Seam Carving

Design and Analysis of Algorithms by Dr. Entezari

99-400 Fall
Mohammad Amin Shahidi
96522177

Table Of Contents

What is used

- Python 3.6
- Pillow module (image editing)
- Numpy module (to use matrix as DP table)
- Python built-in Math module

Content Aware Image Resizing

- Seam carving (or liquid rescaling) is an algorithm for content-aware image resizing. It functions by establishing a number of seams (paths of least importance) in an image and automatically removes seams to reduce image size or inserts seams to extend it.
- The purpose of the algorithm is image retargeting, which is the problem of displaying images without distortion on media of various sizes (cell phones, projection screens) using document standards, like HTML, that already support dynamic changes in page layout and text but not images

Content Aware Image Resizing

Code Stages

Functions detail

Functions:

- energy(x,y,pix)
- energy_map()
- create_dp_table()
- find_lowest_cost_seam()
- reshape()

energy(x,y,pix)

- This function implements “Dual-gradient” technique for a given pixel defined by two given arguments, x and y.

$$energy(x,y) = \sqrt{\Delta_x^2(x,y) + \Delta^2 y(x,y)}$$
$$\Delta_x^2(x,y) = R_x(x,y)^2 + G_x(x,y)^2 + B_x(x,y)^2$$
$$R_x(x,y) = R(x+1,y) - R(x-1,y)$$
$$R(x,y) : Red\ value\ at\ pixel\ x,y.$$

- “pix” is the target image which must be pixelized by .load() function before assigning pix argument.

energy(x,y,pix)

- Because for a given pixel the values of R , G and B are in [0,255] range, the maxmimum value of a pixel’s calculated energy is 624.6 (using dual-gradient method)

Functions detail

Functions:

- energy(x,y,pix)
- energy_map()
- create_dp_table()
- find_lowest_cost_seam()
- reshape()

energy_map()

- This function, uses energy(x,y,pix) function to calculate energy for all the pixels in main image. It creates a image which every pixel (x,y) of it represents the energy of a pixel in main image with same (x,y). brighter (more white) means more energy.
- this image is named energy_map.png.

energy_map()

- If the energy of a pixel is 450, we convert it in [0,255] range to be able to be the value of (R,G,B) of that pixel.
- Example: 450 * (255/624) = 183.8
- For more simplicity we use math.floor() function
- Example: floor(183.8) = 183

Functions detail

Functions:

- energy(x,y,pix)
- energy_map()
- create_dp_table()
- find_lowest_cost_seam()
- reshape()

create_dp_table()

- It creates a matrix of zeros with same size of the image using numpy.zeros(N*M) function.
- Each cell represents the lowest cost path starting that at that cell and ending on the bottom of image
- For each cell, the value is calculated like billow:

value(x,y)=min(value(x,y+1) , value(x+1,y+1) , value(x-1,y+1)) + value(x,y)

DP table

5	10	6	2	5	1

Energy map

12	7	2	2	5	11
8	3	12	0	4	6
3	6	9	14	1	2
5	10	6	2	5	1

DP table

8	11	11			
5	10	6	2	5	1

Energy map

12	7	2	2	5	11
8	3	12	0	4	6
3	6	9	14	1	2
5	10	6	2	5	1

11 = min(10,6,2) + 9

DP table

8	11	11	16	2	3
5	10	6	2	5	1

Energy map

12	7	2	2	5	11
8	3	12	0	4	6
3	6	9	14	1	2
5	10	6	2	5	1

DP table

16	11	23	2	6	8
8	11	11	16	2	3
5	10	6	2	5	1

Energy map

12	7	2	2	5	11
8	3	12	0	4	6
3	6	9	14	1	2
5	10	6	2	5	1

DP table DONE!

23	18	4	4	7	17
16	11	23	2	6	8
8	11	11	16	2	3
5	10	6	2	5	1

Energy map

12	7	2	2	5	11
8	3	12	0	4	6
3	6	9	14	1	2
5	10	6	2	5	1

Functions detail

Functions:

- energy(x,y,pix)
- energy_map()
- create_dp_table()
- find_lowest_cost_seam()
- reshape()

find_lowest_cost_seam()

- **Each time it runs, it finds the lowest-cost path from top to down using already generated DP table. How?**
- **Starting at the first row, the lowest-valued cell represents the starting point of the lowest-cost path, so, at first we find the lowest-valued cell of the first row and go all the way down using 3 lower cells. Also we change the value of that starting cell to 1000000000000 cause we don’t want it to be the starting point of the next lowest-cost path.(if it happen, the two paths will be the same!)**

DP table

Find lowest-valued cell of first row →

	0	1	2	3	4	5
0	**23**	**18**	**4**	**4**	**7**	**17**
1	**16**	**11**	**23**	**2**	**6**	**8**
2	**8**	**11**	**11**	**16**	**2**	**3**
3	**5**	**10**	**6**	**2**	**5**	**1**

Path = [(2,0)]

DP table

Find min(11 , 23 , 2) →

	0	1	2	3	4	5
0	23	18	infinity	4	7	17
1	16	11	23	2	6	8
2	8	11	11	16	2	3
3	5	10	6	2	5	1

Path = [(2,0) , (3,1)]

DP table

Find min(11 , 16 , 2) ⟶

	0	1	2	3	4	5
0	23	18	infinity	4	7	17
1	16	11	23	infinity	6	8
2	8	11	11	16	2	3
3	5	10	6	2	5	1

Path = [(2,0) , (3,1) , (4,2)]

DP table

	0	1	2	3	4	5
0	23	18	infinity	4	7	17
1	16	11	23	infinity	6	8
2	8	11	11	16	infinity	3
3	5	10	6	2	5	1

Find min(2 , 5 , 1) →

Path = [(2,0) , (3,1) , (4,2) , (5,3)]

DP table

For better visibility ,using calculated Path list we turn the pixels with (x,y) existing in Path to (255,0,) to be a red pixel.
Repeating this for all the elements in Path list results a “Red Seam” shining in the energy-map!
We continue finding more Paths for more seams.
To do this we call the **find_lowest_cost_seam()** function multiple times (reduce_amount)

	0	1	2	3	4	5
0	23	18	infinity	4	7	17
1	16	11	23	infinity	6	8
2	8	11	11	16	infinity	3
3	5	10	6	2	5	infinity

The seam: **Path = [(2,0) , (3,1) , (4,2) , (5,3)]**

Functions detail

Functions:

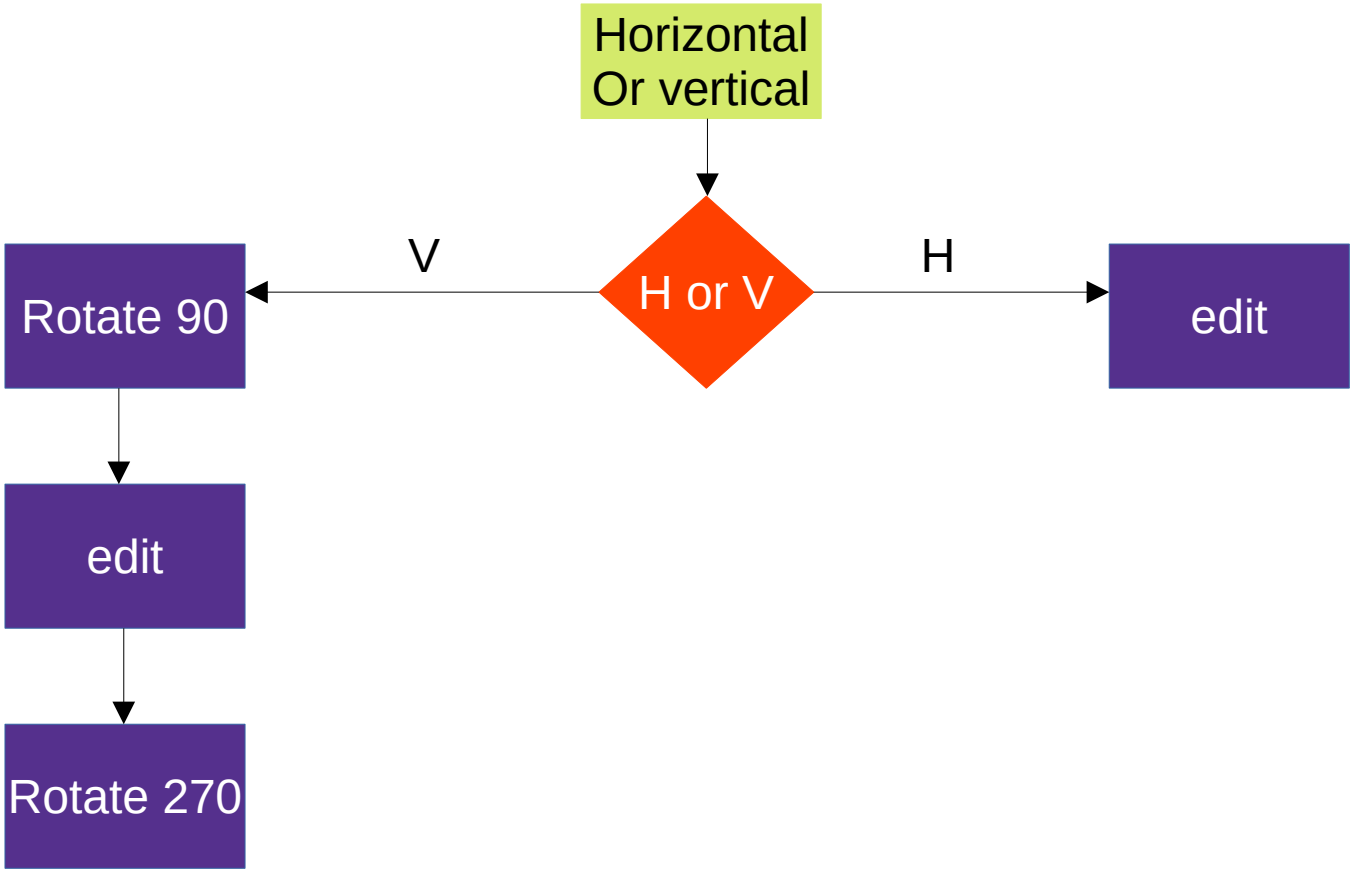
- energy(x,y,pix)
- energy_map()
- create_dp_table()
- find_lowest_cost_seam()
- reshape()

reshape()

- After finding seams, to remove them, in each row we change the (R,G,B) of the pixel (x,y) to the (R,G,B) of the pixel (x + a ,y).
- “a” is the number of red pixels at the left (including the pixel itself) side of the pixel (x,y)
- In fact, to remove seams we shift (some) pixels to the left
- At the end, we remove the extra right-side part of the image to have a clean look reduced in size image (for better understanding, extra sight-sided pixels are shown green in “new_energy.jpeg”)

Horizontal Vs. Vertical

- At first glance this code only for horizontal resizing but it works for vertical resizing too.
- To implement vertical resizing, simply rotate the image 90 degrees to the left, apply changes then rotate the resulting image 90 degrees to the right (or 270 degrees to the left again as it’s implemented in this code) using rotate( ) function.
- All the other images must also rotate.


Horizontal
Or vertical
H or V
V
H
Rotate 90
edit
edit
Rotate 270

How to run an example

An step by step example

Make sure you have Numpy and PIL modules installed

After executing the code, the first input is required like this:

```
(base) amin@amin-HP-ProBook-4530s:~/algorithm_project$ python3 algo.py
Enter picture path including it's name:
```

Enter the desired image path:

```
(base) amin@amin-HP-ProBook-4530s:~/algorithm_project$ python3 algo.py
Enter picture path including it's name: /home/amin/Pictures/ballon.png
```

The next input is required like this:

```
(base) amin@amin-HP-ProBook-4530s:~/algorithm_project$ python3 algo.py
Enter picture path including it's name: /home/amin/Pictures/ballon.png
For Horizontal reduction enter 0 and for vertical reduction enter 1:
```

After writing 0 or 1, next input is like this:

enter the number of pixels you want to reduce your image(in width or height which you selected above)

```
(base) amin@amin-HP-ProBook-4530s:~/algorithm_project$ python3 algo.py
Enter picture path including it's name: /home/amin/Pictures/ballon.png
For Horizontal reduction enter 0 and for vertical reduction enter 1: 0
Reduce amount: 50
```

After some wait messages executing is finished:

```
(base) amin@amin-HP-ProBook-4530s:~/algorithm_project$ python3 algo.py
Enter picture path including it's name: /home/amin/Pictures/ballon.png
For Horizontal reduction enter 0 and for vertical reduction enter 1: 0
Reduce amount: 50
energy-map image created, please wait...
seams found, please wait...
Finished
```

Resulted images are created in the algo.py directory as default in .PNG format