Final Project: Visual Computing

Dinh Ho Gia Bao
Nguyen Trong Tung

1 Problem 1

In this problem, we are provided with pairs of RGB and Depth images where Depth image represents the depth information of the RGB image. We are asked to denoise depth images with the help of bilateral filter and the information in the color images.

Normally, bilateral filter is constructed based on two terms: the space weight and range weight. These were multiplied in a pairwise manner to represent for those signals capturing **how far pixels are relatively located** and **how different intensity of nearby pixels are specified** respectively. The formula below denotes the general computation of bilateral filter being applied at pixel p.

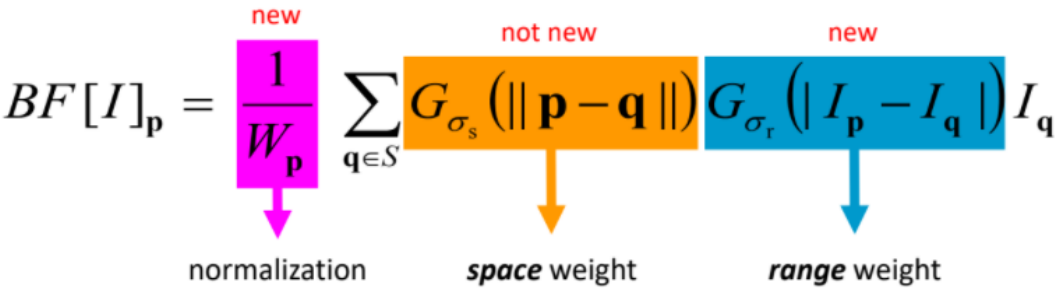

Figure 1: Bilateral Filter Computation

We followed the brute force implementation approach to compute bilateral filter in this problem. On the one hand, the space weight is normally obtained by just simply applying a Gaussian filter based on the distance. On the other hand, the range weight is calculated based on the intensity difference between the nearby pixel in color image. After all of the crucial weights were calculated, the pixel at a specific location of the new image can be achieved by applying the pixel intensity aggregation on a patch of old depth image with weights from space and range term calculated above. You can find some of the results after applying bilateral filter to denoise the depth image in our working folder.

2 Problem 2

2.1 Problem 2.1

The important map(saliency map) of an image can be easily implemented with the approach described in Image Resizing Lecture. This is achieved by summing intensity difference of nearby pixel(vertical and horizontal axis) for the current pixel being considered

2.2 Problem 2.2, 2.3

To perform seam carving on images, we define two maps: saliency map, and important map

For important map, it can be obtained from Problem 2.1 to find optimal vertical and horizontal seams to remove from the image. The idea is to sequentially remove unimportant columns and rows from the image to resize it. In order to do this, we first construct a dynamic programming solution for the problem of finding minimum path starting from each of the pixel on the first row(or column) of image. This can be referred to **FindingLeastImportantSeam** step. After defining the minimum cost path for each of starting pixel on the first column(column), we start to trace along the path from top to bottom(or from left to right) and sequentially remove those pixels out of our image. As a result, the image will be resized to the expected shape without distorting objects inside. Below are some of the results for our algorithm implementation.

On the other hand, Frequency Tuned Saliency Detection was extracted based on the code provided in lab tutorial. Then it is utilized in the same way as important map in previous problem in order to perform the seam carving algorithm

Examples in figures [2, 3, 4, 5, 6, 7] below show the results with the help of **Important Map** and **Frequency Tuned Saliency map** to perform seam carving by removing **30 least important columns and rows**.

2.3 Problem 2.4

Let's run the above algorithm setting in Test Image 3 to report the overall processing time. Moreover, we also give some of the comparisons and observations for the two importance maps being utilized in this problem.

In this setting, we set the sigma value as 5 and kernel size as 10×10 to calculate saliency map. In the important map, there are no parameters to be specified. With this setting, the time record for seam carving perform on important map is **16.9788823 seconds**, and **14.962775 seconds** for the saliency map. In this context, important map is slightly faster than saliency map. However, this can not

Figure 2: Test Image 1

be generally concluded that computation of saliency map is faster than important map. There are two main reasons: no parameters are specified in important map and the time complexity of important map calculation is $O(n^2)$, the parameters sigma and kernel size of saliency map really affect the running time. If the kernel size is small, the Gaussian filter has to iteratively slide through more pixel values than bigger kernel size which results in situation that saliency map takes more time to compute than important map.

Based on the visualized results of two aforementioned maps, they seems to produce fine results after performing seam carving. However, the map from saliency seems to produce better heat map for the image since it is based on the Gaussian filter to smooth the image. The map from important map is peaky since it only captures pixel lying near the edges, or the boundaries defining the object. Therefore, it is preferred to use saliency map instead of important map in many cases.

For the good cases of seam carving algorithm, it can be showed in results of Test Image 2, and Test Image 3. Also, we show some of the bad cases(Figure 9,

Figure 3: Importance map calculated on the left and Resized version of Test Image 1

Figure 4: Saliency map calculated on the left and Resized version of Test Image 1

10) for running the seam carving algorithm. It can be seen that if we remove too many rows, or columns when applying seam carving. The image would result in loosing some parts of object.

3 Team Contribution

Gia-Bao, and Trong-Tung equally contributed to this project. In detail, Gia-Bao is responsible for implementing the code as well as generating results. Trong-Tung worked on building general workflow for the code, and writing documentation.

Figure 5: Test Image 2

Figure 6: Importance map calculated on the left and Resized version of Test Image 2

Figure 7: Saliency map calculated on the left and Resized version of Test Image 2

Figure 8: Test Image 3

Figure 9: Seam Carving results when removing 130 columns on Test Image 3

Figure 10: Seam Carving results when removing 70 columns on Test Image 3