CS 378: 3D Reconstruction and Computer Vision, Extra Credit

Sai Avala
Siena McFetridge
Jack Carlson

September 21, 2014

1 Introduction

For the extra credit portion of the project, we implemented *image cut-out compositing*. We also wrote a script, my_pano_stitcher_lateral, to compute a panorama of a set of images by moving linearly along a planar space.

2 my_pano_stitcher_lateral

We attempted to use our code to stitch a set of images that were taken linearly along the same planar space. The images are in my_panos/ and are labeled gdc1.jpg .. gdc6.jpg. The result of running our stitcher against the images resulted in a a final image, panorama_lateral.png in my_panos/. Looking at the final image, it is obvious that there are some disparities. The reason for this is that because each image is taken in a linear fashion, the view points are all different. This can be further explained through the fact that our implementation of homography() assumes that each image comes from the same view point.

3 Image cut-out compositing

This function takes in a given starting row and starting column, a large image, and a smaller image which will be composited on top of the larger image. We tested this out with images of a puppy and a red bow. The test implementing this function is in pano_stitcher_test.py. The test composites books2 and boots.png (an image in the torture test). The final image is outputted into test_data/composite.png.

4 pyramidBlending()

The script, pyramidBlender.py implemented the pyramid blender. In order to accomplish this, the function took in two images that were the same size. Next, it

computed the gaussian pyramids of each image by using the *cv2.pyrDown()* function. This recomputed each input image at different resolutions and shape sizes, and stored each computation in an array. Once each guassian pyramid was calculated, the script computed the laplacian pyramid of each image. The array was ordered from largest to smallest. In each iteration, *cv2.pyrUp()* is used in order to set the size of the smaller images to be equal to the size of the larger image. The pyramid blender then computed the difference of each image and stored the results in an array. After computing the laplacian for both images, it took both laplacian arrays and stacked them horizontally, scaled each image in the array to the original size and took the sum. This resulted in the pyramid blending of two images. We tested out the script with images of the Mozilla Firefox and Safari.