

**CONTENT-AWARE IMAGE RESIZING WITH SEAM CARVING**

Prepare by: Truong Hoang Son

Student ID: HE160073

Advisor: Mr. Phan Dang Cau

Ha Noi, 2022

1. **Introduction**
   * Seam carving is an algorithm that resizes the image by removing the unimportant pixels while keeping the important pixels(Figure 1).



a. Original image b. Seam Carving

Figure 1. Seam Carving algorithm

* Compare to other algorithm like Cropping and Scaling, Seam Carving has a lot of advantages because image does not lose too much information and the quality of image is much better, without distortion(Figure 2,3).



Figure 2. Scaling Figure 3. Cropping

**2. Seam Carving algorithm**

Seam Carving algorithm include 4 main step:

* Compute energy image
* Find seam line with minimum energy
* Delete minimum seam line
* Repeat till the desired number of rows/columns are deleted

**2.1 Compute energy image**

* The energy of an image represents the relationship of a pixel to its neighbors. The larger the diﬀerence between a pixel and a neighboring pixel, the greater the energy.
* Figure 4 show a sample 3 pixel with high energy(left) and

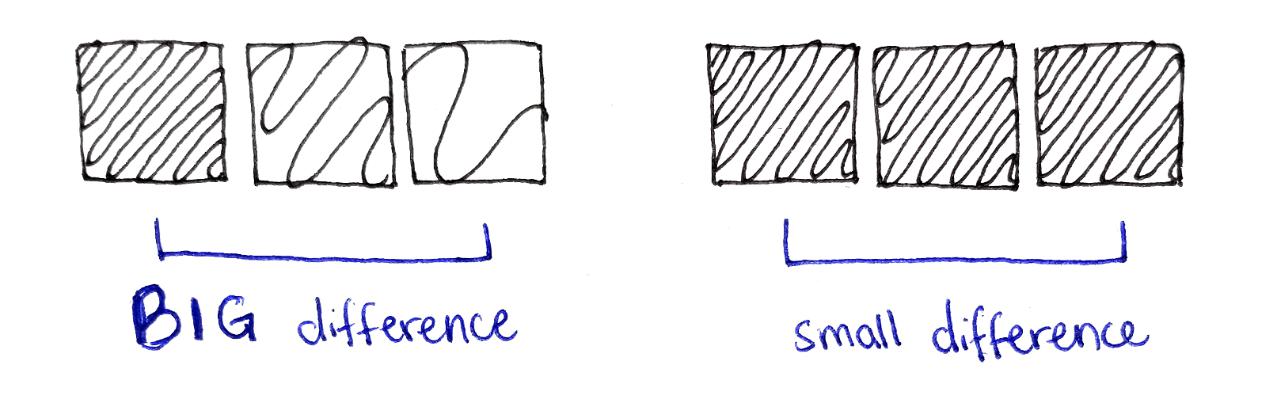


Figure 4. High energy(left) vs Low energy(right)

* In my project, we using 2 sober filter to compute energy. This is a convolutional kernel that is run over the image on 3 channel and then sum all 3 channel to obtain 2 gray scale image that is a derivative of image in the direction of Ox and Oy.

∂*f*

* + In Ox direction:∇*fx* = ∂*x*

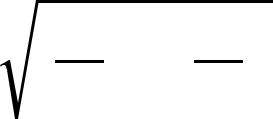
∂*f*

* + In Oy direction:∇*fy* = ∂*y*



* After obtain 2 gray scale image by applied sober filter, we use norm 2 as following equation:

*Energy* = ( ∂∂*fxx* )2 + ( ∂∂*fyy* )2



on 2 image to obtain a single gray scale image that is energy of original image *f* (Figure 5)



Figure 5. Energy of the image

**2.2. Minimum seam**

**2.2.1. Seam line**

* Vertical seam is a line from top to bottom of image in which each seam element (i, j) can only reach to element (i+1, j-1), (i+1, j), (i+1, j+1)
* Similarly, horizontal seam is a line from left to right of image in which element (i, j) can only reach to element (i-1, j+1), (i, j+1), (i+1, j+1)

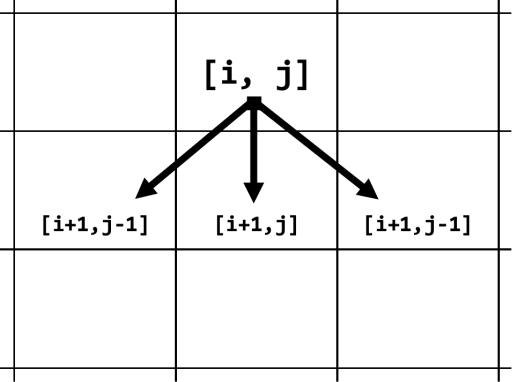


Figure 6. Vertical seam’s element rule

==> Our goal is compute minimum vertical seam to delete column of image and horizontal seam for delete row of image

**2.2.2. Minimum seam**

* Finding minimum seam line is the most important step in seam carving algorithm. The simplest approach to this problem is brute force. However, using brute force in this problem is not feasible because its time complexity is exponential.
* Another approach that can be used is Greedy algorithm. Similar to brute force, we start at top, but in each step, instead of check all 3 pixel, choose

pixel with smallest energy.

==> Problem: get stuck when the image has a high energy region(Figure 7)

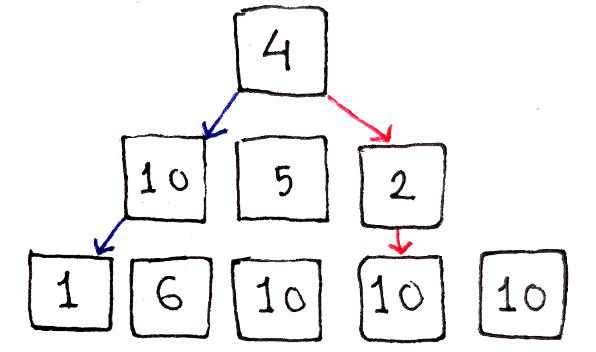


Figure 7. Greedy bottleneck

* Dynamic programming is now a very good method for this problem. To be able to use dynamic programming eﬀectively, we define:
  + *energy*[*i*, *j* ]: Energy of pixel at [*i*, *j* ]
  + *track*[*i*, *j* ]: minimum sum of energy of seam line from top image to

pixel at [*i*, *j* ]

==> Each element [*i*, *j* ] can be calculated based on 3 previous element:

*track*[*i*, *j* ] = min(*track*[*i* − 1,*j* − 1], *track*[*i* − 1,*j* ], *trach*[*i* − 1,*j* + 1])

* *energy*[*i*, *j* ]

Because each element [*i*, *j* ] is computed through previous row *i* − 1.

Therefore, we need to define base case when *i* = 0:

*track*[0,*j* ] = *energy*[0,*j* ]

Figure 8 below shows how dynamic programming works in this problem:

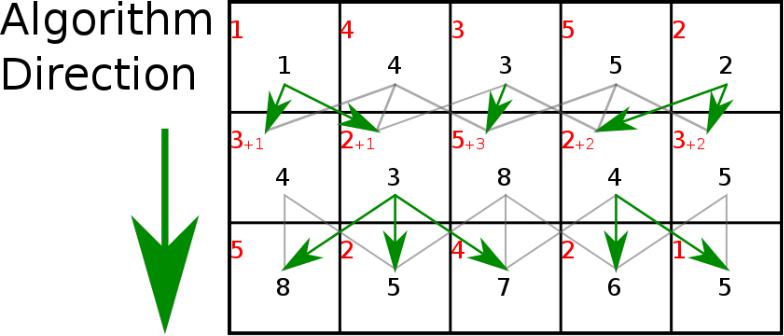


Figure 8. Dynamic programming find minimum seam

* With *track* matrix, we can find sum energy of seam from top to current element [*i*, *j* ]. Therefore, to calculate the path leading to that sum of seam, we lead to define another matrix:
  + *backtrack*[*i*, *j* ] : store index of column with minimum sum-energy in previous row

**2.3 Delete minimum seam**

==> Removed minimum seam by copying all the pixels that is not in the minimum seam to a new image with reduced size:

*image*(*H* × *W* ) ⇒ *new image*[*H* × (*W* − 1)]

**3. References**

<https://en.wikipedia.org/wiki/Seam_carving>

<https://jeremykun.com/2013/03/04/seam-carving-for-content-aware-image-scaling/>