Disparity Map Estimation Based on Block Matching with Variable Convolution Kernels

Group 30

Hangze Gao, Yansong Wu, Yuetao Wu, Zhen Zhou, Zibo Zhou

Introduction

This project is about stereo vision. It's aimed at inferring depth from two or more images. In this project we compute a disparity map from two rectified stereo images and show the 3D image from these two images.

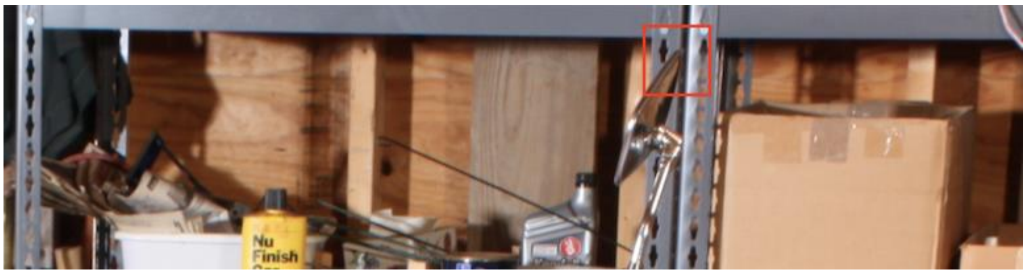
This project mainly includes three parts: Disparity Map, Verification and Unittest.

Disparity Map

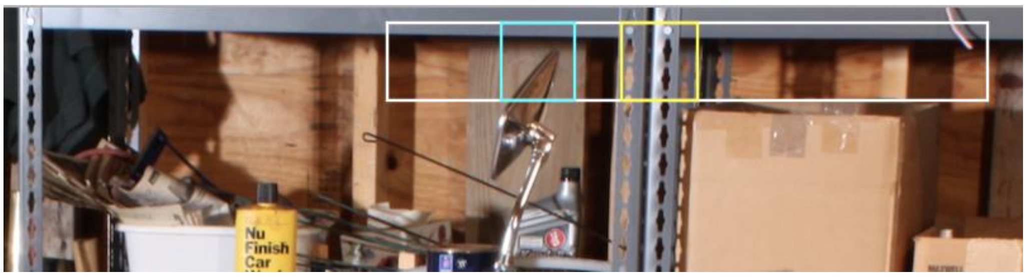
With stereo cameras, objects in the camera's field of view will appear at slightly different locations within the two images due to the cameras' different perspectives on the scene.

Firstly, we do the simple block matching between the two images. Normally we choose a small region of pixels from right image and search the most similar region of pixels in the left image. In a similar way, when we search in the right image, we start from the same coordinate of the template of the left image until the largest distance from left to right. The Disparity Map is the smallest horizontal distance between the block in the right image and the center pixel of matching block in the left image.

For example, we will get the pixel area in the red box on the left image:



Then in the right image we can find the matching block:



We start with the same coordinates as the template (indicated by the yellow box) and look at the right maximum distance to the left and right. The matching block is the blue box in the second image. The horizontal distance between the blue and yellow boxes is the disparity we need.

But how can we get the matching block? This leads to a new concept. *Sum of absolute differences (SAD)* is the criterion to find the matching block. Before calculating the depth image, we need to convert these two images into grayscale so that we have only one value (0-255) for each pixel.

To find the most similar block to the template, we compute the SAD values between the template and each block in the search region, then we just need to choose the block with the lowest SAD value.

After the function of disparity map, the value of Disparity map D, the description of the Euclidean movement as a rotation matrix R and displacement vector T will be given out.

Verification

In this part, we will calculate the *Peak Signal-To-Noise Ratio (PSNR)* between the Disparity Map D and Ground Truth G and elapsed time to evaluate the performance of our algorithm.

PSNR is most easily defined by *Mean Squared Error (MSE).* Given a noise free $m \times n$ monochrome image G and it's noisy approximation D, MSE is defined as;

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [G(i,j) - D(i,j)]^2$$

The PSNR is defined as:

$$PSNR = 10log_{10}\left(\frac{2^B - 1}{MSE}\right), with\ B = 8bits$$

Unittest

This part includes three functions: check toolboxes, check variables and check psnr.

- Check toolboxes: in this part we check that we don't use any functions from toolboxes and we only use the basic functions from MATLAB
- Check variables: in this part we check that the variables in challenge.m are not empty.
- Check psnr: in this part we compare our own implementation of the PSNR with the in the Image Processing Toolbox and check if the results within a reasonable tolerance (we set the tolerance is 0.1).

Block Matching Complexity Reduction

When Simple block matching is used to process large-size images, the complexity of block comparisons will increase dramatically, and the amount of computation is surprisingly large. Therefore, we consider that target and template are not taken in a for-loop structure to calculate SAD. Our team came up with an algorithm that for-loop structure is eliminated to reduce computation time. First we overlap the two images, then shift the image on the left to the right pixel by pixel and the distance of the movement in pixel coordinate system is recorded as d, here d is in 0 to possible maximum disparity d_{max} i.e. the image on the left is moved d_{max} times. Second, compute the absolute value of the difference between the two matrices in overlapping portions. Then the overlapping portion is convoluted with following three convolution kernels (horizontal, vertical and diagonal), which has the same size as block, After that multiply the results of the three convolutions. These three convolution kernels can consider the more direction information of the image. Take 3×3 as an example,

$$\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix} \qquad \begin{bmatrix}1 & 0 & 0\\1 & 0 & 0\\1 & 0 & 0\end{bmatrix} \qquad \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$$

Finally, we get a three-dimensional matrix. The first and second dimensions are the pixel coordinates of the image on the left, and the third dimension represents the distance of the sliding. The value stored by the matrix is the corresponding SAD value. From this matrix we can derive the d coordinate of the smallest SAD corresponding to each pixel coordinate. The d coordinate is the disparity of the corresponding center pixel in the left image. In addition, we can also move the image on the right to the left and get the corresponding disparity of the right image. The use of these two methods is determined by the performance of them.

What we can do to improve the performance of this algorithm is we can adaptively change the window size i.e. convolution kernel size with respect to the gradient of the image. Exploiting larger window to block with larger gradient.

Experiment

The experiment is based on MATLAB 2018a, CPU Intel(R) Core i5-8250U @1.60GHz and RAM 8.00GB. With the algorithm above we can reduce the computation time and get the following results (see Figure 1.).

The PSNR results is showed in the following table.

	Image pairs			
	motorcycle	playground	terrace	sword
PSNR (dB)	14.7709	33.0351	30.6874	10.2896
Elapsed Time (s)	111.065	0.810	0.504	122.044
R	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$
T (m)	$\begin{bmatrix}0\\0.1930\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.0596\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.0599\\0\end{bmatrix}$	$\begin{bmatrix}0\\0.1697\\0\end{bmatrix}$

Table 1. PSNR between Ground Truth and our results, elapsed time of the process and R, T results

We can conclude that our algorithm for *playground* and *terrace* did a great job both on PSNR and visually. In addition, the computation time of our algorithm is quite short even when processing image pairs with large size such as *motorcycle* and *sword*. However, for image *motorcycle* and *sword*, the PSNR is much lower though the visual result of *motorcycle* is quite good. We can see a lot of noise in each depth image. If we combine the gradient information of each block comparison, we would obtain better results.

3D Plot

The *plot_3D* function is used to show a disparity map in 3D form. The pixel $(x(i), y(i))$ of original picture printed at $(x(i), y(i), d)$ with color $I(x(i), y(i))$. The *d* parameter is disparity from the front part because pixels with larger disparities are closer to the camera, and pixels with smaller disparities are farther from the camera. Here, we use the MATLAB function 'surf()' to create this three-dimensional surface plot and set the property as 'none', which means showing a plot without edge-color. The result looks like rendering a 3D surface the corresponding picture. Following figures are from *plot_3D* function for different original figures.

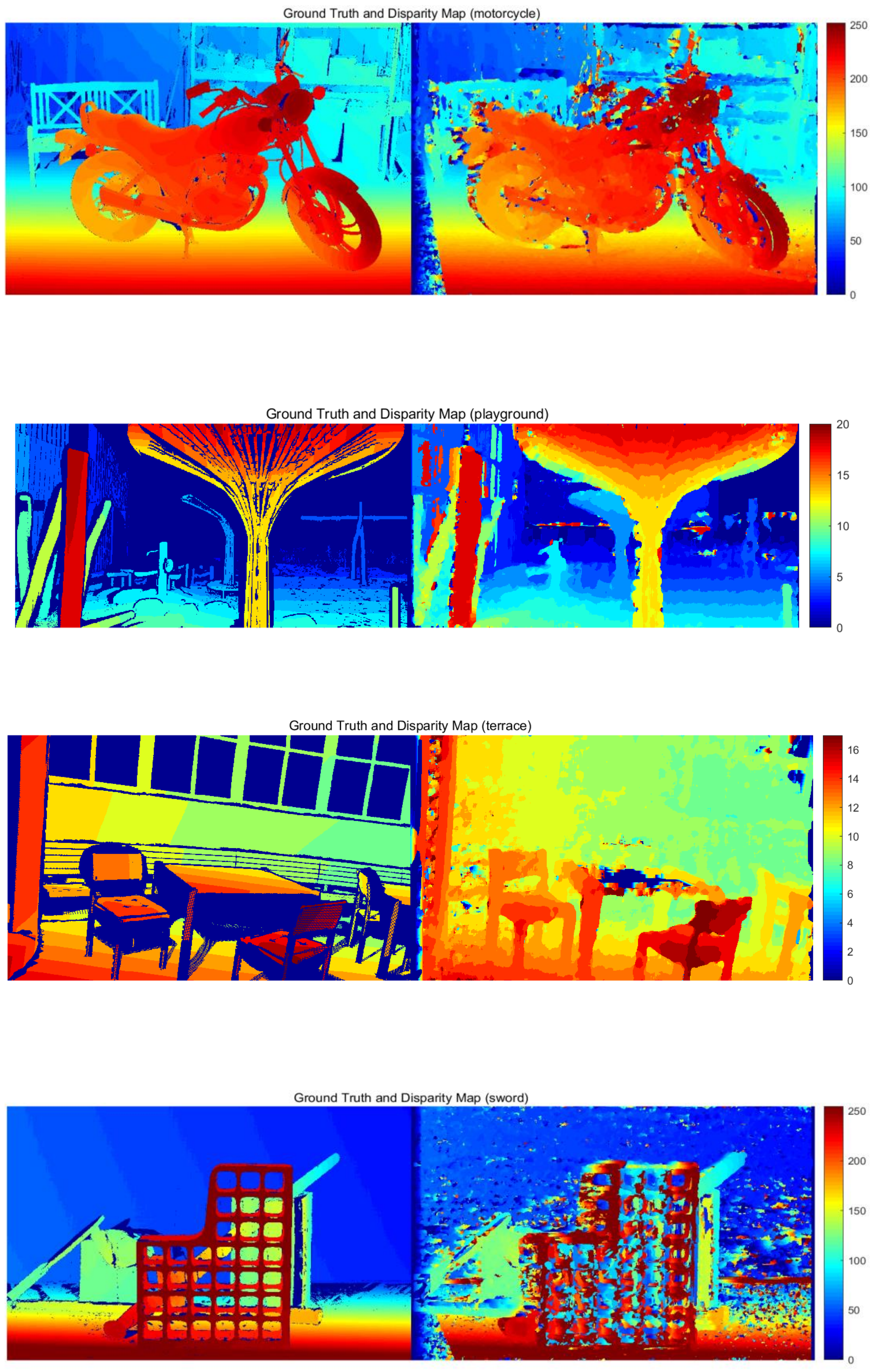
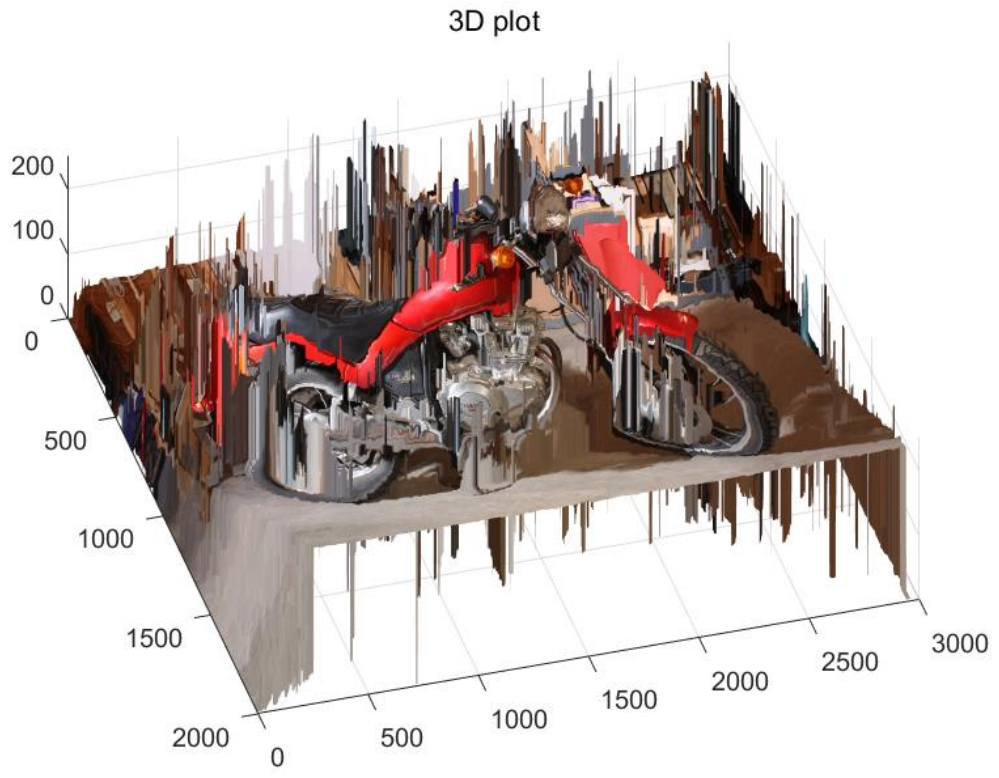
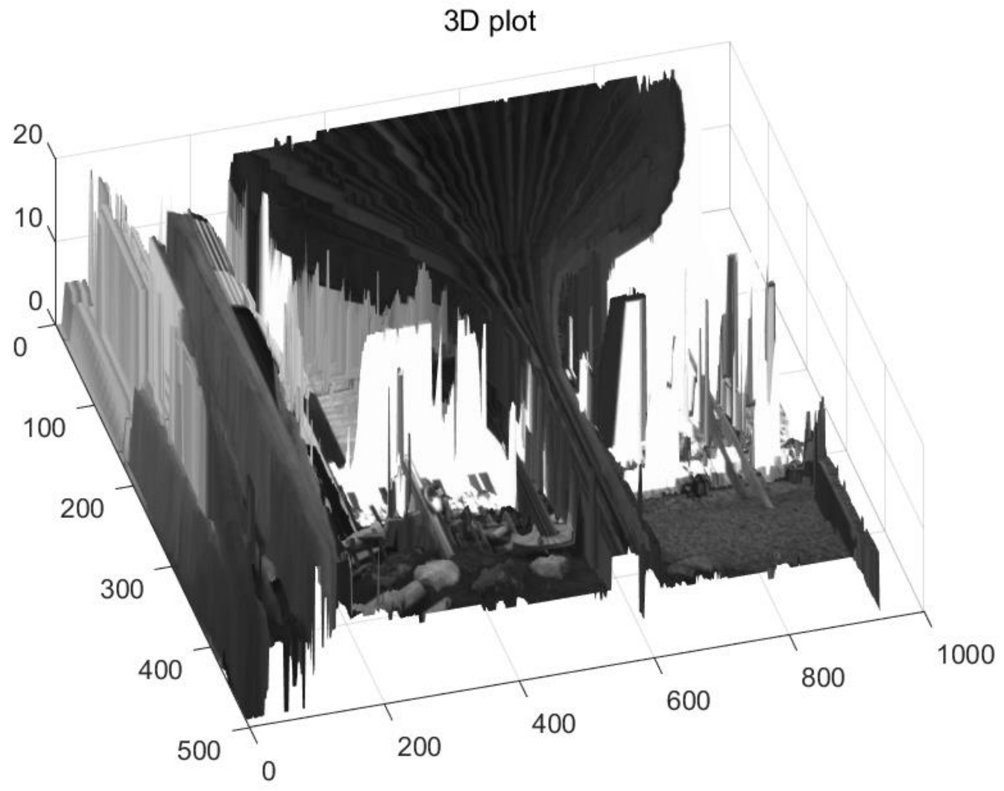
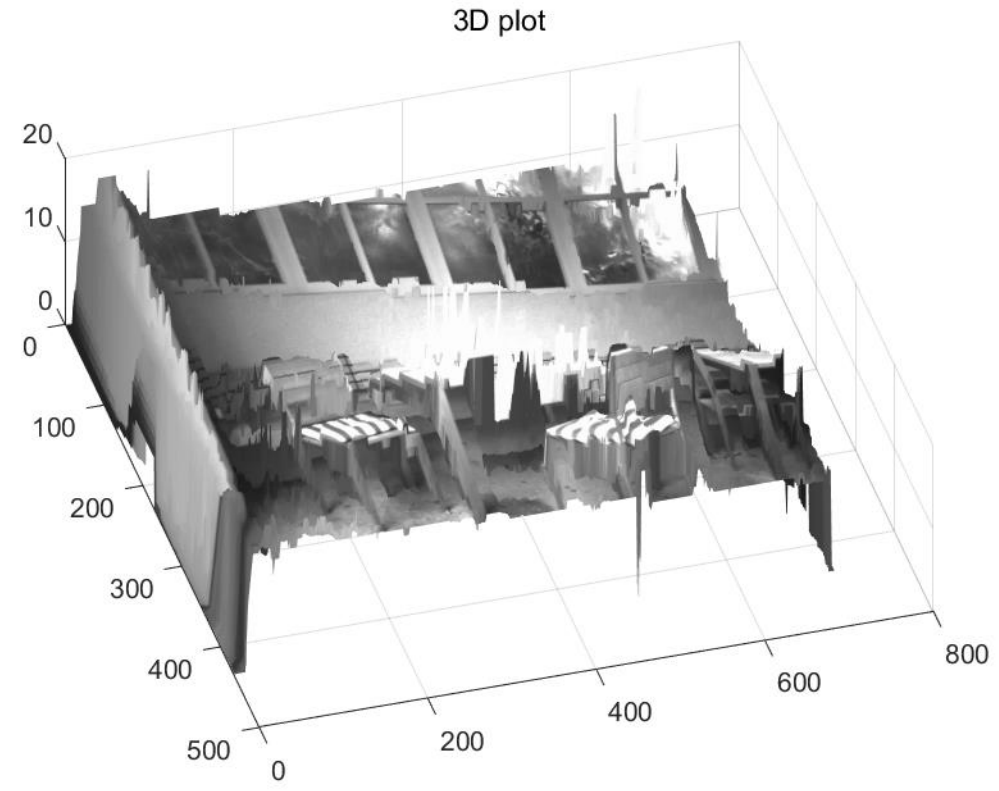


Figure 1. Ground Truth (left) Disparity Map (right) of images pairs (motorcycle, playground, terrace and sword)


3D plot
200
100
0
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000


3D plot
20
10
0
0
100
200
300
400
500
0
200
400
600
800
1000



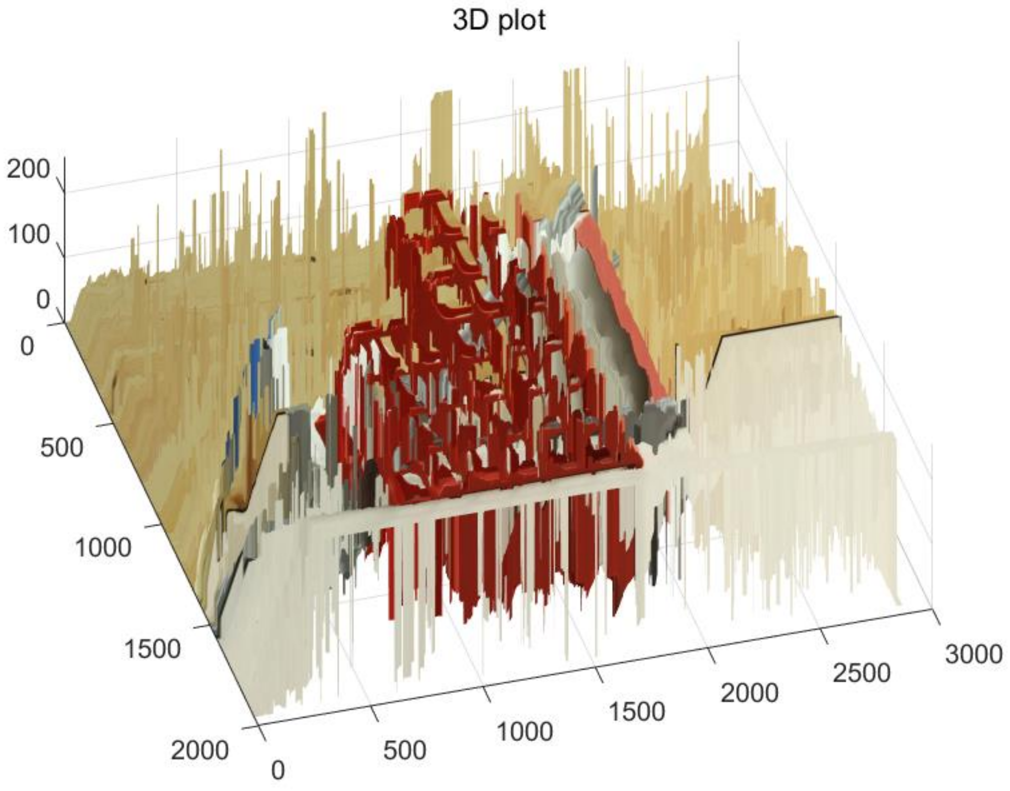
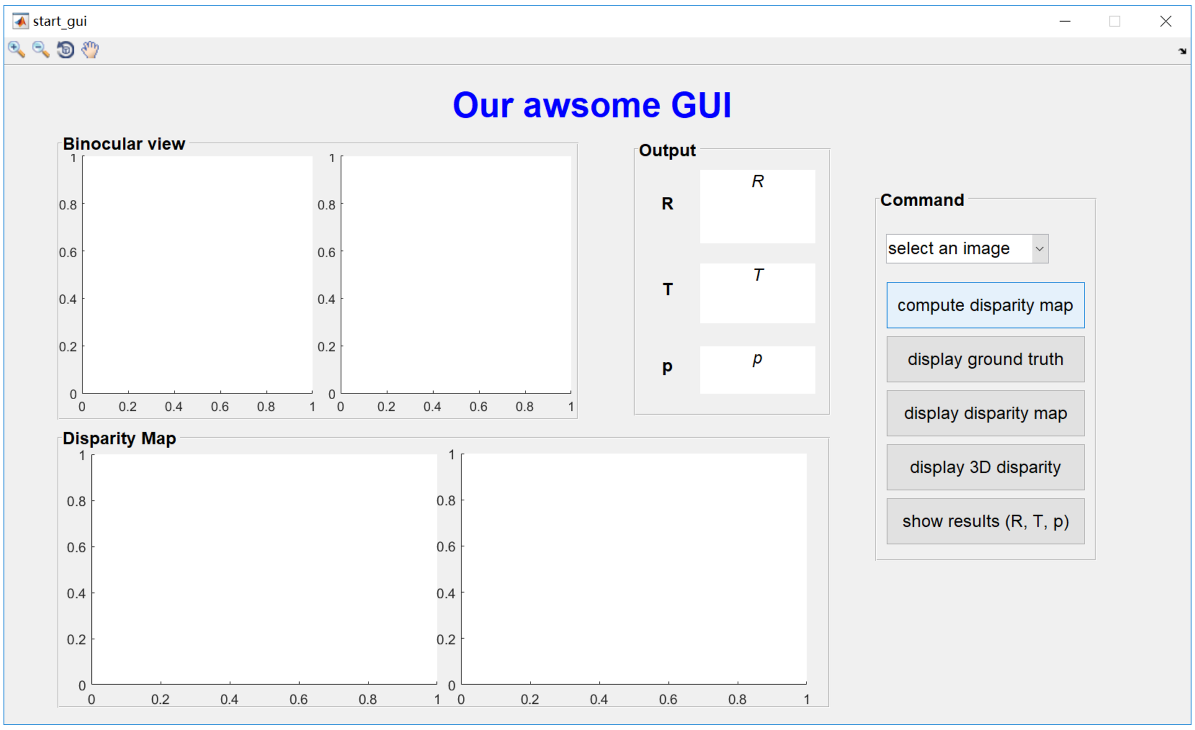


Figure 2. 3D plot of test image pairs (motorcycle, playground, terrace and sword)

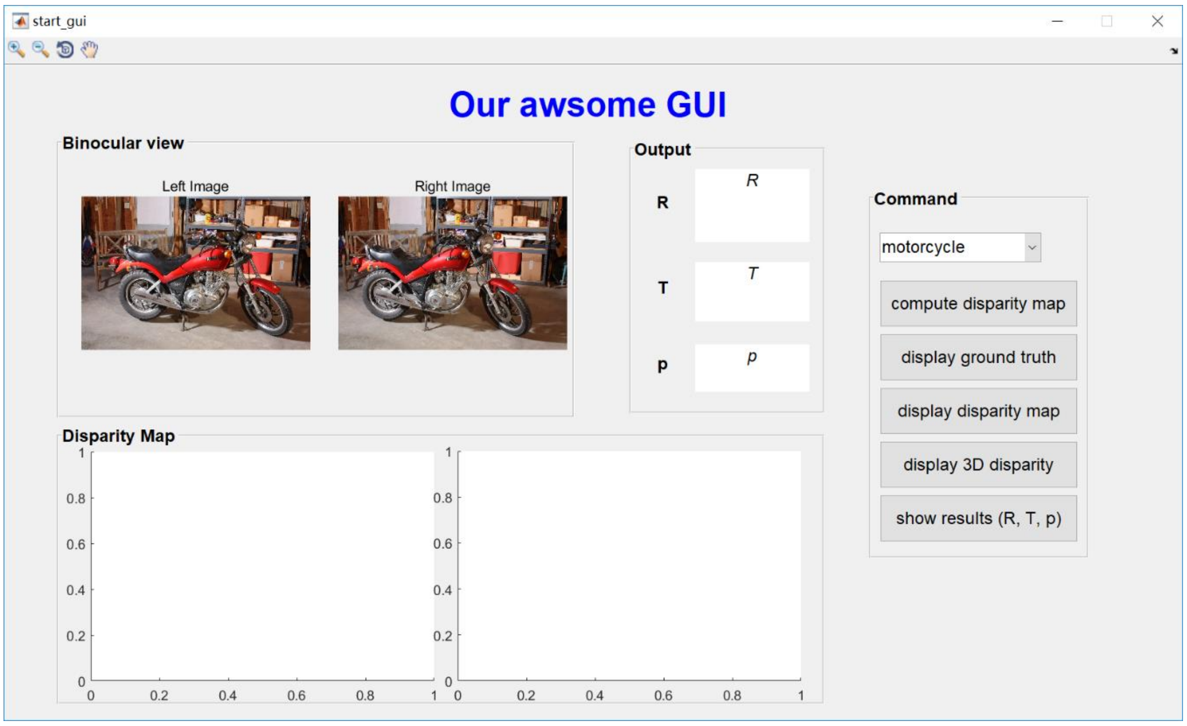
GUI

In this part, we generate GUI for our project, then we will show our GUI.

Firstly, it is our start page.

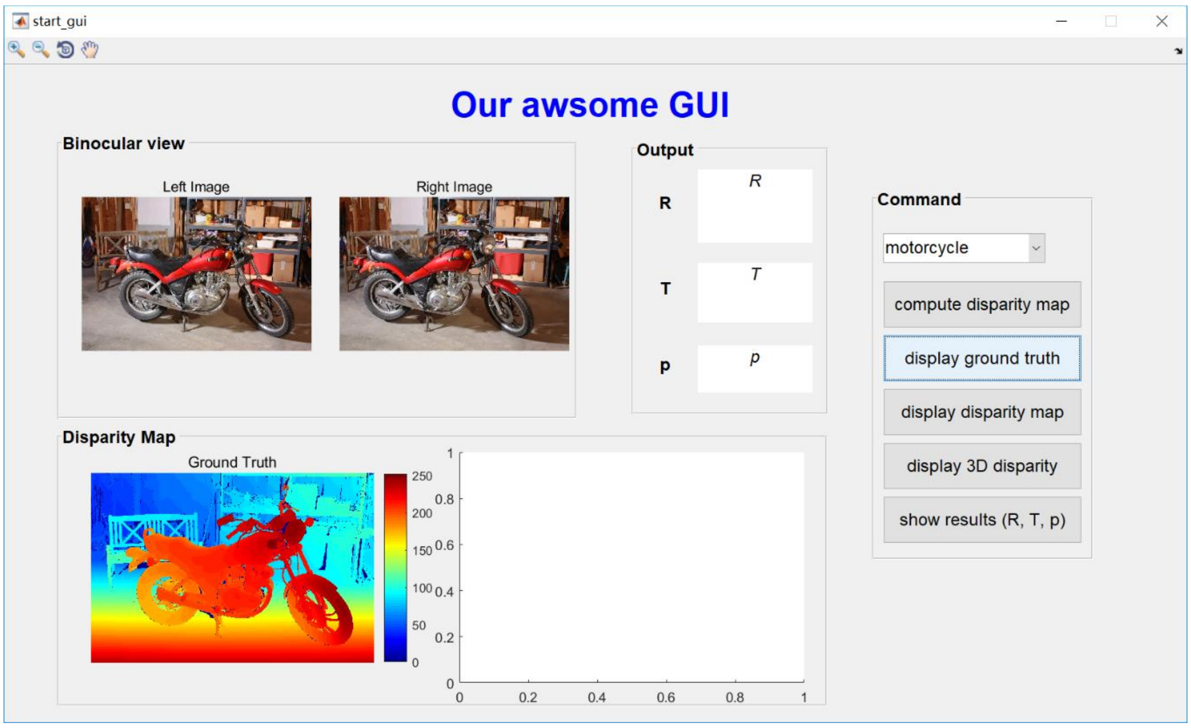


Then, we choose an image *motorcycle* and show the left and right images.

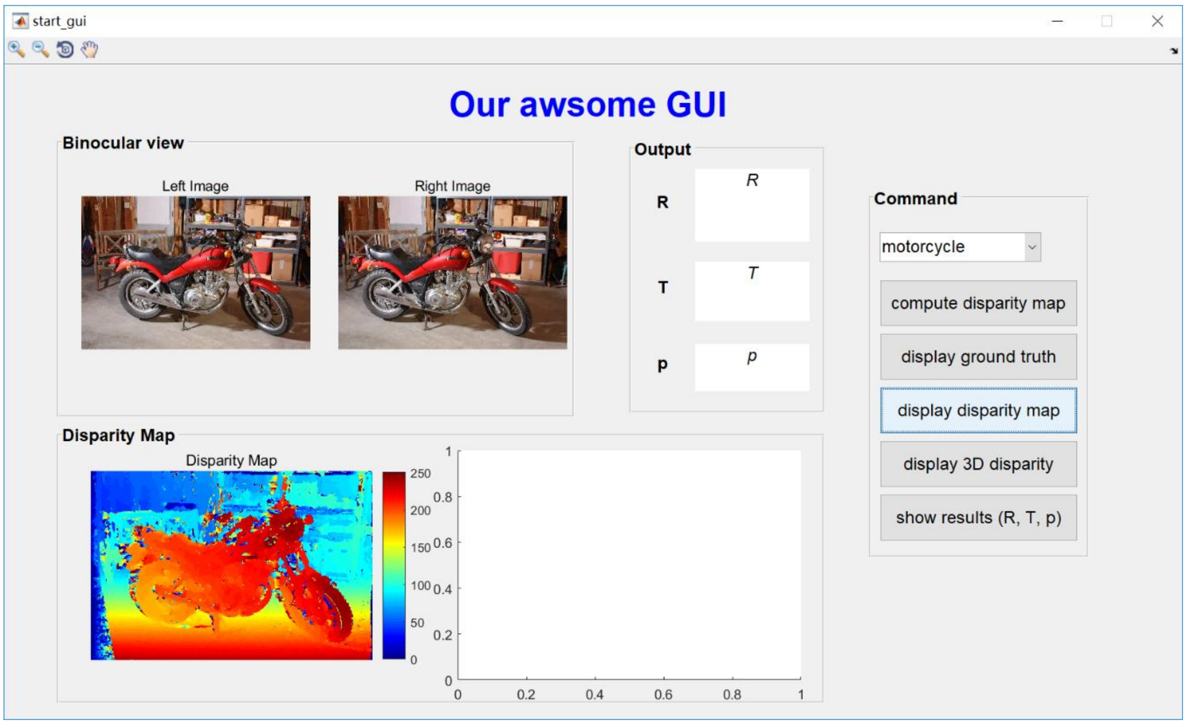


Then, we can use the following buttons: compute disparity map, display ground truth, display disparity map, display 3D map and show results of R,T and p.

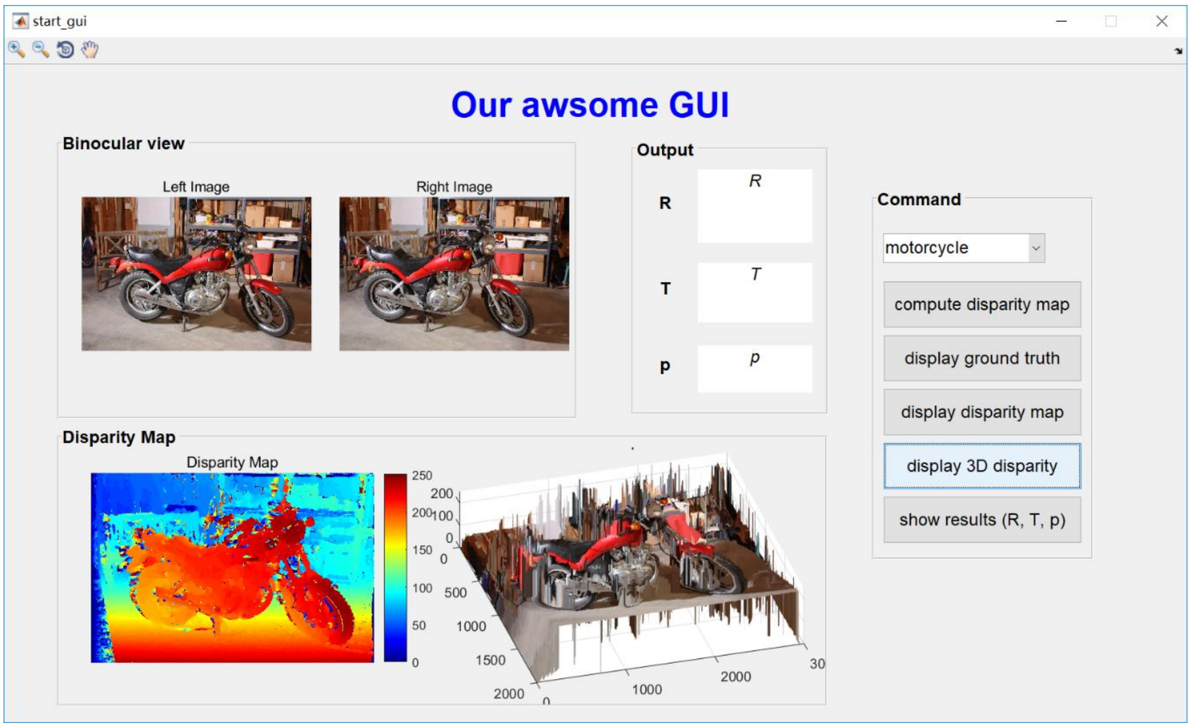
- Display ground truth:



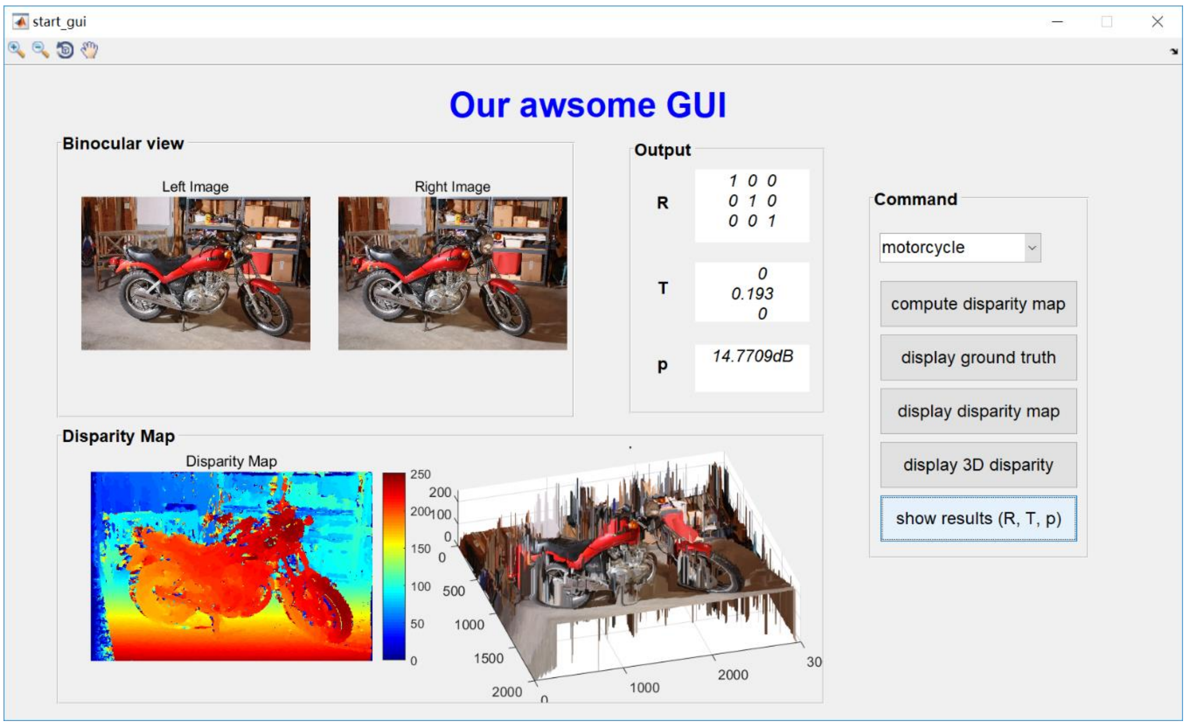
- Display disparity map:



- Display 3D disparity:



- Show results of R, T and p:



R:

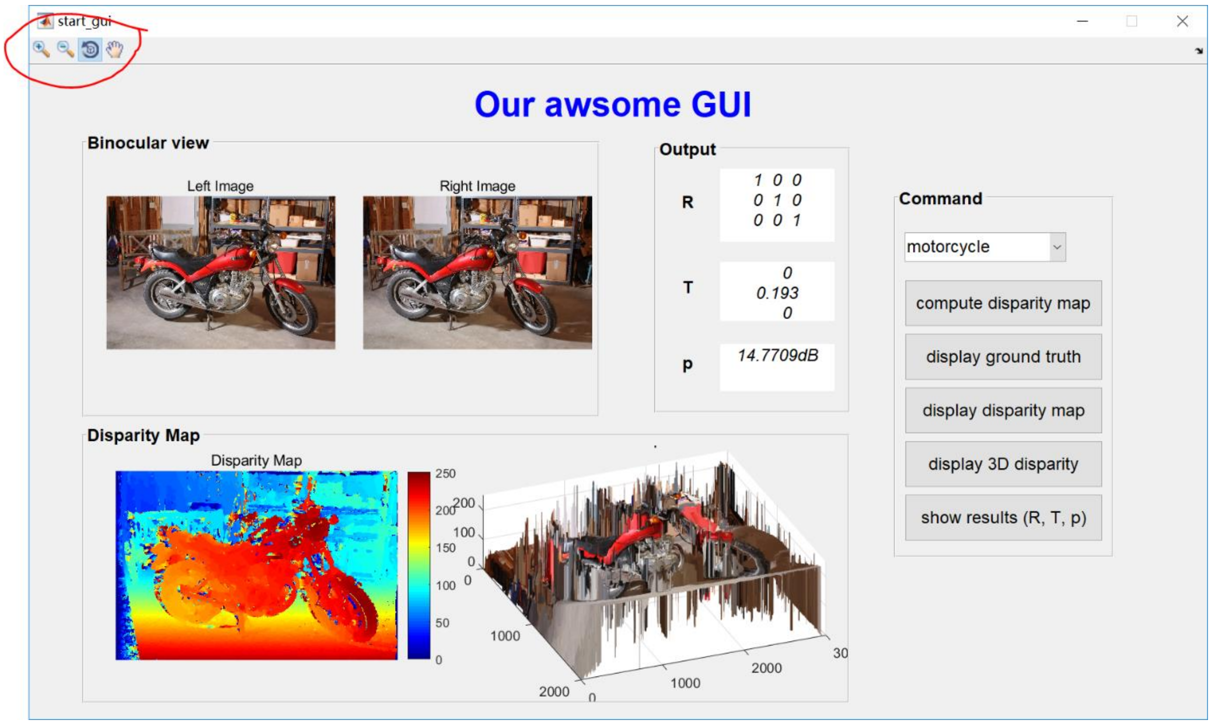
```
1 0 0
0 1 0
0 0 1
```

T:

```
0
0.193
0
```

p: 14.7709dB

In addition, we can use four extra buttons which are circled in the following image to zoom or rotate to show local part of the images we create.



R

```
1 0 0
0 1 0
0 0 1
```

T

```
0
0.193
0
```

p 14.7709dB

References

[1] Chen Y S, Hung Y P, Fuh C S. Fast block matching algorithm based on the winner-update strategy[J]. IEEE Transactions on Image Processing, 2001, 10(8): 1212-1222.

[2] Einecke, Nils, and Julian Eggert. "A multi-block-matching approach for stereo." 2015 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2015.