CMSC 435/634: Introduction to Computer Graphics

Assignment 6
Seam Carving
Due Dec 11, 2018 @ 11:59 PM

The Assignment

One of the papers at SIGGRAPH that made news when it first came out was *Seam Carving for Content-Aware Image Resizing* by Shai Avidan and Ariel Shamir. In this assignment, you'll be implementing the basic algorithm presented therein. To wit, you'll be designing a program which can shrink an image (horizontally and/or vertically) to a given dimension. Your program should support the following usage

```
seamcarving input_image output_image output_width output_height
```

An executable of my program is at:

```
~adamb/public/SeamCarving/seamcarving
```

Details

For a brief overview of the algorithm and some inspiring results, check out the video. Your algorithm will shrink images as follows:

1. Determine the 'importance' each pixel has using an *energy function*.
2. Until image has shrunk to the desired dimension:
    1. Find the lowest-importance *seam* in the image.
    2. Remove it.

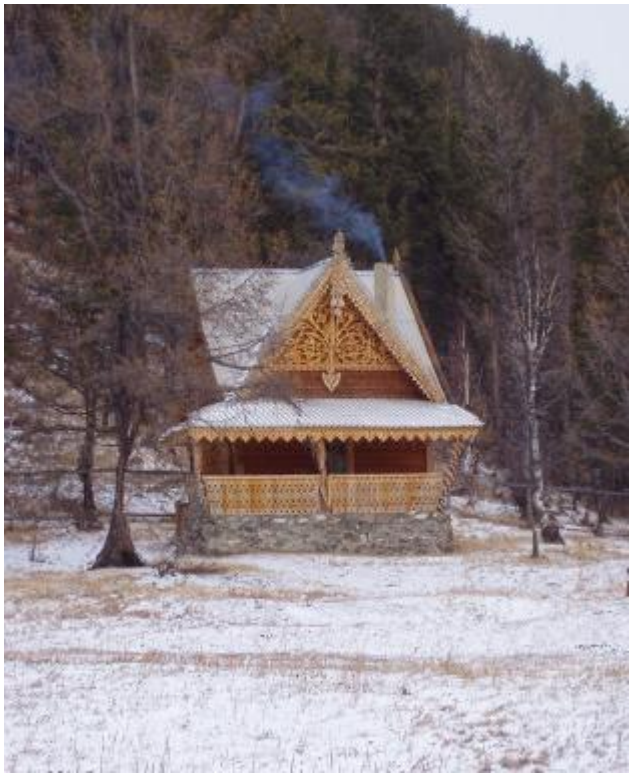
Feel free to use any *energy function* in the paper (or cook up something on your own). Defining a good energy function seems to be the key to achieving good results.

Seam examples. *Black:* horizontal seam. *Gray:* vertical seam. *Green:* not a seam -- contains more than one pixel from some columns.

A horizontal *seam* is a connected path from one side of the image to the other that chooses exactly one pixel from each column. (A vertical seam is the same, but from top to bottom and with rows.) Finding the lowest-importance seam is a simple dynamic programming exercise. If you find yourself spending large amounts of time on it, please ask the course staff for help.

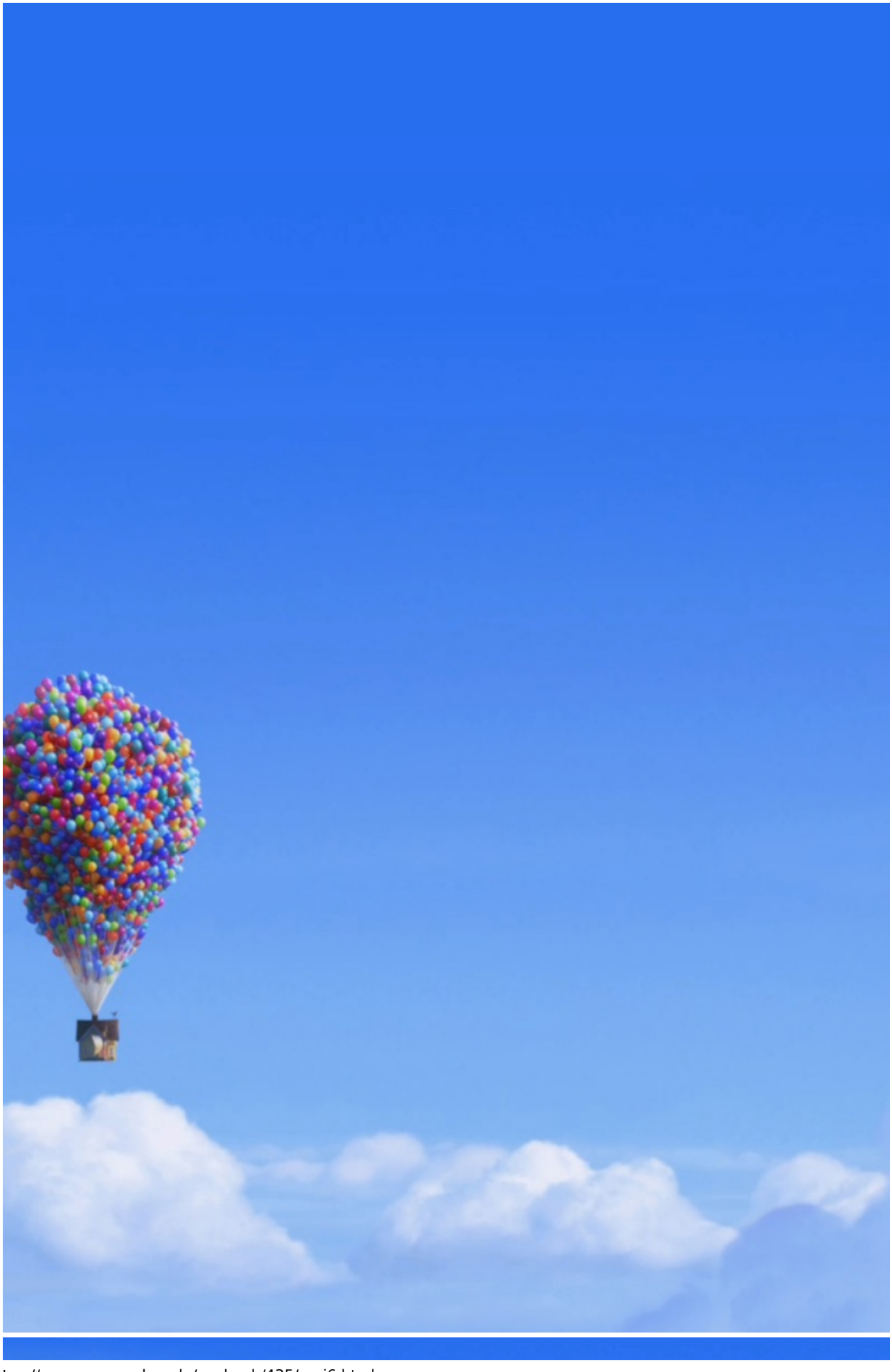
Your final code should be able to, given an input image, produce an output image shrunk to the specified number of pixels. A simple way to achieve this goal is to write your code to shrink only in one dimension and then create a wrapper function that transposes the image to shrink in the other dimension. I also *highly* recommend outputing your an image of your energy function mapped to greyscale values. I set each pixel to pow(energy(i,j)/max_energy, 1.0/3.0). See below:

Example: Horizontal Carving



Example: Vertical Carving

Notice how the content in the image is preserved, and the large, boring, area of blue sky is removed.



Example: Carving Artifacts

As you push the algorithm to its limits, artifacts -- such as shape distortions and odd cubist angles -- will begin to crop up. A lot of time on this project will be spent tweaking your code to mitigate these effects, as well as searching for pictures which don't exhibit them. Generally, highly-structured scenes and human faces will suffer, while scenes with textured objects on simple backgrounds will work better.

Extra Credit

- Implement the Scharr gradient operator. (15 pts)
- Compute the energy for the entire image once, then update it around seams as they are removed. (15 pts)

- Create a web page showcasing your results, include at least one image of yourself, one good case, and one failure case. (15 pts)
- Add seams, instead of removing pixels in the seam add pixels to the left (or right) of the seam, use the average of the two pixels that will be adjacent to the added pixel. (15 pts)

What to turn in

Turn in this assignment electronically by pushing your source code to your `proj6` GIT directory by 11:59 PM on the day of the deadline. We will be looking for multiple checkins documenting your development process.

As always, double check that you have submitted **everything** we need to build and run your submission, but **no** generated files (.o's, executables, or images). Be sure to include a Makefile that will build your project when we run '`make`', and a `readme.txt` file telling us about your assignment. Do not forget to tell us what (if any) help did you receive from books, web sites or people other than the instructor and TA.

(This page: www.csee.umbc.edu/~adamb/435/proj6.html)