Camera Calibration and 3D Distance Estimation in Euclidean Space Using Vector Representation on the Intel RealSense D455 Depth Camera

Muhammad Faiz Alfada Dharma - 13524097
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524097@std.stei.itb.ac.id, [2]faizalfadha@gmail.com

***Abstract*—This paper studies how linear algebra and Euclidean geometry can be used to estimate 3D distance from a depth camera. We use the Intel RealSense D455 as a case study. First, we review the pinhole camera model and the intrinsic camera matrix, which describe how 3D points are projected to 2D image pixels. Then we show how pixel coordinates and depth values can be converted back into 3D coordinates in the camera frame using vector and matrix operations. The distance from the camera to the object is obtained as the Euclidean norm of this 3D vector. We also discuss the idea of camera calibration as a linear system solved by least squares. This gives a simple mathematical framework that links depth sensing with core concepts from linear algebra.**

***Keywords*—Camera Calibration, 3D Distance Estimation, Depth Camera, Intel RealSense D455**

I. Introduction

Depth information is important in a lot of modern applications, such as mobile robots or even 3D mapping. These systems must know where objects are and how far they are in a three dimensional (3D) space, not just see them in a 2D image. A depth camera helps them by giving each pixel a value that represents the distance from the camera to the scene at that point.

A 3D point can be written as a three dimensional vector (X, Y, Z) and the distance from the camera to that point is the Euclidean norm of this vector. The mapping from 3D points to 2D image pixels can be described by the pinhole camera model, which uses matrixes to represent linear transformations and translations.

To use a camera for metric measurements, we first need camera calibration. Calibration finds the internal camera parameters (such as focal lengths and principal point) and the camera pose. In many methods, these unknown parameters are found by solving a system of linear equations using the least-squares method.

In this paper, the Intel RealSense D455 depth camera is used as a concrete example. We focus on the mathematical model. Starting from calibrated intrinsic parameters and depth values, it shows how to recover 3D coordinates in the camera coordinate system and how to compute the 3D distance as the length of a vector in Euclidean space.

More specifically, this work investigates:

- How can 3D points and camera–object distance be represented using vectors and Euclidean distance?
- How does the pinhole camera model use matrixes and linear transformations to map 3D points to 2D pixels?
- How can camera calibration be formulated as a linear system solved by least squares?
- How can pixel coordinates and depth values from the Intel RealSense D455 be transformed into 3D coordinates and distances using this model?

II. Theoretical Foundations

A. Vectors and Euclidean Distance

In linear algebra, many physical quantities can be represented either as scalars or as vectors. A scalar only has magnitude, while a vector has both magnitude and direction [1]. Three Dimensional in euclidean space can be described using,

$$\mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

A vector can be visualized as a directed segment starting from the origin and ending at the point with coordinates $(X,\ Y,\ Z)$ [1].

The euclidean length (Euclidean norm) of a vector p is defined by:

$$d\ =\ ||p||\ = \sqrt{X^2 + Y^2 + Z^2}$$

B. Matrix Representation and Linear Transformations

A mapping T from one vector space to another is called a linear transformation if it preserves vector addition and scalar multiplication,

$$T(u + v) = T(u) + T(v) \quad , \quad T(ku) = kT(u)$$

For all vectors $u,\ v$ and scalars k [2].

When we work with column vectors of numbers, a linear transformation can be represented by a matrix. If the input is a 3D vector and the output is also 3D, there exists a 3 x 3 matrix A such that,

$$T(x) = Ax$$

In camera geometry, two coordinate systems are used, such as world coordinates, which describe the position of points in a global reference frame, and camera coordinates, which describe positions relative to the camera. The mapping from world coordinates to camera coordinates can be written as:

$$X_c = RX_w + t$$

Where:

R is a 3 x 3 rotation matrix and t is a translation vector. This is a combination of a linear transformation and a translation [4].

C. Pinhole Camera Model and Intrinsic Matrix

Computer vision commonly uses the pinhole camera model to describe how a camera forms an image [4]. In this model, a 3D point is projected onto a 2D image plane through a single projection center. Using homogeneous coordinates, the relationship can be written as

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\begin{bmatrix} R & t \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Where:

(u, v) are image pixel coordinates, s is a scale factor, R and t are the extrinsic parameters (pose of the camera), and K is the intrinsic matrix. The intrinsic matrix has the form of,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where:

f_x and f_y are focal lengths measured in pixels, and (c_x, c_y) is the principal point of the camera. These parameters describe how the camera converts physical rays of light into pixel coordinates on the sensor [4].

D. From Pixel and Depth to 3D Coordinates

Depth cameras provide, for each pixel (u, v), a depth value Z, which usually represents the distance along the camera's optical axis. Once the intrinsic parameters f_x, f_y, c_x, and c_y are known from calibration, the 3D coordinates of the corresponding point in the camera coordinate system can be recovered as [4],

$$X = \frac{(u - c_x)}{f_x} Z \ , Y = \frac{(v - c_y)}{f_y} Z$$

Together with the depth Z, the 3D vector will be,

$$\mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The 3D distance from the camera to the point is then the Euclidean length of this vector:

$$d = ||p|| = \sqrt{X^2 + Y^2 + Z^2}$$

E. Camera Calibration as a Linear System

Camera calibration is the process of estimating the intrinsic and extrinsic parameters of the camera. A standard approach is to capture images of a calibration pattern whose 3D geometry is known. Each corner of the pattern has known 3D coordinates and is detected in the image as a 2D pixel location. From each such pair, equations can be derived that involve the unknown camera parameters [4].

By stacking the equations from many points, we obtain a system of linear equations:

$$A\theta = b$$

Where:

A is a matrix built from the known 3D and 2D coordinates, θ is a vector of unknown camera parameters and b is a vector derived from the image measurements. Each 3D–2D correspondence contributes projection equations that can be written, under the pinhole model, as $u_i = f_x(\frac{X_i}{Z_i}) + c_x$ and $v_i = f_y(\frac{Y_i}{Z_i}) + c_y$[6][4]. Rearranging shows the unknown intrinsics appear linearly, for example $u_i = \begin{bmatrix} \frac{X_i}{Z_i} & 1 \end{bmatrix}\begin{bmatrix} f_x & c_x \end{bmatrix}^T$ and $v_i = \begin{bmatrix} \frac{Y_i}{Z_i} & 1 \end{bmatrix}\begin{bmatrix} f_y & c_y \end{bmatrix}^T$ [6]. By stacking many points, the matrix A is formed from terms like $\frac{X_i}{Z_i}$ and $\frac{Y_i}{Z_i}$, while b is formed from the measured pixel values u_i and v_i [6][4]. Since the number of correspondences is typically larger than the number of unknown parameters, the system becomes overdetermined and least squares finds the parameters that minimize the total squared fitting error. [6][4]. This system is usually solved in the least-squares sense

$$\hat{\theta} = arg\,min\,||A\theta - b||^2$$

which is a standard linear algebra technique used in many calibration algorithms [4].

F. Intel RealSense D455 Depth Camera

The Intel RealSense D455 is a depth camera in the D400 family. It provides depth measurements and supports resolutions up to 1280 x 720 at high frame rates [5]. The Intel RealSense D455 depth camera is a stereo based RGB-D sensor that provides synchronized color and depth images over a working range of roughly several meters. The RealSense software development kit exposes the intrinsic parameter f_x , f_y , c_x , and c_y , and can be read directly by software [5]. Its intrinsic parameters and depth images provide the input to the mathematical model described above, while the resulting 3D vectors and distances are interpreted using the concepts of vectors, matrixes, and Euclidean distance

III. METHODOLOGY

A. Overview

The goal of this study is to apply the mathematical model from Section II to real data from Intel RealSense D455 depth camera. A 3D point is represented as a vector $p = (X, Y, Z)^T$ in Euclidean space and the distance from the camera to the point is the Euclidean norm.

The camera is modeled using the pinhole camera formulation with intrinsic matrix K and extrinsic parameters (R, t). In general, the mapping from world coordinates P_w to camera coordinates P_c is given by the linear transformation and translation,

$$P_c = RP_w + t$$

and the projection to image pixels satisfies,

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

In this experiment, the world coordinate system is chosen to coincide with the camera coordinate system, so that R becomes the identity and t is zero.

B. Experimental Setup

An Intel RealSense D455 depth camera is placed on a stable surface facing a static object such as a wall or a door. The camera is connected to a computer running Python. For each experiment:

- The object is placed at a different position along the viewing direction of the camera.
- The physical distance from the front of the camera to the object is measured using a ruler and it is known as the true distance (d_{true}) in meters
- The camera position and orientation are kept fixed throughout all experiments.

The coordinate system used in the experiment is chosen to coincide with the camera coordinate system. In terms of the general model in Section II, this corresponds to using the camera frame as the world frame, so that the extrinsic transformation reduces to the identity rotation and zero translation. This emphasizes the role of the intrinsic matrix K and the vector representation of points in the camera frame.

C. Data Acquisition Using Intel RealSense and Python

Depth data and intrinsic parameters are obtained in Python using the *pyrealsense2* library,

```
import pyrealsense2 as rs
import math

pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640,
480, rs.format.z16, 30)
profile = pipeline.start(config)

for _ in range(10):
    frames = pipeline.wait_for_frames()

frames = pipeline.wait_for_frames()
depth_frame = frames.get_depth_frame()

depth_profile =
depth_frame.profile.as_video_stream_profile()
intr = depth_profile.get_intrinsics()

fx = intr.fx
fy = intr.fy
cx = intr.ppx
cy = intr.ppy
width = intr.width
height = intr.height

print("===== Intrinsics =====")
print("- focal length:")
print("  fx =", fx)
print("  fy =", fy)
print("  cx =", cx)
print("  cy =", cy)
print("- principle point:")
print(" width =", width, " height =", height)
```

First, a depth stream with resolution 640 x 480 and frame rate 30 fps is enabled. This fixes the image size used in the pinhole camera model. After starting the pipeline, several initial frames are discarded to allow automatic exposure and depth estimation to stabilize.

Next, intrinsic parameters of the depth stream are read from the camera. The values f_x , f_y , c_x , and c_y correspond to the focal lengths and principal point coordinates in pixels, and $width$ and $height$ are the

image dimensions. These parameters form the intrinsic matrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

which is the same intrinsic matrix introduced in the pinhole camera model in Section II. These parameters are the result of an internal calibration process that can be expressed by solving a linear system $A\theta = b$. In this study, we use the intrinsic parameters provided by the Intel RealSense SDK as the calibration output.

D. 3D Coordinate and Distance Computation

After the intrinsic parameters are known, it selects one pixel in the depth image and uses its depth value to reconstruct the 3D coordinates and estimate the distance.

```
u = width // 2
v = height // 2

Z = depth_frame.get_distance(u, v)
print("==============================")
print("Pixel center: u =", u, "v =", v)
print("Depth Z =", Z, "meter")
print("==============================")

if Z == 0:
    print("No valid depth at center pixel.")
else:
    X = (u - cx) / fx * Z
    Y = (v - cy) / fy * Z
```

(u, v) denotes the pixel coordinates of the selected point, chosen at the image center. The function $get\_distance\,(u, v)$ returns the depth value Z for that pixel in meters. If $Z = 0$, it indicates that no valid depth is available at that pixel. When a valid depth is obtained, the 3D coordinates (X, Y, Z) in the camera coordinate system are computed using the projection formulas derived from the pinhole camera model. Specifically using the intrinsic parameters f_x , f_y , c_x , and c_y, the coordinates are given by,

$$X = \frac{(u - c_x)}{f_x} Z \ , Y = \frac{(v - c_y)}{f_y} Z$$

The resulting point is then represented as a vector

$$\mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Which is exactly the vector representation in euclidean space. The camera object distance predicted by the model is defined as the Euclidean norm of this vector,

```
d = math.sqrt(X*X + Y*Y + Z*Z)
```

```
    print("===== 3D Distance =====")
    print("X =", X)
    print("Y =", Y)
    print("3D distance d =", d, "meter")

pipeline.stop()
```

D. Error Metrics and Data Collection

For each experiment, the estimated distance (d_{est}) is obtained from the vector norm and it is compared with the measured distance (d_{true}). The absolute and relative errors are defined as,

$$e_{abs} = \left| d_{est} - d_{true} \right| \quad , \quad e_{rel} = \frac{e_{abs}}{d_{true}} \times 100\,\%$$

Where:

e_{abs} is the absolute error in meters, e_{rel} is the relative error in percent, d_{est} is the distance estimated from the 3D vector, and d_{true} is the physical distance measured with a ruler.

IV. Result and Experiment

A. Experimental Data

For the chosen depth resolution of 640 x 480, the Intel RealSense D455 depth camera reported intrinsic parameters with the value of f_x , f_y , c_x , and c_y which form the intrinsic matrix K. These values are obtained from the device using the Intel RealSense SDK, and correspond to the intrinsic matrix described in the pinhole camera model.

Intrinsic Parameter	f_x	f_y	c_x	c_y
value	391.788	391.788	327.168	241.053

Fig 1. Table of Intrinsic Parameter and the values

Using the procedure described in Section III, three different object distances were experimented, approximately around 0.45 m, 0.80 m, and 1.20 m from the camera. For each experiment, one valid depth pixel near the image center was selected and the corresponding depth value Z was recorded. The true distance (d_{true}), the physical distance from the front of the camera to the object, was measured using a ruler. The test samples are listed in Fig. 2.

Sample	$u\,(px)$	$v\,(px)$	$Z\,(m)$	$d_{true}(m)$
1.	320	240	0.454	0.45

2.	320	240	0.8	0.8
3.	320	240	1.202	1.2

Fig 2. Table of Test Samples

B. 3D Reconstruction and Distance Estimation

For each sample in Fig. 2, the 3D camera coordinates (X, Y, Z) were computed using the equations in Section II (from pixels and depth to 3D coordinate). The resulting point was written as a vector

$$\mathbf{p}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}$$

and the estimated camera object distance was obtained using the Euclidean norm,

$$d_{est,\,i} = \left\| p_i \right\|$$

The resulting coordinates and distances are summarized in Fig. 3.

Sample	$X\,(m)$	$Y\,(m)$	$Z\,(m)$	$d_{est}(m)$
1.	-0.0083	-0.00122	0.454	0.454
2.	-0.0146	-0.00215	0.800	0.800
3.	-0.02199	-0.00323	1.202	1.202

Fig 3. Table of Reconstructed 3D Coordinates and Estimated Distances for Each Sample

C. Error Metrics

The table below summarizes the error metrics for each sample, reporting the absolute error (e_{abs}) and the relative error (e_{rel})

Sample	1	2	3
e_{abs} $(\left\|d_{est} - d_{true}\right\|)$	0.004 m	0 m	0.002 m
e_{rel} $(\frac{e_{abs}}{d_{true}} \times 100\,\%)$	0.088%	0%	0.99%

Fig 4. Table of Reconstructed 3D Coordinates and Estimated Distances for Each Sample

C. Example Sample Computation

In this subsection, it is presented sample I as a worked example to illustrate how the pixel coordinate and depth measurement are converted into 3D camera frame coordinates and an estimated distance.

$f_x = 391.788$
$f_y = 391.788$
$c_x = 327.168$
$c_y = 241.053$
Pixel coordinates = $(u,\ v) = (320,\ 240)$
Depth $(Z) = 0.454\ m$
$d_{true} = 0.45\ m$

Compute X and Y coordinates obtained from the pixel and depth values:

- Compute X coordinate

  $X = \frac{(u - c_x)}{f_x} Z$

  $X = \frac{(320 - 327.168)}{391.788} (0.454)$

  $X = -0.00831\ m$

- Compute Y coordinate

  $Y = \frac{(v - c_y)}{f_y} Z$

  $Y = \frac{(240 - 241.053)}{391.788} (0.454)$

  $Y = -0.00122\ m$

Construct the 3D point camera coordinates,

$$\mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \approx \begin{bmatrix} -0.00831 \\ -0.00122 \\ 0.454 \end{bmatrix} m$$

Compute the Distance estimation using euclidean norm

$d_{est} = ||p|| = \sqrt{X^2 + Y^2 + Z^2}$

$d_{est} = \sqrt{(-0.00831)^2 + (-0.00122)^2 + (0.454)^2}$

$d_{est} = 0.454\ m$

Compute error over the measured distance. The true distance for this configuration is $d_{true} = 0.45$. The absolute and relative errors are,

$e_{abs} = \left|d_{est} - d_{true}\right| = |0.454 - 0.45| = 0.004\ m$

$e_{rel} = \frac{e_{abs}}{d_{true}} \times 100\,\% = \frac{0.004}{0.45} \times 100\% = 0.88\%$

D. Program Output and Experiment Documentation

1. Sample I (d_{true} = 0.45 m)

```
===== Intrinsics =====
- focal length:
  fx = 391.7889099121094
  fy = 391.7889099121094
- principle point:
  cx = 327.16815185546875
  cy = 241.0530548095703
==============================
Pixel center: u = 320 v = 240
Depth Z = 0.4540000259876251 m
==============================
===== 3D Distance =====
X = -0.00830636357061793
Y = -0.001220266574208457
3D distance (d) = 0.4540776456984463 meter
```

Fig 5. Program Output for Sample I

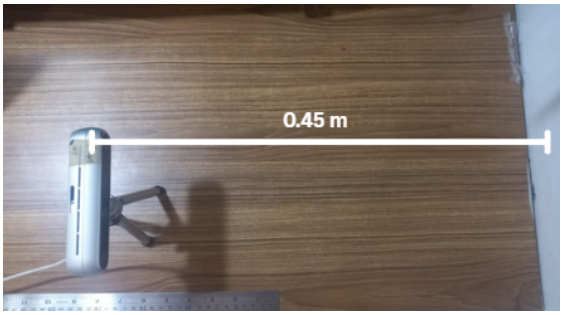
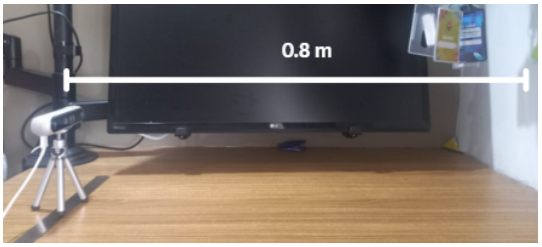


Fig 6. Picture of Distance Between Camera and Object (wall) 0.45 m

2. Sample II (d_{true} = 0.8 m)

```
===== Intrinsics =====
- focal length:
  fx = 391.7889099121094
  fy = 391.7889099121094
- principle point:
  cx = 327.16815185546875
  cy = 241.0530548095703
==============================
Pixel center: u = 320 v = 240
Depth Z = 0.800000011920929 m
==============================
===== 3D Distance =====
X = -0.014636763381363865
Y = -0.002150249378928639
3D distance (d) = 0.8001367867359681 meter
```

Fig 7. Program Output for Sample II



`Fig 8. Picture of Distance Between Camera and Object (wall) 0.8 m

3. Sample III (d_{true} = 1.2 m)

```
===== Intrinsics =====
- focal length:
  fx = 391.7889099121094
  fy = 391.7889099121094
- principle point:
  cx = 327.16815185546875
  cy = 241.0530548095703
==============================
Pixel center: u = 320 v = 240
Depth Z = 1.2020000219345093 m
==============================
===== 3D Distance =====
X = -0.021991737054109565
Y = -0.003230749702654189
3D distance (d) = 1.2022055260947935 meter
```

Fig 9. Program Output for Sample III

`Fig 10. Picture of Distance Between Camera and Object (door) 1.2 m

E. Analysis Over All Samples

From the experiment:

- Because the pixel (u_i, v_i) = (320, 240) is chosen near the principal point (c_x, c_y) = (327.168, 241.053) the pixel offset from the image center is small. As a result, the reconstructed coordinates X_i and Y_i are much smaller than Z_i. This indicates that the tested point lies close to the optical axis in camera coordinates.
- In our three samples, the absolute error stays in the millimetre range (0 – 0.004 m), while the relative error stays below 1% (0 – 0.88%). This indicates that, for the tested range and setup, the estimated distance (d_{est}) closely matches the measured distance (d_{true}).

- For all samples, the estimated distance (d_{est}) equals the recorded depth value Z to three places (0.454, 0.8, and 1.202), which is consistent with X and Y being much smaller than Z for the chosen pixel.

Overall, within the tested range (0.45 – 1.20 m), the estimated distance closely matches the measured distance, with a maximum absolute error of 0.004 m and a maximum relative error of 0.88% in our three samples.

V. Conclusion

In this study, the writer built a simple model to estimate the distance between a camera and an object. The Intel RealSense D455 depth camera provides pixel coordinates and depth values, while the mathematical model represents each observed point as a 3D vector and uses the intrinsic matrix of the camera to map between image space and camera space. By taking the Euclidean norm of this vector, the model turns raw depth data into a clear numerical distance.

Even though the experiment used only a few distances and a single pixel per frame, it still captures the most important ideas, such as, how a 3D point is represented as a vector, how the camera matrix links geometry and image, and how close the model's distance is to the physical measurement. The focus is not on building a full 3D reconstruction system, but on showing clearly how the concepts from a linear algebra course appear inside a real depth camera.

The key contributions of this research are:

- A model that connects pixel coordinates and depth values to 3D coordinates in the camera frame using the intrinsic matrix.
- A Python implementation with the Intel RealSense D455 that demonstrates how to read intrinsic parameters and compute distances using Euclidean norms.
- An experimental comparison between model based distances and tape-measure distances, showing that the relative error stays within a few percent in the tested range.

This model can help students see that abstract topics such as vectors, norms, and linear transformations are not only theory but also tools for understanding real sensors. Although the experiments were limited to a few static setups near the optical axis, the same framework can be extended to more pixels, wider distance ranges, and more complex scenes. In that way, this work provides a small but concrete bridge between linear algebra and modern computer vision hardware.

VI. Appendix

Link github:
https://github.com/Faizalfadaa/Realsense-Algeo.git

Link Youtube:
https://youtu.be/PYcjgvfv4dw

VII. Acknowledgment

I would like to express my sincere gratitude to Allah for His blessings and guidance. I also thank my friends, my parents, and my lecturer for their continuous support, encouragement, and help throughout this work.

References

[1] Munir, R., 2025. "Vektor di Ruang Euclidean (Bagian 1)". Lecture Notes, IF2123 Aljabar Linier dan Geometri, Institut Teknologi Bandung, 2025. [Online]. Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2025.pdf [Accessed: 23 December 2025]

[2] Munir, R., 2025. "Ruang Vektor Umum dan Transformasi Linier (Bagian 4)". Lecture Notes, IF2123 Aljabar Linier dan Geometri, Institut Teknologi Bandung, 2025. [Online]. Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-18-Ruang-vektor-umum-Bagian4-2025.pdf [Accessed: 23 December 2025]

[3] Munir, R., 2025. "Sistem Persamaan Linier (Bagian 1)". Lecture Notes, IF2123 Aljabar Linier dan Geometri, Institut Teknologi Bandung, 2025. [Online]. Available at: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf [Accessed: 23 December 2025]

[4] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge Univ. Press, 2004.

[5] Intel Corporation, *Intel RealSense Depth Camera D455 – Product Specifications and Datasheet*, Intel RealSense D400 Series Documentation.

[6] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed. Springer, 2022.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

Muhammad Faiz Alfada Dharma 13524097