Inpainting Program

Owen Culmer
opc4@uakron.edu

Follow this and additional works at: https://ideaexchange.uakron.edu/honors_research_projects

Part of the Theory and Algorithms Commons

Please take a moment to share how this work helps you through this survey. Your feedback will be important as we plan further development of our repository.

Recommended Citation

Culmer, Owen, "Inpainting Program" (2024). Williams Honors College, Honors Research Projects. 1830.
https://ideaexchange.uakron.edu/honors_research_projects/1830

This Dissertation/Thesis is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.
Object Removal Algorithms: Seam Carving and Inpainting

Owen Culmer

The University of Akron

Williams Honors College

4/26/2024
Abstract

Image processing is a quickly developing field of computer science, especially with the growth of computer vision and generative artificial intelligence. For my project, I focused on object removal, where a targeted object or area can be removed from a digital image. I first accomplished this by implementing a seam carving algorithm, where rows or columns of pixels called *seams* are removed from an image, that prioritizes the removal of seams that include targeted areas. I next explored a digital inpainting algorithm that would replace targeted areas, rather than removing them, in order to maintain the exact image dimensions and to prevent distortion.
Table of Contents

Section I - Seam Carving ............................................................................................................ 4
  Introduction .......................................................................................................................... 4
  Implementation .................................................................................................................. 5
  Results ............................................................................................................................... 7

Section II - Inpainting .......................................................................................................... 10
  Introduction ...................................................................................................................... 10
  Implementation ............................................................................................................... 10
  Results ............................................................................................................................. 13

Conclusions .......................................................................................................................... 15

References .............................................................................................................................. 16
Section I - Seam Carving

Introduction

Seam carving is an algorithm developed by Shai Avidan and Ariel Shamir, first described in a paper they published in 2007. This algorithm allows for the resizing of an image by carving seams out of an image, or by inserting seams into an image. These seams are continuous lines of pixels, either adjacent or diagonal to the preceding and seceding pixels, that traverse the full width or height of the image. The seam carving algorithm optimally selects seams to remove by finding the seam with the lowest possible energy, which is measured as the sum of the differences between the color values of a pixel and its adjacent neighbors. To calculate a seam’s total energy, the energy values of all the pixels it traverses are added together. The algorithm repeats this process of energy calculation and seam removal until the image has been resized to the desired dimensions.

In order to achieve my goal of object/area removal, I modified the seam carving algorithm so that it would prioritize a specified area. This is accomplished by setting the energy values of all pixels within the targeted area to be a negative value of large magnitude. This ensures that if any targeted pixels remain in the image, seams will seek to include these pixels, as during the process of finding the best seam to remove, potential seams seek to minimize their total energy. Additionally, the seams that have the lowest energy values will be the ones that include the most targeted pixels, ensuring that as little of the non-targeted area as possible is removed.
Implementation

I implemented the seam carving algorithm as a console application written in Python. The program first reads in a black-and-white .pgm image file, which must already have a targeted area, marked by a specific color value. In order to preserve the ratio of the image’s dimensions as much as possible, seams are carved both vertically and horizontally in a ratio similar to that of the image dimensions. For example, if the image has a height of 50 pixels and a width of 100 pixels, two vertical seams will be carved for every one horizontal seam. Thus, if the targeted area has been removed after nine seams are carved (six vertical, three horizontal), the new image will have a height of 47 and a width of 94, maintaining the original ratio. The ratio itself is calculated by dividing the width by the height, and rounding that to an integer.

With this, the seam carving process begins. First, the algorithm carves the number of vertical seams determined by the ratio, then carves one horizontal seam. After each of these seams is removed, the image is checked for remaining targeted pixels. If none exist, the process ends. Otherwise, the seams are removed until no more targeted pixels remain. Vertical and horizontal seams are carved using the same method; the image is simply transposed so that the seam is carved horizontally, then re-transposed after the seam has been removed.

To begin the process of removing a seam, the energy of each pixel is calculated. This is done by calculating the differences between the color value of each pixel and its upward, downward, leftward, and rightward neighbors, if they exist. The sum of the absolute values of these differences is assigned as the pixel’s energy value. Then, the energy values of any pixels targeted for removal are overwritten with the value of -10,000 in order to ensure that the seams with the lowest energies are ones that remove targeted pixels. The next step is to identify the seam, out of all possible seams, whose removal results in the loss of as little energy as possible.
This is done by calculating the lowest sum of energy that can be obtained at each pixel if the seams start at the top of the image. Thus, each pixel in the top row has an energy sum equivalent to its own energy; each pixel in the row below them has an energy sum equivalent to its own energy, plus the lowest energy sum out of the pixels that are directly or diagonally above it. These sums are calculated for each row until the bottom row is reached, revealing the lowest possible sum of energy that can be had by a seam ending at each pixel. The seam with the lowest energy sum that contains a targeted pixel is selected for removal. If no seam contains a targeted pixel, then the seam with the lowest energy sum is selected, but this is unlikely due to the extremely low energy of the targeted pixels.

Once a seam has been selected, the process of removal begins. The removal process involves a recursive function, which starts at the bottom and retraces the seam towards the top of the image, marking the pixels that will be removed. The function first marks the pixel for removal, then checks for the base case, that the current pixel is at the top; if so, the function returns, beginning the process of its unwinding. If the current pixel is not the base case, it finds the pixel directly or diagonally above it with the lowest energy sum. The function then recursively calls itself with this new pixel, which will continue until the base case is reached and the unwinding begins.

Once all pixels within the seam have been marked, a modified copy of the image is made. The new width for this copy is one less than the current image’s, as one of the columns will be removed. The values of the pixels are then copied into the new image, except for the ones which have been marked for removal. This copy then replaces the original, and the seam is now removed.
Results

Overall, the seam carving algorithm is effective at removing objects from images. As seen in Figure 1 below, the algorithm can result in the removal of a specified area with minimal distortion to the rest of the image. While the door has been removed, other elements of the image such as the foreground, trees, and building look largely unmodified. Additionally, the ratio of the image dimensions is largely preserved, leaving the image as close to the original as possible while still removing the targeted area.

![Figure 1](image1.jpg)

**Fig. 1.** On the left is the image input into the program, where the door has been targeted for removal. On the right is the image output by the program.

Figure 2 shows the removal of a more naturally shaped area. Again, the specified area is removed with no distortion, and the ratio of the image dimensions is close to that of the original. The amount of the picture taken up by the sky has clearly been reduced, as it is the least energetic portion of the image. The mountain and woods below, on the other hand, are largely unaffected by the carved seams due to their higher energy.
Fig. 2. The left image shows a picture of two mountains, with the mountain on the left marked for removal. On the right is the image output by the program.

However, in some cases seam carving can result in the distortion of an image as a result of the composition of the image. The image in Figure 3, for example, has been distorted due to the distribution of high- and low-energy areas in the image. The lower section of the building, next to the stairs, has little variation in color, and so it has the lowest energy. Thus, the carved seams follow a path that curves around the grass, resulting in the bottom left and right sections “moving up” in relation to the rest of the image.

Fig. 3. The left image shows the picture of Buchtel Hall, where the path leading to the building is marked for removal. At right is the image after seam carving has been completed. In this case, the process has resulted in distortion, as the ground on each side of the path now slopes inward.
Additionally, the seam carving process does not guarantee the smooth transition of texture. As seen in Figure 4, when an area is removed from an image that is surrounded by a patterned texture (bricks, in this case), the pattern may not line up in the processed image. If the wall was textured differently, such as with a solid color, the distortion would be much less noticeable, if at all.

![Fig. 4](image)

**Fig. 4.** The original image is shown at left, and the carved image at right. Note the distortion that appears in the bottom right, where the targeted area was in the original.

In most cases, the seam carving algorithm works well, especially when the targeted area is bordered by low-energy areas, such as in Figure 2. The limitations of the seam carving algorithm are limited to cases where the targeted area is unoptimally located, such as amongst a patterned texture. In these cases, a process which does not remove the targeted area altogether, but instead modifies it so that it disappears, may be desired. Inpainting is one such process.
Section II - Inpainting

Introduction

Inpainting describes any process by which missing areas of an image are filled in, especially in cases in which an image has been damaged. Digital inpainting refers to the use of this process with digital images, often used to remove objects from an image. The use of inpainting for object removal has several advantages over seam carving; the exact image dimensions are preserved, and the process does not result in distortion, which can occasionally occur when seam carving is used. Thus, I explored the usage of inpainting for object removal by attempting to implement an algorithm described by Criminisi et al. in a 2003 paper. This algorithm replaces or fills in parts of an image based on data from the rest of the image. Specifically, patches centered around pixels of the targeted area are filled in with patches sampled from the non-targeted area, chosen based on their similarity to what is known of the targeted patch. My goal was to implement this algorithm in order to better remove or fill in parts of an image.

Implementation

I attempted to implement the inpainting algorithm as a Python console application. Note that I followed the algorithm described by Criminisi et al. as exactly as possible, and I will note deviations as they are described. The user first selects an image to use, which must already have an area targeted for replacement with a specific color value and be a black-and-white .pgm file. The first step of the repeating process as described is to identify the fill front, the patches along the border of the targeted and non-targeted areas. I accomplish this by marking any pixels which have the color value specified for replacement that are adjacent to any pixels which do not. If no
pixels are in the fill front, the image has been completed and the process ends. Otherwise, the potential patches, each centered on a pixel along the fill front, are prioritized based on a calculated priority value. Criminisi et al. describe this value as the product of two terms, the *confidence term* and the *data term*. The confidence term is the average confidence value of all pixels in the patch, where the values are initialized as 1 for non-targeted pixels and 0 for targeted pixels. Thus, the confidence value measures the amount of context each patch has, and the more context a patch has the more prioritized it will be.

The data term is more complex. First, the *isophote* of the patch is found, where the isophote is the maximum value of the image gradient in the patch. Pixels with a higher gradient value are those that are along some kind of border in the image. In my implementation, the image gradient is calculated by convolving the image data with a Sobel kernel. Next, the unit vector orthogonal to the line formed by the fill front, at the center of the patch, is found. In the paper by Criminisi et al., this is calculated based on the control points of the fill front, which are found by using a bidimensional Gaussian kernel. Instead, I calculate the orthogonal vector by approximating the slope of the fill front at the patch based on the pixels along the fill front that precede and secede the center pixel. From this slope, the slope that is orthogonal to it can be calculated. Finally, the dot product of the isophote and orthogonal vector is found, and the absolute value of this is taken. It is then divided by a “normalization factor”, which has the value of 255 for black-and-white images. The purpose of this data term is to account for the image geometry in the prioritization process. Patches that include or are adjacent to strongly-defined lines and shapes will take priority over those that are not, preserving the image geometry.

Once the patch priorities have all been calculated, the patch with the highest priority is selected. Now, a patch from the non-targeted area must be chosen to fill this patch. According to
the paper by Criminisi et al., the most similar patch is found by choosing the patch with the smallest sum of squared differences of the pixels that are filled in both patches. However, in my implementation this resulted in patches with the fewest number of non-targeted pixels being selected in most cases. This was often in a corner of the image, where ¾ of the patch is out of the image bounds and those pixels do not exist. In order to avoid this, I divide the sum of squared differences by the number of non-targeted pixels in the patch, finding the average squared difference of the pixels in the patch and no longer rewarding patches with fewer pixels (and thus a lower sum).

Finally, all the pixels in the patch that have been targeted for replacement are replaced with their counterparts from the similar patch. The confidence values of these pixels are updated, being given the value of the average confidence of the pixels in the patch. The process of finding the patches along the fill front, finding the one with the highest priority, and filling it with data from the most similar patch, repeats until the targeted area is completely filled.
Results

The results of my implemented algorithm were suboptimal. To measure the results of the algorithm, its ability to preserve the image geometry and replicate texture should be analyzed. The algorithm was able to preserve basic geometry with some success, as seen in Figure 5. Simple pictures of basic shapes or patterns could be filled with some accuracy, though errors are still present. However, larger areas are not filled well by my algorithm, which instead seems to create and then replicate its own geometry, instead of preserving the texture, as seen in Figure 6.

![Fig. 5.](image)

**Fig. 5.** On the left is the original image, showing a triangle and three circles, with a triangular area marked for replacement. On the right is the processed image, with mixed results. The top circles were completed, preserving their geometry, but the bottom circle was not as well preserved. Additionally, although the lines of the triangle were extended, the lines stop before connecting.
Fig. 6. On the left, the person in front of the fountain has been marked for removal. As seen at right, the inpainting process did not effectively replicate the fountain texture in most places. Instead of replicating the water's texture, a series of vertical lines were fabricated.

Fig. 7. These images are sourced from the paper by Criminisi et al. In order to compare the effectiveness of my algorithm, I attempted to inpaint the same area of this image with my algorithm, as seen in Figure 6.
The algorithm as shown in the paper by Criminisi et al., however, works well, as shown by their example in Figure 7, and I am uncertain where the algorithms diverge. As my algorithm performed better at preserving geometry than at replicating texture, it is likely that the process for selecting a patch to fill the target patch with is the source of error. Indeed, my algorithm had to use the average squared difference instead of using the sum of squared differences in order to avoid the overuse of filling the target area with data from patches with few pixels. Perhaps my code did not accurately replicate the algorithm as described, or perhaps some information about this process was discluded from the paper. Regardless, my algorithm was not a fully successful implementation of the one described by Criminisi et al.

Conclusions

The seam carving algorithm, despite some limitations, is efficient at removing objects from an image. In most cases, the targeted object or area can be removed without any distortion to other areas of the image. In a few cases, the image may have some distortion, and patterns are not guaranteed to line up if the object removal brings two patterned sections together. These problems could be solved by removing the object or area with an inpainting algorithm instead, but my implementation of this failed to yield good results. Reexamining the selection process for similar patches would likely improve the inpainting outcomes.
References
