EE 5450 Project 01: Simultaneous Monocular Calibration and Pose Estimation

David R. Mohler

February 21, 2018

1 Introduction

Determination of the pose of rigid objects within images is a common problem within many practical application of computer vision, robotics, computer graphics, etc. The following set of experiments are focused on the application of the simultaneous monocular calibration and pose estimation algorithm to a series of images of rigid objects with known dimensions and correspondence points. From this known data, the algorithm is capable of calibrating a single monocular camera with a fixed focal length. The calibration enables simultaneous discovery of the pose of the viewed object, and from this generate a three dimensional representation of that same object.

2 Methods and Results

The foundation of this project is based on the known dimensions of a given rigid object, a box for example, which is manually assigned a number of points on the object corresponding to its corners (i.e. the coordinates in the object frame represent the dimensions of the object). A model of the initial box and its correspondence points can be seen in Figure 1, additionally, the measured dimensions of the object are shown in Table 1. Given this data we are able to proceed with the implementation of the monocular pose algorithm.

Using the object coordinates in their homogeneous form (i.e. $X_o^i = [x_o^i \quad y_o^i \quad z_o^i \quad 1]^T$), we begin with calculating an approximation of the projection matrix, Π_{est}. The projection matrix is such that $\Pi = [KR \quad KT] \in \Re^{3\times4}$, where K is the camera calibration matrix, R is a rotation matrix, and T is the translation vector. Given that we know the location of the correspondence points, χ^{pj}, and their matching locations in the object frame, X_0^j, assuming that a sufficient number of points are provided, we are then able to find an approximation to Π.This approximation can be expressed as a least squares minimization of Equation 1, where e_3 is the standard basis vector $[001]^T$, Π^S is the vectorized or "stacked" version of Π,and $\otimes$ represents the Kronecker product. The least squares estimate of the vectorized projection matrix can be found as the minimum input direction of N, which is the final right singular vector yielded from the singular value decomposition (SVD) of N.

$$N^j\Pi^{pj} = [(X_o^{jT} \otimes I_3) - (X_o^{jT} \otimes \chi^{pj} e_3^T)]\Pi^S = 0 \tag{1}$$

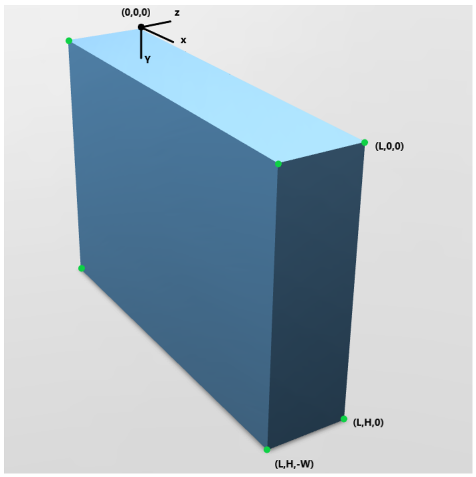

Figure 1: Rigid object model.

Parameter	Value (cm)
$Length$	45.6
$Height$	32.5
$Width$	10.1

Table 1: Object Dimensions

From this we de-vectorize the result of the SVD and obtain Π_{est}. Using this, all necessary components to describe the calibration and pose of the camera are extracted. From order reversing the standard QR decomposition we are able to obtain the scaling factor α, the calibration matrix, and the rotation matrices corresponding to the given set of pixel coordinates. Using this information we can lastly find the translation vector associated with the system from the following equation,where $T^{'} = KT$, and is the rightmost column of Π_{est}:

$$T = \frac{K^{-1}T^{'}}{\alpha} \qquad (2)$$

$$\chi^{pj} = \frac{\Pi_{est}X_0^j}{\lambda^j} \qquad (3)$$

$$X_o^{pj} = R^T K^{-1}(\lambda\chi^p - KT) \qquad (4)$$

Using the information obtained from the algorithm, we next show the ability to project estimated pixel coordinates in to the image plane. These estimated pixel coordinates are described by Equation 3. From this we are able to qualitatively visualize the success of the algorithm in matching provided correspondence points (Figure 2). From the estimated pixel coordinates in noiseless reconstructions of the object, using Equation 4, we calculate the average root mean square error (RMSE) of the position between the true coordinates and their respective estimates in the object frame in order to establish the overall fidelity of the algorithm across multiple images taken with varying rotations, translations, and calibrations. In this case PRMSE is expressed by Equations 5 and 6, where n_{im} is the number of images, and n_f is the number of features used:

$$d_i = \sqrt{(x_o^{ij} - \hat{x}_o^{ij}) + (y_o^{ij} - \hat{y}_o^{ij}) + (z_o^{ij} - \hat{z}_o^{ij})} \quad \text{for} \quad j = 1, 2, ..., n_{im} \qquad (5)$$

$$PRMSE = \frac{\sum_{j=1}^{n_{im}} \sqrt{\sum_{i=1}^{n_f} \sqrt{di/n_f}}}{n_{im}} \qquad (6)$$

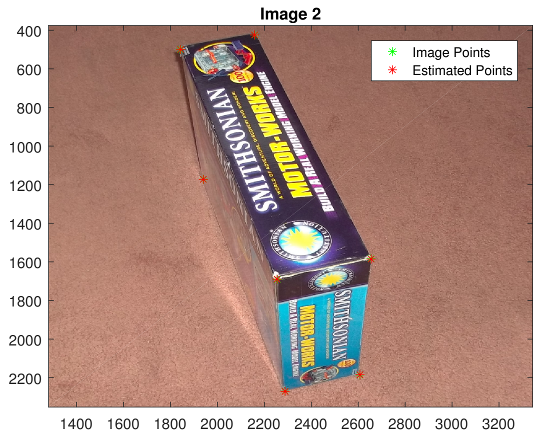

Figure 2: Estimated pixel coordinates

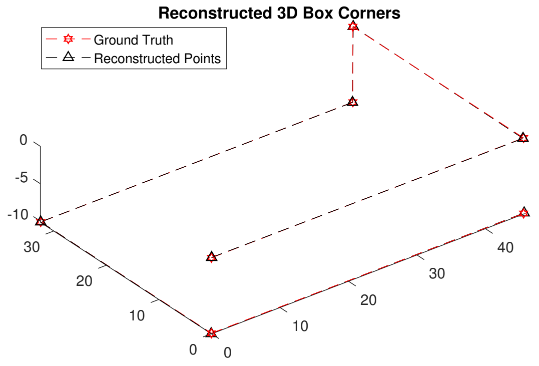

Figure 3: Reconstructed box relative to ground truth

Across the four tested images (with no noise corruption) through 100 iterations of the algorithm, the average distance error was relatively small at $PRMSE = 0.2763 \quad \text{cm}$.

2.1 Corrupted Correspondence Points

Once the results of the algorithm are proven to work on its own data sets we observe the effect of the calibration and pose established by the algorithm on the pure data set (as opposed to the estimated) data to observe the distortions due to noise or error in correspondence. When applying normally distributed to each component of the correspondence points individually with a standard deviation of $\sigma = 100$ pixels we receive reconstruction results similar to those seen in 4. From the calibration received, we test the ability to reconstruct the true (uncorrupted) data. This yields considerable disfigurement of the object. The reconstruction of the original box when corrupted by noise and the ground truth can seen in Figure 5. To quantitatively capture the results of the noise reconstruction we compare the angles created between the correspondence points. Since the object is a box and the correspondence points lie along the edges, it is expected that all adjacent vectors are orthogonal to each other. However, in the presence of noise a representative data set of angles associated with the three visible faces of the box, and known correspondences appears, in Table 2.

Point	Angle(s) (degrees)
1	103.6447
2	82.6421, 90.4116
3	88.9373
4	91.496, 92.3435
5	93.2852
6	85.5876, 76.0264
7	97.667, 88.7342, 89.1479

Table 2: Object Dimensions

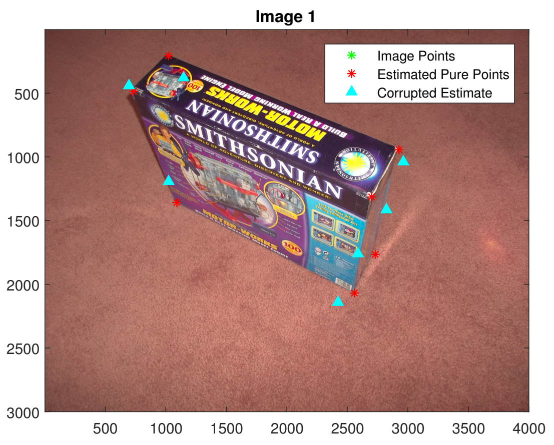

Figure 4: Estimated image coordinates (unique coordinate component noise)

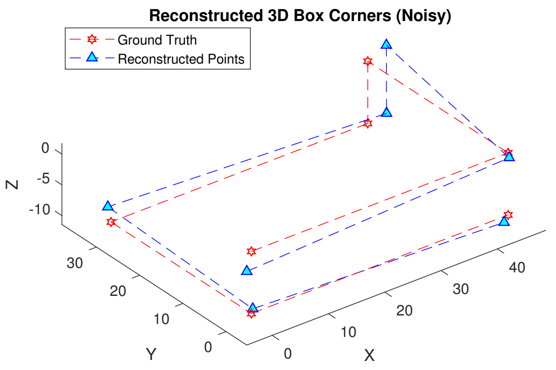

Figure 5: Reconstructed box relative to ground truth

As would be expected, the angles are centered about the appropriate mean value of 90 degrees, however the deviations from precisely right angles by as much as 15.53 degrees shows the obvious loss of the shape of the true object upon reconstruction. Similarly, the reliability of the reconstruction is tied to the amount of noise introduced in to the captured pixel coordinates. Under the condition of $\sigma = 100$ pixels, using 100 successive trials of the algorithm, there is an average PRMSE introduced of 1.65 cm, or an increase of error by approximately 595.4%. Through observation of various noise levels we observe that the RMS position error is (nearly) linear with the standard deviation of the noise (Figure 6). It is essential to note that beyond a standard deviation of approximately 200 pixels it becomes common for pixels to be projected in to non-existent pixel mappings, and as such are obviously invalid results.

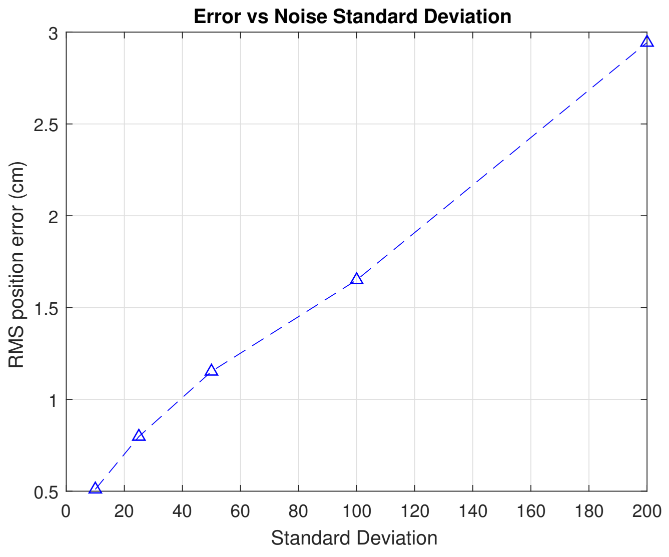

Figure 6: RMS Error as it varies with standard deviation of applied noise

2.2 Improper Dimensions

In order for the monocular pose algorithm to appropriately calibrate and successively reconstruct three dimensional objects it assumes the a priori knowledge of an object within the image's complete dimensions. To observe the effect of the error that is caused when incorrect dimensions are given to the algorithm, the true dimensions of the box were altered. This produced concerning results. By increasing the width of the box by a factor of 10 to 101, we are able to observe the pixel coordinate projection in Figure 7. Given only the image

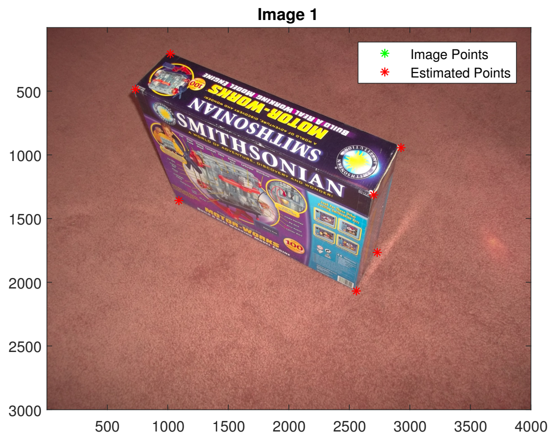

Figure 7: Estimated image coordinates (inaccurate dimensions)

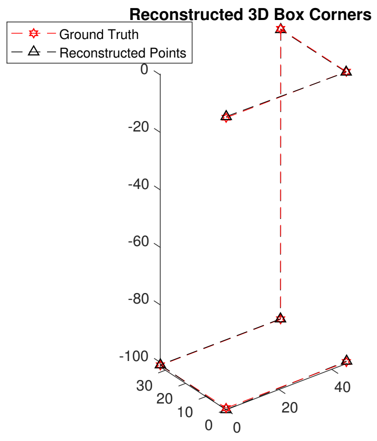

Figure 8: Reconstructed box relative to ground truth (inaccurate dimensions)

coordinate projections it could easily be assumed that the algorithm had appropriately reconstructed the box, despite the wrong dimensions. However, we then project these pixel coordinates and find that in the object frame, the computer believes that the reconstruction should match the dimensions that were provided by the user (see Figure 8).

3 Conclusions

A Code Listings

References