

3d Object Reconstruction From 2d Images.

Seema Firdos, Poornima S, Prajwal S R, Ranjitha, Vineeth M

Asst Professor, Department of Computer Science & Engineering, Mysuru Royal Institute of Technology, Mandya, Karnataka, India
Student, Department of Computer Science & Engineering, Mysuru Royal Institute of Technology, Mandya, Karnataka, India.

Abstract— Three-dimensional object reconstruction from ordinary two-dimensional images involves estimating the structure and appearance of an object from several overlapping views. This project implements a software-based reconstruction pipeline that combines COLMAP, OpenMVS, OpenCV, and a Flask web application. Multiple images of the same object are first uploaded, validated, and organized. COLMAP extracts visual features, matches corresponding image regions, estimates camera parameters and camera poses through Structure from Motion (SfM), and generates a sparse 3D point cloud. The resulting reconstruction is converted into an OpenMVS scene for dense Multi-View Stereo (MVS). OpenMVS estimates dense depth information, generates and fuses dense points, reconstructs a surface mesh, and applies texture information. The generated PLY or OBJ model can be inspected using MeshLab or accessed through the web-based result interface. The project demonstrates a practical workflow for producing a viewable 3D model from ordinary multi-view images without relying on dedicated 3D capture hardware. The supplied project material does not include numerical accuracy values, benchmark datasets, execution times, or quantitative comparisons; therefore, the discussion is limited to the implemented workflow and observable reconstruction outputs

Index Terms— 3D reconstruction, computer vision, Structure from Motion, COLMAP, OpenMVS, Multi-View Stereo, Flask, sparse point cloud, dense point cloud, mesh reconstruction, texture generation.

I. INTRODUCTION

Three-dimensional reconstruction from two-dimensional images has become an important area of computer vision because it enables the creation of digital models from ordinary image collections. Unlike specialized 3D scanning systems, image-based reconstruction uses visual information captured from multiple viewpoints. By analyzing the relationships between these views, the system estimates the position of the cameras and recovers the visible geometric structure of the object.

A single image is generally insufficient for recovering complete three-dimensional information because depth cannot be directly observed from a two-dimensional projection. Multiple overlapping images provide different observations of the same object, allowing common visual features to be identified and compared. These correspondences are used to estimate camera geometry and calculate the spatial arrangement of reconstructed points.

The reconstruction process is generally divided into sparse and dense stages. The sparse stage establishes the camera poses and produces an initial set of 3D points. In this project, COLMAP performs feature extraction, feature matching, camera estimation, and Structure from Motion to generate this initial reconstruction. The sparse result provides the geometric foundation for the subsequent dense stage.

The dense stage improves the representation by estimating more detailed information from the available views. OpenMVS performs Multi-View Stereo, generates dense points, fuses the reconstruction, creates a surface mesh, and

applies texture information. The final output is therefore not limited to a collection of points; it can be represented as a textured 3D model suitable for visualization.

The web application provides facilities for uploading images, controlling the reconstruction process, displaying processing or error information, and accessing the completed model.

II. LITERATURE SURVEY

3D reconstruction from multiple images is a long-standing computer vision problem that aims to recover the geometry of a scene or object from visual observations. The survey paper organizes the field into four

major directions: binocular stereo reconstruction, multi-view stereo

(MVS), object-focused reconstruction, and SLAM-based reconstruction

Binocular stereo methods use two calibrated, rectified images and estimate disparity by finding pixel correspondences between the left and right views. The main challenge is accurate stereo matching under occlusion, lighting variation, and repetitive texture, making this a foundational but difficult reconstruction setting.

Multi-view stereo extends this idea to more than two images captured from known viewpoints. Its goal is to build dense 3D representations such as depth maps, voxel grids, or meshes, giving better scene coverage and geometric completeness than stereo pairs alone.

Object-focused 3D reconstruction targets individual objects rather than full scenes. This branch is useful in practical applications such as robotics, augmented reality, and industrial inspection, where accurate shape recovery of a specific object is more important than reconstructing an entire environment.

SLAM-based 3D reconstruction integrates simultaneous localization and mapping with scene reconstruction. It is especially useful in real-time and mobile settings because it estimates both camera motion and scene structure, making it suitable for robotics, navigation, and autonomous systems.

The survey highlights the growing role of deep learning in improving feature matching, depth estimation, and reconstruction quality. Compared with traditional geometry-based approaches, learning-based methods are more robust to noise and can better handle complex scenes, but they still face challenges such as computational cost, generalization, and dependence on training data.

III. OBJECTIVES

- To develop a secure online voting platform.
- To collect multiple images of the same object from different viewpoints.
- To validate and organize the uploaded images for reconstruction.

- To extract visual features using COLMAP.
- To match corresponding features across the input images.
- To estimate camera poses using Structure from Motion (SfM).
- To generate a sparse 3D point cloud.
- To convert the COLMAP reconstruction into OpenMVS format.
- To generate and fuse a dense 3D point cloud using Multi-View Stereo (MVS).
- To reconstruct a surface mesh and apply texture information.
- To provide and visualize the final PLY/OBJ model through the Flask web application or MeshLab

IV. PROBLEM STATEMENT

Ordinary 2D images do not contain direct depth measurements. The system must therefore infer three-dimensional structure by identifying corresponding visual information across multiple views and estimating the camera geometry associated with those views. This makes the reconstruction process dependent on suitable overlapping images and successful feature matching.

Some existing reconstruction arrangements may depend on specialized hardware, multiple dedicated cameras, or complex capture systems. The problem addressed by this project is to provide a practical software-based workflow that accepts multiple images of an object, processes them through established reconstruction stages, and produces a viewable 3D model using affordable and ordinary image-capture resources.

V. METHODOLOGY

The system begins with multiple overlapping 2D images representing one object. These images are first validated and organized as reconstruction input. The preprocessing stage prepares the image collection for the subsequent computer-vision operations. OpenCV and standard image-handling operations support the validation and preprocessing activities described in the project material.

COLMAP then processes the organized images. It extracts visual features and matches them between images. Based on these correspondences, COLMAP performs camera calibration or estimation and applies Structure from Motion to determine the camera poses and reconstruct sparse 3D points. Image undistortion is also included in the COLMAP module before the reconstruction is passed to the dense-reconstruction stage.

The COLMAP scene is converted into OpenMVS format using the COLMAP-to-OpenMVS interface. OpenMVS performs dense Multi-View Stereo, estimates dense depth information, generates dense points, and fuses those points into a denser representation. The dense result is then used to reconstruct a surface mesh, after which texture information is applied to create the final textured model.

The Flask application provides the user-facing layer. It accepts the input images, supports reconstruction control, displays processing status and errors, and provides a result page through which the final reconstruction can be accessed. The output may also be opened in MeshLab for inspection of the dense point cloud, mesh, and related reconstruction layers.

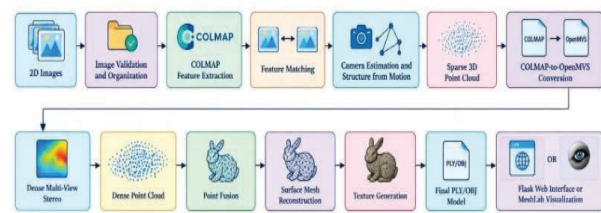


Figure 1: Flowchart of the proposed 3D object reconstruction system.

The flowchart illustrates the complete 3D reconstruction process, from 2D image input and COLMAP processing to dense reconstruction, mesh generation, texturing, and final 3D model visualization.

VI. MODELING AND ANALYSIS

The project is modeled as a software-based pipeline that converts multiple overlapping 2D images into a viewable 3D object model. The images are first validated and organized, after which COLMAP performs feature extraction, feature matching, camera estimation, and Structure from Motion. This produces estimated camera poses and a sparse 3D point cloud that represents the initial scene structure.

The sparse COLMAP reconstruction is then converted into an OpenMVS-compatible scene. OpenMVS applies Multi-View Stereo to estimate dense information from the available image views. The dense points are generated and fused to form a more detailed representation of the object.

The fused dense point cloud is used for surface mesh reconstruction. OpenMVS creates the mesh and applies texture information to produce the final textured 3D model. The project supports PLY and OBJ outputs, which can be visualized using MeshLab or a web-based viewer.

The Flask application acts as the interface layer of the system. It supports image upload, reconstruction control, processing-status information, error/status display, and access to the final result. This connects the user-facing web interface with the COLMAP and OpenMVS reconstruction stages.

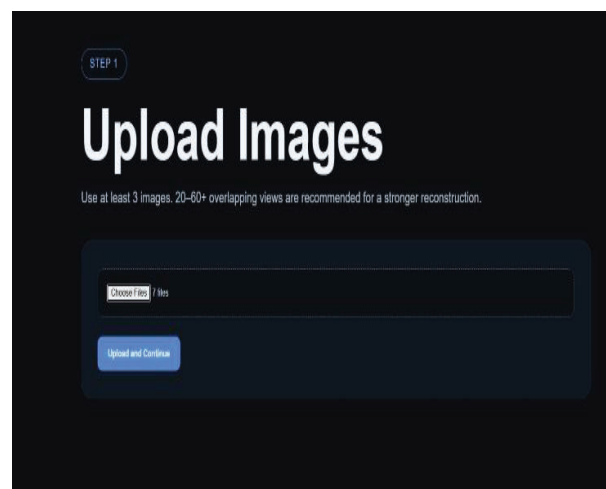


Figure 2: Image Upload Interface for 3D Reconstruction

Table 1: System Modules and Their Functions

Module	Function
Image Upload Module	Accepts and organizes multiple images of the same object from different viewpoints.
Image Validation & Preprocessing	Checks the uploaded images and prepares them for reconstruction..
COLMAP Module	Performs feature extraction, feature matching, camera estimation, and Structure from Motion (SfM).
Sparse Reconstruction	Generates camera poses and a sparse 3D point cloud from the matched images.
OpenMVS Module	Performs dense reconstruction, point fusion, mesh generation, and texture mapping.
Flask Web Module	Provides the web interface for image upload, reconstruction control, status, and result access..
Output & Visualization Module	Provides the final PLY/OBJ model for visualization through the web interface or MeshLab..

The actual output can be analyzed visually at different stages. MeshLab displays the dense point-cloud representation and the reconstruction layers named “scene_dense” and “scene_mesh,” while the Flask result page titled “Textured 3D Reconstruction” displays the completed textured model. These outputs show the implementation of dense reconstruction, mesh generation, texturing, and web-based presentation.

The reconstruction model also includes a visualization layer in which the generated outputs are examined at different stages. MeshLab displays the dense reconstruction and mesh layers, including “scene_dense” and “scene_mesh,” allowing the point-based and surface-based representations to be viewed separately. This provides qualitative confirmation that the pipeline produces the expected intermediate model outputs.

The Flask result interface extends the analysis beyond local model inspection by presenting the completed textured reconstruction through a web page titled “Textured 3D Reconstruction.” This demonstrates that the final mesh and texture output is connected to the web application and can be accessed through the result page. The PLY/OBJ outputs can also be visualized using the web-based viewer or MeshLab.

The current analysis remains qualitative because the supplied project material does not include reference models, accuracy measurements, benchmark datasets, execution times, memory usage, or quantitative comparisons. Therefore, the screenshots support conclusions about the presence and presentation of the dense point cloud, surface mesh, and textured model, but they do not establish reconstruction accuracy or performance.

The modeling pipeline provides a clear transition from image-based observations to a complete 3D representation. COLMAP establishes the camera poses and sparse geometry, while OpenMVS increases the reconstruction detail through dense point generation, point fusion, mesh construction, and texturing.

VII. PROPOSED SYSTEM

The proposed system is a software-based pipeline for reconstructing a three-dimensional object from multiple overlapping two-dimensional images. The system receives

images of the same object captured from different viewpoints and processes them through image validation, sparse reconstruction, dense reconstruction, mesh generation,

texture generation, and visualization. The proposed design follows a sequential modular structure so that each stage produces the input required by the next stage.

The first stage validates and organizes the uploaded images. OpenCV and standard image-handling operations are used for image validation and preprocessing. COLMAP then extracts visual features and matches corresponding features across the input views. Using these correspondences, COLMAP performs camera calibration or estimation and applies Structure from Motion to estimate camera poses and generate a sparse 3D point cloud. Image undistortion is also included in the COLMAP module.

The sparse COLMAP scene is converted into OpenMVS format using Interface COLMAP. OpenMVS performs dense Multi-View Stereo to estimate dense information from the available images and camera poses. The dense points are generated and fused, after which a surface mesh is reconstructed and texture information is applied using the OpenMVS.

The proposed system consists of a web-based application and a computer-vision reconstruction pipeline. The system accepts multiple 2D images through the Flask interface, validates and organizes them, and uses COLMAP for feature extraction, feature matching, camera pose estimation, and Structure from Motion to generate a sparse 3D point cloud. The reconstruction is then transferred to OpenMVS for dense reconstruction, point fusion, surface mesh generation, and texture mapping. The final PLY/OBJ model is accessed through the web interface or inspected using MeshLab.

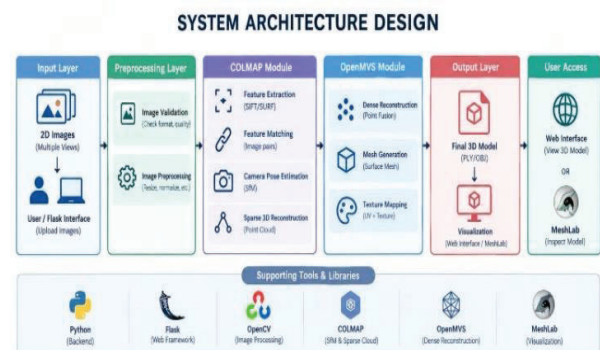


Figure 1: System architecture of the 3D object reconstruction from 2D images.

Figure 3: System architecture of the 3D object reconstruction from 2D images.

The proposed system architecture consists of a Flask-based web application integrated with COLMAP and OpenMVS to reconstruct a 3D object from multiple 2D images. The user first uploads overlapping images through the web interface, where the images are validated and organized. COLMAP then performs feature extraction, feature matching, camera calibration, and Structure from Motion (SfM) to estimate camera poses and generate a sparse 3D point cloud.

The generated COLMAP reconstruction is converted into an OpenMVS scene for dense Multi-View Stereo processing. OpenMVS generates dense depth information, fuses the points, reconstructs the surface mesh, and applies texture to produce the final PLY/OBJ model. The resulting 3D model can then be displayed through the Flask web interface or visualized using MeshLab, providing a complete workflow from image upload to final 3D reconstruction.

VIII. EXISTING SYSTEM

Existing 3D reconstruction systems generally create three-dimensional models from multiple images or views by estimating depth and camera geometry. Since ordinary 2D images do not directly provide depth information, these systems require relationships between different viewpoints to recover the 3D structure. Some existing approaches may depend on specialized 3D capture hardware, multiple cameras, or complex processing pipelines, which can increase the cost and complexity of reconstruction.

Traditional reconstruction workflows commonly involve multiple stages for recovering the geometry of an object from its images. The process requires extracting visual information from different views, matching corresponding features, estimating camera positions, and generating a 3D representation. The reconstruction can then be refined into a surface model for visualization.

A limitation of such existing approaches is their dependence on suitable image acquisition and processing resources. Systems using dedicated 3D capture hardware or multiple cameras can make the reconstruction process less accessible when only ordinary images and affordable hardware are available. Reconstruction quality can also depend on having sufficient overlapping viewpoints and successful feature matching between images.

Therefore, the proposed project addresses these limitations through a software-based workflow that uses ordinary multi-view images. Instead of requiring dedicated 3D capture hardware, the system combines COLMAP for feature extraction, feature matching, camera estimation and Structure from Motion with OpenMVS for dense reconstruction, mesh generation and texturing. A Flask web application manages image upload, processing and access to the final PLY/OBJ model, which can also be visualized using MeshLab.

Table 2: Comparison Between Existing and Proposed System

Feature	Existing System	Proposed System
Input Images	Input Images	Multiple overlapping 2D images
Image Processing	Manual / password based	Facial verification + liveness
Feature Extraction	Conventional image processing	COLMAP feature extraction
Camera Estimation	Limited/manual estimation	Structure from Motion (SfM)
3D Reconstruction	Basic reconstruction	COLMAP + OpenMVS
Surface Generation	Limited mesh generation	Surface mesh + texture generation
Output & Visualization	Basic 3D model	PLY/OBJ model with Flask/MeshLab visualization

IX. IMPLEMENTATION

The implementation is organized as a modular software pipeline that converts multiple overlapping images of one object into a viewable 3D model. The system includes an Image Upload Module, a COLMAP Module, an OpenMVS Module, a Flask Web Module, and an Output Module. This organization separates input handling, reconstruction processing, user interaction, and result storage, making the workflow clear and easier to follow.

The Image Upload Module accepts multiple images captured from different viewpoints. The uploaded files are validated and stored as reconstruction input. OpenCV and standard image-handling operations support image validation and

preprocessing before the files are supplied to COLMAP. The supplied project material does not specify additional preprocessing algorithms or numerical image-quality thresholds.

The COLMAP Module performs feature extraction and feature matching across the input images. It also performs camera estimation and Structure from Motion to estimate camera poses and generate the sparse 3D point cloud. The module includes mapper/SfM processing and image undistortion as part of the sparse-reconstruction stage.

The OpenMVS Module receives the converted COLMAP scene and performs dense reconstruction. Reconstruct Mesh for surface generation, and Texture Mesh for applying texture information. The dense points are fused before the final mesh and textured model are produced.

The Flask Web Module provides the user interface for the reconstruction process. It supports image upload, reconstruction control, processing-status display, error/status information, and access to the final result page. The final reconstructed model can be presented through the Flask interface, while MeshLab can be used to inspect the dense point cloud and mesh layers.

The Output Module stores the dense point cloud and the final mesh and texture files. The resulting model can be exported or viewed in ply or obj format. The implementation screenshots show dense reconstruction, mesh-related layers named "scene.dense" and "scene.mesh," and a web page titled "Textured 3D Reconstruction." These outputs demonstrate the implemented processing stages, but the project material does not provide numerical accuracy, execution time, memory usage, benchmark data, or quantitative comparisons.

The modules operate as a connected workflow in which the output of one stage becomes the input for the next. After the Flask application receives and organizes the images, COLMAP produces the camera information and sparse reconstruction required by OpenMVS and then, generates the dense point cloud, surface mesh, and textured model, while the Output and Flask modules make these results available for MeshLab inspection or web-based presentation.

The operational sequence begins with the Flask Web Module, where the user uploads a set of overlapping images of the same object. The system validates the input files and organizes them for processing before invoking COLMAP. COLMAP then performs feature extraction, feature matching, camera estimation, Structure from Motion, and image undistortion. The resulting camera poses and sparse 3D point cloud provide the geometric foundation required for the subsequent dense-reconstruction stage.

After sparse reconstruction, the generated scene is converted into an OpenMVS-compatible format and processed through dense multi-view stereo. OpenMVS estimates dense information, generates and fuses dense points, reconstructs the object surface, and applies texture mapping to the mesh. The final outputs are stored as dense point-cloud, mesh, texture, PLY, or OBJ files and can be inspected using MeshLab or displayed through the Flask result page. This modular implementation allows each stage to be executed and examined separately while maintaining a continuous workflow from input images to the final textured 3D model.

X. RESULTS AND DISCUSSION

The implemented system produces a viewable 3D model from multiple overlapping images of the same object. The processing pipeline generates a sparse point cloud through COLMAP and subsequently produces a dense point cloud, surface mesh, and textured model through OpenMVS. These outputs demonstrate the successful movement of data from image input to final 3D reconstruction.

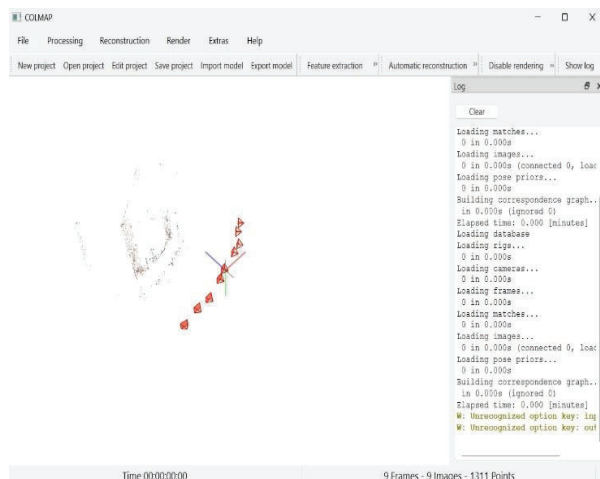


Figure 4: COLMAP – Sparse Reconstruction

The sparse reconstruction contains estimated camera poses and sparse 3D points obtained through feature extraction, feature matching, and Structure from Motion. The dense reconstruction stage expands this representation by estimating dense information and fusing the generated points. The resulting dense point cloud provides the geometric basis for reconstructing the object surface.

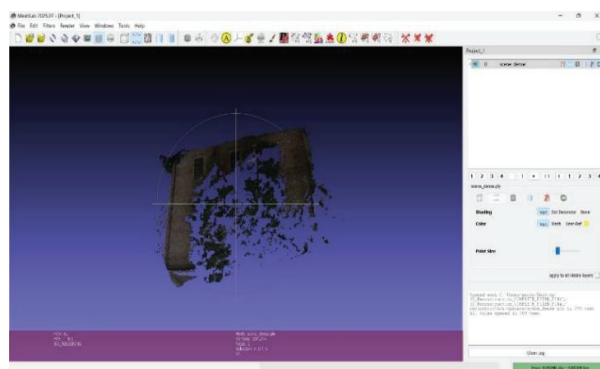


Figure 5: Dense Point Cloud – MeshLab

The surface reconstruction stage generates a mesh from the fused dense point cloud and applies texture information to the mesh. The generated model can be stored in PLY or OBJ format and inspected using MeshLab. The implementation also provides a Flask-based result page for displaying the final textured reconstruction, demonstrating the integration of the reconstruction tools with a web interface..

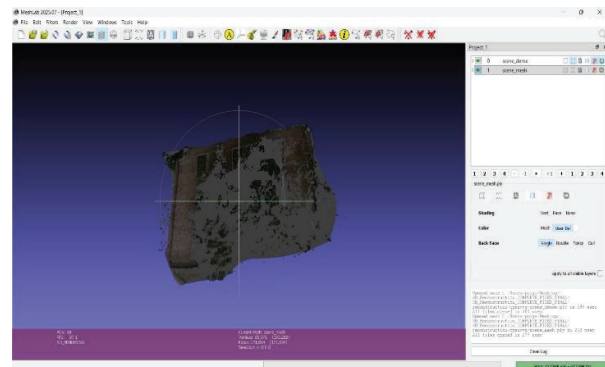


Figure 6: Reconstructed 3D surface mesh in MeshLab

The results show that the proposed workflow combines image preprocessing, COLMAP, OpenMVS, Flask, and visualization tools into one practical system. The use of ordinary multi-view images avoids the need for dedicated 3D capture hardware, while the modular design allows the sparse point cloud, dense point cloud, mesh, and textured model to be examined separately.

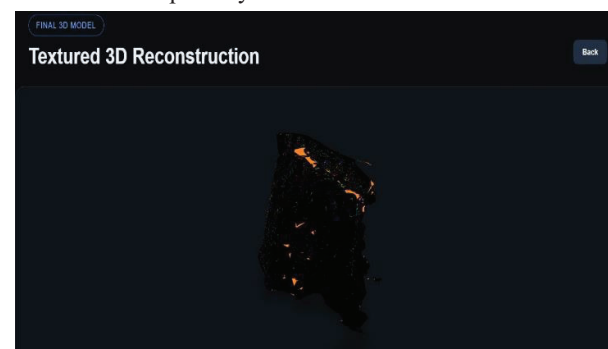


Figure 7: Final textured 3D model displayed through the Flask web .

Overall, the obtained outputs demonstrate the complete flow from input images to sparse reconstruction, dense point cloud, surface mesh, and final textured model. The results show that COLMAP and the subsequent reconstruction stages can be integrated into a practical 3D reconstruction workflow, while MeshLab provides a convenient environment for inspecting intermediate outputs. The quality of the final reconstruction depends on factors such as the number of input images, overlap between viewpoints, image quality, and successful feature matching. The supplied project material does not provide numerical accuracy measurements or benchmark comparisons; therefore, the results are discussed qualitatively based on the generated outputs.

Advantages

- Uses ordinary images and affordable software.
- Provides a complete 2D-to-3D reconstruction pipeline.
- Uses modular COLMAP, OpenMVS, and Flask components.
- Supports PLY and OBJ model formats.
- Provides web-based model visualization.
- Allows inspection of point clouds and meshes in MeshLab.

Limitations

The reconstruction quality depends on sufficient image overlap and successful feature matching. Poor lighting, blurred images, or texture less surfaces may reduce the quality of the

generated model. Dense point-cloud and mesh processing may require considerable time and computing resources. The project does not provide numerical accuracy, benchmark, or performance measurements. Incomplete object views may also result in missing or inaccurate parts of the final 3D model.

XI. FUTURE ENHANCEMENT

Future work may improve image validation and preprocessing so that the system can handle images affected by blur, poor lighting, limited texture, or insufficient overlap. Better preprocessing can improve feature matching, camera-pose estimation, and the completeness of the reconstructed 3D model.

The Flask application may be enhanced with detailed progress monitoring, clearer error messages, improved result management, and more interactive web-based visualization. Additional output formats and improved support for PLY and OBJ models can also make the reconstructed results easier to export, inspect, and share.

Future evaluation may include reconstruction accuracy, processing time, memory usage, and comparisons across different image sets and object types. These measurements would provide quantitative evidence of system performance and help identify improvements required for reliable real-world use.

XII. CONCLUSION

This project presented a software-based approach for reconstructing a three-dimensional object from multiple overlapping two-dimensional images. The implemented pipeline combines COLMAP, OpenMVS, OpenCV, Flask, and visualization tools to process image input and generate a viewable 3D model.

COLMAP performs feature extraction, feature matching, camera estimation, and Structure from Motion to generate camera poses and a sparse 3D point cloud. OpenMVS then performs dense Multi-View Stereo, point fusion, surface-mesh reconstruction, and texture mapping. The final PLY or OBJ model can be inspected using MeshLab or displayed through the Flask web interface.

The project demonstrates that ordinary multi-view images can be used to produce a dense point cloud, reconstructed mesh, and textured 3D model without specialized 3D capture hardware.

However, reconstruction quality depends on suitable image overlap and successful feature matching, and the current work does not include numerical accuracy or performance measurements.

REFERENCES

- [1] Wang et al., "Deep Learning-Based 3D Reconstruction from Multiple Images: A Survey," *Neurocomputing*, 2024.
- [2] Kononov et al., "Reconstruction of 3D Structures from 2D Images using GAN," *ICCV Workshops*, 2023.
- [3] Wang et al., "Advanced 3D Face Reconstruction from a Single Image," *IEEE Transactions on Image Processing*, 2024.
- [4] Chen et al., "Neural Radiance Fields for 3D Reconstruction," *CVPR*, 2024.
- [5] Mohamed et al., "3D Reconstruction from Multi-View Images using EfficientNetB0," *IEEE Access*, 2025.
- [6] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards Internet-Scale Multi-View Stereo," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2010, pp. 1434–1441.
- [7] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards Internet-Scale Multi-View Stereo," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2010, pp. 1434–1441.
- [8] Y. Furukawa and C. Hernández, "Multi-View Stereo: A Tutorial," *Foundations and Trends in Computer Graphics and Vision*, vol. 9, nos. 1–2, pp. 1–148, 2015.
- [9] Y. Furukawa and J. Ponce, "Accurate, Dense, and Robust Multiview Stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1362–1376, 2010.
- [10] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: An Open-Source Mesh Processing Tool," in *Proc. Eurographics Italian Chapter Conf.*, 2008, pp. 129–136.
- [11] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2006, pp. 519–528.
- [12] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building Rome in a Day," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2009, pp. 72–79.
- [13] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, 2006.
- [14] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," in *Proc. Symposium on Geometry Processing*, 2006, pp. 61–70.
- [15] M. Kazhdan and H. Hoppe, "Screened Poisson Surface Reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, 2013.
- [16] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "MVSNet: Depth Inference for Unstructured Multi-View Stereo," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2018, pp. 767–783.
- [17] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise View Selection for Unstructured Multi-View Stereo," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2016, pp. 501–518.
- [18] M. Waechter, N. Moehrle, and M. Goesele, "Let There Be Color! Large-Scale Texturing of 3D Reconstructions," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2014, pp. 836–850.
- [19] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.
- [20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2020.