Part 1: Feature Detection
For this part, I chose SURF detector with threshold = 400 to detect keypoints and compute descriptors. The problem I found for is that some keypoints are not unique enough and may lead to wrong matching in the next stage. For example, some points are not in the corner but at the edge of the object, so it's not that easy to differentiate it with other.

Part3: Outlier Rejection
For this part, I calculate a fundamental matrix from the corresponding matching points in left and right image. And I chose RANSAC algorithm as a method parameter. As a result, I will get a mask for the previous match to get all inliers.
I also consider choosing find homography method, but that one is only suitable for matching in the same plane. But our test case requires a dense stereo matching.

To find fundamental matrix, the opencv used formula $[p_2; 1]^T F [p_1; 1] = 0$. where F is a fundamental matrix, p_1 and p_2 are corresponding key points in the left and right images. There are four method to calculates this fundamental matrix:
CV_FM_7POINT for a 7-point algorithm.
CV_FM_8POINT for an 8-point algorithm.
CV_FM_RANSAC for the RANSAC algorithm.
CV_FM_LMEDS for the LMedS algorithm.
Because we want to remove the outlier, RANSAC algorithm is the most suitable choice among them
As for RANSAC, it's mainly an iterative algorithm to get the best inlier set. At initialization, given a model, find the minimum number of points N required to determine the model parameters. Then we go inside the loop and randomly select N samples and solve the model. And then check how many samples from the rest of your data fits the model, which is inliers and we set C as the number of inliers. If C > inlier ratio threshold or we reached the maximum number of iterations, we will stop the iteration and return the best inlier set. Otherwise, we repetitively randomly select N samples and compute the model, find inliers. Finally, we recompute model parameters using the best inlier set.

The method I used is majorly F, mask = cv.findFundamentalMat (img_points_left, img_points_right, method= cv.FM_8POINT+cv.FM_RANSAC, ransacReprojThreshold=ransacReprojThreshold, confidence=confidence). To get better performance, I tuned the parameter ransacReprojThreshold and confidence. My measurement metric is correct rate of estimating depth value within certain error range for matching points(correct number/total numbers). As a result, when ransacReprojThreshold = 4 and confidence 0.99 I got the highest correct rate.(I also tried all combination of ransacReprojThreshold = 3-5 and confidence = 0.85-0.99)