# Pix2Model: A free, easy-to-use Photogrammetry Webservice

Philipp Erler* TU Wien
Maximilian Riegler† TU Wien
Johannes Eschner‡ TU Wien
Florian Steinschorn§ TU Wien
Sophie Pichler¶ TU Wien

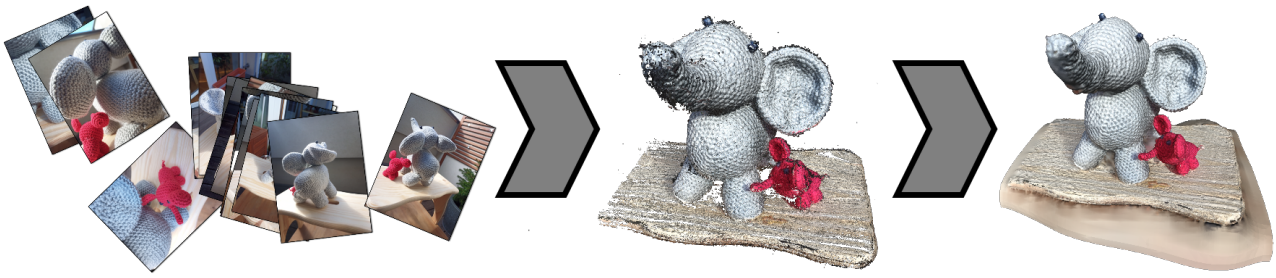


Figure 1: Reconstruction result created with Pix2Model.

## Abstract

TODO: In short, we did well.

The system can be tested here:
`https://netidee.cg.tuwien.ac.at/`
The source code is available on GitHub:
`https://github.com/ErlerPhilipp/Pix2Model`

**CR Categories:** K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

**Keywords:** photogrammetry, webservice, micro services, structure-from-motion, multi-view stereo, surface reconstruction

## 1 Introduction

TODO: We will tell you about how we did well.

1. focus on novice users: as easy as possible
2. minimal maintenance
3. easy to exchange algorithms

## 2 Related Work

For the frontend, we use ideas from modeling tools such as Blender, Maya, and MeshLab. The backend uses COLMAP and Screened Poisson Surface Reconstruction. The webserver implements a micro-service architecture using the technologies Flask and Docker.

*e-mail: perler@cg.tuwien.ac.at
†e-mail: name@example.com
‡e-mail: jeschner@cg.tuwien.ac.at
§e-mail: f.steinschorn@gmail.com
¶e-mail: sophie.pichler@ufficina.de

Similar services include: Images2Mesh [1] and TODO.

### 2.1 Frontend

The user interface of Pix2Model provides functionality to load different models, apply basic transformations, crop away parts of the model and display the dimensions of the loaded model. This functionality is chosen based on the requirements of the pipeline to generate a 3D model based on images. The user needs to be able to view and gradually edit the point cloud model, which is generated in a first step based on the input images. Additionally, he needs to be able to iterate this process after reviewing the reconstruction model that is generated from the edited point cloud. Related tools include Blender, Maya, and MeshLab.

#### 2.1.1 Usability

The implemented UX design is as easy as possible to aid novice users. We compared the applications Blender, Autodesk Maya, and MeshLab in the following aspects. The results of the comparison factor into the development of the interface of Pix2Model while the focus is to reduce complexity even if it does not match standards from existing 3D software.

**Layout**
The layouts of Blender, Maya, and MeshLab are depicted in figure 2. They all provide an overview of the scene objects - which is called *outliner* and is not part of the default layout in Maya - listed in a separate panel next to the viewport. The tools are selectable in a different toolbar panel. By selecting an object, attributes can be changed in an attribute panel within Maya and Blender.

Pix2Model adapts the idea of having a separate panel to see the loaded object, set attributes, and choose tools. But it combines all three panels as the user can only load one object at a time. Therefore it is possible to display the loaded object with its attributes and with the features that are available for the specific object type of the loaded object, i.e. point cloud or mesh without repetition of menu items or confusion about which object is getting edited.

Pix2Model outsources some of the tools that are connected to shortcuts as well to a separate toolbar window on the top left corner. In retrospect, this functionality should have been part of the object panel to serve the purpose of simplicity and to show the user only

[1] `https://images2mesh.com/`

the functionality that is available as soon as it is useful, i.e. as soon as a mesh is loaded.

**Orbiting** Scrolling the middle mouse button (MMB) is a standard user interaction for zooming. With the left mouse button (LMB) the user is orbiting around the scene and panning is implemented in Maya and Blender with a combination of Alt / Shift + LMB and in MeshLab by pressing the MMB while navigating through the scene. Pix2Model also implements scrolling, orbiting, and panning and uses the established user interactions realized in Maya.

**Transformations** The shortcuts used for *Translation*, *Rotation*, and *Scale* are listed in table 1. MeshLab does use key combinations to press while interacting with the scene instead of enabling transformation tools upfront. As there is no general assignment between transformations and shortcuts, Pix2Model implemented the shortcuts according to the initials of the transformations. The center operation in Pix2Model uses *F* as a shortcut for *focusing* or *framing* a selected object.

**Operations** Maya and Blender provide *boolean operations* for cropping. A boolean operation is performed between two objects. One object can be used to cut away the intersecting area between the two objects or to only keep the intersecting area and cut away the parts that do not belong to that intersection. Pix2Model provides a box that can be activated to crop intersecting points from a point cloud. Cropping can be applied iteratively to make complex cut-outs.

**Undo Redo** Maya provides multiple options to undo and redo an action. Buttons within the UI can be used, the functionality can be chosen from a dropdown in the upper menu and the common shortcuts Strg + Z and Strg + Shift + Z are connected to the undo and redo commands. Blender has as well menu items to undo and redo an action and shortcuts to call those. We chose a combination of buttons in the UI and the established shortcuts Strg + Z and Strg + Shift + Z for the undo/redo implementation in Pix2Model. The buttons are part of the implementation to inform the user that this functionality is available.

**Mobile Support** While Maya and Blender are complex applications, MeshLab has reduced functionality. Maya and Blender are cross-platform but are not supported on mobile phones. MeshLab on the other hand has a simplified mobile version in which the user can view an object. Pix2Model as well functions as a 3D viewer on the mobile version without editing options.

| | Blender | Maya | Pix2Model |
|---|---|---|---|
| Translation | G | W | T |
| Rotation | R | E | R |
| Scale | S | R | S |
| Centering | Numpad . | F | F |

Table 1: Shortcuts for Transformations and centering an object

## 2.2 Backend

### 2.2.1 Photogrammetry

COLMAP is a system that generates 3D models out of a number of images of an object by using Structure-from-Motion (SfM) [Schönberger and Frahm 2016] and Multi-View Stereo (MVS) [Schönberger et al. 2016]. It can be used as a command-line application, but also has a graphical interface.
Meshroom is another program implementing a photogrammetry pipeline consisting of SfM and MVS [Griwodz et al. 2021]. It has a graphical user interface, as well as many other features like plugins for popular editors such as Blender [2] or Maya [3].
Screened Poisson Surface Reconstruction (SPSR) by Kazhdan and Hoppe [Kazhdan and Hoppe 2013] is an algorithm to construct 3D meshes from point clouds. This means it only covers the last step of the photogrammetry pipeline after a point cloud has been generated through SfM and MVS. For Pix2Model we use the version implemented by Meshlab [Cignoni et al. 2008].

### 2.2.2 Screened Poisson Surface Reconstruction

Screened Poisson Surface Reconstruction by Kazhdan and Hoppe [Kazhdan and Hoppe 2013] is an algorithm to create watertight surfaces from oriented point clouds. This is done by first converting it into a vector field that describes the gradient of an indicator function. This function is positive inside the model and negative outside. To find this function, its error is minimized by fitting b-splines into the leaves of an octree. The resulting zero-set does then represent the models surface.
The big advantage of SPSR over standard Poisson Surface Reconstruction is an additional term in the error function that discourages its deviation from zero at the sampling locations, therefore sticking closer to the points of the point cloud. Additionally, the boundary condition in the function space is different. SPSR uses the Neumann condition instead of Dirichlet, which leads to the closing of holes, but also creates huge artifacts at the outer areas of non-watertight models, like standing 3D scans. After this reconstruction, the inverse average distance of all points from the surface is added to each point to reduce the overall global error.

TODO: Texturing

## 2.3 Server

TODO: Flask (Build process)

TODO: (nvidia) docker

# 3 Implementation

This is how we did well.

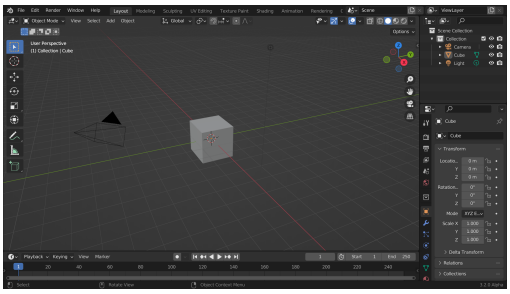
## 3.1 Structure From Motion

In the first step of our reconstruction pipeline, the Structure From Motion (SFM) implementation COLMAP[4] is used to generate a point cloud from the input images. COLMAP utilizes an incremental SFM implementation [Schönberger and Frahm 2016], [Schönberger et al. 2016] to generate a dense point cloud of the input scene. This implementation works as follows:

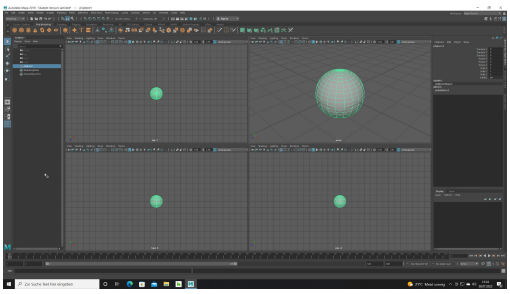
First, a correspondence search is performed in the input images to find overlaps. Then an incremental reconstruction is performed where, starting from an initial image pair, new images are iteratively registered into the scene to add additional data points for the final point-cloud output.

To integrate COLMAP into our multi-staged webservice, we created a Python wrapper to interface with the COLMAP commands. By not modifying the existing source code and, instead, building our own interface we can incorporate possible future COLMAP updates into our pipeline without the need of adapting the COLMAP source. Furthermore, due to this interface nature, exchanging the

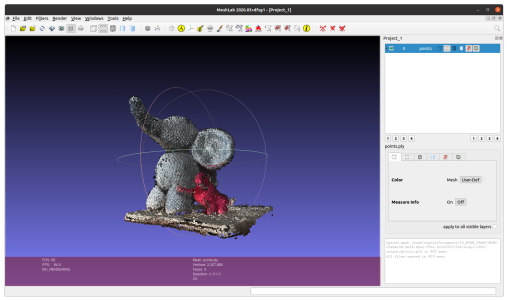
[2] www.blender.org
[3] www.autodesk.com/products/maya/overview
[4] https://colmap.github.io/



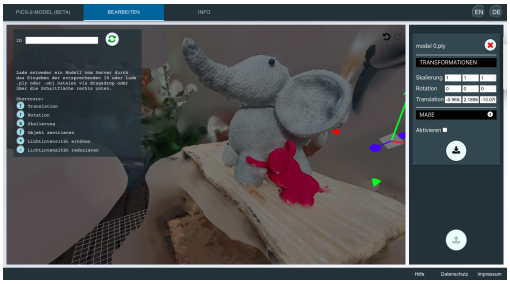
(a) Blender Layout



(b) Maya Layout



(c) Meshlab Layout



(d) Pix2Model Layout

Figure 2: Comparison of the layouts of Blender, Maya, MeshLab and Pix2Model

existing SFM step with a different implementation can be done seamlessly.

In contrast to COLMAP, which by itself also covers mesh reconstruction, our implementation does not use this feature. Instead, our final result of step 1 is a dense point cloud, which is then used as a basis for mesh reconstruction in step 2. The final output generated by our solution is a folder that contains the point cloud in PLY format as well as a log file.

## 3.2 Mesh Reconstruction

Step 2 of our reconstruction pipeline is Mesh Reconstruction. We get the point-cloud files generated in step 1 and apply Screened Poisson Surface Reconstruction (SPSR) to them. This is done using the implementation of Meshlab [Cignoni et al. 2008]. SPSR is described in the following Section 2.2.2. Since we do not have any information about the input cloud, we just use the default arguments for this reconstruction instead of adjusting them specifically. For meshes with over one million faces, we additionally apply the Quadric Edge Collapse Decimation Simplification to reduce the face count to one million. This algorithm is based on Quadric Error Metrics Simplifications by Garland and Heckbert [1998]. This reduces the memory usage of the resulting mesh significantly, which is especially important for the limited performance and data transfers of mobile devices.

### 3.2.1 Screened Poisson Surface Reconstruction

For Pix2Model we use the SPSR version implemented by Meshlab [Cignoni et al. 2008]. SPSR is highly configurable with multiple parameters. Still, for our work, we stick to the default values, since not enough is known about the input clouds to better adjust any parameters. We chose this over the default reconstruction implemented by Colmap (DelaunayMeshing) since we have prior experience with it and it seems to yield better results.

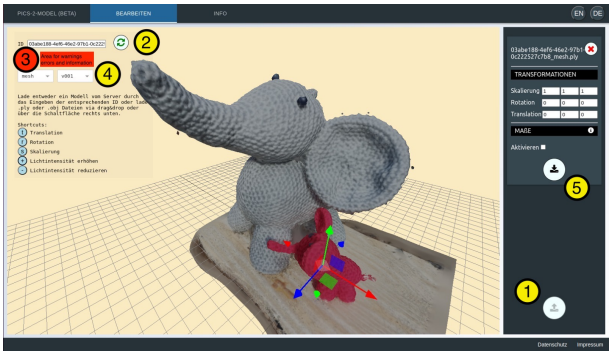
## 3.3 Frontend

Figures 3 and 4 depict the user interface of the application.

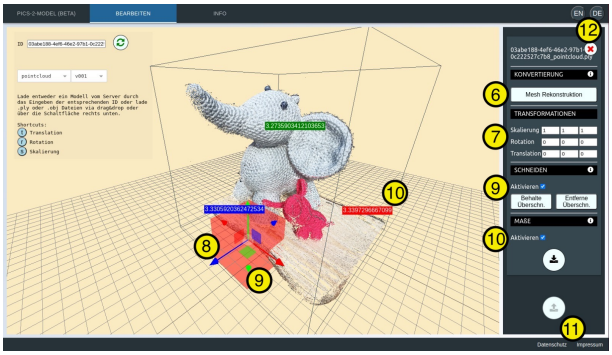
The frontend is implemented in React and the 3D editor is based on threejs. It is a single-page application, which enables the user to upload images (Pix2Model) - Figure 4 - that are further used to reconstruct the 3D mesh, to edit 3D models (EDIT) - Figure 3 - and to receive further information about the project and the team (INFO).

**Pix2Model**
The upload as depicted in figure 3 provides the option to initiate only the point cloud reconstruction as described in Section 3.1 or both the point cloud reconstruction and the mesh reconstruction, discussed in Section 3.2. Additionally, the user can request an email notification after the upload has terminated.
After successfully uploading the images, the upload page displays a link to the edit page. The redirecting link will update the URL to contain the process ID and it will navigate the user to the edit page. The reconstruction takes several minutes, depending on the amount and resolution of the images. For 100 photos, the reconstruction takes about one hour. Therefore, the mesh won't be accessible immediately after redirecting to the edit page.

**EDIT: Load and Download Models**
The user can load models from his local machine or from the server. The former can be performed with drag&drop or by using the *Upload button* (Figure 3, 1). The latter is done by entering the ID to the ID textfield and pressing the *Refresh button* (Figure 3, 2). In case the user navigated to the Edit page via the previously mentioned redirection link, the ID is already copied to the ID textfield. The model reconstruction might still be in progress, such that the UI displays an error message that no model is available for the requested ID (Figure 3, 3). To pull updates, the refresh button needs to be pressed actively again (Figure 3, 2).
The requested reconstruction ID can be linked to multiple models.




(a) Mesh Model: Editor Features




(b) Pointcloud Model: Editor Features

Figure 3: 3D Editor: (1) Upload model, (2) Load model from the server, (3) Information, warnings, and errors, (4) Version selection, (5) Download, (6) Create new version for the model to reconstruct mesh with current adjustments, (7) Transformations, (8) Gizmo which is attached to the model if the crop features are not activated and attached to the crop-box otherwise, (9) Crop feature, (10) Measurement feature, (11) Privacy and Impressum, (12) Change language

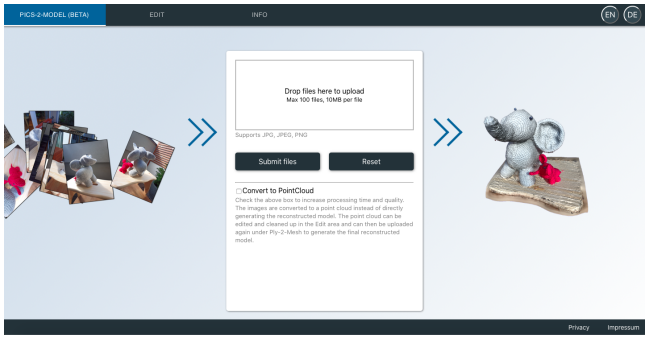



Figure 4: Upload images to start the reconstruction of a 3D model

Initially, it points to the most recently created model, which is the output of step 1, in case step 2 was not created (yet) or to the output of step 2 otherwise. The user can choose between *pointcloud* and *mesh* and different versions for those model types in the *dropdown menu* (Figure 3, 4).
The loaded mesh can be downloaded with the *download button* (Figure 3, 5).

**EDIT: Version Up**
The initial version is v000. Each new version is an increment of the previously created version. New meshes are created for a reconstruction ID, if a user loads a point cloud (output of step 1) and presses the *reconstruct mesh button* (Figure 3, 6). This enables the user to make adjustments on a point cloud before the reconstruction of the mesh. Those adjustments can include the removal of background points and transformations.

**EDIT: Transformation**
Transformations are *scaling*, *rotation* and *translation*. They can be applied textually (Figure 3, 7) or by using the gizmo (Figure 3, 8). The keyboard shortcuts 'T' for Translation, 'R' for Rotation, and 'S' for Scale can be used to switch between the Transformation modes.

**EDIT: Crop**
Cropping is only available for point clouds. The crop functionality needs to be activated first. This spawns a crop box that can be transformed to cover a subset of the points in the point cloud (Figure 3, 9). These points can either be removed or singled out. 3 shows the crop box, which intersects with the points that form the table that the figures stand on.

**EDIT: Measurement**
When the measurement feature is activated, the bounding box that includes all the points indicates the size of the 3D model, and the labels attached to the bounding box display the exact values of the bounding box in each dimension (Figure 3, 10).

**INFO**
Information about the project's goal, its usability, and the team members are listed under *Info*. Privacy information and the Impressum as well as a support section that links to Github to file issues are always accessible on all subpages with the links in the bottom right corner (Figure 3, 11). The user can toggle between English and German with the buttons in the top right corner (Figure 3, 12).

### 3.4 Backend

1. how calls to steps 1 and 2 are started
2. data protection: UUID
3. microservices
4. redis
5. ssh

## 4 Comparison

To compare Pix2Model with other systems available we conducted two types of tests. Firstly, we used real-world datasets and looked at the resulting 3D models. Secondly, we generated synthetic datasets, where we can extract the ground truth and calculate a Chamfer distance for both our results as well as those of other systems. For both tests, the reconstruction was conducted with Pix2Model, COLMAP, and Meshroom.
In the following sections, we will firstly discuss Chamfer Distance (4.1) and then look at the real world (Sec. 4.2) and synthetic data (Sec. 4.3).

### 4.1 Chamfer Distance

For the metric of the quality of a reconstruction, we chose the Chamfer Distance [Barrow et al. 1977]. The Chamfer Distance of

a reconstructed model is defined as the sum of the minimal distances of each reconstructed point to the nearest point on the original model. Alternatively, the average of minimal distances can be used ([Dantanarayana et al. 2016]). The Chamfer Distance of a reconstruction $R$ to the ground-truth $G$ is given in Equation 1.

$$CD(R,G) = \sum_{r_i \in R} \min_{g_j \in G} |r_i - g_i| \quad (1)$$

### 4.2 Real-World Data

As a real-world dataset, we used the 'Gerrard Hall' dataset provided by COLMAP [5]. All three systems managed to reconstruct the building, although Meshroom had some problems, resulting in an additional wing. An overview can be found in Figure 5. As can be seen here, both COLMAP and Pix2Model successfully reconstructed most of the details visible in the dataset. As is expected from the different meshing algorithms, COLMAP is leaving holes where no data was provided, while Pix2Model fills them up to get a watertight model.

### 4.3 Synthetic Data

To generate our synthetic datasets, we used the Blender plugin SfM Flow [Marelli et al. 2022]. This plugin generated a scene around an object, consisting of a floor, sunlight and a camera. It is also used for rendering images of the generated scene according to the path of the camera around it. The objects we used are partly from the dataset provided with the plugin (hydrant, jeep; [Bianco et al. 2018]), and partly found as free downloads online (locomotive). For each object, we created a dataset of 100 pictures using a camera rotating around the scene. An example of such a dataset can be seen in Fig. 6.
Discussions on the results of the different datasets can be found in the following Sections 4.3.1 - 4.3.3.

#### 4.3.1 Jeep

The Jeep dataset yielded the best results of all our synthetic datasets. An overview can be found in Fig. 7.

It is visible in these images, that all systems achieve a reasonable result. The major differences are, that COLMAP leaves a lot of bubble-like shapes around the object, some smaller nearer to it and a few big ones farther away. Meshroom shows the same behavior but to a much smaller extent. Pix2Model on the other hand does not have this problem but creates a sort of 'saddle' under the object because of SPSR's (Sec. 2.2.2) tendency to always generate watertight meshes.
We used the Chamfer Distance 4.1 to calculate the distance from each reconstruction to the ground truth. We did this for both the unedited models, as well as for a version with the 'bubbles' and the 'saddle' removed. The results are listed in Table 2. This comparison shows, that without cleanup, Meshroom yields the closest results to the ground truth, with a distance of 2670. This is mainly due to it having the least amount of outliers. After cleanup Pix2Model is much closer to the original with a distance of 1359, closely beating COLMAP's 1432. The comparison between the original and the cleaned version can be found in Fig. **??**.

#### 4.3.2 Hydrant

The second object we used for our synthetic data tests is a fire hydrant. Some example images from the dataset is given in Figure 9. It was far more difficult for all systems to reconstruct, but

[5] https://colmap.github.io/datasets.html

Table 2: Chamfer distances calculated for both the unedited and the cleaned versions of the results.

| | Unedited | Cleaned |
|---|---|---|
| Pix2Model | 4850.97 | 1358.68 |
| COLMAP | 7063.59 | 1432.21 |
| Meshroom | 2669.58 | 2509.6 |

Pix2Model yielded the best results, as can be seen in Figure 10. COLMAP failed to generate any output and Meshroom resulted in a shape that is not near the ground truth at all. This is mostly due to the failure to correctly determine the position of the cameras in the room in the structure from the motion step. A visualization of this can be found in Figure 11.
Even though Pix2Model is the only system that generates a recognizable hydrant, still only the front has actual detail and the back is just a connecting surface. This is leveraging the advantage of the SPSR algorithm, which tries always to generate watertight objects.

#### 4.3.3 Locomotive

Our last synthetic dataset contains images of a steam locomotive, an example can be seen in Figure 12. Similar to the hydrant, all systems struggled with determining the correct camera positions. The biggest problem seems to be that there are images from the right and left sides and the algorithms weren't able to correctly detect that. An overview of the results is given in Figure 13. For this dataset, Meshroom only managed to reconstruct parts of the left side of the locomotive, the detected camera positions can be found in Figure 14. Pix2Model had the same problem but generated a much more complete surface of the left side, and again, due to the usage of SPSR, created a watertight surface by extending the edges outwards. This leads to the locomotive seemingly being half-melded into a mountain. COLMAP had a different problem with this dataset. It did not detect that images were from different sides, so the resulting mesh is a blend of two mirrored locomotives.

## 5 Conclusion

We did well.

## 6 Acknowledgements

Thanks to xxx for funding this project.

## References

BARROW, H. G., TENENBAUM, J. M., BOLLES, R. C., AND WOLF, H. C. 1977. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings: Image Understanding Workshop*, Science Applications, Inc Arlington, VA, 21–27.

BIANCO, S., CIOCCA, G., AND MARELLI, D. 2018. Evaluating the performance of structure from motion pipelines. *Journal of Imaging 4*, 8.

CIGNONI, P., CALLIERI, M., CORSINI, M., DELLEPIANE, M., GANOVELLI, F., AND RANZUGLIA, G. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, The Eurographics Association, V. Scarano, R. D. Chiara, and U. Erra, Eds.

(a) COLMAP (b) Meshroom (c) Pix2Model

Figure 5: Comparison of the results created from the GerrardHall dataset.

Figure 6: Example images from the generated ‘jeep‘ dataset.


DANTANARAYANA, L., DISSANAYAKE, G., AND RANASINGE, R. 2016. C-log: A chamfer distance based algorithm for localisation in occupancy grid-maps. *CAAI Transactions on Intelligence Technology 1*, 3, 272–284.

GARLAND, M., AND HECKBERT, P. S. 1998. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the Conference on Visualization '98*, IEEE Computer Society Press, Washington, DC, USA, VIS '98, 263–269.

GRIWODZ, C., GASPARINI, S., CALVET, L., GURDJOS, P., CASTAN, F., MAUJEAN, B., LILLO, G. D., AND LANTHONY, Y. 2021. Alicevision Meshroom: An open-source 3D reconstruction pipeline. In *Proceedings of the 12th ACM Multimedia Systems Conference - MMSys '21*, ACM Press.

KAZHDAN, M., AND HOPPE, H. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG) 32*, 3, 29.

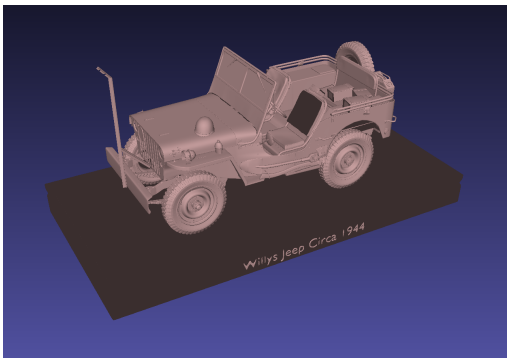
MARELLI, D., BIANCO, S., AND GIANLUIGI, C. 2022. Sfm flow: A comprehensive toolset for the evaluation of 3d reconstruction pipelines. *SoftwareX 17*, 100931.

SCHÖNBERGER, J. L., ZHENG, E., POLLEFEYS, M., AND FRAHM, J.-M. 2016. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*.

SCHÖNBERGER, J. L., AND FRAHM, J.-M. 2016. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4104–4113.


## A Contributions

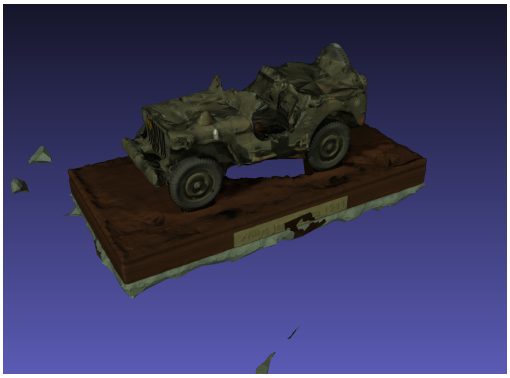
1. Philipp Erler: Project Management
2. Maximilian Riegler: Backend
3. Sophie Pichler: Frontend
4. Johannes Eschner: COLMAP
5. Florian Steinschorn: Reconstruction




(a) Ground Truth

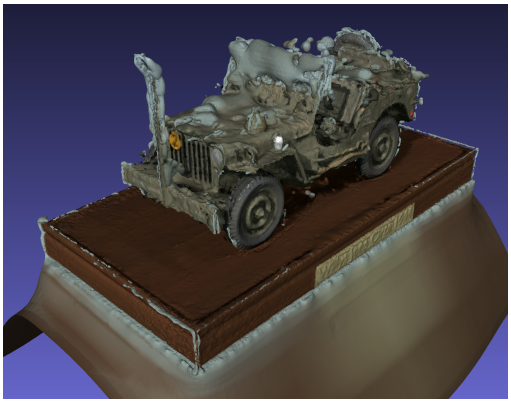
(b) Pix2Model

(c) COLMAP

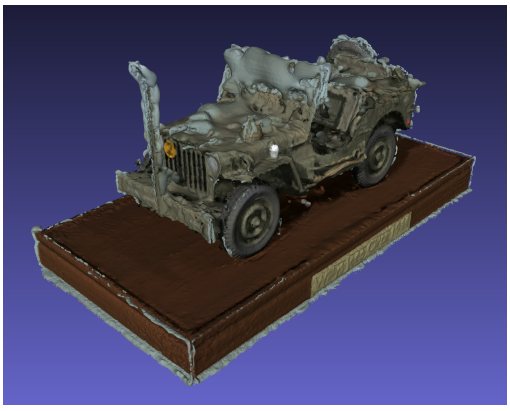


(d) Meshroom

Figure 7: Comparison of the results created from the Jeep dataset.



(a) Pix2Model result



(b) Pix2Model result cleaned

Figure 8: Pix2Model result for the jeep dataset. Initial and cleaned version.

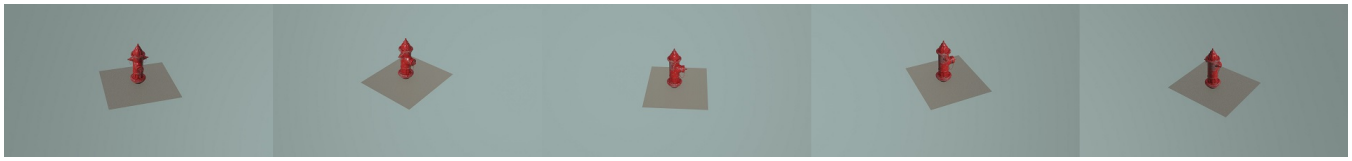
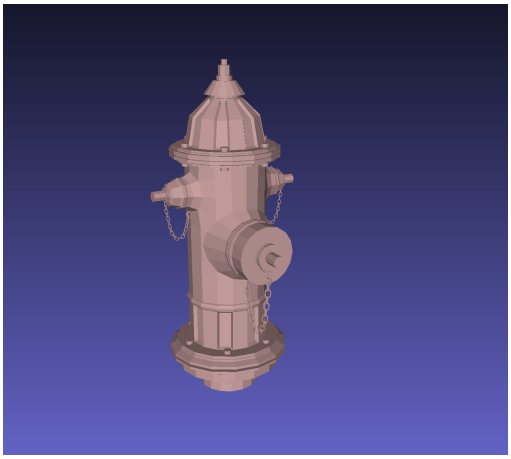
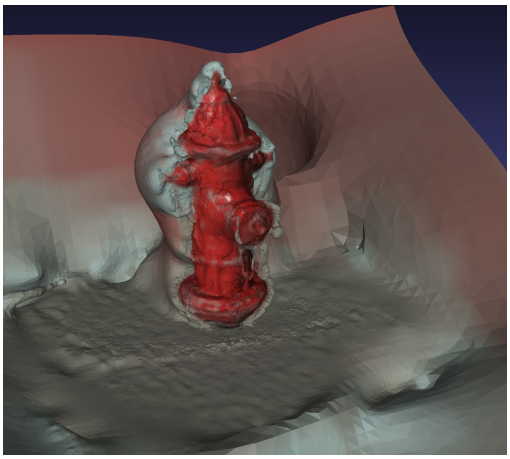


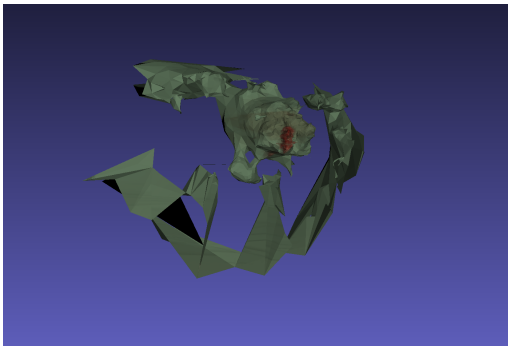
Figure 9: Example images from the Hydrant dataset.



(a) Ground Truth



(b) Pix2Model



(c) COLMAP (reconstruction failed)

(d) Meshroom

Figure 10: Comparison of the results created from the Hydrant dataset.

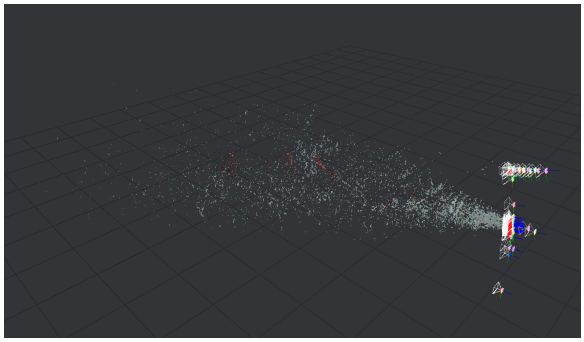


Figure 11: Meshrooms structure from motion result for the Hydrant dataset.

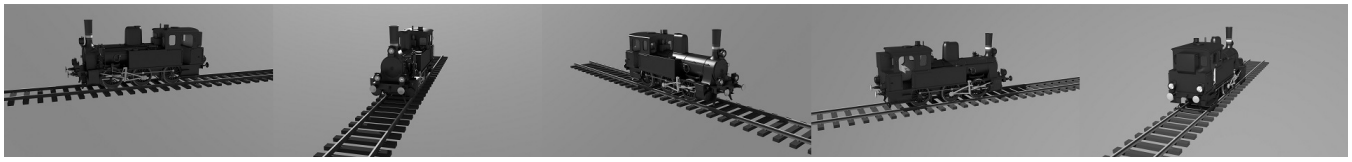
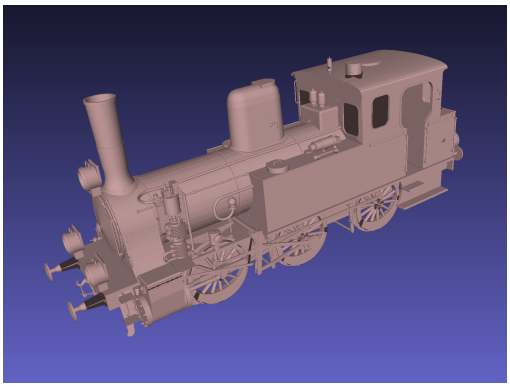
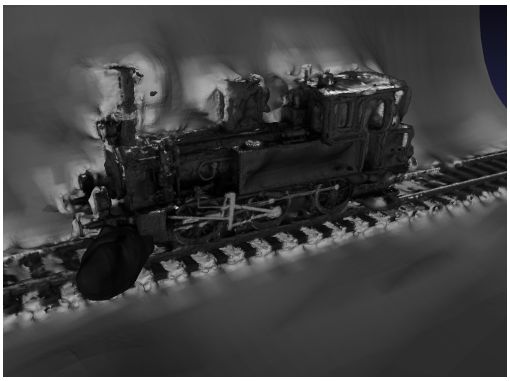


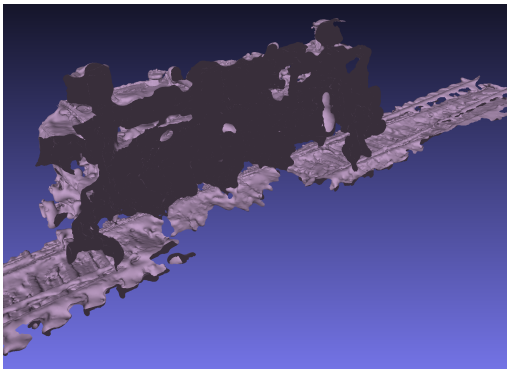
Figure 12: Example images from the Locomotive dataset.



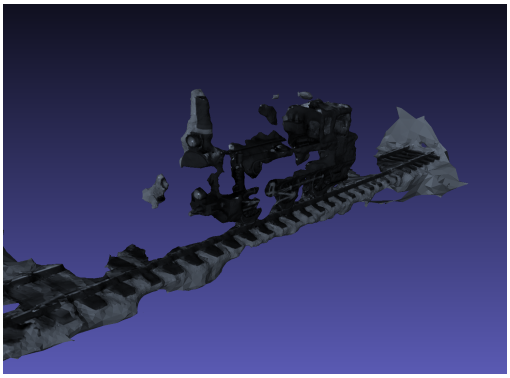
(a) Ground Truth



(b) Pix2Model



(c) COLMAP



(d) Meshroom

Figure 13: Comparison of the results created from the Locomotive dataset.

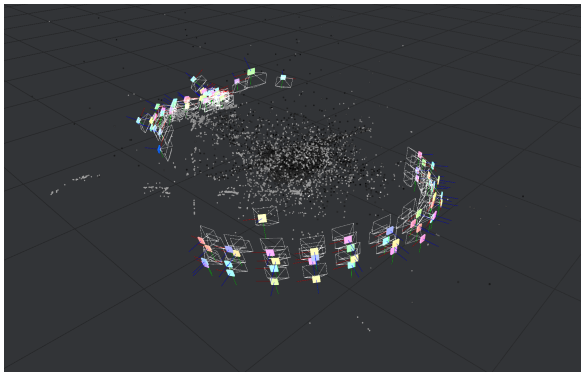


Figure 14: Meshrooms structure from motion result for the Locomotive dataset.