3-D Reconstruction

Computer Vision Challenge

Team member:

Weiqi Luo

Wanting Huang

Yuanjing Liu

Yanglu Zeng

Content

[**1.** **Challenge Introduction** 3](#_Toc524103097)

[**2.** **Image Rectification** 3](#_Toc524103098)

[**2.1** **Feature Correspondence** 4](#_Toc524103099)

[**2.2** **Estimate Fundamental Matrix** 4](#_Toc524103100)

[**2.3** **Epipolar Rectification** 5](#_Toc524103101)

[**3. Disparity map** 6](#_Toc524103102)

[**3.1 Depth perception via triangulation** 6](#_Toc524103103)

[**3.2 Semi-globel block matching** 6](#_Toc524103104)

[**3.3 Implementation and discussion** 8](#_Toc524103105)

[**4. Depth image based rendering** 8](#_Toc524103106)

[**4.1 Rendering model** 8](#_Toc524103107)

[**4.2 Post-processing** 9](#_Toc524103108)

[**4.3 Implementation and discussion** 9](#_Toc524103109)

[**5. Improvement** 10](#_Toc524103110)

[**5.1 Time cost** 10](#_Toc524103111)

[**5.2 Quality** 10](#_Toc524103112)

[**5.3 Graphic interface--GUI** 10](#_Toc524103113)

[**References** 12](#_Toc524103114)

1. **Challenge Introduction**

Given two stereo images of a desk with groceries, we are supposed to find an image from any angle between those of the two given ones. Using parameter p with values from 0 to 1 to represent angles from left view image to right view image.

We handle the task as three parts. First part is pre-processing the raw images, including feature selection and correspondence, fundamental matrix deriving and epipolar rectification. Next part is calculating the disparity map, implementing xxxxxxxx. The last part is to construct the new image from new angle, xxxxxxxxxxx.

Epipolar rectification

DIBR

disparity map

dense matching

F matrix deriving

Feature selection

Fig. 1.1 Flow diagram

1. **Image Rectification**

The file free\_viewpoint.m describes the whole process of this part. To begin with, we get two raw pictures as out input, from here we need to load and analyze them and thus deriving the essential matrix as well as the fundamental matrix.

* 1. **Feature Correspondence**

In this part, we process the two images using several function files, namely rgb\_to\_gray, harris\_detektor, korrespondenzen and RanSaC. Therefore, we get several pairs of correspondent points, which prepare us to calculate the essential matrix and fundamental matrix. The detailed explanations of functions are as follows.

rgb\_to\_gray .m: convert the colorful image to one gray image, using

0.299\*R+ 0.587\*G+ 0.114\*B.

harris\_detektor .m: we can get corner points so that the matching complexity can be reduced by calculating the harris matrix using

H= det(G) – k\*tr2(G).

korrespondenzen .m: according to NCC criterium to match the found point pairs.

RanSaC .m: use RanSaC algorithm to find and avoid incorrect points, as an improvement.

* 1. **Estimate Fundamental Matrix**

Eight points algorithm is implemented here to derive the fundamental matrix from the points we have got.

In which E is the essential matrix and A is the Kronecker product of two corresponding points in a pair, namely:

After implementing 8-point pairs into it, we can get E, therefore also F.

Seeing in achtpunktalgorithmus.m.

* 1. **Epipolar Rectification**

After rectification, the epipolar lines between two pictures will be horizontal, which means the corresponding points will be of the same y value, which apparently makes finding the disparity map easier and more precisely. Seeing the file epipolar\_rectification.m and image\_rectification.m.

**3. Disparity map**

**3.1 Depth perception via triangulation**

The corresponding points are then constrained on the same image scanline after image rectification.

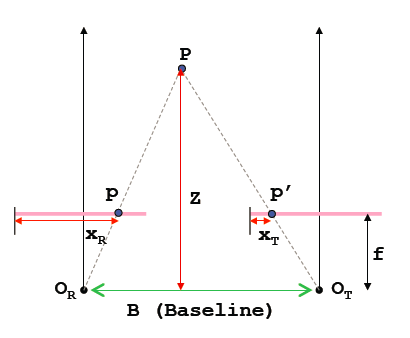


Fig. 3.1 Triangulation

By considering the similar triangles and shown in figure 3.1, we get

Which turns out to be

In which d is disparity, the difference between the x coordinate of two corresponding points. The disparity is higher for points closer to the camera.

**3.2 Semi-globel block matching**

In order to create disparity map, we have to find correspondence for each pixel in the image, after research, our project takes the method of semi-globel block matching (SGBM) for dense reconstruction, which achieves better result than normal blocking method at the cost of consuming more time.

Compared with other method, this method additionally uses a smoothness constraint that penalizes discontinuities. This is typically formulated in a cost function with three parts:

The first part is the sum of all pixel matching costs for the disparities of D. the second part penalizes all pixels q in the neighborhood for p, for which the disparity changes a little bit. The third part adds a larger penalty for all larger disparity changes. The problem then becomes the minimization of energy , which is NP complete. This can be solved by aggregating matching costs in 1D from all directions equally (considering the time cost, our realization uses only 8 directions, which is illustrated in figure 3.2 ). The aggregated cost is calculated by summing up the costs of the 8 directions or paths.

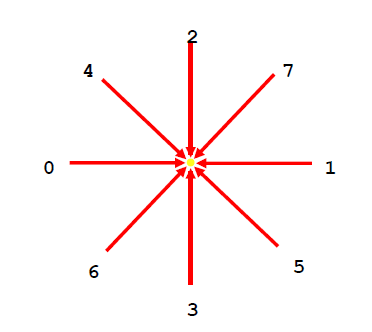


Fig. 3.2 Illustration of 8 directions

Then the disparity d for each pixel p is selected such that the cost is minimized.

**3.3 Implementation and discussion**

This part is realized in file Cal\_disparity.m, which refers the code of Heiko Hirschmueller, who proposed the method of SGBM.

As is shown in the figure 3.3 (disparity map of L1 and R1) and figure 3.4 (disparity map of L2 and R2), the result of the background (i.e. the wall) is disappointed, which is caused by the low texture of the wall. Therefore, the background is segmented in order to get a better result. This will be introduced in chapter 5.2 in details.

Fig. 3.3 Disparity map of L1 and R1

Fig. 3.4 Disparity map of L2 and R2

**4. Depth image based rendering**

With the depth information, our project creates the intermediate view with the help of DIBR technique (Depth image based rendering), which is a classical virtual view synthesis technique. Our realization relies on paper [Free-viewpoint depth image based rendering], which can fill the holes and remove the most of the contour artifacts of the reconstructed image.

**4.1 Rendering model**

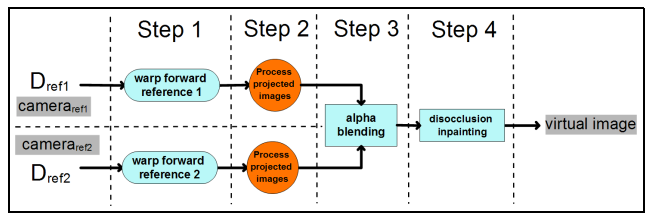


Fig. 4.1 Standard process of rendering

The standard process of rendering is to warp the depth map of reference image and then calculate the world coordinate of pixels in images, and re-projected onto the virtual viewpoint. The target position of virtual viewpoint can be found with rotation (R) and translation(T) parameter and blending the re-projected image pair with weighted parameter(p) can find the target virtual image. However, in our project, we skip the 3D wrap step and find the target virtual image with the help of one depth map and one reference image.

**4.2 Post-processing**

Gaps and holes occur when the previously invisible scene points in the new view point can not be rendered. Which is also caused by the discontinuity of the disparity map.

In addition, contour artifacts may also occur. Contour artifact refers to textures that are rendered to wrong places in new generated view, which happens as a consequence of the inaccuracy of the depth map, this occurs especially at the edges of objects, in which the foreground textures at the edges are rendered to the background.

Paper [Free-viewpoint depth image based rendering] proposed a method to remove the contours by dilating the dis-occlusions. Firstly the holes are filled with Background pixel extrapolation and dis-occlusions are filled in the opposite direction to rendering, then the filled occlusions are smoothed.

**4.3 Implementation and discussion**

This part is realized in file cd2lr2.m with sub functions in file find\_shiftMC3.m, rgb2yuv.m, and yuv2rgb.m. The detailed explanations of functions are as follows.

cd2lr2.m: render the virtual view point, fill the holes and gaps and remove the contour artifacts.

rgb2yuv.m: convert the RGB image to YUV format for better processing result. (Human eyes are less sensitive to chrominance part than to luminance)

yuv2rgb.m: convert back to the original RGB image color space.

The rendering result is displayed in figure 4.2, 4.3,

Fig. 4.

**5. Improvement**

Our project has done some improvements based on fundamental functions with respect to the following three aspects.

**5.1 Time cost**

Considering that the time cost is proportional to the square of the size of the image, we down sample the image to a smaller size for the seek of accelerating the program. After processing, the image is then interpolated to the original size. Though the result is a little bit worse, we gain a lot of time.

**5.2 Quality**

Following the chapter 3, we discussed in details the segmentation process, we segmented the wall out of the original image and then calculate the disparity map for the foreground objects, experiment shows that this step greatly improves the quality of the disparity map and also reduce the processing time.

The implementation of segmentation combines the K-MeansClustering function in matlab library and RanSac, which achieves satisfactory result, shown in figure…,

Fig. 5.

**5.3 Graphic interface--GUI**

In order to better manipulate our program, we create an interface, which is shown in figure …………………..

Our program consists mainly of 3 steps,

Fig. 5.

**References**