Dense 3D Reconstruction Using Stereo Image Rectification and Depth Estimation

Mobile Robotics - Mid-Term Progress Report

Team Members:
Aarav Singla
Nilavra Ghosh

November 20, 2025

Abstract

This project aims to develop a computational pipeline for dense 3D reconstruction using a stereo camera setup. The core objective is to extract depth information from a pair of 2D images by simulating human binocular vision. Up to this stage, we have successfully constructed a stereo rig using smartphones, calibrated the camera intrinsics and extrinsics using a chessboard pattern, implemented stereo image rectification to align epipolar lines, and generated initial dense disparity maps using Semi-Global Block Matching (SGBM). This report details the methodology, implementation challenges, and results achieved thus far.

1 Introduction

Depth perception is a fundamental requirement for mobile robotics, enabling tasks such as obstacle avoidance, mapping (SLAM), and path planning. While LiDAR and RGB-D sensors provide direct depth measurements, they are often expensive or power-intensive. Passive stereo vision offers a cost-effective alternative by inferring depth from two standard RGB cameras.

The goal of this project is to reconstruct a 3D scene by calculating the disparity—the horizontal shift of pixels—between a left and right image pair. The magnitude of this shift is inversely proportional to the depth of the object.

2 Methodology

The project pipeline consists of four sequential stages:

2.1 Stereo Rig Setup Data Acquisition

We constructed a custom stereo rig using two smartphones mounted in a parallel configuration. To ensure accurate calibration, we utilized the "Video + Audio Sync" method. Both phones recorded video simultaneously with focus and exposure locked manually. A sharp audio cue (clap) was used to synchronize the video streams and extract corresponding stereo frame pairs.

2.2 Camera Calibration

Calibration is critical to determine the internal characteristics of the cameras (Intrinsics K) and their spatial relationship (Extrinsics R, T). We utilized a checkerboard pattern displayed on a laptop screen. To resolve edge-detection issues common with screens, we engineered a specific pattern with a white margin ("quiet zone") to ensure robust corner detection by the algorithm.

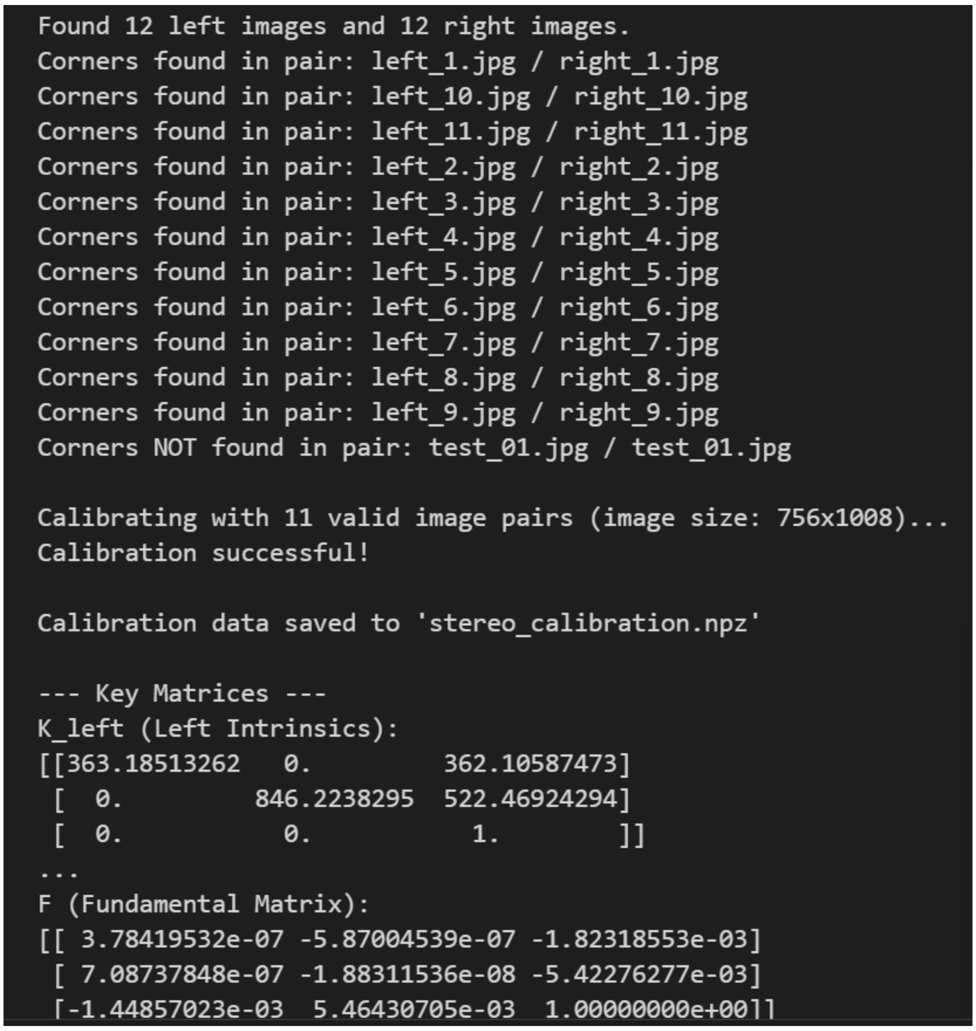
We used OpenCV's `cv2.stereoCalibrate` to minimize the reprojection error, solving for:

- **Intrinsic Matrix (K):** Focal lengths (f_x, f_y) and optical centers (c_x, c_y).
- **Distortion Coefficients (D):** To correct radial and tangential lens distortion.
- **Extrinsics:** The Rotation matrix (R) and Translation vector (T) (Baseline) between the two sensors.

3 Implementation & Results

3.1 Calibration Results

The calibration process was executed using 11 synchronized stereo pairs containing a 7×7 internal corner checkerboard pattern. By utilizing adaptive thresholding flags within `cv2.findChessboardCorners`, we successfully detected corners even in challenging lighting conditions. The stereo calibration routine minimized the Root Mean Square (RMS) reprojection error, yielding a consistent set of intrinsic matrices and the translation vector necessary for the subsequent steps. Figure 1 demonstrates the successful detection of internal corners on the modified digital pattern.



```
Found 12 left images and 12 right images.
Corners found in pair: left_1.jpg / right_1.jpg
Corners found in pair: left_10.jpg / right_10.jpg
Corners found in pair: left_11.jpg / right_11.jpg
Corners found in pair: left_2.jpg / right_2.jpg
Corners found in pair: left_3.jpg / right_3.jpg
Corners found in pair: left_4.jpg / right_4.jpg
Corners found in pair: left_5.jpg / right_5.jpg
Corners found in pair: left_6.jpg / right_6.jpg
Corners found in pair: left_7.jpg / right_7.jpg
Corners found in pair: left_8.jpg / right_8.jpg
Corners found in pair: left_9.jpg / right_9.jpg
Corners NOT found in pair: test_01.jpg / test_01.jpg

Calibrating with 11 valid image pairs (image size: 756x1008)...
Calibration successful!

Calibration data saved to 'stereo_calibration.npz'

--- Key Matrices ---
K_left (Left Intrinsics):
[[363.18513262   0.         362.10587473]
 [  0.         846.2238295  522.46924294]
 [  0.           0.           1.        ]]
...
F (Fundamental Matrix):
[[ 3.78419532e-07 -5.87004539e-07 -1.82318553e-03]
 [ 7.08737848e-07 -1.88311536e-08 -5.42276277e-03]
 [-1.44857023e-03  5.46430705e-03  1.00000000e+00]]
```

Figure 1: Successful corner detection during calibration.

3.2 Rectification Verification

Using the computed camera matrices, we applied the `cv2.stereoRectify` function to compute the rectification transforms. This step warped the input images to compensate for radial distortion and alignment offsets. As illustrated in Figure 2, we verified the results by overlaying horizontal epipolar lines. Features such as the edges of the red bottle and the geometric shapes in the background align perfectly along the scanlines (Y-axis), confirming that the search space for correspondence matching has been successfully reduced from 2D to 1D.

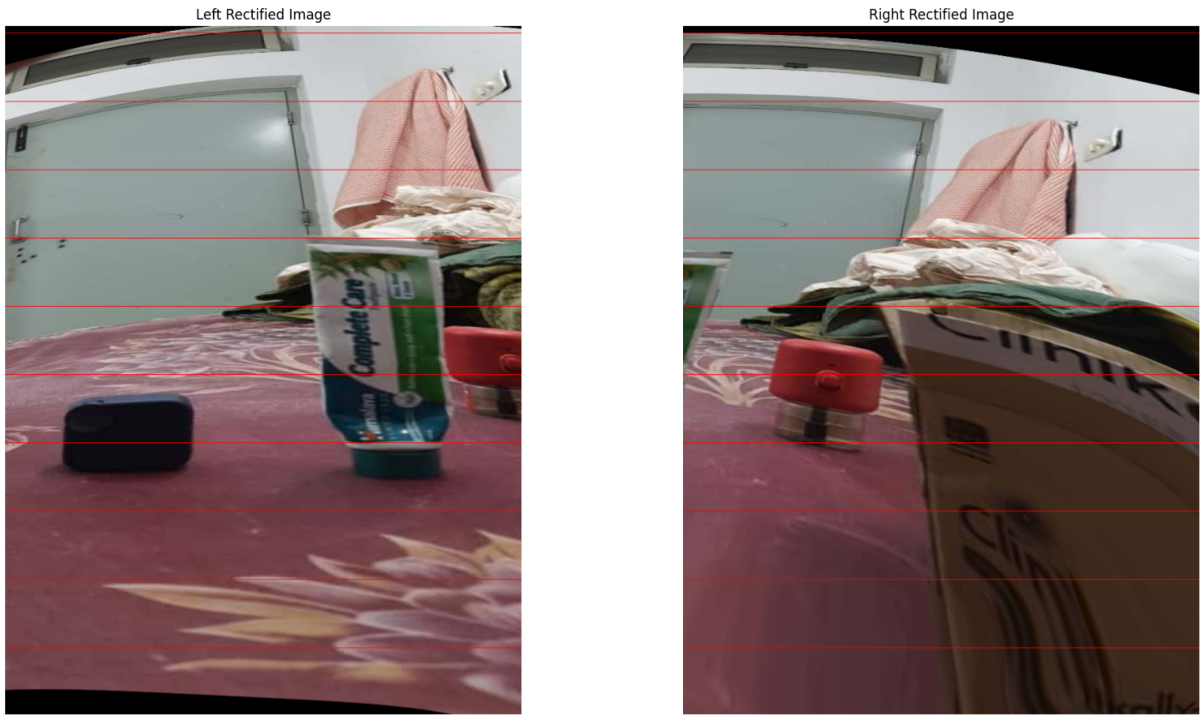


Figure 2: Rectified stereo pair with horizontal epipolar lines verifying alignment.

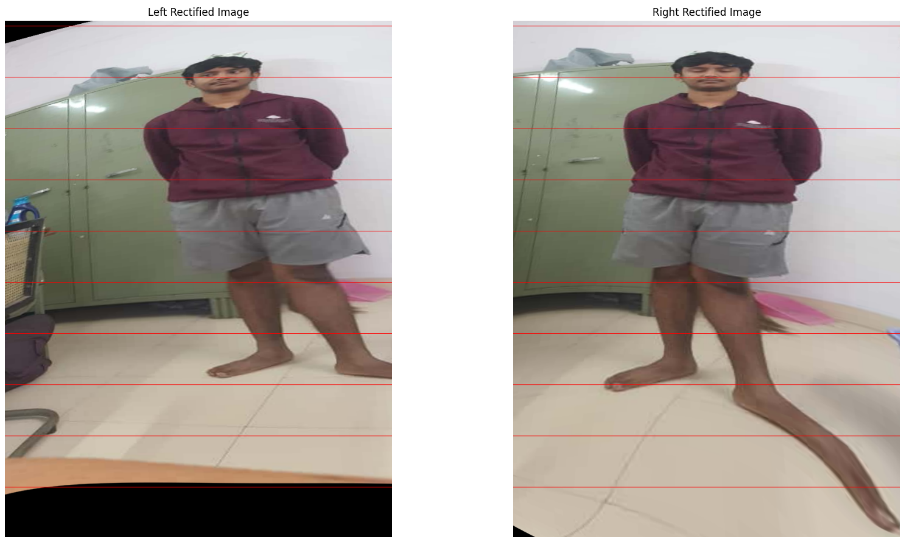


Figure 3: Rectified stereo pair with horizontal epipolar lines verifying alignment.

3.3 Disparity Map Generation

For dense correspondence matching, we employed the Semi-Global Block Matching (SGBM) algorithm. Unlike standard block matching, SGBM optimizes a global energy function, making it more robust to textureless regions. We performed iterative parameter tuning to improve map quality:

- **Block Size:** Set to 5 to preserve fine details while reducing noise.
- **Disparity Range:** Configured to capture both near-field and far-field objects (numDisparities = 64).

- **Smoothing:** Penalties $P1$ and $P2$ were adjusted to enforce smoothness on flat surfaces while respecting depth discontinuities at object edges.

The resulting disparity map (Figure 4) clearly distinguishes the foreground objects (warm colors) from the background (cool colors).

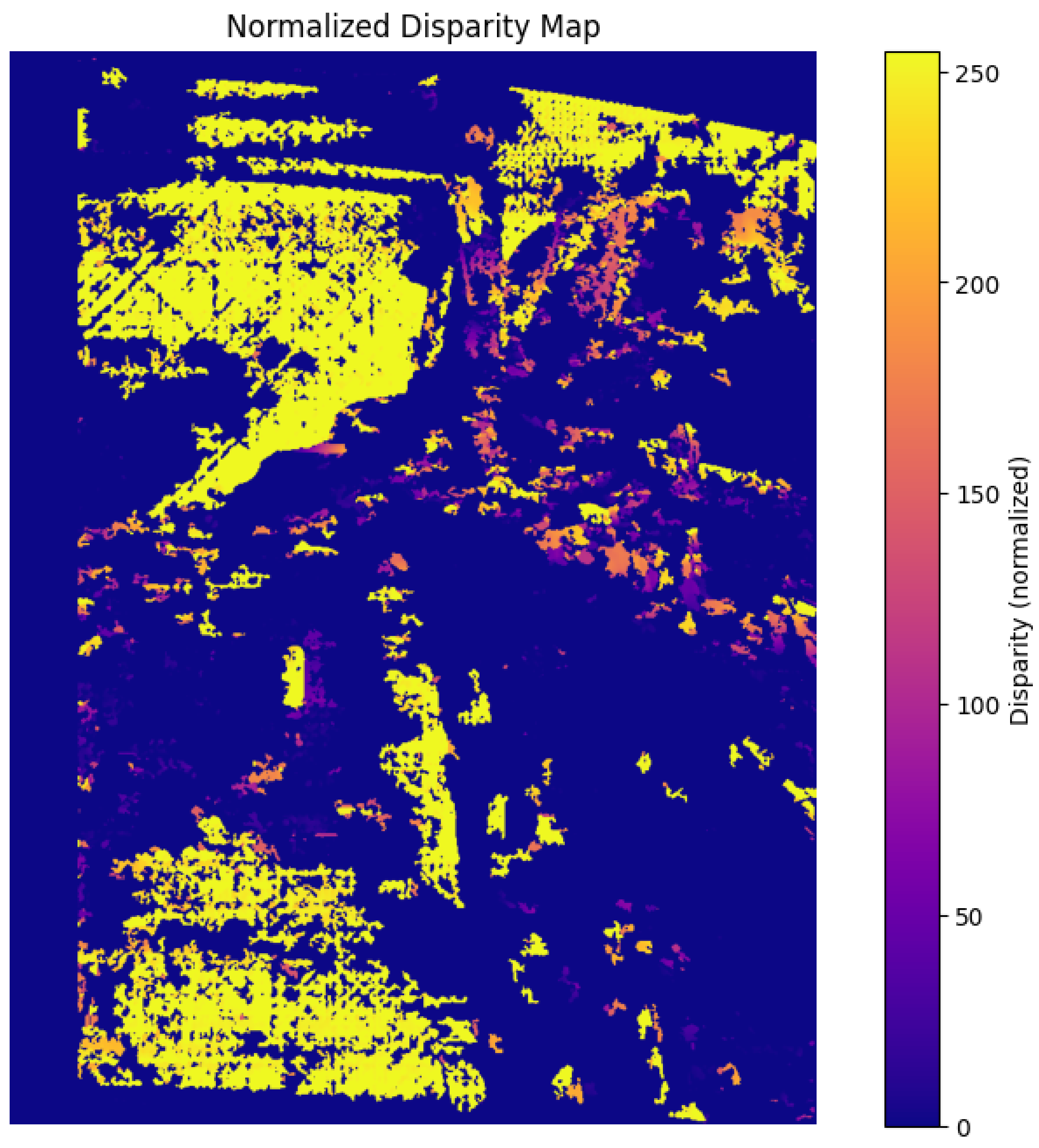


Figure 4: Generated Dense Disparity Map using SGBM.

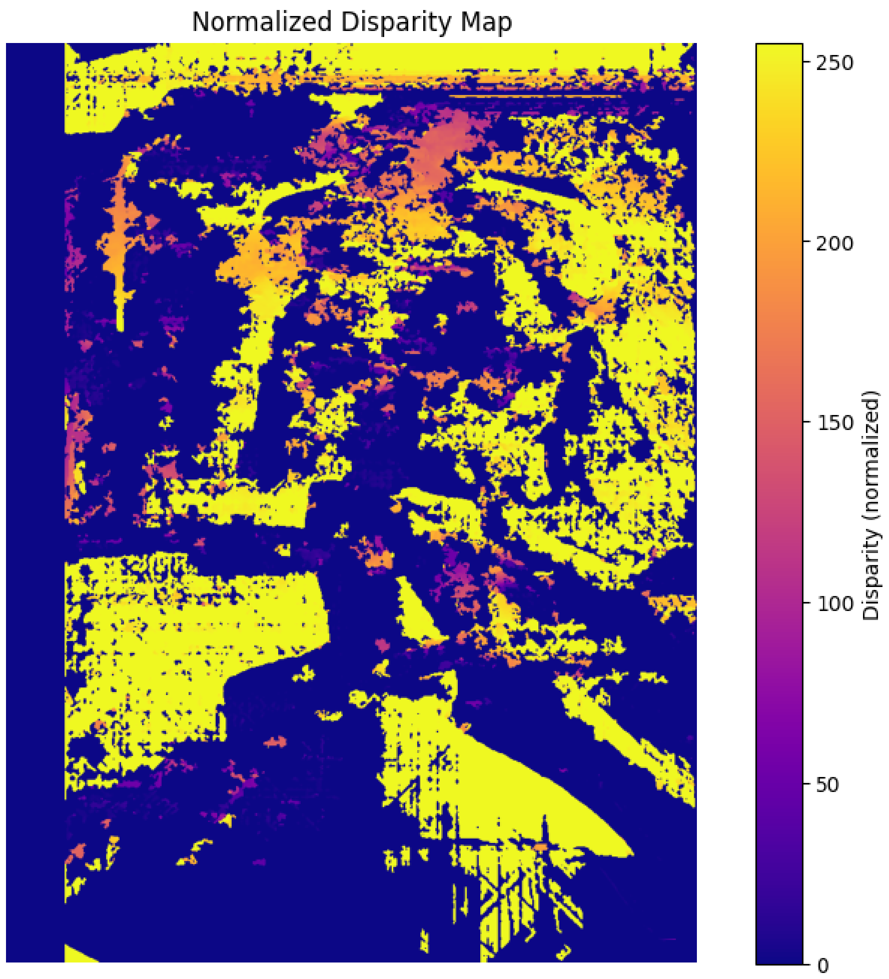


Figure 5: Generated Dense Disparity Map using SGBM.

3.4 Depth Estimation

Finally, the disparity map was converted into metric depth using the Reprojection Matrix (Q). Figure 6 shows the normalized depth map. The intensity values correspond to physical distance, with darker regions representing closer objects and lighter regions representing the background. This map serves as the foundation for the 3D point cloud generation in the next phase.

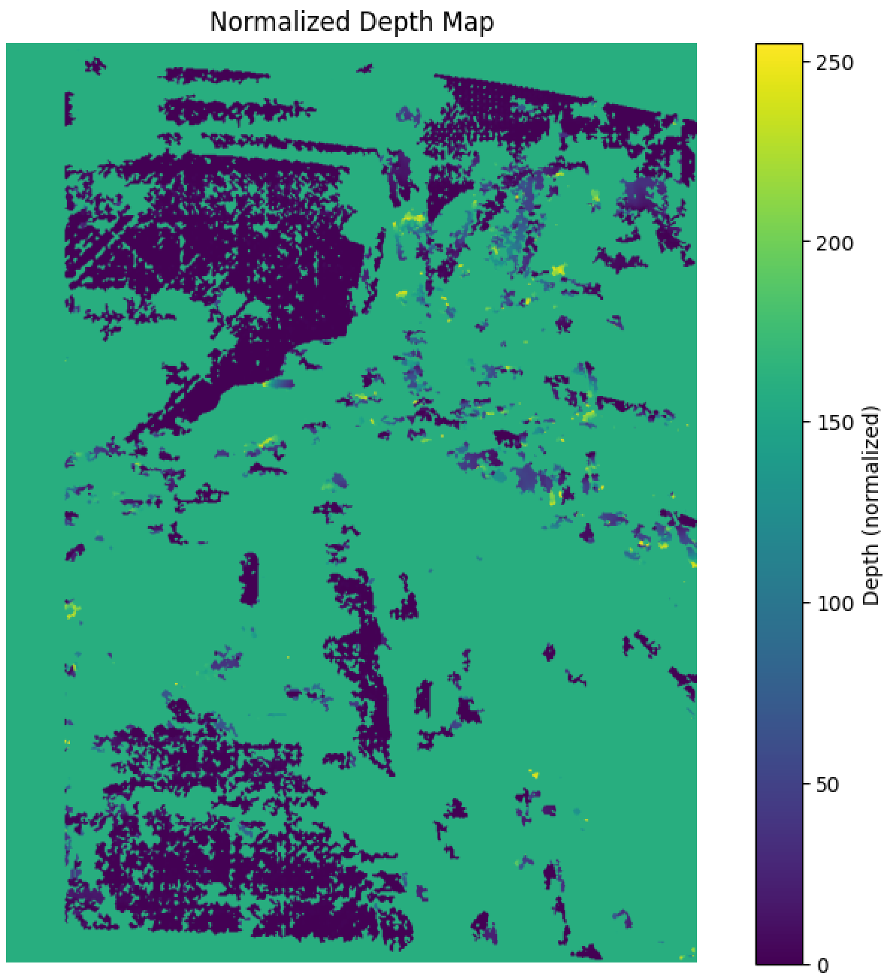


Figure 6: Normalized Depth Map generated from disparity data.

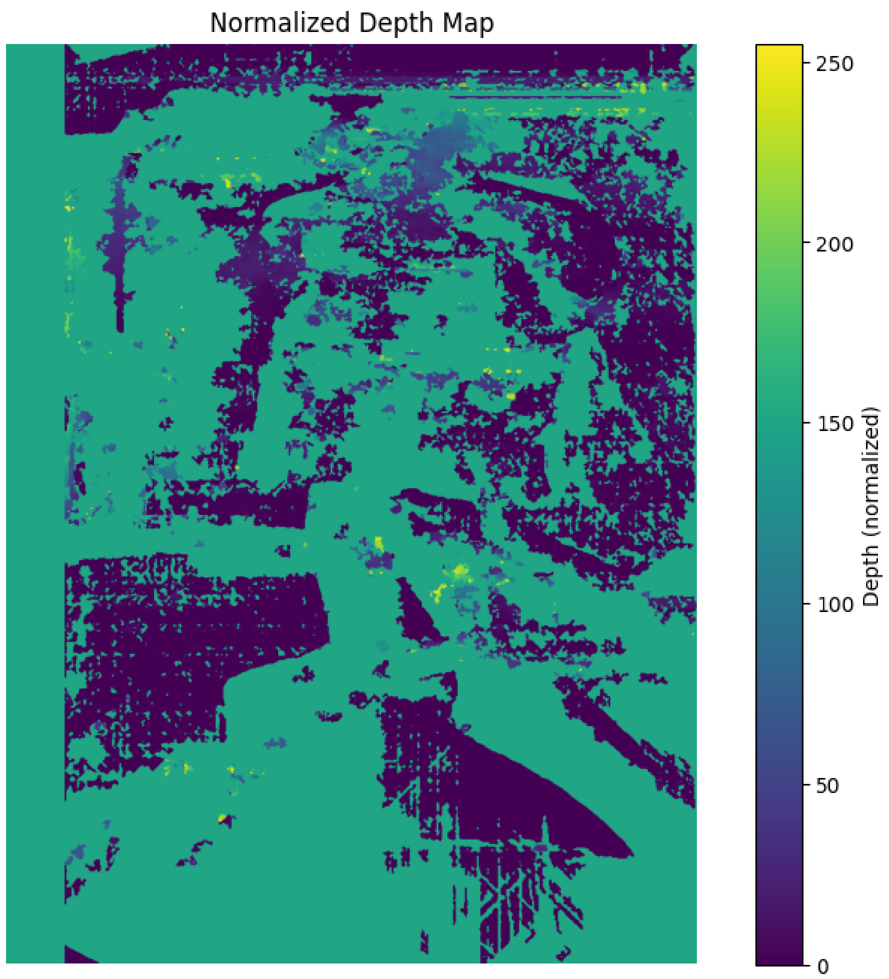


Figure 7: Normalized Depth Map generated from disparity data.

4 Challenges Faced

1. **Screen Border Detection:** Initially, the calibration failed because the checkerboard pattern touched the edges of the laptop screen. We resolved this by modifying the pattern to include a calculated white margin, allowing OpenCV to detect the external corners.

2. **Synchronization:** Manually taking photos resulted in temporal mismatch. We shifted to a video-extraction method using audio spikes for millisecond-level synchronization.

3. **Focus Breathing:** Auto-focus caused the intrinsic parameters to change between shots. We implemented a protocol to lock the manual focus on both devices prior to data acquisition.

5 Future Work

The immediate next steps for the final submission include:

- **Point Cloud Refinement:** Filtering ”flying pixels” and noise from the 3D point cloud using statistical outlier removal.
- **Metric Evaluation:** Measuring real-world distances of objects in the scene and comparing them to the calculated depth Z to quantify error.
- **Visualization:** implementing a fully interactive 3D visualization using Open3D.