Assignment1 for COMP776, ”Computer Vision”, Fall 2015

Dong Nie

1) Bayer Pattern Problem

The problem gives us a mosaicing image, and requires us to demosaicing using linear interpolation. The key point is to set a filter for each channel.

After considering the characteristics of Bayer Pattern, the filters I set for each channel (R,G,B) is:

$$R:\begin{matrix}1/4 & 1/2 & 1/4\\ 1/2 & 1.0 & 1/2\\ 1/4 & 1/2 & 1/4\end{matrix} \quad G:\begin{matrix}0 & 1/4 & 0\\ 1/4 & 1.0 & 1/4\\ 0 & 1/4 & 0\end{matrix} \quad B:\begin{matrix}1/4 & 1/2 & 1/4\\ 1/2 & 1.0 & 1/2\\ 1/4 & 1/2 & 1/4\end{matrix}$$

The original input image (mosaicing) is listed in Fig.1.

Fig.1 mosaicing image

After linear interpolation with the bayer filters, we get the following colorful demosaicing image in Fig.2.

Fig.2 The reconstructed image: demosaicing image

Obviously, the edge part is not well reconstructed, it is because the convolution filter is incomplete when it comes to the edge of the image. I try to use the ‘symmetric’ parameter in imfilter function, the average

per-pixel error become smaller. And the reconstructed image is in Fig.3.

Fig.3 The reconstructed image using ‘symmetric’ parameter in imfilter

The summed squared difference between the demosaicing image and the original colorful image is computed, and then visualize the difference map in Fig.4.

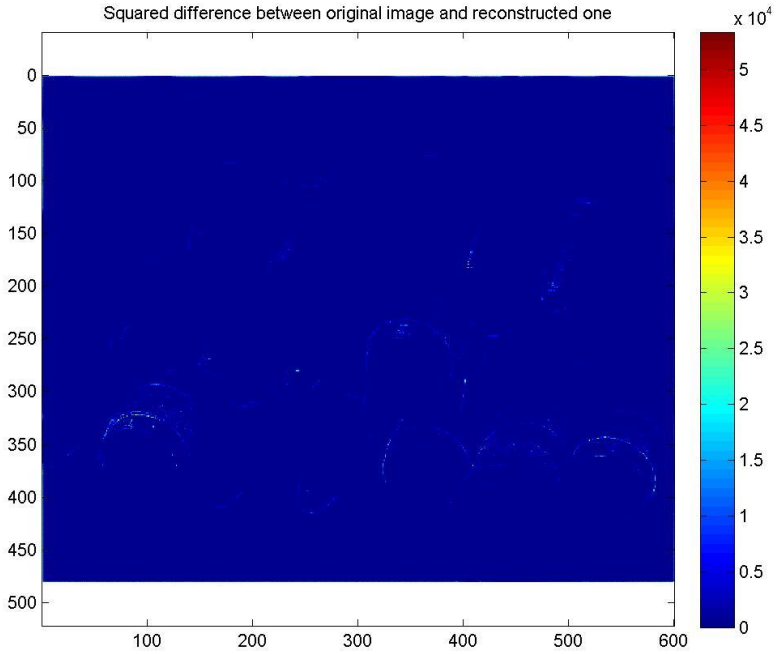

Fig.4 Summed squared difference map

Correspondingly, the average per-pixel error is 131.98, and the maximum per-pixel errors are 53378.0.

At last, I’ll show a close-up patch in which reconstruction is worst performed, and it is presented in Fig.5. In fact, we can know something from observing the summed squared difference map, the per-pixel reconstruction error will be larger in the intersection area for different colors (original colorful image), so I make a judgement that the linear interpolation performs badly in areas where pixel variances is high. And this can be understood easily: linear interpolation is not good at fitting high variance model.

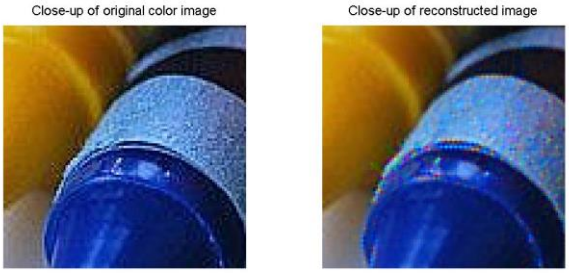

Fig.5 The close-up patch comparison (the size of patch is 50*50), this area has high variance, so this patch has large per-pixel error

2) Image Alignment Problem

This problem gives us three parts for one image, and each represents a color channel. We should align the three parts, and combine them to form a colorful image.

To solve this problem, I firstly extract three parts from the given image and take them as B,G,R channels respectively. Secondly, I use NCC to align the three parts. At last, combine three channels together. I'll talk about the second step in detail.

I do several experiments to determine the order of the channel layers. I found that green channel in the bottom layer would give a best performance. As a result, green channel (G) is employed as the fixed layer.

The following are detailed key points when I do the image aligning with NCC algorithm.

At first, I cut off the white and black edges before computing the correlation. Because the black and white borders of the original input image will affect the computing of SSD and NCC and may cause error in the alignment.

Secondly, I add pads to border of the fixed layer (G) to use NCC method, and fill in these pads with its average gray value so that the normalization will make sense and the margin area will not affect the final result.

At last, only keep the areas which are covered by all three channels, which means I drop border areas which are not fully covered by all three color channels.

Then I get the following results:

Displacement vector: B(-5, -2), R(4, -1)

Displacement vector: B(-4, -2), R(5, 0)

Displacement vector: B(-7, -3), R(7, 2)

Displacement vector: B(-4, -1), R(9,

Displacement vector: B(-5, -3), R(6, 1)

Displacement vector: B(0, 0), R(5, 1)

Fig.6 The results for image alignment

For this task, the displacements window is set at 15 pixels. The time cost for each image alignment is about 0.55 seconds on intel i-5 processor and 8GB memory laptop.

Multiscale alignment (Bonus part)

For the high resolution images, if we still use the method described above, the time cost will be very large. Thus, multiscale alignment is proposed to solve this situation. The big picture for this method: reduce image, and use reduced images to find displacement vector and then use full images to refine the

displacements. Totally speaking, the basic idea is recursively processing the large scale image to a smaller one.

Different from the single alignment, the displacement window should be set much smaller, otherwise the range will be too large to find the correct displacement for those small images on top of the image pyramid.

The same tricks used in the above part are also adopted in this experiment for image extraction and alignment, such as white and black border cut and normalization and so on. Experimental results are presented as follows.

Displacement vector: B(-24, -20), R(47, 14)

Displacement vector: B(-55, -9), R(63, 3)

Displacement vector: B(-72, -39), R(77, 24)
Fig.7 Multiscale alignment for high resolution images

For this task, I once again take the green channel (G) as the fixed layer, and the displacement window is finally set at 3 (I have considered the time cost factor, when the window size is set at 4, the time cost will be 35 seconds). The time cost for each image is about 21.5 seconds on the same laptop. This method is a big step to reduce the time cost.