Structure from Motion 3D Reconstruction

Biruk Abere
Paris Saclay University
biruk-abere.ambaw@etu-upsaclay.fr

Felipe Espinosa
Paris Saclay University
jose.espinosa-orjuela@etu-upsaclay.fr

Ali Khayat
Paris Saclay University
ali.khayat@etu-upsaclay.fr

Abstract

One of the main difficulties in image-based modeling and computer vision is creating a 3D model from 2D images that is as realistic as possible. The development of three-dimensional models from a group of photos is known as 3D reconstruction from multiple photographs. Recent and rapid advances in the domains of autonomous driving and augmented reality, which rely significantly on precise 3D reconstructions of the surrounding world, are approximated by combining depth readings from sensors such as LIDAR, structured light, and other specific sensors. The disadvantage of these sensors is that they require special hardware, which makes them more effective but also more complicated to acquire and use than systems that rely solely on RGB camera systems. This shift aims to reduce cost and complexity without sacrificing the quality and realism of the resulting models. In this work, we explore quality 3D reconstruction from multiple 2D images that uses geometry and computer vision principles. https://github.com/Biruk-Abere/3D-reconstruction-from-multi-view-images

1. Introduction

The current advancements in digital cameras, as well as an improvement in picture resolution and clarity, have opened up new approaches to rebuild 3D images utilizing various techniques that employ merely these cameras rather than pricey special sensors, making the reconstruction process relatively affordable. The goal of the reconstruction is to derive the geometrical structure of a scene from a set of photos, assuming that the camera position and internal parameters are known or can be guessed from the set of images. This is accomplished by employing numerous photos in which the 3D information may be (partially) retrieved by applying the Structure from Motion approach to solve the pixel-wise correspondence problem.

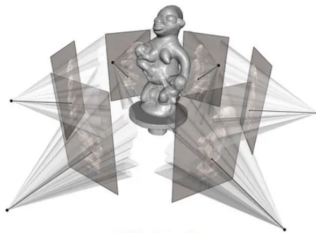

Figure 1. 3D Reconstruction from multi-view images

The process of reconstructing a three-dimensional structure using projections acquired from a succession of photographs from diverse viewpoints is known as Structure from Motion (SfM). This technique produces advanced state-of-the-art findings, but the technique's primary concerns are resilience, precision, completeness, and scalability, which are handled using an incremental approach for the structure from motion. LIDAR-based 3D reconstruction of a scene is costly and prone to artifacts from GPS and IMU. Structure from Motion method uses just low-cost camera images to rebuild a 3D scene while also obtaining the camera poses of the monocular camera in relation to the provided scene.

2. Problem Definition

In this work, we address the problem of 3D scene reconstruction from a sequence of images captured by a single (monocular) camera. Formally, given a set of n images $\{I_1, I_2, \ldots, I_n\}$ that have overlapping fields-of-view, our goal is to estimate both the camera poses and the 3D structure of the scene. For each image I_i, we seek to determine the corresponding camera rotation matrix $\mathbf{R}_i \in SO(3)$ and translation vector $\mathbf{t}_i \in \mathbb{R}^3$, and to reconstruct a set of 3D points $\{\mathbf{X}_j\}_{j=1}^{m}$ such that the projections of these points onto the images are as close as possible to the observed 2D

keypoints.

Let $\mathbf{K} \in \mathbb{R}^{3\times3}$ denote the intrinsic camera matrix and define the projection matrix for image I_i as

$$\mathbf{P}_i = \mathbf{K}\,[\mathbf{R}_i \,|\, \mathbf{t}_i] \in \mathbb{R}^{3\times4}.$$

For a 3D point $\mathbf{X}_j \in \mathbb{R}^3$, its homogeneous coordinate is given by $\widetilde{\mathbf{X}}_j = [X_j,\, Y_j,\, Z_j,\, 1]^\top$. The projection of $\mathbf{X}_j$ onto image I_i is computed by

$$\lambda\, \mathbf{x}_{ij} = \mathbf{P}_i \widetilde{\mathbf{X}}_j,$$

where $\mathbf{x}_{ij} \in \mathbb{R}^2$ is the observed image point corresponding to $\mathbf{X}_j$, and λ is an arbitrary scaling factor.

Our objective is to recover the unknown camera parameters $\{\mathbf{R}_i, \mathbf{t}_i\}$ and the 3D structure $\{\mathbf{X}_j\}$ by minimizing the overall reprojection error:

$$\min_{\{\mathbf{R}_i,\,\mathbf{t}_i\},\,\{\mathbf{X}_j\}} \sum_{i=1}^{n} \sum_{j\in\mathcal{V}_i} \left\| \mathbf{x}_{ij} - \pi\left(\mathbf{K}[\mathbf{R}_i \,|\, \mathbf{t}_i]\widetilde{\mathbf{X}}_j\right)\right\|_2^2, \quad (1)$$

where $\pi(\cdot)$ denotes the projection from homogeneous to Euclidean coordinates (i.e., division by the last coordinate), and $\mathcal{V}_i$ is the set of 3D points visible in image I_i.

The problem is challenging for several reasons:

- **Non-linearity and Non-convexity:** The mapping from 3D points and camera parameters to 2D image points is inherently non-linear, which leads to a non-convex optimization problem with many local minima.
- **Scale Ambiguity:** In monocular SfM, absolute scale cannot be determined from images alone, meaning that the reconstruction is defined only up to a scaling factor.
- **Outlier Sensitivity:** Feature detection and matching are prone to errors; incorrect correspondences (outliers) can drastically affect the quality of the estimated camera poses and 3D structure.
- **High Dimensionality:** The large number of variables (both camera parameters and 3D points) increases the computational complexity and the risk of error propagation throughout the reconstruction process.

To address these challenges, our pipeline employs robust feature extraction (via SIFT), efficient matching (using FLANN with Lowe's ratio test), and robust estimation methods (using RANSAC for both essential matrix and PnP estimation). Our goal is to optimize the camera parameters and 3D point positions such that the total reprojection error (Equation 1) is minimized, thereby producing a geometrically consistent and visually coherent 3D reconstruction of the scene.

The formal definition of our problem underscores both the optimization objective and the inherent hardness of the task, which is compounded by the non-linear, high-dimensional, and scale-ambiguous nature of monocular 3D reconstruction.

3. Related Work

Structure from Motion (SfM) has been a fundamental topic in computer vision for many years, and numerous approaches have been proposed to tackle the problem of recovering camera motion and scene structure from images. Early work introduced feature-based methods to estimate motion by leveraging robust point correspondences across multiple views[1], laying the foundation for incremental SfM pipelines.

Subsequent research has addressed the scalability of correspondence search in large, unordered image collections. VocMatch[2], for instance, uses a visual vocabulary to accelerate feature matching, thus reducing the computational bottleneck typical of exhaustive pairwise matching. This aligns with our approach of efficiently handling large datasets by focusing on robust feature correspondence strategies.

More recently, efforts have been made to systematically evaluate and compare SfM pipelines. In particular, Bianco [3] discuss performance metrics for reconstruction accuracy and camera pose estimation, providing insights into pipeline strengths and weaknesses. Their methodology for quantitative assessment is closely related to our own strategy of evaluating the completeness and accuracy of 3D reconstructions.

Finally, the work of Schönberger and Frahm [4] revisits incremental SfM, proposing enhancements for robustness, accuracy, and efficiency. Their pipeline, which refines both correspondence search and reconstruction initialization, shares our goal of building a more reliable and scalable SfM system.

Overall, our project builds upon these prior contributions by focusing on robust feature matching, thorough evaluation, and improved incremental reconstruction strategies. In particular, we integrate lessons learned from these works into our pipeline design, aiming to further enhance completeness and accuracy in multi-view 3D reconstruction.

4. Methodology

In this work, we address the problem of reconstructing a 3D scene from a sequence of images using a Structure-from-Motion (SfM) pipeline. Our approach is modular and robust, and it consists of several key stages: data collection and preprocessing, initialization, feature extraction and matching, relative pose estimation, triangulation-based 3D point cloud construction, iterative camera pose refinement with Perspective-n-Point (PnP) registration and point cloud update, and final output generation.

The basic idea in 3D image reconstruction is that given a set of images $\{I_1, \ldots, I_N\}$ where each image is taken from a different viewpoint, our goal is to use these images to reconstruct a three-dimensional representation of the object. More specifically, we will find the motions of the cameras with respect to a world coordinate frame F_W. This motion of cameras is also known as camera projection matrices $\{P_1, \ldots, P_N\}$. Using this set of camera projections, we will then use different algorithms to recover the 3D structures of the scene.

In the following, we detail each step, present the mathematical background behind the methods, discuss the specific algorithmic choices and parameters, and highlight the limitations and challenges encountered.

4.1. Data Collection and Preprocessing

Our dataset comprises images captured in common formats (e.g., JPEG, PNG) collected from multiple sources. The images are loaded from a specified directory and sorted in temporal order (e.g., `DSC_001.jpg`, `DSC_002.jpg`, `...`), ensuring sequential consistency essential for reconstruction. To reduce computational load and memory usage, the images are downsampled using OpenCV's pyramid downsampling method. The number of downsampling steps is computed as $\log_2(d)$, where d is the user-defined downsampling factor. This preprocessing stage prepares the data for efficient feature extraction and subsequent processing.

4.2. Camera Intrinsics Setup

The camera intrinsic matrix, often denoted as K, describes how a 3D point in the camera's coordinate system is projected onto the 2D image plane. In many practical applications, the intrinsic parameters of the camera (focal lengths and principal point) may not be known a priori. To address this, we compute a default intrinsic matrix $\mathbf{K}$ from the dimensions of the first image. Assuming square pixels, we set the focal lengths as $f_x = f_y = w$, where w is the width of the image, and the principal point as $(c_x, c_y) = (w/2, h/2)$, with h being the image height. The intrinsic matrix is given by:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

If we scale (downsample) your images by a factor, the apparent focal length in pixel coordinates also changes. Concretely, if we halve your image size (width and height), then the number of pixels per real-world unit also halves. Hence, f_x, f_y, c_x, and c_y must be divided by the same downsampling factor to keep the geometric relationships accurate in the smaller image.

If we *do not* adjust the camera intrinsics after resizing images, subsequent calculations (e.g., pose estimation, triangulation) will be misinformed, causing inaccuracies in the 3D reconstruction.

4.3. Initialization

The initialization phase establishes the reference frame for the reconstruction. We assume that the first camera defines the world coordinate system, setting its pose to an identity transformation:

$$\mathbf{P}_1 = \mathbf{K}[\mathbf{I} \,|\, \mathbf{0}], \tag{3}$$

We initialize data structures for storing the cumulative 3D point cloud and the associated pixel colors. This setup is crucial as it provides a baseline from which all subsequent camera poses and 3D points will be derived.

4.4. Feature Extraction and Matching

Robust feature extraction is critical for establishing reliable correspondences across images. We use the Scale-Invariant Feature Transform (SIFT) due to its invariance to scale, rotation, and illumination variations. SIFT detects keypoints and computes 128-dimensional descriptors that uniquely characterize local image patches. In our implementation, the SIFT detector is instantiated (with a fallback to `xfeatures2d.SIFT_create()` for older versions of OpenCV) and applied to the grayscale versions of the images.

To match the extracted descriptors between images, we employ a FLANN-based (Fast Library for Approximate Nearest Neighbors) KNN matcher. The FLANN matcher is configured with a KD-tree index (using `algorithm = 1` and `trees = 5`) and a search parameter (`checks = 50`), which balance the trade-off between speed and accuracy in high-dimensional spaces. Following descriptor matching, Lowe's ratio test is applied to each pair of candidate matches. Specifically, if the ratio of the distance of the best match to the second-best match is less than 0.7, the match is retained; otherwise, it is discarded. This filtering step removes ambiguous matches and minimizes the effect of outliers, establishing a robust set of correspondences for further processing.

4.5. Relative Pose Estimation

With reliable matches between the first two images, we compute the Essential Matrix $\mathbf{E}$ that encapsulates the epipolar geometry:

$$\mathbf{x}_2^\top \mathbf{E} \mathbf{x}_1 = 0, \tag{4}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the homogeneous coordinates of matched keypoints. The Essential Matrix is estimated using RANSAC (Random Sample Consensus), which iteratively fits the model while rejecting outliers. We set the RANSAC probability to 0.999 and the inlier threshold to 0.4, ensuring high confidence in the estimated model.

Once $\mathbf{E}$ is obtained, we decompose it to recover the relative rotation $\mathbf{R}$ and translation $\mathbf{t}$ (up to a scale factor) between the two cameras. This decomposition is typically performed via singular value decomposition (SVD) and further refined using inlier masks. The recovered $\mathbf{R}$ and $\mathbf{t}$ are then used to construct the second camera's projection matrix:

$$\mathbf{P}_2 = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]. \tag{5}$$

This step is fundamental as it defines the geometric relationship between the views, which is essential for triangulating 3D points.

4.6. Triangulation and Initial 3D Point Cloud Construction

Triangulation converts the 2D correspondences from two views into 3D points. Using the projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, along with the matched keypoints, we apply OpenCV's `cv2.triangulatePoints()` function. This function implements the Direct Linear Transform (DLT) algorithm, which formulates the projection equations for a 3D point $\mathbf{X} = (X, Y, Z, W)^\top$ as:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{P}\mathbf{X}. \tag{6}$$

The solution is obtained in homogeneous coordinates; hence, the output is a $4 \times N$ matrix where each column represents a point in homogeneous form. Normalization is achieved by dividing by the fourth component W, yielding Euclidean coordinates:

$$(X, Y, Z) = \left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right). \tag{7}$$

This initial point cloud provides the basis for incremental reconstruction.

4.7. Iterative PnP Registration and Point Cloud Update

The most critical component of our pipeline is the iterative integration of new images to refine the 3D model. This process involves several steps:

1. **Re-triangulation:** For iterations beyond the first, previously matched keypoints are re-triangulated to refine the existing 3D structure. This involves calling the triangulation routine with the latest projection matrices.
2. **Feature Extraction on New Image:** The next image in the sequence is loaded and downsampled. Features are extracted from this new image and matched with the preceding image using the same SIFT and FLANN-based procedures.
3. **Three-View Correspondence:** To ensure temporal consistency, we identify keypoints common across three consecutive images. Our custom function `three_view_points()` compares keypoints from the previous image pair with those from the new image pair. It outputs indices of common keypoints (used for camera pose estimation) and distinguishes them from newly observed keypoints (to be triangulated later).
4. **PnP-Based Pose Estimation:** With the common keypoints and their corresponding 3D positions from the previous reconstruction, we use the Perspective-n-Point (PnP) algorithm to estimate the camera pose of the new image. We employ a RANSAC-based solver in `cv2.solvePnPRansac()` to robustly compute the rotation $\mathbf{R}$ and translation $\mathbf{t}$ that minimize the reprojection error:

$$\text{Reprojection Error} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2, \tag{8}$$

   where $\mathbf{x}_i$ are the observed 2D keypoints and $\hat{\mathbf{x}}_i$ are the projections of the corresponding 3D points.
5. **New Point Triangulation:** Keypoints that are unique to the new image (i.e., not seen in previous images) are triangulated with the previous view to compute their 3D positions. These new points are then appended to the existing point cloud.
6. **Color Extraction:** The corresponding pixel colors for the newly triangulated points are extracted from the new image, ensuring that the final point cloud is not only spatially accurate but also visually informative.
7. **Update Variables:** Finally, we update the projection matrices, keypoints, and image references to prepare for the next iteration.

This iterative refinement is key to maintaining a consistent and accurate 3D reconstruction as additional viewpoints are incorporated.

4.8. Final Output Generation

Once all images have been processed, the final 3D point cloud—consisting of spatial coordinates and associated RGB colors—is exported to a PLY file. The 3D points are first scaled (by an empirical factor, e.g., 200) to improve visualization, and then outlier removal is performed by filtering points based on their Euclidean distance from the mean. The PLY file header is generated to conform with the standard format, specifying the number of vertices and their properties (coordinates and color channels). The combined data is written in ASCII format, facilitating easy inspection and further processing in visualization tools such as MeshLab. In the color assignment step, each triangulated 3D point is matched with its corresponding pixel value in the image (after verifying that the keypoint falls within image bounds) to ensure that the final model accurately represents the scene's appearance.

4.9. Mathematical Background and Algorithmic Details

Our methodology leverages several foundational algorithms:

- **SIFT (Scale-Invariant Feature Transform):** Introduced by Lowe (2004), SIFT detects keypoints in scale-space and computes descriptors that are invariant to scale, rotation, and illumination. Mathematically, SIFT involves finding extrema in the Difference of Gaussians (DoG) pyramid and computing gradient histograms to form a robust descriptor.
- **FLANN (Fast Library for Approximate Nearest Neighbors):** FLANN provides efficient approximate nearest neighbor searches in high-dimensional spaces. The KD-tree index, configured with parameters such as `trees = 5` and `checks = 50`, is used to balance speed and accuracy during the matching process.
- **RANSAC (Random Sample Consensus):** RANSAC (Fischler and Bolles, 1981) is used to robustly estimate the Essential Matrix and the PnP parameters by iteratively selecting random subsets of matches, fitting a model, and computing inlier counts. High confidence (e.g., 99.9%) and an error threshold (e.g., 0.4) are set to ensure the reliability of the estimation.
- **Essential Matrix Decomposition:** The Essential Matrix $\mathbf{E}$, which satisfies $\mathbf{x}_2^\top \mathbf{E}\mathbf{x}_1 = 0$, is decomposed (often via Singular Value Decomposition) to extract the relative rotation $\mathbf{R}$ and translation $\mathbf{t}$ (up to an unknown scale factor) between views.
- **Triangulation via Direct Linear Transform (DLT):** Given two projection matrices and corresponding 2D points, the triangulation process solves a system of linear equations to find the 3D point in homogeneous coordinates. Normalization of these coordinates yields the actual Euclidean coordinates.
- **Perspective-n-Point (PnP):** The PnP problem involves estimating the camera pose from known 3D-2D correspondences. Using methods such as `cv2.solvePnPRansac()`, we robustly estimate the rotation and translation that best align the observed 2D points with their corresponding 3D points.

4.10. Limitations and Difficulties

While our SfM pipeline successfully reconstructs a sparse 3D scene, several challenges remain:

- **Scale Ambiguity:** Monocular SfM does not provide absolute scale, leading to ambiguity in the size of the reconstructed scene.
- **Computational Complexity:** Feature matching and RANSAC-based estimation are computationally intensive, particularly for large datasets.
- **Error Propagation:** Inaccuracies in feature detection or matching can propagate through the pipeline, affecting camera pose estimation and triangulation.
- **Outlier Sensitivity:** Despite robust methods, some erroneous matches may remain and adversely affect the reconstruction quality.

Future work could focus on incorporating global bundle adjustment and more advanced outlier rejection techniques to further refine the reconstruction.

5. Evaluation

5.1. Reprojection Error

A primary metric for assessing reconstruction quality is the reprojection error, measured as the average 2D distance (in pixels) between the reprojected 3D points and the corresponding detected keypoints in each image. Our pipeline reports this value at multiple stages:

- **Initial Two-View Reconstruction.** After processing the first two images, the system yields an initial average reprojection error. The error is initially very low (on the order of 0.003–0.007 pixels) demonstrating that the initial essential matrix estimation and triangulation yield a coherent baseline reconstruction. This error typically reflects the baseline accuracy of the essential matrix estimation and subsequent triangulation.
- **Incremental Updates.** As each new image is added, the pipeline re-estimates camera pose through the Perspective-n-Point (PnP) procedure and triangulates additional points. The reprojection error is updated accordingly, giving insight into how well the expanding reconstruction remains consistent with incoming views.
  For example, in one run using the `castle-P19` set with an externally provided calibration matrix, the reprojection error evolves as follows:
  - After 2 images: 0.0037 pixels.
  - After 3 images: 9.8868 pixels.
  - After 4 images: 2.2954 pixels.
  - After 5 images: 21.9938 pixels.
  - After 6 images: 2.7056 pixels.

  These fluctuations reflect the inherent challenges in matching features across views that differ in perspective. In some cases, a significant spike in error (e.g., 127.6947 pixels after 14 images in one instance of the castle-P19 set) may indicate difficulties due to extreme viewpoints.
  Similar trends are observed across different sets (e.g., `castle-P30`, `fountain-P11`, `GustavIIAdolf`, and `Herz-Jesus-P8`). Those results can be found on the repository (add link to the repo)

5.1.1. Key Findings

1. **Calibration Impact:** Runs that load an external calibration matrix often report reprojection errors comparable to or slightly lower than those obtained using the default computed matrix. This indicates that, while our default

calibration is robust, having precise calibration data can further improve accuracy.

2. **Incremental Stability:** Despite occasional large error spikes, the overall trend in most datasets is a stable or gradually increasing reprojection error, suggesting that our iterative refinement strategy generally succeeds in maintaining a coherent global reconstruction.
3. **Error Spikes and Outliers:** Sudden increases in error are likely due to outliers in feature matching or challenging imaging conditions. These spikes signal potential areas for future improvement, such as integrating bundle adjustment or more robust outlier rejection mechanisms.

Overall, our tests showed that the reprojection error remains stable or gradually decreases as more images are processed, indicating a coherent global structure. Minor fluctuations may occur if incoming images feature challenging perspectives (e.g., extreme angles or limited overlap).

5.2. Qualitative Evaluation

In this section, we analyze the reconstructed 3D point clouds from a qualitative perspective, focusing on three main factors: the size of the dataset, the consistency of the input images, the availability of camera intrinsic parameters, and an overall assessment of the best results.

5.2.1. Effect of Dataset Size

To study how the number of images influences the final reconstruction, we experimented with two different configurations of the *castle* dataset: one containing 19 images and another containing 30 images. As shown in Figure 2, the reconstruction with 30 images yields a denser and more detailed point cloud compared to the 19-image configuration. The increased number of viewpoints provides more overlapping regions and keypoints, allowing the pipeline to better capture the scene's structure. However, while a larger dataset generally leads to a richer 3D model, it also increases the computational cost of feature extraction and matching.

5.2.2. Consistency in Image Acquisition

Another crucial aspect affecting reconstruction quality is the consistency and smooth transition between consecutive images. In the *fountain-P11* dataset (see Figure 3), only 11 images were used, yet the reconstruction is visually coherent and captures the main features of the fountain. This success stems from the fact that each new image is taken from a viewpoint that overlaps sufficiently with the previous images, ensuring robust feature matching. Abrupt changes in viewpoint or insufficient overlap often lead to mismatches, thereby increasing the reprojection error and degrading the quality of the reconstructed model.

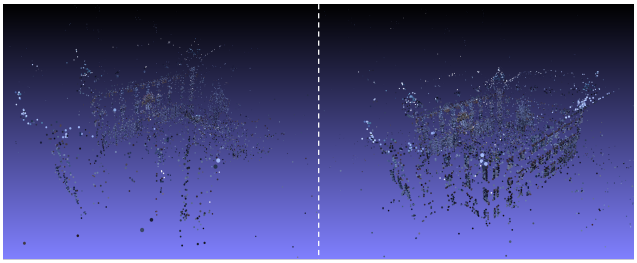

Figure 2. Comparison of two *castle* reconstructions with 19 images (left) vs. 30 images (right). The denser dataset yields a richer point cloud, as indicated by the increased coverage and detail.

5.2.3. Impact of Camera Intrinsic Parameters

When available, the use of known camera intrinsics improves the accuracy of the reconstruction. In our trials, datasets that included a calibration file showed more stable reprojection errors and produced clearer geometric details. By contrast, relying on default or estimated intrinsics can introduce scale and alignment uncertainties, especially for scenes with less texture or large perspective variations. Figure 4 illustrates the difference between reconstructions performed with and without the correct intrinsic parameters. Notably, using the correct $\mathbf{K}$ matrix helps preserve the true geometry of the scene, reducing error spikes during pose estimation.

5.2.4. Overall Best Results

Among the various datasets tested, our best visual outcomes were obtained with the *GustavIIAdolf* dataset (Figure 5). This scenario features:

- **A sufficient number of images**, providing comprehensive coverage around the statue.
- **Consistent overlap**, ensuring robust feature matching from one view to the next.
- **Accurate camera intrinsics**, yielding stable reprojection errors and preserving fine structural details.

These factors collectively lead to a well-defined 3D point cloud that accurately represents the statue's shape and contours, demonstrating the importance of combining a good number of images, consistent viewpoints, and correct camera parameters.

6. Future work

Early works for 3D reconstruction mostly use feature matching between different views of an object. However, the performance of such methods largely depends on accurate and consistent margins between different views of objects and is thus vulnerable to rapid changes between views. Additionally, these methods are not suitable for single-view 3D reconstruction, where only one view of an object is available.

Figure 3. *fountain-P11* reconstruction. Despite having only 11 images, consistent overlap and viewpoints lead to a clear 3D model.

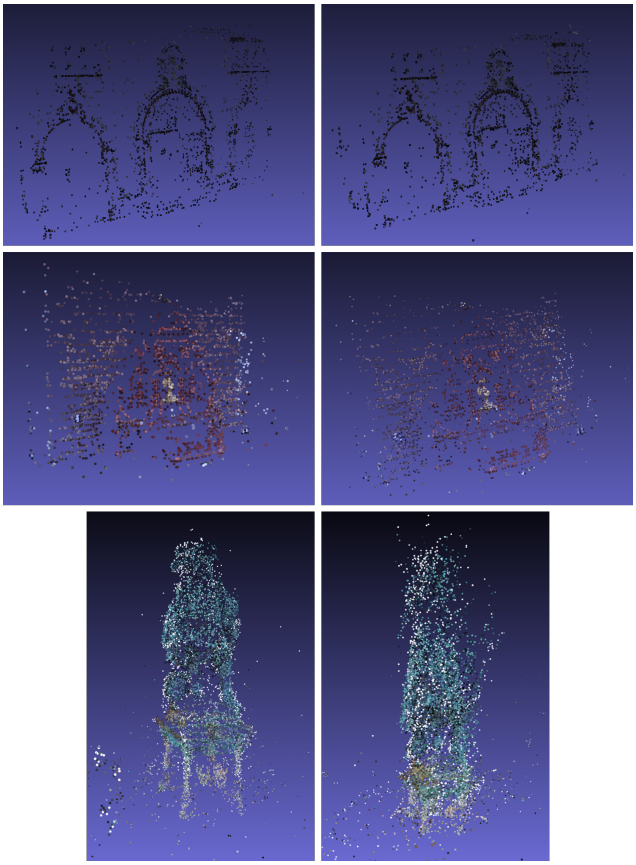

Figure 4. Comparison of reconstructions with known (left) vs. estimated intrinsics (right). Using accurate calibration typically reduces error spikes and preserves scene geometry.

The advances of deep learning have shed some light on neural network-based approaches for 3D reconstruction. On the one hand, some researchers formulate 3D reconstruction as a sequence learning problem and use recurrent neural networks to solve the problem. On the other hand, other researchers employ the encoder-decoder architecture for 3D reconstruction.

Recent methods are capable of performing end-to-end

Figure 5. Best result: *GustavIIAdolf* dataset. The combination of sufficient images, consistent overlap, and correct camera intrinsics leads to a high-quality reconstruction.

single and multi-view 3D reconstruction with Transformers. *3D-RETR* uses a pre-trained Transformer to extract visual features from 2D images. *3D-RETR* then obtains the 3D voxel features by using another Transformer Decoder. The *3D-RETR* model from consists of three main components: a Transformer Encoder, a Transformer Decoder, and a CNN Decoder. The Transformer Encoder takes as input the images, which are subsequently encoded into fixed-size image feature vectors. Then, the Transformer Decoder obtains voxel features by cross-attending to the image features. Finally, the CNN Decoder decodes 3D object representations from the voxel features.

The Vision Transformer uses image x_i as input and splits the image into B^2 patches. The corresponding patch is embedded at each time step by first transforming it into a fixed-size vector, which is then added to positional embeddings. In the Decoder, the M^3 learned positional embeddings are used as inputs, and the Transformer Encoder outputs are cross-attended by the Transformer Decoder. This decoder decodes all input vectors in parallel, rather than auto-regressively.

The Transformer Decoder produces voxel features that are fed into the CNN Decoder, which generates voxel features. The model uses the ShapeNet and Pix3 datasets to evaluate the model using the Intersection over Union (IoU), providing satisfactory results. The *3D-RETR* model used in is more efficient than previous models, as *3D-RETR* achieves better performance with significantly fewer parameters. The *3D-RETR* can be further improved by using additional transformers such as Performer, Reformer, etc.

7. Conclusion

Creating a 3D model from 2D images that is as realistic as possible is one of the fundamental issues of image-based modeling and computer vision. 3D reconstruction from multiple images is the creation of three-dimensional models from a set of images. The goal of multiview 3D reconstruction is to infer the geometrical structure of a scene captured by a collection of images. Usually, the camera position and internal parameters are assumed to be known or they can be estimated from the set of images. Recently, many applications have emerged, such as autonomous driving and augmented reality, which rely heavily upon accurate 3D reconstructions of the surrounding environment. These reconstructions are often estimated by fusing depth measurements from special sensors, such as structured light, time-of-flight, or LIDAR, into 3D models. While these sensors can be extremely effective, they require special hardware, making them more cumbersome and expensive than systems that rely solely on RGB cameras. Furthermore, they often suffer from noise and missing measurements due to low albedo, glossy surfaces, and occlusion. By using multiple images, 3D information can be (partially) recovered by solving a pixel-wise correspondence problem.

This work presents an implementation of a 3D reconstruction pipeline using incremental Structure-from-Motion (SfM) with feature-based data association, pose estimation, triangulation, and iterative point cloud refinement. The approach leverages SIFT for keypoint extraction, followed by Essential Matrix estimation and Perspective-n-Point (PnP) pose recovery to progressively build a sparse 3D point cloud. Triangulation is used to estimate the 3D structure of matched keypoints, and an iterative registration process allows for the integration of additional images to refine the reconstruction.

8. Future work

Early works for 3D reconstruction mostly use feature matching between different views of an object. However, the performance of such methods largely depends on accurate and consistent margins between different views of objects and is thus vulnerable to rapid changes between views. Additionally, these methods are not suitable for single-view 3D reconstruction, where only one view of an object is available.

The advances of deep learning have shed some light on neural network-based approaches for 3D reconstruction. On the one hand, some researchers formulate 3D reconstruction as a sequence learning problem and use recurrent neural networks to solve the problem. On the other hand, other researchers employ the encoder-decoder architecture for 3D reconstruction.

Recent methods are capable of performing end-to-end single and multi-view 3D reconstruction with Transformers. *3D-RETR* uses a pre-trained Transformer to extract visual features from 2D images. *3D-RETR* then obtains the 3D voxel features by using another Transformer Decoder. The *3D-RETR* model from consists of three main components: a Transformer Encoder, a Transformer Decoder, and a CNN Decoder. The Transformer Encoder takes as input the images, which are subsequently encoded into fixed-size image feature vectors. Then, the Transformer Decoder obtains voxel features by cross-attending to the image features. Finally, the CNN Decoder decodes 3D object representations from the voxel features.

The Vision Transformer uses image x_i as input and splits the image into B^2 patches. The corresponding patch is embedded at each time step by first transforming it into a fixed-size vector, which is then added to positional embeddings. In the Decoder, the M^3 learned positional embeddings are used as inputs, and the Transformer Encoder outputs are cross-attended by the Transformer Decoder. This decoder decodes all input vectors in parallel, rather than auto-regressively.

The Transformer Decoder produces voxel features that are fed into the CNN Decoder, which generates voxel features. The model uses the ShapeNet and Pix3 datasets to evaluate the model using the Intersection over Union (IoU), providing satisfactory results. The *3D-RETR* model used in is more efficient than previous models, as *3D-RETR* achieves better performance with significantly fewer parameters. The *3D-RETR* can be further improved by using additional transformers such as Performer, Reformer, etc.

References

[1] P. H. S. Torr and A. Zisserman, "Feature Based Methods for Structure and Motion Estimation," in *Vision Algorithms'99, Lecture Notes in Computer Science (LNCS 1883)*, Springer-Verlag, 2000, pp. 278–294.

[2] M. Havlena and K. Schindler, "VocMatch: Efficient Multiview Correspondence for Structure from Motion," in *European Conference on Computer Vision (ECCV)*, vol. 8691, Springer, 2014, pp. 46–60.

[3] S. Bianco, G. Ciocca, and D. Marelli, "Evaluating the Performance of Structure from Motion Pipelines," *Journal of Imaging*, vol. 4, no. 8, p. 98, MDPI, 2018.

[4] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 4104–4113.