Auto-Cropped Panorama

Saem Jeon
College of Computing & Informatics
Drexel University
Philadelphia, Pennsylvania
sj846@drexel.edu

Abigail Clune
College of Computing & Informatics
Drexel University
Philadelphia, Pennsylvania
aec332@drexel.edu

Git Hub Project

https://github.com/saemjeon/Auto_Cropped_Panorama

ABSTRACT

Panoramic image stitching allows for several images to be properly aligned relative to each other. The aligning of these images is done automatically or by a fixed order. In Brown's article, he introduced automated panoramic image stitching and its detailed algorithm [2]. Finding matching features is needed to achieve this. When the images are stitched together, panorama shows empty black pixels on the edges. With our experiment, we introduce 1) image stitching, 2) auto cropping to remove black pixels, and 3) inpainting of black pixels instead of cropping.

1 Introduction

Image stitching has been a research subject for recent decades. One application of image stitching, which will be the focus of this paper, is the creation of panoramic images. Panoramic images are those with elongated, horizontal fields of view. The entire field of view is from a single perspective, or origin point. This means that in the context of panoramic image stitching, all individual images to be stitched together need to be taken from the same origin point, i.e. a fixed tripod. Panoramic images are often desired for their artistic qualities, as well as the fact that more content can be fit within a wide, panoramic image. Image stitching is useful as it can create a panoramic image when a camera may not have that feature. While stitching can be a solution to creating great panoramas, image stitching also has drawbacks, as well. Often times, the final stitched image may contain empty, black space where an image could not be matched. Stitched images may also be misshaped along the edges, since the homographies used in the process can curve images in order to create the panoramic effect. There are also limitations to panoramic image stitching, such as the orientation of the input images, and slight variations in lighting and color between images.

2 Features

Three main features have been implemented into the panoramic image stitching program. These features include the basic stitching of the image, filling in remaining empty space and black pixels, and also cropping the images to include the most content while excluding the empty, black pixels. Each of these features provides very different results and may be desired depending on what type of content the image should show. While the following features are important for creating panoramic images, these features only scratch the surface of what could be possible in the future of panoramic image stitching and editing.

2.1 Image Stitching

2.1.1 Related Work

Image stitching is based on matching. Matching refers to both features within the images and the images themselves. There are several stages to stitch images together. First, an algorithm is used to match SIFT features, which are features that are located at scale-space maxima/minima of a difference of Gaussian function [2]. Such features are also invariant under rotation and scale changes [2]. This means that feature points within individual images are connected to matching points in other images, and images are rotated and scaled accordingly to match to appropriate points. At the next stage of mapping, images are mapped to other images, beyond simply feature points. This type of matching is to determine which images overlap and where. Image matching uses a probabilistic model which takes into account the number of matching features between images and the number of potential matching images and orientation of those matches to determine which images will get stitched together [2]. After it is determined which images will be stitched and at which features, a homography is applied to each image to align it to its matching, overlapping image. Finally, blending is applied to the overlapping sections to decrease the presence and obviousness of image edges and slight coloration discrepancies.

2.1.2 Contribution

Image stitching can be achieved easily using OpenCV's Stitcher_create() method which creates a stitcher object to stitch images together and blend images, creating a single, panoramic image. Since we did not want to reinvent the wheel, we simply utilized OpenCV's built-in functions for image stitching. This functionality only requires a list of images, and the stitcher then returns a single image. The built in stitcher accounts for blending necessary scaling to reduce artifacts.

2.1.3 Results

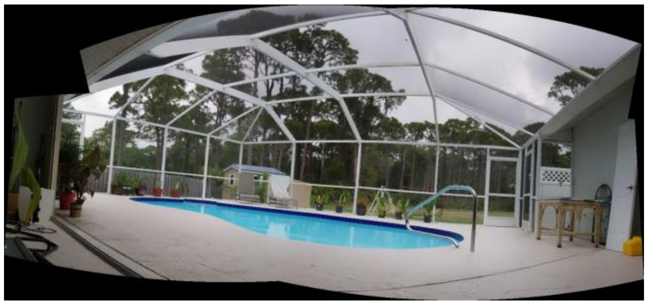

Figure 1

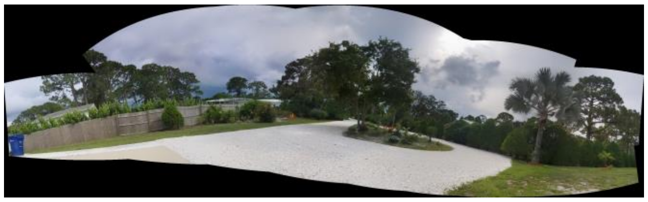

Figure 2

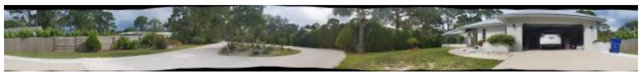

Figure 3

2.2 Filling Black Pixels

2.2.1 Related Work

When images are stitched black pixels are created on the edges of the result image. One way of removing the presence of black pixels is to fill those black pixels, which is called inpainting. There are two different algorithms that can be used for inpainting in OpenCV: Navier-Stokes based method and Alexandru Telea's method. For most inpainting algorithms, they began by using a rectangular domain that assumed that the shape of where the inpainting was needed to recover content was a regular shape. This has many limitations in most cases since most times, inpainting is required for irregular domains. In the case of a panoramic image, the empty pixels along the edge of the image are rarely regular shapes. Additionally, since they are along the edge, there is only one edge that can be referenced for what to fill in the empty pixels. The Navier-Stokes method accounts for irregular domains by finding the smallest bounding box around the pixels that need inpainting [1]. The inpainting is performed within this smaller bounding box, rather than over the entire image, which produces more accurate and more precise results. This algorithm propagates the smoothness across the empty pixels and iterates over time steps with the Navier-Stokes equations until a steady state solution is met [1]. When used on images, this appears to extend gradients and edges from both sides across the area of empty pixels. Telea’s method of inpainting works by mimicking manual inpainting techniques. Telea iteratively applies inpainting equations to discrete pixels to smooth the gradients over the empty pixels by increasing distance from the initial inpainting region [3]. This means that empty pixel areas closest to known image points are filled in first, then pixels farther from the known known image are filled [3]. This results in more realistic inpainting since known images gradients and edges are extended into the empty pixels.

2.2.2 Contribution

There is a clear difference between two methods. The filling was accomplished using built in methods and constants from Open-CV, and creation of a mask based on the empty pixels. After testing with different images, we used Telea’s method of infilling since filled edges using the Navier-Stokes shows more vertical lines than that of Telea's method. Both methods show artifacts, mainly due to the location of the black pixels. Those two methods work better for smaller areas and pixels that are in the middle of the image rather the edges. Because there is only one side of the empty pixels with known image reference points, there cannot be a realistic gradient filled between two areas of known pixels.

2.2.3 Results

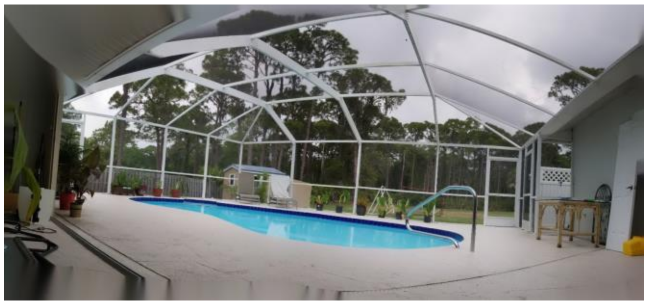

Figure 4: Applied inpainting on Figure 1

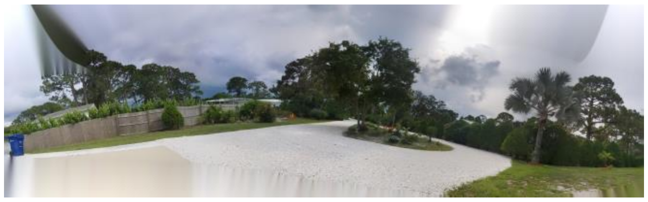

Figure 5: Applied inpainting on Figure 2

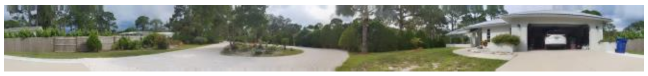

Figure 6: Applied inpainting on Figure 3

2.3 Auto-Cropping

2.3.1 Related Work

The good way of removing the black pixels created on the edges after image stitching is to crop out black pixels. Finding the biggest rectangular area to leave is challenging. There is not existing method that effectively crop the image. While researching different method of removing black pixels, we found an algorithm introduced by Adrian Rosebrock. His approach was first to find the gray version of the result image to create the mask and compute the smallest rectangular bounding box enclosing the panoramic image. Then, iterating through the mask image, when the black pixels are found, the size of the bounding box is reduced. His method, however, has some limitations. Since the algorithm decreases the size of the bounding box when the black pixels are encountered, it can sometimes decrease the bounding box too much thus the size of the remaining image can be very small compared to the original image. Additionally, we found this method to even leave behind small regions of black pixels after the cropping was complete.

2.3.2 Contribution

In our research, we could not find any built-in functionality to crop images, and as stated before, we only found one implementation of cropping which had many drawbacks. We decided to focus on cropping and create our own implementation of a cropping algorithm to correctly crop images while retaining as much content as possible and excluding all black pixels.
Our approach to overcome the drawbacks of Rosebrock's algorithm is to find the largest rectangle inside the panorama image instead of reducing the bounding box. There are few approaches that can be used to find the largest rectangle which only contains the actual image not the black pixels. The one we chose is to utilize the algorithm to find maximum rectangular area under histogram. We used a very well-known algorithm which is using stack approach. To apply this algorithm on the panoramic image, we need to first find sub arrays with the maximum height of the histogram each row. To distinguish black pixels from the image, we found the mask comprising of two integers, 0 and 255. 255 is for the black pixels, and 0 is for the non-black pixels. This makes the computation of row by row process to find sub arrays easier. We looped through the mask from the bottom and created sub arrays with maximum continuous count of 0s.

2.3.3 Results

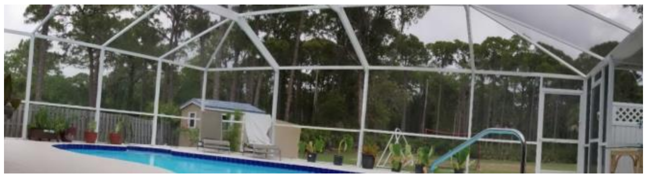
Figure 7: Applied existing implementation on Figure 1

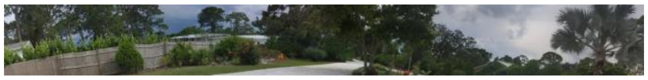
Figure 8: Applied existing implementation on Figure 2

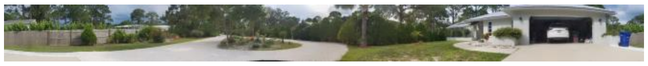
Figure 9: Applied existing implementation on Figure 3

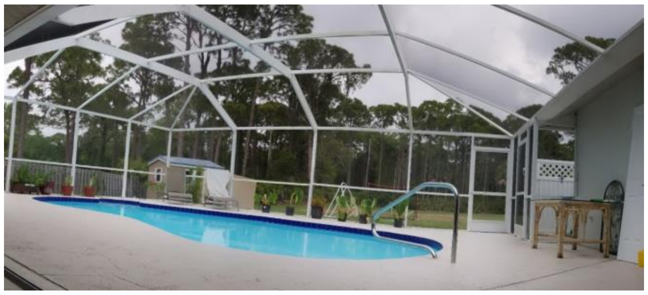
Figure 10: Applied our implementation on Figure 1

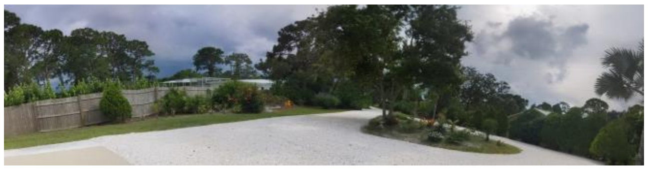
Figure 11: Applied our implementation on Figure 2

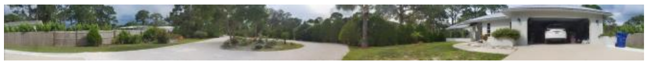
Figure 12: Applied our implementation on Figure 3

3 Future Research

While we did find interesting results from our research and exploration into panoramic stitching and cropping, there is a lot left to be explored. Several areas of interest that we would have attempted to implement, given more time, would be panoramic straightening, filling with textures, and video stitching. Video stitching is a very fascinating idea, in that it is truly a large scale, multi-image stitching. A single wide-angle video can be created by stitching images frame by frame [4]. The time complexity of this would be extremely high, and possibly take many hours to stitch a panoramic video of only a few minutes. Panoramic straightening would involve taking a stitched panoramic image that has a curve or wave to it and straightening the content so that the field of view is balanced and so cropping won't eliminate more essential content than need be. Filling images with textures would mean more than simply extending pixels from the known image, but also extending patterns and movement that is present in the image. These three areas of study are ones that there is some research done already, but not much implementation has been done.

REFERENCES

[1] Au, W. and Takei, R., n.d. Image Inpainting With The Navier-Stokes Equations. [online] Elynxsdk.free.fr.

[2] M. Brown and D. Lowe, Automatic Panoramic Image Stitching using Invariant Features,Lowe. International Journal of Computer Vision. 74(1), pages 59-73, 2007

[3] Telea, A., 2004. An Image Inpainting Technique Based On The Fast Marching Method. [online]

[4] Wei, L., Zhong, Z., Lang, C. and Yi, Z. A Survey On Image And Video Stitching.