Multi-View Thin Structure Reconstruction via 3D Lines and Points (#12)

Montiel Abello [mabello] | Sudharshan Suresh [sudhars1]

### Introduction and Motivation

- Scenes with thin structures and edges are poorly reconstructed by multi-view stereo algorithms, due to appearance variation.
- We propose the integration of line segments into the SfM point-cloud via a radial sampling technique.
- We then perform dense reconstruction and mesh texturing on this combined point-cloud.
- Results highlight the recovery of thin structures, along with enhanced visibility of edges.

### Pipeline

![Pipeline Diagram]

**Figure 1.** Incorporating thin structure modeling into a multi-view reconstruction pipeline

### Global SfM

- To generate a sparse model of the 3D structure of the scene, we use the SfM library from openMVG.
- Given an unordered set of images with camera intrinsics specified, openMVG estimates the global pose \( P_i \) of each camera, and global position of features points \( X^T \).
- openMVG optimizes the following cost function:

\[
\min_{x_1, \ldots, x_N} \sum_{j=1}^{M} \sum_{i=1}^{N} \left( x_i^T - K_i P_i x_j \right)^2
\]

where \( x_i^j \) is the image coordinate of the \( j \)th feature point in the \( i \)th camera.

**Figure 2.** Image of scene and optimized sparse 3D points

### Line-based 3D Model

- From the SfM output, we get the ordered image sequence sparse point cloud and visibility information.
  \[
  I = \{I_1, \ldots, I_N\}, \quad X = \{X_1, \ldots, X_K\}
  \]
- We use Line3D++, a pipeline to abstract 3D scene information using line segments.
- We run a line segment detector on every image, establish correspondences across images with supporting views.

\[
S_{f}(i,j) = \frac{2|X(i)\cdot X(j)|}{|X(i)| + |X(j)|}
\]

- Generate 3D hypothesis based on plausible correspondences; cluster the 2D segments based on them.
- Refine estimate to get final 3D line model.

**Figure 3.** Optimized 3D line segments

### Thin Structure Sampling Strategy

- For each line segment represented by points \( P_1 \) and \( P_2 \), we sample \( N = \text{round} (mL) \) points on the cylinder of radius \( r \) around the segment, where \( L = \|P_1 - P_2\|_2 \) and \( m \) is a scaling factor.
- We do this by sampling \( N \) angles \( \theta_i \in [-\pi, \pi] \) and \( N \) lengths \( s_i \in [0, L] \) and computing the corresponding 3D points.

**Figure 4.** Points sampled around line segment

### Dense Reconstruction

We use openMVS to convert our sparse point cloud to a dense 3D mesh. The components of the pipeline are:

- Densify the sparse point cloud with patch match algorithm
- Use visibility information in a graph-cut algorithm to reconstruct a mesh from the densified points
- Refine the mesh by subdividing triangles based on size and visibility information
- Texture the mesh using the original images

**Figure 5.** Optimized 3D line segments

### Results

We evaluate our method on our nsh_a_floor dataset. The reconstructions recover many structures and edges that are lost in the standard pipeline.

**Figure 6.** Final textured meshes from the (top) standard pipeline and (bottom) our pipeline. We improve reconstruction of thin structures - we recover the end of the object, along with a better reconstructed frame.

### Future Work

- Fitting splines to clustered line segments will better model curved surfaces
- Separate treatment of thin and regular structures will allow for optimum parameter tuning during dense reconstruction
- Evaluation on scenes with objects of varying shape and size

---

**Acknowledgement:** We thank Zimo Li for his advice with the project.