FMAN40 Project in Applied Mathematics

Piecewise Planar Reconstructions from Multiple Homographies
Project Report

Gustav Hanning Jialong Li

May 18, 2023

1 Introduction

The goal of the project is to create a piecewise planar 3D reconstruction from two or more images. First the images are processed with the Structure-from-Motion pipeline COLMAP [1] [2] to get image points, matches and accurate camera poses. Then planar regions are identified in the images by drawing polygons in a MATLAB GUI. For each region the parameters of the plane are estimated from point matches using MLESAC. The polygons are projected onto the planes and tessellated to create a triangle mesh. Next the mesh is textured by warping the images into the planes. The resulting 3D model is saved to the standard file format glTF so that it can be opened in external viewer applications. The code is available in a public GitHub repository [3].

2 Theory

This section contains the theory behind the most important concepts of the project. First a robust plane estimation method is presented. Next an algorithm for polygon triangulation is described, followed by derivation of the formulas for warping an image into a plane. Throughout the project a pinhole camera model is used.

2.1 Robust plane estimation

2.1.1 Linear equation system

Giving two cameras from two images as

$$P_1 = K\begin{bmatrix} I & 0 \end{bmatrix} \quad \text{and} \quad P_2 = K\begin{bmatrix} R & t \end{bmatrix} \tag{1}$$

Consider a plane with normal n that can be seen in both images, the 3D points of this plane can be given as $\{X \mid n^T X + 1 = 0\}$. Further, let the 2D points on image 1 that is corresponding to X as $[\bar{x}_1^i \in \bar{X}_1,\ (i = 1, 2, \ldots, N)]$, and the points from image 2 as $[\bar{x}_2^i \in \bar{X}_2,\ (i = 1, 2, \ldots, N)]$, the norm of this plane can be estimated by a linear equation system, as shown in equation (2).

$$\begin{bmatrix} t(\bar{x}_1^1)^T & \bar{x}_2^1 & 0 & 0 & \ldots & 0 \\ t(\bar{x}_1^2)^T & 0 & \bar{x}_2^2 & 0 & \ldots & 0 \\ t(\bar{x}_1^3)^T & 0 & 0 & \bar{x}_2^3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ t(\bar{x}_1^N)^T & 0 & 0 & 0 & \ldots & \bar{x}_2^N \end{bmatrix} \begin{bmatrix} n \\ \lambda^1 \\ \lambda^2 \\ \lambda^3 \\ \vdots \\ \lambda^N \end{bmatrix} = \begin{bmatrix} R\bar{x}_1^1 \\ R\bar{x}_1^2 \\ R\bar{x}_1^3 \\ \vdots \\ R\bar{x}_1^N \end{bmatrix} \tag{2}$$

Once the norm n is estimated, the homography between these two 2D plane points set can then be computed as:

$$H = K(R - tn^T)K^{-1} \tag{3}$$

In this project, the homography can be used to evaluate how accurate the estimated n is.

By using such a linear equation system, the normal of a plane can be estimated once the cameras and 2D plane points are known. However, such a method build on an assumption that all corresponding relations between points of image 1 and image 2 (such as x_1^1 to x_2^1) are correct, in other words, the correspondence is expected to have no outliers. This is, however, usually not the case. In order to make the plane estimation process more robust, we introduced the MLESAC (maximum likelihood consensus) method into this process.

2.1.2 MLESAC

The MLESAC estimator is developed based on the RANSAC (Random sample consensus) algorithm, the idea of the algorithm is to select the subset that comes with the least number of outliers among the whole data set, to maximize the *likelihood* of giving matches (between 2D points of image 1 and 2) to be the true correspondences [4].

The difference between MLESAC and RANSAC is that, compared to RANSAC, which uses the number of inliers as the scoring method, MLESAC states that the scoring method for each subset can be expressed as equation (4).

$$\rho(e^2) = \begin{cases} e^2, & e^2 < T^2 \\ T^2, & e^2 \geq T^2 \end{cases} \tag{4}$$

e is the error, T is the threshold and ρ is the scoring function. This way, inliers can also be scored on how well they fit the data. The specific implementation of this algorithm is done by coupling the linear equation system and MLESAC method together. The specific implementation of this hybrid algorithm can be outlined as algorithm 1. Note that the minimum required number of point pairs to do the estimation, denote as N_{min}, should be selected such that the number of equations is larger than the degree of freedom of the objective vector, in our case it should be $N_{min} = 3$.

Algorithm 1 Robust plane estimation

Input: 2D points set $\bar{X}_1$ and $\bar{X}_2$, number of iterations i, inlier threshold T
Output: Estimated optimal norm n^*

$Min\_cost = 0$
for number of iterations i **do**
  $cost = 0$
  Randomly select N_{min} points out of $\bar{X}_1$, also find the matched points in $\bar{X}_2$
  Solve for norm $n^{'}$, by equation system (2)
  $H^{'} = K(R - t(n^{'})^T)K^{-1}$
  $H^{'}\bar{X}_1 = \lambda\bar{X}_2^{'}$
  for Every point $x_j \in \bar{X}_2$ and $x_j^{'} \in \bar{X}_2^{'}$ **do**
    $cost = cost + min(\|x_j - x_j^{'}\|^2, T^2)$
  end for
  if $cost < Min\_cost$ **then**
    $n^* = n^{'}$
  end if
end for

2.2 Polygon triangulation

In this project planar surfaces are represented as polygons. However most file formats for 3D models require the model to be in the form of a triangle mesh. The process of subdividing a polygon into a set of triangles is called *triangulation* or *tessellation*.

One simple algorithm for polygon triangulation is *ear clipping*. It utilizes the fact that any simple polygon (i.e. without self-intersections or holes) has at least two *ears* [5]. An ear is a vertex for which the edge joining its two neighbors lies fully inside the polygon, see figure 1.

To determine if a vertex is an ear one checks if it is convex, and if there are any other vertices inside the triangle formed by the vertex and its two neighbors. The ear clipping algorithm iteratively

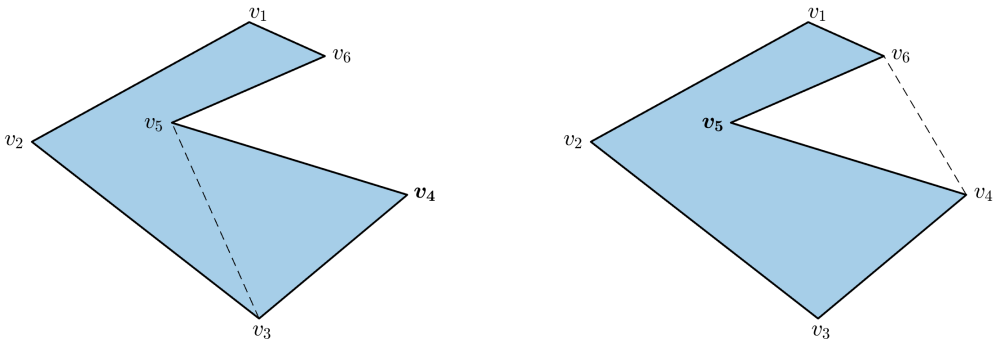
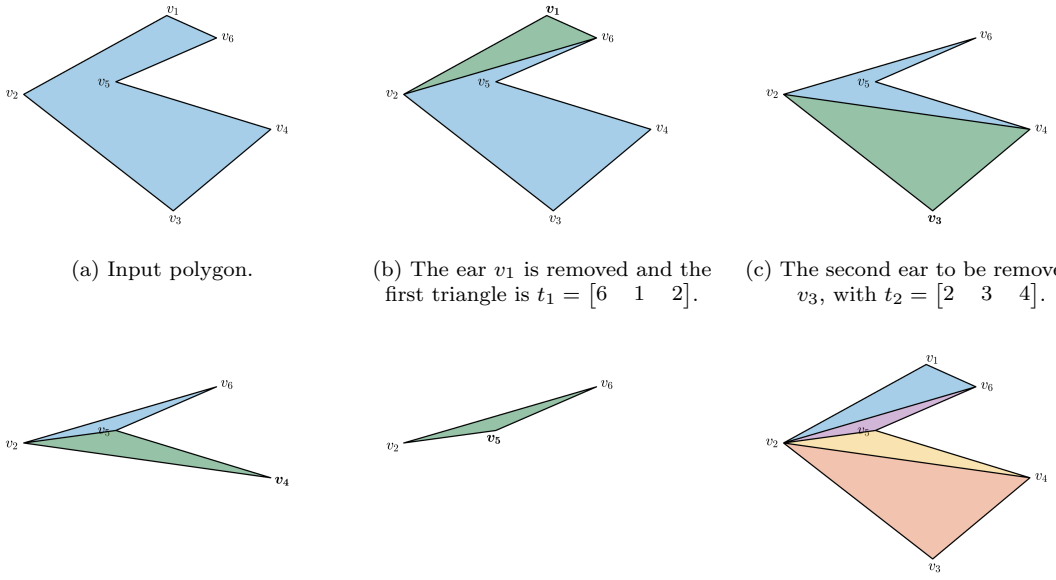


Figure 1: Example polygon where the vertex v_4 is an ear and v_5 is not.

searches for an ear among the vertices of the polygon and when finding one removes it and outputs a triangle, which is a triplet consisting of the indices of the ear and the neighboring vertices. For a polygon with N vertices the result is a list of $N-2$ triangles. Figure 2 shows an example of how the method works and it is summarized in algorithm 2.



(a) Input polygon.

(b) The ear v_1 is removed and the first triangle is $t_1 = \begin{bmatrix}6 & 1 & 2\end{bmatrix}$.

(c) The second ear to be removed is v_3, with $t_2 = \begin{bmatrix}2 & 3 & 4\end{bmatrix}$.

(d) v_4 is removed resulting in $t_3 = \begin{bmatrix}2 & 4 & 5\end{bmatrix}$.

(e) The last ear is v_5 and $t_4 = \begin{bmatrix}2 & 5 & 6\end{bmatrix}$.

(f) Final triangulation.

Figure 2: Ear clipping.

2.3 Image warping

Each planar surface is textured with the contents of one image used in the 3D reconstruction. Given a polygon $(v_i)_{i=1}^N$ defined in the pixel coordinates of this image, the camera matrix $P = K\begin{bmatrix}R & t\end{bmatrix}$ and a 3D plane with normal n such that

$$n^T X + 1 = 0 \tag{5}$$

for scene points X in the plane the goal is to warp the image into the plane. The relation between X and an image point $\boldsymbol{x}$ is given by the camera equations

$$\lambda \boldsymbol{x} = K\left(RX + t\right) = K\begin{bmatrix}R & t\end{bmatrix}\boldsymbol{X} \tag{6}$$

from which an expression for the scene point can be derived as

Algorithm 2 Ear clipping.

Input: Polygon P with vertices $v_1, \ldots, v_N$
Output: Triangles $t_1, \ldots, t_{N-2}$

```
while NUMVERT(P) ≥ 3 do
    for vertex v in P do
        if ISEAR(v) then
            output triangle [PREV(v)  v  NEXT(v)]
            remove v from P
            break
        end if
    end for
end while
```

$$X = R^T \left(\lambda K^{-1} \boldsymbol{x} - t \right). \tag{7}$$

If X lies in the plane it follows from equations (5) and (7) that

$$\lambda = \frac{n^T R^T t - 1}{n^T R^T K^{-1} \boldsymbol{x}}. \tag{8}$$

To perform the image warping a new coordinate system is defined in the 3D plane. We select the origin c to be the centroid of the polygon

$$\frac{1}{N} \sum_{i=1}^{N} v_i \tag{9}$$

projected onto the plane using equations (7) and (8). The z axis $e_z = \begin{bmatrix} e_{z,x} & e_{z,y} & e_{z,z} \end{bmatrix}^T$ of the coordinate system is parallel to the plane normal n, with direction chosen so that it points away from the camera. Assuming that either $e_{z,x}$ or $e_{z,y}$ is non-zero then the x and y axes can be selected as

$$e_x = \begin{bmatrix} e_{z,y} \\ -e_{z,x} \\ 0 \end{bmatrix} \tag{10}$$

$$e_y = e_z \times e_x \tag{11}$$

and they are subsequently scaled so that the polygon area is preserves when transformed into the new coordinate system. The global coordinates of a point $\boldsymbol{y} = \begin{bmatrix} y_x & y_y & 1 \end{bmatrix}^T$ in the 2D coordinate system $\{e_x, e_y\}$ are

$$\boldsymbol{X} = \begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} y_x e_x + y_y e_y + c \\ 1 \end{bmatrix} = \begin{bmatrix} e_x & e_y & c \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{y} \tag{12}$$

and from equations (6) and (12) it can be seen that the homography from our new coordinate system in the plane to the camera is

$$H = K \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} e_x & e_y & c \\ 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

Finally the image warping is done by looping over the pixels in the destination image, applying H to get the coordinates in the source image and sampling from it using e.g. bilinear interpolation. In figure 3 the image to the left is warped into the plane of the roof resulting in the image to the right, which is used as a texture for the final 3D model.

Figure 3: Original (left) and warped (right) image and polygon.

3 Implementation

The implementation of this project can be broken down into three components: the open-source reconstruction pipeline COLMAP, a Matlab GUI to draw polygons and a C++ program for estimating planes, triangulating polygons and warping images.

3.1 COLMAP

COLMAP is an application that is able to reconstruct a sparse model from images based on Structure-from-Motion. In this project we use this tool to extract information from given images such as the cameras, image points and matches between points. Such information is required to be able to accomplish the plane estimation.

3.2 MATLAB GUI

A MATLAB GUI has been created where each image is presented to the user and zero or more polygons can be drawn overlaid on the image, representing planar regions to be reconstructed, as shown in figure 4. The user has to make sure that polygons drawn in different images align reasonably well in 3D. When the GUI is closed a text file containing the polygons and their corresponding image filenames is written to disk.

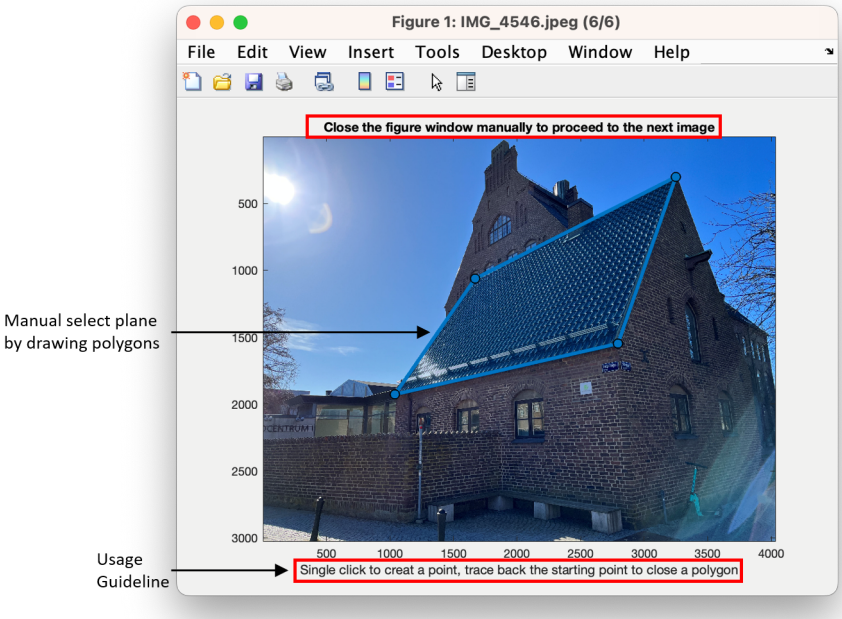


Figure 4: The MATLAB GUI.

3.3 C++ program

The remaining steps of the piecewise planar 3D reconstruction is performed by a C++ program. The program has two dependencies: the *Eigen* linear algebra library and *libjpeg* for reading and writing JPEG images. It runs on Linux and macOS. As input it takes the sparse reconstruction from COLMAP (in text format), the polygons from the MATLAB GUI and the images. Polygons are processed sequentially as follows:

1. Estimate a plane from matches of points inside the polygon.
2. Triangulate the polygon and project it onto the plane to create a triangle mesh.
3. Warp the image into the plane to create a texture image.
4. Compute texture coordinates by warping the polygon the same way as the image.

The output of the program is a glTF file and the related buffer files and texture images.

3.4 Pipeline

The full reconstruction pipeline consists of the following steps:

1. Collect images.
2. Use COLMAP to extract the cameras, the image points and point matches.
3. Draw polygons in the MATLAB GUI that define the planar regions to be reconstructed.
4. Run the C++ program to create the resulting 3D model.

The pipeline is visualized as a flow chart in figure 5.

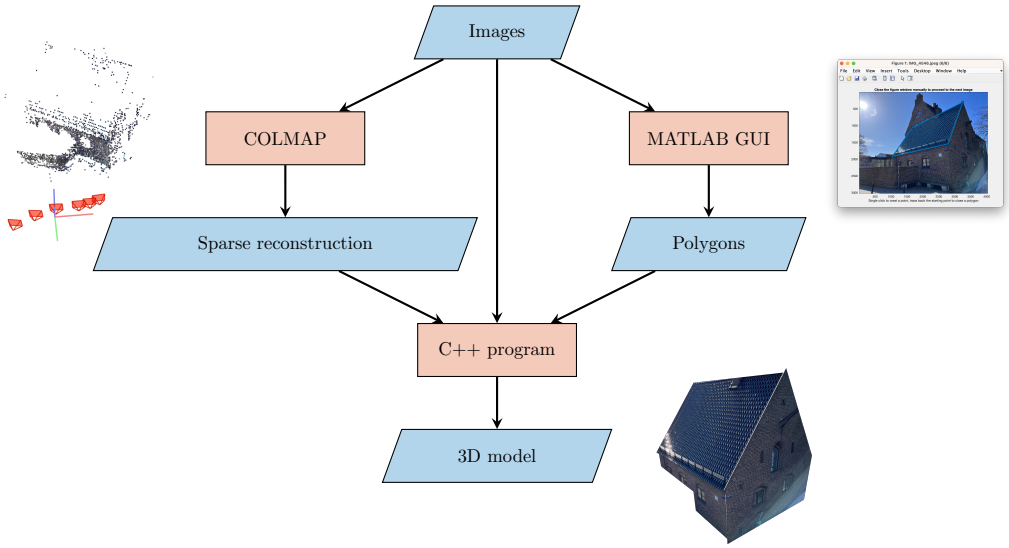


Figure 5: Reconstruction pipeline.

4 Results

Four datasets, ranging from six to twelve images of size 4032×3024, were captured with an iPhone 13. Each dataset was processed as described in section 3 and the resulting piecewise planar 3D models can be found in figure 6. On a MacBook Pro M1 the execution time of the C++ program was around one second per dataset, using 100 MLESAC iterations.

5 Discussion

In this project report we have presented a pipeline for piecewise planar 3D reconstruction. The theoretical background was given in section 2 and the implementation details in section 3. The results in section 4 show that the pipeline is able to produce high-quality textured 3D models in

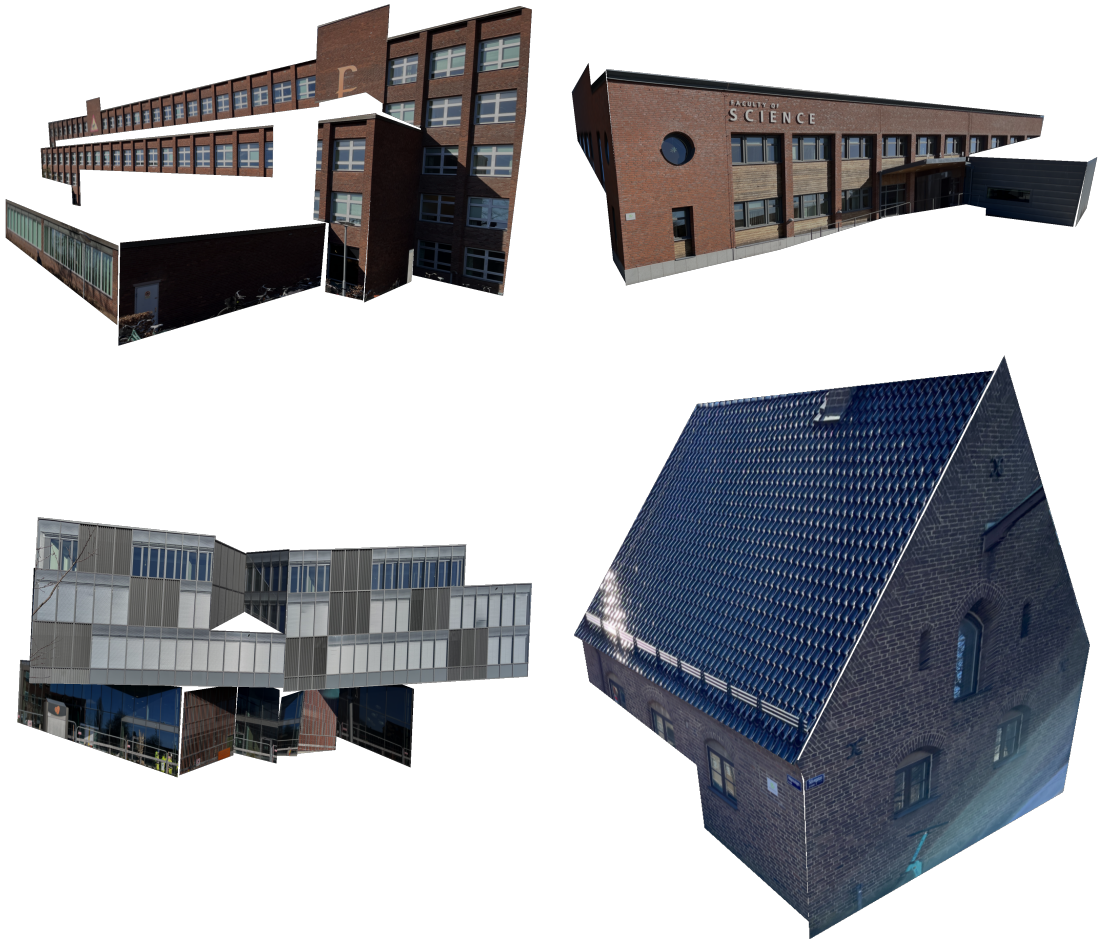


Figure 6: Piecewise planar 3D reconstructions.

several different scenarios. Our C++ program is robust to outliers in the sparse reconstruction, runs quickly and has few dependencies.

There are several aspects of the pipeline that can be improved. Replacing the MATLAB GUI with automatic detection of planar surfaces in the C++ program would make the process more automated, however it's not obvious how this is best done. Small gaps and overlaps between adjacent polygons could be removed by snapping the edges. By combining several different images the texturing could be improved.

References

[1] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[3] G. Hanning and J. Li, "gu6225ha-s/FMAN40 GitHub repository." https://github.com/gu6225ha-s/FMAN40, 2023.

[4] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[5] G. H. Meisters, "Polygons have ears," *The American Mathematical Monthly*, vol. 82, no. 6, pp. 648–651, 1975.