CIS581: Computer Vision and Computational Photography
Project 2, Part B: Seam Carving
Due: Oct. 25, 2019 at 11:59 pm

Instructions

- Seam Carving is an **individual** project. 'Individual' means each student must hand in their **own** answers, and each student must write their **own** code in the homework. It is admissible for students to collaborate in solving problems. But to help you actually learn the material, what you write down must be your own work, not copied from any other individual. You **must** also list the names of students (maximum two) you collaborated with.
- If you prefer, you can do the entire part on your own. However, completion of the project is MANDATORY and no extra credit will be offered if you do it individually.
- You **must make two submissions** on Canvas. We recommend that you include a README.txt file in both submissions to help us execute your code correctly.
  - Place your **code**, **resulting images and videos** into a folder named "Seam_Carving". Submit this as a zip file named `<Pennkey>_seam.zip`
- Your submission folder should include the following:
  - your .m or .py scripts for the required functions.
  - .m or .py scripts for generating the seam carving video.
  - any additional .m or .py files with helper functions you code.
  - the images and videos you used
  - video files generated for seam carving.
  - Any other files required for your mini-project.
- This handout provides instructions for two versions of the code: MATLAB and Python. You are free to select **either one of them** for this project.
- Feel free to create your own functions as and when needed to modularize the code. For MATLAB, ensure that each function is in a separate file and that all files are in the same directory. For Python, add all functions in a helper.py file and import the file in all the required scripts.
- **Start early!** If you get stuck, please post your questions on Piazza or come to office hours!
- **Follow the submission guidelines and the conventions strictly! The grading scripts will break if the guidelines aren't followed.**

Seam Carving

For this project, you will be implementing image resizing utilizing scene carving. The structure of this part of the project is based heavily on the methods outlined by Avidan & Shamir in their paper:

`"Seam Carving for Content-Aware Image Resizing"`; Avidan, S. & Shamir, A.; 2007

You are strongly encouraged to read this paper prior to starting this part of the project (a link to the paper will be made available on the course wiki).

You are tasked with filling in 5 functions: `carv.py, cumMinEngVer.py, cumMinEngHor.py, rmVerSeam.py, rmHorSeam.py`

We begin by computing the energy map, $e(I) = |\frac{\partial I}{\partial x}| + |\frac{\partial I}{\partial y}|$. We have provided a function, genEdgeMap.py which computes this for you, in order to maintain consistency in the method by which the energy map is computed. Your task is to fill in the code that follows. For this project, we will be testing you strictly on shrinking an image (making it smaller), though the ideas applied towards enlarging an image are very similar to those of shrinking one.

You must create a video depicting the image resizing operation. You can pad the resized images with zeros to maintain dimensionality.

1 Computing Cumulative Minimum Energy:

You need to complete the code for the functions `cumMinEngVer.py` and `cumMinEngHor.py`, which correspond to computing the cumulative minimum energy over the vertical and horizontal seam directions, respectively. The function `cumMinEngHor.py` has the following structure:

[My, Tby] = cumMinEngHor(e)

- (INPUT) e: n $\times$ m matrix representing the energy map.
- (OUTPUT) My: n $\times$ m matrix representing the cumulative minimum energy map along horizontal direction.
- (OUTPUT) Tby: n $\times$ m matrix representing the backtrack table along horizontal direction.

You'll notice that Avidan & Shamir, as well as our class notes, define seams for an n $\times$ m image, I as follows:

- A vertical seam $s^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n$, s.t $\forall i$, $|x(i) - x(i-1)| \leq 1$, where x is a mapping $x : [1, ..., n] \rightarrow [1, ..., m]$ i.e. the corresponding column for a seam pixel at row i along an 8-connected path from the top to the bottom of the image.
  `Note` that for any vertical seam, there only exists one seam pixel per row.
- Similarly, a horizontal seam is defined as $s^y = \{s_j^y\}_{j=1}^m = \{(y(j), j)\}_{j=1}^m$, s.t $\forall j$, $|y(j) - y(j-1)| \leq 1$, where y is a mapping $y : [1, ..., m] \rightarrow [1, ..., n]$ As with the vertical seam, there exists one seam pixel per column along an 8-connected seam path.

For a vertical seam, the seam cost is defined as:

$$E(s^x) = E(I_{s^x}) = \sum_{i=1}^{n} e(\mathbf{I}(s_i))$$

For horizontal seam, the seam cost is defined as:

$$E(s^y) = E(I_{s^y}) = \sum_{j=1}^{m} e(\mathbf{I}(s_j))$$

. We seek to identify the optimal seam, s^*, that minimizes the seam cost, i.e. $s^* = \min_s E(s)$

`To quote Avidan & Shamir:`

The optimal seam can be found using dynamic programming. The first step is to traverse the image from the second row to the last row and compute the cumulative minimum energy M for all possible connected seams for each entry (i, j):

$$M_x(i,j) = e(i,j) + \min\left(M_x(i-1,j-1), M_x(i-1,j) M_x(i-1,j+1)\right)$$

$$M_y(i,j) = e(i,j) + \min\left(M_y(i-1,j-1), M_x(i,j-1), M_x(i+1,j-1)\right)$$

At the end of this process, the minimum value of the last row in M_x will indicate the end of the minimal connected vertical seam. Note that you have to record a backtrack table along the way. Hence, in the second step we backtrack from this minimum entry on M_x to find the path of the optimal seam. Also, in order to maintain consistency, if the minimum value is found at multiple indices, choose the smaller index. The definition of My for horizontal seams is similar.

2 Removing a Seam

For this task, you will fill in the two functions, `rmVerSeam.py` and `rmHorSeam.py`, which remove vertical and horizontal seams respectively. This task should be fairly simple. You should identify the pixel from M_x or M_y from which you should begin backtracking in order to identify pixels for removal (rmIdx), and remove those pixels from the input image. You will receive two inputs to each function, the corresponding cumulative minimum energy map, and the image. Utilizing these, you should output an image with one (appropriate) seam removed. The function `rmVerSeam.py` has the following structure:
[Iy, E] = rmHorSeam(I, My, Tby)

- (INPUT) I: n $\times$ m $\times$ 3 matrix representing the input image.
- (INPUT) My: n $\times$ m matrix representing the cumulative minimum energy map along the horizontal direction.
- (INPUT) Tby: n $\times$ m matrix representing the backtrack table along horizontal direction.
- (OUTPUT) Iy: (n-1) $\times$ m $\times$ 3 matrix representing the image with the row removed.
- (OUTPUT) E: the cost of seam removal.

3 Discrete Image Resizing

Now that you're able to handle finding seams of minimum energy, and seam removal, we shall now tackle resizing images when it may be required to remove more than one seam, sequentially and potentially along different directions. When resizing an image from size n $\times$ m to n' $\times$ m' (we will assume $n' < n$ and $m' < m$), the sequence of removing vertical and/or horizontal seams is important.

For this task, fill in the function `carv.py`, making use of the method described below. Utilizing recursive calls of the functions completed in the previous 2 tasks, complete this function to output the resized image.
[Ic, T] = carv(I, nr, nc)

- (INPUT) I: n $\times$ m $\times$ 3 matrix representing the input image.
- (INPUT) nr: the numbers of rows to be removed from the image.
- (INPUT) nc: the numbers of columns to be removed from the image.
- (OUTPUT) Ic: (n - nr) $\times$ (m - nc) $\times$ 3 matrix representing the carved image.
- (OUTPUT) T: (nr + 1) $\times$ (nc + 1) matrix representing the transport map.

The following method is derived directly from the works of Avidan & Shamir:

We define the search for the optimal order as an optimization of the following objective function:

$$\min_{s_x, s_y, \alpha} \sum_{i=1}^{k} E(\alpha_i s_i^x + (1-\alpha_i) s_i^y)$$

where $k = r_t + c_t$, $r_t = (m - m'), c_t = (n - n')$. α_i is a parameter that determines where a horizontal or vertical seam is removed at step i. $\alpha_i \in \{0, 1\}$, $\sum_{i=1}^{k} \alpha_i = r_t$, $\sum_{i=1}^{k} (1 - \alpha_i) = c_t$

We find the optimal order using a transport map T that specifies, for each desired target image size $n' \times m'$, the cost of the optimal sequence of horizontal and vertical seam removal operations. That is, entry

$T(r,c)$ holds the minimal cost needed to obtain an image of size $(n-r)\times(m-c)$. We compute T using dynamic programming. Starting at $T(0,0)=0$ we fill each entry (r,c) choosing the best of two options - either removing a horizontal seam from an image of size $(n-r)\times(m-c+1)$ or removing a vertical seam from an image of size $(n-r+1)\times(m-c)$:

$$T(r,c)=\min(T(r-1,c)+E(s^x(I_{(n-r+1)\times(m-c)}),T(r,c-1)+E(s^y(I_{(n-r)\times(m-c+1)}))$$

where $I_{(n-r)\times(m-c)}$ denotes an image of size $(n-r)\times(m-c)$, $E(s^x(I))$ and $E(s^y(I))$ are the costs of the respective seam removal operation. We can store a simple binary map which indicates which of the two options were chosen in each step of the dynamic programming.

Choosing a left neighbor corresponds to a vertical seam removal while choosing the top neighbor corresponds to a horizontal seam removal. Given a target size $n'\times m'$ where $n'=n-r_t$ and $m'=m-c_t$, we backtrack from $T(r_t;c_t)$ to $T(0,0)$ and apply the corresponding removal operations.

Also, for the sake of consistency,

$$T(r,c)=T(r-1,c)+E(s^x(I_{(n-r+1)\times(m-c)}))$$

$$T(r-1,c)+E(s^x(I_{(n-r+1)\times(m-c)}))=T(r,c-1)+E(s^y(I_{(n-r)\times(m-c+1)}))$$

Further notes:

In addition, in order to calculate $E(s^x(I))$ and $E(s^y(I))$, you have to record a table for $I_{(n-r)\times(m-c)}$. More specifically, TI is the trace table and should be of size $(n-r_t+1)\times(m-c_t+1)$. TI1,1 is the source image, and TIr + 1, c + 1 is the image $I_{(n-r)\times(m-c)}$, which is the source image removed r rows and c columns.