A Photogrammetry-based Framework to Facilitate Image-based Modeling and Automatic Camera Tracking

Sebastian Bullinger[a], Christoph Bodensteiner[b] and Michael Arens[c]
Department of Object Recognition, Fraunhofer IOSB, Ettlingen, Germany

Keywords: Image-based Modeling, Camera Tracking, Photogrammetry, Structure from Motion, Multi-view Stereo, Blender.

Abstract: We propose a framework that extends Blender to exploit Structure from Motion (SfM) and Multi-View Stereo (MVS) techniques for image-based modeling tasks such as sculpting or camera and motion tracking. Applying SfM allows us to determine camera motions without manually defining feature tracks or calibrating the cameras used to capture the image data. With MVS we are able to automatically compute dense scene models, which is not feasible with the built-in tools of Blender. Currently, our framework supports several state-of-the-art SfM and MVS pipelines. The modular system design enables us to integrate further approaches without additional effort. The framework is publicly available as an open source software package.

1 INTRODUCTION

1.1 Photogrammetry-based Modeling and Camera Tracking

Many tasks in the area of image-based modeling or visual effects such as sculpting or motion and camera tracking involve a substantial amount of user interaction to achieve satisfying results. Even with many modern tools like Blender (Blender Online Community, 2020), designers require to perform many steps manually. With the recent progress of *Structure from Motion* (SfM), *Multi-View Stereo* (MVS) and texturing techniques the automation of specific steps such as the determination of the camera motion and the reconstruction of the scene geometry (including texture computation) has become feasible.

Using common mesh data formats, the majority of modeling tools allows to import the reconstructed geometry and corresponding textures of state-of-the-art MVS (Fuhrmann and Goesele, 2014; Jancosek and Pajdla, 2014; Schönberger et al., 2016; Ummenhofer and Brox, 2017) and texturing libraries (Burt and Adelson, 1983; Waechter et al., 2014). However, such data formats do not include camera calibration
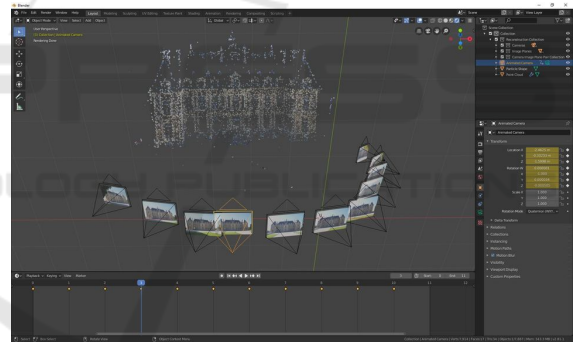

Figure 1: Reconstruction result of the sceaux dataset (Moulon, 2012) in the 3D view of Blender including the reconstructed camera poses as well as the corresponding point cloud.

and camera motion information, which is crucial for many tasks such as creating visual effects.

In order to overcome these limitations, we created a framework that enables us to integrate reconstructions of different state-of-the-art SfM and MVS libraries into Blender - see Fig. 1 for an example. Since the source code of all components is publicly available, the full pipeline is not only suitable for modeling or creating visual effects, but especially for future research efforts.

At the same time, our framework servers in combination with Blender's animation and rendering capabilities as a tool for the photogrammetry community that offers sophisticated inspection and visualiza-

[a] https://orcid.org/0000-0002-1584-5319
[b] https://orcid.org/0000-0002-5563-3484
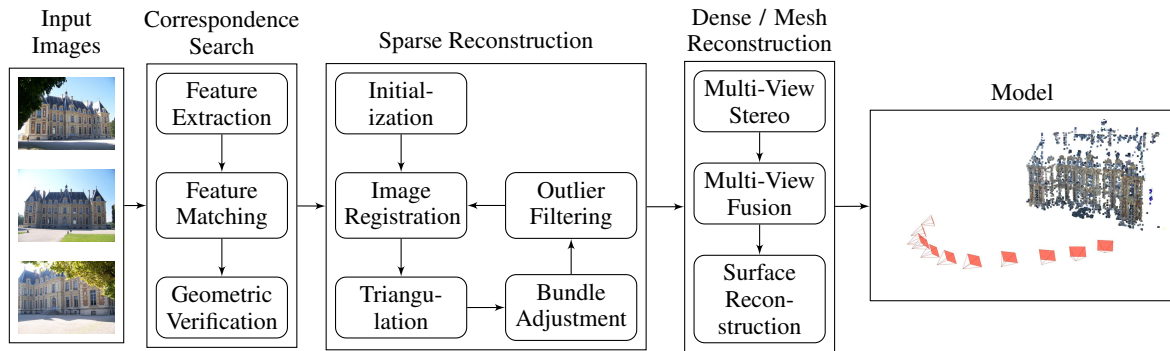[c] https://orcid.org/0000-0002-7857-0332

Figure 2: Building blocks of state-of-the-art incremental Structure from Motion and Multi-view Stereo pipelines. The input images are part of the Sceaux Castle dataset Moulon (2012).

tion functionalities, which are not present in other editors such as Meshlab (Cignoni et al., 2008) or Cloud-Compare (Daniel Girardeau-Montaut, 2020).

1.2 Related Work

SfM is a photogrammetric technique that estimates for a given set of (unordered) input images the corresponding three-dimensional camera poses and scene structures. There are two categories of SfM approaches: *incremental* and *global* SfM. Incremental SfM is currently the prevalent state-of-the-art method (Schönberger and Frahm, 2016). In order to manage the problem complexity of reconstructing real-world scenes, incremental SfM decomposes the reconstruction process into more controllable subproblems. The corresponding tasks can be categorized in *correspondence search* (including *feature detection*, *feature matching* and *geometric verification*) and *sparse reconstruction* (consisting of *image registration*, *point triangulation*, *bundle adjustment* and *outlier filtering*). During bundle adjustment, SfM minimize the reprojection error of the reconstructed three-dimensional points for each view.

MVS uses the camera poses and the sparse point cloud obtained in the SfM step to compute a dense point cloud or a (textured) model reflecting the geometry of the input scene. Similarly to SfM, MVS divides the reconstruction task in multiple subproblems. The *multi-view stereo* step computes a depth map for each registered image that potentially includes surface normal vectors. *Multi-view fusion* fuses the depth maps into a unified dense reconstruction that allows to reconstruct a watertight surface model in the *surface reconstruction* step. Fig. 2 shows an overview of essential SfM and MVS subtasks and their dependencies.

Currently, there are several state-of-the-art photogrammetry libraries that provide full SfM and MVS pipelines such as Colmap (Schönberger, 2020),

Meshroom (AliceVision, 2020a), Multi-View Environment (Fuhrmann et al., 2014), OpenMVG (Moulon et al., 2013) & OpenMVS (Cernea, 2020) as well as Regard3D (Hiestand, 2020). For a quantitative evaluation of state-of-the-art SfM and MVS pipelines on outdoor and indoor scenes see Knapitsch et al. (2017), which provides a benchmark dataset using laser scans as ground truth.

While the usage of the reconstructed (textured) models is widely supported by modern modeling tools, the integration of camera-specific information such as intrinsic and extrinsic parameters are often-times neglected. There are only a few software packages available that allow to import camera-specific information into modeling programs. The majority of these packages address specific proprietary reconstruction or modeling tools such as AliceVision (2020b), SideEffects (2020) or Uhlík (2020). The most similar tool compared to the proposed framework is presumably Attenborrow (2020), which also provides options to import SfM and MVS formats into Blender. However, the following capabilities of our framework are missing in Attenborrow (2020): visualization of colored point clouds, representation of source images as image planes and creation of point clouds from depth maps. Further, Attenborrow (2020) supports less SfM and MVS libraries and provides less options to configure the input data.

1.3 Contribution

The core contribution of this work are as follows. (1) The proposed framework allows to leverage image-based reconstructions (*e.g.* automatic calibration of intrinsic camera parameters, computation of three-dimensional camera poses and reconstruction of scene structures) for different tasks in Blender such as sculpting or creating visual effects. (2) We use available data structures in Blender to represent the integrated reconstruction results, which al-

Table 1: Overview of photogrammetry pipelines that are supported by the proposed framework. For each pipeline the table shows the corresponding methods to compute the different reconstruction steps including Structure from Motion (SfM), Multi-view Stereo (MVS), surface reconstruction (Surface Rec.) and texturing. In many cases the pipelines allow to substitute specific pipeline steps using alternative implementations. This table shows the default or the recommended pipeline configurations.

Pipeline	Colmap
SfM	Schönberger and Frahm (2016)
MVS	Schönberger et al. (2016)
Surface Rec.	Kazhdan and Hoppe (2013)
Texturing	-

Pipeline	Meshroom
SfM	Moulon et al. (2012)
MVS	Hirschmuller (2005)
Surface Rec.	Jancosek and Pajdla (2014)
Texturing	Burt and Adelson (1983)

Pipeline	MVE
SfM	Fuhrmann et al. (2014)
MVS	Goesele et al. (2007)
Surface Rec.	Fuhrmann and Goesele (2014)
Texturing	Waechter et al. (2014)

Pipeline	OpenMVG / OpenMVS
SfM	Moulon et al. (2012)
MVS	Barnes et al. (2009)
Surface Rec.	Jancosek and Pajdla (2014)
Texturing	Waechter et al. (2014)

Pipeline	Regard3D
SfM	Moulon et al. (2012)
MVS	Langguth et al. (2016)
Surface Rec.	Fuhrmann and Goesele (2014)
Texturing	Waechter et al. (2014)

Pipeline	VisualSfM
SfM	Moulon et al. (2012)
MVS	Furukawa and Ponce (2010)
Surface Rec.	-
Texturing	-

lows the framework to compute automatic camera animations (including extrinsic and intrinsic camera parameters), represent the reconstructed point clouds as particle system or attach the source images to the registered camera poses. Using the available data structures in Blender ensures that the integrated results can be further utilized.

(3) This framework provides (together with Blender's built-in tools) different visualization and animation capabilities for image-based reconstructions that are superior to tools offered by common photogrammetry-specific software packages.

(4) The framework supports already many state-of-the-art open source SfM and MVS pipelines and is (because of its modular design) easily extensible.

(5) The source code of framework is publicly available[1].

2 FRAMEWORK

2.1 Overview

The proposed framework allows us to import the reconstructed scene geometry represented as point cloud or as (textured) mesh, the reconstructed cameras (including intrinsic and extrinsic parameters), point clouds corresponding to the depth maps of the cameras and an animated camera representing the camera motion. The supported libraries

include the following photogrammetry pipelines: Colmap (Schönberger, 2020), Multi-View Environment (Fuhrmann et al., 2014), OpenMVG (Moulon et al., 2013) & OpenMVS (Cernea, 2020) and Meshroom (AliceVision, 2020a) as well as VisualSfM (Wu, 2011). Table 1 contains an overview of each pipeline with the corresponding reconstruction steps. An example reconstruction result of Colmap is shown in Fig. 3a.

In addition to the SfM and MVS libraries mentioned above, the system supports the integration of camera poses or scene structures captured with RGB-D sensors using Zhou et al. (2018) as well as point clouds provided in common laser scanning data formats.

2.2 Architecture

We followed a modular design approach in order to simplify the extensibility of the framework. An overview of the different components and their dependencies is shown in Fig. 4. Each supported library requires the implementation of a corresponding *FileHandler* and *ImportOperator*. The *FileHandler* parses library specific file formats or directory structures and returns library agnostic information of cameras, points and meshes. The *ImportOperator* may use different classes provided by the framework such as the *CameraImporter*, *PointImporter* and *MeshImporter* to define the required import options and to import the reconstruction information extracted by the *FileHandler*.

---

[1]Source code is available at https://github.com/SBCV/Blender-Addon-Photogrammetry-Importer
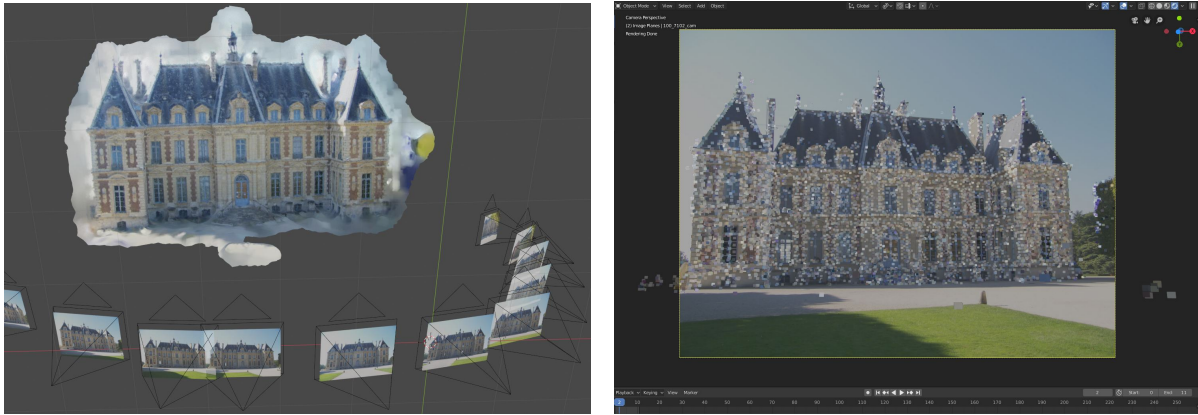
(a) Reconstructed geometry represented as mesh with vertex colors in the 3D view of Blender.

(b) Reconstructed point cloud in the 3D view of Blender from the perspective of one of the reconstructed cameras. The background image shows the corresponding input image used to register the camera.

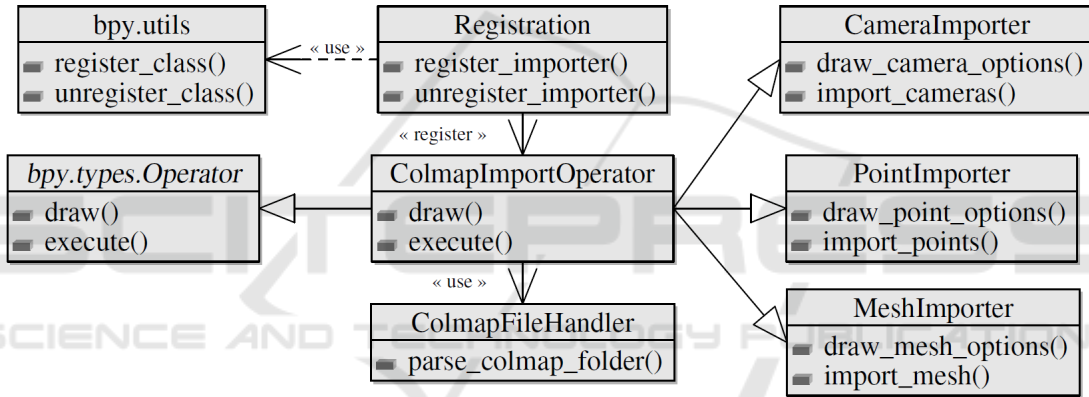Figure 3: Reconstructed geometry in the 3D view of Blender.

Figure 4: Integration of the proposed framework in Blender - illustrated with the Colmap importer. The class *bpy.types.Operator* provided by Blender allows to define custom operators that can be register with *bpy.utils*. In order to support additional SfM and MVS libraries it is sufficient to implement the corresponding import operators and file handlers. To simplify the figure only relevant classes and methods are shown.

2.3 Camera and Image Representation

The supported SfM and MVS libraries use different camera models to represent the intrinsic camera parameters during the reconstructions process. Further, the conventions describing the extrinsic camera parameters are also inconsistent. We convert the different formats into a unified representation that can be directly mapped to Blender's *camera objects*.

In addition to the integration of the geometric properties of the reconstructed cameras, the framework provides an option to add each input image as *background image* for the corresponding camera object. Viewing the scene from the perspective of a specific camera allows to assess the consistency of virtual objects and the corresponding source images, which is especially useful for sculpting tasks. It also of-

fers convenient capabilities to visualize and inspect the reconstructed point clouds and meshes, which are not feasible with other photogrammetry-specific tools such as CloudCompare and Meshlab. Fig. 3b shows for example a comparison of the projected point cloud and the color information of the corresponding input image.

To further enhance the visualization, the system provides an option to add the original input images as separate image planes as shown in Fig. 1.

In order to ease the usage of the reconstruction for animation and visual effect tasks, the framework offers an option to create an animated camera using the reconstructed camera poses as well as the corresponding intrinsic parameters such as the focal length and the principal point. All parameters are animated by using Blender's built-in *f-curves*, which
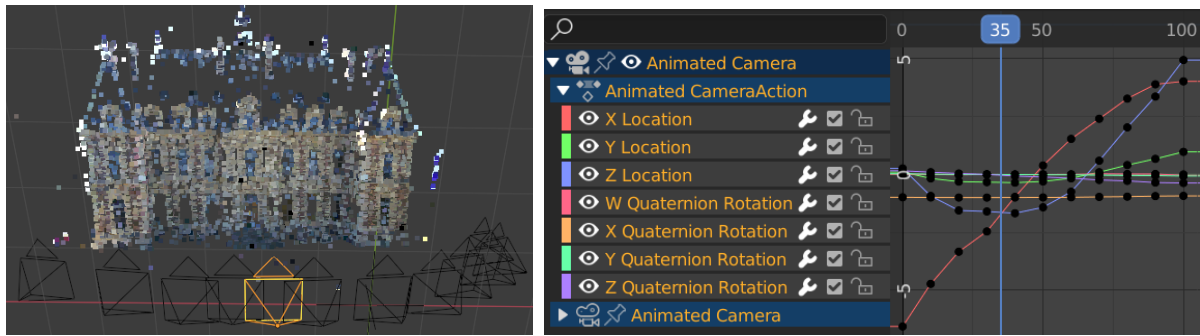
(a) Reconstruction result in Blender's 3D view. The reconstructed cameras are shown in black and the animated camera in orange, respectively.

(b) Interpolation values of the translation and the rotation corresponding to the animated camera in the left image. The black dots denote the values of the reconstructed camera poses and the vertical blue line indicates the interpolated values at the position of the camera in the left image.

Figure 5: Example of a camera animation using 11 images of the Sceaux Castle dataset (Moulon, 2012). By interpolating the poses of the reconstructed cameras, we obtain a smooth trajectory for the animated camera.
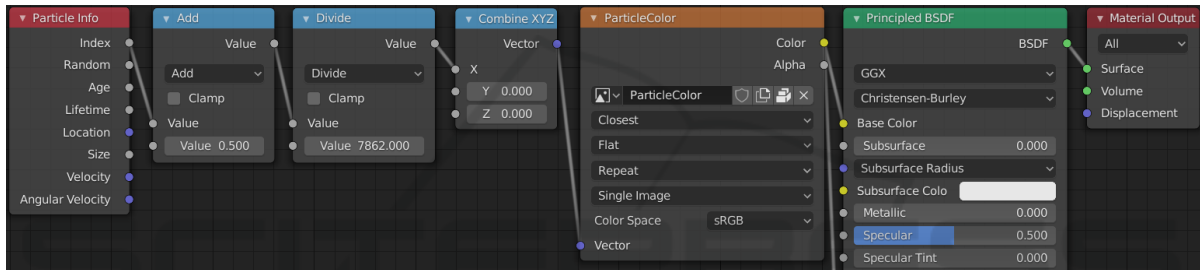

Figure 6: Node configuration created by the framework to define colors of the particles in the particle system. The light blue nodes use the particle index to compute the texture coordinate with the corresponding color - the value in the divide node represents the number of total particles. The principled BSDF node has been cut to increase the compactness of the figure.

allows to post-process the result with Blender's animation tools. The camera properties between two reconstructed camera poses are interpolated to enable the creation of smooth camera trajectories. Fig. 5 shows an example of the animated camera and the corresponding interpolated properties. The f-curves use quaternions to define the camera rotations of the animated camera. We normalize the quaternions representing the rotations of the reconstructed cameras to avoid unexpected interpolation results caused by quaternions with different signs, *i.e.* we ensure that the quaternions show consistent signs, which minimizes the distance between two consecutive quaternions.

2.4 Representation of Scene Geometry

Photogrammetry-based reconstruction techniques frequently use two types of entities to represent the reconstructed scene structure: point clouds and (textured) meshes. While meshes provide a more holistic representation of the scene geometry, point clouds are typically more accurate - since the reconstructed

points correspond to image correspondences.

Currently, there is no Blender entity that permits to directly represent colored point clouds. Our framework circumvents this problem by providing the following two point cloud representations: point clouds represented with Blender's *particle system* and point clouds visualized with *OpenGL* (Woo et al., 1999).

The particle system allows us to represent each 3D point with a single particle, which enables us to post-process and render the reconstructed result. We define the particle colors with Blender's *node system*. The proposed framework uses the ID of each particle to determine a texture coordinate of single texture containing all particle colors. The corresponding nodes are shown in Fig. 6.

In contrast to Blender's particle system, the drawing of point clouds with OpenGL is computationally less expensive and enables to visualize larger point clouds. Thus, it is better suited for the visualization of large point numbers such as point clouds representing the depth maps of multiple input images. Fig. 7 shows an example.
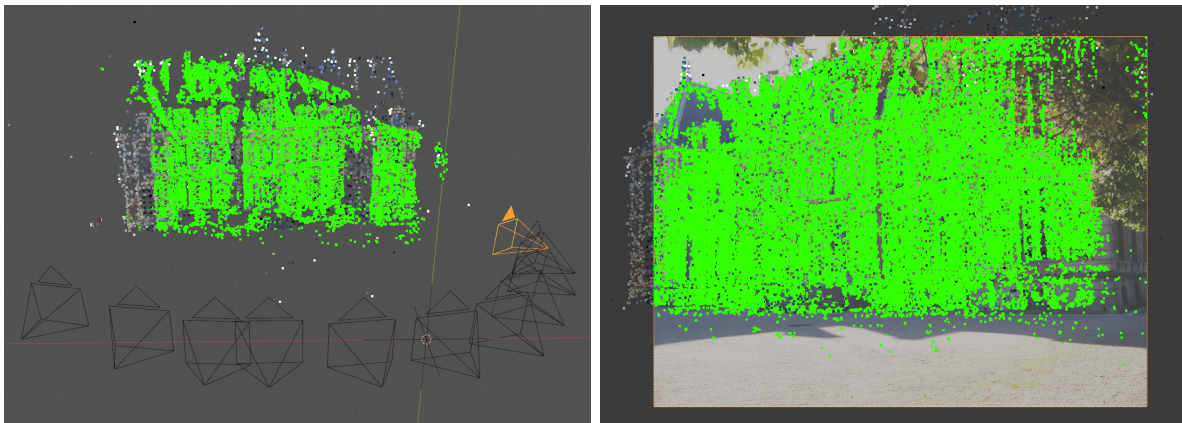
(a) Reconstructed point cloud and depth map of the rightmost camera highlighted with orange.

(b) Depth map from the perspective of one of the reconstructed cameras. The background image shows the input image used to register the camera.

Figure 7: Representation of a depth map using OpenGL. Both images show the triangulated points corresponding to the depth map of the same camera. Using Blender's 3D view allows us to assess the consistency of the depth map w.r.t. to the point cloud (see Fig. 7a) as well as the consistency of the depth w.r.t. to visual cues in source image (Fig. 7b).

We use Blender's built-in data structures to represent the reconstructed and potentially textured meshes. This allows us to integrate the triangulated scene geometry (*i.e.* points and meshes) in the same coordinate system as the registered cameras as shown in Fig. 3.

3 CONCLUSION

This paper proposes a publicly available extension of Blender that enables to integrate reconstructions of different SfM and MVS libraries to facilitate tasks such as image-based modeling and automatic creation of camera animations. We presented a general overview of the core components of modern SfM and MVS methods and provided a summary of widely used state-of-the-art reconstruction pipelines that are supported by our system. The paper provides a detailed description of the objects used to model the reconstructed cameras and the corresponding three-dimensional scene points as well as the representations used to integrate camera motion and additional image information. We showed several automatically computed examples that illustrate the usefulness of the presented framework. We are convinced that the application of SfM and MVS libraries is a crucial step to facilitate image-based modeling and camera tracking tasks. The analysis of the download statistics show that our Blender extension is adopted by a variety of artists and researchers.

REFERENCES

AliceVision (2020a). Meshroom: A 3D reconstruction software. https://github.com/alicevision/meshroom. [Accessed October 2020].

AliceVision (2020b). MeshroomMaya: A Maya plugin that enables to model 3D objects from images. https://github.com/alicevision/MeshroomMaya. [Accessed October 2020].

Attenborrow, S. (2020). Blender Photogrammetry. https://github.com/stuarta0/blender-photogrammetry. [Accessed October 2020].

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3).

Blender Online Community (2020). Blender - a 3D modelling and rendering package. http://www.blender.org. [Accessed October 2020].

Burt, P. J. and Adelson, E. H. (1983). A multiresolution spline with application to image mosaics. *ACM Trans. Graph.*, 2(4):217–236.

Cernea, D. (2020). OpenMVS: Multi-view stereo reconstruction library. https://cdcseacave.github.io/openMVS. [Accessed October 2020].

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R. D., and Erra, U., editors, *Eurographics Italian Chapter Conference*. The Eurographics Association.

Daniel Girardeau-Montaut (2020). Cloudcompare. http://www.cloudcompare.org/. [Accessed October 2020].

Fuhrmann, S. and Goesele, M. (2014). Floating scale surface reconstruction. *ACM Trans. Graph.*, 33(4).

Fuhrmann, S., Langguth, F., and Goesele, M. (2014). Mve: A multi-view reconstruction environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage*, GCH '14, page 11–18, Goslar, DEU. Eurographics Association.

Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multi-view stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. (2007). Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

Hiestand, R. (2020). Regard3D: A free and open source structure-from-motion program. [Accessed October 2020].

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814 vol. 2.

Jancosek, M. and Pajdla, T. (2014). Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces. *International Scholarly Research Notices*, 2014:1–20.

Kazhdan, M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*.

Knapitsch, A., Park, J., Zhou, Q.-Y., and Koltun, V. (2017). Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4).

Langguth, F., Sunkavalli, K., Hadap, S., and Goesele, M. (2016). Shading-aware multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Moulon, P. (2012). Sceaux castle dataset. https://github.com/openMVG/ImageDataset_SceauxCastle. [Accessed October 2020].

Moulon, P., Monasse, P., and Marlet, R. (2012). Adaptive structure from motion with a contrario model estimation. In *Asian Conference on Computer Vision (ACCV)*.

Moulon, P., Monasse, P., Marlet, R., and Others (2013). OpenMVG. An open multiple view geometry library. https://github.com/openMVG/openMVG/. [Accessed October 2020].

Schönberger, J. L. (2020). COLMAP: A general-purpose Structure-from-Motion and Multi-View Stereo pipeline. https://github.com/colmap/colmap. [Accessed October 2020].

Schönberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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)*.

SideEffects (2020). Game Development Toolset for Houdini. https://github.com/sideeffects/GameDevelopmentToolset. [Accessed October 2020].

Uhlík, J. (2020). Agisoft Photoscan Importer for Blender. https://github.com/uhlik/bpy. [Accessed October 2020].

Ummenhofer, B. and Brox, T. (2017). Global, dense multi-scale reconstruction for a billion points. *International Journal of Computer Vision*, pages 1–13.

Waechter, M., Moehrle, N., and Goesele, M. (2014). Let there be color! large-scale texturing of 3d reconstructions. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 836–850, Cham. Springer International Publishing.

Woo, M., Neider, J., Davis, T., and Shreiner, D. (1999). *OpenGL programming guide: the official guide to learning OpenGL, version 1.2.* Addison-Wesley Longman Publishing Co., Inc.

Wu, C. (2011). Visualsfm: A visual structure from motion system. http://ccwu.me/vsfm/.

Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847.*