HW3

Theo McArn

November 14, 2025

Problem 1: Camera calibration

Python code written for camera calibration can be found in camera_calibration.py

Camera Intrinsics Matrix:

$$K = \begin{bmatrix} \alpha_u & s & p_u \\ & \alpha_v & p_v \\ & & 1 \end{bmatrix}$$

Calculated Intrinsics Matrix:

$$K = \begin{bmatrix} 1473.396714 & 0.0 & 546.319951 \\ & 1468.212739 & 943.040516 \\ & & 1 \end{bmatrix}$$

Problem 2: Perspective-n-Point AprilTag Pose Estimation

a) **Formulating the PnP problem:**

$$\hat{X} = \underset{X \in SE(3)}{\operatorname{argmin}} \sum_{i=1}^{N} \|\tilde{u}_i - \pi(K, X, P_i)\|_2^2 \tag{1}$$

where $\{\tilde{u}_i\}_{i=1}^N \subset \mathbb{R}^2$ are the 2D projections of the corresponding real world points $\{P_i\}_{i=1}^N \subset \mathbb{R}^3$, $\pi : \mathbb{R}^3 \to \mathbb{R}^3$ is our pinhole camera model, K is our intrinsics model, and X^* is the approximated camera extrinsics.

b) **Solving the PnP problem using GTSAM:**
Python code written for camera calibration can be found in pnp_estimation.py

Calculated pose of the camera in April Tag 0's coordinate frame:

$$\hat{X} = \begin{bmatrix} 0.999 & -0.035 & 0.019 & -0.016 \\ 0.036 & 0.999 & -0.023 & 0.029 \\ -0.018 & 0.024 & 1.000 & -0.081 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Plot can be seen in Figure 1

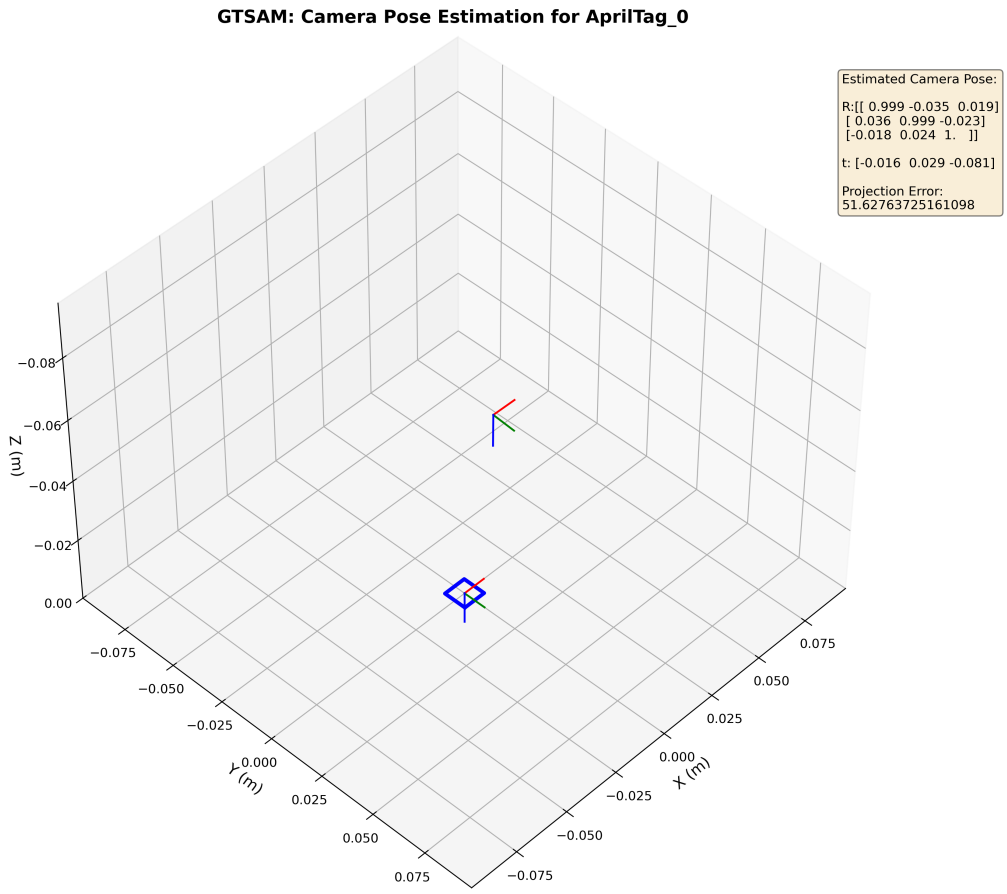

Figure 1: Pose of Camera in Frame one w.r.t. April Tag 0

Problem 3: Visual SLAM

a) Deriving an expression for the pose of the target in the camera frame: X_{CT} (known) in terms of the camera extrinsics: X_{WC} (unknown) and the targets world pose: X_{WT} (unknown):

$$X_{CT} = X_{CW} X_{WT}$$

Substuting in:

$$X_{WC}^{-1} = X_{CW}$$

We arrive at:

$$X_{CT} = X_{WC}^{-1} X_{WT}$$

b) **Negative Log Likelihood Expression:**

$$-\log p(\tilde{X}_{CT} | X_{CT}, X_{WT})$$

$$-\log p(\tilde{X}_{CT} | X_{CT})$$

$$-\log p(\tilde{X}_{CT} | X_{WC}^{-1} X_{WT})$$

Plugging in the probability density function for a multivariate gaussian distribution (ignoring the normalization constant):

$$-\log\left[\exp\left(-\frac{1}{2}(\tilde{X}_{CT}-X_{WC}^{-1}X_{WT})^{T}\Sigma^{-1}(\tilde{X}_{CT}-X_{WC}^{-1}X_{WT})\right)\right] \tag{2}$$

c) **Maximum Likelihood Estimation:**

Extending this negative log likelihood expression over multiple camera poses $\{X_i\}_{i=1}^{N}$, multiple tag poses $\{Y_j\}_{j=1}^{M}$, and multiple noisy camera measurements $\tilde{Z}_{ij}$ representing the relative pose of tag j in camera frame i, we get the following Maximum Likelihood Estimation:

$$\hat{X},\hat{Y}=\underset{X_i,i\in[N];Y_j,j\in[M]}{\operatorname{argmin}}\sum_{i=1}^{N}\sum_{j=1}^{M}-\log\left[\exp\left(-\frac{1}{2}(\tilde{Z}_{ij}-X_i^{-1}Y_j)^{T}\Sigma^{-1}(\tilde{Z}_{ij}-X_i^{-1}Y_j)\right)\right] \tag{3}$$

d) **Implementation:**

Python implementation can be found in the file vslam.py. Resulting plot of camera and tag positions can be seen in figure 2. A live animation of the pose estimation can be found here.

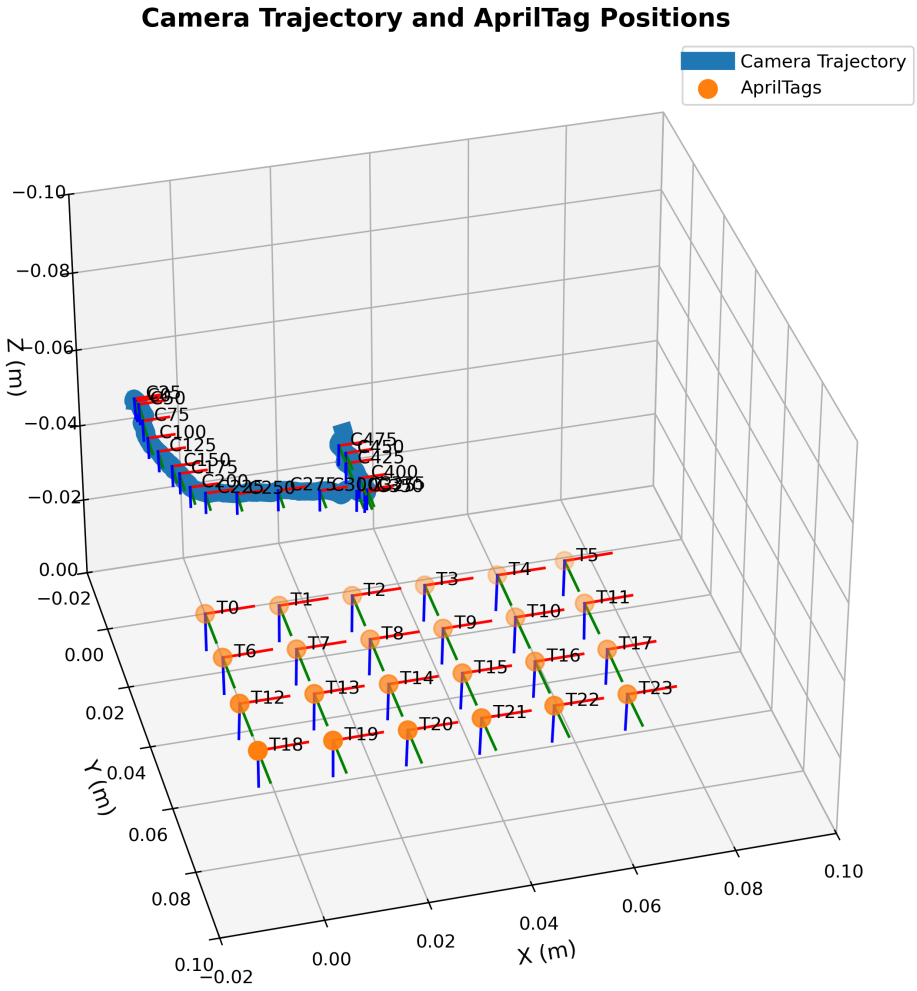

Figure 2: Estimated Camera Trajectory and Tag Positions