

```
In [201...
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import distance
```

The Camera Rig

Intrinsic Parameters

- I created a function to compute the image points using the intrinsic parameters of the camera

```
In [202...
def image_points(w):
    skew = 0
    delta = np.array([256, 256])
    phi = np.array([200, 200])
    x = (phi[0]*w[0] + skew*w[1])/w[2] + delta[0]
    y = (phi[1]*w[1])/w[2] + delta[1]
    return x, y

def rotation_translation(w, R, T):
    w_prime = R@w + T
    return w_prime
```

Extrinsic Parameters

- I created an Ω matrix and τ vector for the three cameras

```
In [203...
angle_1 = 0*np.pi/180
omega_1 = np.array([[np.cos(angle_1), 0, np.sin(angle_1)], [0, 1, 0], [np.sin(angle_1), 0, np.cos(angle_1)]])
T_1 = np.array([[0], [0], [0]])

angle_2 = 35*np.pi/180
omega_2 = np.array([[np.cos(angle_2), 0, -np.sin(angle_2)], [0, 1, 0], [np.sin(angle_2), 0, np.cos(angle_2)]])
T_2 = np.array([[200], [0], [0]])

angle_3 = (-35)*np.pi/180
omega_3 = np.array([[np.cos(angle_3), 0, -np.sin(angle_3)], [0, 1, 0], [np.sin(angle_3), 0, np.cos(angle_3)]])
T_3 = np.array([[-200], [0], [0]])
```

Measurements

Computing the synthetic image points

1. I transformed all the cube points using the extrinsic parameter for each cameras.
2. I computed the image points for the three cameras using the three new coordinate 3D points.

```
In [204...
CC = [[0, 0, 200], [25, 0, 200], [50, 0, 200], [50, 25, 200], [50, 50, 200], [25, 50, 200], [0, 50, 200], [0, 25, 200], [0, 0, 225], [50, 0, 225], [50, 25, 225], [50, 50, 225], [25, 50, 225], [0, 50, 225], [0, 25, 225], [0, 0, 250], [50, 0, 250], [50, 25, 250], [50, 50, 250], [25, 50, 250], [0, 50, 250], [0, 25, 250], [0, 0, 275], [50, 0, 275], [50, 25, 275], [50, 50, 275], [25, 50, 275], [0, 50, 275], [0, 25, 275], [0, 0, 300], [50, 0, 300], [50, 25, 300], [50, 50, 300], [25, 50, 300], [0, 50, 300], [0, 25, 300], [0, 0, 325], [50, 0, 325], [50, 25, 325], [50, 50, 325], [25, 50, 325], [0, 50, 325], [0, 25, 325], [0, 0, 350], [50, 0, 350], [50, 25, 350], [50, 50, 350], [25, 50, 350], [0, 50, 350], [0, 25, 350], [0, 0, 375], [50, 0, 375], [50, 25, 375], [50, 50, 375], [25, 50, 375], [0, 50, 375], [0, 25, 375], [0, 0, 400], [50, 0, 400], [50, 25, 400], [50, 50, 400], [25, 50, 400], [0, 50, 400], [0, 25, 400], [0, 0, 425], [50, 0, 425], [50, 25, 425], [50, 50, 425], [25, 50, 425], [0, 50, 425], [0, 25, 425], [0, 0, 450], [50, 0, 450], [50, 25, 450], [50, 50, 450], [25, 50, 450], [0, 50, 450], [0, 25, 450], [0, 0, 475], [50, 0, 475], [50, 25, 475], [50, 50, 475], [25, 50, 475], [0, 50, 475], [0, 25, 475], [0, 0, 500], [50, 0, 500], [50, 25, 500], [50, 50, 500], [25, 50, 500], [0, 50, 500], [0, 25, 500], [0, 0, 525], [50, 0, 525], [50, 25, 525], [50, 50, 525], [25, 50, 525], [0, 50, 525], [0, 25, 525], [0, 0, 550], [50, 0, 550], [50, 25, 550], [50, 50, 550], [25, 50, 550], [0, 50, 550], [0, 25, 550], [0, 0, 575], [50, 0, 575], [50, 25, 575], [50, 50, 575], [25, 50, 575], [0, 50, 575], [0, 25, 575], [0, 0, 600], [50, 0, 600], [50, 25, 600], [50, 50, 600], [25, 50, 600], [0, 50, 600], [0, 25, 600], [0, 0, 625], [50, 0, 625], [50, 25, 625], [50, 50, 625], [25, 50, 625], [0, 50, 625], [0, 25, 625], [0, 0, 650], [50, 0, 650], [50, 25, 650], [50, 50, 650], [25, 50, 650], [0, 50, 650], [0, 25, 650], [0, 0, 675], [50, 0, 675], [50, 25, 675], [50, 50, 675], [25, 50, 675], [0, 50, 675], [0, 25, 675], [0, 0, 700], [50, 0, 700], [50, 25, 700], [50, 50, 700], [25, 50, 700], [0, 50, 700], [0, 25, 700], [0, 0, 725], [50, 0, 725], [50, 25, 725], [50, 50, 725], [25, 50, 725], [0, 50, 725], [0, 25, 725], [0, 0, 750], [50, 0, 750], [50, 25, 750], [50, 50, 750], [25, 50, 750], [0, 50, 750], [0, 25, 750], [0, 0, 775], [50, 0, 775], [50, 25, 775], [50, 50, 775], [25, 50, 775], [0, 50, 775], [0, 25, 775], [0, 0, 800], [50, 0, 800], [50, 25, 800], [50, 50, 800], [25, 50, 800], [0, 50, 800], [0, 25, 800], [0, 0, 825], [50, 0, 825], [50, 25, 825], [50, 50, 825], [25, 50, 825], [0, 50, 825], [0, 25, 825], [0, 0, 850], [50, 0, 850], [50, 25, 850], [50, 50, 850], [25, 50, 850], [0, 50, 850], [0, 25, 850], [0, 0, 875], [50, 0, 875], [50, 25, 875], [50, 50, 875], [25, 50, 875], [0, 50, 875], [0, 25, 875], [0, 0, 900], [50, 0, 900], [50, 25, 900], [50, 50, 900], [25, 50, 900], [0, 50, 900], [0, 25, 900], [0, 0, 925], [50, 0, 925], [50, 25, 925], [50, 50, 925], [25, 50, 925], [0, 50, 925], [0, 25, 925], [0, 0, 950], [50, 0, 950], [50, 25, 950], [50, 50, 950], [25, 50, 950], [0, 50, 950], [0, 25, 950], [0, 0, 975], [50, 0, 975], [50, 25, 975], [50, 50, 975], [25, 50, 975], [0, 50, 975], [0, 25, 975], [0, 0, 1000], [50, 0, 1000], [50, 25, 1000], [50, 50, 1000], [25, 50, 1000], [0, 50, 1000], [0, 25, 1000], [0, 0, 1025], [50, 0, 1025], [50, 25, 1025], [50, 50, 1025], [25, 50, 1025], [0, 50, 1025], [0, 25, 1025], [0, 0, 1050], [50, 0, 1050], [50, 25, 1050], [50, 50, 1050], [25, 50, 1050], [0, 50, 1050], [0, 25, 1050], [0, 0, 1075], [50, 0, 1075], [50, 25, 1075], [50, 50, 1075], [25, 50, 1075], [0, 50, 1075], [0, 25, 1075], [0, 0, 1100], [50, 0, 1100], [50, 25, 1100], [50, 50, 1100], [25, 50, 1100], [0, 50, 1100], [0, 25, 1100], [0, 0, 1125], [50, 0, 1125], [50, 25, 1125], [50, 50, 1125], [25, 50, 1125], [0, 50, 1125], [0, 25, 1125], [0, 0, 1150], [50, 0, 1150], [50, 25, 1150], [50, 50, 1150], [25, 50, 1150], [0, 50, 1150], [0, 25, 1150], [0, 0, 1175], [50, 0, 1175], [50, 25, 1175], [50, 50, 1175], [25, 50, 1175], [0, 50, 1175], [0, 25, 1175], [0, 0, 1200], [50, 0, 1200], [50, 25, 1200], [50, 50, 1200], [25, 50, 1200], [0, 50, 1200], [0, 25, 1200], [0, 0, 1225], [50, 0, 1225], [50, 25, 1225], [50, 50, 1225], [25, 50, 1225], [0, 50, 1225], [0, 25, 1225], [0, 0, 1250], [50, 0, 1250], [50, 25, 1250], [50, 50, 1250], [25, 50, 1250], [0, 50, 1250], [0, 25, 1250], [0, 0, 1275], [50, 0, 1275], [50, 25, 1275], [50, 50, 1275], [25, 50, 1275], [0, 50, 1275], [0, 25, 1275], [0, 0, 1300], [50, 0, 1300], [50, 25, 1300], [50, 50, 1300], [25, 50, 1300], [0, 50, 1300], [0, 25, 1300], [0, 0, 1325], [50, 0, 1325], [50, 25, 1325], [50, 50, 1325], [25, 50, 1325], [0, 50, 1325], [0, 25, 1325], [0, 0, 1350], [50, 0, 1350], [50, 25, 1350], [50, 50, 1350], [25, 50, 1350], [0, 50, 1350], [0, 25, 1350], [0, 0, 1375], [50, 0, 1375], [50, 25, 1375], [50, 50, 1375], [25, 50, 1375], [0, 50, 1375], [0, 25, 1375], [0, 0, 1400], [50, 0, 1400], [50, 25, 1400], [50, 50, 1400], [25, 50, 1400], [0, 50, 1400], [0, 25, 1400], [0, 0, 1425], [50, 0, 1425], [50, 25, 1425], [50, 50, 1425], [25, 50, 1425], [0, 50, 1425], [0, 25, 1425], [0, 0, 1450], [50, 0, 1450], [50, 25, 1450], [50, 50, 1450], [25, 50, 1450], [0, 50, 1450], [0, 25, 1450], [0, 0, 1475], [50, 0, 1475], [50, 25, 1475], [50, 50, 1475], [25, 50, 1475], [0, 50, 1475], [0, 25, 1475], [0, 0, 1500], [50, 0, 1500], [50, 25, 1500], [50, 50, 1500], [25, 50, 1500], [0, 50, 1500], [0, 25, 1500], [0, 0, 1525], [50, 0, 1525], [50, 25, 1525], [50, 50, 1525], [25, 50, 1525], [0, 50, 1525], [0, 25, 1525], [0, 0, 1550], [50, 0, 1550], [50, 25, 1550], [50, 50, 1550], [25, 50, 1550], [0, 50, 1550], [0, 25, 1550], [0, 0, 1575], [50, 0, 1575], [50, 25, 1575], [50, 50, 1575], [25, 50, 1575], [0, 50, 1575], [0, 25, 1575], [0, 0, 1600], [50, 0, 1600], [50, 25, 1600], [50, 50, 1600], [25, 50, 1600], [0, 50, 1600], [0, 25, 1600], [0, 0, 1625], [50, 0, 1625], [50, 25, 1625], [50, 50, 1625], [25, 50, 1625], [0, 50, 1625], [0, 25, 1625], [0, 0, 1650], [50, 0, 1650], [50, 25, 1650], [50, 50, 1650], [25, 50, 1650], [0, 50, 1650], [0, 25, 1650], [0, 0, 1675], [50, 0, 1675], [50, 25, 1675], [50, 50, 1675], [25, 50, 1675], [0, 50, 1675], [0, 25, 1675], [0, 0, 1700], [50, 0, 1700], [50, 25, 1700], [50, 50, 1700], [25, 50, 1700], [0, 50, 1700], [0, 25, 1700], [0, 0, 1725], [50, 0, 1725], [50, 25, 1725], [50, 50, 1725], [25, 50, 1725], [0, 50, 1725], [0, 25, 1725], [0, 0, 1750], [50, 0, 1750], [50, 25, 1750], [50, 50, 1750], [25, 50, 1750], [0, 50, 1750], [0, 25, 1750], [0, 0, 1775], [50, 0, 1775], [50, 25, 1775], [50, 50, 1775], [25, 50, 1775], [0, 50, 1775], [0, 25, 1775], [0, 0, 1800], [50, 0, 1800], [50, 25, 1800], [50, 50, 1800], [25, 50, 1800], [0, 50, 1800], [0, 25, 1800], [0, 0, 1825], [50, 0, 1825], [50, 25, 1825], [50, 50, 1825], [25, 50, 1825], [0, 50, 1825], [0, 25, 1825], [0, 0, 1850], [50, 0, 1850], [50, 25, 1850], [50, 50, 1850], [25, 50, 1850], [0, 50, 1850], [0, 25, 1850], [0, 0, 1875], [50, 0, 1875], [50, 25, 1875], [50, 50, 1875], [25, 50, 1875], [0, 50, 1875], [0, 25, 1875], [0, 0, 1900], [50, 0, 1900], [50, 25, 1900], [50, 50, 1900], [25, 50, 1900], [0, 50, 1900], [0, 25, 1900], [0, 0, 1925], [50, 0, 1925], [50, 25, 1925], [50, 50, 1925], [25, 50, 1925], [0, 50, 1925], [0, 25, 1925], [0, 0, 1950], [50, 0, 1950], [50, 25, 1950], [50, 50, 1950], [25, 50, 1950], [0, 50, 1950], [0, 25, 1950], [0, 0, 1975], [50, 0, 1975], [50, 25, 1975], [50, 50, 1975], [25, 50, 1975], [0, 50, 1975], [0, 25, 1975], [0, 0, 2000], [50, 0, 2000], [50, 25, 2000], [50, 50, 2000], [25, 50, 2000], [0, 50, 2000], [0, 25, 2000], [0, 0, 2025], [50, 0, 2025], [50, 25, 2025], [50, 50, 2025], [25, 50, 2025], [0, 50, 2025], [0, 25, 2025], [0, 0, 2050], [50, 0, 2050], [50, 25, 2050], [50, 50, 2050], [25, 50, 2050], [0, 50, 2050], [0, 25, 2050], [0, 0, 2075], [50, 0, 2075], [50, 25, 2075], [50, 50, 2075], [25, 50, 2075], [0, 50, 2075], [0, 25, 2075], [0, 0, 2100], [50, 0, 2100], [50, 25, 2100], [50, 50, 2100], [25, 50, 2100], [0, 50, 2100], [0, 25, 2100], [0, 0, 2125], [50, 0, 2125], [50, 25, 2125], [50, 50, 2125], [25, 50, 2125], [0, 50, 2125], [0, 25, 2125], [0, 0, 2150], [50, 0, 2150], [50, 25, 2150], [50, 50, 2150], [25, 50, 2150], [0, 50, 2150], [0, 25, 2150], [0, 0, 2175], [50, 0, 2175], [50, 25, 2175], [50, 50, 2175], [25, 50, 2175], [0, 50, 2175], [0, 25, 2175], [0, 0, 2200], [50, 0, 2200], [50, 25, 2200], [50, 50, 2200], [25, 50, 2200], [0, 50, 2200], [0, 25, 2200], [0, 0, 2225], [50, 0, 2225], [50, 25, 2225], [50, 50, 2225], [25, 50, 2225], [0, 50, 2225], [0, 25, 2225], [0, 0, 2250], [50, 0, 2250], [50, 25, 2250], [50, 50, 2250], [25, 50, 2250], [0, 50, 2250], [0, 25, 2250], [0, 0, 2275], [50, 0, 2275], [50, 25, 2275], [50, 50, 2275], [25, 50, 2275], [0, 50, 2275], [0, 25, 2275], [0, 0, 2300], [50, 0, 2300], [50, 25, 2300], [50, 50, 2300], [25, 50, 2300], [0, 50, 2300], [0, 25, 2300], [0, 0, 2325], [50, 0, 2325], [50, 25, 2325], [50, 50, 2325], [25, 50, 2325], [0, 50, 2325], [0, 25, 2325], [0, 0, 2350], [50, 0, 2350], [50, 25, 2350], [50, 50, 2350], [25, 50, 2350], [0, 50, 2350], [0, 25, 2350], [0, 0, 2375], [50, 0, 2375], [50, 25, 2375], [50, 50, 2375], [25, 50, 2375], [0, 50, 2375], [0, 25, 2375], [0, 0, 2400], [50, 0, 2400], [50, 25, 2400], [50, 50, 2400], [25, 50, 2400], [0, 50, 2400], [0, 25, 2400], [0, 0, 2425], [50, 0, 2425], [50, 25, 2425], [50, 50, 2425], [25, 50, 2425], [0, 50, 2425], [0, 25, 2425], [0, 0, 2450], [50, 0, 2450], [50, 25, 2450], [50, 50, 2450], [25, 50, 2450], [0, 50, 2450], [0, 25, 2450], [0, 0, 2475], [50, 0, 2475], [50, 25, 2475], [50, 50, 2475], [25, 50, 2475], [0, 50, 2475], [0, 25, 2475], [0, 0, 2500], [50, 0, 2500], [50, 25, 2500], [50, 50, 2500], [25, 50, 2500], [0, 50, 2500], [0, 25, 2500], [0, 0, 2525], [50, 0, 2525], [50, 25, 2525], [50, 50, 2525], [25, 50, 2525], [0, 50, 2525], [0, 25, 2525], [0, 0, 2550], [50, 0, 2550], [50, 25, 2550], [50, 50, 2550], [25, 50, 2550], [0, 50, 2550], [0, 25, 2550], [0, 0, 2575], [50, 0, 2575], [50, 25, 2575], [50, 50, 2575], [25, 50, 2575], [0, 50, 2575], [0, 25, 2575], [0, 0, 2600], [50, 0, 2600], [50, 25, 2600], [50, 50, 2600], [25, 50, 2600], [0, 50, 2600], [0, 25, 2600], [0, 0, 2625], [50, 0, 2625], [50, 25, 2625], [50, 50, 2625], [25, 50, 2625], [0, 50, 2625], [0, 25, 2625], [0, 0, 2650], [50, 0, 2650], [50, 25, 2650], [50, 50, 2650], [25, 50, 2650], [0, 50, 2650], [0, 25, 2650], [0, 0, 2675], [50, 0, 2675], [50, 25, 2675], [50, 50, 2675], [25, 50, 2675], [0, 50, 2675], [0, 25, 2675], [0, 0, 2700], [50, 0, 2700], [50, 25, 2700], [50, 50, 2700], [25, 50, 2700], [0, 50, 2700], [0, 25, 2700], [0, 0, 2725], [50, 0, 2725], [50, 25, 2725], [50, 50, 2725], [25, 50, 2725], [0, 50, 2725], [0, 25, 2725], [0, 0, 2750], [50, 0, 2750], [50, 25, 2750], [50, 50, 2750], [25, 50, 2750], [0, 50, 2750], [0, 25, 2750], [0, 0, 2775], [50, 0, 2775], [50, 25, 2775], [50, 50, 2775], [25, 50, 2775], [0, 50, 2775], [0, 25, 2775], [0, 0, 2800], [50, 0, 2800], [50, 25, 2800], [50, 50, 2800], [25, 50, 2800], [0, 50, 2800], [0, 25, 2800], [0, 0, 2825], [50, 0, 2825], [50, 25, 2825], [50, 50, 2825], [25, 50, 2825], [0, 50, 2825], [0, 25, 2825], [0, 0, 2850], [50, 0, 2850], [50, 25, 2850], [50, 50, 2850], [25, 50, 2850], [0, 50, 2850], [0, 25, 2850], [0, 0, 2875], [50, 0, 2875], [50, 25, 2875], [50, 50, 2875], [25, 50, 2875], [0, 50, 2875], [0, 25, 2875], [0, 0, 2900], [50, 0, 2900], [50, 25, 2900], [50, 50, 2900], [25, 50, 2900], [0, 50, 2900], [0, 25, 2900], [0, 0, 2925], [50, 0, 2925], [50, 25, 2925], [50, 50, 2925], [25, 50, 2925], [0, 50, 2925], [0, 25, 2925], [0, 0, 2950], [50, 0, 2950], [50, 25, 2950], [50, 50, 2950], [25, 50, 2950], [0, 50, 2950], [0, 25, 2950], [0, 0, 2975], [50, 0, 2975], [50, 25, 2975], [50, 50, 2975], [25, 50, 2975], [0, 50, 2975], [0, 25, 2975], [0, 0, 3000], [50, 0, 3000], [50, 25, 3000], [50, 50, 3000], [25, 50, 3000], [0, 50, 3000], [0, 25, 3000], [0, 0, 3025], [50, 0, 3025], [50, 25, 3025], [50, 50, 3025], [25, 50, 3025], [0, 50, 3025], [0, 25, 3025], [0, 0, 3050], [50, 0, 3050], [50, 25, 3050], [50, 50, 3050], [25, 50, 3050], [0, 50, 3050], [0, 25, 3050], [0, 0, 3075], [50, 0, 3075], [50, 25, 3075], [50, 50, 3075], [25, 50, 3075], [0, 50, 3075], [0, 25, 3075], [0, 0, 3100], [50, 0, 3100], [50, 25, 3100], [50, 50, 3100], [25, 50, 3100], [0, 50, 3100], [0, 25, 3100], [0, 0, 3125], [50, 0, 3125], [50, 25, 3125], [50, 50, 3125], [25, 50, 3125], [0, 50, 3125], [0, 25, 3125], [0, 0, 3150], [50, 0, 3150], [50, 25, 3150], [50, 50, 3150], [25, 50, 3150], [0, 50, 3150], [0, 25, 3150], [0, 0, 3175], [50, 0, 3175], [50, 25, 3175], [50, 50, 3175], [25, 50, 3175], [0, 50, 3175], [0, 25, 3175], [0, 0, 3200], [50, 0, 3200], [50, 25, 3200], [50, 50, 3200], [25, 50, 3200], [0, 50, 3200], [0, 25, 3200], [0, 0, 3225], [50, 0, 3225], [50, 25, 3225], [50, 50, 3225], [25, 50, 3225], [0, 50, 3225], [0, 25, 3225], [0, 0, 3250], [50, 0, 3250], [50, 25, 3250], [50, 50, 3250], [25, 50, 3250], [0, 50, 3250], [0, 25, 3250], [0, 0, 3275], [50, 0, 3275], [50, 25, 3275], [50, 50, 3275], [25, 50, 3275], [0, 50, 3275], [0, 25, 3275], [0, 0, 3300], [50, 0, 3300], [50, 25, 3300], [50, 50, 3300], [25, 50, 3300], [0, 50, 3300
```

```
Y[1,i] = y_2
X[2,i] = x_3
Y[2,i] = y_3
```

Noise Addition

1. Created a vector of a random normal signal with parameter mu and sigma.
2. Added the random noise to the image points

In [296]

```
mu, sigma = 0, 2
noise = np.random.normal(mu, sigma, [2,20])

for i in range(N):
    X[i] = X[i] + noise[0]
    Y[i] = Y[i] + noise[1]
```

Estimation Method

Coordinate Normalization

- Multiplied the augmented x and y vectors by the inverse of the Λ matrix to get the normalized image coordinates

$$\Lambda = \begin{bmatrix} 200 & 0 & 256 \\ 0 & 200 & 256 \\ 0 & 0 & 1 \end{bmatrix}$$

In [297]

```
lambda_ = np.array([[200, 0, 256],[0, 200, 256], [0, 0, 1]])
lambda_inverse = np.linalg.inv(lambda_)

X_norm = np.zeros((N,len(C)))
Y_norm = np.zeros((N,len(C)))
Z_norm = np.zeros((N,len(C)))

for i in range(N):
    for j in range(len(C)):
        x_aug = np.vstack([X[i,j], Y[i,j], 1])
        x_norm = lambda_inverse@x_aug
        X_norm[i,j] = x_norm[0]
        Y_norm[i,j] = x_norm[1]
        Z_norm[i,j] = x_norm[2]
```

Solving Linear system of Equations

$$\begin{bmatrix} \omega_{31j}x'_j - \omega_{11j} & \omega_{32j}x'_j - \omega_{12j} & \omega_{33j}x'_j - \omega_{13j} \\ \omega_{31j}y'_j - \omega_{21j} & \omega_{32j}y'_j - \omega_{22j} & \omega_{33j}y'_j - \omega_{23j} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \tau_{xj} - \tau_{zj}x'_j \\ \tau_{yj} - \tau_{zj}y'_j \end{bmatrix}$$

1. I solved the equation:

solution for linear system of equations of the form: $Ax = b$

Using the

$$\begin{bmatrix} \omega_{31j}x'_j - \omega_{11j} & \omega_{32j}x'_j - \omega_{12j} & \omega_{33j}x'_j - \omega_{13j} \\ \omega_{31j}y'_j - \omega_{21j} & \omega_{32j}y'_j - \omega_{22j} & \omega_{33j}y'_j - \omega_{23j} \end{bmatrix}$$

2. I created an A matrix for each camera in the form:

$$\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

3. Found the \mathbf{w} points using the following equation for each camera \mathbf{A} matrix:

4. Using the reconstructed \mathbf{w} , I computed the image points \mathbf{x} using the intrinsic and extrinsic parameters for each camera

In [298]

```
A = []
b = []

for i in range(len(C)):
    A_1 = np.array([(omega_1[2,0]*X_norm[0,i]-omega_1[0,0]), (omega_1[2,1]*X_norm[0,i]-omega_1[0,1]), (omega_1[2,2]*X_norm[0,i]-omega_1[0,2]),
                    [(omega_1[2,0]*Y_norm[0,i]-omega_1[1,0]), (omega_1[2,1]*Y_norm[0,i]-omega_1[1,1]), (omega_1[2,2]*Y_norm[0,i]-omega_1[1,2])])

    A_2 = np.array([(omega_2[2,0]*X_norm[1,i]-omega_2[0,0]), (omega_2[2,1]*X_norm[1,i]-omega_2[0,1]), (omega_2[2,2]*X_norm[1,i]-omega_2[0,2]),
                    [(omega_2[2,0]*Y_norm[1,i]-omega_2[1,0]), (omega_2[2,1]*Y_norm[1,i]-omega_2[1,1]), (omega_2[2,2]*Y_norm[1,i]-omega_2[1,2])])

    A_3 = np.array([(omega_3[2,0]*X_norm[2,i]-omega_3[0,0]), (omega_3[2,1]*X_norm[2,i]-omega_3[0,1]), (omega_3[2,2]*X_norm[2,i]-omega_3[0,2]),
                    [(omega_3[2,0]*Y_norm[2,i]-omega_3[1,0]), (omega_3[2,1]*Y_norm[2,i]-omega_3[1,1]), (omega_3[2,2]*Y_norm[2,i]-omega_3[1,2])])

    b_1 = np.array((T_1[0]-T_1[2]*X_norm[0,i],T_1[1]-T_1[2]*Y_norm[0,i]))
    b_2 = np.array((T_2[0]-T_2[2]*X_norm[1,i],T_2[1]-T_2[2]*Y_norm[1,i]))
    b_3 = np.array((T_3[0]-T_3[2]*X_norm[2,i],T_3[1]-T_3[2]*Y_norm[2,i]))
```

```
A.append(np.vstack((A_1,A_2,A_3)))
b.append(np.vstack((b_1,b_2,b_3)))
```

```
In [299... W = np.zeros((len(C),3))
for i in range(len(A)):
    A_ = A[i]
    b_ = b[i]
    w = (np.linalg.inv(A_ .T@A_ ))@A_ .T@b_
    W[i,:] = w.T
```

```
In [300... X_hat = np.zeros((N,len(C)))
Y_hat = np.zeros((N,len(C)))

for i in range(len(C)):
    w = np.vstack(W[i])
    Wp_1 = w
    Wp_2 = rotation_translation(w, omega_2, T_2)
    Wp_3 = rotation_translation(w, omega_3, T_3)

    x_1,y_1 = image_points(Wp_1)
    x_2,y_2 = image_points(Wp_2)
    x_3,y_3 = image_points(Wp_3)

    X_hat[0,i] = x_1
    Y_hat[0,i] = y_1
    X_hat[1,i] = x_2
    Y_hat[1,i] = y_2
    X_hat[2,i] = x_3
    Y_hat[2,i] = y_3
```

Sensitivity Analysis

Re-projection error

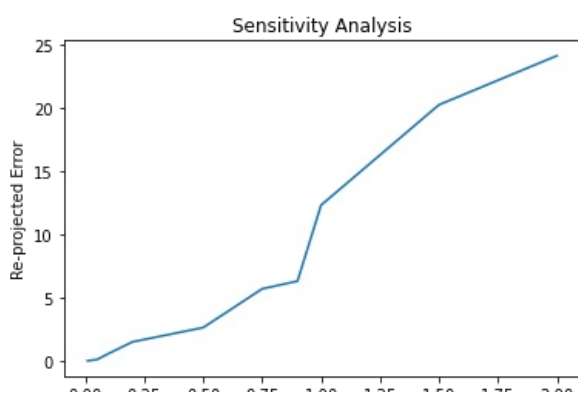
- I found the re-projection error between the original **X** and **Y** computed using the intrinsic and extrinsic parameters and the original 3D points, and the re-projected **X** and **Y** using the estimation method.
- I repeated the whole estimation method after adding different levels of noise and computed the re-projection error.

```
In [210... E_ = []
```

```
In [301... E = 0
for j in range(N):
    for i in range(len(C)):
        x1 = np.vstack((X_hat[j,i],Y_hat[j,i]))
        x2 = np.vstack((X[j,i],Y[j,i]))
        e = distance.euclidean(x1, x2)
        E = E + e
E_.append(E)
```

```
In [312... s = [0.01, 0.05, 0.1, 0.2, 0.5, 0.75, 0.9, 1, 1.5, 2]
plt.plot(s, E_)
plt.title('Sensitivity Analysis')
plt.xlabel('Sigma')
plt.ylabel('Re-projected Error')
```

```
Out[312... Text(0, 0.5, 'Re-projected Error')
```



0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
Sigma

Sigma	Error
0	1.02e-12
0.01	0.100321
0.05	0.579971
0.1	1.492708
0.2	2.620154
0.5	5.678475
0.75	6.288986
0.9	6.288986
1	12.28457
1.5	20.20812
2	24.08086

Processing math: 100%