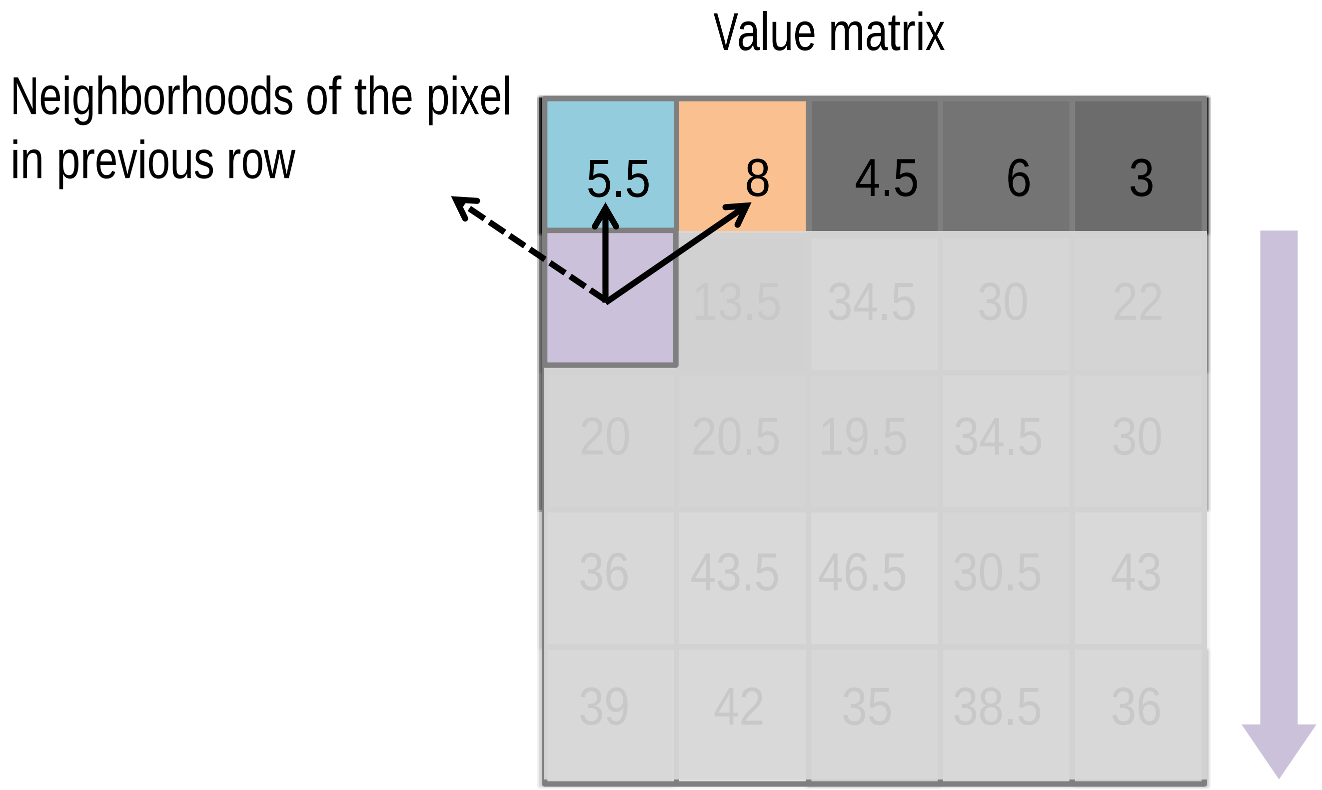
- Start with 2nd row, and propagate row by row

Value matrix V

5.5	8	4.5	6	3
?	13.5	34.5	30	22
20	20.5	19.5	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

Step2: Propagation

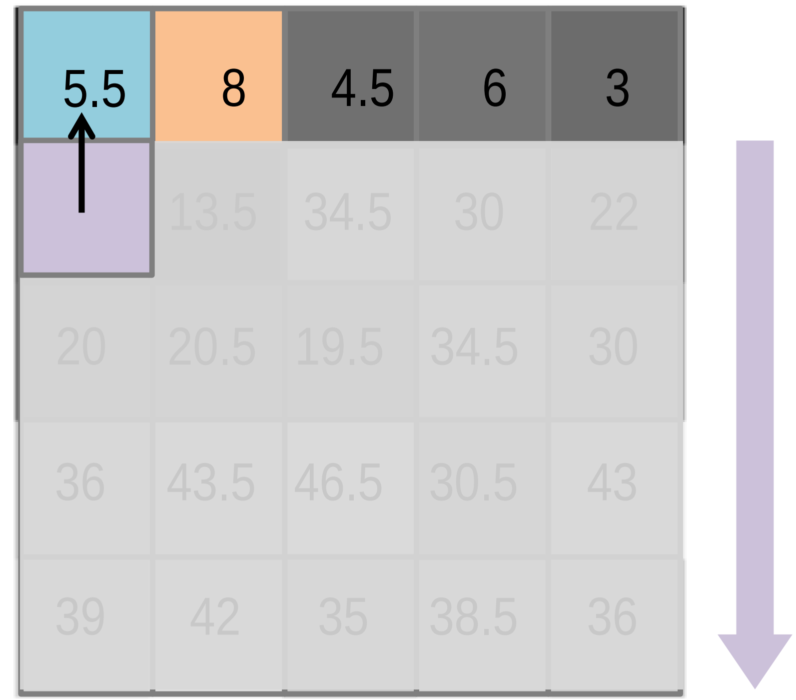
- Start with 2nd row



Step2: Propagation

- Start with 2nd row
- Find the ***neighbors*** of the pixel in the previous row

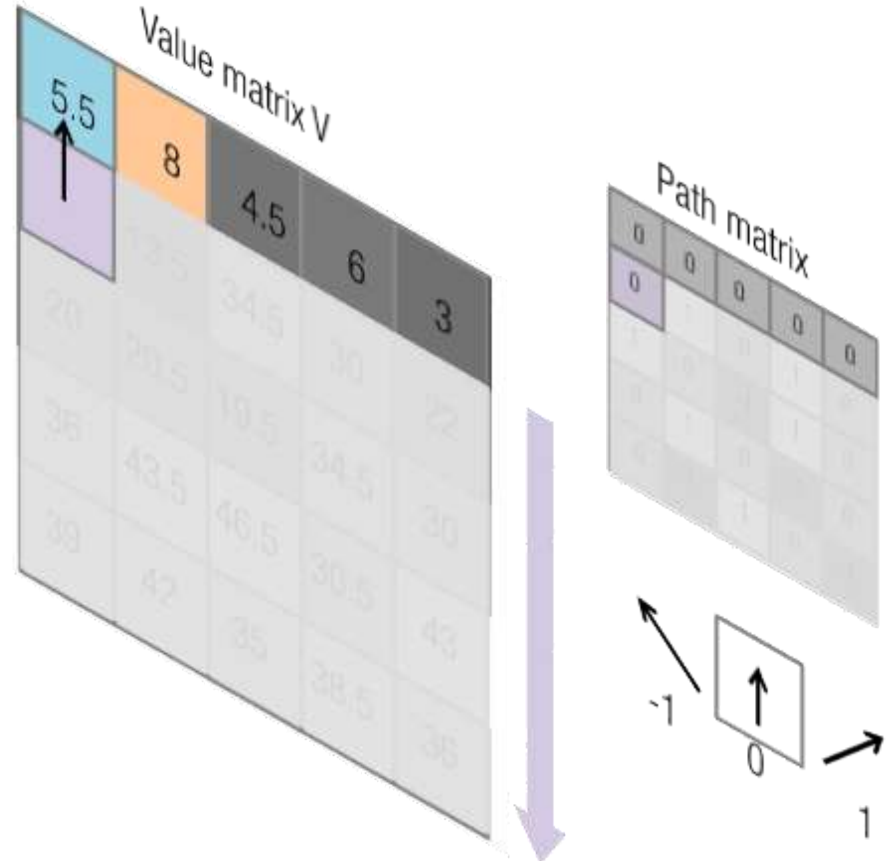
Value matrix V



$$V(2,1) = \min \begin{pmatrix} V(1,1) \\ V(1,2) \end{pmatrix} \begin{matrix} 5.5 \\ 8 \end{matrix} + e(2,1)$$

Step2: Propagation

- Start with 2nd row
- Find the ***neighbors*** of the pixel in the previous row
- Find the ***minimum*** among neighbors



Step2: Propagation

- Start with 2nd row
- Find the ***neighbors*** of the pixel in the previous row
- Find the ***minimum*** among neighbors, record it in Path matrix

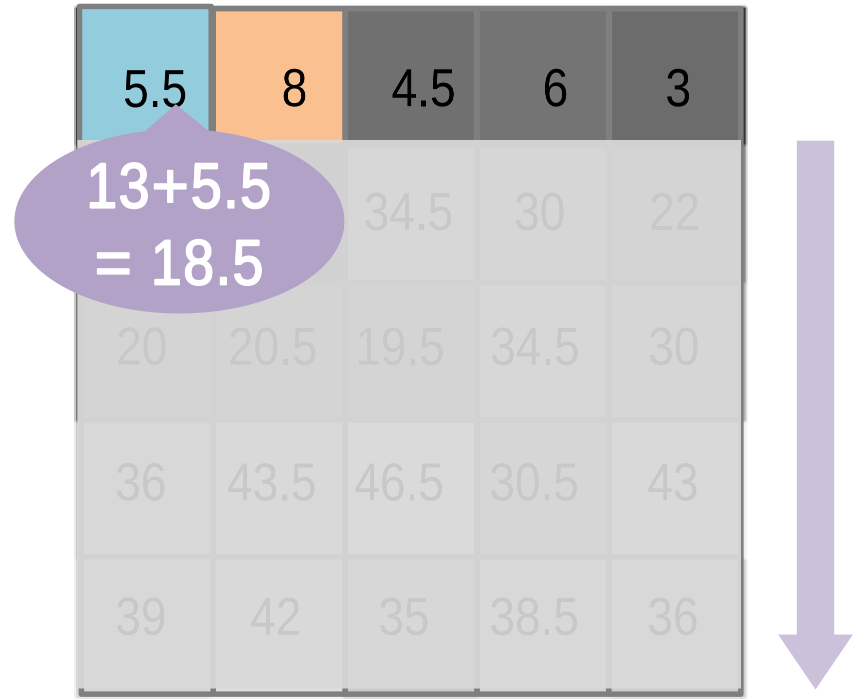
Energy matrix

5.5	8	4.5	6	3
13	9	30	27	19

$$V(2,1) = \min\begin{pmatrix} V(1,1) \\ V(1,2) \end{pmatrix} + e(2,1)$$

$$\begin{matrix} 5.5 \; \surd \\ 8 \\ 13 \end{matrix}$$

Value matrix V



Step2: Propagation

- Start with 2nd row,
- Find the ***neighbors*** of the pixel in the previous row
- Find the ***minimum*** among neighbors
- Add the **energy value** of the pixel with the minimum

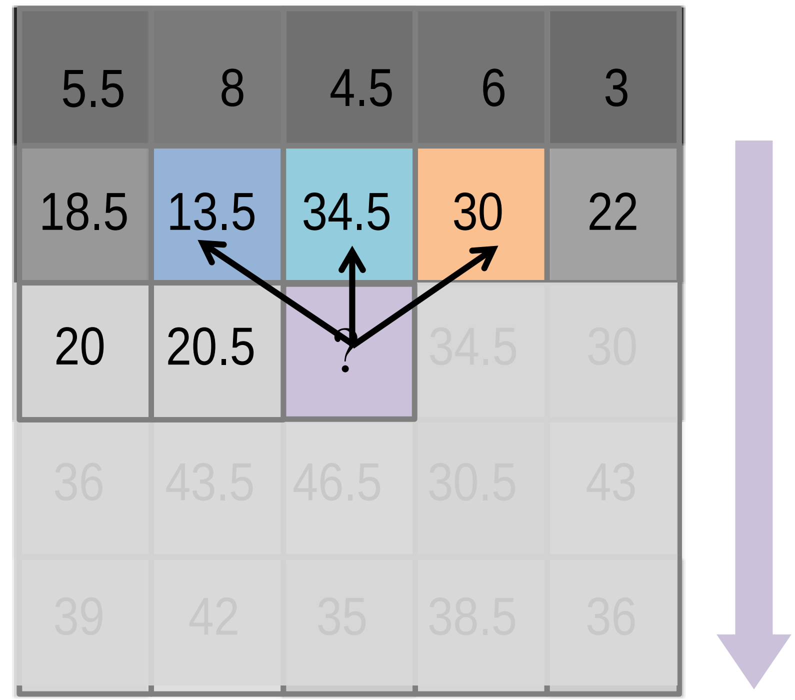
Value matrix V

5.5	8	4.5	6	3
18.5	13.5	34.5	30	22
20	20.5	?	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

Step2: Propagation for every row

Value matrix V

Neighborhoods of the pixel in previous row



Step2: Propagation for every row

- Find the *neighbors* of the pixel in the previous row

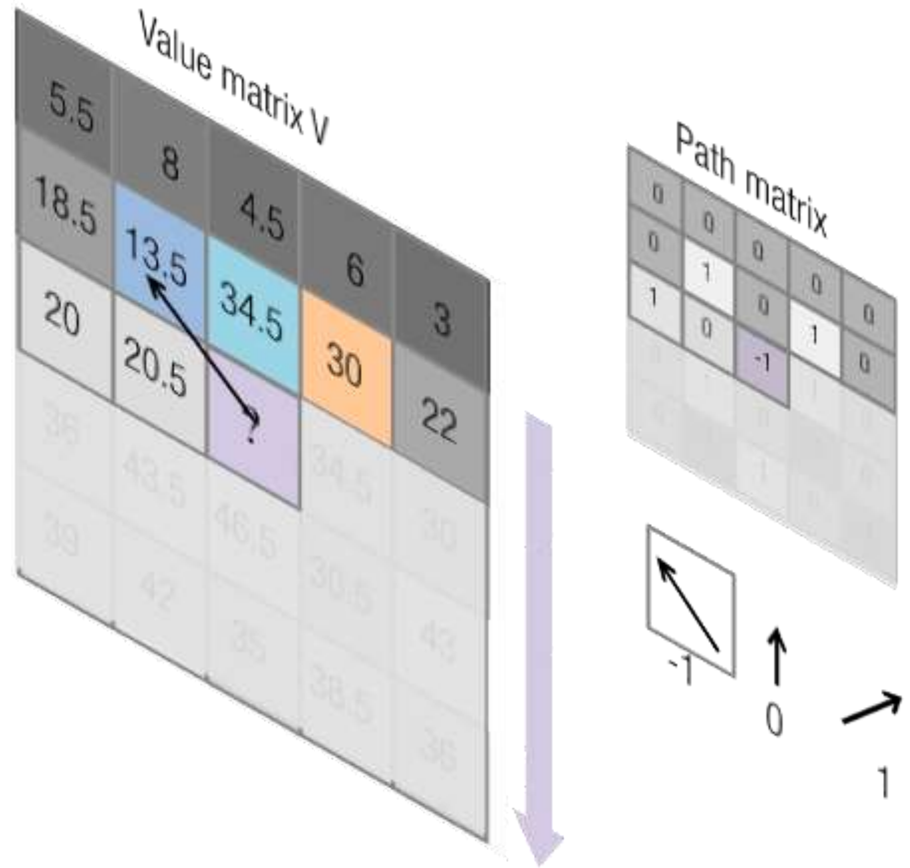
$$\min\begin{pmatrix} V(i-1,j-1) \\ V(i-1,j) \\ V(i-1,j+1) \end{pmatrix} \begin{matrix} 13.5 \ \surd \\ 34.5 \\ 30 \end{matrix}$$

$$+ \quad e(i,j)$$

Value matrix V

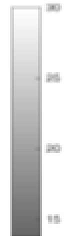
5.5	8	4.5	6	3
18.5	13.5	34.5	30	22
20	20.5	?	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

- Find the ***neighbors*** of the pixel in the previous row
- Find the ***minimum*** among neighbors



- Find the *neighbors* of the pixel in the previous row
- Find the *minimum* among neighbors, record it in Path matrix

5.5	8	4.5	6	3
13	9	30	27	19
6.5	7	6	12.5	8



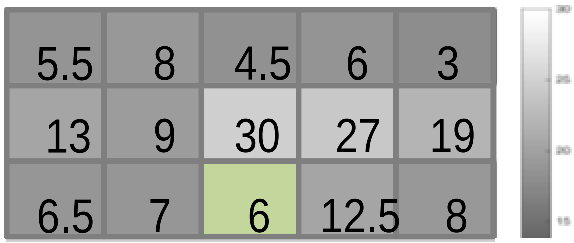
Value matrix V

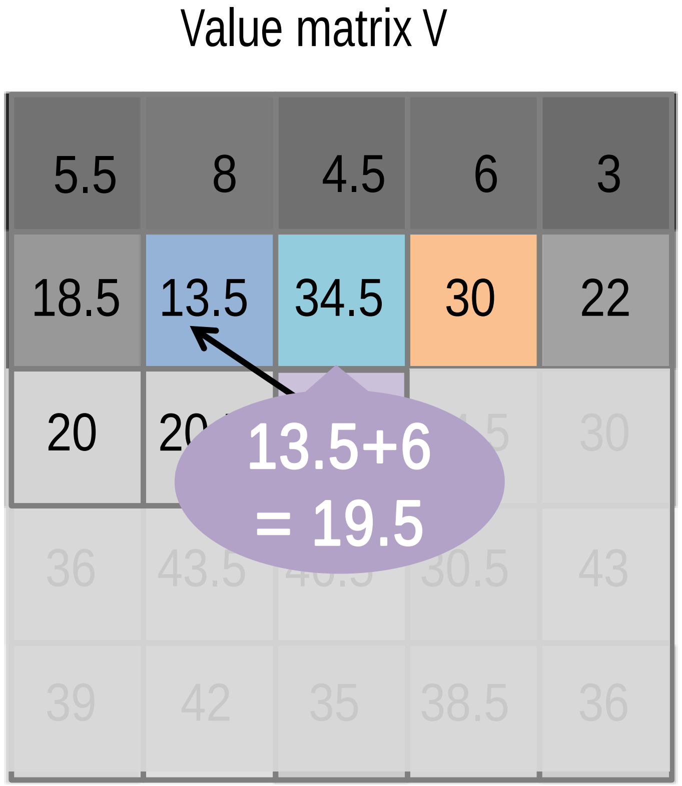
5.5	8	4.5	6	3
18.5	13.5	34.5	30	22
20	20.5	?	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

$$\min\begin{pmatrix} V(i-1, j-1) \\ V(i-1, j) \\ V(i-1, j+1) \end{pmatrix} \begin{matrix} 13.5 & \surd \\ 34.5 & \\ 30 & \end{matrix}$$

$$+ \quad e(i, j) \qquad 6$$

- Find the ***neighbors*** of the pixel in the previous row
- Find the ***minimum*** among neighbors
- Add the **energy value** of the pixel with the minimum





- Find the ***neighbors*** of the pixel in the previous row
- Get the ***minimum*** among neighbors
- Add the **energy value** of the pixel with the minimum

Step3: path resolving

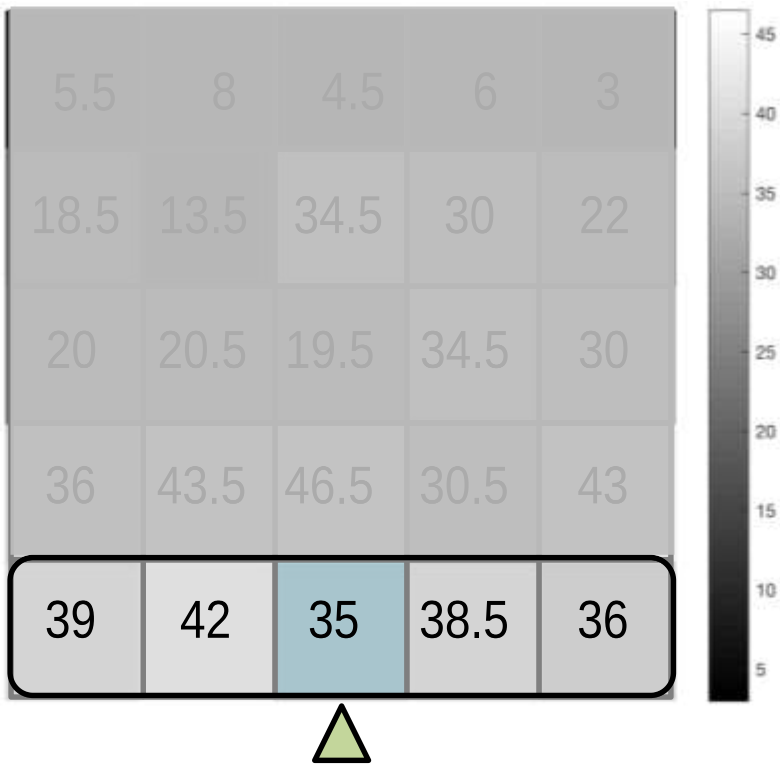
Value matrix

5.5	8	4.5	6	3
18.5	13.5	34.5	30	22
20	20.5	19.5	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

Path matrix

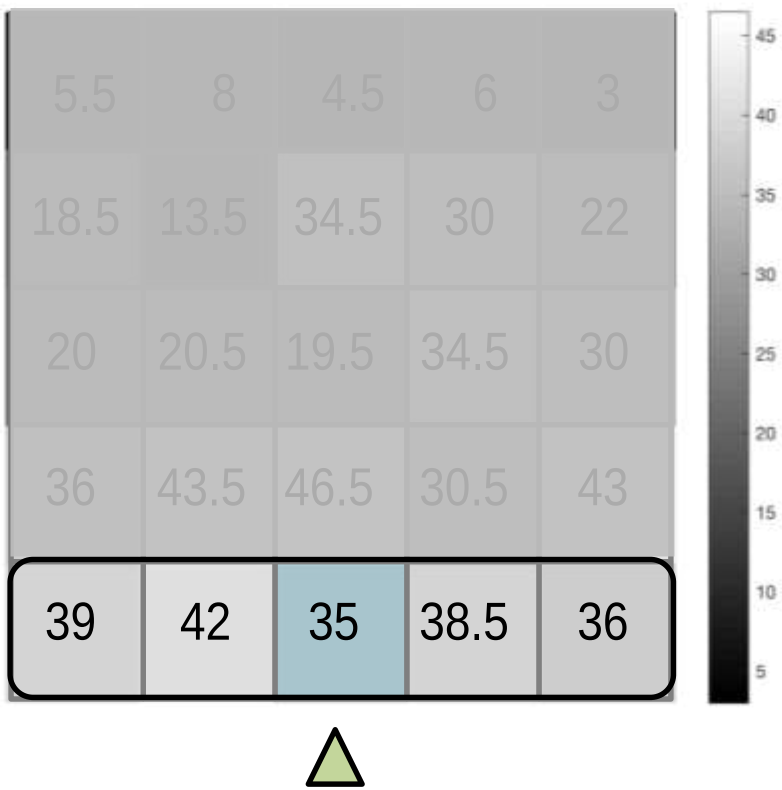
0	0	0	0	0
0	1	0	1	0
1	0	-1	1	0
0	1	0	-1	0
0	-1	1	0	-1

Value matrix

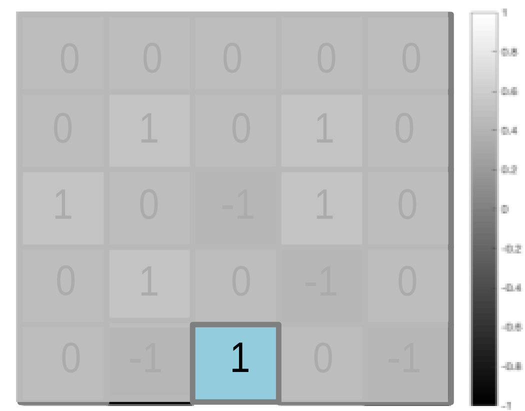


- Find the ***minimum*** of the last row of the Value matrix,
- Find the ***predecessor*** of that pixel

Value matrix

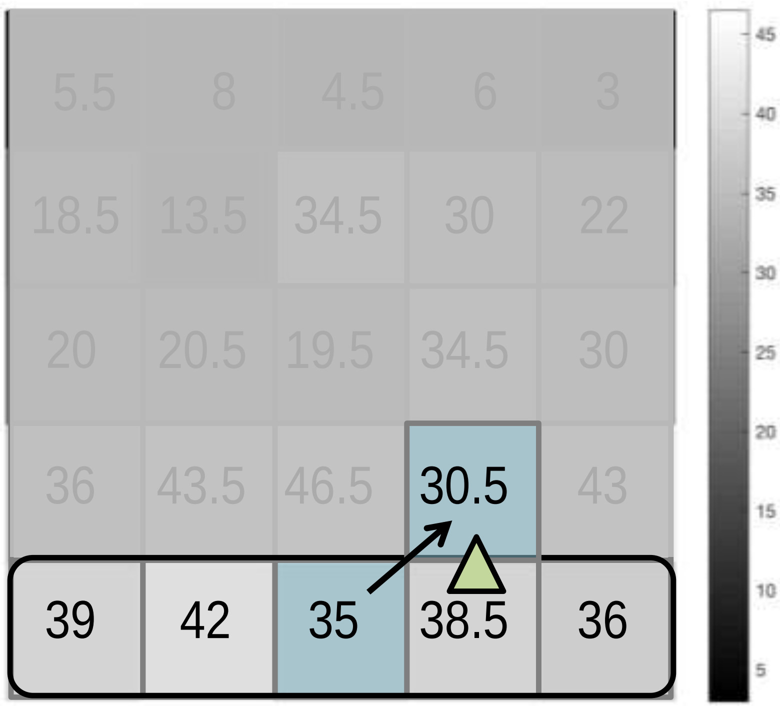


Path matrix

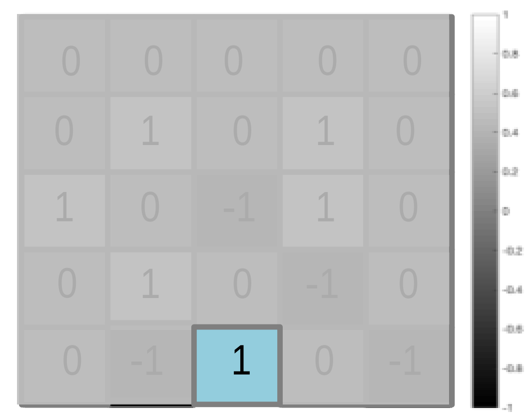


- Find the *minimum* of the last row of the Value matrix,
- Find the *predecessor* of that pixel, using Path matrix

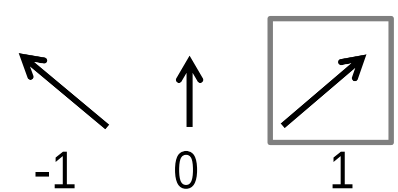
Value matrix



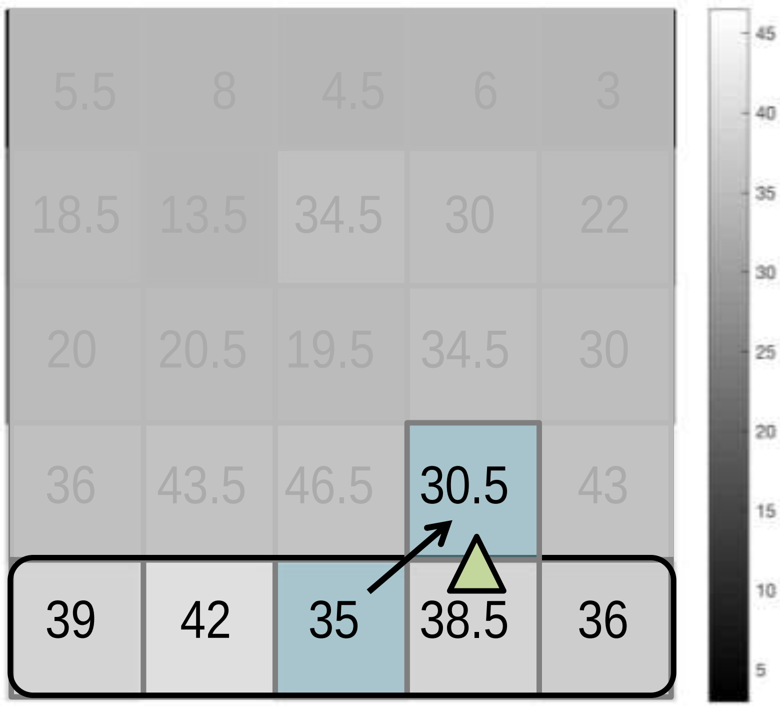
Path matrix



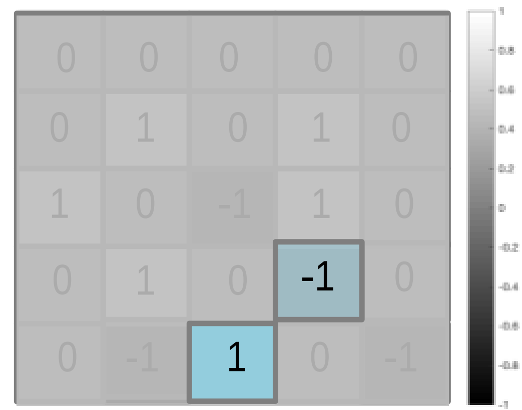
- Move to its ***predecessor***



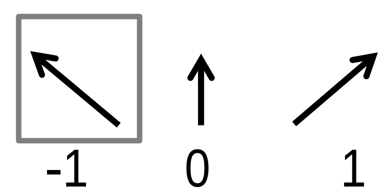
Value matrix



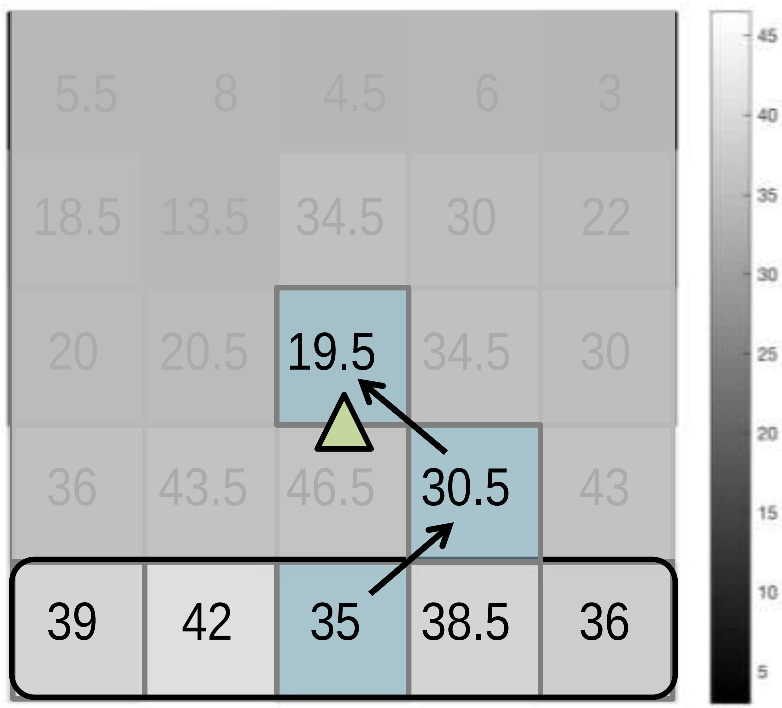
Path matrix



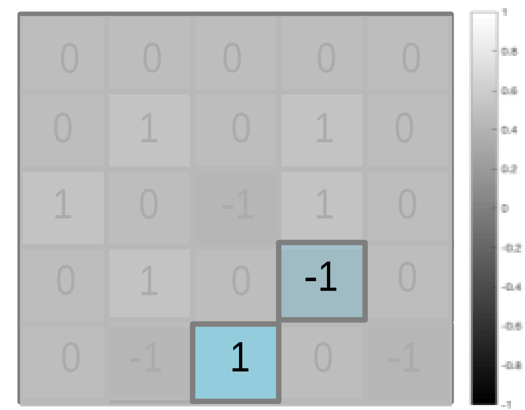
- Follow *predecessor* of current pixel,

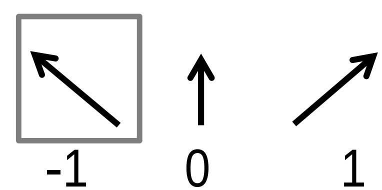


Value matrix



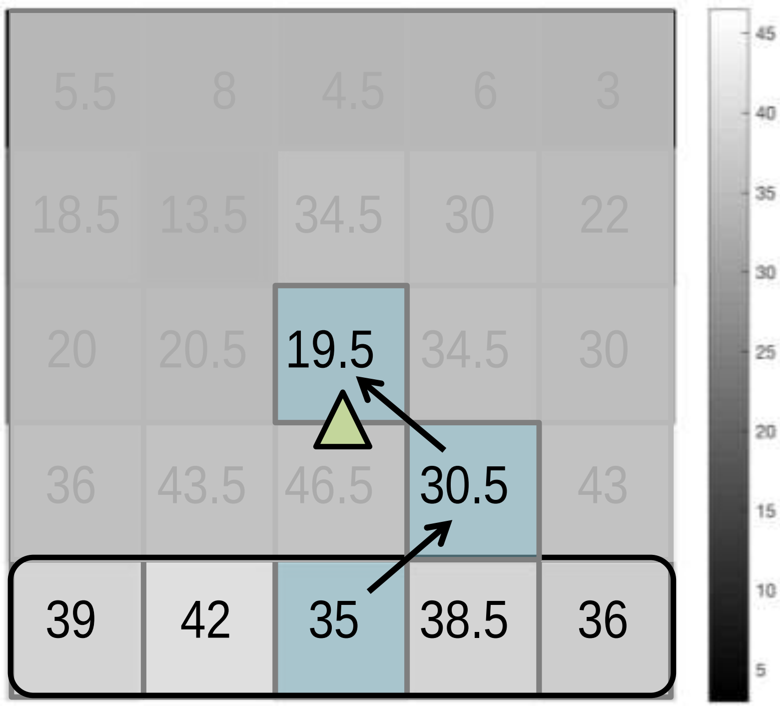
Path matrix



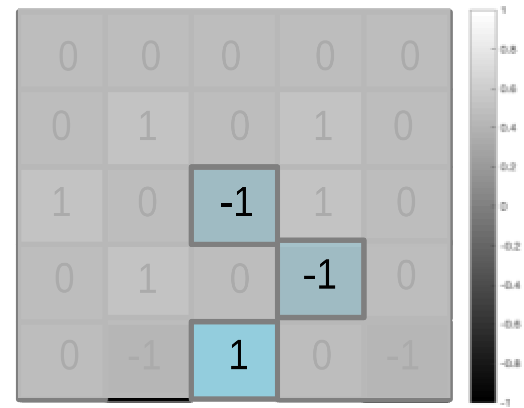


- Move to its ***predecessor***

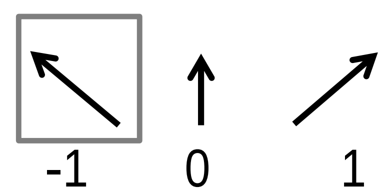
Value matrix



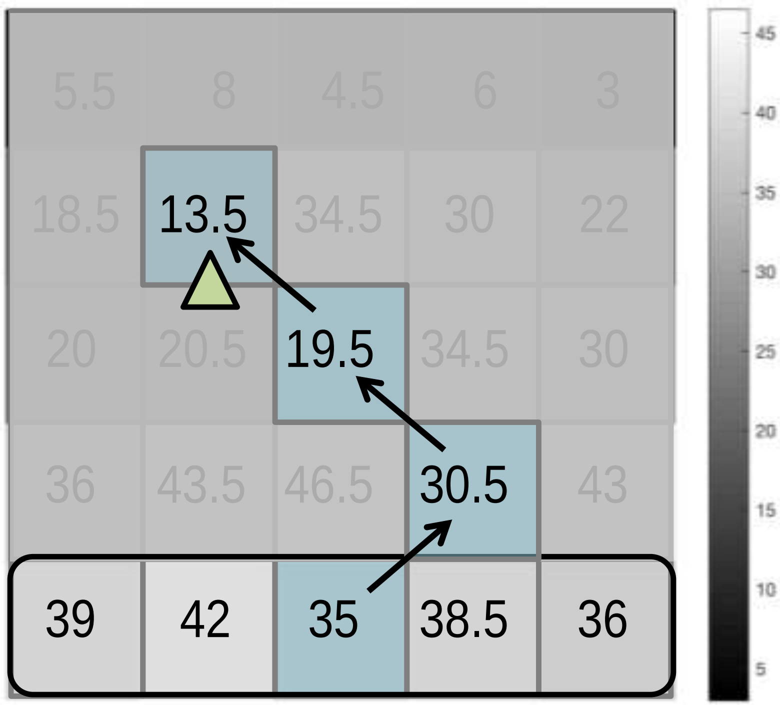
Path matrix



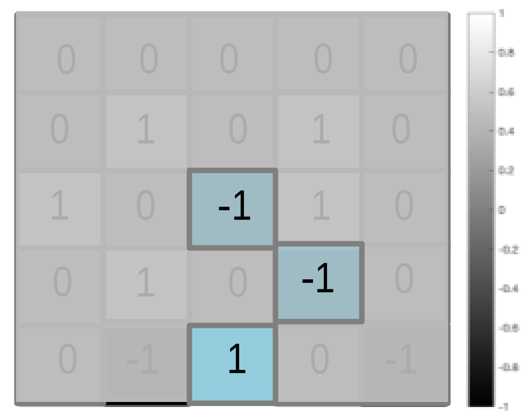
- Find the ***predecessor*** of current pixel



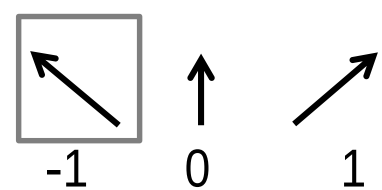
Value matrix



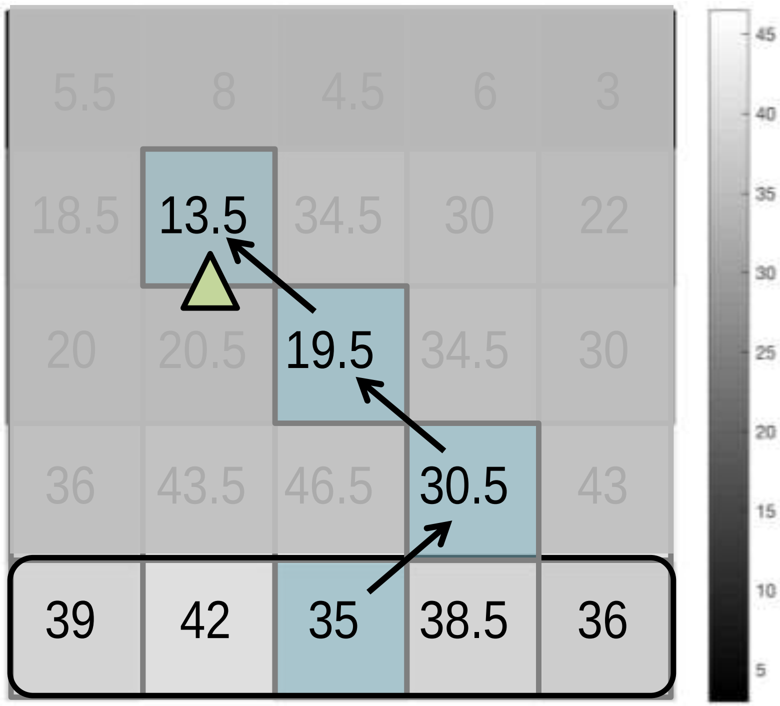
Path matrix



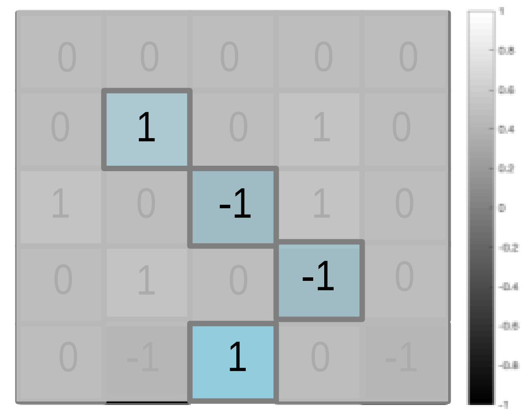
- Move to its *predecessor*



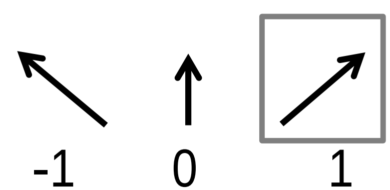
Value matrix



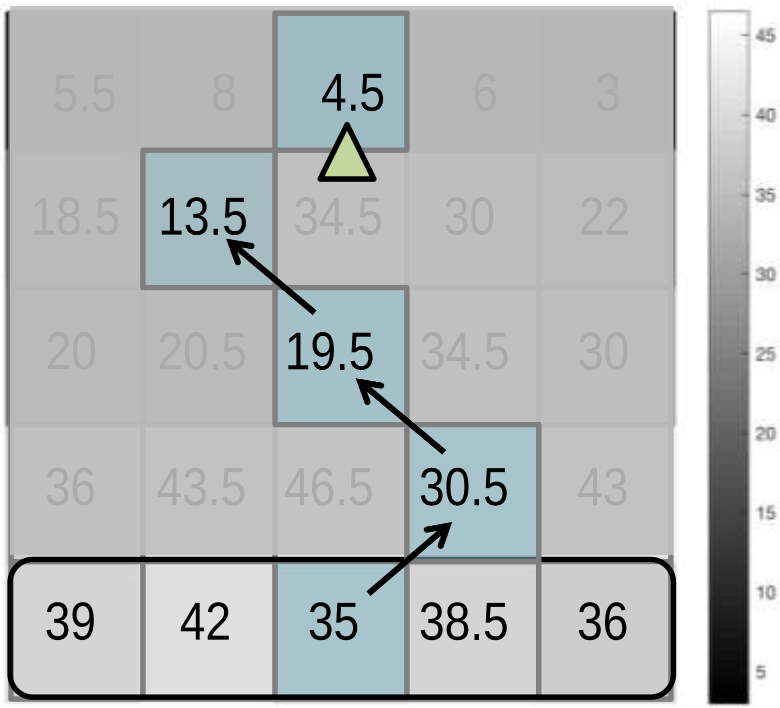
Path matrix



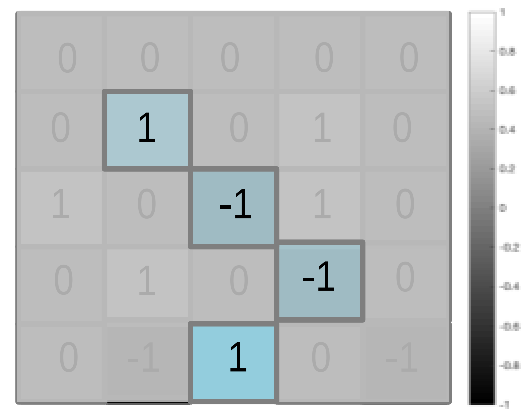
- Find the *predecessor* of current pixel



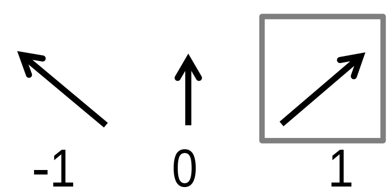
Value matrix



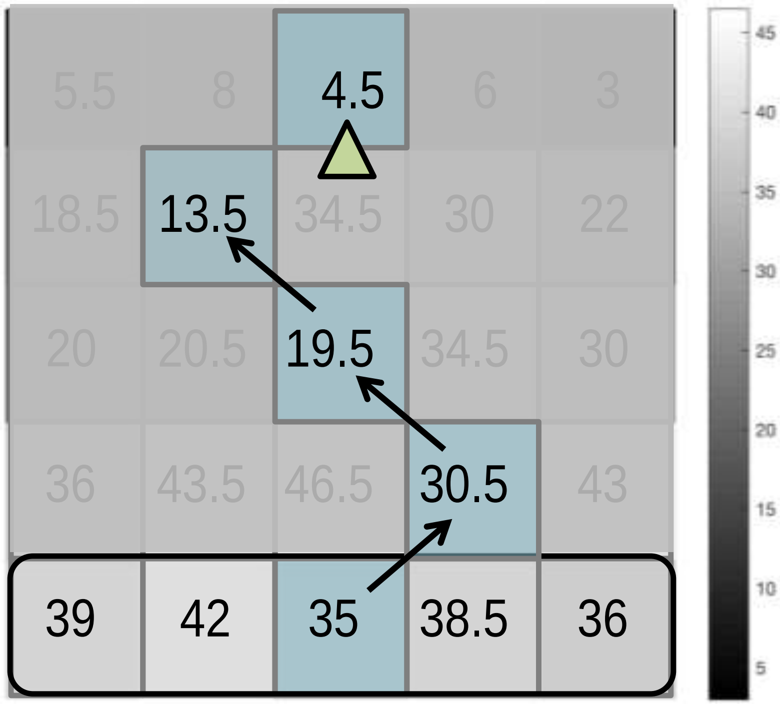
Path matrix



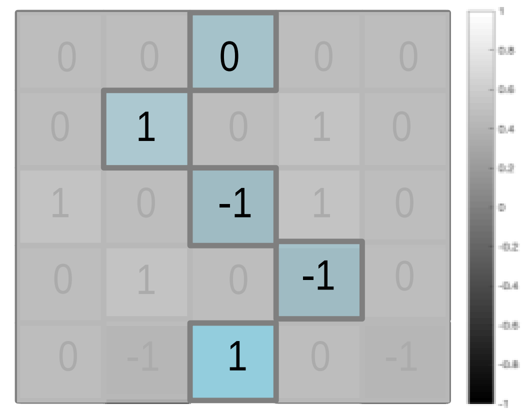
- Stop when reaching the ***first row***



Value matrix

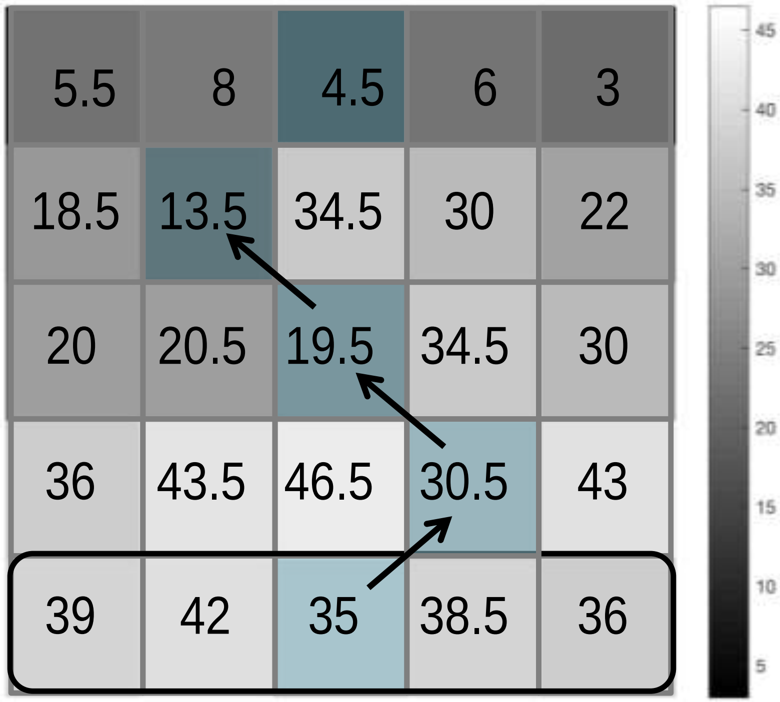


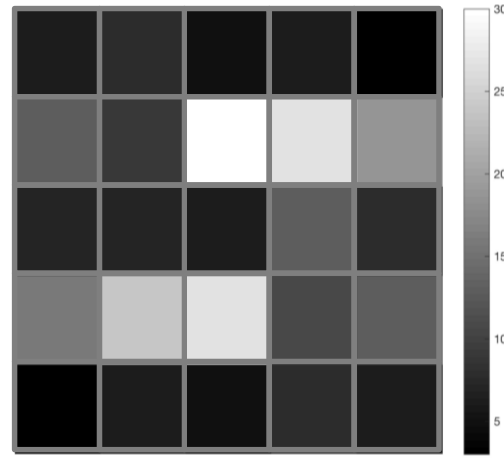
Path matrix



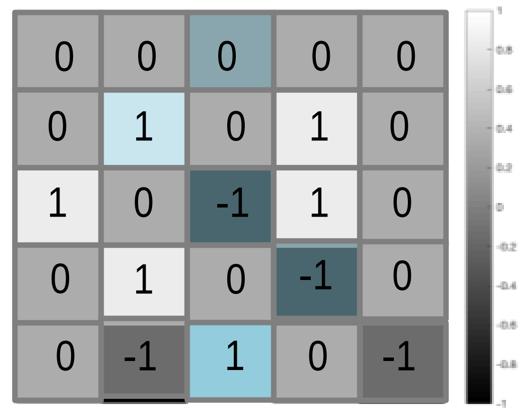
- Seam carving is to delete the path with minimum cost

Value matrix



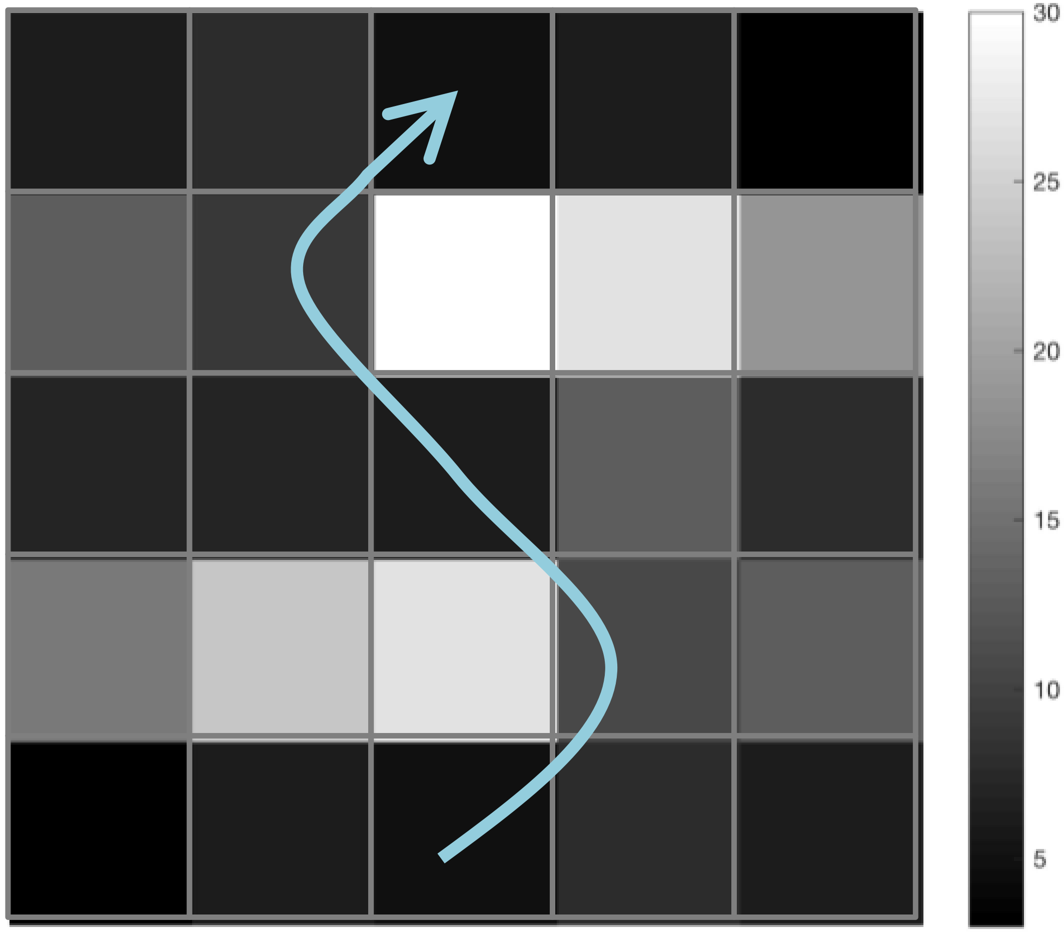


Path matrix



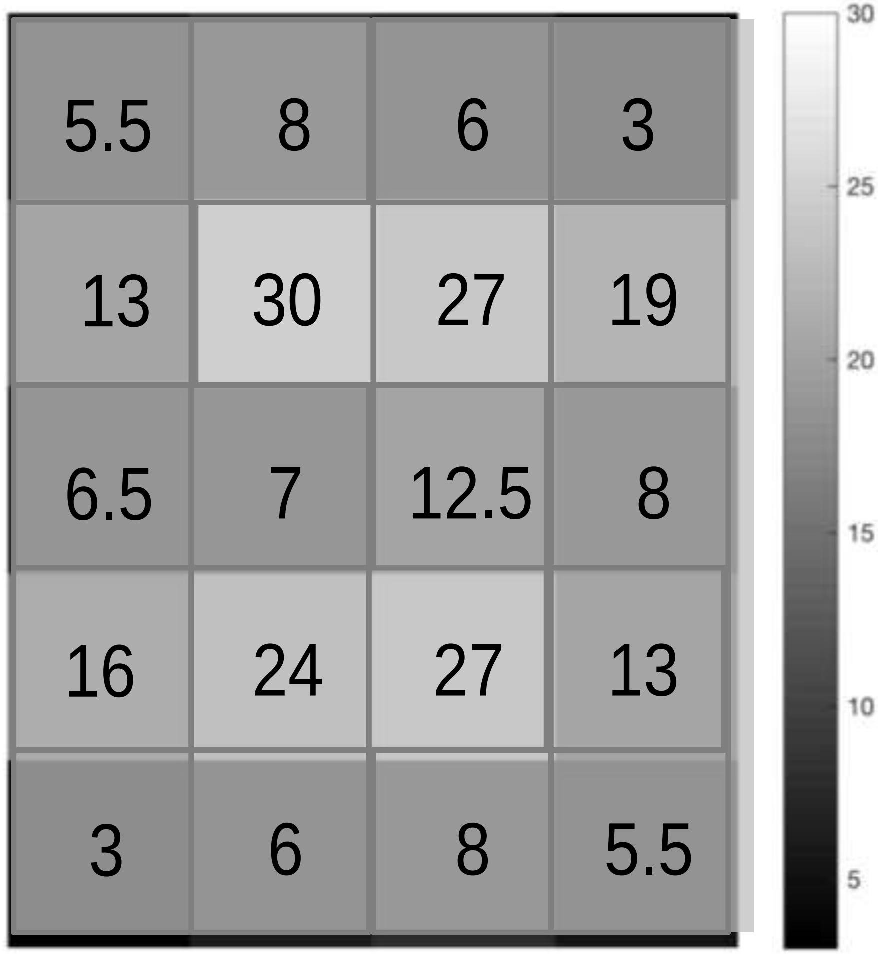
- Seam carving is to delete the path with minimum cost

Energy matrix

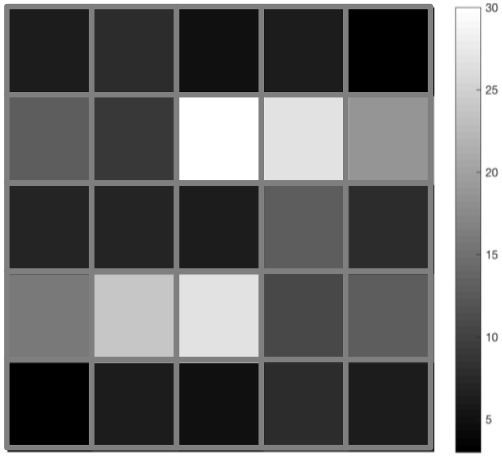


- Seam carving is to delete the path with minimum cost

Carved energy matrix



- Seam carving is to delete the path with minimum cost



Value matrix

5.5	8	4.5	6	3
18.5	13.5	34.5	30	22
20	20.5	19.5	34.5	30
36	43.5	46.5	30.5	43
39	42	35	38.5	36

Path matrix

0	0	0	0	0
0	1	0	1	0
1	0	-1	1	0
0	1	0	-1	0
0	-1	1	0	-1

- What if we want to remove a seam that ends on particular pixel?