Seam Carving

20602 Computer Science (Algorithms)

Final Project

Ivan Golovko DSBA’23

Motivation

- It's often desirable to resize images without preserving the aspect ratio
- Cropping leads to loss of information

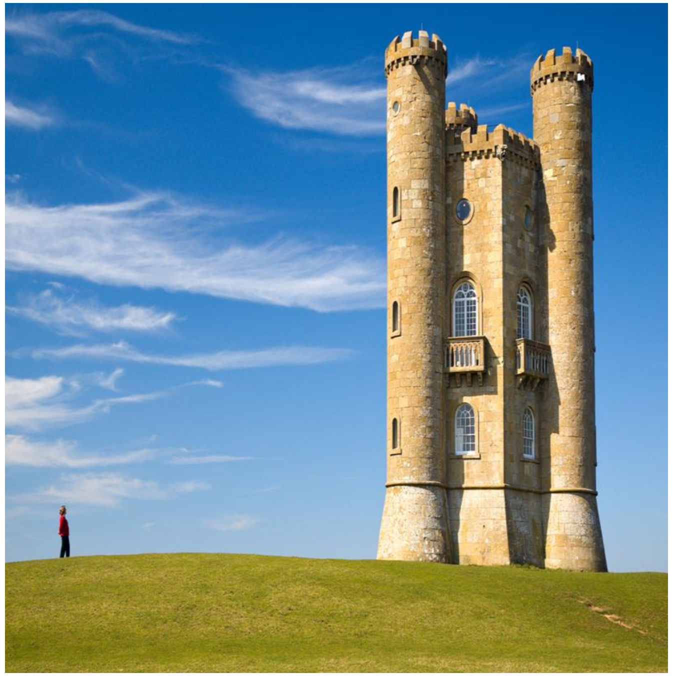
Original image

Cropped image

Motivation

- It’s often desirable to resize images without preserving the aspect ratio
- Rescaling distorts the objects in the picture

Original image



Rescaled image

Motivation

- It's often desirable to resize images without preserving the aspect ratio
- Solution – **seam carving** (a.k.a. "liquid rescaling")

Original image

?

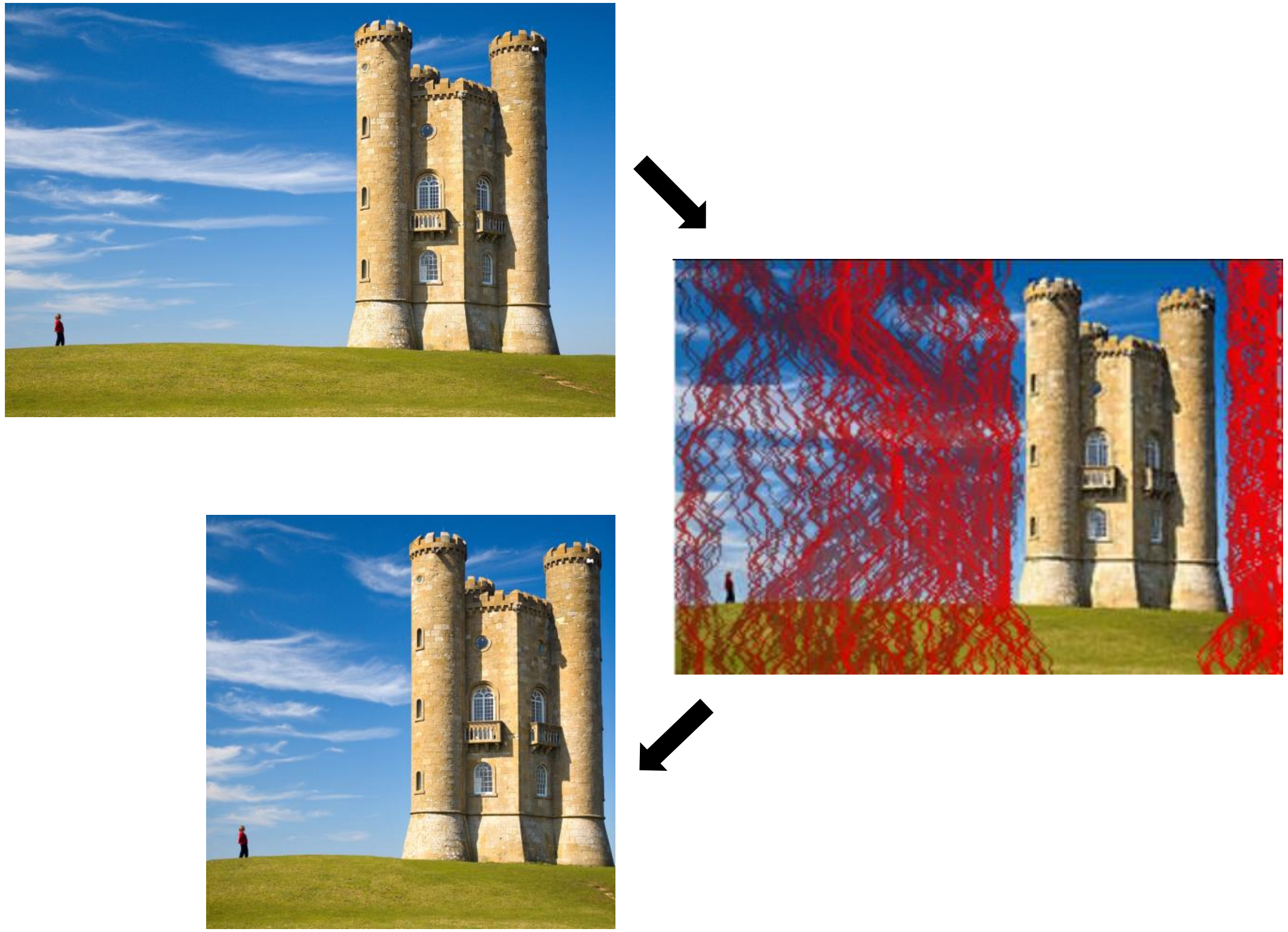
Seam carving result

Seam carving

- Content-aware image resizing algorithm, created in 2005 by scientist from Mitsubishi Electric Research Lab

- General idea – identify and remove the least important areas

- At each step remove continuous paths of pixels (seams) that avoid borders of significant object



Step 1: Energy function

- We want to assign to each pixel certain measure of importance
- Possible solution – gradient magnitude (used widely in image processing)
- Basically, norm of 2-dimensional gradient of the grayscale image
- However, other energy functions might be used (Laplacian, local entropy, etc.)

Gradient magnitude

1. Convert to grayscale

2. Calculate gradient as the following convolution:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I$$

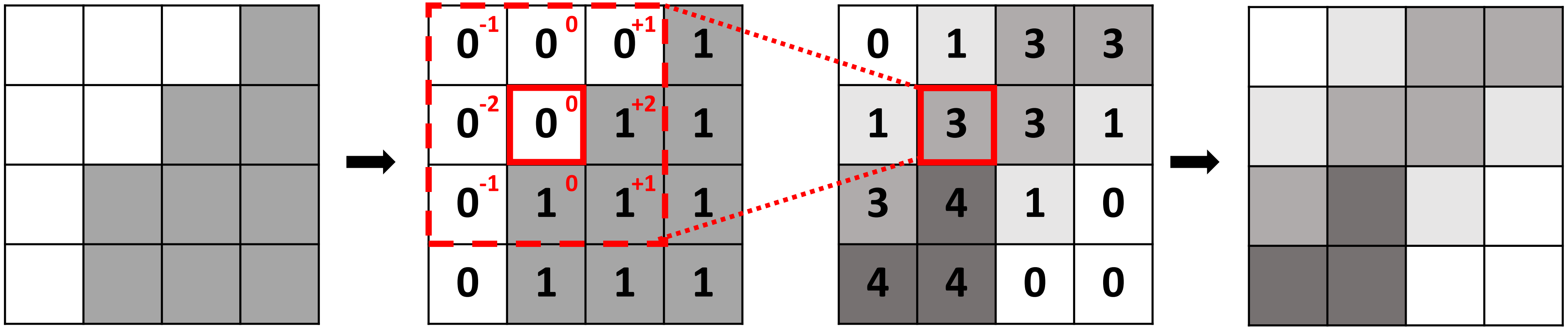
$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

3. Calculate magnitude (L_2- or L_1-norm):

$$G = |G_x| + |G_y|$$

Step 1: Energy Function

Example of computing G_x (for G_y the mechanism is the same):



Step 1: Energy Function

$$|\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$$

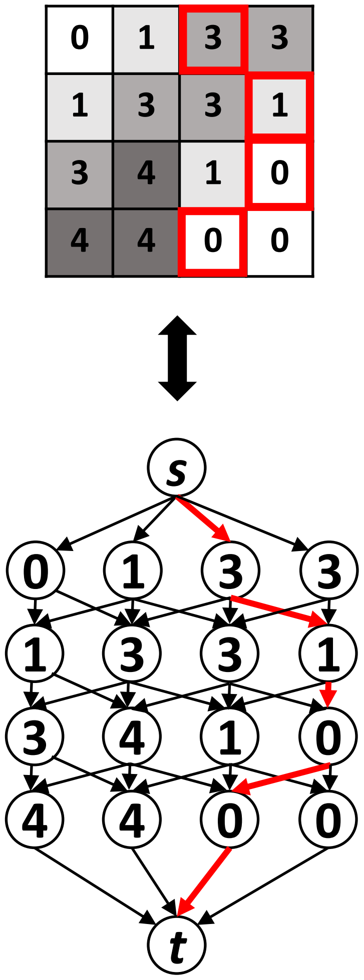
Step 2: Seam with lowest energy

Problem can be formulated as a shortest path problem in an edge-weighted DAG with no negative weights as follows:

1. Graph is constructed as shown in the picture
2. Each edge going into "pixel" vertex has weight equal to energy of this pixel
3. Edges going to ***t*** have weight 0

Then **Dijkstra's algorithm** can be applied

For an image with N pixels ***V ~ N***, ***E ~ 3N***, so running time would be *O(E* log*V)* = ***O(N logN)***



Step 2: Seam with lowest energy

However, in this particular case it’s by far not the best idea, and length of all shortest paths can be found via **simple DP algorithm** with ***O(N)*** running time:

DP algorithm for Seam carving (part 1)

Let d**[i, j]** be length of shortest path to pixel (i, j), and **e[i, j]** – its energy. Then:

1. **d[0, j] = e[0, j]**
2. **d[i, j] = e[i, j] +** ***min*****{d[i-1, j-1] , d[i-1, j] , d[i-1, j+1}** for all i > 0

In the end, length of shortest seam is $\boldsymbol{min_j}$**{d[n, j]}**, i.e. smallest value in last row

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

Shortest paths matrix d[i, j]

0	1	3	3

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

$$1 + \mathit{min}\{0, 1\} \longrightarrow$$

Shortest paths matrix d[i, j]

0	1	3	3
1			

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

$3 + min\{0, 1, 3\}$

Shortest paths matrix d[i, j]

0	1	3	3
1	3		

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	**3**	1
3	4	1	0
4	4	0	0

$$3 + \mathit{min}\{1, 3, 3\} \longrightarrow$$

Shortest paths matrix d[i, j]

0	1	3	3
1	3	**4**	

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

$1 + \mathit{min}\{3, 3\}$

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
$\boxed{3}$	4	1	0
4	4	0	0

$3 + \mathit{min}\{1, 3, 3\} \longrightarrow$

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4
$\boxed{4}$			

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

...

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4
4			

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	**1**	**3**	**3**
1	**3**	**3**	**1**
3	**4**	**1**	**0**
4	**4**	**0**	**0**

Shortest paths matrix d[i, j]

0	**1**	**3**	**3**
1	**3**	**4**	**4**
4	**5**	**4**	**4**
8	**8**	**4**	**4**

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

Length and endpoint of “shortest” seam

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4
4	5	4	4
8	8	$\boxed{4}$	4

Step 2: Seam with lowest energy

Once we have the paths matrix and the endpoint of the shortest seam, we can trace it back to the top recursively:

DP algorithm for Seam carving (part 2)

Let d**[i, j]** be length of shortest path to pixel *(i, j)* and **p[i]** – column index of pixel in the shortest seam in row *i*

1. **p[n] =** $\boldsymbol{argmin_j}$**{d[n, j]}**

2. **p[i-1] =** $\boldsymbol{argmin_{s = \{-1, 0, 1\}}}$**{d[n, p[i] + s]}** for all i < n

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4
4	5	4	4
8	8	4	4

Length and endpoint of “shortest” seam

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4
4	5	4	4
8	8	4	4

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4
4	5	4	4
8	8	4	4

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

Shortest paths matrix d[i, j]

0	1	3	3
1	3	4	4
4	5	4	4
8	8	4	4

Step 2: Seam with lowest energy

Energy matrix e[i, j]

0	1	3	3
1	3	3	1
3	4	1	0
4	4	0	0

Seam to be removed

Shortest paths matrix d[i, j]

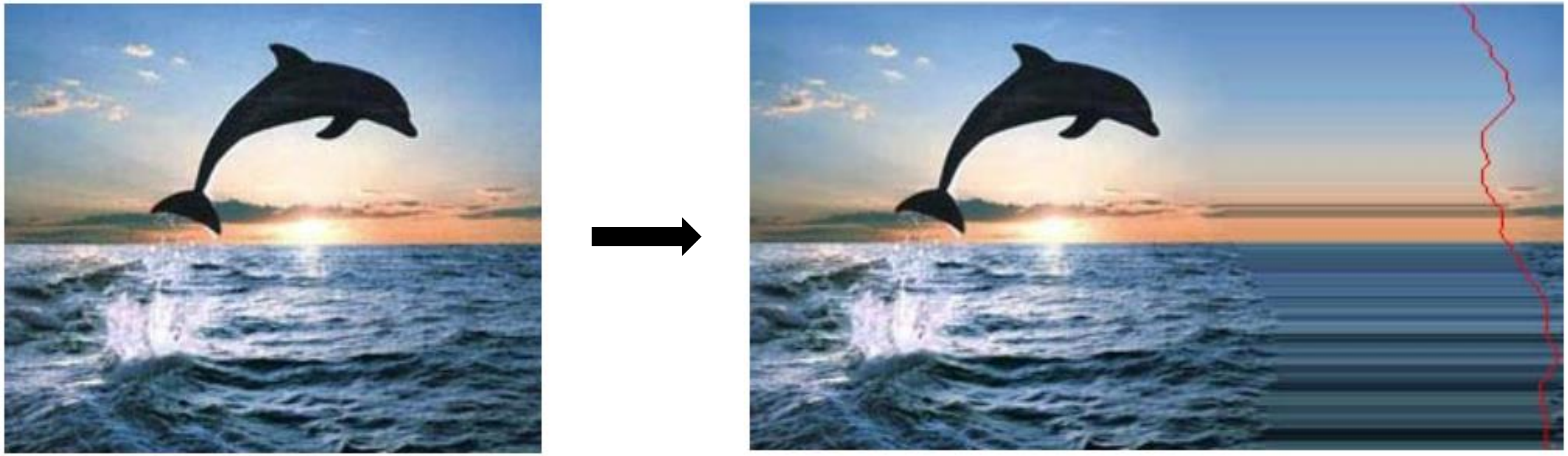
0	1	3	3
1	3	4	4
4	5	4	4
8	8	4	4

Results

Algorithm proceeds iteratively removing seams one by one, until the desired degree of compression is achieved. The process in progress looks like this:

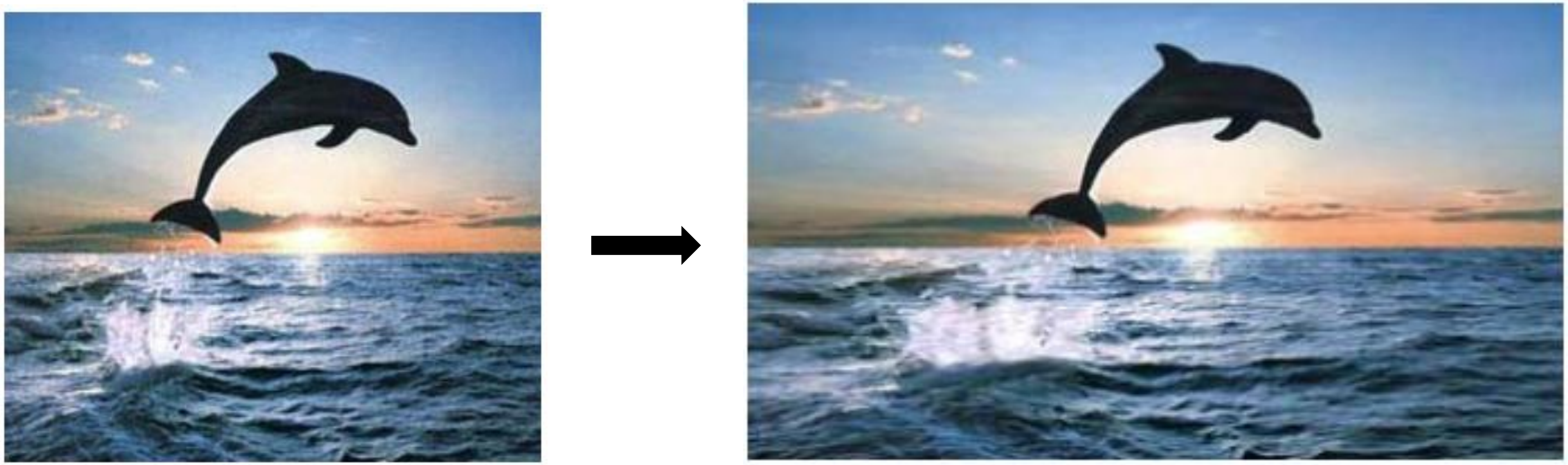
SC for image enlargement

- Seam carving algorithm can be easily adapted for enlarging images with similar idea – **insert copies of least important seams**
- However, a caveat is that if done straightforwardly, one seam will be inserted repeatedly, causing strange results:



SC for image enlargement

- Solution – first proceed like if we wanted to delete same number of seams
- Insert one copy of each seam that was removed in the first step



Extensions

- Very flexible, can work with different energy functions
- Can preserve the marked areas (assigns large costs)
- Can do object removal (remove seams until all the marked pixels disappear)
- However, doesn’t always produce good-looking results

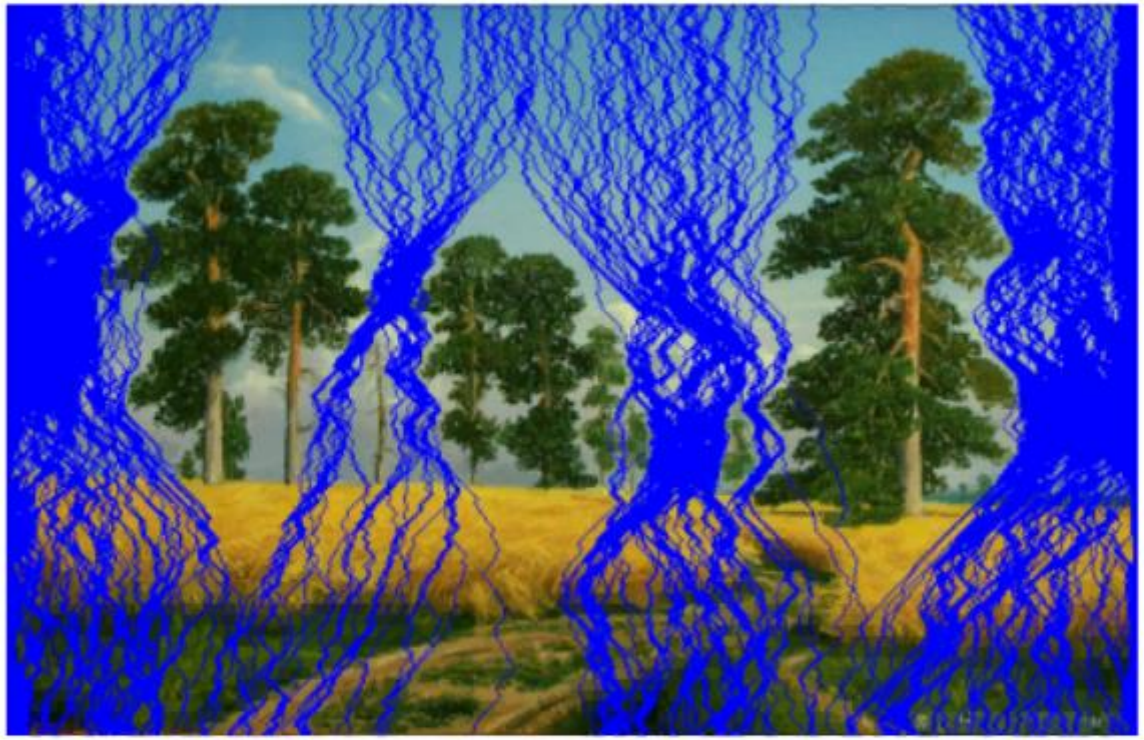
Some examples

Original image

Rescaled (x0.6)

Some examples

Seams to be deleted



SC-compressed (x0.6)

Some examples

Rescaled (x1.3)

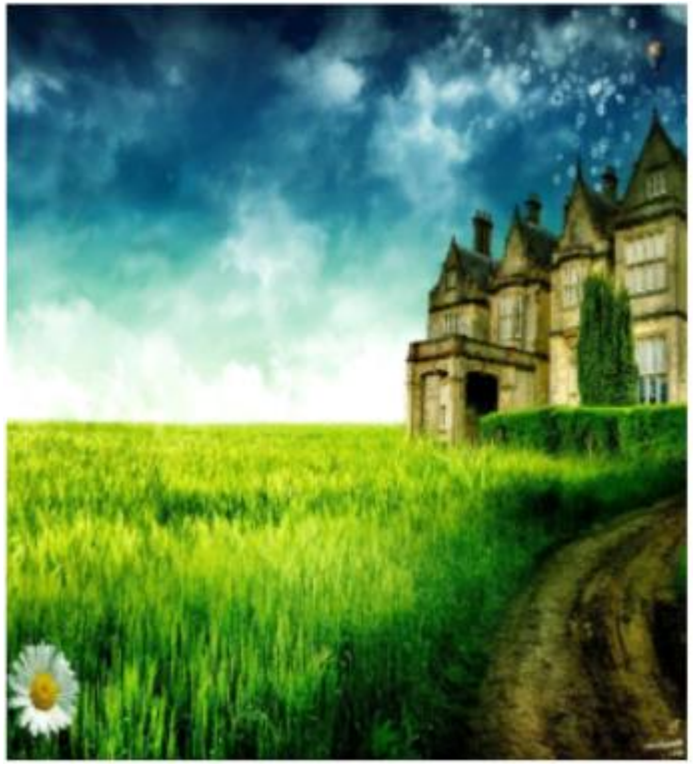
Some examples

SC-enlarged (x1.3)

Some examples

Original image

Rescaled (x0.6)



Some examples

Seams to be deleted

SC-compressed (x0.6)

Some examples

SC-compressed (x0.6)

Rescaled (x0.6)

References

- Original paper by Avidan & Shamir:
  https://perso.crans.org/frenoy/matlab2012/seamcarving.pdf

- GitHub repository with my implementation:
  https://github.com/igolovko3/seam_carving

Discussion: Compressing in two dimensions

Assume we want to remove both vertical and horizontal seams from an image**.** A priori, it is not clear what should the order of removal be (horizontal first, vertical first, alternately, etc.)

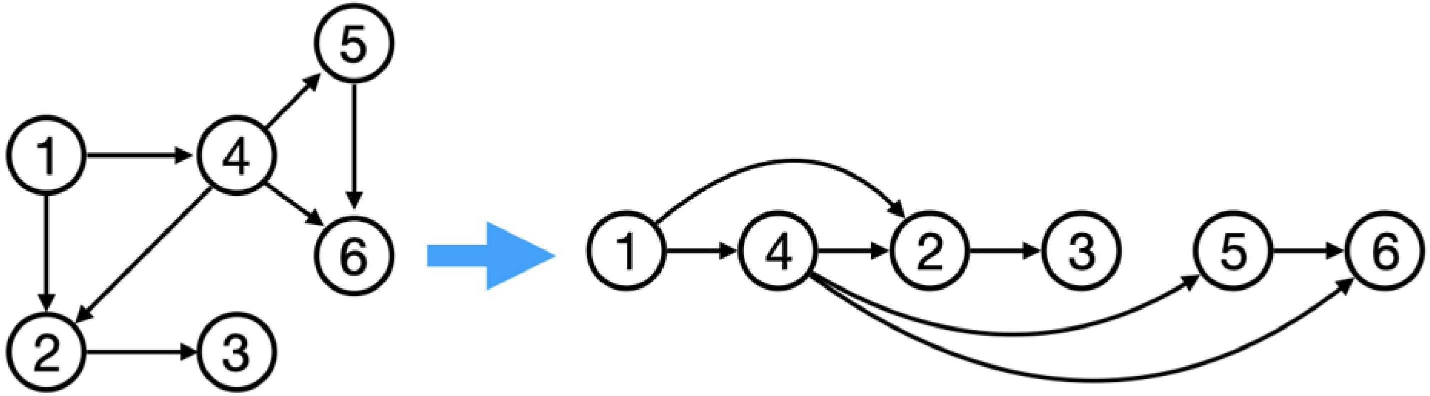
Suppose ***C[i, j]*** – minimal cost of removing ***i*** horizontal and ***j*** vertical seams, Im[i, j] is the "optimal" image of corresponding size. Then:

1. **C[0, j]** and C[i, 0] are known for any i and j
2. **C[i, j]** = ***min***{
   "Cost to remove one horizontal seam from **Im[i - 1, j]**" + **C[i - 1, j]**,
   "Cost to remove one vertical seam from **Im[i, j - 1]**" + **C[i, j - 1]**
   } for all i, j > 0

Discussion: Topological sort VS DP

- Topological sorting – linear ordering of vertices of a DAG, such that if there is an edge ***v* → *u***, then ***v*** comes before ***u*** in that ordering

- Can be done in *O(V + E)* by a modification of a DFS

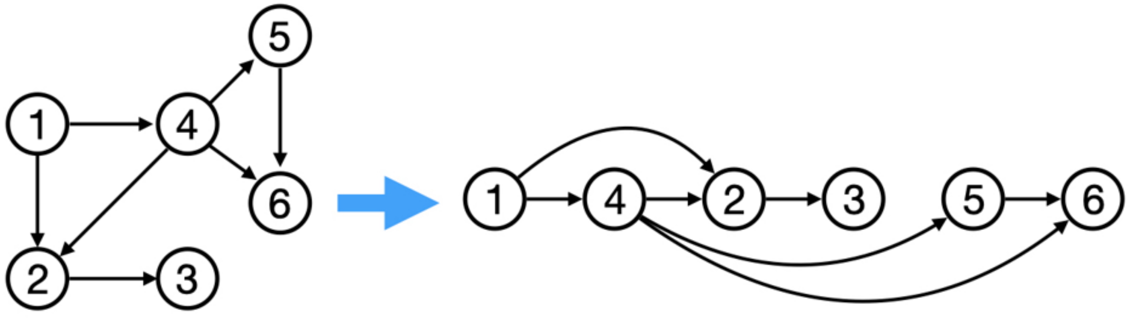


Discussion: Topological sort VS DP

Can be applied to solve shortest path problem:

1. Get topological sort of graph – $O(E + V)$
2. Proceed like in Dijkstra's, but instead of searching for vertex with smallest distance to ***s*** on each step, just follow the topological order – $O(E + V)$

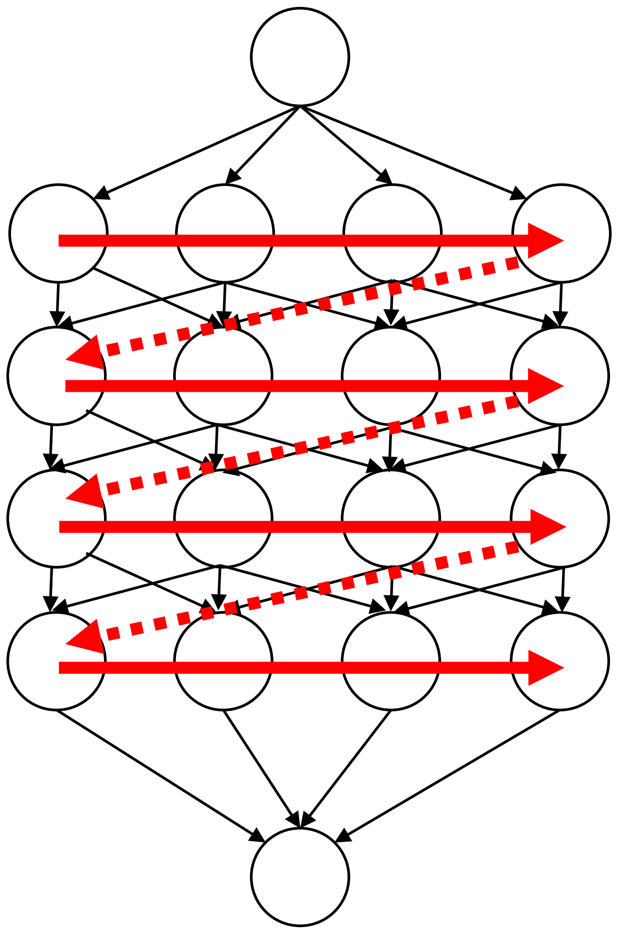
Correct, since any vertex ***v*** is unreachable from any of the following vertices, so shortest path to it doesn't depend on what happens after



Discussion: Topological sort VS DP

In the seam carving problem, graph is structured so that going over it **row by row** yields a **topologically-ordered traversal**, since all edges go "down"

Thus, start with step 2 right away



Discussion: Topological sort VS DP

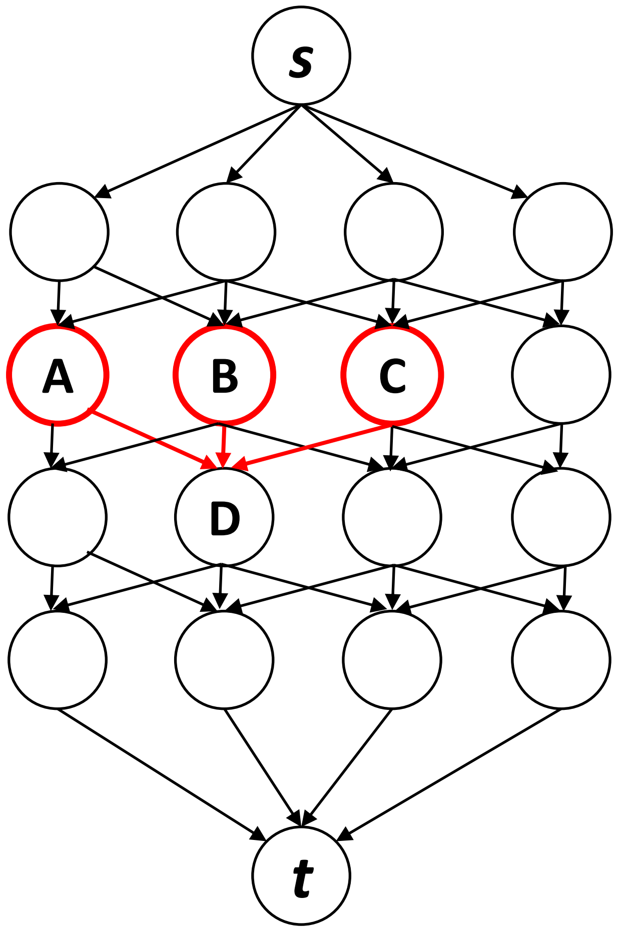
- Let d[v] be the length of shortest path from ***s*** to ***v***

- After relaxing edges from A, B & C, **d[D]** will be implicitly be set to:

  d[D] = *min*{d[A] + |AD|, d[B] + |BD|, d[C] + |CD|}

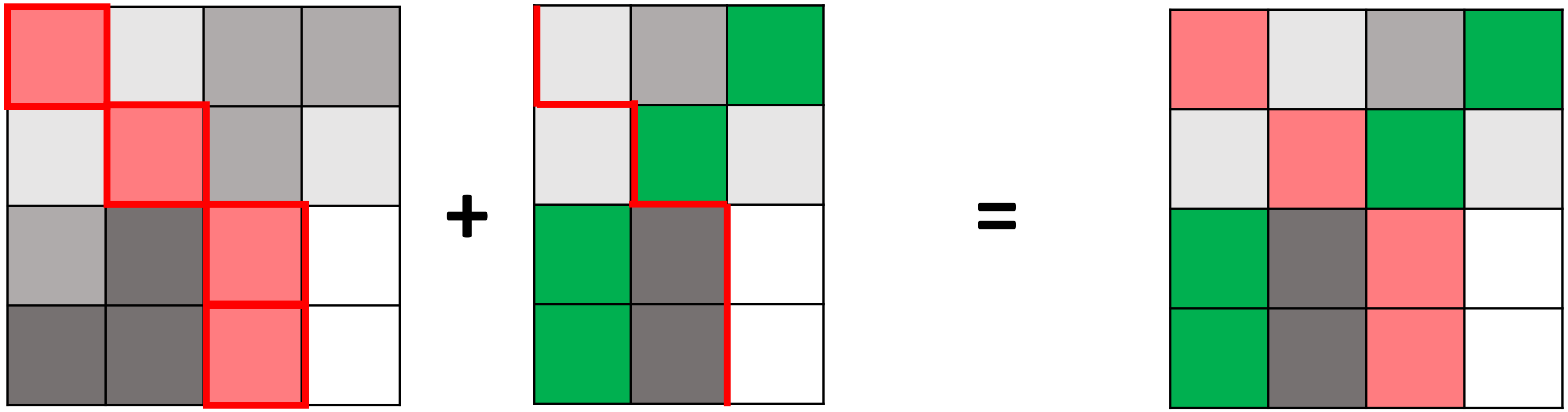
- However, since |AD|=|BD|=|CD|= *energy*(D), this is the exact same expression as in DP algorithm

- They are **absolutely equivalent!**



Discussion: Fancy fact about seams

Though each seam is continuous on the image from which it is deleted, on the original image they might cross each other and “tear”



Discussion: Fancy fact about seams

- However, it can be shown that any set of number of seams obtained by iterative removals can be replaced by the same number of non-crossing continuous seams on the original image, covering same set of pixels

- Idea – each time take "left envelope" of the set of pixels, proceed iteratively