Project 2
Content-Aware Seam Carving

Trevor Alexander Davidson
CS6475 Spring 2024
tdavidson30@gatech.edu

I. Discussion of Algorithms

A. Backwards Energy Algorithm

The backward energy seam carving algorithm used in this paper is based on the methodology outlined by Avidan and Shamir in section 3 of their 2007 paper [1]. The purpose of this approach is to identify and remove pixels that are similar to their neighbors, thereby minimizing information loss across the image. To begin, this requires selection of an appropriate quantitative measure of information. Avidan and Shamir's initial recommendation for a simple energy function is *e1*, which adds the absolute values of the image gradients together. Although they examine other possible energy functions, the paper ultimately determines *e1* to be an acceptable energy function roughly equivalent to the *eHOG* option they specifically propose [1]. As a result, *e1* was used as the energy function of choice for this project.

Avidan and Shamir do not specify a specific kernel to use in the calculation of gradients. After some experimentation, I found that using the Sobel operator with ksize=3 provided the good results across most images, although these exact values were adjusted depending on the image set. For each execution of the algorithm shown in this project, I converted the input image to grayscale. This increased the speed of the algorithm and provided better overall results than splitting and recombining image channels.

Avidan and Shamir also specify a dynamic programming process to determine the cumulative minimum energy of each pixel. As specified in the paper, the CME is determined by combined by adding the current pixel energy value and the smallest value from that pixel's 8-connected neighbors from the row above. Once this matrix is complete, retracing the minimum path from the bottom row of the image identifies the best seam for removal.

It is not explicitly specified by the writers whether the energy values for each pixel and subsequent arrays should be recalculated at each seam removal. Although not specified, it would logically follow that removal of a seam from the M-matrix would require the matrix value to be recalculated, since the minimum neighbor may have changed. This in turn implies that the energy values should be calculated as well. Therefore, after each pixel was removed, I recalculated the energy and cumulative energy values for each pixel in the same manner until all required seams were removed.

B. Forwards Energy Algorithm

The forward energy algorithm used in this project is based on the subsequent collaboration and improvements made to the original seam carving algorithm by Rubenstein, Shamir, and Avidan in 2008 [2]. This algorithm is quite similar to the Backwards Energy Algorithm: I still used the Sobel operator to calculate the e1 values, a cumulative minimum energy array is created, and a seam is repeatedly removed based on backtracking through the array once filled. However, Rubenstein specifically propose an additional cost in section 5.1 of their work [2]. The additional penalty is included because seam removal creates new boundaries between pixels that did not exist prior. The precise cost functions are dependent on direction: vertical movement of the seam creates a lower cost than moving left or right.

The Rubenstein paper does not address cases where the seam can approach the edge. In this case, the number of seams created is decreased. For example, removal of two pixels connected vertically and placed along the edge of the image creates no new boundaries between pixels, only between pixels and the image edge. Additionally, a seam which escapes from the edge creates only one new boundary, a horizontal line between (i, j) and (i - 1, j). This cost difference is reflected in my forward energy cumulative energy calculation.

Outside of this change, the forward energy algorithm keeps the backward energy methodology in place.

II. Comparative Metrics

A. Difference Images

The goal of the difference image is to visualize the areas where the two seam-carved algorithms have an alternative pixel value, with some method for differentiating how much those pixel values vary.
To execute this, I first converted both images to int32 and computed the absolute difference between each pixel value across all channels. I then calculated the average of these values across each channel to get a pixel value for each (row, column) pair. This one-channel mean was then distributed between three channels and reconverted to uint8.

The output is a black and white difference image where white pixels indicate a 255 average difference in pixel value while a 0 average difference reports as black. Large differences are then highly visible, as the white pops from the black, but gray patches are still noticeable in areas where there are minor changes. Ultimately, this allows the user to view any shifts in major features.

B. Quantitative Metrics

Defining visual similarity requires a more detailed algorithm than a simple difference image. In addition to shifts of the image features from the carving algorithm, small details may be removed from the image, altering the image content. Therefore, a good algorithm will evaluate what content is visible in the images more than the exact location. The Bidirectional Similarity Measure (referred to as BDS going forward) proposed by Simakov et al. does this exactly [3]. BDS iterates through both the "source" (original) image and the "target" (seam carved) image and compares patches between each image to the other and calculates a dissimilarity score.
This method was originally developed for evaluating information loss from the seam retargeting algorithm. For the purpose of this project, I have implemented it for comparison between two retargeted images and renamed it altBDS. To reduce computational time, both my result image and the comparison image were converted to grayscale before starting altBDS.
Similar to the variables outlines in [3], let S and T be the two equal size input images. Let P and Q be 7x7 subsets sampled every 9th pixel of S and T respectively. Let D(,) be the normalized sum of square difference between two image patches and let L be the number of layers in the gaussian pyramid for both images. Let N be the number of total valid P and Q locations in S and T across all L.

$$d(S,T) = \frac{1}{N}\sum_{L=1}^{3}\sum_{P\subset S}\min_{Q\subset T} D(P,Q) + \sum_{Q\subset T}\min_{P\subset S} D(Q,P)$$

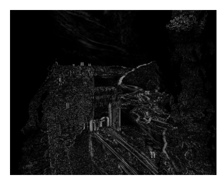
As specified by Simakov in [3], a gaussian pyramid is created for both images. Since Simakov specified "multiple scales" [3], I chose to use a 3-level pyramid. I reused the code from my Assignment 2 submission for this purpose.
At each pyramid level, I then created a 7x7 template patch from each image and used cv2.matchTemplate to find the most similar patch from the other image. BDS specifies this should be done for every valid template location in each image [3]. For the purpose of reducing processing time, I cut this down to every third column position and every third row position. This reduced runtime to approximately 1/9th of the original runtime while ensuring center pixel values were included in the template at least twice each.
The minimum correlation value at each template was determined using the normalized sum of square differences method from OpenCV. These results were added together and divided by the number of template locations, creating the final dissimilarity metric similar to the original BDS method.
Both backward and forward seam carving aim to avoid crossing into high energy pixels. In this project, high energy is defined by a large gradient. Patches with multiple gradients are often good features. The gradient-feature relationship is also why this algorithm works: feature preservation results in a low dissimilarity score.

III. Replicated Images and Comparison of Results

A. Backwards Energy Seam Removal (2007): Conwy

(a) Conwy Comparison Image

(b) Conwy Result Image



(c) Conwy Difference Image

Fig. 1: Conwy Backwards Removal Results

Comparison	altBDS
conwy backward energy vs. comp	0.01210

The conwy image result generally maintains almost all of the key features. The castle is relatively similar, all boats are present, the beach looks similar, the clouds look similar and the rock structure mostly appears the same. The fine details around the railroad tracks are a bit different, but otherwise they look similar. There are some minor differences in the beach and the rock structure, but other changes are difficult to identify.
The difference image highlights some of these. The coastline appears to be slightly shifted to the right in the result image, suggesting there is a bit less rock formation after seams ran

through them. Other changes are in smaller features, and are likely similar noise from the rock movement.
Finally, the altBDS metric suggests the result image is moderately different from the comparison with a score of 0.01210. With many small features, the matching system is going to struggle more on this image to match templates than a smaller image. However, the image match is strong enough that altBDS evaluates the score favorably.

The preservation of the castle, boaters, train tracks, and sky suggests the energy function was likely okay. The only likely issue appears to be with the rock formation. Some of this may be due to the effects of grayscaling the images, which may have blended some otherwise different areas more seamlessly, creating possible seam paths.

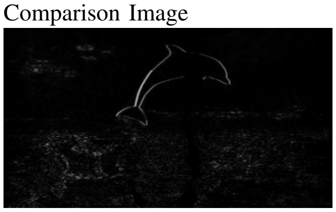
B. Backwards Energy Seam Insertion (2007): Fig. 8 Dolphin

(a) Dolphin Backward Insert Comparison Image with Red Seams

(b) Dolphin Backward Insert Result Image with Red Seams

(c) Dolphin Backward Insert Comparison Image

(d) Dolphin Backward Insert Result Image



(e) Dolphin Backward Insert Difference Image

(f) Dolphin Backward Double Insert Difference Image

(g) Dolphin Backward Double Insert Comparison Image

(h) Dolphin Backward Double Insert Result Image

Fig. 2: Dolphin Backwards Energy Insertion Images

Comparison	altBDS
res_dolphin_back_ins vs. comp	0.001736
res_dolphin_back_double vs. comp	0.001835

The both the single insert and double insert dolphin images appear extremely similar to the comparison images. The only notable point of difference is a hitch in the dolphin's back in both comparison images which is not present in the result images. This makes sense based on the red seam comparison image, which shows a few red lines through the dolphin's body where the result red seam image does not have any seams in those places.
The difference image also suggests there is a slight scaling and positional difference of the dolphins, likely due to some of the seam variations.
The altBDS results reflect both of these realizations with an exceptionally good matching score. All of the primary features are carried over, including more difficult features like clouds, making the feature matching process very effective and reducing dissimilarity.

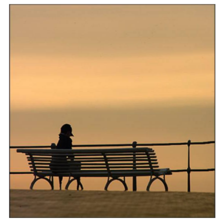
The redseams image suggests there is a code artifact affecting results slightly. To determine which seam to cut at each point, I use np.argmin. However, this function has the downside of picking the first minimum element. To offset this and try to balance the process, I subtract 1 from the middle image, meaning ties will now go to the vertical seam. This is a possible explanation for why the seams on the right side of the dolphin are so straight an do not re-contour around the dolphin near the top of the image. With little gradient in the sky, this subtraction could influence the outcome.

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

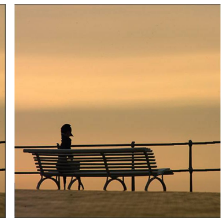
Comparison	altBDS
bench backward energy vs. comp	0.002183
bench forward energy vs. comp	0.003456

The backwards removal images appear to produce relatively similar results: the comparison has a thinner right post while the result image has less person remaining, but otherwise both images match well. This is reflected in a very similar red seams image for both. However, while the forward energy images for the comparison keep the person almost entirely intact and avoid the right post completely, the result image does not improve much at all from backwards energy. Both the post and the person are still mostly consumed by seams. The difference images reflect this reality as well. In both images, the difference in the person are the most clear and are the primary point of difference. Although there is some difference in the benches, the red seam images suggest this is affected by the proportion of seams on each side of the bench. Following altBDS directly would suggest that the backwards energy method works better for this image set. However, results from [2] suggest that this is more likely a cause of the energy function.

In particular, the chosen seams do not tend to stray towards the image. Based on [1], *eHOG* would likely do a better job with this image since the edges are ideal places for seams to be removed in this image, particularly to the right of the bench.



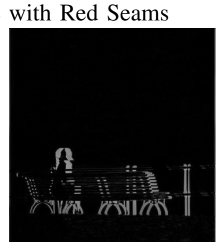
(a) Bench Backwards Energy Removal Comparison Image



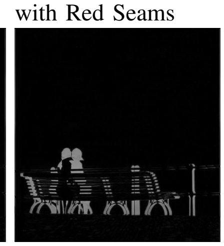
(b) Bench Backwards Energy Removal Result Image

(c) Bench Backwards Energy Removal Comparison Image with Red Seams

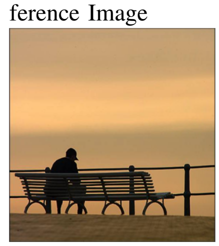
(d) Bench Backwards Energy Removal Result Image with Red Seams



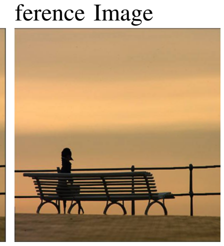
(e) Bench Backwards Energy Removal Difference Image



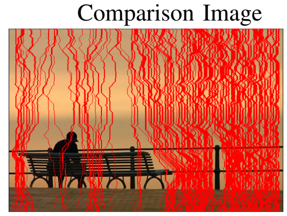
(f) Bench Forwards Energy Removal Difference Image



(g) Bench Forwards Energy Removal Comparison Image



(h) Bench Forwards Energy Removal Result Image

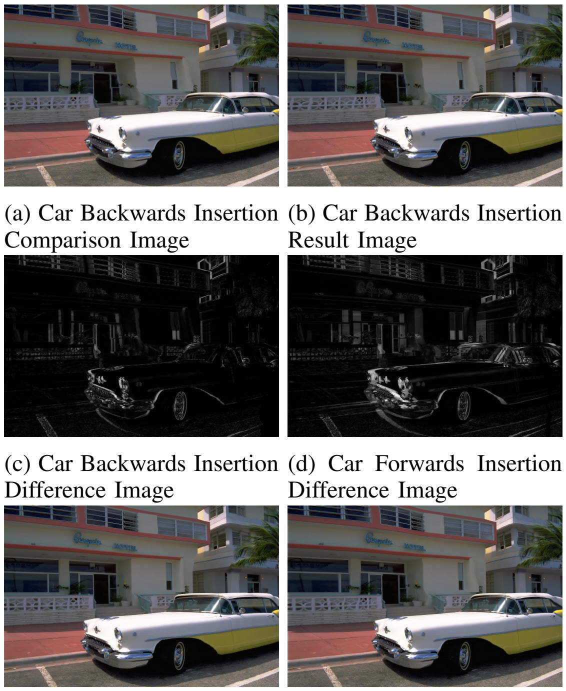


(i) Bench Forwards Energy Removal Comparison Image with Red Seams

(j) Bench Forwards Energy Removal Result Image with Red Seams

Fig. 3: Backwards and Forwards Energy Seam Removal for Bench Image

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



(a) Car Backwards Insertion Comparison Image (b) Car Backwards Insertion Result Image

(c) Car Backwards Insertion Difference Image (d) Car Forwards Insertion Difference Image

(e) Car Forwards Insertion Comparison Image (f) Car Forwards Insertion Result Image

Fig. 4: Car Backwards and Forwards Insertion Images

Comparison	altBDS
car backward energy vs. comp	0.006733
car forward energy vs. comp	0.007374

For the backwards insertion images, the curvature of the wheel well is the most noticeable difference: the comparison image has sharp corners and jagged edges while the result image has a smoother but clearly elongated profile. Other differences, such as the facade profile of the the hotel, are more subtle but still visible. Generally, this is a good match
The forward images correct much of these differences, particularly in the windows. However, the facade still has the same issues and wheel well differences are still noticeable. There are some undesirable slalom curves that are left in the forward image.
The difference images mostly reveal the same details: the facade is slightly different in each case. However, the forwards insertion actually has more issues around the wheel well and the front of the car. This suggests there may be more seams running through the car in comparison to the backward energy.
The altBDS results validate this difference: the forward insertion does a slightly worse job of matching the comparison than the backwards image, mostly due to the changes in the car. However, the difference is pretty minimal, as the altBDS scores are fairly similar.

The result images likely differ from the comparison image because of the energy function and color space of the image. Based on the number of seams going into the image and the extension of the car in both image sets, the possible concept is there are fewer seams going around the left side of the car and more going through the car, resulting in the insertion extending the car and shifting the image to the (relative) left. This likely due to some combination of the color contrast not being strong enough in the grayscale image or a less than ideal energy function. In particular, *eHOG* may have been a strong option here, as [1] specifically mentions that *eHOG* does an improved job of pulling seams to the edges of the image, which could bring them around the car.

IV. Ambiguities and Issues

Replicating the energy function was a consistent issue across a images in both research papers. Not only does neither paper list the energy function used for each image in this project, but they also provide very limited detail on what other inputs they used. For example, I used a Sobel operator in this project for calculations of gradients, but a Scharr operator could easily have been used. This likely would have affected my answer, and required me to experiment as much as I could with the kernel size to get the best results.

The second issue was with channels usage. Although the project Readme specified that all channels must be used, the papers never clarified how they manipulated the channels or combined them. I stuck with BGR colorspace since that was what I was most familiar with, but I did convert to grayscale across all images, thinking this would produce results that best matched human perception of image gradients. Based on the results of this project, I suspect there is a better way to integrate the channels together.

Finally, neither of the papers deals with cases near the edge of the image. Theoretically, it would seem possible to reflect the energy values across the image border. However, since this was never specified in the papers, I accounted for the edges by reducing the pixels reviewed to find the minimum path for backward energy and adjusting the additional cost parameters imposed by forward energy.

References

[1] Avidan, S., Shamir, A. (2007). Seam carving for content-aware image resizing. ACM SIGGRAPH 2007 Papers. https://doi.org/10.1145/1275808.1276390

[2] Rubinstein, M., Shamir, A., Avidan, S. (2008). Improved seam carving for video retargeting. ACM SIGGRAPH 2008 Papers. https://doi.org/10.1145/1399504.1360615

[3] Simakov, D., Caspi, Y., Shechtman, E., Irani, M. (2008). Summarizing visual data using bidirectional similarity. 2008 IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2008.4587842