COMP61342 : Computer Vision

Stereo Vision Algorithms

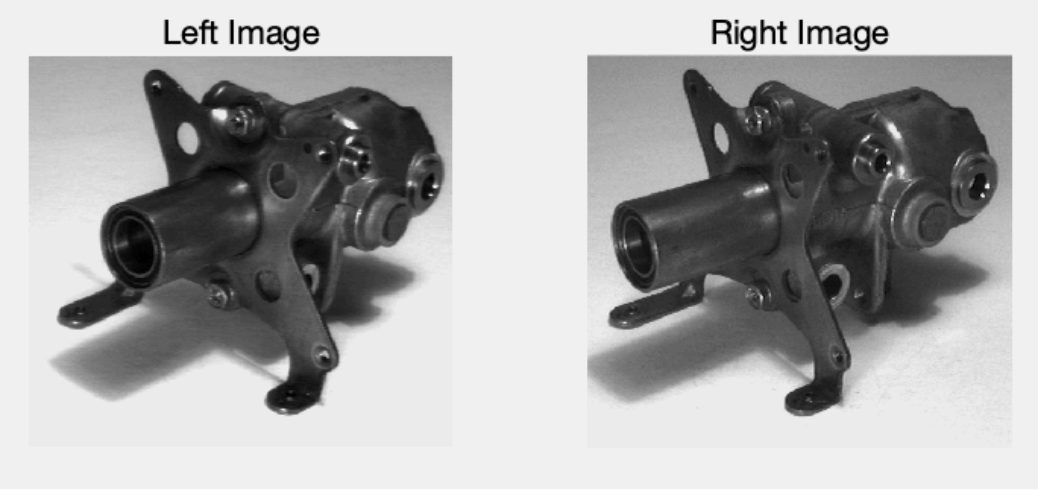
Bowen Zhang 10289193

1. Experiment Overview

The purpose of this experiment is to generate industrial structures in the 3D world by matching two 2D images. The key steps to achieve this goal through Matlab are as follows:
(1). Load 2D image through method ‘imread’.
(2). Use the method ‘edge’ to detect interesting points in 2D images.
(3). Match the above interesting points to corresponding points.
(4). Calculate the disparities between corresponding points between two pictures.
(5). Calculate the X, Y, and Z coordinate from these disparities.
(6). Render views.

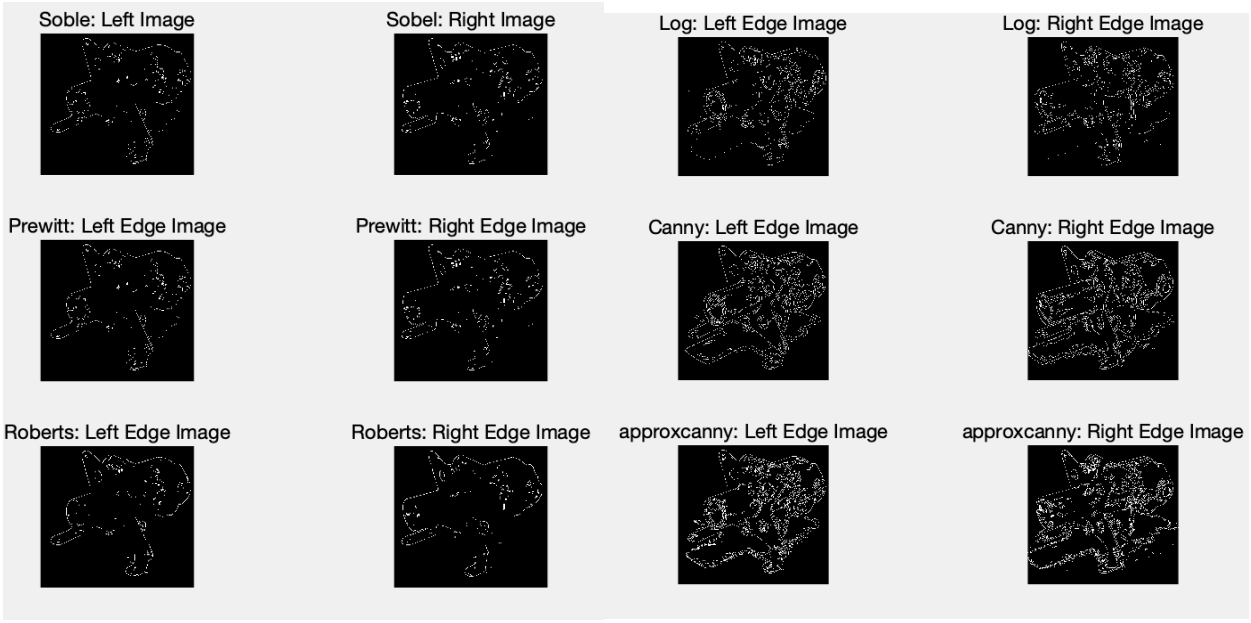
2. Experiments and results

According to the steps above, first import the images required for the experiment into the Matlab workspace through the method ‘imread’, as shown below.



2.1 Feature Extraction

In this experiment, I used the method ‘edge’ for image feature extraction. After consulting the official Matlab documentation, I found that the method ‘edge’ has multiple edge detectors, so I tried each edge detector in turn for comparison. The following figure shows the output of each edge detector.



From the above picture, we can see that the edge detectors 'log', 'Canny', and 'approxcanny' are better, but the three edge detectors also do edge processing on the shadow part, but the shadow is not the real object Characteristics, so it should be regarded as noise. After comparing the edge detectors ‘Sobel’, ‘Prewitt’, and ‘Roberts’, I found that ‘Sobel’ works best, and the other two have lost too many features of the object, so I used ‘Sobel’ for subsequent experiments.

2.2 Match Correspondences

We can match the correspondence between the left and right images according to the above interesting points. According to the assignment, these two pictures have been corrected, so the polar lines in the pictures are aligned, so that we can match in units of rows without the need for an additional global search.

In the matching operation, we traverse the interesting points in each row of the left image in turn, calculate the distance from the interesting points in the right image, and then select the interesting point with the shortest Euclidean distance. The purpose of performing this step is to ensure that the feature points of the left and right images have similar positions.

2.3 Calculate Coordinates

According to the disparities we obtained above, we can use the following formula to calculate the X, Y, Z coordinates of each point.

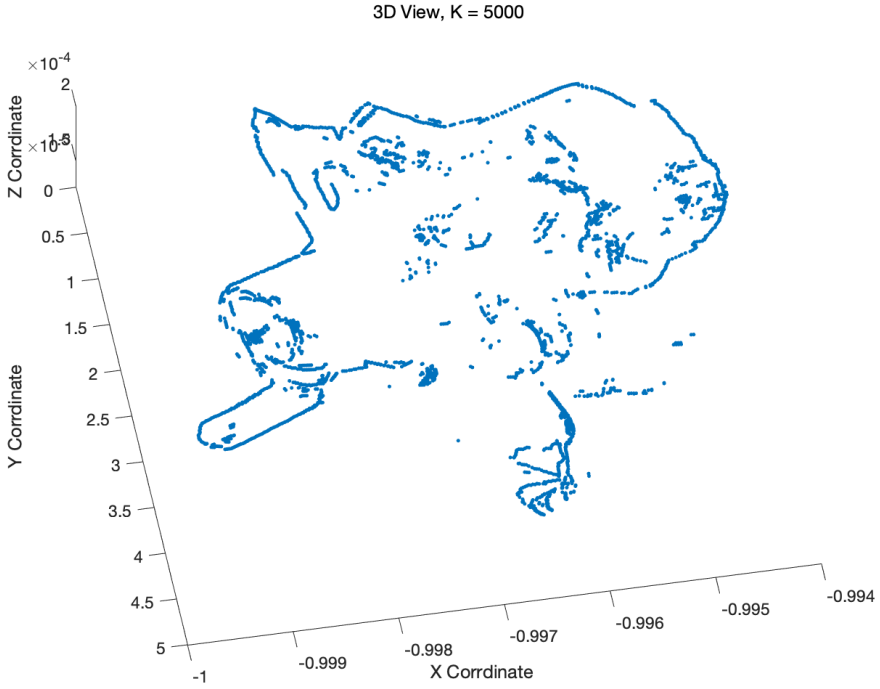
$$\begin{cases} Z = \dfrac{1}{K + \ disparity} \\ X = \ Z * \dfrac{x_l}{f} - \dfrac{b}{2} \\ Y = Z * \dfrac{y_l}{f} \end{cases}$$

In the formula, disparity is the minimum distance we calculated. f is the focal length of the camera (17.0 mm), b is the baseline value. I first set the K value to 5000. I will discuss the setting of the K value in the following section.

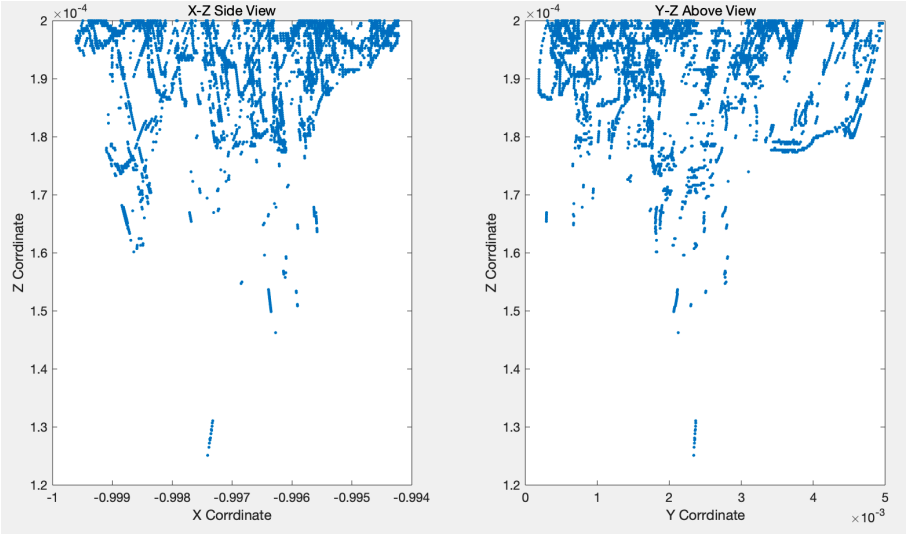
The code of the process is shown in the figure below.

```
for row = 1:num_rows
    left_edge_pixels = find(left_edge_image(row,:));
    right_edge_pixels = find(right_edge_image(row,:));
    for xl = left_edge_pixels

        %using minimal value of disparities
        disparities = (right_edge_pixels - xl)';
        disparities = abs( disparities );
        disparities = min( disparities );
        disparities = disparities * pixelSize * 1000;

        num_matches = num_matches + 1;

        Z = ( 1.0 / ( k + disparities ) );
        X = ( Z * (xl/f) - baseline/2 );
        Y = ( Z*  (row/f) ) ;

        points = [ points; [X, Y, Z] ];
    end
end
```

After performing these operations, the 3D image I obtained is shown in the following figure.

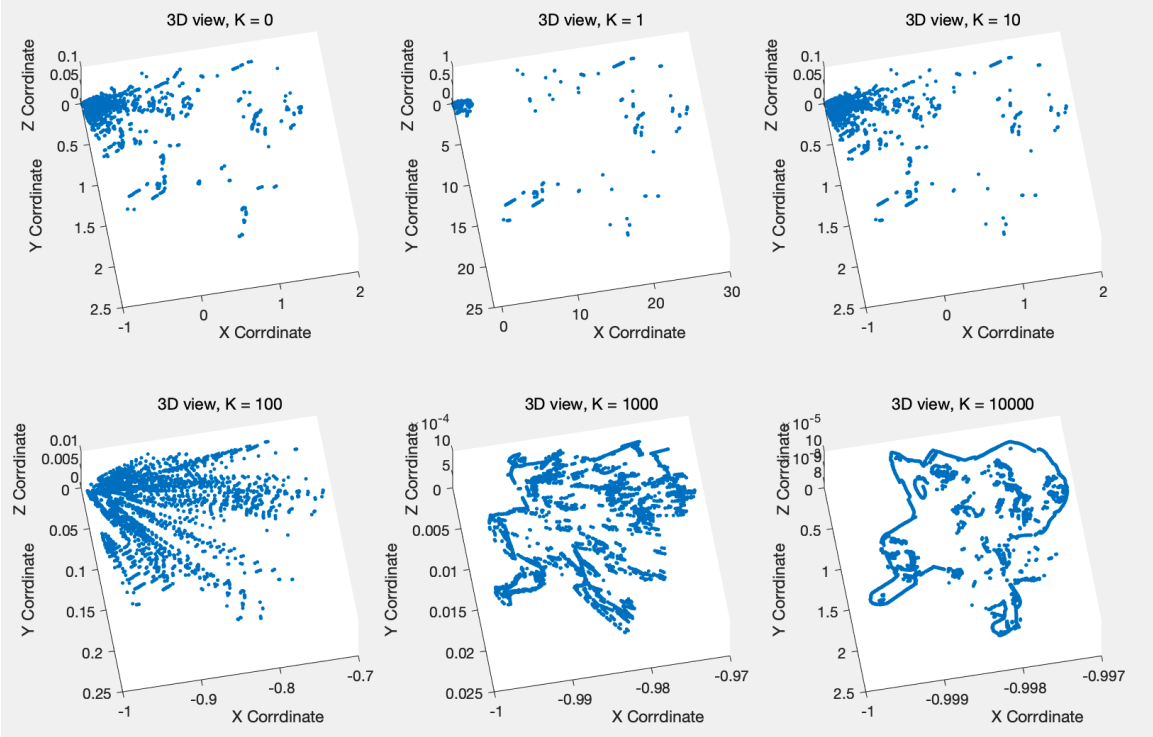


From the 3D image, we can see the main outline of the object, but most of the information is missing below, and the edge is also broken. This should be due to the mismatch of related features or the lack of edge extraction. I also generated a side view and a top view of the image. As shown in the figure below, we can see that the main points are clustered together and the main structure cannot be observed.



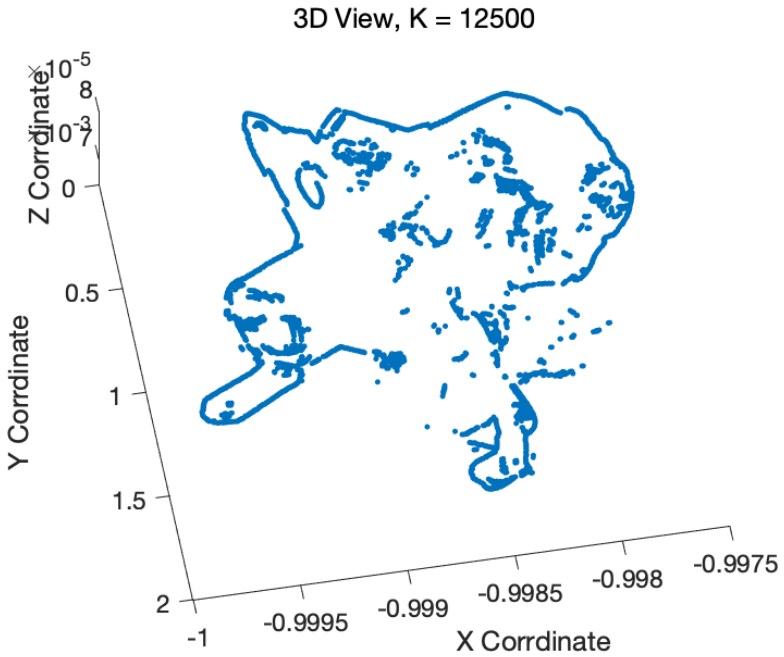
2.4 Adjust parameter K

In order to find the appropriate value of the parameter K, I set the value of K from 0 to 10000 and output different 3D images to study the effect of the K value on the distribution in the 3D image. The following figure shows the different results.

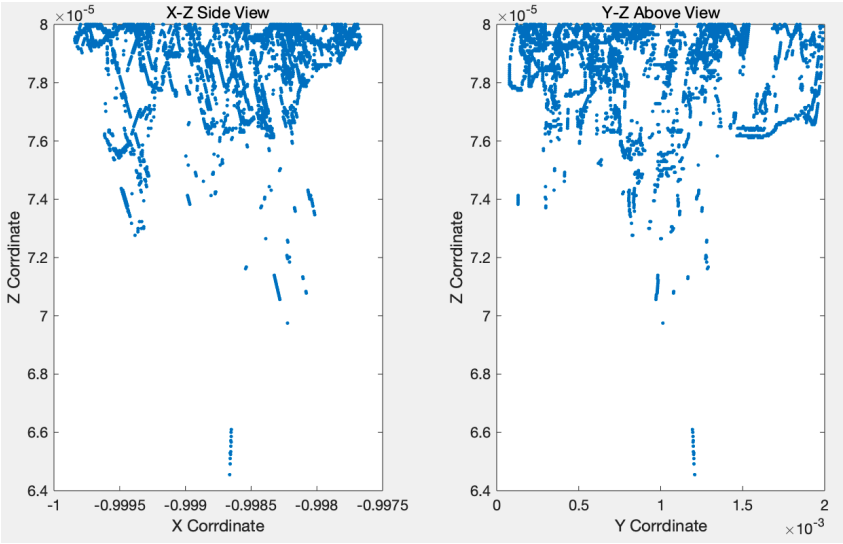


From the figure, we can see that when K is 0, all the points tend to gather at one point. As the value of K increases, the outline of the object becomes more and more obvious. As the Z coordinate is proportional to 1.0/ (K + disparity), the K value controls the scale of Z coordinate.

Through experiments, I found that when the K value is around 12500, the image restoration is better, and when the K value is greater than 12500, the point distribution is too scattered.



The following image is the top view and side view when the K value is 12500.



2.5 Thinking about the calculation of Z coordinate

After referring to some papers, I found that when calculating the Z coordinate, I can introduce an overall scaling factor in the above Z coordinate calculation formula. Thus, the Z coordinate calculation formula becomes:

$$Z = \frac{1}{K + disparity} * \alpha = \frac{bf}{disparity}$$

In the above formula, b represents the baseline and α is the overall scaling factor. So we can calculate α by the following formula.

$$\alpha = bf(\frac{K}{disparity} + 1)$$