Project 2
Content-Aware Seam Carving

Andrew Parmar
CS6475 Fall 2022
aparmar32@gatech.edu

I. Discussion of Algorithm

A. Backwards Energy Algorithm

My implementation of the Backward Energy algorithm for seam carving comes from [1]. The central idea in this algorithm is to use an energy function to select pixels of the lowest energy for removal. Selecting the lowest energy pixels produces the least noticeable change when removed from the image. Furthermore, selecting a seam such that pixels are 8-connected and constrained to only a single pixel per row produces results with the least visual artifacts.

My implementation starts with detecting gradients in the image in both the x and y directions. As shown in Listing-1, this is achieved by applying a 2D convolution using an asymmetric kernel. This is repeated for the y direction as well (the kernel is rotated 90 degrees clockwise). The values in all three channels are summed and the absolute value of the x and y gradient matrices are added together to produce a single 2D array of the same height and width as the source image which is our energy map.

```
kernel_x = np.array([[0, 0, 0], [1, 0, -1], [0, 0,
    0]], dtype=np.float64)
img_x_grad = cv2.filter2D(src=image, ddepth=-1,
    kernel=kernel_x)
...
np.sum(np.abs(img_x_grad), axis=2) + np.sum(np.abs(
    img_y_grad), axis=2)
```

Listing 1: Calculating energy map

Cells of the energy map with the lowest values are areas with the lowest energy. There was little change between pixels in those areas, so they are potential candidates for removal. Given this energy map we want to select a seam, i.e a sequence of pixels that both remove the least amount of energy possible from the image, while also being 8-connected from row-to-row. Using Dynamic-Programming (DP) is an effective technique to solve for this lowest energy cost seam. Using Listing-2, the DP matrix is calculated as a cumulative sum of lowest-energy pixels from subsequent rows. Starting at the lowest values cell in the last row, of this DP matrix, M,

```
for i in range(1, h):
    for j in range(w):
        M[i,j] = min(
            M[i-1, max(0, j-1)],
            M[i-1, j],
            M[i-1, min(w-1, j+1)]
        ) + energy_map[i,j]
```

Listing 2: Cumulative energy map: backward

Starting at the lowest value cell on the last row in the DP matrix I backtrack in an 8-connected constraint selecting the lowest cost route. By collecting the (row, col) values in a list I get a seam, one pixel per row, that defines the seam to be removed in the current iteration.

B. Forwards Energy Algorithm

My implementation of the Forward Energy algorithm for seam carving comes from [2]. While Backward Energy focuses on the removal of seams with the lowest energy, it ignores the fact that new boundaries are created between previously non-adjacent pixels, that could potentially have higher energy, thereby increasing the energy in the image. Forward Energy address this by looking at the seam that would be produced at each pixel after it is removed, and only picking those seams that insert the least energy after removal. This implementation is also based on DP.

My implementation starts with a border-wrap padded image that is then converted to grayscale. Three cost matrices are calculated, `c_u, c_l, c_r` that represent the cost at each pixel of removing the current pixel plus one of three options the upper pixel, the left upper left, and the upper right pixel respectively.

Then for each cell of the DP matrix, we calculate the value using

```
a = m[i - 1, j - 1] + c_l[i - 1, j - 1]
b = m[i - 1, j] + c_u[i - 1, j - 1]
c = m[i - 1, j + 1] + c_r[i - 1, j - 1]
m[i, j] = min(a, b, c)
```

Listing 3: Calculating cumulative energy map: forward

Once this DP matrix M is filled out, the same backtracking mechanism from the backward algorithm is used to generate the seam to remove.

II. Comparative Metrics

A. Difference Images

My implementation of the difference image does a subtraction of the two images and sets values in the green channel for positive values and the red channel for negative values. To allow for negative values without integer overflowing during computation, I had to convert the input images from uint8 to float64. After performing the subtraction and taking the absolute values, I converted the result back to uint8 before returning.

```
diff = result_gray.astype(np.float64) -
    comparison_gray.astype(np.float64)
h, w = diff.shape
result = np.zeros((h, w, 3))
result[:, :, 1][diff > 0] = diff[diff > 0]
result[:, :, 2][diff < 0] = np.abs(diff[diff < 0])
```

Listing 4: Calculating difference image

My method of calculating the difference image gives more value to pixels that are further away from the comparison image. Areas on the image that show green mean that the resulting image has stronger pixel values there compared to the comparison image. Similarly, stronger red pixels show the comparison image has higher values at that cell location. A black pixel on the diff-image implies no difference.

B. Quantitative Metrics

For my numerical comparison, I'm using a Root Mean Squared Error (RMSE) metric. RMSE is an error metric, it determines how much the resulting image varies with respect to the comparison image. The higher the value of the RMSE, the more the deviation. The equation for RMSE is

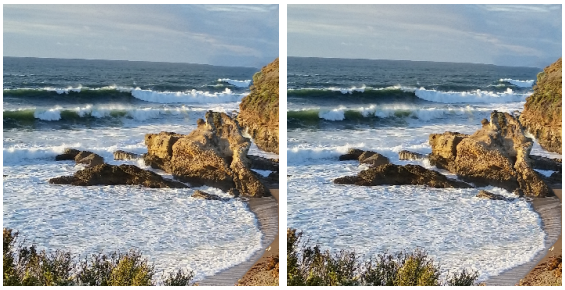
$$rmse = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}$$

Here y_i is a pixel from the result image, and x_i is from the comparison image. The summation is over all n cells in the array representing the image. This includes all three channels for a color image.

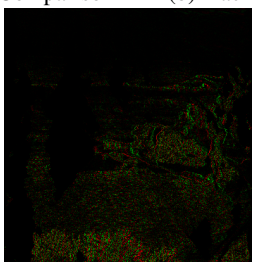
```
error = result_image.astype(np.float64) -
    comparison_image.astype(np.float64)
rmse = np.sqrt(np.square(error).mean()
```

Listing 5: Calculating difference image

RMSE has two advantages. First, it gives higher weight to larger errors. It is not enough to know that two pixels do not match. Knowing how far off they are in value adds a high level of error detail. This attribute is a result of the squaring of error. The larger the error, the higher the result value of the square. This in turn affects the mean. Second, instead of just using MSE, which also provides the same weighting as just discussed, the square root operation ensures the error is in the same units as the base item being compared. In our case, this would be pixel intensity values.



(a) Backward Comparison (b) Backward Result



(c) Backward difference image

Fig. 1: Beach - Seam Removal

III. Replicated Images and Comparison of Results

A. Backwards Energy Seam Removal (2007): Beach

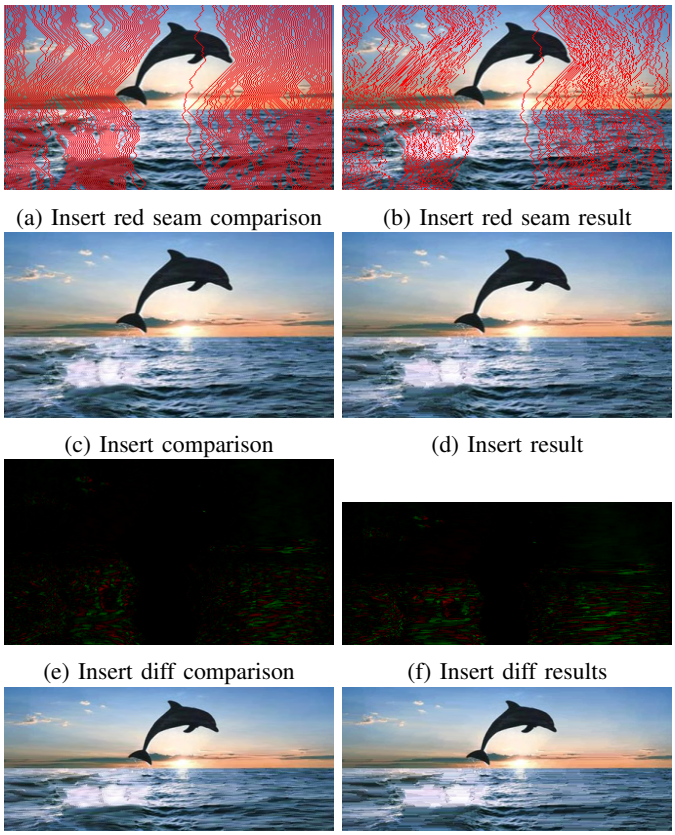
Comparison	RMSE
beach backward energy vs. comp	30.693

This natural scene has quite a few objects with a textured pattern. This makes it a little difficult to see the differences directly. For example in the shrubs on the lower left or the foam in the water on the shore. A closer inspection shows that the tide line is further towards the right edge than in the comparison image. The large boulder touching the right edge of the image is also thinner. On the whole, the boulders and sand look like they have shifted to the right.

The difference image shows us black areas signifying no change difference at all. We can see strong edges on the boulders showing the movement of these rock formations relative to the comparison image. There is also quite a bit of movement in the lower left. The shrubs have shifted a bit, and due to their speckled nature, we see the small differences as dots. There are only small specs on the horizon meaning they match quite well.

The main difference between the expected and result image is most likely due to the energy function. Experimenting with energy functions that use a gaussian blur affects the RMSE score (both cv2.Sobel and cv2.Scharr resulted in higher RMSE errors). The image has a lot of natural texture which can be seen as noise and smoothed out in some energy functions. Purely based on RMSE score 30.693, using filter2D with an asymmetric kernel provided a lower RMSE score than Sobel or Scharr.

B. Backwards Energy Seam Insertion (2007): Fig. 8 Dolphin



(a) Insert red seam comparison (b) Insert red seam result

(c) Insert comparison (d) Insert result

(e) Insert diff comparison (f) Insert diff results

(g) Insert double comparison (h) Insert double result

Fig. 2: Main caption

Comparison	RMSE
res_dolphin_back_ins vs. comp	12.473
res_dolphin_back_double vs. comp	16.473

Single insert images: In relation to the comparison image 2c, the ocean waves are right-shifted in my result image 2d. The clouds on the right side of the horizon area are also slightly right-shifted. A closer inspection of the difference images shows that while the dolphin matches in both images, there is one small area between the lower fin and body that sticks out on the result image when the images are toggled back and forth. This little patch is not on the original dolphin image and suggests the algorithm used to generate the comparison had a seam going through the image as well. This is verified in the red seam comparison image 2a and my red seams result image 2b which also has a seam going through the dolphins body. The difference occurs in how the fill is handled. The comparison image uses dark pixels of the fill, while I use the average of the fin and the sky which tends towards a lighter fill.

Double insert images: The double insert difference image has similar characteristics to the single insert image. The waves and sky seem right shifted in my result image 2h. While quite similar, I subjectively prefer the waves in the comparison image 2g. They seem less stretched out, almost as though a little blur was applied to the local area after the seam was inserted.

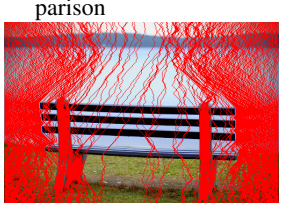
The RMSE value is higher on the double insert than the single one. My implementation applies the same seam twice. If the comparison image double insert was implemented in a similar fashion the RMSE should have stayed similar. Since it increased it shows that the second inserts were not handled in a similar way.

During the energy function implementation, I had experimented with a Sobel and Scharr gradient operation and my 2D filter had a lower RMSE score than both. OpenCV's Soble and Scharr operators combine gaussian blurring which might have been a reason for the large difference.

C. Seam Removal by Two Methods (2008): Fig. 8 Bench

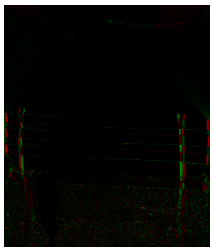
(a) Backward comparison

(b) Backward result

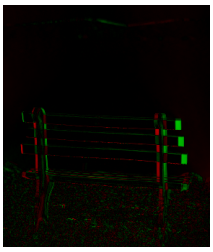


(c) Backward red comparison

(d) Backward red result



(e) Backward diff



(f) Forward diff

(g) Forward comparison

(h) Forward result

(i) Forward red comparison

(j) Forward red result

Fig. 3: Bench back

Comparison	RMSE
bench backward energy vs. comp	17.342
bench forward energy vs. comp	22.888

The main difference between the comparison 3a and the result 3b for the backward energy algorithm is the bench in the result image is wider and the bench legs are slightly farther apart. In both the red seam images 3c and 3d we can see that the seams are concentrated on the image edges and the bench legs. The difference image 3e also shows that a large portion of the image, especially the water, has very little difference. The offsets are quite visible at the edge of the bench and in the grass, as the grass is textured, even small misalignment can produce a speckled result as we are seeing.

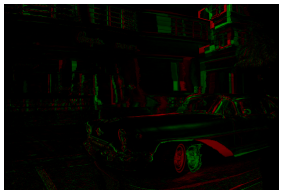
On the whole, the forward energy algorithm produced more visually pleasing seam removal results. The main visual difference again is an offset. My result image 3h is slightly left shifted relative to the bench in the comparison image 3g. The planks on the back-rest also have a tapered effect on the left side of the bench in the forward result image. While both comparison and result show the taper on the right-hand side. We can see this taper quite clearly in the difference image. Lastly, because of the right shift, the outlines of the mountains in the background also don't line up which is seen as a v-shaped green and red line in the top section of the difference image.

This image was very strongly affected by the choice of the energy function. It was during the development of functions to produce this image set that I realized I need to use a border wrap on my image while taking gradients in the x and y directions. Without the gradients, all seams were targeted at the edges. The effects appeared as though the image was being cropped from the sides while keeping the center completely intact. For the most part, this image is dominated by parallel lines running from left to right. Having a slight angle in between the left and right edges of the shore and mountain range really comes into play when using a border wrap, as the edges are favored less for removal than the centers. A good number of the seams also pass through the bench legs. As the bench legs are a man-made material, there is less texture compared to the grass so removing pixels from that section doesn't increase the overall energy in the image.

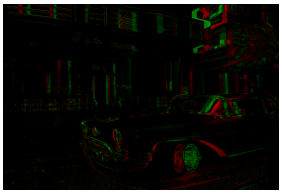
D. Seam Insertion by Two Methods (2008): Fig. 9 Car

(a) Backward insert comparison

(b) Backward insert result



(c) Backward insert diff



(d) Forward insert diff

(e) Forward insert comparison

(f) Forward insert result

Fig. 4: Main caption

Comparison	RMSE	Metric2
car backward energy vs. comp	21.21094684	Value2
car forward energy vs. comp	22.60241777	Value2

This was one of the harder images to align. There are many visible differences using the backward energy algorithm, most notably the wheel is both offset and visibly distorted in the result 4b. The edge of the wheel well is also offset to the right relative to the comparison image 4a. There are many vertical lines in this image, like the door frames, and the railing on the buildings. The seam insertion is such in the backward algorithm that the vertical door frames are not kept intact.

The difference images in both the backward 4c and forward 4d energy algorithsm show that the result image has a significant misalignment as the image goes from left to right. This makes sense as seams closer to the left edge will affect all pixels to the right, but not the other way around.

For the rest of the image, the energy function I used did reasonably well. Where the backward energy algorithm didn't do well, the forward energy results aligned pretty well with the comparison image. In this case, both backward and forward have issues complying as can be seen in the very close RMSE scores. As the forward algorithm doesn't use an energy function per se, it is most likely a different insertion and book keeping mechanism that needs to be used for this type of an image.

IV. Ambiguities and Issues

The primary papers used for the algorithm did not explicitly address if any special handling was required for the left and right edges of the image. The algorithms discuss steps where we need to look to the left and right of the current column, for example when calculating gradients, and needs to be addressed in a particular way if indexing lower than 0 or higher than the image width could be an issue. In my implementation, I need to use a border-padded image with a wrap-around border to get suitable results. This was discovered through my own assumptions about how to handle this edge case.

Seam offsets for image insertions are not very clearly discussed in the paper. Fortunately, this was discussed during office hours and the student forum but would have been a major stumbling block for someone trying to implement this algorithm without this insight. Using this direction and creating toy matrices in Excel to visualize the offset over a few iterations was key to figuring out the offset model.

Lastly, the papers do not discuss how, if possible at all, the algorithms can be tuned - the algorithm described is as is. There are no parameters that can be tweaked to tune the system. My implementation produces some images that do not align well with the comparison image. However, there is no clear mechanism by which to tune the results. For backward energy, using different energy functions is a way, but forward energy seems to have a set structure.

References

[1] S. Avidan and A. Shamir, 'Seam carving for content-aware image resizing', ACM SIGGRAPH, 2007

[2] M. Rubinstein, A. Shamir, and S. Avidan, 'Improved seam carving for video retargeting', ACM SIGGRAPH, 2008

[3] OpenCV docs, Tutorial Sobel Derivatives, cv2.Sobel, Accessed: Nov 10, 2022

[4] OpenCV docs, Image Gradients, Image Gradients, Accessed: Nov 10, 2022

[5] The Datascientist, Performance measures: RMSE and MAE, RMSE, Accessed: Nov 12, 2022

[6] Statology, Performance Metrics, MSE vs. RMSE, Accessed: Nov 12, 2022