Comparative Evaluation and Refinement of Linear Algebra-Based Camera Calibration Algorithms

Eunkyu Kim[†] and Lucia Rhode[‡]

Project Advisor: Mili Shah[§¶]

Abstract. This paper introduces a linear algebra-based formulation of camera calibration algorithms, focusing on two methods: the regular method and the linearized method. The regular method is computationally more efficient while the linearized method can be adapted to an iterative process to enhance calibration accuracy. To demonstrate the effectiveness of these methods, a marker-based motion tracking experiment utilizing real-life data is conducted. The results showcase the regular method's computational efficiency, while the linearized method's ability to remove outliers is demonstrated through reverse coordinate computation. With a 2.00 mm upper limit threshold, reverse coordinate computation achieved accuracy up to a Frobenius norm of 0.18 mm.

1. Introduction.

Camera calibration finds its applications in various fields including robotics, augmented reality, 3D reconstruction, and object tracking [8]. The availability of reliable calibration algorithms has greatly enhanced the accuracy and reliability of computer vision systems, allowing for more precise and robust analysis of visual data [12].

The real world coordinates refer to the physical three-dimensional space with respect to some origin while the camera coordinates represent the coordinate system of the camera itself with respect to the camera's optical center or the focal point. These two coordinate systems are related by a transformation process. The image captured by the camera consists of a two-dimensional grid of pixels. Each pixel represents a discrete element of the image, containing color and intensity information. The coordinates of a pixel are defined in terms of its position within the image grid. To establish a transformation correspondence between the real-world coordinates and the image coordinates, intrinsic and extrinsic parameters are needed.

Camera calibration is a fundamental process in computer vision and photogrammetry that aims to accurately determine the intrinsic and extrinsic parameters of a camera. Intrinsic characteristics include skew factor (s), focal length (f), and center of the camera ($\mathbf{C}$). On the other hand, extrinsic characteristics include translation ($\mathbf{T}$) and rotation ($\mathbf{R}$) matrices that determine transformation from world to 3D camera coordinates [1]. The purpose of camera calibration is to establish a reliable transformation between the 3D world coordinates and the projected 2D pixel coordinates captured by the camera (Figure 1). By accurately calibrating a camera, one can accurately measure distances, angles, and positions of objects in the real world using images or video acquired from the camera.

Previously, there have been numerous works related to camera calibration in the field of photogrammetry and camera vision due to its potential application in sensors, robotics, and drones. Zhang [12] proposed a closed-form solution calibration method that utilizes planar

[¶]This work was performed under the following financial assistance award 70NANB23H248 from U.S. Department of Commerce, National Institute of Standards and Technology.

[†]Department of Mechanical Engineering, The Cooper Union, New York, NY (kim76@cooper.edu).

[‡]Department of Electrical Engineering, The Cooper Union, New York, NY (lucia.rhode@cooper.edu).

[§]Department of Mathematics, The Cooper Union, New York, NY (mili.shah@cooper.edu).

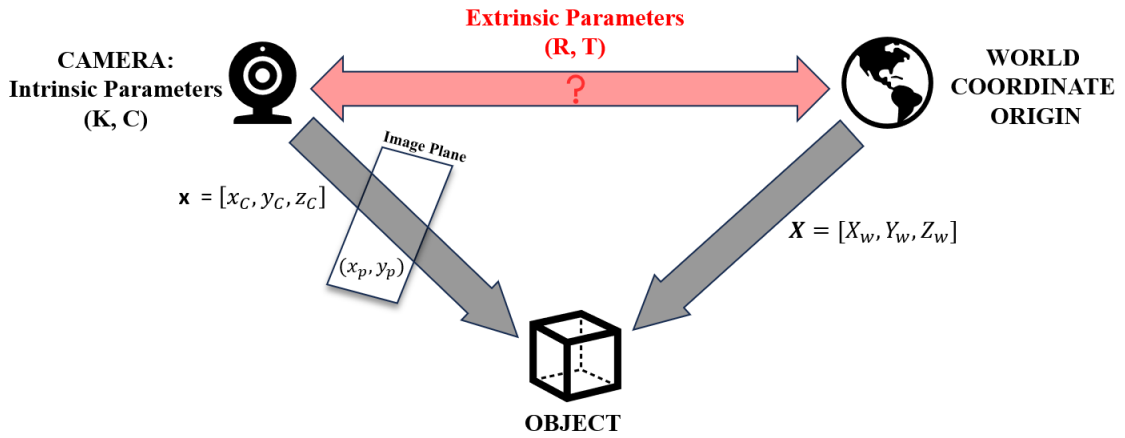

Figure 1: Coordinate systems present in a typical calibration problem. $\mathbf{X}$ is collected via a motion detection system. (x_p, y_p) is collected via image processing on MATLAB. Calibration aims to find $\mathbf{K}$, an upper triangular matrix that contains positional information of the camera, $\mathbf{C}$, and $\mathbf{T}$.

patterns such as a checker board for test-set data. Based off of Zhang's work, many researchers have worked on elevating calibration precision and simultaneously optimizing the iteration processes [7]. Ganesh et al. [2] proposed real time calibration using three pose estimates: camera, fiducial markers, and motion capturing system. In 2017, Syahputra and Pulungan [10] demonstrated obtaining extrinsic parameters via Singular Value Decomposition (SVD).

Most literature involves the presence of a calibration pattern, for example a checkerboard [10]. In this paper, we present a linear algebra based formulation of a camera calibration algorithm which only requires experimental data coordinates of an object. Furthermore, we discuss its underlying principles, and evaluation of the proposed algorithm through a marker-based motion detection experiment.

The paper is organized as follows. The mathematical theory behind the camera calibration algorithm is in Section 2, and the proposed algorithm is presented in Section 3. The experimental set up and results are in Section 4 and Section 5. Conclusions follow in Section 6.

2. Mathematical Theory behind Camera Calibration.

In Figure 2, the 3D world coordinates of a point P can be described as $[X_w, Y_w, Z_w]^T$ while in camera coordinates, it can be described as $[x_C, y_C, z_C]^T$. The object's coordinates in world coordinate system and the camera coordinate system can be connected via a homogeneous transformation as [1]:

$$\begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{2.1}$$

An additional row of 1 is introduced to match the sizes of the matrix products where

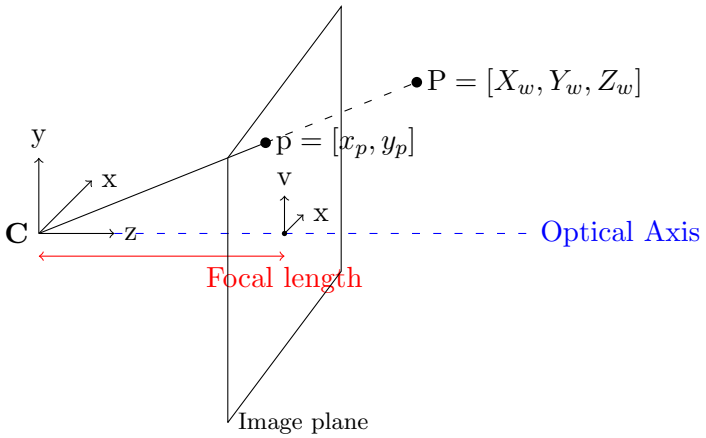

Figure 2: Formation and Capturing of Images: P is the coordinates of a point from the real world coordinates, p is the projected 2D pixel coordinates in the image plane, and **C** is the camera's center.

$\mathbf{R} \in \mathbb{R}^{3\times3}$ and $\mathbf{T} \in \mathbb{R}^3$. Furthermore, introducing the relationship between the 3D camera coordinates and the camera plane coordinates

$$\begin{aligned} u &= fx_C \\ v &= fy_C \\ w &= z_C \end{aligned}$$

where f is the focal length of the camera, Equation (2.1) can be rewritten as [4]:

$$\begin{aligned} \begin{pmatrix} u \\ v \\ w \end{pmatrix} &= \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \end{aligned}$$

Now, by noting that an image is projected to an individual pixel where the corresponding coordinates are [1]:

$$\begin{aligned} u' &= fk_x x_C + x_0 z_C = \alpha_x x_C + x_0 z_C \\ v' &= fk_y y_C + y_0 z_C = \alpha_y y_C + y_0 z_C \\ w' &= z_C \end{aligned} \tag{2.2}$$

k_x and k_y are camera scaling factors and x_0 and y_0 are the coordinates of the image center in pixels and using the Equation (2.2), we arrive at a transformation equation between the 2D

pixel coordinates and the 3D world coordinates [6]:

$$\begin{aligned}\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} &= \begin{pmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}\end{aligned} \tag{2.3}$$

Note that a skew factor s is typically added account for the skew between the axes of the image plane. Typically, one can obtain image plane data in the form of:

$$x_p = \frac{u'}{w'}$$

and

$$y_p = \frac{v'}{w'}$$

Introducing new notations $\tilde{\boldsymbol{x}} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$, $\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} = \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$, and a scaling factor $\lambda = w'$, we can address the issue. With the new notations and the scaling factor, Equation (2.3) is now rewritten as:

$$\lambda \tilde{\boldsymbol{x}} = \boldsymbol{P}\tilde{\boldsymbol{X}} \tag{2.4}$$

where $\boldsymbol{P}$ is a 3×4 matrix containing the unknown characteristics of the camera. Then we now consider,

$$\begin{aligned}\lambda \tilde{\boldsymbol{x}} = \boldsymbol{P}\tilde{\boldsymbol{X}} &= \begin{pmatrix} \boldsymbol{p_1^T} \\ \boldsymbol{p_2^T} \\ \boldsymbol{p_3^T} \end{pmatrix} \tilde{\boldsymbol{X}} \\ &= \begin{pmatrix} \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 & 0 \\ 0 & \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 \\ 0 & 0 & \tilde{\boldsymbol{X}}^{\boldsymbol{T}} \end{pmatrix} \begin{pmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \boldsymbol{p_3} \end{pmatrix}\end{aligned}$$

Subsequently, simple algebraic manipulation leads to the following matrix multiplication relationship.

$$\mathbf{0} = \begin{pmatrix} \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 & 0 \\ 0 & \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 \\ 0 & 0 & \tilde{\boldsymbol{X}}^{\boldsymbol{T}} \end{pmatrix} \begin{pmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \boldsymbol{p_3} \end{pmatrix} - \lambda \tilde{\boldsymbol{x}}$$

$$\mathbf{0} = \begin{pmatrix} \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 & 0 & -x_p \\ 0 & \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 & -y_p \\ 0 & 0 & \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \boldsymbol{p_3} \\ \lambda \end{pmatrix} = \mathbf{A}_\lambda \mathbf{P}_\lambda \tag{2.5}$$

Note that Equation (2.5) contains only one pixel coordinate data. In the most generic case, $\mathbf{A}_\lambda \mathbf{P}_\lambda$ can be written as follows:

$$\begin{pmatrix} \tilde{\boldsymbol{X}}_{\boldsymbol{1}}^{\boldsymbol{T}} & 0 & 0 & -x_1 & 0 & 0 & \dots \\ 0 & \tilde{\boldsymbol{X}}_{\boldsymbol{1}}^{\boldsymbol{T}} & 0 & -y_1 & 0 & 0 & \dots \\ 0 & 0 & \tilde{\boldsymbol{X}}_{\boldsymbol{1}}^{\boldsymbol{T}} & 1 & 0 & 0 & \dots \\ \tilde{\boldsymbol{X}}_{\boldsymbol{2}}^{\boldsymbol{T}} & 0 & 0 & 0 & 0 & -x_2 & \dots \\ 0 & \tilde{\boldsymbol{X}}_{\boldsymbol{2}}^{\boldsymbol{T}} & 0 & 0 & 0 & -y_2 & \dots \\ 0 & 0 & \tilde{\boldsymbol{X}}_{\boldsymbol{2}}^{\boldsymbol{T}} & 0 & 0 & 1 & \dots \end{pmatrix} \begin{pmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \boldsymbol{p_3} \\ \lambda^1 \\ \lambda^2 \\ \vdots \end{pmatrix}$$

for N = number of distinct data points.

Onwards, the calibration method using $\mathbf{A}_\lambda$ is called the linearized method.

The disadvantage of this form is that the matrix size grows as additional points are considered. As an additional data point is added, an additional 3×12 matrix is added below the left matrix with a column of zeros appearing on the right end. Simultaneously, a new λ appears at the bottom of the right matrix. Therefore, we propose an alternative method to approach the calibration problem [13]. Equation (2.4) implies that the $\lambda\tilde{\boldsymbol{x}}$ and $\boldsymbol{P}\tilde{\boldsymbol{X}}$ are collinear 3×1 vectors. Therefore, the cross product between $\lambda\tilde{\boldsymbol{x}}$ and $\boldsymbol{P}\tilde{\boldsymbol{X}}$ is 0 for all data points [1]. Hence,

$$\mathbf{0} = \lambda\tilde{\boldsymbol{x}} \times \boldsymbol{P}\tilde{\boldsymbol{X}} = \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} \times \begin{pmatrix} \boldsymbol{p_1^T}\tilde{\boldsymbol{X}} \\ \boldsymbol{p_2^T}\tilde{\boldsymbol{X}} \\ \boldsymbol{p_3^T}\tilde{\boldsymbol{X}} \end{pmatrix} = \begin{pmatrix} v'\boldsymbol{p_3^T}\tilde{\boldsymbol{X}} - w'\boldsymbol{p_2^T}\tilde{\boldsymbol{X}} \\ w'\boldsymbol{p_1^T}\tilde{\boldsymbol{X}} - u'\boldsymbol{p_3^T}\tilde{\boldsymbol{X}} \\ u'\boldsymbol{p_2^T}\tilde{\boldsymbol{X}} - v'\boldsymbol{p_1^T}\tilde{\boldsymbol{X}} \end{pmatrix} \tag{2.6}$$

Dividing through Equation (2.6) by λ,

$$\begin{aligned} \mathbf{0} &= \begin{pmatrix} y_p\boldsymbol{p_3^T}\tilde{\boldsymbol{X}} - \boldsymbol{p_2^T}\tilde{\boldsymbol{X}} \\ \boldsymbol{p_1^T}\tilde{\boldsymbol{X}} - x_p\boldsymbol{p_3^T}\tilde{\boldsymbol{X}} \\ x_p\boldsymbol{p_2^T}\tilde{\boldsymbol{X}} - y_p\boldsymbol{p_1^T}\tilde{\boldsymbol{X}} \end{pmatrix} \\ &= \begin{pmatrix} 0 & -\tilde{\boldsymbol{X}}^{\boldsymbol{T}} & y_p\tilde{\boldsymbol{X}}^{\boldsymbol{T}} \\ \tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 & -x_p\tilde{\boldsymbol{X}}^{\boldsymbol{T}} \\ -y_p\tilde{\boldsymbol{X}}^{\boldsymbol{T}} & x_p\tilde{\boldsymbol{X}}^{\boldsymbol{T}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \boldsymbol{p_3} \end{pmatrix} = \mathbf{AP} \end{aligned} \tag{2.7}$$

Again in the most generic case, $\mathbf{AP}$ can be written as follows:

$$\begin{pmatrix} 0 & \tilde{\boldsymbol{X}}_1^T & y_p^1 \tilde{\boldsymbol{X}}_1^T \\ \tilde{\boldsymbol{X}}_{\mathbf{1}}^{\boldsymbol{T}} & 0 & -x_p^1 \tilde{\boldsymbol{X}}_{\mathbf{1}}^{\boldsymbol{T}} \\ -y_p^1 \tilde{\boldsymbol{X}}_{\mathbf{1}}^{\boldsymbol{T}} & x_p^1 \tilde{\boldsymbol{X}}_{\mathbf{1}}^{\boldsymbol{T}} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \tilde{\boldsymbol{X}}_{(i)}^T & y_p^{(i)} \tilde{\boldsymbol{X}}_{(i)}^T \\ \tilde{\boldsymbol{X}}_{(\boldsymbol{i})}^{\boldsymbol{T}} & 0 & -x_p^{(i)} \tilde{\boldsymbol{X}}_{(\boldsymbol{i})}^{\boldsymbol{T}} \\ -y_p^{(i)} \tilde{\boldsymbol{X}}_{(\boldsymbol{i})}^{\boldsymbol{T}} & x_p^{(i)} \tilde{\boldsymbol{X}}_{(\boldsymbol{i})}^{\boldsymbol{T}} & 0 \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \boldsymbol{p_3} \end{pmatrix}$$

for $i = 1, 2, ..., N$ where $N =$ number of distinct data points.

Onwards, the calibration method using $\mathbf{A}$ is called the regular method.

First of all, one can see that ($x_p \times$the first row)$-$($y_p \times$the second row) is equal to the third row. Hence, this is a rank deficient system. In other words, there are 2 independent equations for 11 unknowns ignoring the scale factor. Therefore, we need 6 point correspondences that are not all co-planar. Second, this form has an advantage of the matrix size not growing as rapidly as Equation (2.5). In fact, for every data point, an additional 3×9 matrix is being added with no additional zeros being added. Therefore, it is computationally more efficient. As an additional data point is added, additional columns do not appear as λ terms are not being added to the right matrix. However, this also means that λ, the focal length, cannot be directly extracted from the system. As will be shown in Section 4, the λ terms play a pivotal role in determining the accuracy of calibration.

To find the components of $\mathbf{P}$ in Equation (2.4), the generic idea is that Equation (2.1) can be expressed as a product of an upper triangular matrix $\boldsymbol{K}$, an orthogonal matrix $\boldsymbol{R}$ and the camera's center $\boldsymbol{C}$ [1]. Furthermore another important relationship is $\mathbf{T} = -\mathbf{RC}$ where $\mathbf{T}$ is the translation vector to align the world coordinate origin to the camera coordinate's origin. The reason for the presence of the negative sign is because $\mathbf{RC}$ describes the camera center in the camera coordinates system. Hence, the negative sign essentially aligns the world coordinate system in such a way that the camera center appears to be at the origin and oriented along the standard axes.

$$\begin{aligned} \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} &= \begin{pmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{T} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{I_{3\times 3}} & \boldsymbol{0_{3\times 1}} \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{-RC} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & -\boldsymbol{RC} \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{R} \begin{pmatrix} \boldsymbol{I}_{3\times 3} & -\boldsymbol{C} \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \\
&= \boldsymbol{KR} \begin{pmatrix} \boldsymbol{I}_{3\times 3} & -\boldsymbol{C} \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}
\end{aligned} \tag{2.8}
$$

while noting that $\boldsymbol{C^T}$ is a null vector of $\boldsymbol{P}$ [3]:

$$
\boldsymbol{KR} \begin{pmatrix} \boldsymbol{I_{3\times 3}} & -\boldsymbol{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{C} \\ 1 \end{pmatrix} = \boldsymbol{KR}(\boldsymbol{C} - \boldsymbol{C}) = \boldsymbol{0} \tag{2.9}
$$

3. Algorithm. Let 3D world coordinates of an object be $\boldsymbol{X_i}$. Let image data point be $\boldsymbol{x_i}$. Formulate matrix $\boldsymbol{A_\lambda}$ using Equation (2.5) (Linearized Method) or $\boldsymbol{A}$ using Equation (2.7) (Regular Method).

$$
\boldsymbol{A_\lambda} = \begin{pmatrix} \boldsymbol{\tilde{X}_1^T} & 0 & 0 & -x_1 & 0 & 0 & \dots \\ 0 & \boldsymbol{\tilde{X}_1^T} & 0 & -y_1 & 0 & 0 & \dots \\ 0 & 0 & \boldsymbol{\tilde{X}_1^T} & 1 & 0 & 0 & \dots \\ \boldsymbol{\tilde{X}_2^T} & 0 & 0 & 0 & 0 & -x_2 & \dots \\ 0 & \boldsymbol{\tilde{X}_2^T} & 0 & 0 & 0 & -y_2 & \dots \\ 0 & 0 & \boldsymbol{\tilde{X}_2^T} & 0 & 0 & 1 & \dots \end{pmatrix} \tag{3.1}
$$

$$
\boldsymbol{A} = \begin{pmatrix} 0 & \tilde{\boldsymbol{X}}_1^T & y_p^1 \tilde{\boldsymbol{X}}_1^T \\ \boldsymbol{\tilde{X}_1^T} & 0 & -x_p^1 \boldsymbol{\tilde{X}_1^T} \\ -y_p^1 \boldsymbol{\tilde{X}_1^T} & x_p^1 \boldsymbol{\tilde{X}_1^T} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \tilde{\boldsymbol{X}}_N^T & y_p^N \tilde{\boldsymbol{X}}_N^T \\ \boldsymbol{\tilde{X}_N^T} & 0 & -x_p^N \boldsymbol{\tilde{X}_N^T} \\ -y_p^N \boldsymbol{\tilde{X}_N^T} & x_p^N \boldsymbol{\tilde{X}_N^T} & 0 \end{pmatrix} \tag{3.2}
$$

Algorithm 3.1 Camera Calibration Algorithm

1: Collect 3D world coordinate data of the object using motion detection system. Collect 2D pixel data of the object using a camera image.
2: Linearized Method: Calculate SVD of $\boldsymbol{A_\lambda}$ and let $\boldsymbol{p}$ be the unit right singular vector corresponding to the smallest σ of $\boldsymbol{A_\lambda}$ [11].
Regular Method: Calculate SVD of $\boldsymbol{A}$ and let $\boldsymbol{p}$ be the unit right singular vector corresponding to the smallest σ of $\boldsymbol{A}$
3: Normalize the vector $\boldsymbol{p}$ by the norm of $\boldsymbol{p}(9:11)$.
4: Let

$$\boldsymbol{P} = [\boldsymbol{p}(1:4)^T; \boldsymbol{p}(5:8)^T; \boldsymbol{p}(9:12)^T]$$

noting that the focal length values of each data point can be found in $\boldsymbol{p}(13,:)$ for linearized method only [9].
5: Calculate the SVD of $\boldsymbol{P}$. Set $\tilde{\boldsymbol{C}}$ to be the unit right singular vector corresponding to the smallest σ of $\boldsymbol{P}$.
6: Scale $\tilde{\boldsymbol{C}}$ such that the last entry is 1.
7: Calculate the QR factorization

$$[\boldsymbol{q}, \boldsymbol{r}] = qr(\boldsymbol{P}(1:3, 1:3)^{-1})$$

[5].
8: Let $\boldsymbol{K} = \boldsymbol{r}^{-1}$ and $\boldsymbol{R} = \boldsymbol{q}^{-1}$. Scale $\boldsymbol{K}$ such that $\boldsymbol{K}(3,3) = 1$
9: Compute $\boldsymbol{T} = -\boldsymbol{RC}$

(a) Image of the motion detection calibration wand

(b) Image processing for obtaining 2D coordinates

Figure 3: Experimental Setup

4. Experimental Set Up. In this section, we present an experimental demonstration of the camera calibration algorithm. Figure 3a shows the motion detection system calibration wand that we used as a marker-based position tracking object. The wand consists of five infrared (IR) markers that the motion detection system can track and obtain 3D world coordinates. Each marker can be considered as a distinct data point with well known 3D coordinates with respect to the motion detection system origin. In total, 15 different positions were captured adding up to 75 distinct data points. At the same time, 2D pixel coordinates were collected via a cellphone. The pixel coordinates were obtained from a video recording of the cellphone via image processing toolbox in MATLAB as shown in Figure 3b.

5. Results. We present the $\mathbf{K}, \mathbf{R}$, and $\mathbf{C}$ matrices which contain all the camera properties. Table 1 below summarizes the experimental results after running the calibration algorithm with 75 marker position data.

Table 1: Results Summary

	Linearized Method	*Iterative Linearized Method*	*Regular Method*
$\mathbf{K}$	$\begin{pmatrix} -1531.5 & 145.5 & 852.9 \\ 0 & -1597.5 & 497.1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1537 & 191.4 & 751.9 \\ 0 & -1562.4 & 326 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1532.4 & 39.9776 & 1078.6 \\ 0 & -1471.1 & 65.056 \\ 0 & 0 & 1 \end{pmatrix}$
$\mathbf{R}$	$\begin{pmatrix} -0.9994 & -0.0138 & -0.0303 \\ -0.276 & -0.1655 & 0.9858 \\ -0.0186 & 0.9861 & 0.1650 \end{pmatrix}$	$\begin{pmatrix} -0.9952 & -0.0919 & -0.0345 \\ -0.0083 & -0.2722 & 0.9622 \\ -0.0978 & 0.9578 & 0.2701 \end{pmatrix}$	$\begin{pmatrix} -0.9921 & 0.1248 & -0.0110 \\ -0.0649 & -0.4365 & 0.8974 \\ 0.1072 & 0.8910 & 0.4411 \end{pmatrix}$
$\mathbf{C}$	$\begin{pmatrix} -619.4 \\ -1715 \\ 728.6 \end{pmatrix}$	$\begin{pmatrix} -632.7 \\ -1765.8 \\ 782.8 \end{pmatrix}$	$\begin{pmatrix} -621.7 \\ -1731.7 \\ 706.6 \end{pmatrix}$
$\|\|\mathbf{C} - \mathbf{C}_{\text{truth}}\|\|_2$	61.8	17.3	70.9

The linearized method has strong preference over the regular method as one can utilize the intrinsic and extrinsic camera characteristics to convert measured 2D pixel coordinates to corresponding 3D world coordinates. This can be done by reversing the derivation shown in 2.9 (Appendix A). A comparative analysis between reverse-calculated coordinates and experimental coordinates serves as a reliable means of conducting a rigorous sanity check. Hence, we specifically focus on analyzing and improving the linearized method results. The $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{C}$ matrices obtained after 56 iterations with $2mm$ Euclidean norm threshold are shown in the right-most column of Table 1.

In Figure 4, the red Os represent the actual 3D world coordinates obtained via the motion detection system. The blue rhombuses represent the approximated 3D world coordinates based on the camera characteristics obtained via the linearized method algorithm. Note that this type of comparison is not trivial for the regular method as focal length λ information cannot be directly obtained from the regular method. The "backtracking" analysis enables one to gauge the accuracy of data at hand. By setting an appropriate point-by-point Euclidean norm threshold, one can go through an iterative process to remove so called off-set data points and approximate the camera characteristics with a greater degree of accuracy. The bottom Oriented View plot shows the approximated coordinates after 30 iterations of 0.2 mm thresh-

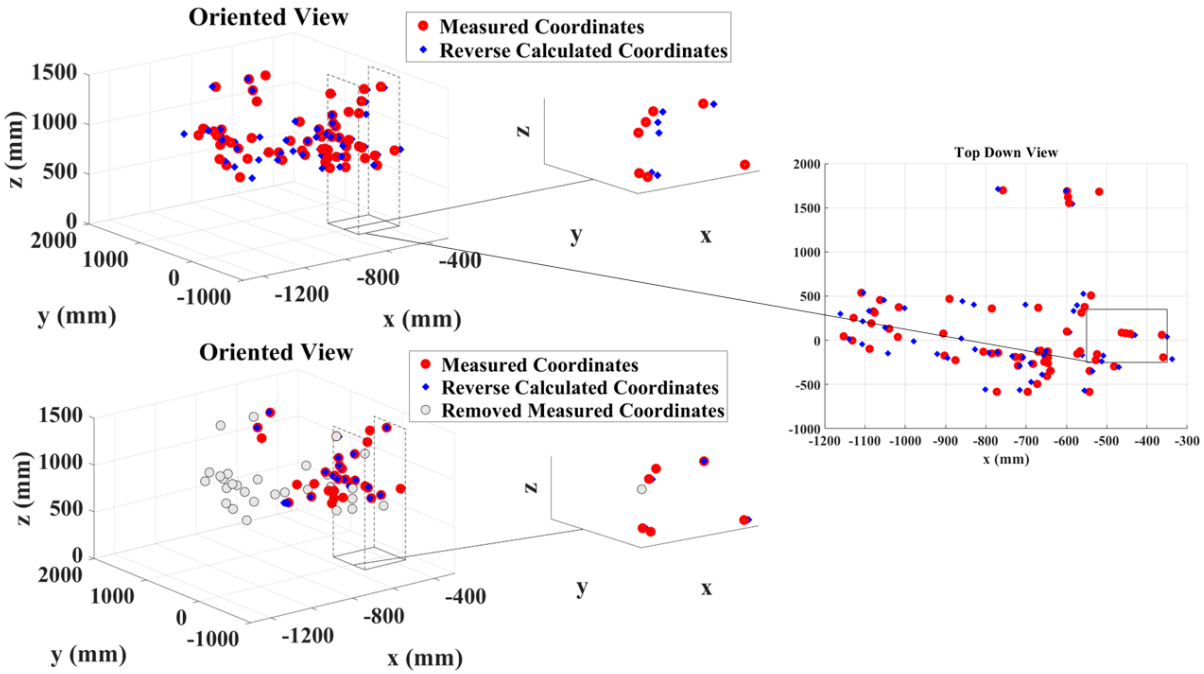

Figure 4: Reversed-calculated coordinates comparison plots. The top Oriented View plot shows the coordinates after initial calibration. The bottom Oriented View plot shows the coordinates after 30 iterations. The zoomed-in plots show the cuboid volume indicated by dotted lines. The Top Down View plot shows the points enclosed by a patch of area in the XY plane.

old. The grey circles are the removed data points. One can see that the approximations after 30 iterations are closer to actual measurements than the initial calibration. This suggests that the linearized method, through an iterative process, can be a viable tool for experimentally determining the camera properties to a greater accuracy compared to the regular method. With the above threshold parameter, the frobenius norm of the world coordinates and reverse calculated coordinates decrease from 4.08 mm to 0.18 mm.

During data collection, 4 pearl hard markers were attached, not all planar, on the back surface of the smartphone. Although not precise, the observed (experimental) coordinates of the markers provide a rough indication of where the ground truth $\mathbf{C}_{\text{truth}}$ is. The approximate Euclidean norm between the ground truth and calculated center of the camera is presented in the last column of Table 1. To identify the validity of the $\mathbf{K}, \mathbf{R}, \mathbf{C}$ matrices obtained from the proposed algorithm, we simply plot the matrix $\mathbf{C}$ as a three dimensional vector. Since the $\mathbf{C}$ matrix contains the coordinates of the center of the camera in the world coordinate system, the resulting point should be approximately in a region enclosed by the 4 markers.

In Figure 5, The solid-line patch represents the actual geometry of the smartphone used. The dotted-line patch represents the approximate smartphone case on which the markers were attached. The approximate location of the camera is shown for reference purposes.

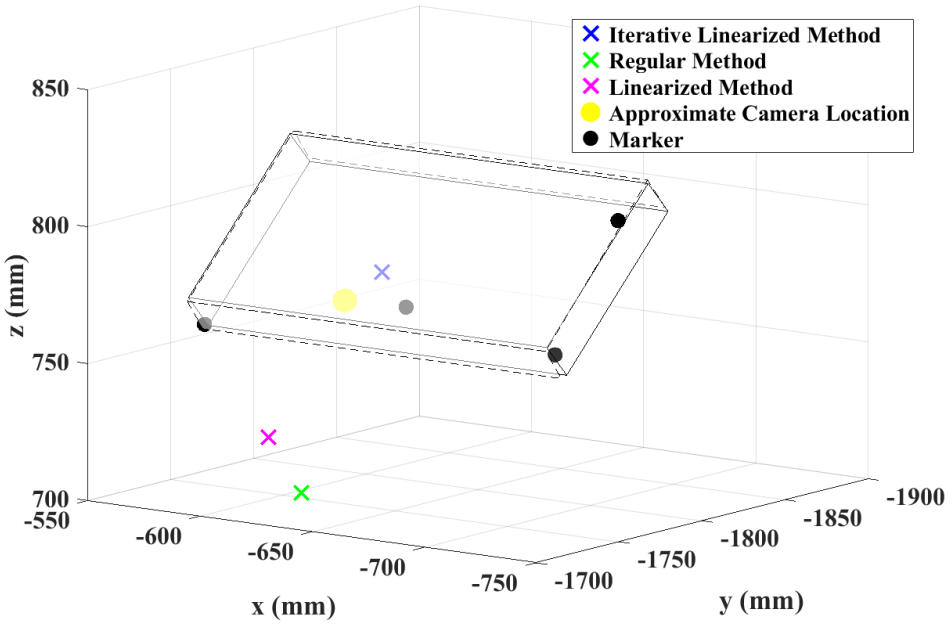

Figure 5: 3D Scatter plot of the **C** Vectors obtained from linearized method, iterative linearized method, and regular method with the coordinates of pearl markers. The approximate location of the camera center is also shown.

The markers are not exactly aligned with the patch because some of them were placed on the corner of the smartphone case to create non-planar geometry. The **C** vectors obtained from the regular method and the initial linearized method fail to pass the ground truth validity test. However, the iterative linearized method shows that the linearized method can still obtain proper camera parameters via an iteration process. The 2-norm error for iterative linearized method is 17.3 mm which is expected as the motion detection system had inherent vibrations due to building conditions. Furthermore, the pixel data collection was done using basic MATLAB image processing library with an ordinary laptop mouse. On the other hand, the regular method, although computationally faster than the linearized method, has no method of detecting experimental anomalies and iterating to reduce computational discrepancies.

6. Conclusions. In this paper, we presented a linear algebra based camera calibration algorithm utilizing SVD and QR factorization. The proposed algorithm was verified using a marker-based motion detection system. Although the regular method has the advantage of being computationally more efficient, the linearized method has the strong preference as the initial calibration offset can be overcome with an outlier-removal iterative process. In the future, we wish to add rigorous analysis on why the iterative linearized method is preferable as well as a convergence proof for such method. We hope to use this work as a basis to

study dynamic motion calibration, markerless-motion capturing or multi-camera calibration problems.

Appendix A. 2D coordinates transformation to 3D coordinates. In Equation (2.9), the relationship between image coordinate system and the world coordinate system was shown as follows

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \boldsymbol{K}\boldsymbol{R}\begin{pmatrix} \boldsymbol{I}_{3\times 3} & -\boldsymbol{C} \end{pmatrix}\begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$$

Suppose that the camera properties $\boldsymbol{K}, \boldsymbol{R}$, and $\boldsymbol{C}$ are found using the calibration algorithm. Then, one can "reverse" calculate the camera coordinates into world coordinates as follows

$$\begin{aligned} \boldsymbol{R^T K^{-1}}\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & -C_1 \\ 0 & 1 & 0 & -C_2 \\ 0 & 0 & 1 & -C_3 \end{pmatrix}\begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} x_w - C_1 \\ y_w - C_2 \\ z_w - C_3 \end{pmatrix} \\ &= \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} - \boldsymbol{C} \end{aligned} \tag{A.1}$$

where C_1, C_2, and C_3 are the entries of the column vector $\boldsymbol{C}$. Then,

$$\begin{aligned} \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} &= \boldsymbol{R^T K^{-1}}\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} + \boldsymbol{C} \\ &= \boldsymbol{R^T K^{-1}}\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} - \boldsymbol{R^T T} \\ &= \boldsymbol{R^T}(\boldsymbol{K^{-1}}\begin{pmatrix} x_p w' \\ y_p w' \\ w' \end{pmatrix} - \boldsymbol{T}) \\ &= \boldsymbol{R^T}(\boldsymbol{K^{-1}}\begin{pmatrix} x_p f \\ y_p f \\ f \end{pmatrix} - \boldsymbol{T}) \end{aligned} \tag{A.2}$$

Acknowledgments. We would like to acknowledge our research advisor, Dr. Mili Shah, for endless support and motivation throughout the whole process. We also would like show gratitude to Dr. D. M. Luchtenburg and Mr. Michael Giglia of The Cooper Union Mechanical Engineering department for letting us use the motion detection system.

REFERENCES

[1] P. Corke, *Robotics, Vision and Control Fundamental Algorithms in MATLAB*, Springer, 2nd ed., 2017.

[2] P. Ganesh, K. Volle, P. Buzaud, K. Brink, and A. Willis, *Extrinsic calibration of camera and motion capture systems*, SoutheastCon 2021, https://doi.org/10.1109/SOUTHEASTCON45413.2021.9401911.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, third ed., 1996.

[4] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2 ed., 2004, https://doi.org/10.1017/CBO9780511811685.

[5] R. I. Hartley, *Self-calibration of stationary cameras*, Int. J. Comput. Vis., 22 (1997), pp. 5–23, https://doi.org/10.1023/A:1007957826135, https://doi.org/10.1023/A:1007957826135.

[6] J. Heikkila and O. Silven, *A four-step camera calibration procedure with implicit image correction*, in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997, pp. 1106–1112, https://doi.org/10.1109/CVPR.1997.609468.

[7] F. li Guan, A. jun Xu, and G. yu Jiang, *An improved fast camera calibration method for mobile terminals*, JIPS(Journal of Information Processing Systems), 15 (2019), pp. 1082–1095, https://doi.org/10.3745/JIPS.02.0117, http://jips-k.org/q.jips?cp=pp&pn=704.

[8] K. Liao, L. Nie, S. Huang, C. Lin, J. Zhang, Y. Zhao, M. Gabbouj, and D. Tao, *Deep learning for camera calibration and beyond: A survey*, 2023, https://arxiv.org/abs/2303.10559.

[9] P. Sturm, Z. Cheng, P. Chen, and A. Poo, *Focal length calibration from two views: method and analysis of singular cases*, Computer Vision and Image Understanding, 99 (2005), pp. 58–95, https://doi.org/https://doi.org/10.1016/j.cviu.2004.11.002, https://www.sciencedirect.com/science/article/pii/S1077314204001699.

[10] H. Syahputra and R. Pulungan, *Camera calibration for 3d leaf-image reconstruction using singular value decomposition*, International Journal of Advanced Computer Science and Applications, 8 (2017), https://doi.org/10.14569/IJACSA.2017.080950, http://dx.doi.org/10.14569/IJACSA.2017.080950.

[11] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, 1997.

[12] Z. Zhang, *A flexible new technique for camera calibration*, 1998, https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf (accessed 2018-08-13).

[13] L. Zhou, Z. Li, and M. Kaess, *Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences*, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 5562–5569, https://doi.org/10.1109/IROS.2018.8593660.