Aaron Lo
atl2374
31 October 2023

CS376 Assignment 3

I. Programming: Camera Calibration
   A. Picking 2D Keypoints
      1. I use ginput to collect user data. Then, I use Matlabs SURF feature detection to find the strongest corner features in the image. For each user collected point, it snaps to the closest selected SURF feature using a nearest neighbor search. The points I chose were the points in the corners where the boxes intersected.
   B. Setting 3D Keypoints
      1. I set the bottom corner of the box closest to the camera to be the origin, the y-axis to pointing up, the x-axis to be going rightwards (horizotnal to the camera) and the z-axis to be going leftwards (away from the camera). Each "unit" of the coordinate plane was the square in the box. Each point plotted by the 2D keypoint has a corresponding one on the 3D axis, and I set them as such.
   C. KRT
      1. det(R) = 1

```
K =

   1.0e+03 *

    2.0995   -0.0006    1.9987
         0    1.9873    1.2998
         0         0    0.0010

R =

   -0.6826    0.0035    0.7308
    0.0606    0.9968    0.0518
   -0.7283    0.0797   -0.6806

t =

   -0.0001
   -5.8534
   -6.9719
```

   D. Optimal Keypoints
      1. The best points to choose are a set of points that selects a wide range of areas. It is good to select points from both close to the center point of the box as well as more towards its outside.
   E. 20 Marked Keypoints to 10 Marked Keypoints

1. First, we just declare the error as a Square Means Error of distance away from converting our 3D points into 2D with the calibrated matrix compared with the actual 2D points. Using these 20 points, we run the idea of RANSAC to find the the 10 points with a good K that minimizes the total error.

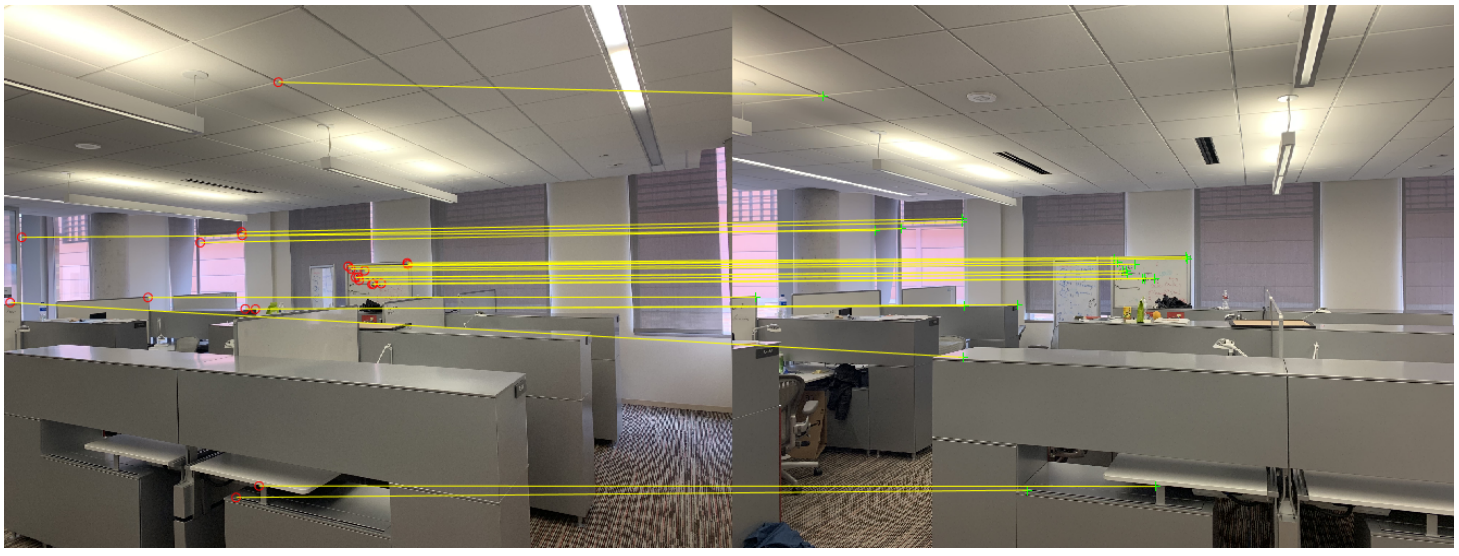
F. Lines for Calibration

1. We define usage of 2D Lines with their slope and intercept (M and B). The 3D Lines will be defined by their slope as viewed in parallel to the Camera (M_x), slope as view in perpendicular to the Camera (M_z) and the intercept to the y-p (B_y). Using this translation, we can keep the same formulas for calculating the projection matrix. A slight drawback is that we can not use vertical lines or else the slope will be too big.

II. Programming: Structure From Motion

A. Feature Correspondence

1. I used MATLABs feature correspondence methods to find my matching points. I use detectSURFFeatures to find good points in both images, extractFeatures for both, and then matchFeatures with a MaxRatio of 0.25 to find matching points between the two images. The points for both the SourceImage and TargetImage are stored in a file. SourceImage is stored in motions.mat in a Matrix called sCoord2D while the TargetImage points are stored in motiont.mat in a Matrix called tCoord2D. The parameters for detectSURFFeatures and matchFeatures need to be chosen well so that it selects good points as well as removing mos of the noise.



B. Extract RT

1. det(R) = 1

```
R =                                        T =
    0.6037   -0.0298   -0.7967
    0.0119   -0.9989    0.0464        -0.9562
   -0.7971   -0.0375   -0.6026        -0.0376
                                       0.2902
```

C. 5 Sets of Correspondence

1. I found that a set of points very close together or all very far apart gave poor results. The best so of points splayed the matching points across the entire scene. In addition, when faced with the choice of a lot of points (but potentially not the best) compared to fewer points but more accurate, the more accurate one held up better against noise.