Computational Photography
Assignment #5
Panoramas

Ronak Patel
Spring 2019

Results - Test Images

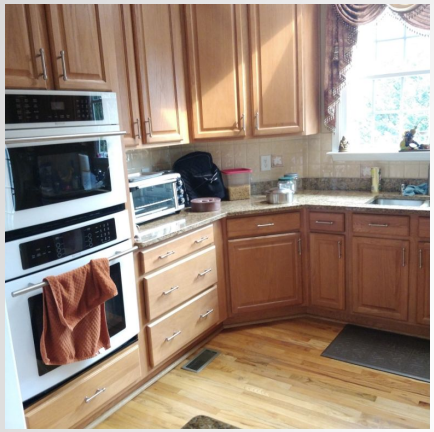
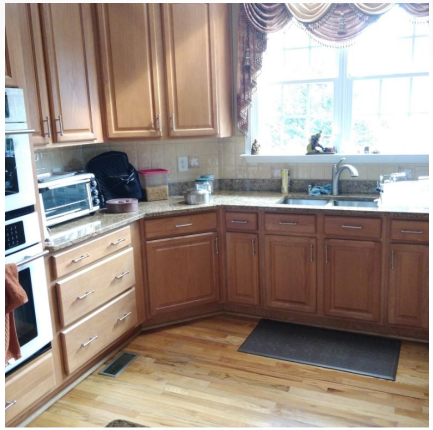
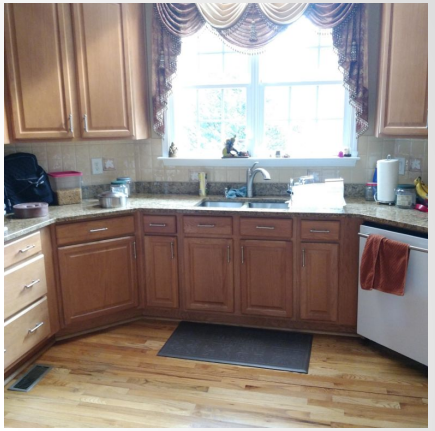
Setup and Panorama Type

- My camera was set up on my kitchen island and was held in place with a camera mount. Camera settings for each photo are described below:
  - Exposure Time: 1/40
  - Aperture: f/1.7
  - ISO: 800
  - Focal Length: 4.10MM
  - Resolution: 3456x3456

- My camera was rotated on my camera mount gimbal. Each rotation was done by hand.

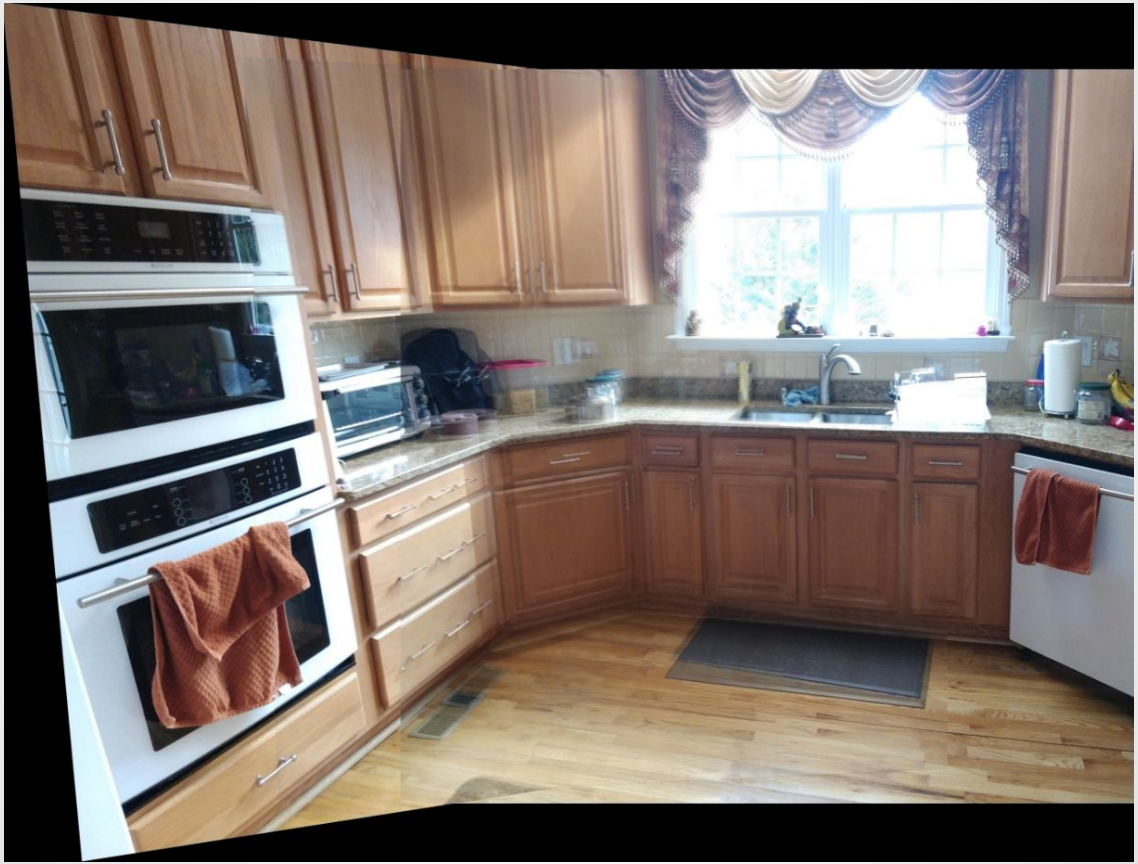
- **What kind of panorama did you make?**
  - An inner-cylinder panorama was taken since the camera remained stationary and was rotated on an axis to capture to scene.

Original Input Images

The scene captured was my kitchen. I took 3 images of my kitchen by rotating my camera on an axis between each image capture. I chose my kitchen for my panorama since it would algorithm would be able to find many unique features between each image for matching.

Results - Original Images



warpCanvas()

- Why multiply x_min and y_min by -1 in the warping function?

A negative translation is needed to place images onto the main panorama image/canvas. This allows for relevant features to be overlapped together, if this was not done, then the canvas would not fit all three images, features would not be matched, and the “panorama” would look like three separate images translated some x_min, y_min distance away from each other.

blendImagePair()

```
def blendImagePair(image_1, image_2, num_matches):
    kp1, kp2, matches = findMatchesBetweenImages(
        image_1, image_2, num_matches)
    homography = findHomography(kp1, kp2, matches)
    corners_1 = getImageCorners(image_1)
    corners_2 = getImageCorners(image_2)
    min_xy, max_xy = getBoundingCorners(corners_1, corners_2, homography)
    output_image = warpCanvas(image_1, homography, min_xy, max_xy)
    # WRITE YOUR CODE HERE - REPLACE THIS WITH YOUR BLENDING CODE

    temp = np.array(output_image)
    min_xy = min_xy.astype(np.int)
    output_image[-min_xy[1]:-min_xy[1] + image_2.shape[0],
                 -min_xy[0]:-min_xy[0] + image_2.shape[1]] = image_2

    yy = -min_xy[1] + image_2.shape[0]
    xx = -min_xy[0] + image_1.shape[1] + image_2.shape[1] - output_image.shape[1]

    i, j = 0, 0
    for r in range(-min_xy[1], yy):
        for c in range(-min_xy[0], xx):
            if temp[r,c].any() != 0:
                alpha = 0.45
                output_image[r, c] =  (alpha)*temp[r, c] +  (1 - alpha)*image_2[j,i]
            i += 1
        i = 0
        j += 1

    return output_image
    # END OF FUNCTION
```

Discuss your implementation for blendImagePair().

My blendImagePair function uses an alpha blend between the combined photos in the panorama to produce a final image panorama. An alpha value used was 0.45.

In the code the xx and yy values are the x and y dimensions for the end of the blending coordinates. Values min_xy[0] and min_xy[1], provide the dimensions for the start of the blending coordinates. Each pixel in the warped image (or the “temp” variable in the code) and image_2 is blended if the warped image does not contain any black pixels (pixel value of 0).

Discussion & Analysis

- Consider your blend function and answer the following questions - What worked well? What did not work well? Were there any problems you couldn't solve? If you had more time, what would you do to improve upon your existing blend implementation?

  There were areas in the panoramas where the boundaries between the images were blended well in the kitchen photo (top cabinets, microwave), but overall blending function could have performed better. In the sample photo, the blending operation did not seem to produce any drastic results and the borders of each image is still prevalent. In the kitchen image, the simple alpha blend resulted in ghostly artifacts (additional drawer handles in the kitchen photo) and general blurriness. To improve results, I would implement a pyramid blend to avoid these issues or better tune the alpha value (when alpha blending) based on local information or a blending mask.

- If you started the project over again, what could you do differently and how would you go about doing it? (Do not say there is nothing you could do differently. There's always a way to improve.)
  - To ensure better results I would fix my camera better so it only revolves around a vertical axis when taking pictures. Next, I would take photos that have more distinct features for matching. This will result in better warping and combining of images. Finally, I would perform a better blend between my images. Here, I did a simple alpha blend between my images, but if I were to do this again, I would try to correctly implement a pyramid blend function for better results or further improve my alpha blend by using blending mask.

Resources

R, T. R. (n.d.). Blend overlapping images in python. Retrieved March 10, 2019, from https://stackoverflow.com/questions/29106702/blend-overlapping-images-in-python