EUROGRAPHICS Workshop on Graphics and Cultural Heritage (2014)
R. Klein and P. Santos (Editors)

MVE – A Multi-View Reconstruction Environment

Simon Fuhrmann Fabian Langguth Michael Goesele

TU Darmstadt, Germany

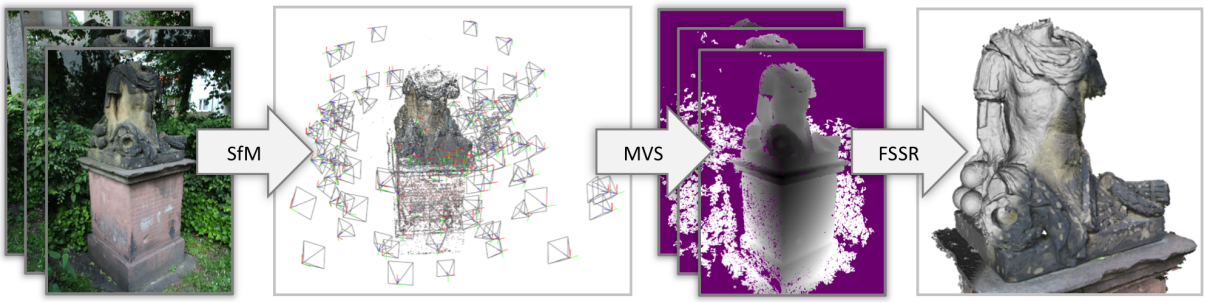

Figure 1: *Our multi-view reconstruction pipeline. Starting from input images, structure-from-motion (SfM) techniques are used to reconstruct camera parameters and a sparse set of points. Depth maps are computed for every image using multi-view stereo (MVS). Finally, a colored mesh is extracted from the union of all depth maps using a surface reconstruction approach (FSSR).*

Abstract

We present MVE, the Multi-View Environment. MVE is an end-to-end multi-view geometry reconstruction software which takes photos of a scene as input and produces a surface triangle mesh as result. The system covers a structure-from-motion algorithm, multi-view stereo reconstruction, generation of extremely dense point clouds, and reconstruction of surfaces from point clouds. In contrast to most image-based geometry reconstruction approaches, our system is focused on reconstruction of multi-scale scenes, an important aspect in many areas such as cultural heritage. It allows to reconstruct large datasets containing some detailed regions with much higher resolution than the rest of the scene. Our system provides a graphical user interface for structure-from-motion reconstruction, visual inspection of images, depth maps, and rendering of scenes and meshes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1 Introduction

Acquiring 3D geometric data from natural and man-made objects or scenes is a fundamental field of research in computer vision and graphics. 3D digitization is relevant for designers, the entertainment industry, and for the preservation as well as digital distribution of cultural heritage objects and sites. In this paper, we introduce *MVE*, the *Multi-View Environment*, a new, free software solution for low-cost geometry acquisition from images. The system takes as input a set of photos and provides the algorithmic steps necessary to obtain a high-quality surface mesh of the captured object as final output. This includes structure-from-motion (SfM), multi-view stereo (MVS) and surface reconstruction.

Geometric acquisition approaches are broadly classified into active and passive scanning. Active scanning technologies for 3D data acquisition exist in various flavors. Time of flight and structured light scanners are known to produce geometry with remarkable detail and accuracy. But these systems require special hardware and the elaborate capture setup is expensive. Real-time stereo systems such as the Kinect primarily exist for the purpose of gaming, but are often used for real-time geometry acquisition. These systems are based on structured light which is emitted into the scene.

DOI: 10.2312/gch.20141299

They are often of moderate quality and limited to indoor settings because of inference with sunlight. Finally, there is some concern that active system may damage objects of cultural value due to intense light emission.

Passive scanning systems do not emit light, are purely based on natural illumination, and will not physically harm the subject matter. The main advantage of these systems is the cheap capture setup which does not require special hardware: A consumer-grade camera (or just a smartphone) is usually enough to capture datasets. These systems are based on finding visual correspondences in the input images, thus the geometry is usually less complete, and scenes are limited to static, well textured surfaces. The inexpensive demands on the capture setup, however, come at the cost of much more elaborate computer software to process the unstructured input. The standard pipeline for geometry reconstruction from images involves three major algorithmic steps:

- Structure-from-Motion (SfM) reconstructs the extrinsic camera parameters (position and orientation) and the camera calibration data (focal length and radial distortion) by finding sparse but stable correspondences between images. A sparse point-based 3D representation of the subject is created as a byproduct of camera reconstruction.
- Multi-View Stereo (MVS), which reconstructs dense 3D geometry by finding visual correspondences in the images using the estimated camera parameters. These correspondences are triangulated yielding 3D information.
- Surface Reconstruction, which takes as input a dense point cloud or individual depth maps, and produces a globally consistent surface mesh.

It is not surprising that software solutions for end-to-end passive geometry reconstruction are rare. The reason lies in the technical complexity and the effort required to create such integrated tools. Many projects cover parts of the pipeline, such as *VisualSfM* [Wu13] or *Bundler* [SSS06] for structure-from-motion, *PMVS* [FP10] for multi-view stereo, and *Poisson Surface Reconstruction* [KH13] for mesh reconstruction. A few commercial software projects offer an end-to-end pipeline covering SfM, MVS, Surface Reconstruction and Texturing. This includes *Arc3D*, *Agisoft Photoscan* and *Acute3D Smart3DCapture*. We offer a complete pipeline as open source software system free for personal use and research purposes.

Our system gracefully handles many kinds of scenes, such as closed objects or open outdoor scenes. It avoids, however, to fill holes in regions with insufficient data for a reliable reconstruction. This may leave gaps in models but does not introduce artificial geometry, common to many global reconstruction approaches. Our software puts a special emphasis on multi-resolution datasets which contain both detailed and less detailed regions, and it has been shown that inferior results are produced if the multi-resolution nature of the input data is not considered properly [MKG11, FG11, FG14].

In the remainder of the paper, we will first give an overview of our system (Section 2). The practical applicability of our system is demonstrated in a hands-on guide in Section 3. We then show reconstruction results on several datasets in Section 4 and demonstrate the superiority of our pipeline in comparison to alternative state of the art software. We briefly describe our software framework in Section 5 and finally conclude in Section 6.

2 System Overview

Our system consists of three main steps, namely structure-from-motion (SfM) which reconstructs the parameters of the cameras, multi-view stereo (MVS) for establishing dense visual correspondences, and a final meshing step which merges the MVS geometry into a globally consistent, colored mesh. In the following, we give a concise overview of the process, using the dataset of *Anton's Memorial* as an example for a cultural heritage artifact, see Figure 1. For a more detailed explanation of the approaches, we refer the interested reader Szeliski's textbook [Sze10].

2.1 Structure-from-Motion

Structure-from-motion is one of the crowning achievements of photogrammetry and computer vision which started its roots in [AZB94, PKG98] and opened up to a wider audience in [PGV*03, SSS06]. In essence, SfM reconstructs the parameters of cameras solely from sparse correspondences in an otherwise unstructured image collection. The recovered camera parameters consist of the extrinsic calibration (i.e. the orientation and position of the camera), and the intrinsic calibration (i.e. the focal length and radial distortion of the lens). Although, at least in theory, the focal length can be fully recovered from the images for non-degenerate configurations, this process can be unstable. However, a good initial guess for the focal length is usually sufficient and can be optimized further.

Feature detection: First, machine-recognizable features are detected in the input images (Figure 2, left) and matched in order to establish sparse correspondences between images. Differences in the images require invariance of the features with respect to certain transformations, such as image scale, rotation, noise and illumination changes. Our system implements and jointly uses both *SIFT* [Low04] and *SURF* [BETVG08] features which are amongst the top performing features in literature.

Feature matching: The detected features are matched between pairs of images (Figure 2, right). Because corresponding points in two images are subject to the epipolar constraints of a perspective camera model [LF95], enforcing these constraints removes many false correspondences. The pairwise matching results are then combined and expanded over several views, yielding feature tracks. Each track corresponds to a single 3D point after SfM. Depending on the size

of the scene, matching can take a long time, because every image is matched to all other images, resulting in a quadratic algorithm. As an expedient, the state after matching (containing the feature detections and the pairwise matching) can be saved to file and reloaded later in case the remaining procedure is to be repeated under different parameters. This state is called the *prebundle*.

We also investigated in accelerating the matching time of our system. Common approaches include matching fewer features per image, reducing the number of pairs in a preprocessing step, or accelerating the matching itself using parallelism. We use a practical combination of the approaches: By matching a few low-resolution features, one can identify image pairs that potentially do not match, and reject the candidates before full-resolution matching is performed. Although low-resolution matching rejects some good image pairs, we could not observe any loss of quality in the reconstruction. It has been shown by Wu [Wu13] that this can considerably accelerate the matching time.

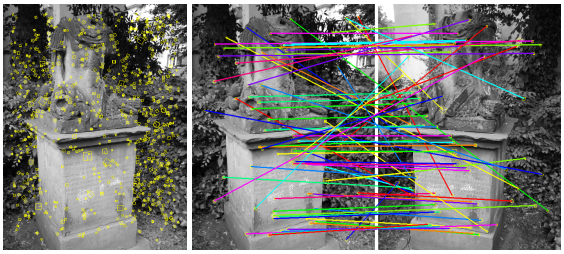

Figure 2: *Feature detection (left) and feature matching between two views (right). The horizontal lines are mostly good matches, and more slanted lines are outliers. Enforcing two-view constraints will remove most outliers.*

Incremental reconstruction: The relative pose of a good initial image pair is estimated, and all feature tracks visible in both images are triangulated. What follows is the incremental SfM, where suitable next views are incrementally added to the reconstruction, until all reconstructable views are part of the scene (Figure 3). Lens distortion parameters are estimated during reconstruction. The performance of subsequent algorithms is considerably improved by removing the distortion from the original images.

Not many software solutions for SfM have been published, probably because the theoretic concepts and algorithmic details are involved. Freely available software for this purpose includes *Bundler* [SSS06] and *VisualSfM* [Wu13].

2.2 Multi-View Stereo

Once the camera parameters are known, dense geometry reconstruction is performed. MVS algorithms exist in various flavors [SCD*06]. Some approaches work with volumetric representations [KBC12] and usually do not scale well to large datasets. Others reconstruct global point clouds,

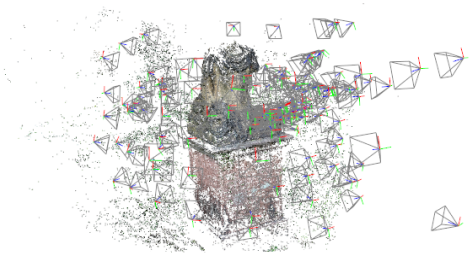

Figure 3: *Structure-from-motion reconstruction showing the sparse point cloud and the camera frusta.*

e.g. the popular *PMVS* implementation of Furukawa et al. [FP10]. Scalability issues further motivated work that clusters the scene into smaller manageable pieces [FCSS10]. Although *PMVS* is widely used, we aim at creating much denser point clouds for mesh reconstruction in order to preserve more details in the final result. A third line of work directly reconstructs global meshes [VLPK12] and couples MVS and surface reconstruction approaches in a mesh evolution framework.

We use the Multi-View Stereo for Community Photo Collections approach by Goesele et al. [GSC*07] which reconstructs a depth map for every view (Figure 4). Although depth map based approaches produce lots of redundancy because many views are overlapping and see similar parts of the scene, it effortlessly scales to large scenes as only a small set of neighboring views is required for reconstruction. In a way, this can be seen as an out-of-core approach to MVS. Another advantage of depth maps as intermediate representation is that the geometry is parameterized in its natural domain, and per-view data (such as color) is directly available from the images. The excessive redundancy in the depth maps can be a burden; not so much in terms of storage but processing power required for depth maps computation. On the positive side, this approach has proven to be capable of producing highly detailed geometry, and to overcome the noise in the individual depth maps [FG11, FG14].

2.3 Geometry Reconstruction

Merging the individual depth maps into a single, globally consistent representation is a challenging problem. The input photos are usually subject to large variations in viewing parameters. For example, some photos show a broad overview of the scene while others show small surface details. The depth maps inherit these multi-scale properties which leads to vastly different sampling rates of the observed surfaces.

Many approaches for depth map fusion have been proposed. The pioneering work by Curless and Levoy [CL96] renders locally supported signed distance fields (SDF) of the depth maps into a volumetric representation. Overlapping SDFs are averaged, which effectively reduces noise, but

Figure 4: *An input image and the corresponding depth map reconstructed with multi-view stereo. Each depth value encodes the distance from the camera.*

Figure 5: *The final surface reconstruction with color (left) and shaded (right). Notice how even the engraved text is visible in the geometry.*

also quickly eliminates geometric details if depth maps with different resolution are merged. To this end, Fuhrmann and Goesele [FG11] present a solution based on a hierarchical SDF which avoids averaging geometry at different resolutions. We use the follow-up work by Fuhrmann and Goesele [FG14]. They present a point-based reconstruction approach (*Floating Scale Surface Reconstruction*, *FSSR*), which additionally takes per-sample scale values as input. In contrast to point-based approaches that do not use scale, such as *Poisson Surface Reconstruction* [KH13], the method is able to automatically adapt the interpolation and approximation behavior depending on sample scale and redundancy without explicit parameter settings. An important aspect of *FSSR* is that it does not interpolate regions with insufficient geometric data. Instead, it leaves these regions empty which is useful for incomplete or open (outdoor) scenes. This stands in contrast to many global approaches that often hallucinate geometry, requiring manual cleanup.

In order to generate the input samples for *FSSR*, all depth maps are triangulated and colored using the input image. The connectivity information is used to compute a normal for each vertex. Additionally, the lengths of all edges emanating from a vertex are averaged and used as scale value for the vertex. The union of all vertices from all depth maps is then used as input to *FSSR*. An hierarchical, signed implicit function is constructed from the samples and the final surface (Figure 5) is extracted as the zero-level set of the implicit function using a hierarchical variant of the *Marching Cubes* algorithm [KKDH07].

3 Reconstruction Guide

We now demonstrate in a practical walkthrough of a reconstruction session starting with photo acquisition all the way to the final geometry. Specifically, we point out best practices and explain how to avoid common pitfalls.

Capturing photos: A dataset reconstructs best if a few simple rules are observed. First, in order to successfully reconstruct a surface region, it must be seen from at least five views. This is a requirement of the MVS algorithm to reliably triangulate any 3D position. Photos should thus be taken with a good amount of overlap. Usually, unless the dataset becomes really large, more photos will not hurt quality, but there is a tradeoff between quality and performance. As a rule of thumb, taking twice as many photos as one might think is a good idea. In order for triangulation to work, parallax is required. The camera should be re-positioned for every photo. (This is exactly opposite to how panoramas are captured, where parallax in the images must be avoided.) This is also important for SfM: Triangulating a feature track with insufficient parallax results in a small triangulation angle and a poorly conditioned 3D position. Figure 6 shows some input images of our exemplary dataset.

Figure 6: *Five out of 161 input photos of the* Anton's Memorial *dataset.*

Creating a scene: A view is a container that contains per-viewport data (such as images, depth maps and other data). A scene is a collection of views, which make up the dataset. A new scene is created using either the graphical interface of our software, *UMVE*, or the command line tool *makescene*. Technically, the scene appears as a directory in the file system (with the name of the dataset). It contains another directory `views/` with all views stored as files with the extension `.mve`. Creating a new scene will solely create the `views/` directory for now. Importing photos will create a `.mve` file for every photo. This process will also import meta information from the images (EXIF tags), which is required to get a focal length estimate for every photo. If EXIF tags are not

available, a default focal length will be assumed. This, however, can lead to SfM failures if the default value is a bad guess for the actual focal length. Figure 7 shows our graphical interface *UMVE* after importing images into a new scene.

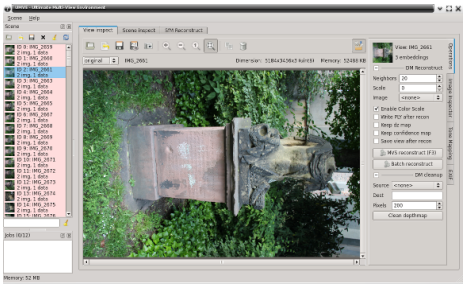

Figure 7: UMVE *after creating a new scene from images. The left pane contains a list of views. A view appears in red if no camera parameters have (yet) been reconstructed. The central part is the view inspector where the individual images of the selected view can be inspected. The right-hand side contains various contextual operations, such as UI elements to start the MVS reconstruction.*

SfM reconstruction: The SfM reconstruction can be configured and started using *UMVE*, or the command line tool *sfmrecon*. The UI guides through feature detection, pairwise matching and incremental SfM. What follows is the SfM reconstruction starting from an initial pair, and incrementally adding views to the reconstruction. Finally, the original images are undistorted and stored in the views for the next step. Figure 8 shows a rendering of the SfM reconstruction with the sparse point cloud and the camera frusta. Note how dense the frusta are spaced around the object to achieve a good reconstruction.

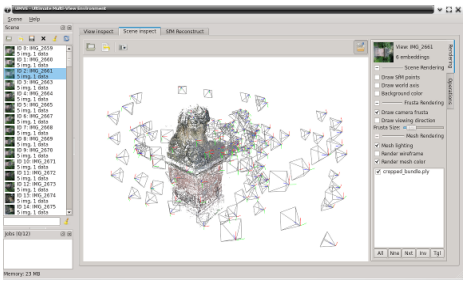

Figure 8: UMVE *rendering the SfM reconstruction. The central element is the 3D scene inspector. The right hand side offers various rendering options.*

MVS reconstruction: Given images with camera parameters, dense geometry is reconstructed using MVS. This can be done with either *UMVE* or the command line tool *dmrecon*. The most important parameter is the resolution level at which depth maps are reconstructed. A level of 0, or L_0, reconstructs at the original image size, L_1 corresponds to half the size (quarter the number of pixels), and so one. Looking at the resolution of recent digital cameras, a full-size L_0 reconstruction is rarely useful as finding dense correspondences gets more difficult, often leading to sparser depth maps at much higher computational cost. Using smaller images (we often use L_2), the process is faster and depth maps become more complete. See Figure 9 for a depth map computed at L_2.

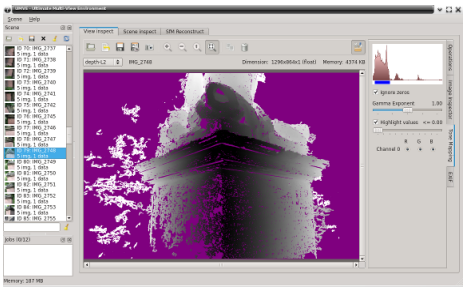

Figure 9: UMVE *displaying the reconstructed depth map. The mapping of depth values to intensities can be controlled using the histogram in the upper right corner. Purple pixels correspond the unreconstructed depth values.*

Surface reconstruction: The *scene2pset* tool combines all depth maps in a single, large point cloud. At this stage, a scale value is attached to every point which indicates the actual size of the surface region the point has been measured from. This additional information enables many beneficial properties using the *FSSR* surface reconstruction approach [FG14]. Then, the *FSSR* tools compute a multi-scale volumetric representation from the points (which does not require setting any explicit parameters) and a final surface mesh is extracted. The mesh can appear cluttered due to unreliable regions and isolated components caused by inaccurate measurements. The mesh is thus cleaned by deleting small isolated components, and removing unreliable regions from the surface. See Figure 10 for the final reconstruction.

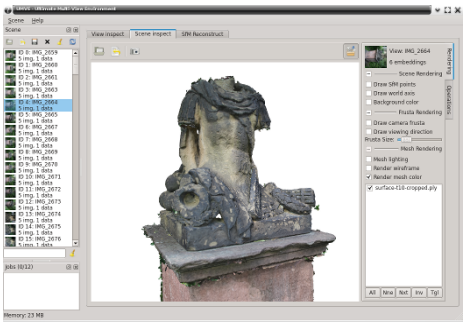

Figure 10: UMVE *rendering the reconstructed surface. The reconstructed vegetation has been manually cropped from the geometry with a bounding box in order to isolate the statue.*

Figure 11: *The* Der Hass *dataset. The top row shows 4 out of 79 input images and a depth map. The bottom row shows the reconstruction with color (left) and shaded (right).*

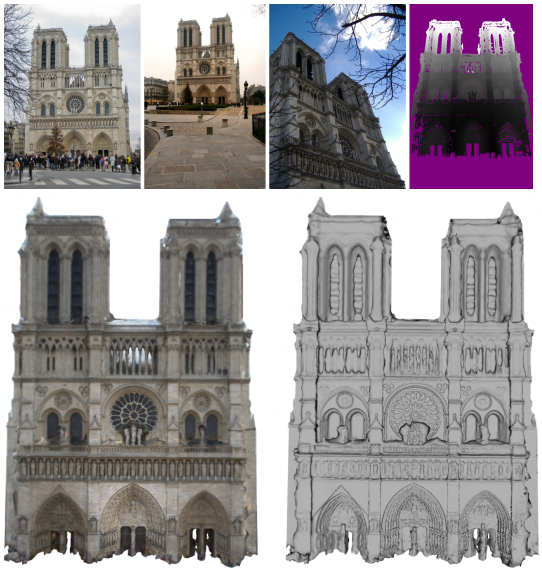

Figure 12: *The* Notre Dame *dataset. The top row shows 3 images and 1 depth map from a total of 131 input images. The bottom row shows the reconstruction with color (left) and shaded (right). Images taken by Flickr users* April Killingsworth*,* Alex Indigo *and* Maria Boismain*.*

4 Reconstruction Results

In the following, we show results on a few datasets we acquired over time. We report reconstruction times for all datasets in Table 1. A complete reconstruction usually takes several hours, depending on the size of the dataset and the available computing resources.

Der Hass: The first dataset is called *Der Hass* and contains 79 images of a massive stone sculpture, see Figure 11. This is a relatively compact dataset with uniform scale as the images have the same resolution and are evenly spaced around the object. Notice that although the individual depth maps contain many small holes, the final geometry is quite complete. Here, redundancy is key as all of our algorithms are completely local and no explicit hole filling is performed.

Notre Dame: Next, we reconstruct the façade of *Notre Dame* in Paris from 131 images downloaded from the Internet. We demonstrate that our pipeline is well suited even for Internet images: The features we use are invariant to many artifacts in the images, such as changing illumination. The MVS algorithm [GSC*07] uses a color scale to compensate for changing image appearance and is well suited for community photo collections. The surface reconstruction [FG14] handles the unstructured viewpoints well; it will, however, not produce a particularly good model colorization. This is because the original images have non-uniform appearance and color values are computed as per-vertex averages. In Figure 12, e.g., the portal appears slightly brighter as it has been reconstructed from brighter images.

Citywall: We conclude our demonstration with the *Citywall* dataset in Figure 13. The 363 input images depict an old historic wall with a fountain. This dataset demonstrates the multi-scale abilities of our system. While most of the views show an overview of the wall, some photos cover small details of the fountain. These details are preserved during reconstruction yielding a truly multi-resolution output mesh.

4.1 Runtime Performance

In Table 1, we present timings for all datasets in this paper. The reconstructions have been computed on an Intel Xeon Dual CPU system with 6×2.53GHz per CPU. Usually 4GB of main memory are sufficient for the smaller datasets. For large datasets, we recommend at least 8 GB of main memory (such as for the *Citywall* dataset, where surface reconstruction is quite demanding). Since most parts of the pipeline are parallelized, multiple CPUs will considerably improve the computation time. Currently we do not perform computations on the GPU as only few steps of our pipeline would benefit from GPU accelleration.

All datasets listed here use exhaustive matching, however, we investigated in reducing the time for feature matching. In order to quickly reject candidates that are unlikely to match, we first match a few hundred low-resolution features instead of the full set of featues. We evaluated this approach on the *Citywall*, the largest of our datasets. As can be seen in Table 1, this reduces the matching time by about $2/3$, rejecting about $2/3$ of all potential image pairs.

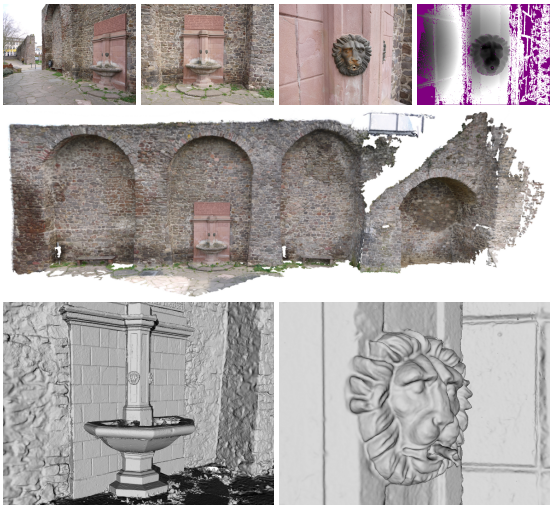

Figure 13: *The* Citywall *dataset. The top row shows 3 out of 363 input images and one depth map. The middle row shows the full reconstruction in color, and the bottom row shows the fountain and a small detail on the fountain.*

4.2 Limitations

A practical limitation in the presented system is the memory consumption in some parts of the pipeline. For example, surface reconstruction keeps all points in memory for the evaluation of the implicit function. Since our algorithm is purely local, we plan to implement out-of-core solutions to further scale the approach. Datasets with many images pose a bottleneck in runtime performance of SfM. This is because every image is matched to all other images, resulting in a quadratic algorithm in the number of images. Although low-resolution matching reduces runtime performance, our system is not limited to, but suitable for a few hundred images.

A general limitation of multi-view reconstruction approaches is the lack of texture on the geometry. Since stereo algorithms rely on variations in the images in order to find correspondences, weakly textured surfaces are hard to reconstruct. This is demonstrated in Figure 14. Although most of the depth map has been reconstructed, multi-view stereo fails on the textureless forehead in the relief.

Dataset	Images	SfM [min]	MVS [min]	FSSR [min]
Der Hass	79	33+5	61	17
Notre Dame	131	109+8	32	26
Anton Memorial	161	180+10	121	88
Citywall	363	945+56	267	203
Citywall LRM	363	345+55	254	181

Table 1: *Runtime performance for various datasets. The SfM timings are broken down into feature detection with matching and incremental SfM. The bottom row (*Citywall LRM*) shows the timing using low-resolution matching considerably reducing matching time.*

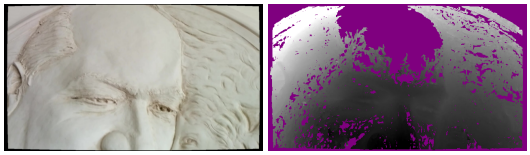

Figure 14: *One image of a dataset with a weakly textured relief, and the corresponding depth map. The depth map contains a big hole in the homogeneous region as visual correspondences cannot reliably be established.*

5 Software

The principles behind our software development make our code base a versatile and unique resource for practitioners (use it) and for developers/researchers (extend it). One notable example of an extension (also relevant in the context of cultural heritage) is the texturing approach by Waechter et al. [WMG14]. The core functionality of MVE is available as a small set of easy to use, cross-platform libraries. These libraries solely build upon `libjpeg`, `libtiff` and `libpng` for reading and writing image formats, and do not require any other external dependencies. We strive for a user-friendly API and to keep the code size at a maintainable minimum. The correctness of many components in our code is backed by unit tests. Our GUI application requires (aside from our own libraries) the widely used QT framework for the user interface. We ship with our libraries a few command line applications for the entire pipeline to support computation on server machines without a graphical interface. MVE is tested and operational under Linux, MacOS and Windows. The source code is available from our website[†].

6 Conclusion

In this paper we presented *MVE*, the *Multi-View Environment*, a free and open 3D reconstruction application, relevant to the cultural heritage community. It is versatile and can operate on a broad range of datasets. This includes the ability to handle quite uncontrolled photos and is thus suitable for reconstruction amateurs. Our focus on multi-scale data enables to put an emphasis on interesting parts in larger scenes with close-up photos. We belief that the effort and expert knowledge that went into *MVE* is an important contribution to the community.

[†] `http://www.gris.informatik.tu-darmstadt.de/projects/multiview-environment/`

Acknowledgements

Part of the research leading to these results has received funding from the European Commission's FP7 Framework Programme under grant agreements ICT-323567 (HARVEST4D) and ICT-611089 (CR-PLAY), as well as the DFG Emmy Noether fellowship GO 1752/3-1.

References

[AZB94] ARMSTRONG M., ZISSERMAN A., BEARDSLEY P. A.: Euclidean Reconstruction from Uncalibrated Images. In *British Machine Vision Conference (BMVC)* (1994), pp. 509–518. 2

[BETVG08] BAY H., ESS A., TUYTELAARS T., VAN GOOL L.: Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding (CVIU) 110*, 3 (June 2008), 346–359. 2

[CL96] CURLESS B., LEVOY M.: A Volumetric Method for Building Complex Models from Range Images. In *Proceedings of ACM SIGGRAPH* (1996), pp. 303–312. 3

[FCSS10] FURUKAWA Y., CURLESS B., SEITZ S. M., SZELISKI R.: Towards Internet-scale Multi-view Stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2010). 3

[FG11] FUHRMANN S., GOESELE M.: Fusion of Depth Maps with Multiple Scales. In *Proceedings of ACM SIGGRAPH Asia* (2011), pp. 148:1 – 148:8. 2, 3, 4

[FG14] FUHRMANN S., GOESELE M.: Floating Scale Surface Reconstruction. In *Proceedings of ACM SIGGRAPH* (2014). 2, 3, 4, 5, 6

[FP10] FURUKAWA Y., PONCE J.: Accurate, Dense, and Robust Multi-View Stereopsis. *Transactions on Pattern Analysis and Machine Intelligence (PAMI) 32*, 8 (2010), 1362–1376. 2, 3

[GSC*07] GOESELE M., SNAVELY N., CURLESS B., HOPPE H., SEITZ S.: Multi-View Stereo for Community Photo Collections. In *International Conference on Computer Vision (ICCV)* (Oct 2007), pp. 1–8. 3, 6

[KBC12] KOLEV K., BROX T., CREMERS D.: Fast Joint Estimation of Silhouettes and Dense 3D Geometry from Multiple Images. *Transactions on Pattern Analysis and Machine Intelligence (PAMI) 34*, 3 (2012), 493–505. 3

[KH13] KAZHDAN M., HOPPE H.: Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics 32*, 3 (2013), 29:1–29:13. 2, 4

[KKDH07] KAZHDAN M., KLEIN A., DALAL K., HOPPE H.: Unconstrained Isosurface Extraction on Arbitrary Octrees. In *Eurographics Symposium on Geometry Processing (SGP)* (2007). 4

[LF95] LUONG Q.-T., FAUGERAS O.: The Fundamental matrix: theory, algorithms, and stability analysis. *International Journal of Computer Vision (IJCV) 17* (1995), 43–75. 2

[Low04] LOWE D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV) 60*, 2 (Nov. 2004), 91–110. 2

[MKG11] MÜCKE P., KLOWSKY R., GOESELE M.: Surface Reconstruction from Multi-Resolution Sample Points. In *Vision, Modelling and Visualization (VMV)* (2011). 2

[PGV*03] POLLEFEYS M., GOOL L. V., VERGAUWEN M., CORNELIS K., VERBIEST F., TOPS J.: 3D Recording for Archaeological Field Work. *Computer Graphics and Applications (CGA) 23*, 3 (2003), 20–27. 2

[PKG98] POLLEFEYS M., KOCH R., GOOL L. V.: Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters. In *International Conference on Computer Vision (ICCV)* (1998), pp. 90–95. 2

[SCD*06] SEITZ S., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2006). 3

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo Tourism: Exploring Photo Collections in 3D. *ACM Transactions on Graphics 25*, 3 (2006), 835–846. 2, 3

[Sze10] SZELISKI R.: *Computer Vision: Algorithms and Applications*. Springer, 2010. 2

[VLPK12] VU H.-H., LABATUT P., PONS J.-P., KERIVEN R.: High Accuracy and Visibility-Consistent Dense Multiview Stereo. *Transactions on Pattern Analysis and Machine Intelligence (PAMI) 34*, 5 (2012), 889–901. 3

[WMG14] WAECHTER M., MOEHRLE N., GOESELE M.: Let there be color! — Large-scale texturing of 3D reconstructions. In *Proceedings of the European Conference on Computer Vision* (2014). 7

[Wu13] WU C.: Towards Linear-Time Incremental Structure from Motion. In *International Conference on 3D Vision (3DV)* (June 2013), pp. 127–134. 2, 3