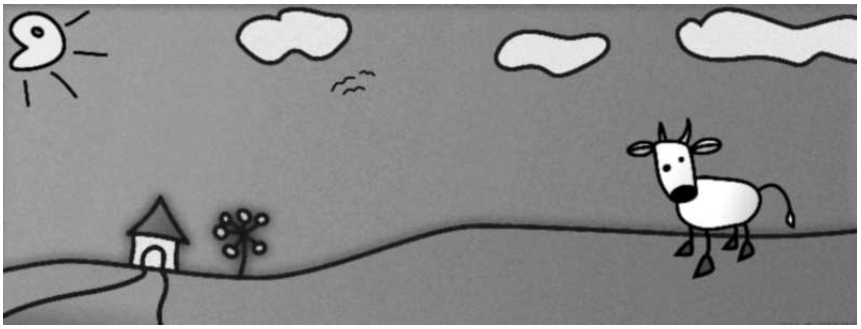
CPSC 535
Assignment 3b: Seam Carving for Content-Aware Image Resizing

The goal of this assignment is to implement the Seam Carving algorithm [1]. You are encouraged to review the original paper (and accompanying video) to help you with this assignment. Also, this assignment uses the derivative of an image (as learned in class) and dynamic programming. You are encouraged to review these before starting this assignment.

1 Seam Carving

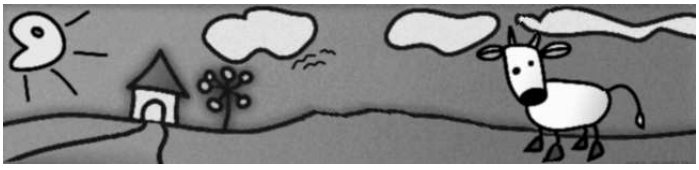
When an image is scaled, there are two main problems: one, the image must be scaled proportionally on both axis to maintain aspect ratio, and two, main objects are treated as being equal to the background. Seam carving helps to alleviate these problems because it is content-aware, allowing for unimportant information to be scaled more aggressively than the important bits. This is illustrated in Figure 1. As you can see, the seam-carving example maintained the aspect ratio of the features of the image while shrinking primarily the plain background.

As part of this algorithm you have to decide on an image feature to use to represent the level of information (or



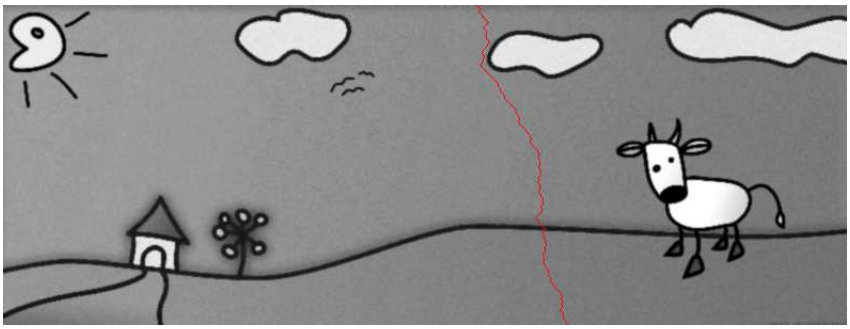
(a) Original image

(b) Remove 150px from both axis by scale



(c) Remove 150px from both axis by seam removal

Figure 1: Seam Carving Example to remove 150 pixels from each dimension



(a) A vertical seam

(b) Seam traversal constraint. The vertical seam can only follow the arrows from the gray pixel

Figure 2: Description of a seam

energy). While we recommend starting with the image gradient, there is no easy (or correct) answer here. Which feature you choose will determine which areas of the image are removed first. With the image gradient, flat and unchanging areas will have a lower energy value than busy, quickly-changing areas. The energy of a seam is simply the sum of the energies at each pixel in the seam.

A seam is defined to be a line from one end of the image to another. This seam does not need to be straight, but it does need to hit each row once and only once to maintain the composition of the rest of the image. Generally, vertical and horizontal seams are removed separately, and seams are limited to moving straight, or one pixel to either direction. This is illustrated in Figure 2. The base idea of this algorithm, then, is to successively find seams with the least amount of energy and remove them.

(HINT: get your algorithm working for vertical seams first. Then, apply the same function to the transpose of the resulting image.)

2 Algorithm

The algorithm for determining the energy required to reach a given pixel is easily defined recursively. Given an image of size $x, y = m, n$, and $E_{x,y}$ is the energy required to pass through, and $ME_{x,y}$ is the minimum energy required to reach pixel x, y from the image edge, then in the vertical case:

$$ME_{x,y} = \min(ME_{x-1,y-1}, ME_{x,y-1}, ME_{x+1,y-1}) + E_{x,y}$$

$$ME_{-1,y} = ME_{m+1,y} = inf \text{ (horizontal boundary condition)}$$

$$ME_{x,-1} = 0 \text{ (vertical boundary condition)}$$

That is, of the three pixels in the previous row that can be used to reach pixel x, y, take the one that requires the least energy to reach, then add the energy required to traverse the current pixel, and that gives you the minimum energy required to reach and traverse a given pixel. Further, it takes no energy to reach the first row of pixels, and the seam cannot pass off the horizontal edge.

I highly recommend that you approach this algorithm using dynamic programming methods. Using a recursion programming approach will be painfully slow and inefficient. Here are some hints to start you and keep you on the right track:

1. start above the top of the image, with energy values all at zero, and work downwards.
2. when calculating the minimum energy required to reach each pixel, do one row at a time in a single operation and only refer to the immediately-previous row. You can avoid for loops here. Use a for loop to iterate through the rows.
3. at each row, store the source indice that allows a given pixel to be reached with minimum energy. This allows you to track backwards when you find the end-point with the lowest-energy path.

3 Octave

Having a strong understanding of octave can really make this assignment a great deal easier. Here are a few things to consider.

- look-up linear indices and the functions `sub2ind`, `ind2sub`, and `reshape`. For example, what does the following code return? `A = [1 2 ; 3 4 ; 5 6]; A(5)`. This is useful when you carve out the seams.
- what happens when you delete an octave item using the `variable(x) = []` syntax? What does the following code do? `B = [1 2; 3 4]; B(3) = []`

4 Assignment

Implement the seam carving algorithm and remove 200px from both the X and the Y dimensions. You can use the image included or use your own. Show your results.

5 Questions

1. Discuss the implications of the energy function. For example, what happens if you maximize rather than minimize energy? What are some other functions you could think of?
2. Reflect on the following questions
   (a) How could this algorithm be used to *grow* an image rather than shrink it?
   (b) What happens if we choose non-axis-aligned seams (i.e., rotate the image first or follow a different seam path structure)?
   (c) When would seam carving NOT be a desireable alternative to scaling or cropping?
3. How far does it make sense to carve an image? What happens as the image gets smaller?
4. Explain any artifacts in your final resulting image. Why did these happen and what can be done about them?

Hand In

Hard copy and electronic copy (email to TA):

1. Cover sheet with your name, course number, and assignment number only.
2. Name and student ID number inside the cover sheet.
3. Writeup
4. All plots, images, and source code. Make sure you label the plots to indicate what they are.

Marking

Assignment will be graded as outlined in the course marking guidelines sheet.

Collaboration

The assignment must be done individually so everything that you hand in must be your original work, except for the code copied from a cited source or that supplied by your instructor. When someone else's code is used like this, you must acknowledge the source explicitly. Copying work that is not your own without acknowledgement is academic misconduct. Contact your instructor if you have problems or questions regarding this.

References

[1] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics, SIGGRAPH 2007*, 26(3), 2007.