Advanced Geometric Project Report

Wanting JIN and Xirui LIU

January 17, 2019

Tools: Matlab, Computer Vision toolbox, KITTI data, KITTI development kit

1 Needed Data

$image\_00data$: the images captured by grey scale camera 00
$calib\_cam\_to\_cam$: the projection matrix after rectification of camera 00

2 Data Processing

According to the $'readme.txt'$ file, the meaning of each parameter in $'calib\_cam\_to\_cam.txt'$ are shown below.

- K_xx: 3x3 calibration matrix of camera xx before rectification
- S_rect_xx: 1x2 size of image xx after rectification
- R_rect_xx: 3x3 rectifying rotation to make image planes co-planar
- P_rect_xx: 3x4 projection matrix after rectification

Matrix P_rect_0i means the projection matrix of centre of camera 00 in camera $0i$, with all cameras are rectified. Since the data we used are already synced and rectified, the given matrix K_00 is useless. The real calibration matrix of camera 00 and matrix P_rect_00 should respect following relationship.

$$P\_rect\_00 = K_0[I|0] \tag{1}$$

3 Personal Codes

We wrote Harris corner detection **function [corner,corner_count]= Harris_point_detection (im,mask_type,window_size)** which has three inputs: raw image, type of masks and window size of Gaussian filter. This function returns a binary matrix in which the elements that are detected as corners are assigned with value 1 and the number corners is accounted.

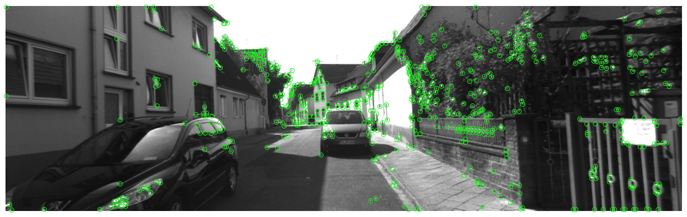


Figure 1: Harris points using Prewitt mask

There are three kinds of masks that could be implemented, Gradient, Sobel and Prewitt. The result by using "Prewitt" masks with window size as 3 can be seen in the following:

The match correlation function is **[match1,match2,matchIndex] = match_correlation (im1,im2,cor1,cor2,cnt1,cnt2)** . It requires two images with corners' index and number of corners as input. And returns the location of matched corner points in both images and the index of matched corners. The distance measurement function is Sum of Squared Differences (SSD). The threshold of correlation is a user-defined parameter which can be tuned to find better matches. The result of matches can be seen in following:

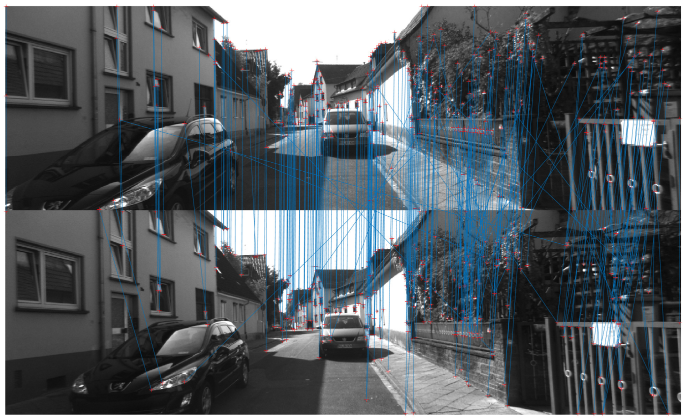
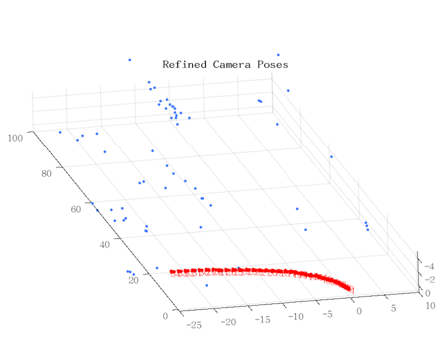


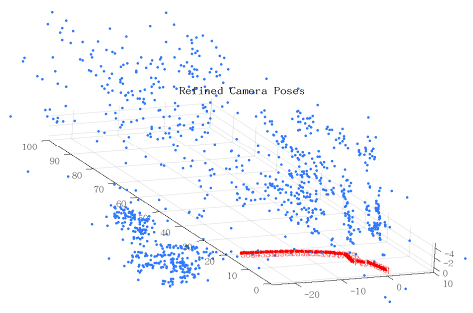
Figure 2: Matches using SSD as correlation function

4 Following steps using toolbox

A **viewSet** is created to save all the views. The first view is added into **viewSet** at the origin of global frame at beginning.
Robust estimation of the essential matrix is achieved by using function **helperEstimateRelativePose**. At first, the function calculates essential matrix using MSAC while ensuring the number of inliers are large enough. Then, retrieve relative camera pose from essential matrix and the current camera pose in the global can be calculated recursively. At the time, the matches between adjacent images are restored into **viewSet**.
Before applying triangulation, the point tracks across all view are obtained by function **find-**



(a) Before bundle adjustment



(b) After bundle adjustment

Figure 3: 3D points and camera poses

Tracks. Then applying **triangulateMultiview** function, a set of 3D points are calculated. However, those 3D points are not refined yet. In other word, the reprojection error of those 3D points are very large.
In order to refine those 3D points and camera poses, it is reasonable to apply bundle adjustment now. The reprojection errors are reduced significantly to small values. The difference of point clouds before and after bundle adjustment is shown in figure

5 Dense reconstruction

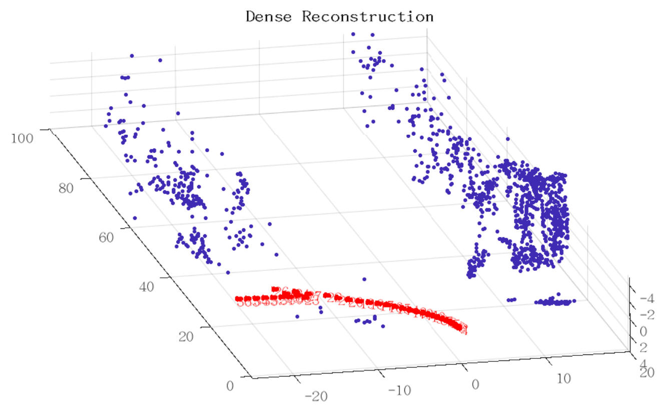


Figure 4: 3D points and camera poses in dense reconstruction

In order to perform dense reconstruction, we go though all the images, detect a dense set of corners and repeat the other steps above again. The result of dense reconstruction is shown in figure 4.