Computer Vision Challenge Report

Group 01
Xingcheng Zhou, Zongyue Li, Jiapeng Zheng
July 2019

1 Introduction

The main task of this computer vision challenge is to calculate the disparity map. Disparity refers to the difference in image location of an object taken by left and right cameras. We used several methods to calculate the disparity map and compared them with each other. Meanwhile we calculated the Euclidean transformation parameters between two cameras by RanSaC method. Beyond that we made 3D reconstruction to recovery the object in three dimensions. At last we wrote several test functions and GUI to perfect our challenge code.

2 Disparity Calculation

1. Basic idea

We tried several methods in this part, after taking our computer calculation speed and running time into consideration, we decide to use local matching approach with BM algorithm, SAD+CENSUS cost combined with median filter.

The general idea of calculating disparity map is that, we compare two input images, the left view and right view of stereo camera pictures, and find the most similar pixel in right view picture for each pixel in left view picture, then calculate the difference of their positions, and put its value into corresponding disparity matrix entry. The main problem is, how to tell that two pixels are similar and point to a same place.

To solve this problem, we use a cost function

$$cost(r,c) = \alpha * (1 - \exp(-cost_{sad}(r,c)) + (1-\alpha) * (1 - \exp(-cost_{census}(r,c)))$$

where the cost contains two part, the first part calculates the sum absolute difference of two windows.

$$\mathrm{cost_{sad}}(r,c) = \sum_{(r\prime,c\prime)\in win(r,c)} \mathrm{abs(left(r',c') - right(r',c'))}$$

The second part uses census transform to calculate its census cost, with census transform, we can get the general relationship between the center pixel and the pixels around it.
For example, for a 3*3 window, we compare each pixel with all its 8-connected neighbors.

$$\zeta(x,x') = \begin{cases} 0, & x > x' \\ 1, & x \leq x' \end{cases}$$

Then we transfer a 3*3 matrix to 8-bit value, and use a XOR gate to calculate the difference between two windows, sum up the number of 1, which is exactly the second part of cost function.

After getting the cost of one pixel in left view picture and its corresponding possible matching pixels within reasonable search range in right view picture, we view the pixel with lowest cost in right view picture as its match and get its disparity according to equation.

$$\mathrm{d_p} = \mathrm{arg}_{d \in (\mathrm{d_{min}}, \mathrm{d_{max}})}^{\;min} \mathrm{cost}(\mathrm{p}, \overline{\mathrm{p_d}})$$

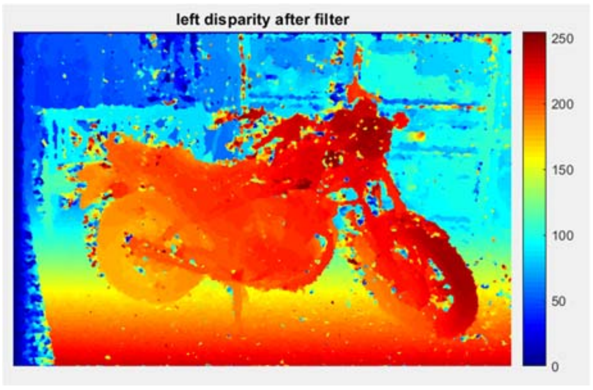
By calculating the disparity, we face a problem that, if we want to traverse all pixels of a large picture, such as motorcycle with 1988*2964 resolution, it will take rather long time. To deal with this problem, we use a gap in our code, it means we calculate the cost every gap+1 pixels and give them same disparity in this area. It has similar effect to subsampling, but when searching possible matching point in another picture, this method is more likely and stable to find the correct match because when we subsample the original picture, its correct matching pixel may be subsampled and we could only get another similar but wrong matching point.

For different images, we choose the size window according to its image size. For given motorcycle picture, we use window size=27 whereas for smaller pictures like playground, we use only window size=15, and after comparing the effect of two parts we set α as 0.4 which means that the census part weighs a little bit more than sad part.

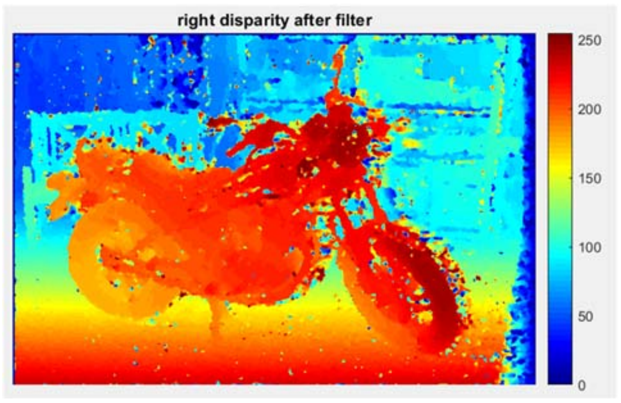
After computing the disparity map, we use median filter as post processing to reduce the noise in the picture. Some of the noises are generated because of the image occlusion and some other come from mismatching. With this process, we can make the picture more smoothly which is close to what the real disparity should look like. For large image such as motorcycle pictures, we use filter size as 21 while for small images we use size as 3.

2. Result and analysis

According to our codes, we can get disparity maps of two scenes at the same time. This is useful for checking left right consistence.

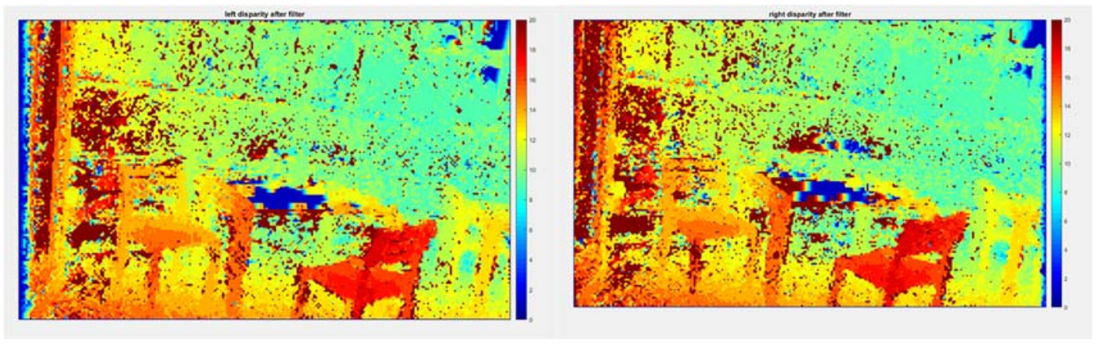


Left Disparity Map of Motorcycle



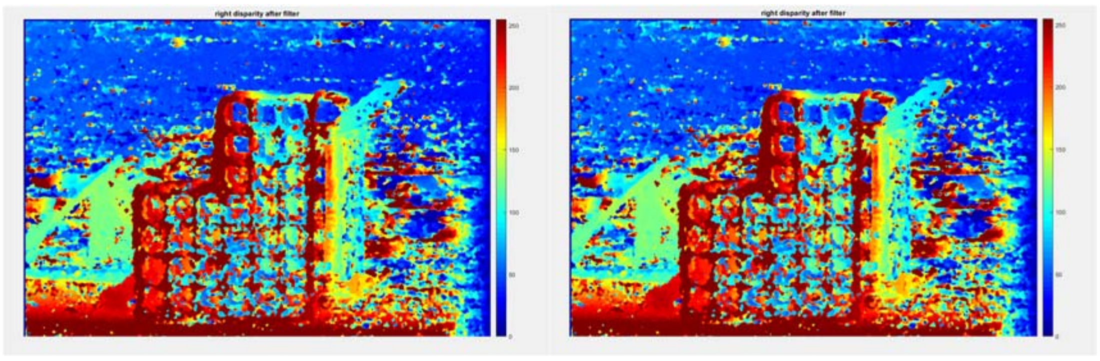
Right Disparity Map of Motorcycle

The final PSNR for motorcycle picture is 15.3556, and its running time is 615s. The general outline of this picture is clear, but when facing some straight line and occlusion points, there still exist some noises, especially in window part.



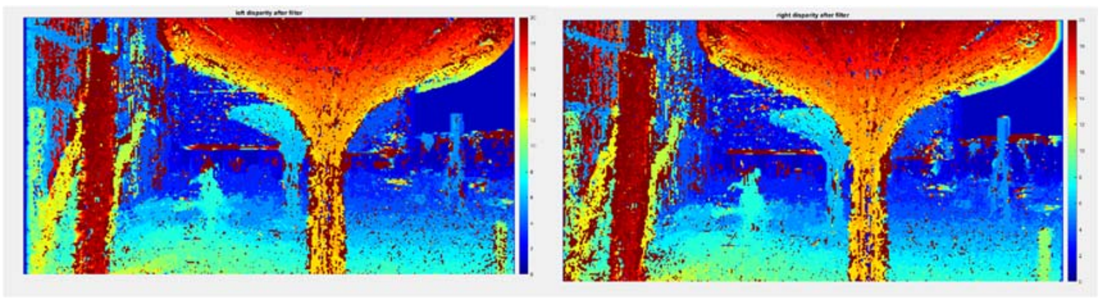
Disparity Map of Terrace (left and right)

The final PSNR for terrace picture is 16.5743, its running time is 189s. The noises of this picture seem to be a little bit more than other pictures, and the disparity on the table seems unreasonable. This is probably because the given maximum disparity of this picture is the whole picture.



Disparity Map of Sword (left and right)

The final PSNR for sword picture is 9.816 and its running time is 719s. The method of this image doesn't show very good performance, the possible reason is that there're lots of straight line in this picture, which share similar characters. Hence the local method tends to mismatch these similar points.

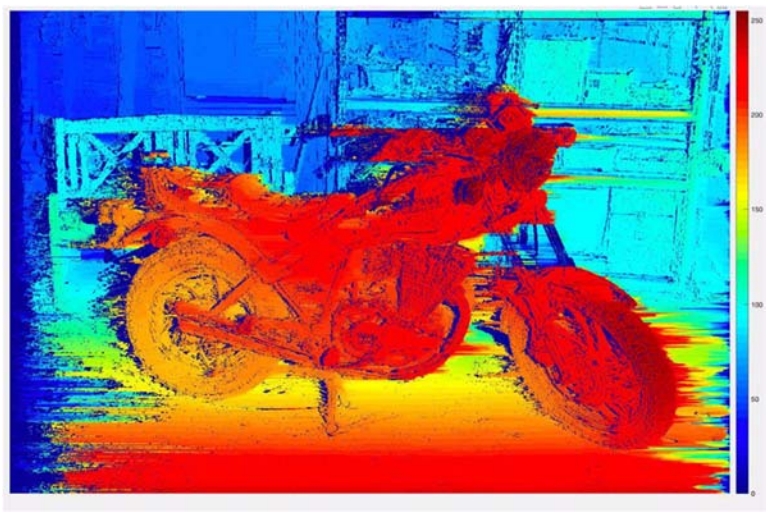

Disparity Map of Playground (left and right)

The final PSNR for playground is 13.1618 and its running time is 392s. The pillar in this image performs better than other parts, which is perhaps because of mass color congregation in this area.

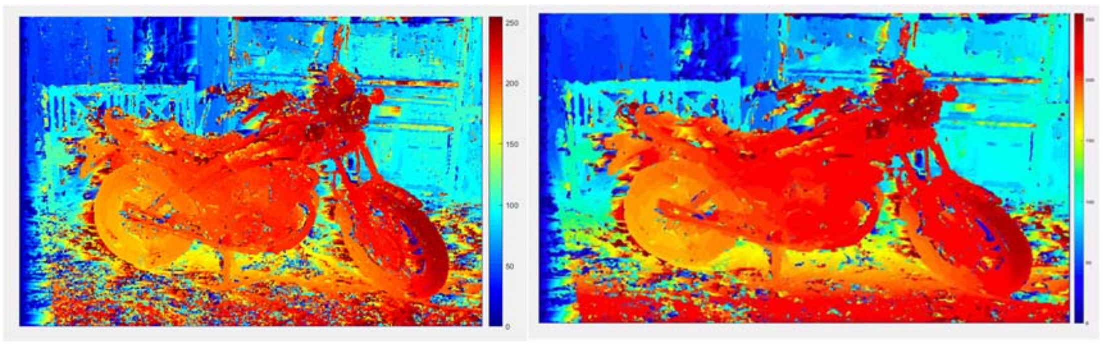
3. Some other tried methods

Besides BM algorithm with SAD+CENSUS cost function, we also tried some other methods, some of them also show good performance.

We tried a method using dynamic programming which proposed in the paper "A maximum likelihood stereo algorithm". This method is suitable for large images such as motorcycle, and sword. Except for some little flaws in image borders, it shows pretty good result. However, when we perform smaller pictures with this method, there would exist many unmatched points which is unstable even if applying extra matching constraints.


Disparity Map by using dynamic programming

In addition, we tried some other combination of cost functions such as SSD, CENSUS, SSD+CENSUS with BM algorithm and then compare their performance.



Disparity Map by using census tranmsformation

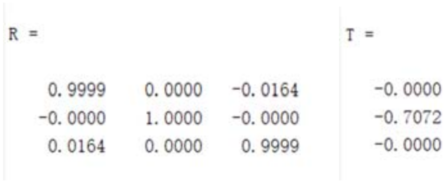
The left picture above takes SSD cost. It is obvious that the chair at the back is better but the ground before motorcycle is worse. The probable reason is that, the squared term extracts the line features more sensitively compared with absolute term in SAD. The right picture above uses SSD+CENSUS cost. We can see that with help of census part, the ground in the picture is more precise and the noise becomes less.

3 Calculation of R and T Matrix

Considering two images are taken from stereo camera, the idea rotation matrix R is identity matrix and transform matrix T should be a 3*1 vector with one non-zero value and two zero values. By calculation we implement the basic 8 points algorithm and RanSaC algorithm.

After obtaining the disparity map from previous section, we can derive pixel pairs with minimum cost, namely the pairs which are more likely to be the correct correspondence. Then we use RanSaC algorithm to calculate robust essential matrix and pick correct RT pairs after decomposition by validation of depth's sign. After iterating a fixed number of times, RanSaC guarantees that essential matrix E reaches convergence.

As is shown in the following pictures, R approaches identity matrix and T has two entries approximately equal to zero.



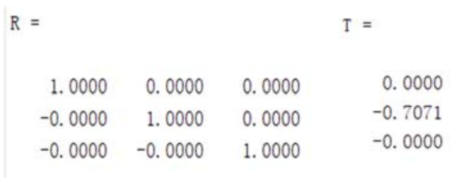
```
R =

    0.9999    0.0000   -0.0164
   -0.0000    1.0000   -0.0000
    0.0164    0.0000    0.9999
```

```
T =

   -0.0000
   -0.7072
   -0.0000
```

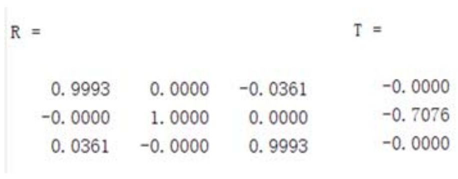
R,T of motorcycle



```
R =                                      T =

    1.0000    0.0000    0.0000               0.0000
   -0.0000    1.0000    0.0000              -0.7071
   -0.0000   -0.0000    1.0000              -0.0000
```

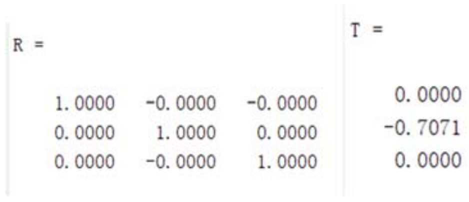
R,T of playground



```
R =                                      T =

    0.9993    0.0000   -0.0361              -0.0000
   -0.0000    1.0000    0.0000              -0.7076
    0.0361   -0.0000    0.9993              -0.0000
```

R,T of sword



```
R =                                      T =

    1.0000   -0.0000   -0.0000               0.0000
    0.0000    1.0000    0.0000              -0.7071
    0.0000   -0.0000    1.0000               0.0000
```

R,T of terrace

We can infer from these four pairs that the standard T matrix is about [0;-0.7071;0] and R approaches identity matrix.

4 3D Reconstruction

Every point from picture locates in pixel coordinate which is in two dimensions. Therefore, the coordinate of each point (vector) only have two components x and y. Their homogeneous coordinates contain three components which are x, y and 1 at last.

When we get a picture, we can determine x and y coordinate components of every points in picture. From left to right y is from 1 to the width of the picture. And from top to down x is from 1 to height of the picture. For instance, the given motorcycle picture is 1988×2964, the x coordinate is from 1 to 2964 and y is from 1 to 1988. The only thing we need for 3D reconstruction is the third coordinate z. To convert from the floating-point disparity value d (pixels) to depth Z (mm) the following equation can be used:

$$Z = \text{baseline} * f / (\text{disparity} + \text{doffs})$$

baseline: camera baseline in mm

doffs: x-difference of principal points, doffs = cx1 - cx0

f: focal length in pixels which is stored in camera matrices cam0,1

cam0,1: camera matrices for the rectified views and in form:

$$\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

c_x, c_y: principal point

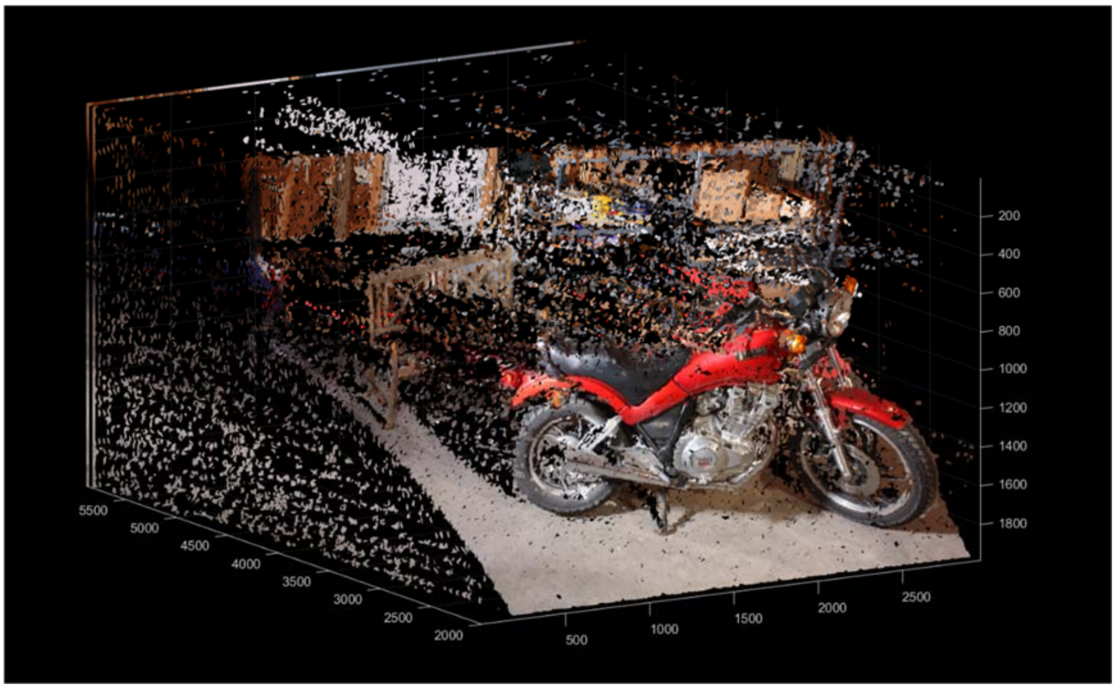
According to perspective projection the relation between pixel coordinate with world coordinate is: (treat left camera locating in origin of world coordinate, therefore, there is no transformation matrix)

$$X \sim K_s K_f \prod P = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P$$

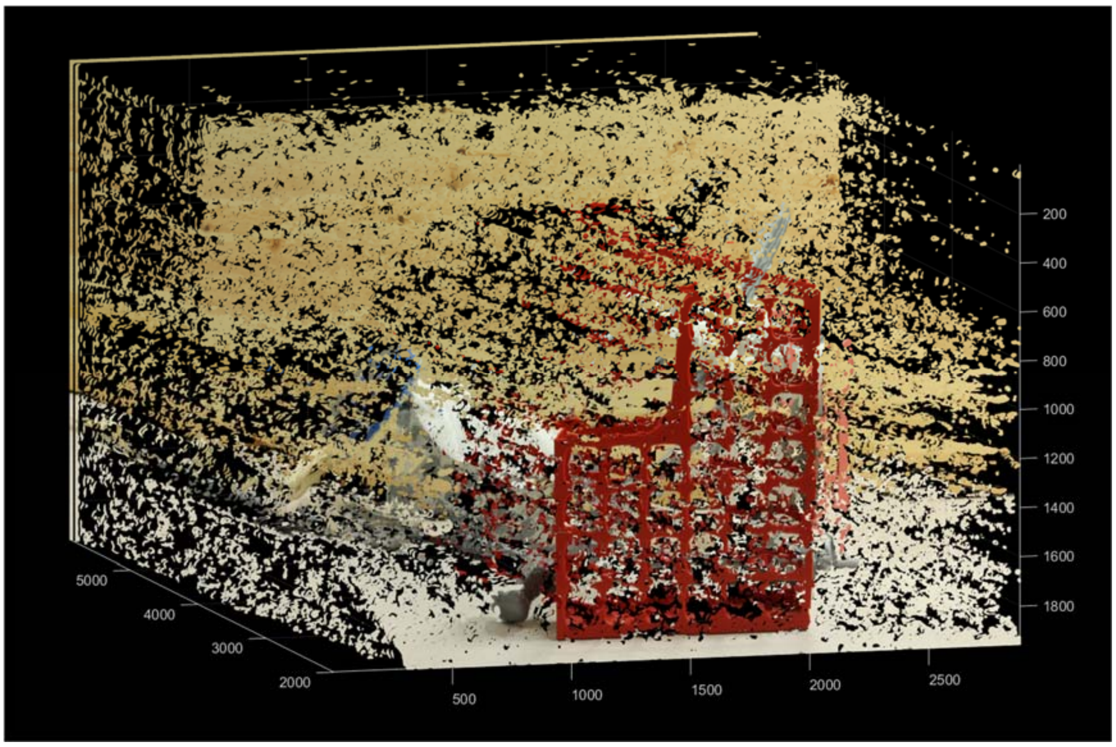
$X = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ is homogeneous coordinate in pixel coordinate

$P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ is homogeneous coordinate in world coordinate

The goal of 3D reconstruction is to recovery P. According to above formula we can calculate P easily. At last we plot all points in three dimensions by using pcshow. The following figures are 3D reconstruction of motorcycle and sword.



3D Reconstruction of Motorcycle



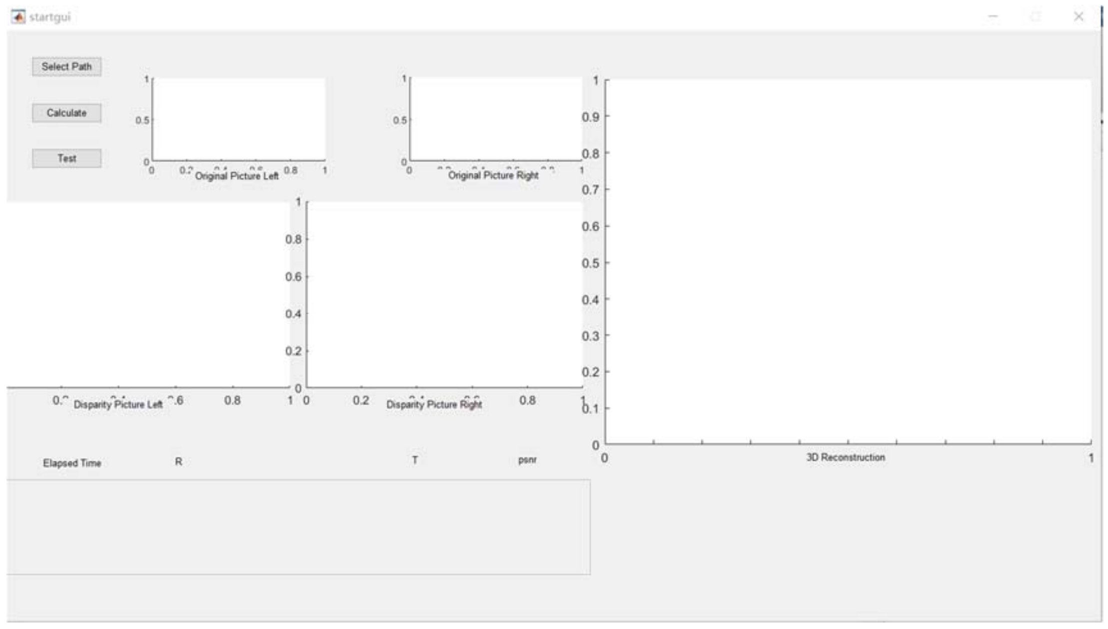
3D Reconstruction of Sword

From these figures we can easily see that the 3D reconstruction shows the z-component differences between all points very intuitively. It is obviously the 3D reconstruction of floor is very smooth and precise. But we can also see that there are also many points which contain not appropriate z-components. This unappropriate 3D reconstruction is caused because of error of calculation of disparity map. How to reduce the error of calculation of disparity map further is our future work.

5 GUI

In this part, we have designed the graphical user interface with the help of guide in command line. The following picture shows our design of GUI

By clicking Run in startgui.m or directly start gui in startgui.fig, the interface will be started in front of the monitor. The layout of the gui is shown as follow:



GUI

There are three buttons which have different functions:

The first one is aiming at selecting the path of pictures, after choosing the path properly, the path will be displayed after the button 'Select Path'. Of course, the 2 original pictures which located in the path will also be displayed in where they should be (the 2 blank axes on the top of the GUI).

The second one works just like his name, it is used to calculate the disparity map of pictures and also be able to calculate the PSNR. It calls functions disparity_map.m, threedreconstruction.m and verify_dmap.m after the calling, the disparity map of both left and right original picture will be calculated and displayed in the middle of the GUI, at the same time, 3D reconstruction of the left picture will be shown in the right axe, the value of R, T, PSNR and the running time of calculating disparity map will be showed in the panel on the bottom of the GUI.

The third button is to test the toolbox usage, variable usage, and difference between PSNR that are calculated in different ways with the help of calling the unittests.

To what we should pay attention is: if you haven't selected path, however, clicked the button calculate or test, it wouldn't work, because the program has no idea what it should calculate or test.

Another thing we should pay attention is that after we run GUI to process images one time, we should close GUI, and run startgui.m again, and select path of other images, and then calculate and test just like described above. Every time we want to use GUI to process images, we should close GUI which has already been run for last task and rerun function startgui.m.

Step by step, you are expected to select path at the most first, otherwise, there will be a mistake to remind you there is no path if you use the button 'Calculate' or 'test'.

And our steps to build GUI is as follows:

Step1
Start GUIDE by typing guide at the MATLAB prompt. In this part, we have used a graphical window to help us build a visible GUI, in which we could easily construct the basic layout, for example, the position to show images and some other information.

Step2.
In this step, we have written scripts to control the GUI to interact with users. So, we can easily conclude that the most important thing is callback of each pushbutton. In pushbutton callback we have defined actions for example to show images, call the functions which are properly programmed and return some results and to display some of them on the GUI.

After these two steps, we have finished the designing of GUI.

6 Unittests

In this part there are three important things to concentrate:

1. Test Toolbox
2. Test Variables in challenge
3. Test difference between hand by hand calculated PSNR and PSNR calculated by Toolbox

The first step we did is to call codetools.requiredFilesAndProducts which will be used in testing a '.m' file and returns a plist which indicates the usage of toolbox product. The flag is to denote whether toolboxes have been used. If it is true, that means the number of used toolboxes is not equal to 1. The other toolboxes which are been used here will then be printed. Otherwise, there is only one toolbox which is MATLAB which was used in the previous files, namely census.m, challenge.m, verify_dmap.m and disparity_map.m.

Secondly, in order to test the variable in challenge, we have used assignin() function to assign all important variables to the workspace and then, we will test all of them whether they are empty or not, with the help of isempty(). We have created an array which contains logical numbers false or true to record the results from isempty() function. If they are all true, the test will be passed.

At last, we have made a comparation between integer number variable ‘tolerance’ and abs(subtraction of two PSNRs). It will be failed if the absolute result of subtraction would larger than tolerance. On the contrary, it will be successfully passed.

7 Summary

In this challenge we calculated the disparity map from two images which are taken by left and right cameras. We tried different methods to calculate the disparity map and compared each method with others. According to advantages and disadvantages of each method we choose one method which has best performance as our final version. The PSNR by using the best method is 16.5743. Besides we calculated the rotation matrix R and translate matrix T. With R and T the Euclidean movement of this two cameras can be determined. At last we made 3D reconstruction to recovery object in three dimensions. The result of disparity map and 3D reconstruction by using our method is much better than by using block matching with SAD as similarity metric, especially the disparity map and the 3D reconstruction of floor is smooth and accurate when we process motorcycle images. How to reduce matching error further is our future work.

Reference

[1] Cox, Ingemar J., et al. "A maximum likelihood stereo algorithm." Computer vision and image understanding 63.3 (1996): 542-567.

[2] Xing Mei1,2, Xun Sun1, On Building an Accurate Stereo Matching System on Graphics Hardware 2011 IEEE International Conference on Computer Vision Workshops

[3] X. Sun, X. Mei, S. Jiao, M. Zhou, and H. Wang. Stereo matching with reliable disparity propagation. In Proc. 3DIMPVT, pages 132–139, 2011.