Scene Reconstruction with Minimal Data Transfer

Sarah Shuhaibar
Stanford CS 231A
Winter 2022
saraht2@stanford.edu

Abstract

Transferring large files such as videos can be difficult or impossible, especially in low bandwidth environments. In this paper we implement a prior work to explore how information from a video can be extracted to reduce the file size and allow for scene reconstruction after transmission. We simulate a drone flown around a single building to collect a ground truth video, and then use a Structure-from-Motion (SfM) pipeline with plane segmentation and texture mapping to virtually reconstruct the scene.

1. Introduction

When a scene is captured in a video, the resulting video file will be large and may be challenging to transmit or store. To address the need to share information about an environment along with low storage or low-bandwidth limitations, information about the scene can be extracted from the video and transmitted in its place. In this paper we explore how to extract and clean a small amount of information that will still allow us to reconstruct the original scene.

The specific use case for this paper is a drone flying around a single building. As the drone circles around the scene it records a video that provides valuable information about the building's structure and facades. We use frames from the video in a Structure-from-Motion (SfM) pipeline to create a point cloud that is segmented into planes, producing plane vertices that are then used to recreate a mesh of the building. Texture from the video frames is applied to the mesh planes, and finally we take a video of the reconstructed scene to compare to the original.

2. Background and Related Work

This project was inspired by C.M. Arnold's 2019 Master's thesis at the Air Force Institute of Technology [2]. The author addressed the trade-off between sending high quality data and being able to transport the data quickly by proposing a SfM method to compress high-resolution data and deliver it in real time. The author achieved more than a 90% reduction of the original video size.

Scene	Dataset	Trad. SfM	Imprvmnt	SfM w/ PHE	Imprvmnt
Tall Building	178 MB	13.6 MB	92.4%	2.53 MB	98.6%
City Block	178 MB	26.9 MB	84.9%	9.29 MB	94.8%
Arab House	178 MB	9.47 MB	94.7%	11.8 MB	93.4%

Figure 1. Arnold (2019) Storage Size Comparison

There are several steps to the process, one of which is plane segmentation of point clouds. A. Honti et al [4] proposed a RANSAC-based algorithm to improve the efficiency of plane estimation. While the open3D library performed sufficiently well for plane segmentation in this case, an alternative algorithm may be applicable in cases where sped up performance (real time data processing) is a requirement.

3. Approach

This section describes the approach to extract building information from the video and reconstruct it as a textured mesh. The source code is available at `https://github.com/saraht2/CS231AFinalProject/`.

3.1. Data Collection

We create the scene by placing a textured building model in Blender and rendering an animation with the camera circling 360° around the building. The video output is the ground truth that we will use to compare to our reconstructed results.

The initial building model selected for this paper [3] has three identical facades, which resulted in an incomplete reconstruction from the SfM pipeline because there is no way to distinguish between the facades in the images. In order to use this method on such a building, there would need to be some form of marking the facades prior to the SfM pipeline in order to build a complete reconstruction. Instead, in its place we use a model that has four unique facades in order to differentiate between the sides of the building [1]. We

render a uniform 360° animation of the camera circling the building in Blender.

3.2. Video sampling and sparse point cloud construction

The rendered animation video is sampled once every five frames to create a set of photos with sufficient overlap between the frames to identify common points between the images (Figure 2). We use these frames to create a sparse reconstruction point cloud of the building using COLMAP's SfM pipeline (Figure 3). For this simple scene a sparse reconstruction is sufficient to understand the building structure, but in a more complex scene with adjacent buildings or curved walls a dense reconstruction may be necessary.

Figure 2. Sampled video frames

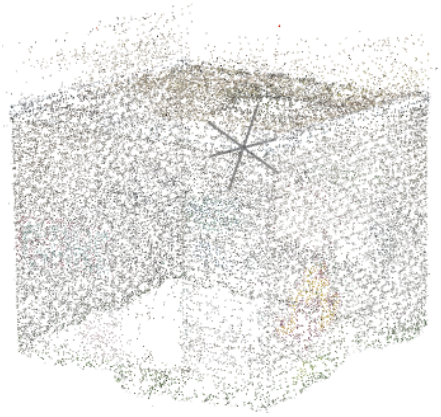

Figure 3. Sparse point cloud reconstruction

3.3. Plane Segmentation

Next we load the point cloud into Python for plane segmentation and cleanup. Using the open3D library, Algorithm 1 loops over the point cloud to segment out five planes, with the assumption that the building has four flat walls and one flat roof.

Algorithm 1 Plane Segmentation

```
remainingPointCloud = pointCloud
numPlanes = 5
planes = {}          ▷ Individual plane point clouds
for i = 0; i < numPlanes; i++ do
   inliers = segment points from rpcd
using RANSAC
   planes[i] = inliers
   remainingPointCloud =
remainingPointCloud - inliers
```

The RANSAC parameters for the plane segmentation are:

```
probability = 0.99
outlier_proportion = 0.93
distance_threshold = 0.01
s = 3
```

The probability threshold is set high so that for each plane we have a good chance of finding a sample without outliers. Our outlier proportion is also very high, because on our first iteration we are segmenting out one plane from five, meaning that even with no building outliers 80% of the initial point cloud does not belong to the first plane. On inspection, around 13% of the points in the original cloud do not end up belonging to any of our final building planes, so our outlier proportion is set to 93%. A distance threshold of 0.01 produces planes that do not have too much thickness, and finally three points are required to define a plane.

This initial algorithm produces a segmented point cloud that includes outliers visually far from the building in space. While these points fall on the same planes as either a wall or the roof, they are noise that should be removed. To remove these points, we define a threshold for how close each point should be to a number of its neighbors compared to the average of the points in the plane, and remove those that fall outside of the threshold.

In addition to these outliers, the initial point cloud also includes walls that extend beyond the top of the roof (purple plane in Figure 4). To address these points, we first must rotate the entire point cloud to make the roof parallel to the X-Z plane, so that the maximum Y-coordinate of the roof's points can be used to impose a maximum height on the walls.

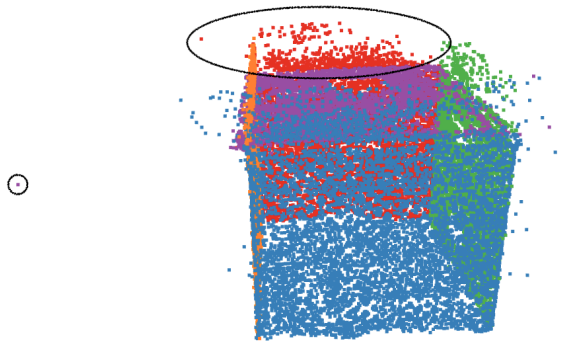

Figure 4. Initial segmented point cloud, with outliers and noisy walls

3.4. Defining the roof plane

Because we are assuming the building has four walls and one roof, we can use the difference between the minimum and maximum values in each coordinate direction for each plane to identify the roof. For example, in Figure 5 the blue plane has a relatively small difference between its minimum and maximum Z values, compared to its span in the X and Y directions.

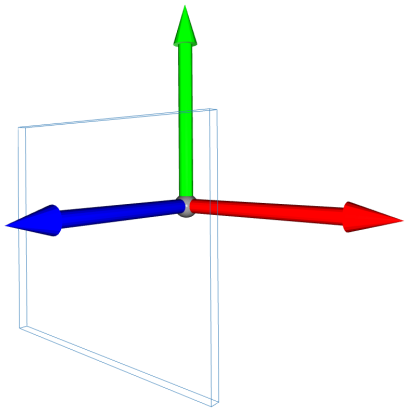

Figure 5. A plane with a small span in the Z direction

Given the structure of the building, each of the walls will have a "partner" plane with a similar span. The roof will be the only plane with a unique span composition, allowing us to identify it by calculating the min-max difference in each direction for each plane (Algorithm 2).

Once the roof is identified, we use the scipy library to calculate the rotation matrix between the normal of the roof plane and the Y-axis. This rotation is applied to the entire building point cloud. The maximum height of the roof is then imposed as a restriction on the height of all of the walls.

Algorithm 2 Find Building Roof

```
numPlanes = 5

for i = 0; i < numPlanes; i++ do
    Calculate:
    plane[i]xmax - xmin
    plane[i]ymax - ymin
    plane[i]zmax - zmin
    Determine in which coordinate direction the plane has the smallest max - min value
     Determine which coordinate direction has a single plane with a small max - min value and which plane it corresponds to; this is the roof.
```

3.5. Final building alignment

At this point we have our final building point cloud, without extreme outliers and noise along the tops of the walls. To simplify the bounding boxes, we transform the point cloud to the world coordinate frame by:

1. Transforming by the inverse of the rotation matrix applied earlier
2. Translating to the center of the world frame
3. Re-rotating by our earlier rotation matrix
4. Performing a final rotation with a new matrix, constructed from the angle between a wall normal and the X-axis

From this final aligned building point cloud, we have two options for how we represent the structure of the building. The first is to calculate the absolute min and max in the X, Y, and Z directions and use those values to construct the vertices of the building. This represents a very small amount of data that is needed to define the building's edges.

Our second option is to construct bounding boxes around each of the planes. While this comes with a slightly larger data requirement (each bounding box requires eight vertices), it reduces the impact of noise along the edges of walls on the location of adjacent walls.

3.6. Create building mesh

Now that we have the parameters for our building, we can render a building mesh. Blender was used for this paper, and the built-in Python template to add a mesh was modified to add the building's vertices. The mesh is a simple cube.

3.7. Extract building's texture

The second piece of information we need to recreate the building is the texture of the facades. We can get this from the video frames that were used in the SfM pipeline. For

each of the five building planes, we select a frame where that plane is visible and manually locate the $N = 4$ corners in the image. Each facade's texture needs to be transformed so that it is flat and square, which we do by calculating a 3x3 homography matrix.

The homography matrix is the least squares solution to $Ab = 0$, where A is a $2N$ x 8 matrix and b is a $2N$ vector. For each manually-selected point sx, sy in the video frame, the two corresponding rows of A are:

$$\begin{bmatrix} sx & sy & 1 & 0 & 0 & 0 & -dx * sx & -dx * sy \\ 0 & 0 & 0 & sx & sy & 1 & -dy * sx & -dy * sy \end{bmatrix} \quad (1)$$

where dx and dy are the x and y coordinates of the destination coordinate space.

After applying a homography transformation to each of the building facades, we have a texture that can be applied to the mesh (Figure 6).

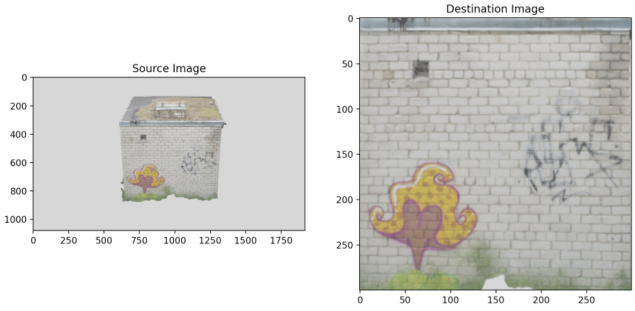

Figure 6. Facade transformed to texture

3.8. Reconstructed video

The final step in the process is to render a video of the reconstructed building. Because we know the original camera parameters, we can render the new video with the same camera settings as the original. However, we cannot recover the scale of the original structure. The reconstructed mesh was uniformly scaled to match the footprint of the original building for comparison purposes.

4. Experiment

The goal of this project is to share information about a scene without sending a full video file. There are three measures we use to evaluate the final results:

1. Size of the original file compared to the information needed to create the reconstructed scene
2. A visual comparison of the original and reconstructed videos
3. A quantitative comparison of video frames from the original and reconstructed videos

4.1. File size comparison

The original video output from the Blender animation render is 195.9 MB. This file size is our baseline against which we will compare a reduced set of files that can be extracted and used to reconstruct the scene.

There are two main components for scene reconstruction: the building shape and texture. The building's shape is found from the segmented and cleaned point cloud, and can be represented in one of two ways: as eight vertices (178 bytes), or as the bounding boxes of the five planes (516 bytes). For the purpose of comparison the smaller file size is used; however individual plane bounding boxes may give a more precise representation of the building. Both options can be saved and transmitted in a simple text file.

The building's texture is stored as five images that are the homography transform of the original images of the walls and roof. The largest single texture image file produced for this project is 159 KB, and the total size of the five files is 767 KB. There was no attempt to compress these image files as the results were already much smaller than the original video file, but lossless compression could be incorporated here as well.

In addition to the main building information, in practice we would need to share supplementary information for correct scene reconstruction. This includes camera intrinsics of the original camera, as well as a mapping of the building's planes to the texture images. We estimate the size of this information stored in a text file to be 500 bytes.

Table 1: Original Video

File	Size
Original Video File	195.9 MB
Total Size: 195.9 MB	

Table 2: Reconstruction Files

File	Size
Building Structure	178 bytes
Texture Files	767 KB
Supplemental Data	500 bytes
Total Size: 767.7 KB	

Size of reconstruction files compared to original video: 0.39%.

4.2. Visual video comparison

The second assessment criteria is a visual comparison of the original and reconstructed video. The original and reconstructed videos are included as supplemental materials at this link: `https:`

//drive.google.com/drive/u/3/folders/1S2rCoEjy3FXd176bpGTtLZKqvVJZusDL. Individual frames are included here for comparison.

Visually, we can see that the two buildings are similar but not identical (Figure 7). The reconstructed building is taller and the colors are faded. While we can tell that this is the same structure, it's an issue for the original goal of gaining information about the scene. This flawed representation of the building's area may not be a close enough representation for viewers who need more precision.

Figure 7. Original (top) and reconstructed (bottom) buildings

4.3. Quantitative frame comparison

Because the reconstructed building is taller than the original a pixel-to-pixel comparison of video frames will not be a good comparison metric. Instead we use two image comparison metrics: a Structural Similarity Index Measure (SSIM) and a Feature-based Similarity Index (FSIM), both of which range between 0 and 1. With SSIM image degradation is considered as the change of perception in structural information, while FSIM maps the features and measures the similarities between two images [5].

We use the same frame from each video for comparison, and get these results:

Table 3: Video Comparison Metrics

Metric	Score
Structural Similarity Index Measure	0.9906
Feature-based Similarity Index	0.3139

Both metrics are likely getting a boost from the unchanging grey background, but it is still surprising to see the SSIM measure so high. The structural information is certainly different between the two images; more exploration is needed.

5. Conclusion

In this section we discuss the quality of the results and propose future improvements.

5.1. Assessment of results

This project met one of its two goals: it succeeded in extracting a small amount of information from the original video to represent the scene, but the reconstructed building is not at the desired quality level. The lower quality of the facade images is due to both manually selecting corners as well as the images getting stretched to cover the taller building.

A theory for the increased building height is that there may be noise at the bottom of the point cloud, similar to the walls that were taller than the roof, that went unnoticed during data processing. There are a few ways to address this:

1. **Add a ground plane to the scene.** The simplest way to address this issue is to add a ground plane to the original scene and impose a Y-coordinate minimum on the point cloud.

2. **Impose edge ratios.** If the ground plane is not visible, we could calculate ratios for the original building's edges and impose those ratios on the point cloud.

One way to explore whether noise along the bottom of the point cloud is actually the cause of the extended height is to place multiple buildings of varying heights in the scene and explore whether the distance between their roofs hold, or if the entire point cloud is stretched in the vertical direction. This would also require more advance point cloud cleaning and texture mapping.

5.2. Comparison to prior work

Arnold achieved a smaller compression ratio, but significantly better reconstruction results. The suggestions for implementation details that fall short in this paper compared to Arnold echo conclusions discussed elsewhere:

1. Impose a ground plane in the scene to reduce vertical stretching of the point cloud.

2. Use keypoints from the SfM pipeline to crop and match textures onto the building planes.

One aspect of implementing this approach in the real world that is not addressed in this paper is real-time processing. Arnold's goal to transmit high-quality data in real time in the real world requires a robust pipeline to instantly handle multiple shapes and potential noise, while this approach

involves manual file review and data transfers. Building a complete pipeline to collect and process streaming video data would be a substantial future expansion.

5.3. Next steps and improvements

In this subsection, we discuss four ways to improve and expand upon this project's implementation.

5.3.1 Improve Texture

One of the more disappointing aspects of this project is the low quality texture mapping that requires us to pick images that correspond to each of the facades and manually select the building's corners in each of those images. This is a bottleneck in the process that opens the door to a number of errors. Making use of the COLMAP's keypoint outputs may be an avenue for improved texture mapping, to automatically match texture coordinates to plane coordinates.

5.3.2 Scaling

Another issue that was not addressed is the scale of the scene. Although we cannot recover scale from a SfM pipeline, this information may be important to a viewer. One way to address this with the information already available is to use average estimates of known quantities, such as the distance between building floors or the height of doorways. Without exact measurements, this would only give an estimate of a building's size. Another option is to incorporate more real-world information in the scene, such as human beings, cars, or lampposts.

5.3.3 Dense Reconstruction

Because this was a simple scene reconstruction, a sparse point cloud was sufficient to recover the building's structure. In more complex scenes with multiple adjacent building, curved facades, and obstacles causing occlusions, a dense point cloud would likely be a better option. Generating a dense point cloud in COLMAP also has the advantage of producing depth and normal maps, which could help with texture mapping.

5.3.4 Shadow Carving

One final area for improvement is shadow carving. While the original building only had a door in the texture (but not in the mesh), no attempt was made to explore creating a more complex mesh with doorways or windows. In real world scenarios, shadow carving could assist with removing alleyways between building from the scene mesh.

References

[1] Abandoned Scans. "Brick Building free" (https://skfb.ly/6XRTD). Licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/).

[2] C. M. Arnold. High resolution low-bandwidth real-time reconnaissance using structure from motion with planar homography estimation. Master's thesis, 2019.

[3] LowlyPoly. "Building Apartment 13" (https://skfb.ly/6T9wO). Licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/).

[4] R. Honti, A. Kopacik, and J Erdelyi. Plane segmentation from point clouds. Article in Pollack Periodica, 2018.

[5] U. Sara, M. Akter, M. S. Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. Journal of Computer and Communications, 2019.