Emily Eljamal
Maria Herne

Github Link: https://github.com/EmilyEljamal/Seam-Carving-FPSE

Project Overview

The purpose of this project is to develop an image transformation tool that uses seam carving to resize images while preserving critical visual elements. Regular resizing tools tend to distort or lose important parts of an image whereas seam carving uses vertical seams (paths of low importance pixels) that are either removed or inserted to maintain the integrity of key features. This makes it ideal for resizing images with complex details where certain areas should remain untouched despite a change in dimensions.

We will develop a seam carving algorithm in OCaml that will apply functional programming concepts. OCaml's Core library will support functional abstractions and efficient data manipulation while ImageMagick will be used to handle the storage of images and handle the output of the final image product.

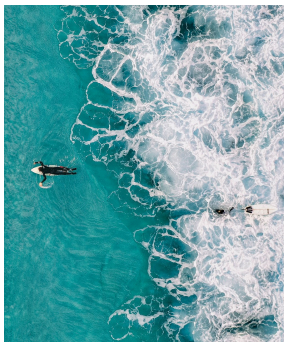
The project will also include a dynamic visual component, where each intermediate step of the resizing process is saved as a frame to create a GIF or video of the resizing progression. We will be using ImageMagick to complete this. This feature will provide users with an animated view of how the image adjusts as seams are removed or added, highlighting the algorithm's ability to maintain the visual integrity of the image's important features.

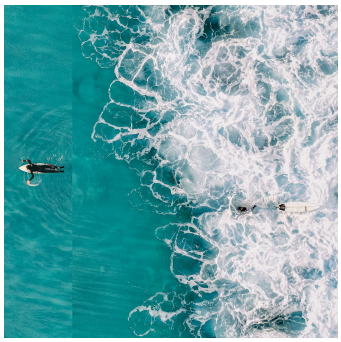
Overall, this project serves as a deep dive into algorithmic image processing, emphasizing efficient memory management, data persistence, and functional problem-solving in OCaml. The result will be a tool that enables image resizing and serves as a foundational piece for more advanced applications in adaptive media and computer vision.

Example: This is the original image

This is the vertical seam to be removed

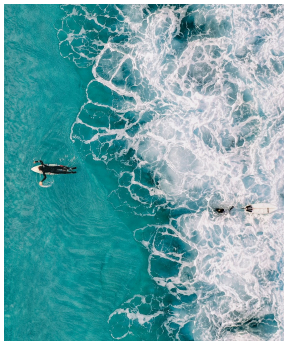
This is the image after removing 1024 pixels. The main focus, the surfer, is untouched and the waves remain. The resized image without seam carving would look like the image to the right.





Mock Use

$ ./seam_carve.exe ~/Desktop/Screenshots/image.png 50 ~/Desktop/output



Outputted Image

GIF

The user runs our program ./seam_carve.exe with the image path, the desired size they want their image then the path where they want their image and gif to be downloaded.

Algorithm Description

The algorithm we will use to implement Seam Carving is derived from the MIT lesson from Fall 2020
Seam Carving | Week 2, lecture 7 | 18.S191 MIT Fall 2020. The main sequence of the algorithm is

1. Create an Energy Grid of values 0-1 (black-white) to assign importance to pixels based on edge detection

a. Time Complexity: O(N)

To compute the energy of a single pixel, we look at the pixels to the left and right of that pixel. We find the squared component-wise distance between them, that is compute the squared difference between the red components, the squared difference between the green components and the squared difference between blue components, then add them up. We do the same for the pixels above and below the center pixel. Finally, we add up the horizontal and vertical distances.

$$|\Delta x|^2 = (\Delta r_x)^2 + (\Delta g_x)^2 + (\Delta b_x)^2$$
$$|\Delta y|^2 = (\Delta r_y)^2 + (\Delta g_y)^2 + (\Delta b_y)^2$$

$$e(x, y) = |\Delta x|^2 + |\Delta y|^2$$

2. Calculate the Minimal Energy to Bottom Grid using dynamic programming
    a. **Time Complexity: O(3N) = O(N), where N is number of pixels**
    b. The goal is to build from the bottom row up the minimal total value of a cell to the bottom of the grid.
        i. 1st, copy the bottom row
        ii. 2nd, traverse through each cell, bottom to top, left to right
        iii. 3rd, for each cell, check adding cell value to each the diagonal bottom L neighbor, bottom, and diagonal bottom R neighbor and save the lowest value of the 3 and the corresponding direction (-1,0,1)
3. Using the prev grid, calculate from top to bottom, the *path of least importance or the seam* aka the vertical path of pixels that will be removed

List of Libraries

- Core
- Sys_unix
- Stdio
- ImageMagick (Demo included)

Implementation Plan

11/22: Types + Image Life Cycle

- ☐ Finalize Types
- ☐ Image_Process.ml (not including calculate_energy_map)
- ☐ main.ml - command line interface

- [ ] Test loading image + trivial pixel removal + output

11/29: First Algorithm

- [ ] Calculate_energy_map + visual

12/6: Second Algorithm

- [ ] Seam_identification.ml + test visual output
- [ ] Test creating GIF from pixel removal snapshots

12/13:

- [ ] Testing

Complexity Notes:

https://pages.cs.wisc.edu/~moayad/cs766/index.html
If the project scope is small, then we would add a object removal/protection feature to the application as mentioned in the above link. This means that similar to photoshop's object removal from a photo that recreates a background, a complex application of seam carving should accomplish this. We would also build then a front-end website with a drag-drop feature to load images, a slider to realtime-condense or expand images sizes using seam carving.