SfM 3D Reconstruction Images

Presented by:

Ali Khayat
Biruk Abere
Felipe Espinosa

INTRODUCTION

- **3D reconstruction** is widely used in autonomous driving, AR/VR, robotics, etc...
- Traditional methods rely on expensive sensors (LiDAR, structured light, depth cameras)
- Recent advances in **Computer Vision** allow 3D modelling using only images
- How can we **reconstruct accurate 3D scenes** using only a **monocular RGB camera**, without relying on depth sensors?
- Develop a **Structure from Motion (SfM)** pipeline that estimates **camera poses and 3D structure** from 2D images

MOTIVATION

- **SfM is a cost-effective alternative**, eliminating the need for specialized hardware.
- **More flexible than depth sensors**, as it works in various lighting and environmental conditions.
- **Challenges in SfM include** scale ambiguity, feature matching errors, and high computational complexity.
- **This work addresses these challenges** through **robust feature detection**, **pose estimation**, and **optimized triangulation**.

RELATED WORK

- “Feature Based Methods for Structure and Motion Estimation,” by P. H. S. Torr and A. Zisserman,

- “Evaluating the Performance of Structure from Motion Pipelines” by S. Bianco, G. Ciocca, and D. Marelli.

- “Structure-from-Motion Revisited,” by J. L. Sch¨onberger and J.-M. Frahm

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.

- Data collection and preprocessing
- Camera intrinsic setup
- Feature extraction and matching
- Relative pose estimation
- Triangulation - based 3D point cloud construction
- Iterative camera pose refinement with perspective n-Point registration
- Point cloud update and final generation

Data collection and Preprocessing

- Our dataset comprises images captured in common formats (e.g JPEG, PNG) collected from multiple sources.
- To reduce computational load and memory usage, the images are downsampled using OpenCV’s pyramid downsampling method

Camera calibration setup

- The intrinsic matrix K, shows how a 3D point in the camera's coordinate system is projected onto the 2D image plane of the camera
- The intrinsic matrix contains important parameters, such as the camera's focal length and principal point.
- If the intrinsic parameters are not known, a default K matrix can be computed based on the dimensions of the first image
- The focal lengths are assumed same in both horizontal and vertical directions (fx = fy = w, where w is the width of the image)
- The principal point (center of the image) is set at cx = w/2, cy = h/2
- If the image is downsampled, the focal lengths in pixel coordinates also change, so to maintain geometric relationship we will divide them by downsampling factor
- If the camera intrinsic parameters are not adjusted during image resizing it could lead to errors in subsequent calculations, such as pose estimation and 3D reconstruction

Feature Extraction and Matching

- We use the Scale invariant feature transform (SIFT) due to its invariance to scale, rotation, and illumination variation.
- SIFT detects keypoints and compute 128 dimensional descriptors that uniquely characterize local image patches.
- To match the extracted descriptors between images, we used a FLANN-based (Fast library for approximate nearest neighbors) KNN matcher (k = 2)
- The KNN maps features from two of the images that are close in the Euclidean space, so that we can find which point in Image A corresponds to which point in the image B.
- Following the matching, Low 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

Essential Matrix and Recover Pose

- When we compute the Essential Matrix E, it gives us a mathematical relationship between the two images taken from different views of the same scene.

$$x_2^T E x_1 = 0$$

  - X1 and x2 are corresponding points in the first and second image
  - Essential matrix describes the geometric relationship, it's what connects the two views together mathematically

Recover Pose & Projection Matrix

- To get the actual camera movement, we decomposed E into rotation and translation matrices
  - Rotation R, how the camera rotated
  - Translation t, how the camera shifted

So this tell us where the second camera is positioned relative to the first

Projection matrix for the second camera

- Once we have R and t, we can define the second's camera's projection matrix, which describes how 3D points in space project onto the second image
- Give a 3D point, here's what it lands in the 2D image
- Each camera or each pixel in the camera has a projection matrix

Triangulation and initial 3D point cloud construction

- Imagine we have two cameras looking at the same object point. We can draw a line for each camera's position to that object point. These two lines form a triangle that goes between the two cameras. Where the two lines cross in space is the object's actual 3D location, hence triangulation.
- So we plug the matched 2D points and the camera projection matrices into a mathematical formula (opencv triangulatepoints()), which tries to find the 3D coordinates (X, Y, Z) that would project to those 2D points in both images
- Once we triangulate many such points, we build up a 3D point cloud representing the scene.

Iterative PNP Registration and Point cloud update

- Iterative integration of new images to refine the 3D model
- **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.
- **Three-view correspondence :-** We compare key points from the previous image pair with those from the new image pair
    - It figures out which points are the same (so we can use them to estimate how the camera moved) and which points are new (so we can add them to our 3D scene later)

PNP Registration

- **PNP - Based pose estimation :-** With the common keypoints (the ones that also exist in the previous images), we already know their 3D positions from the last step of reconstruction
- Use old 3D points + new 2D matches -> figure out the camera's new position (its rotation and translation to match 3D to 2D)

PNP Registration

- **New point triangulation :-** Key points 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.
- **Update variables** :- Finally we update the projection matrices, key points, 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.

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 PLY file header is generated to conform with the standard format, specifying the number of vertices and their properties (coordinates and color channels).

EVALUATION

To evaluate the performance of our SfM pipeline we use two different approaches:

1. Quantitative results:
- Reprojection error
2. Qualitative results:
- Effect of dataset size or number of samples
- Consistency in image acquisition
- Impact of camera intrinsic parameters

DATASETS

Castle:

Two version

1. 19 photos
2. 30 photos

Entry: 10 photos

Fountain: 11 photos

GustavIIAdolf: 57 photos

Herz: 8 photos

QUANTITATIVE EVALUATION

Initial two view reconstruction

For our test, the initial error range is from 0.003 to 0.007 pixels.

This represent a coherent baseline reconstruction

Castle 19:

```
Reprojection Error after 2 images: 0.0037
```

Castle 30:

```
Reprojection Error after 2 images: 0.0067
```

Entry:

```
Reprojection Error after 2 images: 0.0023
```

Fountain:

```
Reprojection Error after 2 images: 0.003
```

Gustav:

```
Reprojection Error after 2 images: 0.0038
```

Herz:

```
Reprojection Error after 2 images: 0.0048
```

Incremental updates

The reprojection error is updated with each new image, giving insight into how well the expanding reconstruction remains consistent with incoming views.

Example: using the castle-P19

- 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.
- ***After 14 images: 127.6947 pixels***

Key Findings

- Calibration Impact
- Incremental Stability
- Error Spikes and Outliers

QUALITATIVE EVALUATION

we analyze the reconstructed 3D point clouds focusing on three main factors:

1. Effect of Dataset Size
2. Consistency in Image Acquisition
3. Impact of Camera Intrinsic Parameters

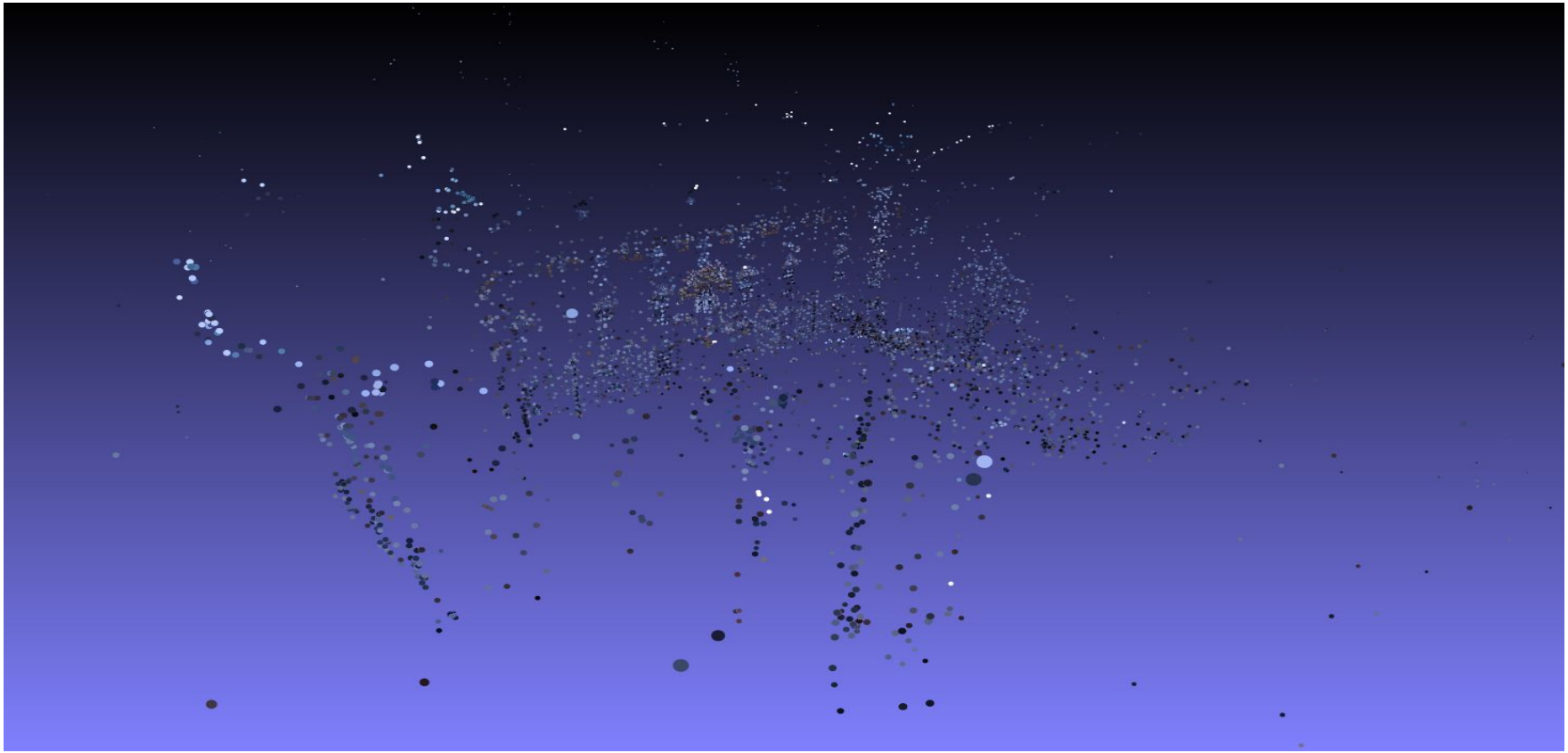
Effects of dataset size

- We experimented with the two different configurations of the castle dataset

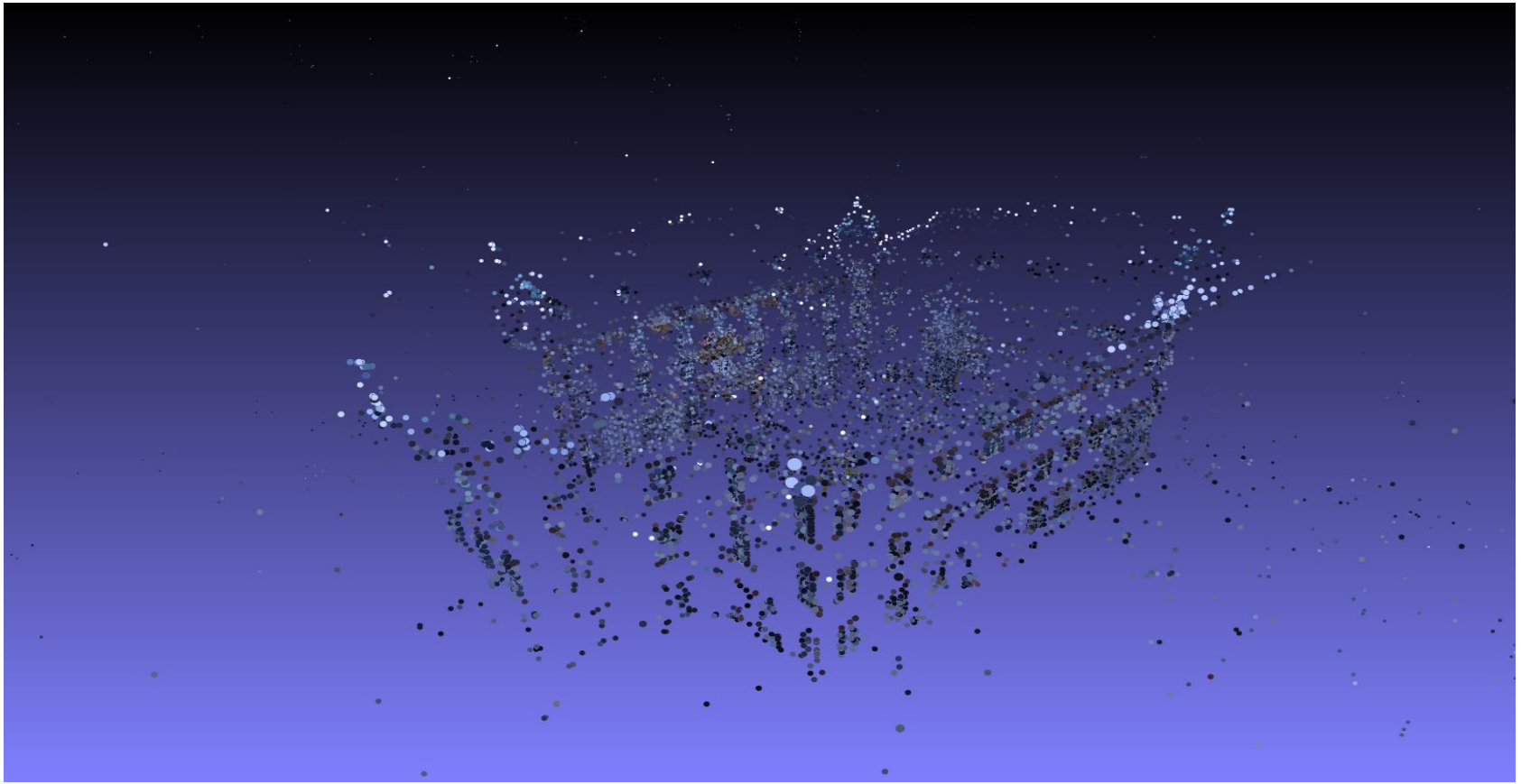
- The increased number of viewpoints provides more overlapping regions and keypoints

- Increases the computational cost

Castle 19



Castle 30



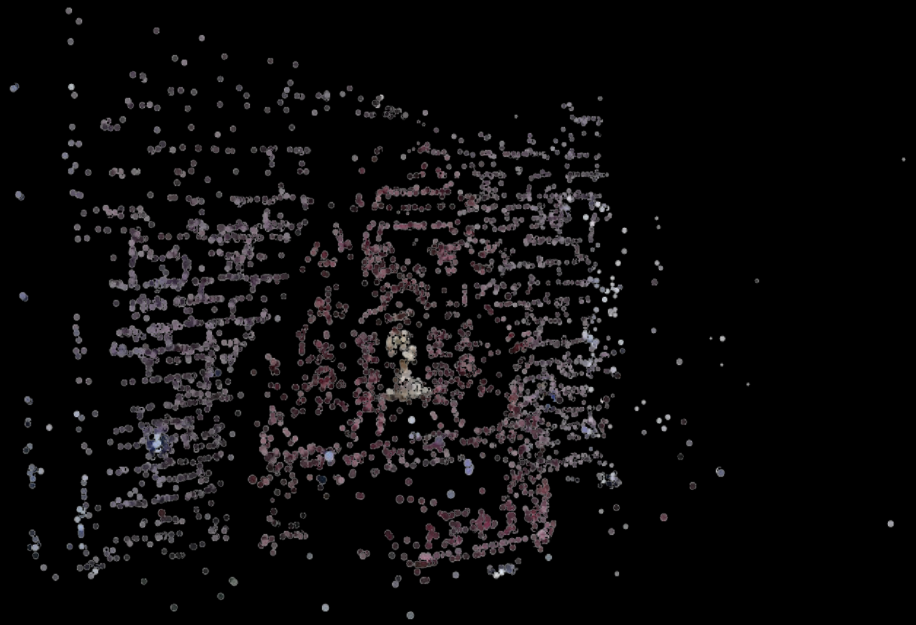
Consistency

- Reconstruction quality improve with consistency and smooth transition between consecutive images

- Ordered images provide robust feature matching

- Abrupt changes in viewpoint or insufficient overlap often lead to mismatches

Fountain



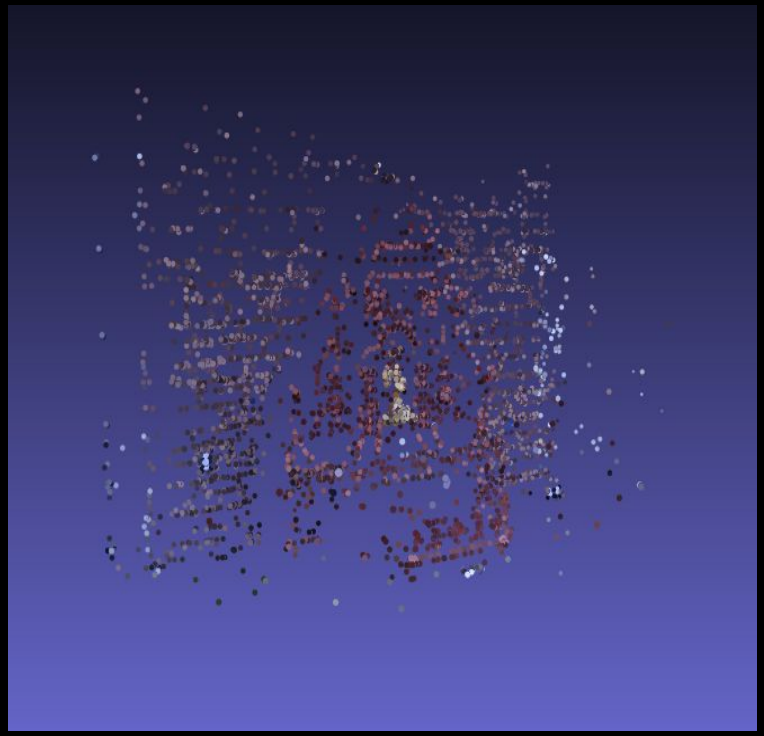
Camera Intrinsic Parameters

- The use of known camera intrinsics improves the accuracy of the reconstruction.

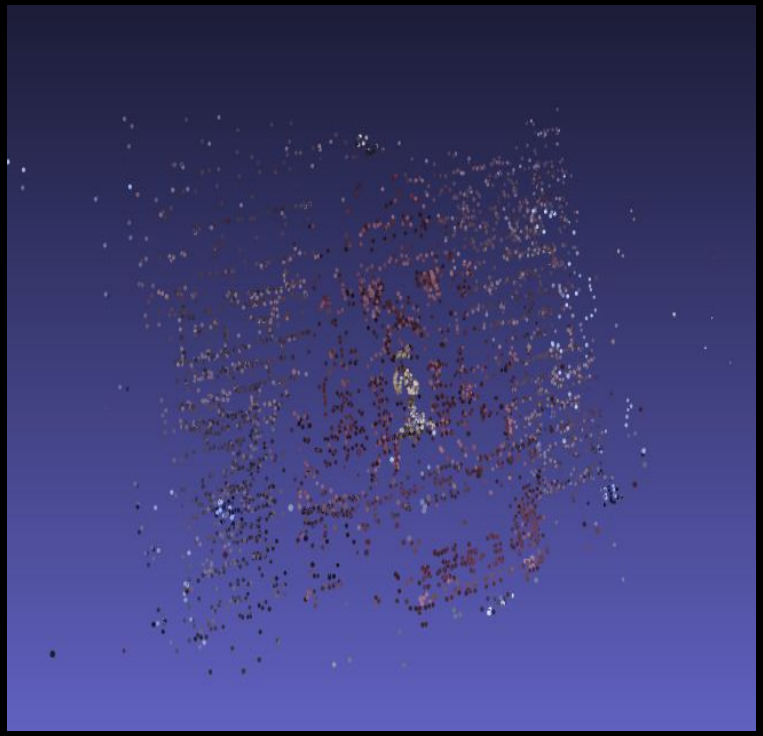
- In some cases, relying on default or estimated intrinsics can introduce scale and alignment uncertainties

- Slightly reduce error reprojection spikes

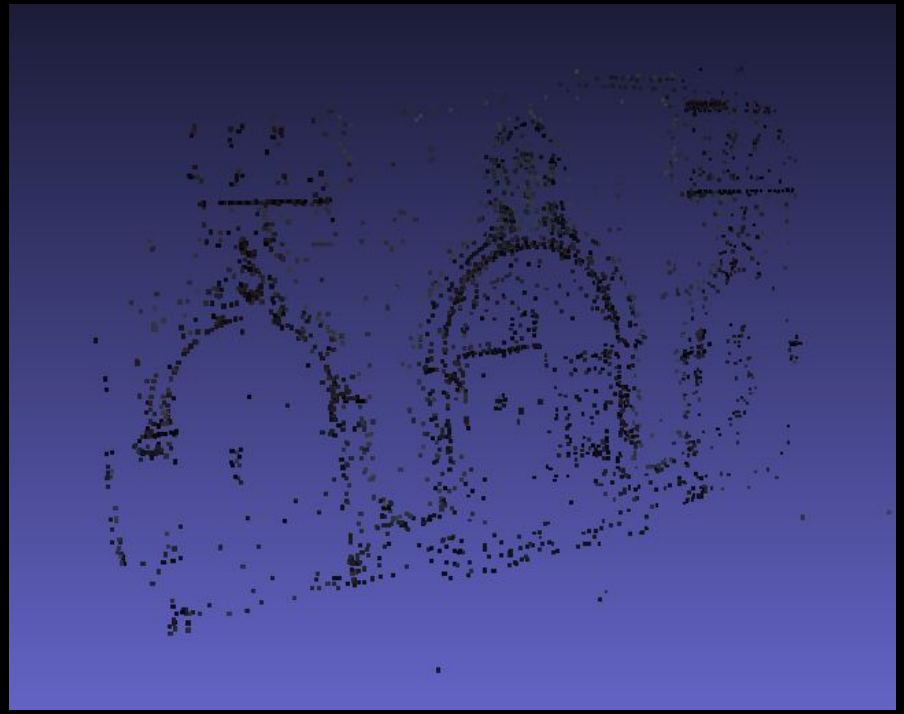
Camera K



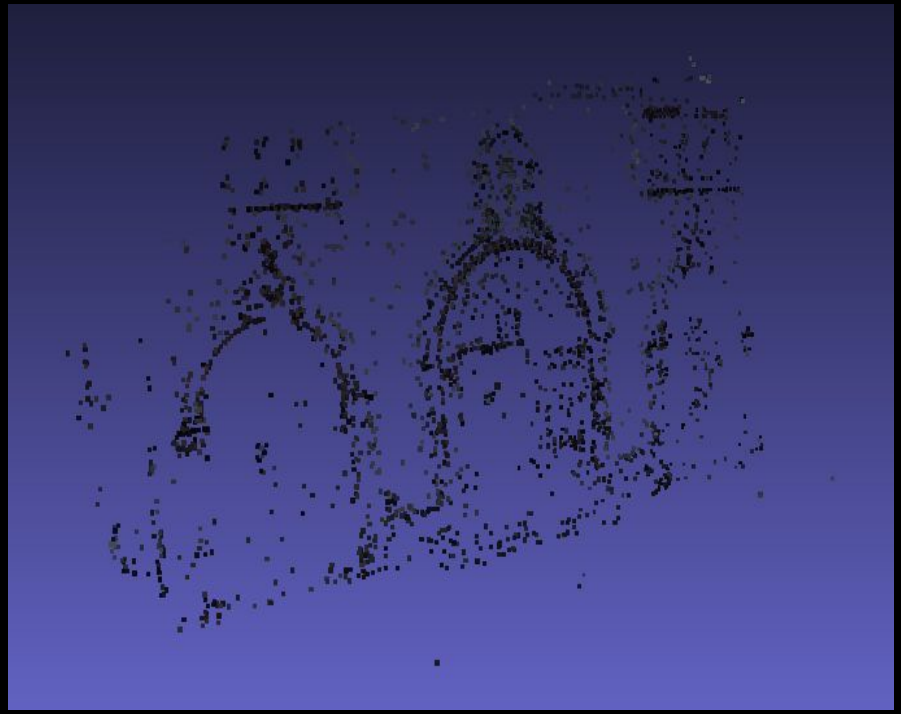
Default K



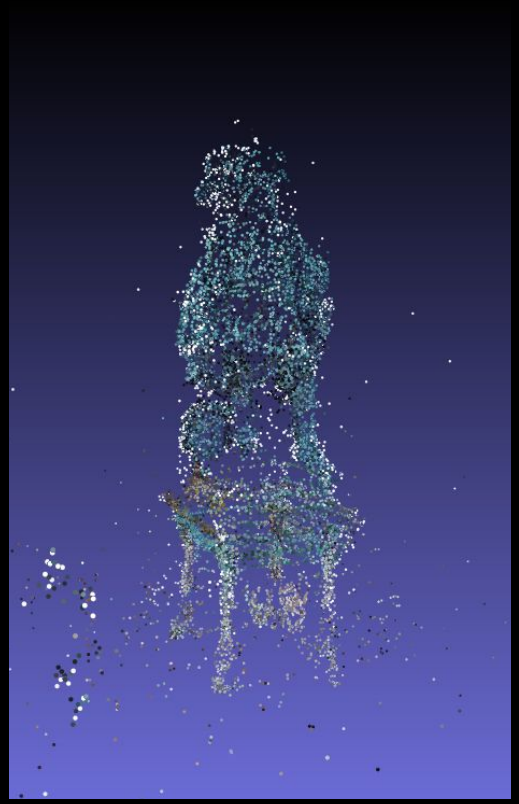
Camera K



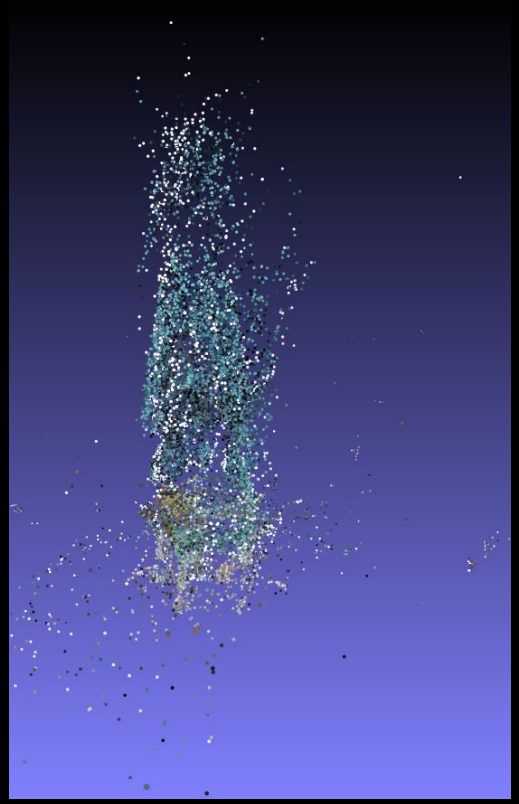
Default K

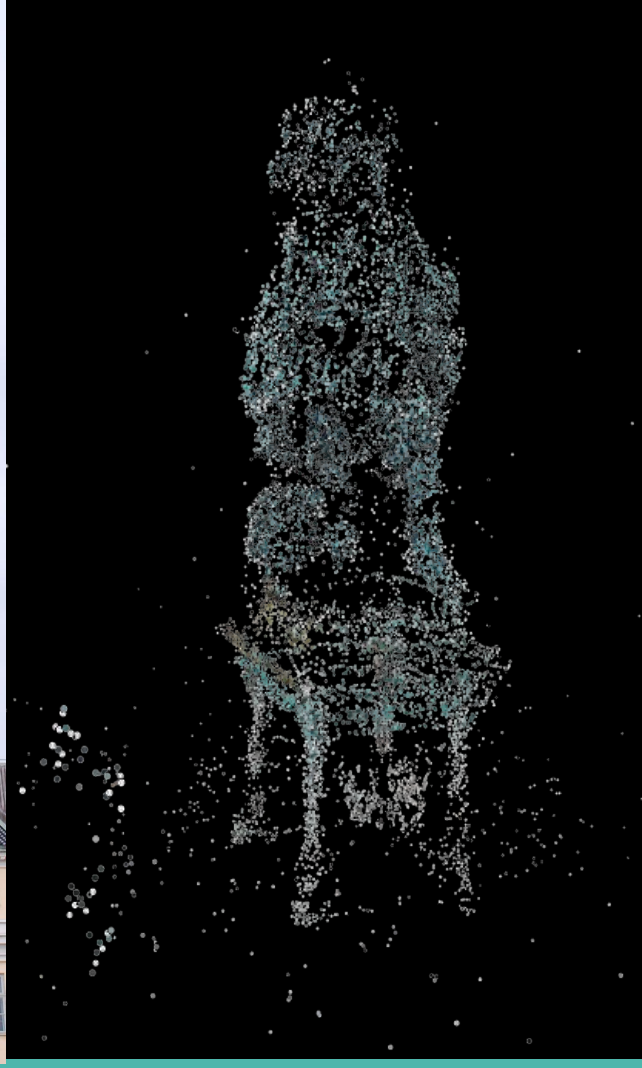


Camera K



Default K





Best results:

- A sufficient number of images
- Consistent overlap
- Accurate camera intrinsics

CONCLUSIONS

- This work successfully implemented an SfM pipeline that reconstructs 3D models from 2D images.
- **Feature-based techniques** (SIFT, FLANN) and robust estimation methods (RANSAC) were used to improve **camera pose estimation and structure recovery**.
- The approach demonstrated that **accurate 3D reconstructions can be achieved without depth sensors**, making it an efficient alternative.
- The quality of reconstruction **depends on the number of images, feature matching accuracy, and optimization techniques**.

FUTURE WORK

- Early works for 3D reconstruction mostly use feature matching between different views of an object. But, 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.

FUTURE WORK

FUTURE WORK

The model uses the ShapeNet and Pix3D datasets to evaluate the 3D-RETR

improved by using additional transformers such as Performer,

Reformer, etc.

THANKS