Camera Lidar Calibration

Team NERF-TM

Mitansh Kayathwal (2021101026)
Vineeth Bhat (2021101103)
P Gnana Prakash (2021111026)

Camera Intrinsic Calibration

- Started off with camera calibration without lidar owing to simplicity
- Used available datasets initially
- Also tested it on our own data
- Used Zhang's method

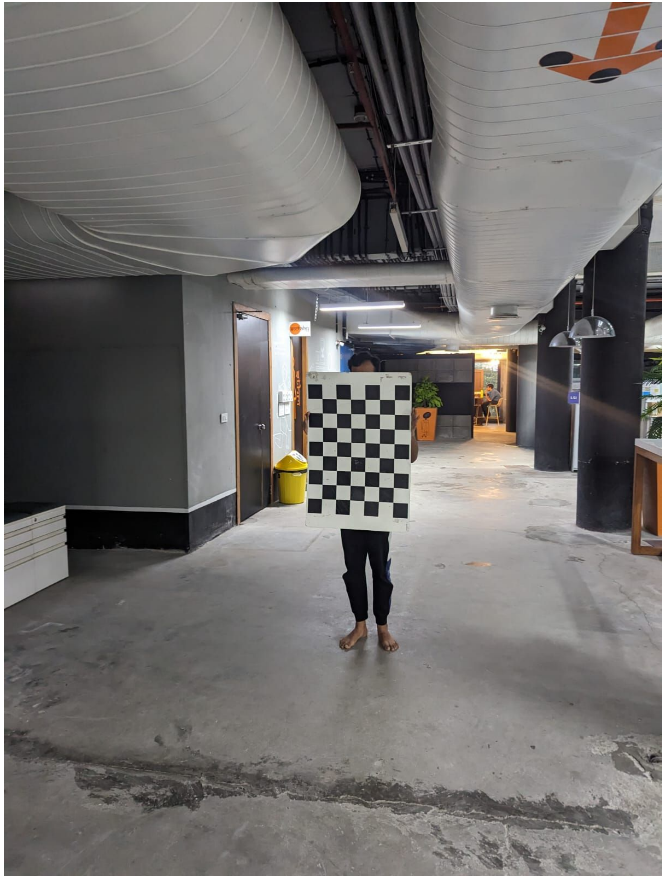


Fig: GP holding the checkerboard

Zhang's method

- Target based calibration (Checkerboard target)

- Set world coordinate system to a corner of the checkerboard. All points lie in the X/Y plane.

- We can delete the 3rd column of the extrinsics matrix

- We estimate a 3x3 homography instead of a 3x4 projection matrix

- Solving a system of linear equation leads to an estimate of H.

-

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} c & cs & x_H \\ 0 & c(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cancel{r_{13}} & t_1 \\ r_{21} & r_{22} & \cancel{r_{23}} & t_2 \\ r_{31} & r_{32} & \cancel{r_{33}} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ \boxed{\cancel{Z}} \\ 1 \end{bmatrix}$$

$$\begin{aligned} \boldsymbol{h} &= (h_k) = \mathrm{vec}(\mathsf{H}^{\mathsf{T}}) \\ \boldsymbol{a}_{x_i}^{\mathsf{T}} &= (-X_i,\ -Y_i,\ -\cancel{Z_i},\ -1, 0,\ 0,\ \cancel{0},\ 0, x_iX_i,\ x_iY_i,\ x_i\cancel{Z_i},\ x_i) \\ \boldsymbol{a}_{y_i}^{\mathsf{T}} &= (0,\ 0,\ \cancel{0},\ 0, -X_i,\ -Y_i,\ -\cancel{Z_i},\ -1, y_iX_i,\ y_iY_i,\ y_i\cancel{Z_i},\ y_i) \end{aligned}$$

Zhang's method (Continued...)

- Get the "**b**" vector from SVD using the last equation
- Then, when obtain the intrinsics from the Cholesky decomposition
- The extrinsics can be directly obtained from equations and the data we have till now (Homography matrix **H** and also the intrinsics **K** we have calculated
- These values serve as initial estimates for the LM algorithm from which we obtain better values for calibration

$$\mathrm{B} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{pmatrix}$$

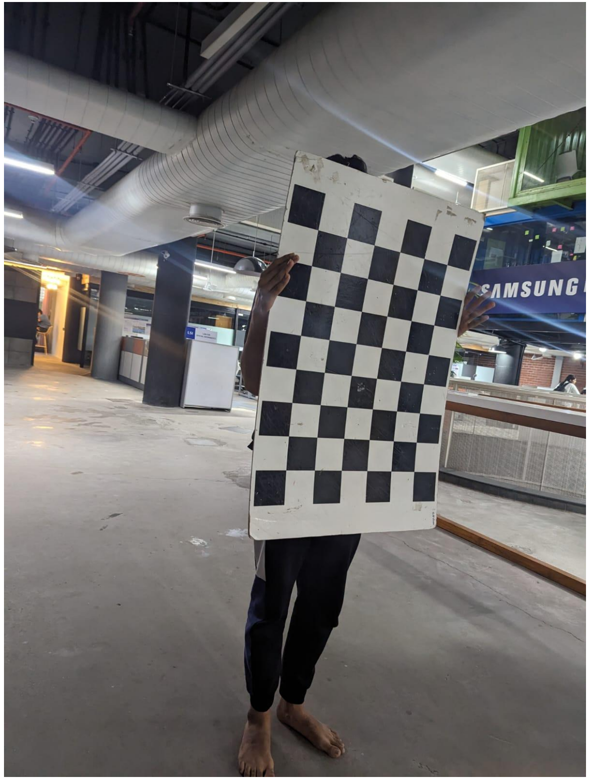
$$\mathrm{chol(B)} = \mathrm{AA}^T$$

$$\mathrm{A} = \mathrm{K}^{-T}$$

$$V = \begin{pmatrix} \boldsymbol{v}_{12}^T \\ \boldsymbol{v}_{11}^T - \boldsymbol{v}_{22}^T \end{pmatrix} \quad \text{with} \quad \boldsymbol{v}_{ij} = \begin{bmatrix} h_{1i}h_{1j} \\ h_{1i}h_{2j} + h_{2i}h_{1j} \\ h_{3i}h_{1j} + h_{1i}h_{3j} \\ h_{2i}h_{2j} \\ h_{3i}h_{2j} + h_{2i}h_{3j} \\ h_{3i}h_{3j} \end{bmatrix}$$

$$\boldsymbol{b}^* = \arg\min_{\boldsymbol{b}} \|V\boldsymbol{b}\| \quad \text{with} \quad \|\boldsymbol{b}\| = 1$$

Calibration Results



Calibration Results

Calibration Results

Extrinsic Calibration

- Used ROS simulations
- Added normal camera, Intel RealSense Camera, 2D and 3D “sic” lidar
- Manually annotated point clouds
- *Assumption:* Checkerboard pattern ends correspond to board’s ends (can be corrected later if needed)

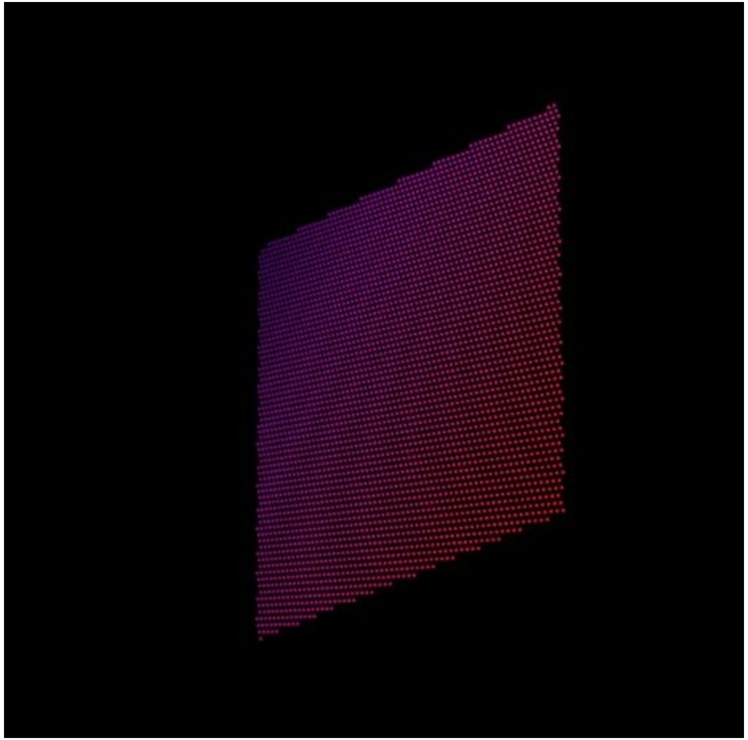


Fig: Point cloud of checkerboard

ROS Simulations

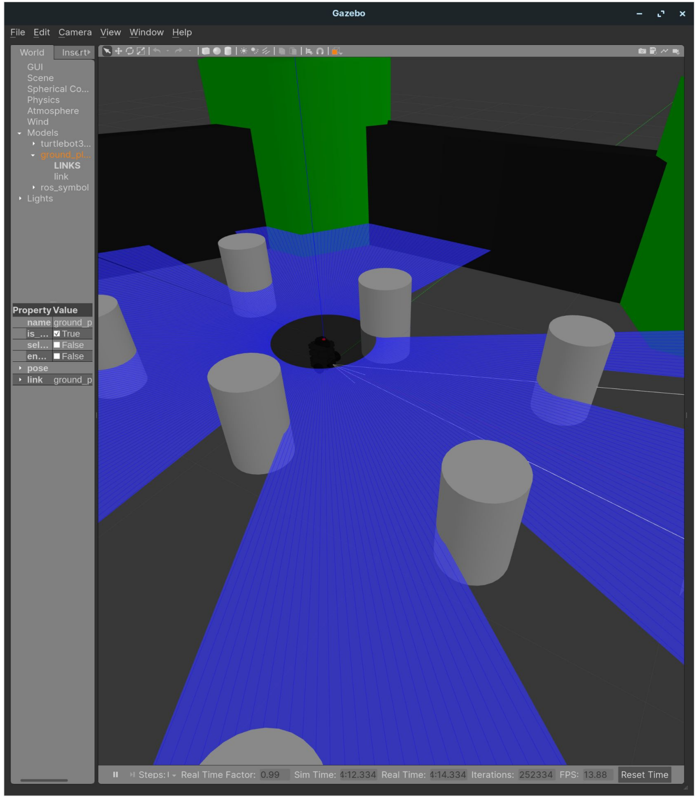


Fig: 2D Lidar Scan

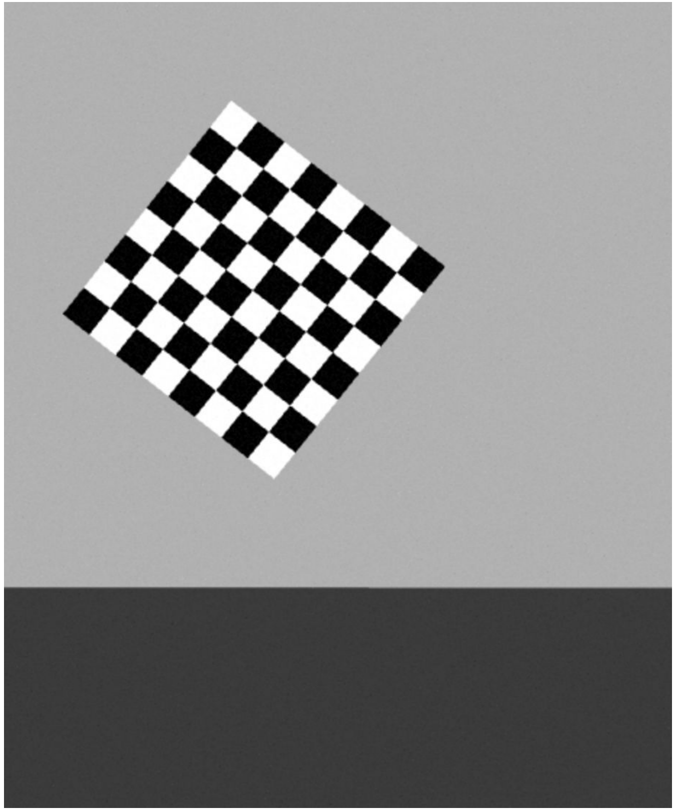


Fig: Image taken during 3D data collection

Data Collection

- Collected data through self-written subscribers
    - Camera Data (raw image)
    - 3D Lidar (PointCloud2)
    - 2D Lidar (LaserScan)
- Manually annotated data later on to extract point cloud edges
- Code is well documented through READMEs

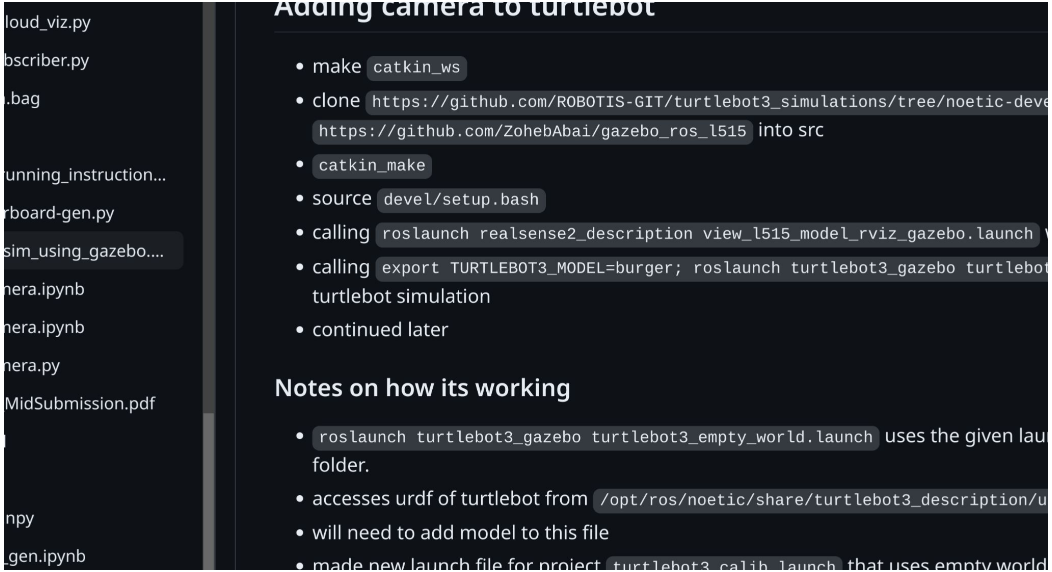


Fig: Screenshot of documentation

3D calibration

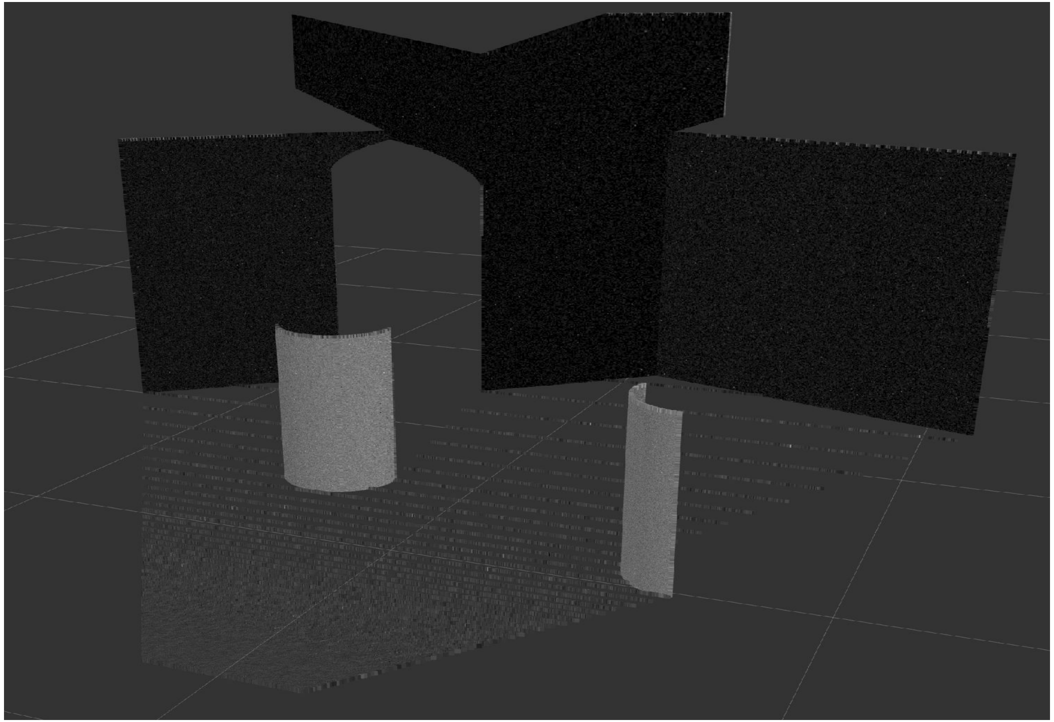
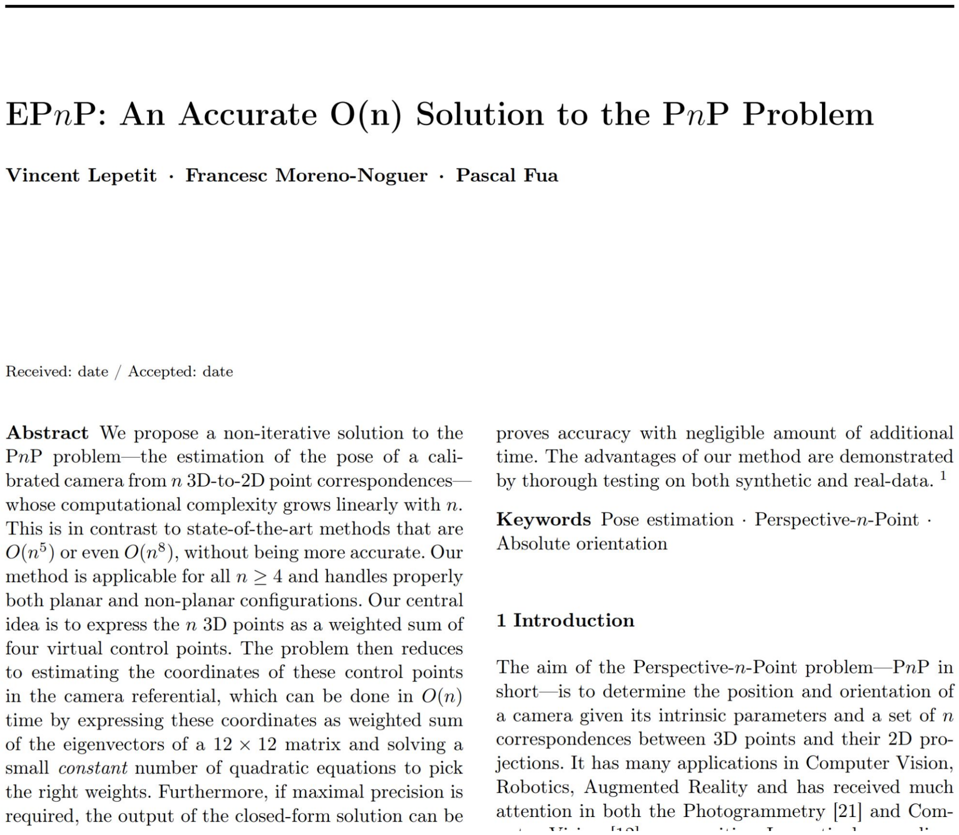


Fig: Some background data in point cloud

- Did extensive Literature Review - nothing was sweet and simple
- Finally, wrote from scratch code that
  - Takes in Image
  - Gets checkerboard endpoints
  - Takes in point cloud (.bag data)
  - Manually annotate the endpoints of checkerboard
  - Run PnP
- Why? Next slide

PnP

- Given 3D world coordinates and 2D image points, returns the extrinsic calibration matrix
- If we assume lidar to be the world frame, the 3D coordinates are in world frame
- So, running PNP gives the camera extrinsics which is the transformation between camera and lidar



EPnP: An Accurate O(n) Solution to the PnP Problem

Vincent Lepetit · Francesc Moreno-Noguer · Pascal Fua

Received: date / Accepted: date

Abstract We propose a non-iterative solution to the PnP problem—the estimation of the pose of a calibrated camera from n 3D-to-2D point correspondences—whose computational complexity grows linearly with n. This is in contrast to state-of-the-art methods that are $O(n^5)$ or even $O(n^8)$, without being more accurate. Our method is applicable for all $n \geq 4$ and handles properly both planar and non-planar configurations. Our central idea is to express the n 3D points as a weighted sum of four virtual control points. The problem then reduces to estimating the coordinates of these control points in the camera referential, which can be done in $O(n)$ time by expressing these coordinates as weighted sum of the eigenvectors of a 12×12 matrix and solving a small *constant* number of quadratic equations to pick the right weights. Furthermore, if maximal precision is required, the output of the closed-form solution can be proves accuracy with negligible amount of additional time. The advantages of our method are demonstrated by thorough testing on both synthetic and real-data. [1]

Keywords Pose estimation · Perspective-n-Point · Absolute orientation

1 Introduction

The aim of the Perspective-n-Point problem—PnP in short—is to determine the position and orientation of a camera given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projections. It has many applications in Computer Vision, Robotics, Augmented Reality and has received much attention in both the Photogrammetry [21] and Com-

Fig: The EPnP paper

2D Calibration

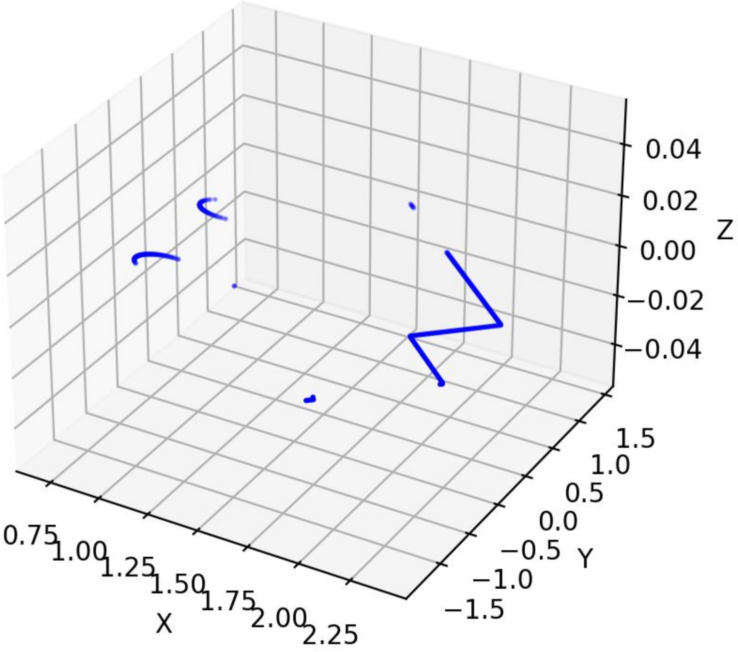


Fig: 2D Lidar Scan from ROS

- Made assumptions
  - Rotation between camera and lidar is only through one axis
    This means that we assume that we know that the lidar scan line is horizontal to image's horizontal axis
- If we don't make such assumptions then then calibration becomes very hard
- Did not find existing literature that tackled this in a simple way
- Here's what we did

2D Calibration

- Data collection
  - Take an image
  - Take the laser scan reading
  - Manually annotate correspondences
- Wrote math and tried figuring it out
- Made extra assumption that points won’t be collinear
- Got a few equations
- Realised that it's the same as Grunert’s P3P
- Used PnP to solve this as well

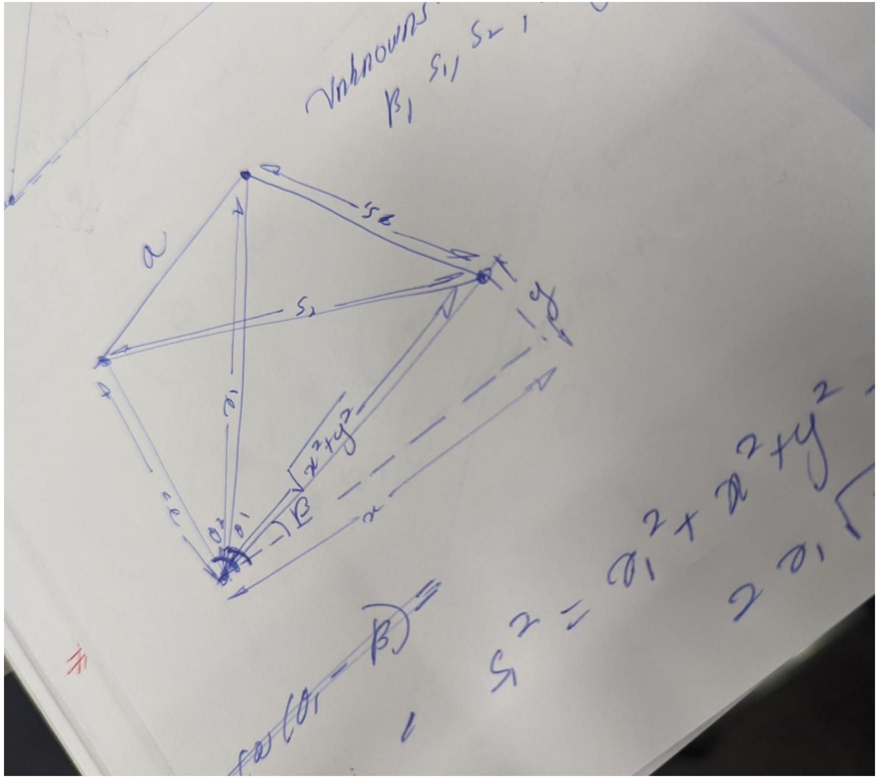


Fig: Some math we tried

Conclusions

- Our code works
- Real world demonstration of intrinsic calibration
- Simulated data for extrinsic calibration
- We spent close to 25-30 hours per person
    - ROS simulations alone took up about 15 hours to figure out and another 5 for data collection
    - Zhang implementation from scratch took up around 15 hours to code and test
    - Writing code for extrinsic calibration took up 20 hours to debug - first tried on external datasets and then made our own basic rudimentary data
    - Debugging took around 5 hours

Fig: Vineeth tired from doing math

Thank you