Programming Assignment 1: Depth-from-focus

0. Introduction

In computer vision class, I got programming assignment 1 "Depth from focus". This assignment is the implementation task of "Depth from Focus with Your Mobile Phone"[1]. The goal of this assignment is to get the all-in-focus image and clear depth map from the given dataset, which is a set of pictures captured about the same scene with a different focus.

The assignment consists of five tasks - Aligning image, Initial depth estimation, Apply Graph-Cut algorithm, All-in-focus stitching, Refining depth map. I will explain each step of the assignment in the following sections and show the results of the implemented code. I use MATLAB 2020b and several toolboxes-IAT, gco-v3.0, and WMF toolbox for implementing code.

1. Image aligning

```
…

[d1, l1]=iat_surf(img1); %find keypoint of image1

[d2, l2]=iat_surf(img2); % find keypoint of image2

…
%apply RANSAC algorithm and find warping network
[inliers, ransacWarp]=iat_ransac( X2h, X1h,'affine','tol',.05, 'maxInvalidCount', 10);
…
%inversely warping image2(align to image1)
[wimage, support] = iat_inverse_warping(img, ransacWarp, 'affine', 1:N, 1:M);
```

Code 1 Main part of image aligning code

Pictures of the given dataset have misalignment, maybe that is caused by various factors. So we need to remove misalignment for getting a good result in the following step. I used IAT(Image Alignment Toolbox) for aligning the image. I used the feature-based alignment method, use the SURF algorithm, which is similar to the SIFT algorithm, for feature matching. After find features in two images, and matching corresponding features. But, outlier can cause a severe error at image transformation, I applied the RANSAC algorithm to remove outliers.

---

[1] Suwajanakorn, S., Hernandez, C., & Seitz, S. M. (2015). Depth from focus with your mobile phone. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3497-3506).

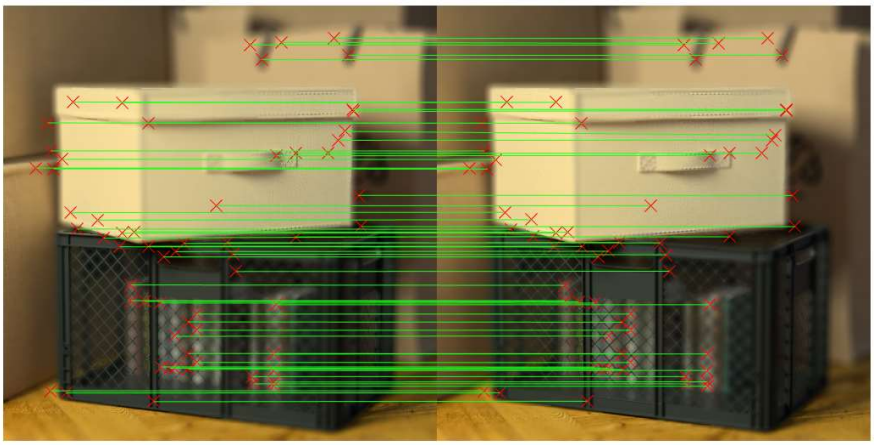

Figure 1 Matched features in two misaligned images(after RANSAC process)

Finally, I could get a warping matrix which represents the transformation of image1 to image2. So I could warp image2 to image1 by using a warping matrix to align two images.

Figure 2 Error map about misaligned frame (frame 5 and 6 in 'boxes' dataset)

Figure 3 Error map about aligned frame (same as Figure 1)

Like the above 2 figures, I could successfully align paired frames(frame t and $t+1$) in the dataset. The additional result will append in Appendix1 and the experiment results in the dataset.

2. Getting Initial Depth Map

Generally, well-focused regions in the photo are more 'sharp' than poor-focused region. So we can get the focused region at given data by finding sharp parts in the image, this is the edge of the image. Based on this fact, we can get well-focused regions in the image by edge filtering.

At the sharp region(edge), the pixel's intensity changes rapidly, but the blurry region is not. We can find an edge through this tendency to change. Especially, 2nd derivative of intensity represents tendency well, we ordinary use 2nd derivative kernel to find well-focused regions. These 2nd derivative-like kernels are called 'focus measure', and there are many kinds of kernels. In this assignment, I used the Laplacian kernel to find well-focused regions.

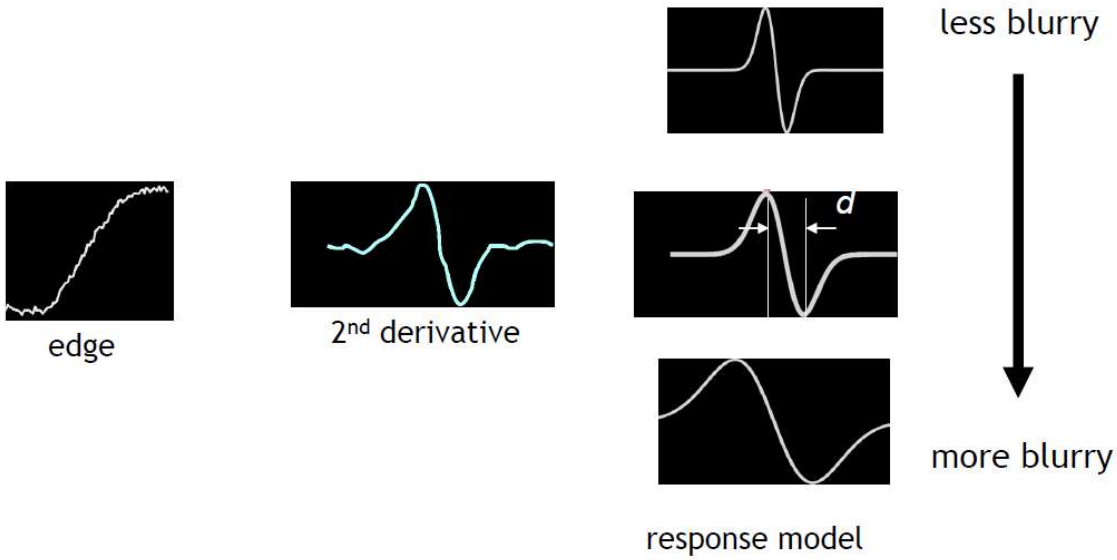

Figure 4 Response of convolution edge term and 2nd derivative by blurry levels

In actual implementation, I converted each RGB image to a gray image(intensity map) using by '**rgb2gray()**' function in MATLAB. After that, I convolved the Laplacian kernel to each intensity map and could create cost volume by collecting results. Finally, I could get an initial depth map to apply the winner-take-all manner to cost volume because the sharp region has a high activation value about the Laplacian convolution filter. To take the maximum value in cost volume, about each pixel location, I used '**max()**' function in MATLAB, apply to cost volume.

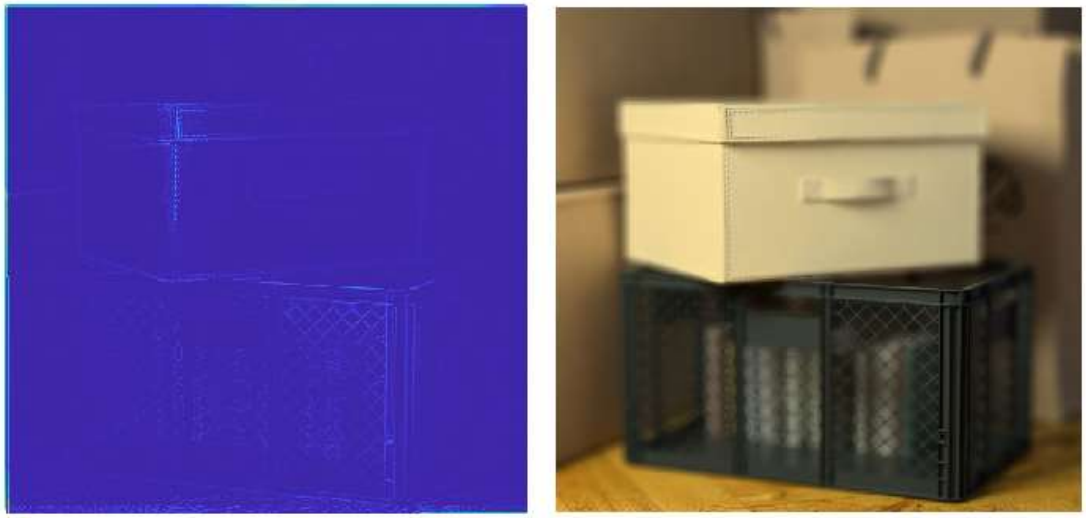

Figure 5 (A): Disparity map in cost volume,

(B): Original image corresponding to (A)

The above figure is gained the disparity map and the original image. As I said before, well-focused regions in the original image have high activation in the corresponding region in the disparity map. In the cost volume, there are disparity maps like above, and all of them show high activation in different regions. So, we could make an initial depth map, by aggregating pixels which have high activation value.

Figure 6 Initial depth map

3. Apply Graph-Cut method (max-flow/min-cut algorithm)

The Graph-cut method is graph dividing method, ordinarily used to segment region in given image data. To implement graph-cut, we can use the max-flow/min-cut algorithm. 'flow' of the max-flow algorithm means weights of edges of the graph. Simply, this algorithm maximizes the summation of weights of edges. This is the reason for max-flow is equal to min-cut because we should cut edge which has a minimum value to maximize flow. In practice, we use an energy function for the global optimization of this problem. (I think the energy function works like the 'regularizer' in the ML field.)

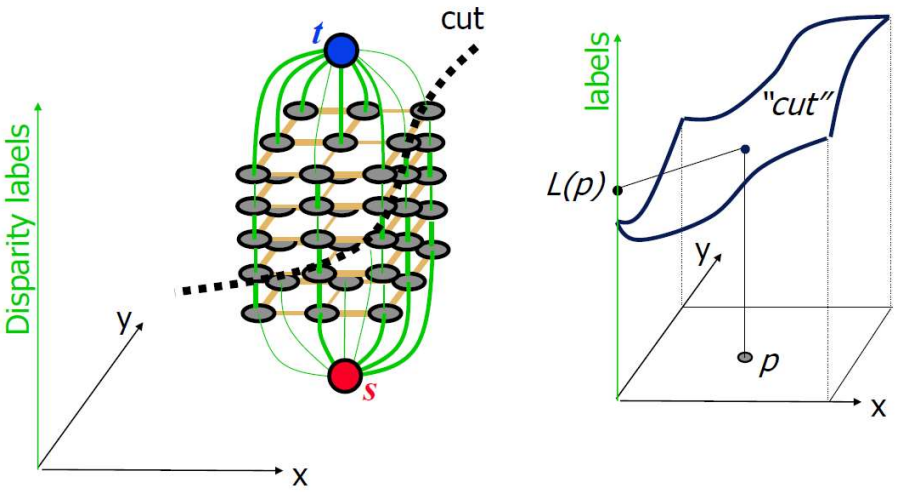

Figure 7 Max-flow/Min-cut algorithm about given 2d data.

$$\mathrm{E(L)} = \sum_{p \in P} D_p(L_p) + \sum_{(p,q) \in N} V_{p,q}(L_p, L_q)$$

Equation 1 Energy function of graph-cut method

In above equation, D_p is data penalty function, $V_{p,q}$ is interaction potential.

Even if energy functions are used, the min-cut algorithm's basic philosophy is the same. So I could apply this algorithm to the cost volume obtained from Chapter2. In general, we distinguish the boundary of the meaningful region in the image by the difference in value between a pixel and around pixels. If there exists a large difference, we can distinguish the boundary of the region. And, in order to perform a given task, we must determine the area of the result map along the pixel having the larger value in the frame of each cost volume.

Based on this fact, I thought I should apply the 'max-cut' algorithm to cost volume. 'max-cut' algorithm is simply the opposite of the min-cut algorithm. So, I modified it to have a negative value by multiplying -1 before applying graph-cut to the cost volume. The modified input has a small difference value in pixel value at the edge regions because the absolute value is the same as the

original cost volume, but the modified input has the opposite sign.

To put data, I reshaped 3D (modified) cost volume to 2D input data, by changing each 2D disparity maps to 1D array. Data label is an index of each disparity map in cost volume. To reshape the cost volume, I used '**reshape()**' function in MATLAB. And I utilized Multi-label optimization toolbox[2] to apply the graph-cut algorithm.

$$\mathrm{E(I)} = \sum_{p \in P} D(p, d_p) + \sum_{p,q \in N} K_{p,q} \cdot T(d_p \neq d_q)$$

Equation 2 Potts energy[3]

When we use the graph-cut method, how to model the function V is also an issue. In energy function, a term which is the sum of V is also called the smoothness term. I choose the 'Potts' model for the smoothness term. In the above equation, the smoothness term(second term in eq.) is consist of penalty term $K_{p,q}$ and $T()$. Function T has value 1 if condition in parentheses and has 0 otherwise. So, the Potts model assigns the same weight regardless of each label except itself.

When implementing code, we can specify smoothness term by function '**GCO_SetSmoothCost()**' contained in graph-cut toolbox[2]. We can apply the Potts model to the smoothness term, by putting in a specific matrix to the above function. The input matrix has elements, which value is 1 except the diagonal term, it has value 0.

---

[2] Provided in "https://vision.cs.uwaterloo.ca/code/"

[3] Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, *26*(9), 1124-1137.

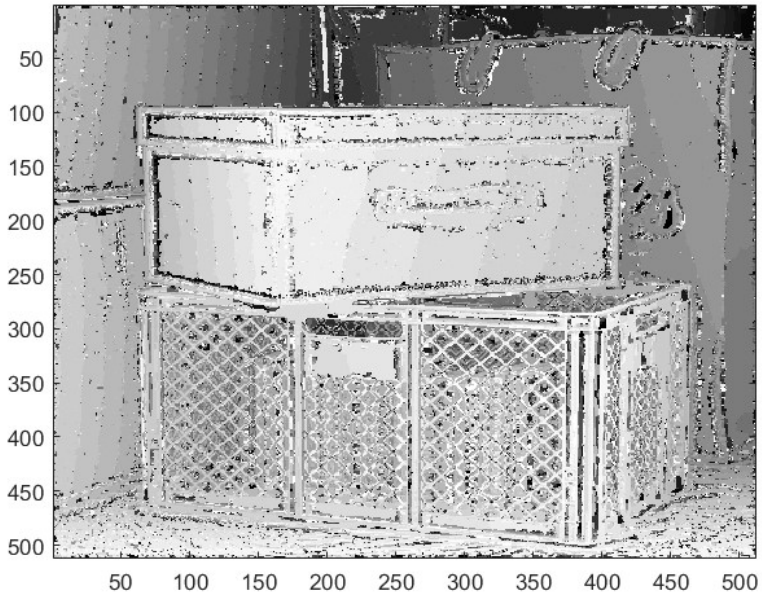

Figure 8 Result depth map given by Graph-Cut method

At the result of the graph cut method, each pixel value represents the label of data. In detail, it means that the pixel at the corresponding location in the corresponding frame about pixel value is the best solution. 'Corresponding frame' is a 2D disparity map which has a corresponding index number with a pixel value of result map of the graph cut method.

4. Final refining depth map (apply weighted median filtering)

I could get a good depth map like the above depth map(Figure 8), but this is imperfect. In the depth map, there is the noise that distinguishes it from the surrounding label. These noises are too white or black and located around object edges or boundaries of depth labels. To get the best result, we should refine the given depth map. There are various methods for refining the depth map, and the most famous of them is the filtering method.

We can use various filters for the enhancement of the image. For example, Gaussian filters are very effective in removing random noise. Like the previous example, we can choose an appropriate filter for a refining depth map. In PA1, I used weighted median filter for the image, proposed in "Constant Time Weighted Median Filtering for Stereo Matching and Beyond".

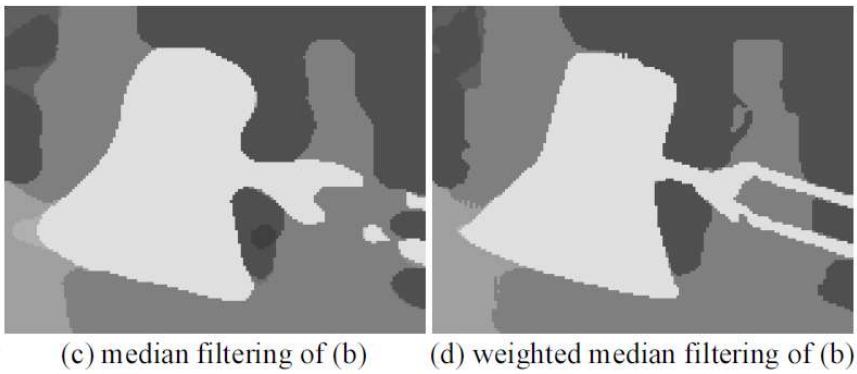

Figure 9 Compare refined depth map given by (a) median and (b) weighted median[4]

The weighted median filter is different from the original median filter. The original median filter selects a median value in the histogram of the value of pixels in the local window. But, the weighted median filter multiplies the bilateral weight obtained based on the central pixel and nearby pixel values when consisting histogram of the pixel value. By reference paper[5], the bilateral weight suppresses the effect of the pixel which has a different value from the center pixel. So, the weighted median filter can preserve the sharpness of the original input depth map, unlike the original median filter.

I use weighted median codeset served in http://kaiminghe.com/. To use the weighted median filter function, I should put in several parameters-disparity map, guidance image, vector of disparities, kernel size, and epsilon. The first three inputs are determined by the earlier step[6], o I should determine the last two parameters, kernel size and epsilon. I set according to the queue proposed in the original paper[57]. The result map is below image.

---

[4] Image from original paper[5]

[5] Ma, Z., He, K., Wei, Y., Sun, J., & Wu, E. (2013). Constant time weighted median filtering for stereo matching and beyond. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 49-56).

[6] (i) depth map (result of Graph-Cut), (ii) all-in-focus image(from cost volume), (iii) 1:30 vector(30 labels)

[7] Radius : ceil(max(512,512) / 40), epsilon : 10^{-4}

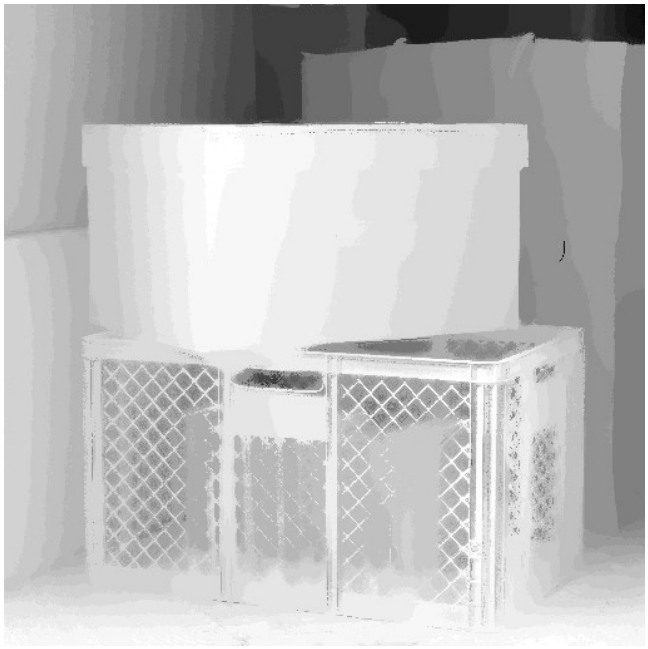

Figure 10 Result map given by weighted median filtering

Finally, I could get pretty good depth map about given dataset.

5. all-in-focus image

Using the obtained depth map, we can get all-in-focus image. So, we can get all-in-focus image from both depth map-initial depth map and refined depth map. The way to make it is simple.

(i) Initial depth map

When I make an initial depth map, I take pixel in disparity map in cost volume. I implemented a function to make a depth map from cost volume. For a specific pixel location, this function checks all frames in the cost volume to get the index value of the frame with a pixel with maximum activation value, and gets the color pixel value from the color image correspond with that frame. Repeat this process for all 512*512 pixels.

Figure 11 All-in-focus image from initial depth map

After get this all-in-focus image, I used this image to refine depth map. When I use weighted median filter, I need guidance image, I put in this all-in-focus image with other data.

(ii) Refined depth map

Figure 12 All-in-focus image from refined depth map

I could get all-in-focus image from the initial depth map, but I thought it's not the best result. So, I tried to get all-in-focus image from a refined depth map.

The pixel value of a refined depth map is an index of the label(frame) in cost volume, we can make all-in-focus image by this value. To get color pixel value at a specific pixel location, I took the corresponding pixel value in the color image. This color image corresponds to the frame in the cost volume, which is specified by the index specified by the pixel value in the refined depth map. Finally, I got a better all-in-focus image compare with the result using the initial depth map.

A. Appendix 1: result about "cotton" dataset

In the previous text, all of them were described with experimental results for the "boxes" dataset. However, the given dataset was a total of two - "boxes" and "cotton" dataset. So I would like to show you the experimental results of the cotton dataset that I had not shown above. The first three main results, the first depth map, the graph-cut result image, and finally the refined depth map obtained from the weighted median filtering result. Additionally, I would like to show all-in-focus images based on the depth map.

(i) Initial depth map

Figure 13 Initial depth map about "cotton" dataset

(ii) Graph-Cut result

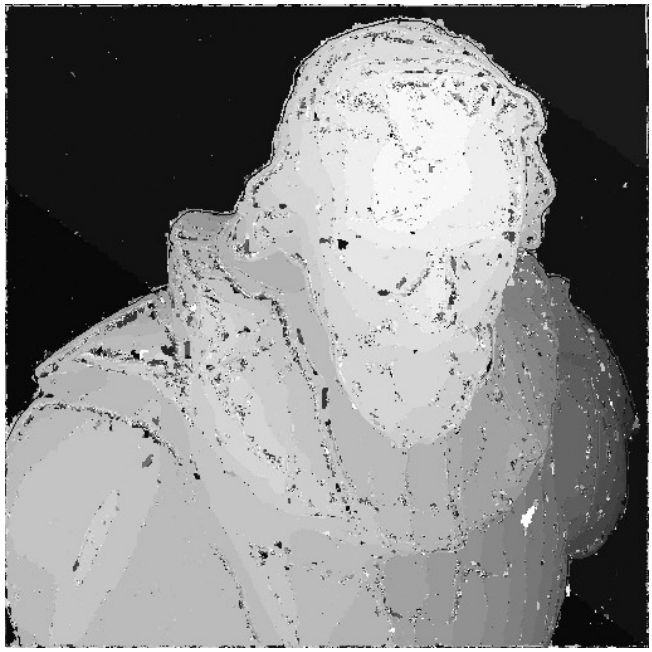

Figure 14 Result of Graph-Cut method

(iii) Refined depth map

Figure 15 Final depth map obtained by WMF

(iv) All-in-focus image

(a) (b)

Figure 16 All-in-focus image (a) from cost volume (b) from refined depth map

Looking at the image above, you can see that the quality of the two all-in-focus images is quite different. The image based on the cost volume is blurry, I guess it looks blurry because it is not perfectly aligned in the process of aligning the images of the dataset.

In comparison, the all-in-focus image using the refined depth map seems to have been able to achieve better results, since it is not affected by the factors mentioned above. If you look closely, you can see the detail preserved in the image on the right.

B. Appendix 2: Description

The attached codes have codes that I directly imputed to carry out this task. In this section, the codes for each task are briefly described.

(i) Image aligning

At first, for aligning two images, I implemented the "alignfeature" function referred to the tutorial of IAT toolbox example. Using this function, I was able to successfully align all images. Align result images were saved separately by creating a folder called "alignimage" in the dataset. Additionally, the error map between the misaligned images and the error map between the aliased images were also saved separately. Each folder was assigned the names "original_error" and "align_error".

(ii) Getting initial depth map

At this stage, the function "gen_initial_depthmap" was implemented to obtain the initial depth map and cost volume. To apply the focus measure, I implemented the "applyLaplacian" function

which convolves the Laplacian kernel with intensity image. I implemented a "focus_measure_all_images" function to create a cost volume by collecting images obtained from the convolution result. Through these functions, I could get the initial depth map and cost volume successfully.

(iii) Apply Graph-Cut method

I applied the graph-cut function which is implemented in multi-label optimization toolbox(gco-v3.0). To apply the graph-cut method, I implemented the function "apply_graphcut" in the toolbox folder. This function modifies cost volume to negative scale and 2D flat grid before applying graph-cut. The resulting map is "GC_result_depthmap.png" in the dataset folder to be saved.

(iv) Apply WMF

To use the WMF, I only set two parameters and apply the implemented function. I implemented the "make_refined_depth" function in the WMF toolbox folder, to apply the WMF to the depth map. As mentioned earlier, I decided on parameters as in the original paper. The resulting image is "WMF_refined_depthmap.png" in the dataset folder to be saved.

(v) Make all-in-focus image

I implemented "gen_initial_all_in_focus" for getting all-in-focus image from cost volume, which is the result of task 2. And, I implemented "gen_all_in_focus" for getting all-in-focus image from a refined depth map. Both functions were implemented in the same way as described earlier. Additionally, the result images are each "init_all_in_focus.png" and "Final_all_in_focus.png" to be stored in the dataset folder.

(vi) Script for whole process

To more easily output the results, a script was created that easily stores the results of each task by simply changing the dataset name. It's named "FinalScript". If you specify 'Dataset_name' variable either "boxes" or "cotton", and execute the script, you will see result images in the corresponding dataset folder.