3D Reconstruction using Stereo Vision: SAD Disparity and Classical Stereo Triangulation

Dhanush Srinivas (1002232331)
Department of Computer Science
University of Texas at Arlington
Course: CSE 6367 - Computer Vision
Email: dxs2331@mavs.uta.edu
GitHub: CSE6367 final project-3D-Reconstruction-using-Stereo-Vision

***Abstract*—Stereo vision allows machines to infer depth from two images by mimicking human stereopsis. In this paper, we present a 3D reconstruction pipeline using the 2014 Middlebury Stereo Dataset[1]. Using Sum of Absolute Differences (SAD) for region matching, followed by post-processing steps, we construct a depth map and reproject 2D image points into 3D space. Calibration parameters from the dataset are incorporated for spatial accuracy. The final output is visualized as a colored point cloud using Open3D and Plotly.**

I. Introduction

Estimating depth from 2D images is a critical task in computer vision with applications in robotics, 3D modeling, and autonomous systems. While state-of-the-art (SOTA) models like RAFT-Stereo and PSMNet use deep learning and massive datasets[2], these approaches often demand substantial computing power, extensive training time, and large GPU memory.

In contrast, stereo vision methods based on classical geometry and simple matching strategies remain highly practical, especially in resource-constrained environments. These methods are also easier to interpret and debug. Another popular but expensive alternative—LiDAR—offers highly accurate depth but at a steep hardware and operational cost.

We adopt a purely classical stereo pipeline using Sum of Absolute Differences (SAD)[3], and apply it to high-resolution stereo images from the 2014 Middlebury Dataset, which is widely recognized as a benchmark in stereo vision research.

The dataset includes calibrated image pairs along with camera parameters. This makes it suitable for precise depth reconstruction using triangulation principles. Our goal is to produce accurate and visually meaningful 3D reconstructions using only efficient and transparent algorithms.

II. Theory

The foundation of computer stereo vision is rooted in *stereopsis*, the biological mechanism humans use to perceive depth. Our eyes capture slightly different views of the same object due to their horizontal separation. This difference, called **parallax**, forms the basis for computing depth[4].

In stereo vision, two rectified 2D images are captured using cameras separated by a known distance b and with known focal length f. A point $P(x, y, z)$ in space appears at pixel coordinates x_l in the left image and x_r in the right. The disparity d is defined as:

$$d = x_l - x_r$$

Using the geometry of triangulation, the depth z can be calculated as:

$$z = \frac{f \cdot b}{x_l - x_r}$$

The disparity for each pixel is computed via region matching: a square window is slid across the same row in both images to find the best match. By assuming both cameras are horizontally aligned, search is limited to horizontal scanlines. This results in a disparity matrix that allows us to reconstruct the full 3D scene[5].

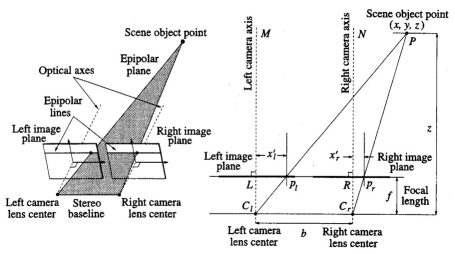

Fig. 1: Stereo vision model with two cameras separated by baseline b and focal length f.

III. Methodology

The stereo vision pipeline comprises the following stages:

- **Data Acquisition:** Use of high-resolution rectified stereo pairs (2864×1924) from Middlebury 2014.
- **Image Preprocessing:** Conversion to grayscale and downscaling to reduce computation time.
- **Disparity Estimation:** Region matching using SAD, computed by sliding a fixed window across horizontal scanlines.
- **Post-Processing:** Includes disparity filtering using mode replacement, smoothing via averaging, and clipping outliers.

- **Depth Calculation:** Triangulation using focal length, disparity offset, and baseline values from the provided calibration.
- **Point Cloud Generation:** Use of OpenCV's reproject function, followed by Open3D-based export and rendering.

IV. Implementation

A. Dataset and Preprocessing

We used the "Recycling" image pair from the 2014 Middlebury stereo dataset. Each image was converted to grayscale using the Pillow library and resized to 358×240 pixels[6].

(a) Left image (b) Right image

Fig. 2: Stereo pair from Middlebury Dataset

B. Disparity Estimation

A sliding window approach was used with the Sum of Absolute Differences (SAD)[5]:

$$\text{SAD}(x, y, d) = \sum_{i,j \in W} |I_L(x+i, y+j) - I_R(x+i-d, y+j)|$$

The disparity d yielding the minimum SAD was chosen. The result is a raw disparity map (Fig. 3).

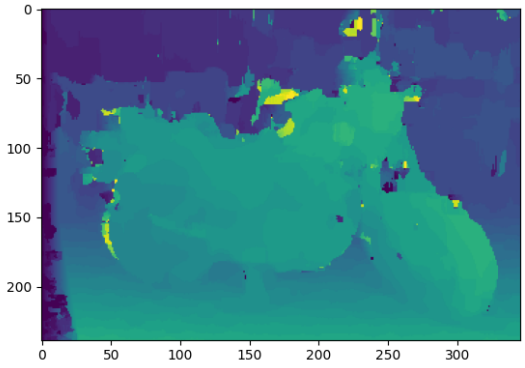

Fig. 3: Raw disparity map

C. Post-Processing

- **Mode filtering:** A 25×25 window is used to replace noisy disparity values.
- **Average smoothing:** If a center pixel deviates more than 5 units from the window mean, it is replaced.
- **Clipping:** All values above 60 are removed to reduce spikes in flat regions.

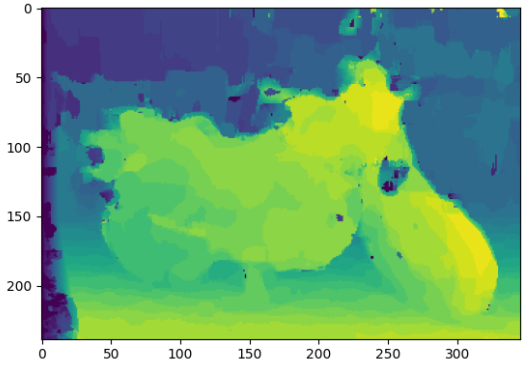

Fig. 4: Post-processed disparity map

D. Depth Computation

Calibration from `calib.txt` provided:

- $f = 2945.377$ (focal length)
- $b = 178.232$ mm (baseline)
- $doffs = 170.681$ (offset)

Depth is calculated using[7]:

$$z = \frac{f \cdot b}{d + doffs}$$

E. 3D Point Cloud Generation

3D points were generated using OpenCV's `reprojectImageTo3D` with the computed disparity and calibration matrix Q[8]. Open3D was used to export a colored '.ply' file. Plotly was used to view the point cloud interactively in the browser.

V. Results

The final 3D reconstruction preserved the spatial geometry of the recycling bin and background. Post-processing significantly improved continuity and reduced noise in flat areas.

We generated a '.ply' file containing approximately **22,000 points**, which can be opened in tools like Blender, MeshLab, or CloudCompare for further rendering or mesh processing. The final point cloud retained color and position fidelity based on the original image.

- **Original image size:** 2864×1924
- **Resized input:** 358×240 (1/8 scale)
- **SAD window size:** 11×11
- **Disparity range:** 0 to 32 (rescaled from Middlebury's 260)

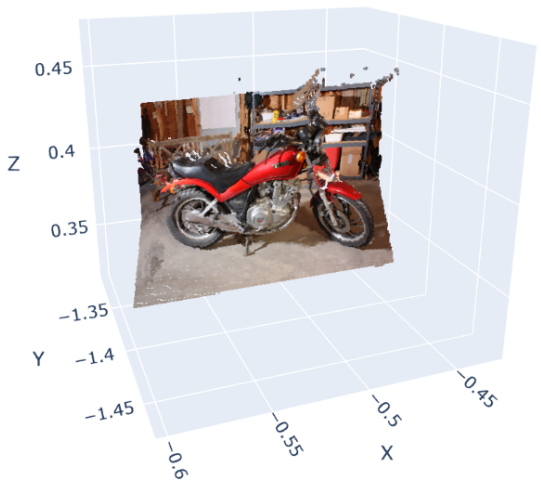

Fig. 5: 3D point cloud reconstruction of the motorcycle

- **Runtime:** ∼28 seconds on RTX 4060 (including post-processing)
- **Output .ply file size:** 2.3 MB

Compared to deep learning models, this method required no training and ran faster on modest hardware.

VI. Conclusion

We demonstrated that classical stereo vision methods can still yield robust 3D reconstruction with the help of good calibration and refinement steps. Unlike LiDAR, this method is inexpensive, requires only cameras, and produces dense depth maps.

Use Cases for Low-Resource Stereo

- **Field robotics**: Environments where LiDAR is cost-prohibitive.
- **Embedded vision**: Drones or edge devices with limited memory.
- **Education & prototyping**: Quick 3D scanning using webcams.

Environment and Tools

- **Python Libraries:** NumPy (1.24), OpenCV (4.8), Pillow (9.5), Open3D (0.18), Plotly (5.14)
- **IDE:** Visual Studio Code
- **OS:** Windows 11
- **System:** AMD Ryzen 7, 16GB RAM
- **GPU:** NVIDIA RTX 4060

References

[1] G. A. Kordelas, D. S. Alexiadis, P. Daras, and E. Izquierdo, "Enhanced disparity estimation in stereo images," *Image and Vision Computing*, vol. 35, pp. 31–49, 2015.

[2] L. Lipson, Z. Teed, and J. Deng, "Raft-stereo: Multilevel recurrent field transforms for stereo matching," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 218–227.

[3] Z. Yang, Y. Liang, D. Lin, J. Li, Z. Chen, and X. Li, "Real-time stereo vision hardware accelerator: Fusion of sad and adaptive census algorithm," *IEEE Access*, 2024.

[4] I. P. Howard, *Perceiving in depth, volume 1: basic mechanisms*. Oxford University Press, 2012.

[5] K. Mühlmann, D. Maier, J. Hesser, and R. Männer, "Calculating dense disparity maps from color stereo images, an efficient implementation," *International Journal of Computer Vision*, vol. 47, pp. 79–88, 2002.

[6] P. Madhav, T. Sreeram, B. S. B. S. Pavansai, M. Vivek, and P. Vidyullatha, "A real-time image recognition using tensorflow framework," in *Doctoral Symposium on Computational Intelligence*. Springer, 2023, pp. 389–400.

[7] M. Kytö, M. Nuutinen, and P. Oittinen, "Method for measuring stereo camera depth accuracy based on stereoscopic vision," in *Three-dimensional imaging, interaction, and measurement*, vol. 7864. SPIE, 2011, pp. 168–176.

[8] J. Howse, S. Puttemans, Q. Hua, and U. Sinha, *OpenCV 3 Blueprints*. PACKT Publishing Ltd, 2015.