Assignment 3 Report

Stereo Reconstruction and Non-Linear Optimization

Puru Gupta(20171187) , Meher Shashwat Nigam(20171062)

Question 1 (q1/Q1.ipynb)

Given information :

- A set of rectified and synchronized stereo image pairs from a KITTI sequence *(q1/img2 and q1/img3)*
- The associated calibration data *(calib.txt)*

  K :

  [7.070912e+02, 0.000000e+00 , 6.018873e+02]

  [0.000000e+00, 7.070912e+02, 1.831104e+02]

  [0.000000e+00, 0.000000e+00,1.000000e+00]

  Baseline:

  0.53790448812 m

- The absolute ground truth poses of each stereo pair *(poses.txt)*

Objective :

Given a sequence of 21 rectified stereo image pairs, generate a dense 3D point cloud (dense refers to all points being used for reconstruction in contrast with sparse).

Procedure:

Step 1 : Generating disparity map for each stereo pair

We will use SGBM (semi-global block matching, using **cv::StereoSGBM**) for calculating disparity for each stereo pair (rectified and undistorted).

SGBM parameter values used:

window_size = 5
min_disp = -39
num_disp = 144
disp12MaxDiff = 1,
blockSize = 5,
*P1 = 8 * 3 * window_size ** 2,*
*P2 = 32 * 3 * window_size ** 2,*
uniquenessRatio = 10,
speckleWindowSize = 100,
speckleRange = 32,
preFilterCap = 63

Step 2 : Generating point clouds in camera frame

Using the generated disparity map for each stereo pair, parameters like focal length and baseline information we project the points into 3D, ignoring invalid disparity values.

We multiply the parallax map : **[x, y, disparity, 1]** with **Q**

Where **Q = [[1, 0, 0, -l/2],**

[0, 1, 0, -w/2],

[0, 0, 0, focal_length],

[0, 0, -1/baseline, 0]] .

Here x, y are the image coordinates and disparity is the corresponding disparity. L and W refers to the dimensions of the image.

The 3D points obtained from multiplying Q and the parallax map are in the camera frame.

Step 3 : Registering all the point clouds obtained into the world frame

From poses.txt we get the (ground truth poses) orientation([R|t]) of successive camera frames in the world frame. We then transform each point cloud from their frames into the common world frame and make an aggregated point cloud. We use **Open3d** for storing our point clouds.

Step 4 : Visualizing the point cloud using open3D

Output :

output.mp4 (video from projected images)

img{460-480}.png (output images)

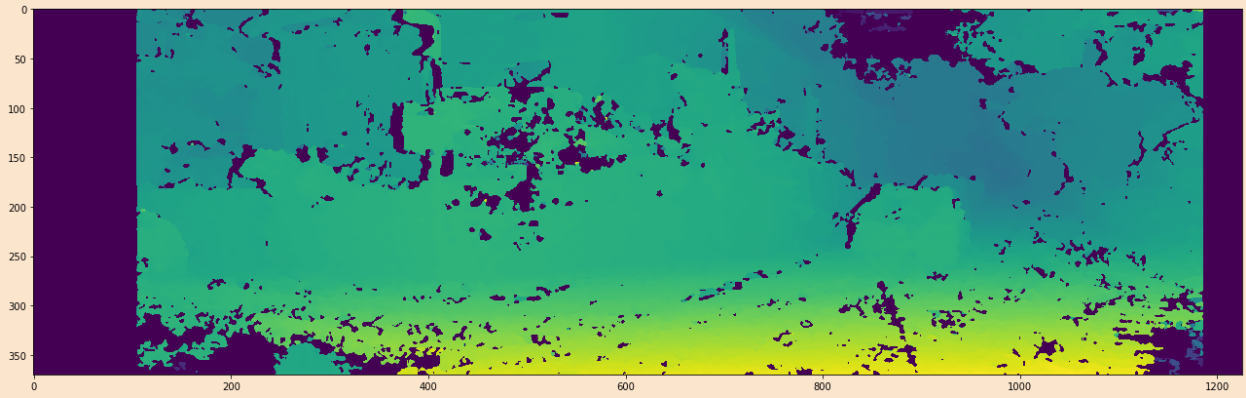
[reconstructed img463.png]

Point_cloud.pcd (aggregated point cloud)

BONUS:

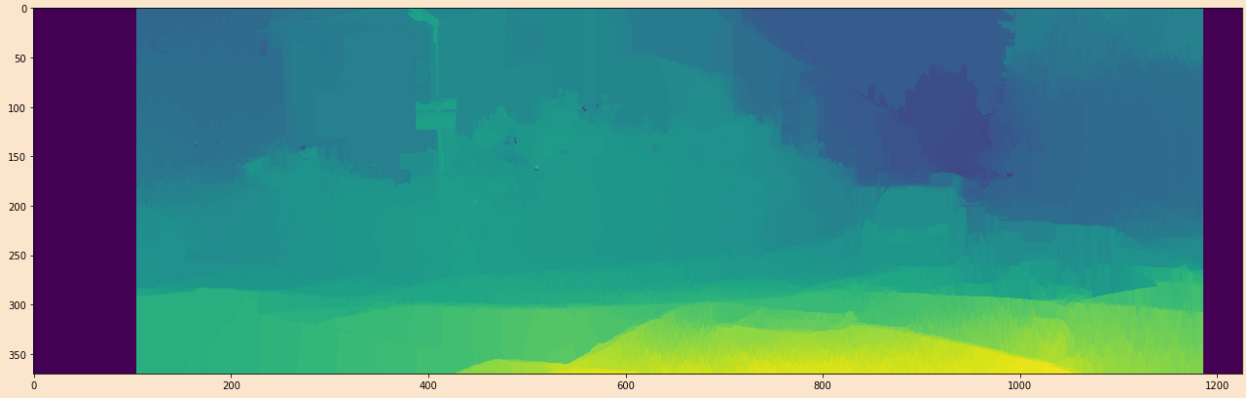
Disparity Maps for different methods:

1. SGBM



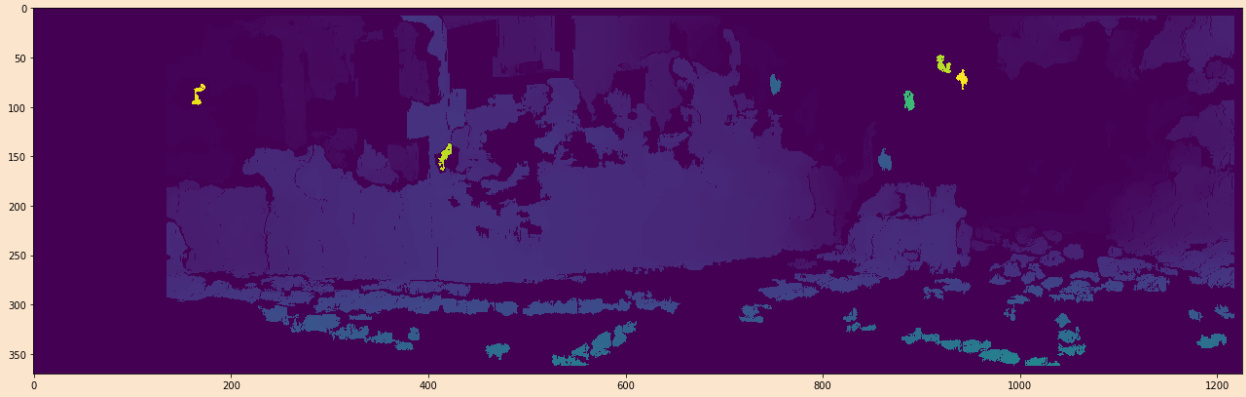
Semi Global Block Matching appears to produce an unclear map but one of the most detail preserving map.

2. SGBM with WLS Filtering



Semi Global Block Matching with WLS filtering appears to produce the most clear map but a lot of details have been lost due to smoothing of the object boundaries in the map.

3. BM



Block Matching is comparatively faster than other algorithms, but fails to provide good disparity matching as seen in the above image.

Overall, SGBM alone provides the best suited disparity map for 3D point cloud generation.

Question 2 (q2/q2.ipynb)

Given information :

- Generated 3d reconstruction from question 1 (point_cloud.pcd)

Objective :

- Synthesize a new image taken by a virtual monocular camera fixed at any arbitrary position and orientation.
- Recover this pose using an iterative Perspective-from-n-Points (PnP) algorithm via Gauss Newton.

Procedure :

Step 1: Obtaining a set of 2D-3D correspondences between the image and the point cloud

We use the first stereo pair (img460.png) and the corresponding pose from poses.txt to get 2d-3d correspondences. We project the 3d point cloud using given K, R, t to get the corresponding image and the correspondences.

Step 2: Getting estimate of transformation matrix

We sample around 100 points from the obtained set of 2d-3d correspondences to get an estimate of the transformation matrix using DLT.

Step 3: Refining the estimate of transformation matrix using iterative PnP (gauss newton method)

The initial estimate is refined further using gauss newton and the error in reprojection is reduced below a certain threshold. Final reprojection error was obtained as 1.22e-09 after 1000 iterations.

If we take a random initialization(np.random.uniform) for the transformation matrix, gauss newton doesn't converge or stops at some local minimum.

- np.random.uniform(10e-03,-10e-03,12)).reshape((3,4)) -> led to a local minima with final error as 2922.3 after 7 iterations.
- ([10e-03,10e-03,10e-03,10e-03,10e-03,10e-03,10e-03,10e-03,10e-03,10e-03,10e-03,10e-03]).reshape((3,4)) again stuck in a local minima with error 2922.322, after 675 iterations.

We tried many different initializations, only a good initialization like the one from DLT converged to a very low reprojection error.

[Projected image from the estimated transformation matrix from gauss newton]

Final [R|t] matrix obtained:

[[3.29753302e-02 -1.50510096e-03 -1.43431566e-02 6.95553111e+00]

[-1.92005481e-03 -3.59297528e-02 -6.47758754e-04 -2.57308472e-01]

[1.48027117e-02 -1.47812198e-03 3.27679567e-02 1.00000000e+00]]

Projected points = $K[R|t]X = TX$ (transformation matrix)

$= \pi(X)$

$\varepsilon = x - \pi X'$

$\frac{\partial \varepsilon}{\partial T} = \frac{\partial}{\partial T}\pi(X')$

$= \frac{\partial}{\partial X'}\pi(X')\frac{\partial X'}{\partial T}$

$x' \rightarrow$ projected points

$x' = [R|t]X$

$\pi(x') = \begin{bmatrix} f & 0 & t_x \\ 0 & f & t_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} fx' + t_x z' \\ fy' + t_y z' \\ z' \end{bmatrix} = \begin{bmatrix} f\frac{x'}{z'} + t_x \\ f\frac{y'}{z'} \\ 1 \end{bmatrix}$

$\frac{\partial \pi(x')}{\partial x'} = \begin{bmatrix} \frac{f}{z'} & 0 & -\frac{fx'}{z'^2} \\ 0 & \frac{f}{z'} & -\frac{fy'}{z'^2} \end{bmatrix}$ (2×3)

$x' = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

$= \begin{bmatrix} r_{11}x + r_{12}y + r_{13}z + t_1 \\ r_{21}x + r_{22}y + r_{23}z + t_2 \\ t_{31}x + r_{32}y + r_{33}z + t_3 \end{bmatrix}$

$\frac{\partial x'}{\partial T} = \begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x & y & z & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & y & z & 1 \end{bmatrix}$

$P = K[R|t]$

$p = P.\text{reshape}(12,1)$

$p^{i+1} \leftarrow p^i - \text{update}$

$J = -\left(\frac{d\pi(x')}{dx'}\right)\left(\frac{dx'}{\partial T}\right)$ (2×3) (3×12) $= (2\times12)$ matrix → stackall → $(2N \times 12)$

$\text{update} = (J^T J)^{-1} J^T \varepsilon$

$\varepsilon \Rightarrow (2N \times 1)$ matrix

12×1 matrix