SLAM – EX2

Git: https://github.com/Miryam-Schwartz/SLAM/blob/main/VAN_ex/code/ex2.py

Miryam Schwartz, miryam.schwartz@mail.huji.ac.il

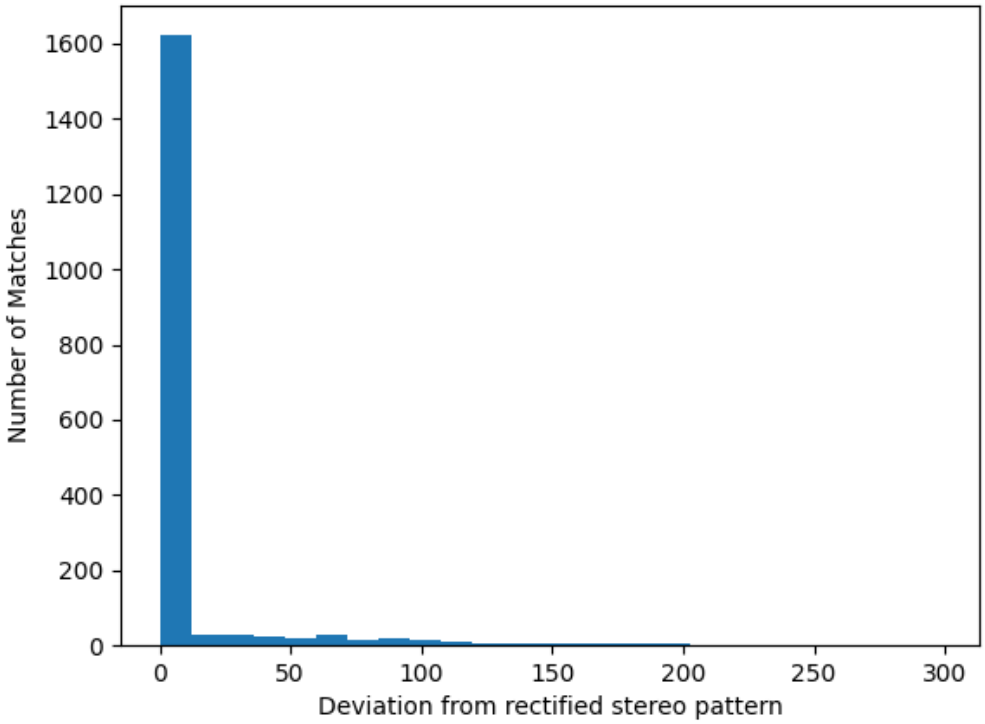
Nava Goetschel, nava.goetschel@mail.huji.ac.il

In this exercise we explore a geometric outlier rejection policy and use the stereo matches for triangulation to produce a 3D point cloud.

2.1. In this part, we worked with a pair of rectified stereo images.

- The special pattern of correct matches on such images: A straight line connects the matching points, The y values are the same. This is because the matches are between aligned stereo images.

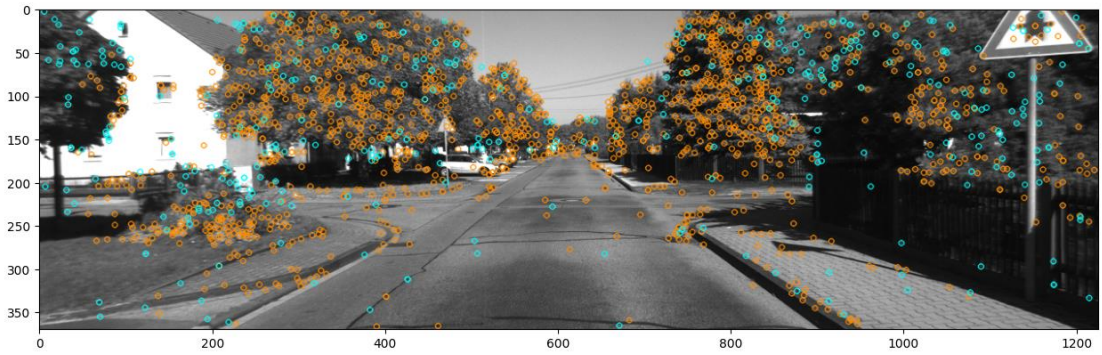
- a histogram of the deviations from this pattern for all the matches:



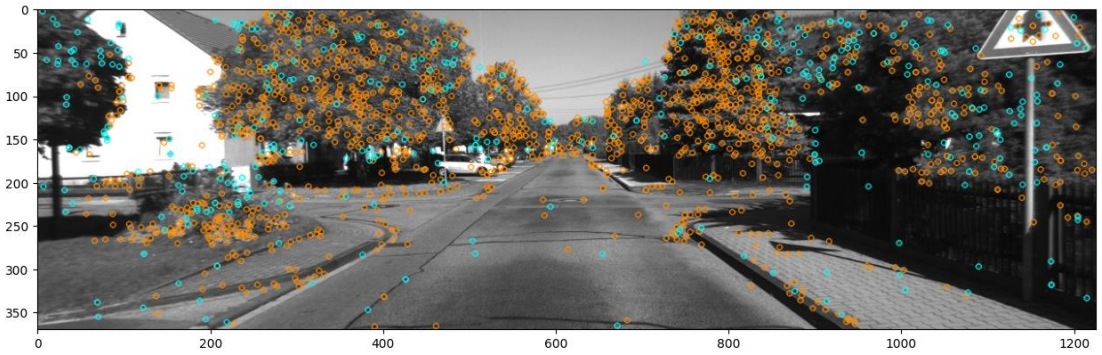
- the percentage of matches that deviate by more than 2 pixels: 16.20

2.2. In this part, we used the rectified stereo pattern to reject matches.

- The images bellow presents the resulting matched, accepted matches (inliers) in orange and rejected matches (outliers) in cyan. We used threshold value = 1
    - Left image:



- Right image:

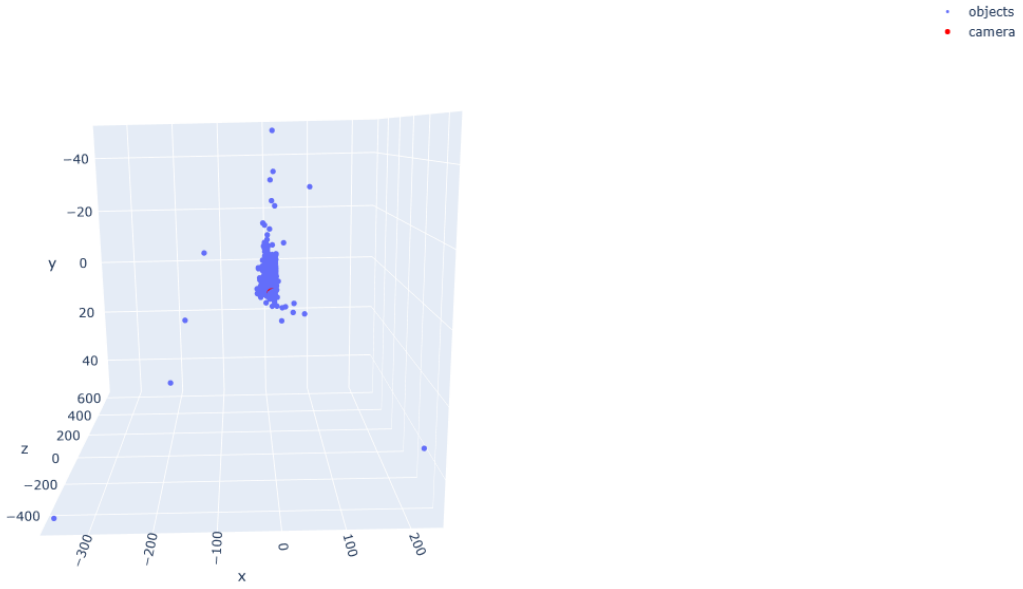


- 374 matched were discarded by the rectified test (from 1858 matches at all)

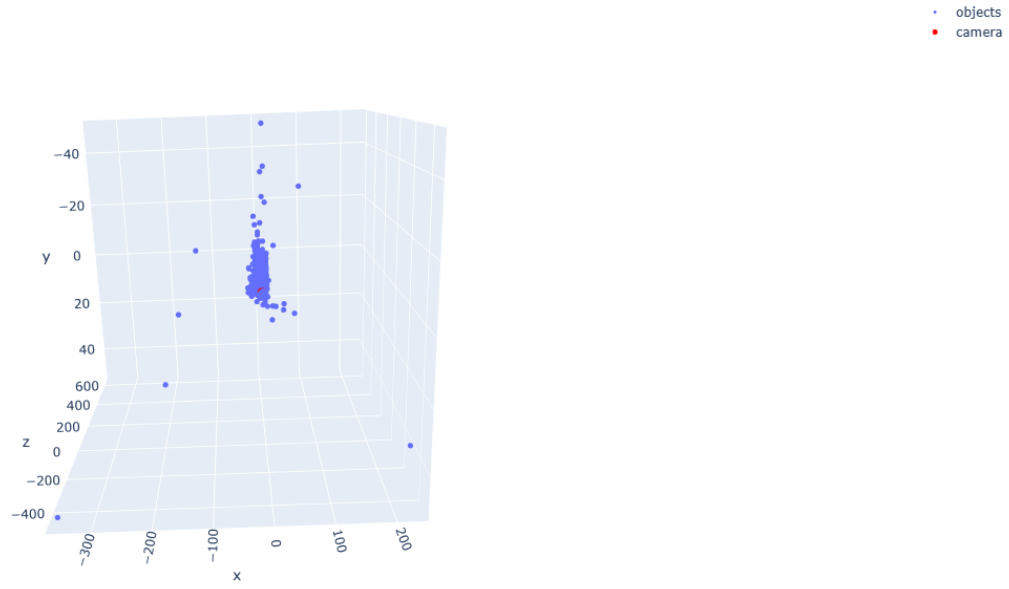
- Assuming the Y-coordinate of erroneous matches is distributed uniformly across the image, we expect that the ratio $\frac{accepted\ matches\ (inliers)}{all\ matched}$ will be
$$\frac{threshold\ value}{number\ of\ rows\ pixels\ in\ the\ image} * number\ of\ matches$$
Because for each match the probability to be accepted by the test is the probability that the difference between the y values will be smaller than the $threshold\ value$.
- This assumption (uniform distribution along the Y-axis) is not realistic, because the images are aligned, that is, identical objects should appear at the same height (same position on the y-axis) in both images (because the cameras are at the same height, and only in different positions on the x-axis). So even if there is some error in the alignment, it will be very small and therefore the differences between the y's will be very small and will not be distributed uniformly on the entire y-axis but only on a limited range of the y-axis.
Therefore, we expect that the number of $rejected\ matches\ (outliers)$ will be smaller and therefore the number of $accepted\ matches\ (inliers)$ will be higher, so we get that the ratio $\frac{accepted\ matches\ (inliers)}{all\ matched}$ will be higher.

2.3. In this part, we used the matches and the camera matrices to define and solve a linear least squares triangulation problem.

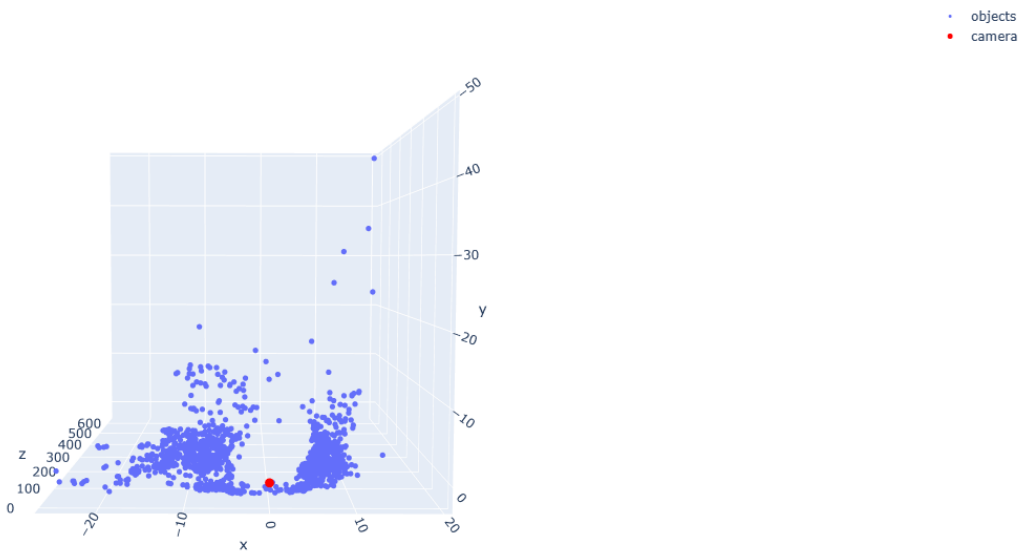
- A 3D plot of the calculated 3D points (using our triangulation function): click here to open interactive plot



- A 3D plot of the calculated 3D points (using cv2 triangulation function): click here to open interactive plot

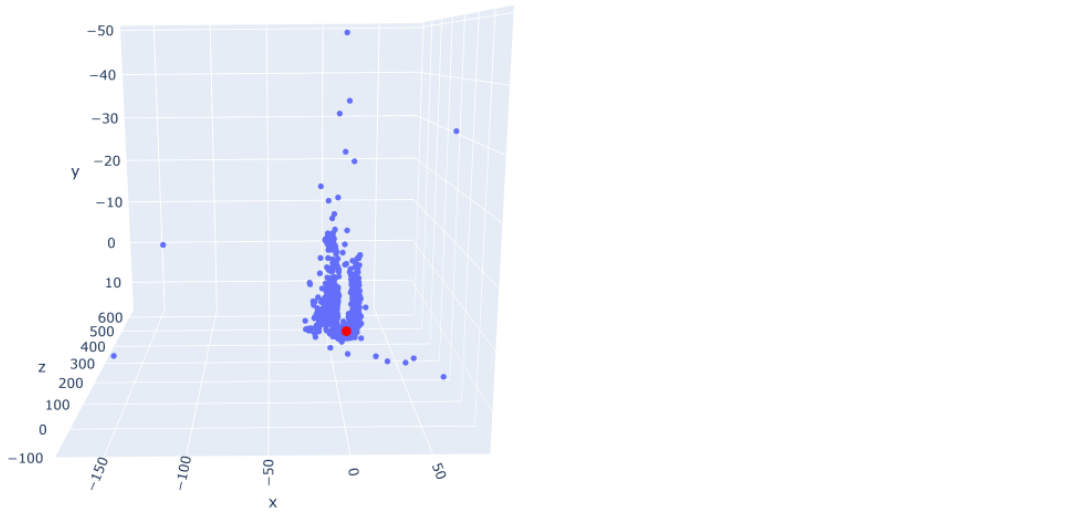


  - The median distance between the corresponding 3d points: 6.341926411336984e-14
  - We want to note that when we also applied the triangulation over points that passed the rectified test and we got a clearer plot: click here to open interactive plot



2.4. In this part, we apply matching and triangulation over a few pairs of images (img500, img1000, img1500, img2000, img2500).

- We can notice that all the points that received a negative z value have an erroneous location (because that apply that this points are behind the camera)
- We think that the reason for the error is because normally, we expect each point in the right image to be more left-handed than its corresponding point in the left image (right_x < left_x).
  However, when there is an error in some match, i.e., the X value of the point in the right image is more right-handed than in the left image, then, the ray leaving the right camera to the image plane and the ray leaving the left camera to the image plane are cut behind the camera, due to the deviation in the X-axis. Since the camera is the origin of the axes in the world coordinate system, a point behind it will get a negative Z value.
- A relevant criteria for outlier removal can be to threw away the matches where the point in the right image, is more right than in the left image (=right_x >left_x).
  When we apply it, we got the next plot (almost without negative Z values):


objects
camera
−50
−40
−30
−20
y
−10
0
10
600
500
400
z
300
200
100
0
−100
−150
−100
−50
0
50
x