Computational Photography

Assignment 4 - Retargeting
Summer 2023

Introduction

For this assignment we are going to implement some image retargeting approaches, allowing us to change the size and aspect ratio of an image. Techniques will include sub-sampling and seam carving.

Grading

Theory Questions	35pts
Image Resizing	15pts
Energy Matrix	10pts
Optimal Seam Discovery	20pts
Seam Removal	20pts
TOTAL	100pts

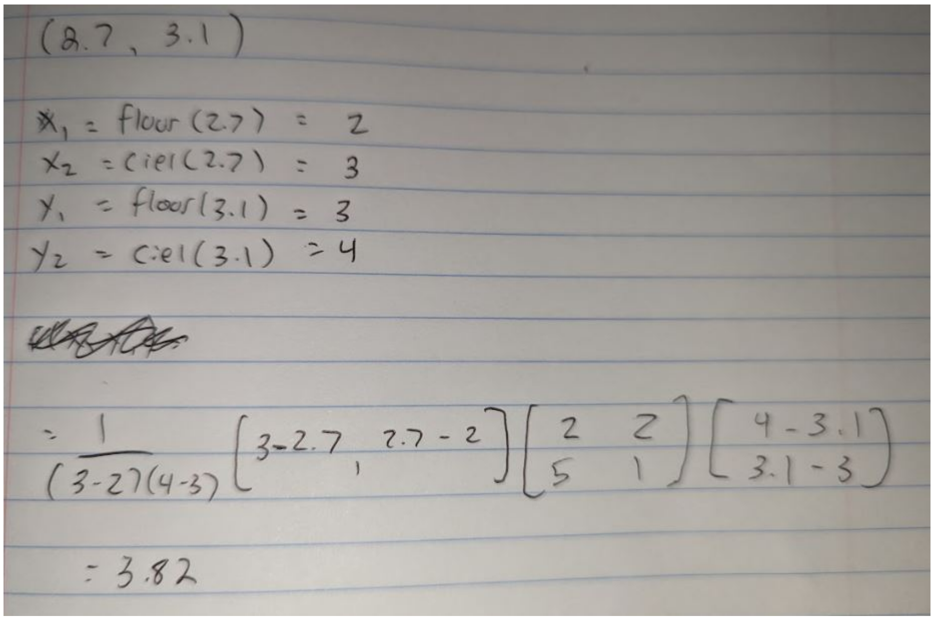
Table 1: Grading Rubric

1 (35pts) Theory Questions

1. Given the grayscale image, I, below, if the top left location in our image is $(x = 1, y = 1)$, what would be value of a pixel at location $x = 2.7, y = 3.1$, if we were to:

   (a) Use the nearest neighbor in I. (3pts)
   The closest value to (2.7, 3.1) would be (3, 3), and that matches to value of 5 in I

   (b) Perform bi-linear interpolation using I. Show your supporting work. (7pts)



Formula from: https://en.wikipedia.org/wiki/Bilinear_interpolation

$$I = \begin{bmatrix} 2 & 3 & 4 & 5 & 1 \\ 1 & 0 & 2 & 2 & 1 \\ 4 & 3 & 5 & 1 & 2 \\ 4 & 4 & 4 & 4 & 6 \\ 4 & 5 & 2 & 0 & 2 \\ 2 & 3 & 3 & 0 & 3 \end{bmatrix}$$

2. If the matrix below is the gradient magnitude image

$$G = \begin{bmatrix} 2 & 3 & 4 & 5 & 1 \\ 1 & 0 & 2 & 2 & 1 \\ 4 & 3 & 5 & 1 & 2 \\ 4 & 4 & 4 & 4 & 6 \\ 4 & 5 & 2 & 0 & 2 \\ 2 & 3 & 3 & 0 & 3 \end{bmatrix}$$

   (a) Construct the cost matrix if we assume vertical seams (10pts).
   Upon Using Cost Matrix Formula:

$$C(i, j) = G(i, j) + min(C(i-1, j-1), C(i-1, j), C(i-1, J+1))$$

We are left with Cost Matrix:

$$CostMatrix = \begin{bmatrix} 2 & 3 & 4 & 5 & 1 \\ 3 & 2 & 5 & 3 & 2 \\ 6 & 5 & 7 & 3 & 4 \\ 9 & 9 & 7 & 7 & 9 \\ 14 & 11 & 9 & 7 & 9 \\ 13 & 12 & 10 & 7 & 10 \end{bmatrix}$$

(b) What is the optimal seam (2pts)?
After Tracing for the minimum of each row, starting from the last 6th row, we can find the optimal seam to be:

$$Optimal = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 7 \\ 7 \\ 7 \end{bmatrix}$$

3. In lecture we discussed applying Poisson Blending to a simple 1D case. Now let's extend this for the 2D case (after all, we're dealing with images!)!.

   For simplicity, let's assume a 1×1 square mask. Figure 1 shows two images, the one we're copying *to* and the one that we're copying *from*. The masked area is indicated in green.

   `masks.png`

   Figure 1: Images for Theory Question #3

   (a) (5pts) First let's establish the cost function J for this 2D case. We'll once again use the squared error. To avoid confusion between the unknowns x and the direction x, we'll use t for the values in our *target* image (as opposed to x in the 1D example).

   The cost with regards to pixel (i, j) and the *x-gradient* is as follows. It's basically the same as the 1D case in the lecture slides:

$$J^x_{i,j} = ((t_{i,j} - t_{i-1,j}) - (s_{i,j} - s_{i-1,j}))^2$$

   The cost for this same pixel, with regards to the *y-gradient* is:

$$J^y_{i,j} = ((t_{i,j} - t_{i,j-1}) - (s_{i,j} - s_{i,j-1}))^2$$

   Write out the terms of $J^x_{i,j}$ and $J^y_{i,j}$ that include at least one unknown in the target image.

$$J^x_{i,j} = ((t_{(2,2)} - t_{(1,2)}) - (s_{(2,2)} - s_{(1,2)}))^2$$
$$J^x_{i,j} = ((2 - 5) - (3 - 0))^2 = \mathbf{36}$$
$$J^y_{i,j} = ((t_{(2,2)} - t_{(2,1)}) - (s_{(2,2)} - s_{(2,1)}))^2$$
$$J^y_{i,j} = ((2 - 1) - (3 - 4))^2 = \mathbf{4}$$

   (b) (5pts) Using your answer from the previous part, what is the coeficient matrix A, the unknown vector t and the known vector b so that we can write J as $J = (At-b)^T(At-b)$? You should have a single matrix A and a single vector b.

$$A = \begin{bmatrix} 1 \end{bmatrix}$$
$$t = \begin{bmatrix} t_{(2,2)} \end{bmatrix} \; B = \begin{bmatrix} 4 \\ 36 \end{bmatrix}$$

   (c) (3pts) Finally, what are the values for the new pixel within the target mask? For partial credit, make sure it's clear where you got your values from.

$$X = A^{-1}b$$

2 (15 points) Image Resizing

First grab two images of interest to you so that we can multiple example I/O of each approach. Next allow a user (either via prompt, parameter passing, or changing variable values) to provide a desired height and width.

We'll refer to the original height and width as h and w, respectively, and the new height and width as h' and w', respectively. Do this all in color! You may **not** use any Matlab functions (for instance, *imresize* or *interp2*) to do these changes for you.

Implement the following resizing techniques:

- Do nearest neighbor sampling. Go through each location in your new target image, (x', y'), and assign it the value of the nearest pixel, whose location is $(x, y) = round(x'\frac{w}{w'}, y'\frac{h}{h'})$.
- Do bi-linear interpolation. Go through each location in your new target image, (x', y'), and compute the *ideal* floating point location as $(x, y) = (x'\frac{w}{w'}, y'\frac{h}{h'})$. Compute the values for location (x', y') by linearly interpolating the values of the four pixels nearest (x, y).

For your report provide sample I/O for both your test images, using both of these techniques, for two different values of (w', h') (8 total images).

2.1 Original Images

2.2 Nearest Neighbors

Nearest Neighbor

Nearest Neighbor

2.3 Bi-linear interpolation

3 (10 points) Energy Function

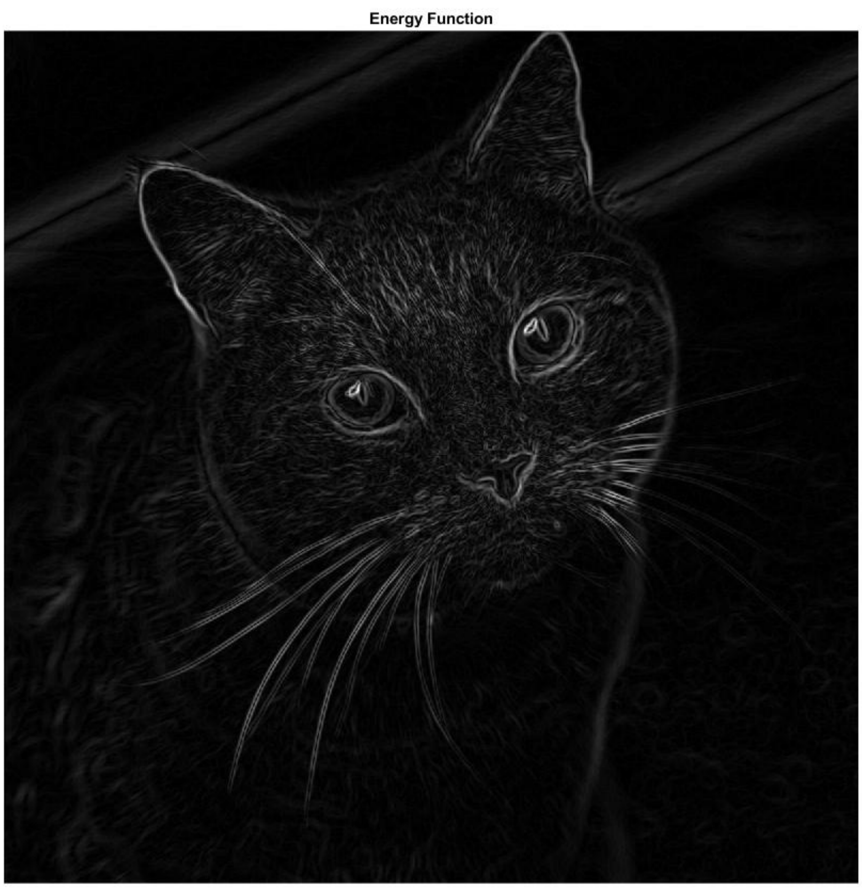
We're also going to want to explore seam carving. The first part of that requires computing the energy functions of your images.

For each of your images, compute its energy and visualize this as an image.

Some implementation details:

- Do this in grayscale
- First smooth your grayscale image using a Guassian filter prior to getting the gradients (or do this in one step). Choose parameters that make sense for you (and report them!).

Since you already demonstrated in prior assignments your to implement RGB to Gray, Gaussian and Gradient kernels and convolution, for this assignment **may** use Matlab functions like $conv2, rgb2gray$. In addition, for this and subsequent parts, you **may** use Matlab's *imgradxy* to compute the gradients, since we've already implemented it ourselves in prior assignments.



Energy Function

4 (20 points) Optimal Seam

Now that you have your energy images, we must find the optimal seam in it.

Using the technique discussed in class, for each of your images,

- Use its energy image to compute a seam matrix.
- Find the optimal seam in this seam matrix via backtracing.
- Superimpose on your color image the optimal seam in red.

Additional Details:

- We will do vertical seam carving, starting at the top of the image.
- You'll likely have to think about how to handle the edge cases.

Figure **??** shows an example output.

5 (20 points) Seam Carving

Finally, let's use seam carving to reduce the aspect ratio. Since we found the optimal *vertical* seam in the previous part, we'll just reduce the width, from it's original width down to one pixel wide.

For each of your images, create a video showing the seam removal process. Each frame of the video should depict the current color image with the current optimal seam superimposed (like in the previous part).

Note:

- You may need to "pad" your frames so that they all have the same size in order to render as a movie.
- To create a movie in Matlab use the *VideoWriter* class. In addition, to keep the movies relatively small in file size, use the *MPEG-4* profile for your VideoWriter object.

Videos Included in ZIP file

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF writeup that includes:
   (a) Your answer to the theory question(s).
   (b) Your two original images.
   (c) Eight resized images for Part 2
   (d) Your two energy function images for Part 3.
   (e) Your two optimal seam images for Part 4.
2. A README text file (not Word or PDF) that explains
   - Features of your program
   - Name of your entry-point script
   - Any useful instructions to run your script.
3. Your source files
4. The chosen images that you are processing.
5. The videos generated for Part 5.