Computer Vision Assignment 4

Saurabh Vaishampayan
svaishampaya@student.ethz.ch

November 2021

1 RANSAC

1.1 Results

The data is assumed to be generated by:

$$y = kx + b$$

The following is the table for the values of the parameters of the problem (ground truth, estimation by RANSAC, Estimation by Least Squares)

Algorithm/Parameter	k	b
Ground Truth	1	10
Least Squares	0.6159656578755458	8.961727141443642
RANSAC	0.9987449792570884	9.997009758791004

The following is the plot for the fitting:

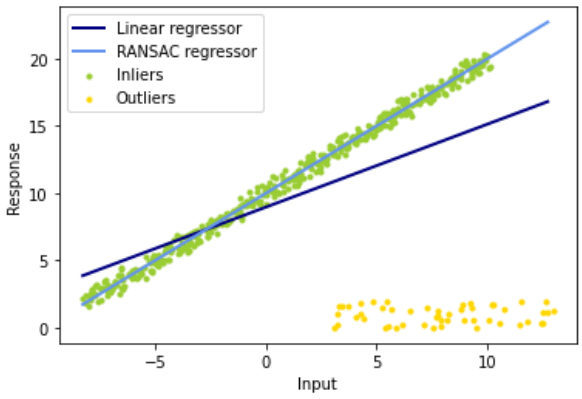

Figure 1: Plots for parameter fitting RANSAC vs Least Squares

RANSAC assumes that the data is generated by a combination of outliers and gaussian noise-corrupted true values. It applies Least Squares on a random subset of the data to estimate the true parameters. This is done for multiple random subsets. The estimated values are picked from that subset which gives maximum number of inliers for the estimated parameters. This is in contrast to Least Squares which assumes

all data is only corrupted by gaussian white noise (no outliers). Hence RANSAC is robust to outliers.

We can see from both the tables of parameter values as well as the fitting plots that the estimates given by the RANSAC routine are very close to the ground truth, because it reduced the effect of the outlier data points in estimation. The least squares estimates are quite far off from the true values, because the estimation is skewed by outlier points, since Least Squares estimation does not assume outliers. One can see that the slope of the estimated line is bent towards the outliers, while for RANSAC the line follows the true points closely.

2 Multi View Stereo

2.1 Differentiable Warping: Pixel correspondence given depth

Given a pixel $\mathbf{p}$ in the reference image and a depth d_j, our task is to find the corresponding pixel locations $\mathbf{p_{i,j}}$ in the source view i. Then one can extract the features for each correspondence and compute the depth wise similarity, which is later regressed to estimate depth. We assume the reference and source projection matrices to be:

$$\mathbf{P_{ref}} = \begin{bmatrix} K_r R_r & K_r t_r \\ 0 & 1 \end{bmatrix} \tag{1}$$

$$\mathbf{P_{src}} = \begin{bmatrix} K_s R_s & K_s t_s \\ 0 & 1 \end{bmatrix} \tag{2}$$

We assume for now that we have access to $K_r, R_r, t_r, K_s, R_s, T_s$ for now. As we get to the final equation, it turns out that for the final solution, we only need access to $\mathbf{P_{ref}}, \mathbf{P_{src}}$.

The relation between pixel coordinates of the reference and the source given the depth (w.r.t reference) does not depend on the global coordinate syste. So without loss of generality, shift the coordinates such that the reference camera is now a canonical camera.

$$\hat{\mathbf{P}_{\mathbf{ref}}} = \mathbf{I_{4\times 4}} \tag{3}$$

$$\hat{\mathbf{P}_{\mathbf{src}}} = \begin{bmatrix} K_s \hat{R}_s & K_s \hat{t_s} \\ 0 & 1 \end{bmatrix} \tag{4}$$

We can show mathematically that

$$\hat{\mathbf{P}_{\mathbf{src}}} = \begin{bmatrix} K_s \hat{R}_s & K_s \hat{t_s} \\ 0 & 1 \end{bmatrix} = \mathbf{P_{src}} \mathbf{E_{ref}^{-1}} \tag{5}$$

Here $\mathbf{E_{ref}}$ is the 4×4 Essential matrix for the reference.
We are given a pixel $\mathbf{p} = (x_r, y_r)$ in the reference image. We first convert it to homogenous coordinates:

$$\mathbf{x_{ref}} = [x_r, y_r, 1]^T$$

Now we lift this point to the normalized image plane, after which we can apply the intuition of canonical pinhole camera.

$$\hat{\mathbf{x_{ref}}} = K_r^{-1} \mathbf{x_{ref}} = [x'_r, y'_r, z'_r]$$

The 3d point corresponding to $\hat{\mathbf{x}_{\mathbf{ref}}}$ located at distance d_j from the reference has the coordinates given by:

$$\hat{\mathbf{x}_{\mathbf{ref,j}}} = K_r^{-1}\mathbf{x_{ref}} = [d_j\frac{x'_r}{z'_r}, d_j\frac{y'_r}{z'_r}, d_j]$$

In homogenous coordinates let us denote this by $\hat{\mathbf{X}_{\mathbf{ref,j}}}$.

$$\hat{\mathbf{X}_{\mathbf{ref,j}}} = [d_j\frac{x'_r}{z'_r}, d_j\frac{y'_r}{z'_r}, d_j, 1]$$

However we know that $K_{ref,3,3} = 1$. Therefore $K^{-1}_{ref,3,3} = 1$, where $K_{ref,3,3}$ represents the bottom right element of the reference intrinsic matrix.
Hence $z'_r = 1$ and,

$$\hat{\mathbf{x}_{\mathbf{ref}}} = K_r^{-1}\mathbf{x_{ref}} = [d_j x'_r, d_j y'_r, d_j] = K_r^{-1}\mathbf{x_{ref}}d_j$$

The projection of the 3d point $\hat{X_{ref,j}}$ onto source camera will give us the corresponding pixel locations in the source image.

$$\hat{\mathbf{X}_{\mathbf{src,j}}} = \hat{\mathbf{P}_{\mathbf{src}}}\hat{\mathbf{X}_{\mathbf{ref,j}}} \tag{6}$$

But $\hat{\mathbf{X}_{\mathbf{ref,j}}}$ is the homogenous version of point $\hat{\mathbf{x}_{\mathbf{ref}}} = K_r^{-1}\mathbf{x_{ref}}d_j$.
Therefore

$$\hat{\mathbf{X}_{\mathbf{src,j}}} = \hat{\mathbf{P}_{\mathbf{src}}}(\mathbf{K_r^{-1}x_{ref}d_j})_{\mathbf{hom}}$$

Here $(K_r^{-1}\mathbf{x_{ref}}d_j)_{hom}$ refers to homogenised version of $(K_r^{-1}\mathbf{x_{ref}}d_j)$.
To simplify further, let us extend $\mathbf{x_{ref}}d_j = [x_r d_j, y_r d_j, d_j]$ to 4D coordinates $\mathbf{x_{ref,d_j,4d}} = [d_j x_r, d_j y_r, d_j, 1]$. Hence the final pixel value simplifies to

$$\hat{\mathbf{X}_{\mathbf{src,j}}} = (\hat{\mathbf{P}_{\mathbf{src}}}\mathbf{K_r^{-1}})\mathbf{x_{ref,d_j,4d}} \tag{7}$$

But we know that

$$\hat{P_{src}} = \mathbf{P_{src}}\mathbf{E_{ref}^{-1}}$$

and also,

$$P_{ref} = (K_r E_r)_{4d}$$

Here $(K_r E_r)_{4d}$ refers to 4d extension of $(K_r E_r)$. Therefore

$$P_{ref}^{-1} = (K_r E_r)_{4d}^{-1} = (E_r^{-1}K_r{-}1)_{4d}$$

Thus simplifying we get the following

$$\hat{\mathbf{X}_{\mathbf{src,j}}} = \mathbf{P_s}\mathbf{P_r^{-1}}\mathbf{x_{ref,d_j,4d}} \tag{8}$$

Here $\mathbf{x_{ref,d_j,4d}} = [d_j x_r, d_j y_r, d_j, 1]$.
Summarising, we just have to first convert the pixel coordinate to 3d hommogeneous coordinates, then multiply by desired depth d_j, then convert to 4d homogeneous coordinates and then simply apply $\mathbf{P_s}\mathbf{P_r^{-1}}$ to get the source pixel coordinates.

2.2 Training results and plots

Due to time and complexity constraints we could only run our training for 3 epochs. We provide the screenshots of the tensorboard dashboard and comment on the observations.

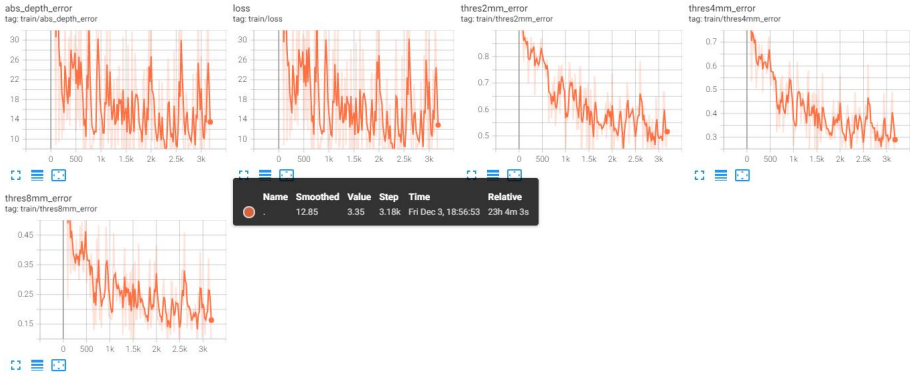

Figure 2: Plots for the training routine

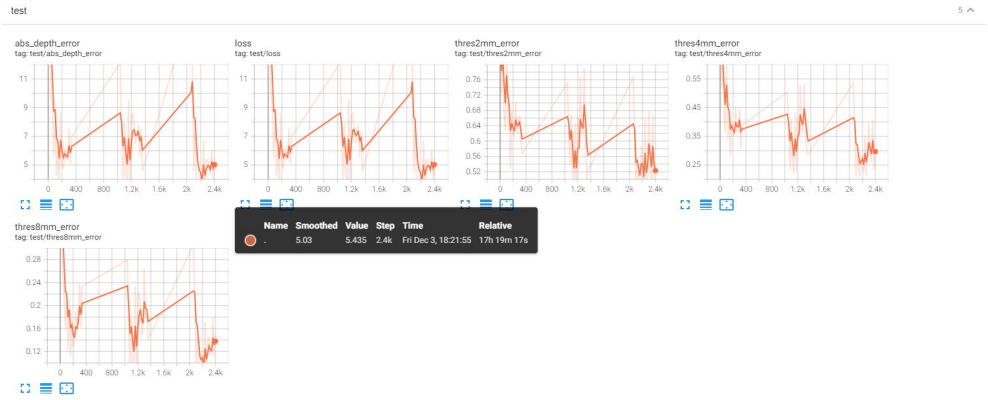

Figure 3: Plots for the validation routine

2.2.1 Screenshots

2.2.2 Conclusions on the training and validation process

Each train epoch has 1029 iterations. We have run the training for 3 epochs hence one can see that the time axis extreme value is just over 3000. In the plots, end of each epochs can be identified by steps of 1k in the time axis. In the training stage, the loss starts from a very high value (close to 70) initially but rapidly decreases in the first epoch itself. By the end of the first epoch, the loss reaches a value close to 20. After the first epoch the loss decreases but with a lesser rate. In the plots there are spikes and fluctuations in the loss functions but this is expected because of the small batch size and stochasticity of optimization.

A similar trend is observed in the plots for validation. The jumps are attributed to the start of a new epoch, when the optimizer starts to go again through the dataset again. The jumps can be attributed to a very quick convergence, and with a slower decay rate the spikes would have been smaller just after the beginning of each new epoch. The fluctutaions are fine, since the trend is that the average loss decreases with time consistently.

2.3 Testing

Due to time and complexity constraints we could only run our training for 3 epochs instead of 4. Therefore the output of the model checkpoint has the filename "model_000002.ckpt", and this must be modified in the eval.sh file.

2.3.1 Explanation of Geometric Consistency filtering

We have estimated the depths for each pixel in the reference. However, some of these values may be unreliable. In this function we perform a sanity check on our estimated depth output. We have estimated the depth for each pixel value in the reference image. Additionally, we know the relation between source and reference cameras.

Now we find the corresponding pixel locations in the source image. We have the estimated depth for each pixel value in the source image. Using this, we perform reprojection of the pixel and distance. That is, using pixel value (and estimated depth) in the source image we find the corresponding point and depth in the reference image.

If both the reprojected pixel coordinate and the reprojected depth in the reference are within the set tolerance level, then that particular depth value is geometrically consistent.

2.3.2 Visualisation: Scan for scene 1

(a) Perspective 1 (b) Perspective 2

Figure 4: Different perspectives for 3D point cloud for Scene 1

2.3.3 Visualisation: Scan for scene 9

(a) Perspective 1 (b) Perspective 2

Figure 5: Different perspectives for 3D point cloud for Scene 9

2.4 Answers to questions

1. Sampling depth in the inverse range: If depth is sampled uniformly in range (d_{min}, d_{max}), then the lowest depth resolution our network has available remains constant over all depths in the above range. The fractional error therefore is high for small depths and low for large depths. Typically, in scene reconstruction upto scale we are interested in the fractional error. Thus by the above method we are undersampling at small distances and oversampling at large distances. When we sample uniformly in the range $(1/d_{max}, 1/d_{min})$, the fractional error in depth resolution at depth d is:

$$\frac{\delta d}{d} = d\delta x$$

Here δx is difference between adjacent samples in $(1/d_{max}, 1/d_{min})$ and is constant. Therefore the relative minimum resolution grows with depth. given by.Thus this sampling is better at small distances than the previous, but worse off at large distances. The better choice will be to move to a log scale, where log(d) is sampled uniformly in $(log(dmin), log(dmax)$.

2. Integrating matching similarity by averaging will provide some robustness against occlusions, because we are. Moreover, in this case, since we have a softmax later, adding similarities can be interpreted as multiplication after softmax. This gives the interpretation of multiplying probabilities (of depth) from different source views, and the value will be high if more source images agree on the estimated depth (or alteratively, the pixel feature are highly correlated over multiple views for the given depth). An even better version will be to introduce majority voting w.r.t source view similarities to get more robustness against occlusion.