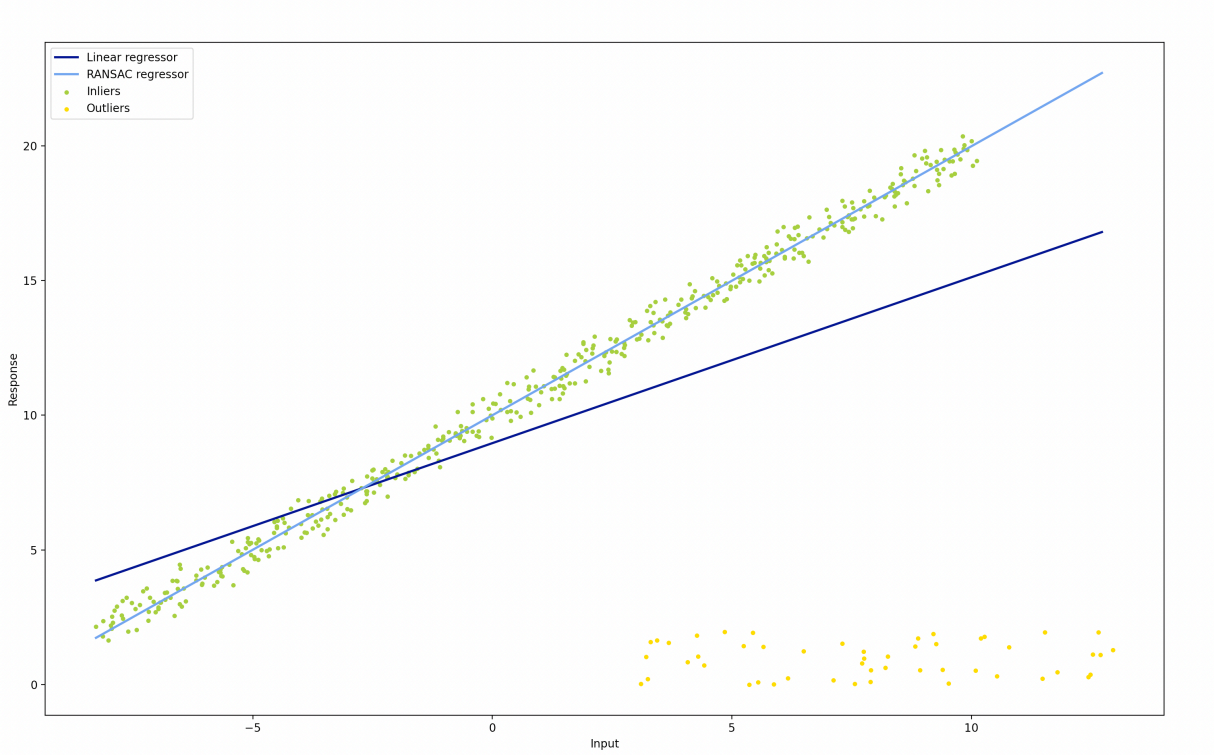
COMPUTER VISION - ASSIGNMENT 4 REPORT

MODEL FITTING

The model fitting task consisted in the implementation of the RANSAC algorithm to fit a linear model among a dataset with noisy observations and outliers. The proposed algorithm manages to fit a line among the observation and seems to correctly detect the inliers and the outliers. The algorithm follows the implementation proposed in paragraph 2.12 of the assignment. In particular, each point $\boldsymbol{p = (x , y)}$ is labeled as an outlier whenever its distance from a candidate line $\boldsymbol{l}$ overcomes a certain threshold $\boldsymbol{t}$. The Euclidean distance between the point and line, arising from the orthogonal projection of the point $\boldsymbol{p}$ onto the line $\boldsymbol{l}$ is calculated as a distance metric.

The implementation of the RANSAC algorithm yields the following results:



	K	B
GROUND-TRUTH	1	10
LEAST- SQUARES	0.615965657875546	8.96172714144364
RANSAC	0.964389494656899	9.98292129496683

Differentiable Warping

For a pixel p in the reference image , given a certain depth hypothesis value d_j, its corresponding pixel $p_{i,j}$ *i*n the source image I_i is computed as:

$$p_{i,j} = K_i * \left(t_{0,i} + R_{0,i} * \left(K_0^{-1} * p * d_j \right) \right)$$

where:

- K_0 and K_i are the calibration matrices for the reference image and the source image I_i
- R_0, i and t_0, i are respectively the relative rotation and the relative translation between the reference image frame and the source image I_i frame.

To obtain this equation, we can proceed as follows:

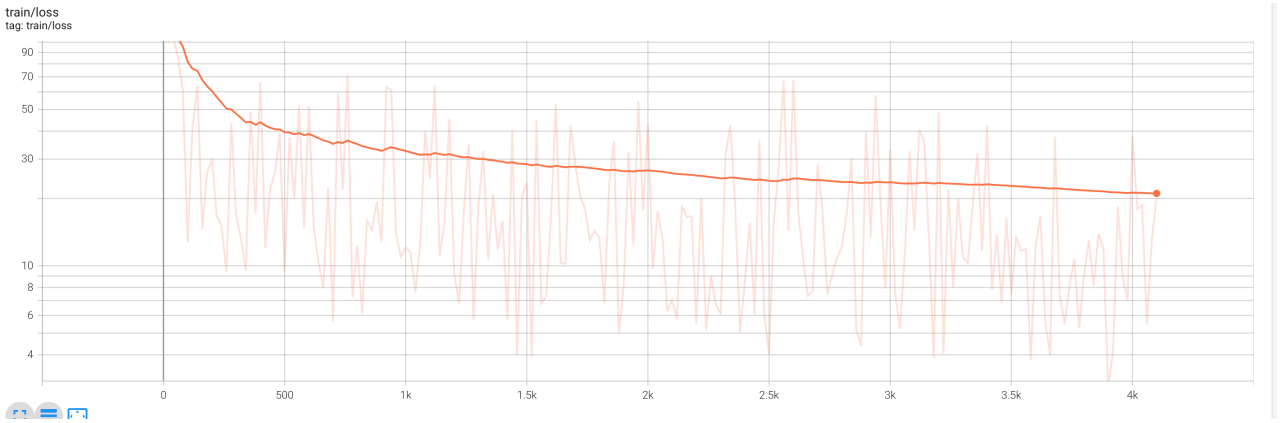
- We first express p in normalized image coordinates, by multiplying by K^{-1}.
- Then, in order to take into consideration the hypothesis depth value, we multiply the result by d_j, thus forcing the 3D point to have depth d_j
- Finally, we make use of the intrinsic and extrinsic parameters of the source view camera. In particular, we first rotate and translate the 3D point, using the relative pose of the source view camera with respect to the reference view, and then we project it onto the source view image, via the calibration matrix K_i

This operations is performed for a set of depth values sampled between DEPTH-MIN and DEPTH_MAX.

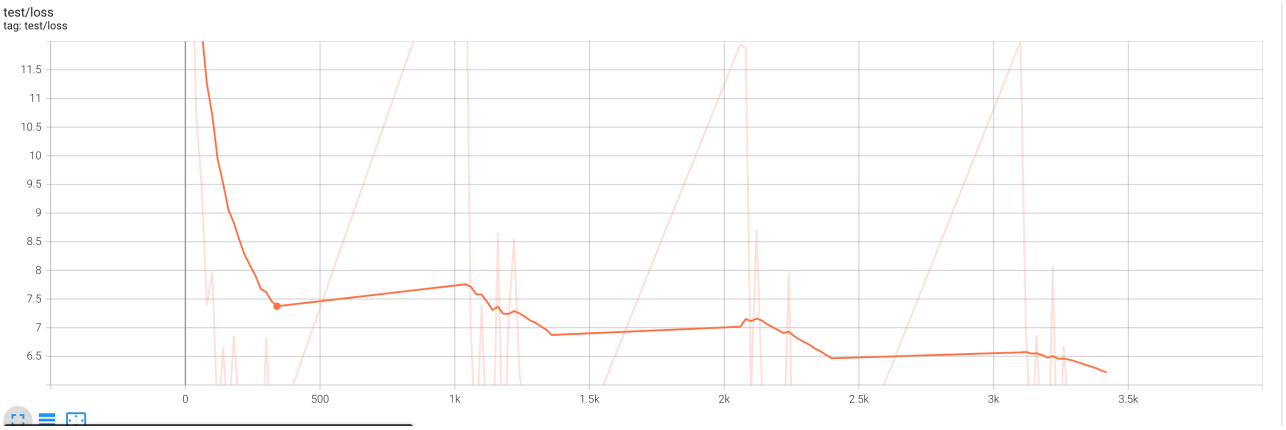
Training and Testing

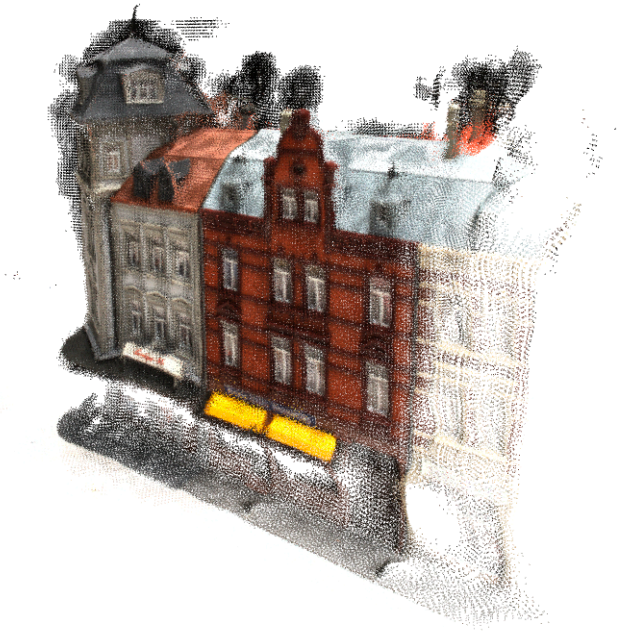
After having launched the training and testing (18 hours on CPU), I obtain the following trends on Tensorboard. The smoothing parameter of Tensorboard was set to one in order to plot a clear convergence of the loss function.

Training loss



Testing loss





Here are the plotted point cloud is reconstructed after **filter depth estimation.**

Geometric Consistency Filtering

The geometric consistency filtering aims at measuring the consistency among multiple views. For a point p in the reference image we are able to project such point in a pixel p_i in another view using the estimated depth d. Similarly, we are able to reproject p_i back to the reference image via estimated depth d_i. In this way we are able to obtain a reprojected point p_r, and his depth d_r, on the reference image. The next step is to evaluate the **euclidean distance** between p and p_r as well as the **relative difference** between d and d_r**.** The depth estimation is consistent if:

- The Euclidean distance between p and p_r is lower than 1
- The relative difference between d and d_r is lower than 0.01

Moreover we require that all depths' estimation should be consistent for at least 3 views.

Questions

1. Sampling depths in an inverse depth range [1 / DEPTH_MAX , 1 / DEPTH_MIN] should yield better and **more robust results for large-scale scenes**. For points at infinity, a direct sampling will cause numerical instability, thus failing to deal with deep points in the image. Sampling between the inverse depth range avoids dealing with numerical issues.

2. Occlusion is basically an unsolved problem in Multi-View Stereo and every algorithm should take into consideration this issue. Our algorithm should be quite to robust given the sufficient amount of views used. It is highly unlikely that a point will be occluded in many views, provided that the position of the cameras differ significantly from each other. The robustness of the algorithm to the occlusion problem increases with the number of views at our disposal, since averaging is more effective.