MVE – An Image-Based Reconstruction Environment

Simon Fuhrmann, Fabian Langguth, Nils Moehrle, Michael Waechter and Michael Goesele

Graphics, Capture and Massively Parallel Computing – TU Darmstadt – Germany

Abstract

We present an image-based reconstruction system, 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 textured surface mesh as result. The system covers a structure-from-motion algorithm, multi-view stereo reconstruction, generation of extremely dense point clouds, reconstruction of surfaces from point clouds, and surface texturing. 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 visual inspection of the individual steps of the pipeline, i.e., the structure-from-motion result, multi-view stereo depth maps, and rendering of scenes and meshes.

Keywords: Image-Based Reconstruction, Structure-from-Motion, Multi-View Stereo, Surface Reconstruction, Texturing

1. Introduction

Acquiring 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 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, multi-view stereo, surface reconstruction and texturing.

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 expensive hardware and elaborate capture planning and execution. 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 infra-red light which is emitted into the scene. They are often of moderate quality and limited to indoor settings because of inference with sunlight's infrared component. Finally, there is some concern that active systems may damage objects of cultural value due to intense light emission.

Passive scanning systems do not emit light, are purely based on the existing illumination, and will not physically affect 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 enough to capture datasets. The reconstruction process is based on finding visual correspondences in the input images, which, compared to active systems, usually leads to less complete geometry, and limits the scenes 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 four major algorithmic steps (see Figure 1):

- Structure-from-Motion (SfM) infers the extrinsic camera parameters (position and orientation) and the camera calibration (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 by-product of camera reconstruction.
- Multi-View Stereo (MVS) reconstructs dense 3D geometry by finding visual correspondences in the images using the estimated camera parameters. These correspondences are triangulated yielding dense 3D information.
- Surface Reconstruction takes as input a dense point cloud or individual depth maps and produces a globally consistent surface mesh.
- Surface Texturing computes a consistent texture for the surface mesh using the input images.

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 tools. Many projects cover parts of the pipeline, such as *Bundler* [1], *VisualSfM* [2], or *OpenMVG* [3] for structure-from-motion reconstruction, *PMVS* [4] for multi-view stereo, and *Poisson Surface Reconstruction* [5] for mesh reconstruction. A few commercial software projects offer complete end-to-end pipelines

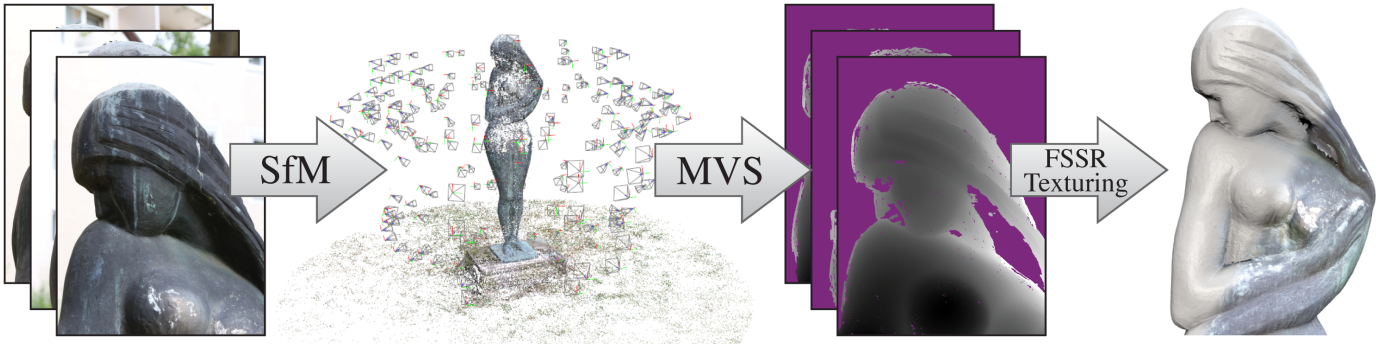

Figure 1: Our multi-view reconstruction pipeline. Starting from input images, structure-from-motion (SfM) techniques are used to reconstruct camera parameters and a sparse set of points. Depth maps are computed for every image using multi-view stereo (MVS). Finally, a colored mesh is extracted from the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

covering SfM, MVS, Surface Reconstruction and Texturing. This includes *Arc3D*, *Agisoft Photoscan* and *Acute3D Smart-3DCapture*. All of them are, however, closed source and do not facilitate research. In contrast, we offer a complete pipeline as a free, open source software system, which was introduced in an earlier version of this paper [6].

Our system handles many kinds of scenes, such as compact objects, open outdoor scenes, and controlled studio datasets. It avoids to fill holes in regions with insufficient data for a reliable reconstruction. This may leave holes in the surfaces but does not introduce artificial geometry, common to many global reconstruction approaches. Our software puts a special emphasis on multi-resolution datasets which can contain very detailed regions in otherwise less detailed datasets. It has been shown that inferior results are produced if the multi-resolution nature of the input data is not considered properly [7, 8, 9].

In the paper's remainder we first give a technical overview of our system and introduce its individual components in Section 2. A few practical aspects and limitations of our system are discussed in Section 3. We then show reconstruction results on several datasets with different characteristics and demonstrate the versatility of our pipeline in Section 4. We briefly describe our software framework and conclude in Section 5.

2. System Overview

Our system consists of four steps: Structure-from-motion (SfM) which reconstructs the parameters of the cameras, multi-view stereo (MVS) for establishing dense visual correspondences, a meshing step which merges the MVS geometry into a globally consistent mesh and finally a texturing step creating seamless textures from the input images. In the following, we give a concise overview of the process, using the *Bronze Statue* dataset as an example for a cultural heritage artifact, see Figure 1. For a more detailed explanation of the theoretic background of these approaches we refer the interested reader to Szeliski's textbook [10].

2.1. Structure-from-Motion

SfM is one of the crowning achievements of photogrammetry and computer vision. Its foundations were laid by Armstrong et al. [11] and Pollefeys et al. [12] and it was opened up to a wider audience by Pollefeys et al. [13] and the seminal Photo Tourism paper [1]. Not many software solutions for SfM have been published, probably because the theoretic background and the algorithmic details are involved. Freely available software for this purpose includes *Bundler* [1], *VisualSfM* [2] and *OpenMVG* [3]. In essence, SfM reconstructs the parameters of cameras solely from sparse correspondences in an otherwise unstructured image collection. The recovered camera parameters are the extrinsic calibration (i.e., camera orientation and position), and the intrinsic calibration (i.e., focal length and radial distortion of the lens). The SfM reconstruction pipeline is subdivided into several individual steps, illustrated in Figure 2. These steps are now explained in more detail.

Feature detection. The first step is to detect features in each input image (Figure 3, left). Our system implements and jointly uses both *SIFT* [14] and *SURF* [15], which are among the top performing features in literature. These algorithms first search for points of interest in the images which are potentially discriminative from each other (at least for a machine). A neighborhood around these points is extracted and stored within a feature descriptor. Variations in the images require invariance of the feature descriptors with respect to certain transformations, such as image scale, rotation, noise, exposure and contrast changes.

Feature matching. Next, the feature descriptors are matched between pairs of images (Figure 3, right) by finding for each descriptor a corresponding descriptor in the second image with small Euclidean distance, which amounts to a nearest neighbor search in a high-dimensional space. Because corresponding points in two images are subject to the epipolar constraints of a perspective camera model (described by the fundamental matrix [16]), filtering the matches by enforcing these constraints removes many false correspondences. Matching can take a long time because every image is matched to all other images, resulting in runtime that is quadratic in the number of images. As an expedient in our system, the pairwise matching information can be saved to a file and reloaded later in case SfM is repeated with different parameters. This state is called the *prebundle*.

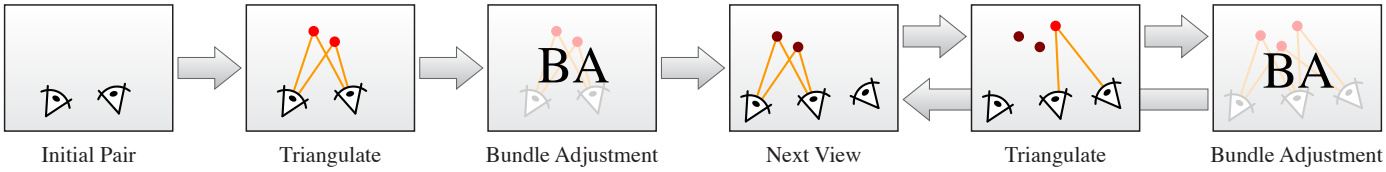

Figure 2: Incremental SfM: Starting from the initial camera pair, all pairwise matches of the pair are triangulated. The camera poses and the 3D points are then optimized using bundle adjustment. Suitable new cameras are incrementally added to the reconstruction, new tracks are triangulated and bundle adjustment is performed.

Figure 3: Feature detection (left) and feature matching between two views (right). The horizontal lines are mostly good matches, while the heavily slanted lines are outliers. Enforcing two-view constraints will remove most outliers.

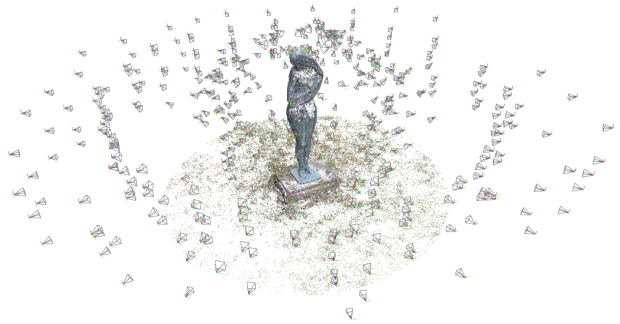

Figure 4: Structure-from-motion reconstruction showing the final 3D point cloud and all of the 437 camera poses. The SfM point cloud has been cropped for visual clarity to remove background points.

Accelerating matching. We also investigated accelerating the matching time of our system. Common approaches include matching fewer features per image, reducing the number of pairs in a pre-processing step, or accelerating the matching itself using parallelism. We use a practical combination of these approaches: By matching a few low-resolution features, one can quickly identify image pairs that potentially do not match, and reject the candidates before a matching of all features is performed. It has been shown by Wu [2] that this can considerably accelerate the matching time. Although low-resolution matching rejects some good image pairs, we could not observe a loss of reconstruction quality.

Track generation. The pairwise matching results are then combined and expanded over several views, yielding feature tracks. During the next steps, each track will be triangulated and yields a single 3D point in the reconstruction.

Initial Pair. The incremental reconstruction is bootstrapped using an initial camera pair. It is important to select this pair carefully to avoid degenerate configurations. In general, a good pair has many pairwise matches, but also a good amount of parallax. If the motion between the two cameras is too small, the triangulation becomes unstable and leads to badly conditioned 3D points. Also, if many pairwise matches correspond to a planar region in the scene, the camera's focal length becomes indistinguishable from scene depth. Both must be avoided and can be detected by fitting a homography to all feature matches of the pair. If a majority of the feature matches can be explained by a homography, the motion between the cameras is small or the features are in a degenerate planar configuration, and the camera pair should be avoided. Once an initial pair is selected, relative camera poses are extracted from the fundamental matrix and the pairwise matches are triangulated into 3D points.

Bundle Adjustment. The triangulated 3D points and the parameters of the cameras are not optimal with respect to the geometric distance between the projection of the 3D points and the original feature observations. This distance is called the *reprojection error* and minimizing it attains the Maximum Likelihood estimate under the assumption of Gaussian noise in the feature observations. The goal of Bundle Adjustment is to globally and jointly refine the camera parameters and 3D point positions by minimizing the sum of reprojection errors. This leads to a high-dimensional, non-linear, least-squares optimization problem and specialized solvers have been developed. We use *PBA* by Wu et al. [17] in our system.

Incremental reconstruction. In the following process new cameras and new 3D points are incrementally added to the reconstruction. The next best camera is chosen as the view with the highest number of already reconstructed tracks. The pose of this camera is then estimated from the correspondences between 3D points and image features using the *Perspective 3-Point* algorithm [18]. New tracks become available for triangulation and all parameters are again optimized by Bundle Adjustment. These steps are iterated until all cameras are reconstructed, or no suitable new camera can be found. Figure 4 shows the final state of a reconstruction with the camera frusta and a sparse set of 3D points.

2.2. Multi-View Stereo

Once the camera parameters are known, dense geometry reconstruction is performed. MVS algorithms exist in various

Figure 5: One of the input images and the corresponding depth map reconstructed with multi-view stereo. Each depth value encodes the distance from the camera center to the geometry.

flavors [19]. Some approaches work with volumetric representations [20] but usually do not scale well to large datasets. Others reconstruct global point clouds, e.g., the popular *PMVS* implementation of Furukawa et al. [4]. The scalability issues of this technique further motivated work that clusters the scene into smaller, more manageable pieces [21]. 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 [22] 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. [23] which reconstructs a depth map for every view (Figure 5). For a given reference view the algorithm first selects a set of 20 neighboring views according to parallax, overlap, and image resolution in a global view selection. The depth map of the reference view is initialized using the depth values of the sparse SfM points. These initial depth values are first refined in an optimization procedure, and then propagated to neighboring pixels in a region-growing fashion. Unrefined depth values are processed in order of their reliability, starting from the most reliable pixels. The depth refinement chooses a suitable subset of 4 neighboring views with a good parallax distribution and a high photo-consistency. The latter is measured by reprojecting a 5×5 patch around the pixel into the neighboring images and computing the normalized cross correlation (NCC). The final, optimized depth value for the pixel then maximizes the photo-consistency score for all neighboring views.

Although the resulting depth maps contain a lot of redundancy because of the large overlaps in the views, the approach effortlessly scales to large scenes: Only a small set of neighboring views is required for reconstructing a single depth map. In a way, this can be seen as an out-of-core approach to MVS. The excessive redundancy in the depth maps can be a burden; not so much in terms of storage but processing power required for depth map 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 [8, 9]. Another advantage of depth maps is that per-view data (such as color) is directly available from the images.

Figure 6: The final surface rendered with shading and with texture.

2.3. Surface 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 [24] renders locally supported signed distance fields (SDF) of the depth maps into a volumetric representation. Overlapping SDFs are averaged, which effectively reduces noise, but also quickly eliminates geometric details if depth maps with different resolution are merged. Fuhrmann and Goesele [8] 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 [9]. 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* [5], the method is able to automatically adapt the interpolation and approximation behavior depending on sample scale and redundancy without explicit parameter settings.

In order to generate the input samples for *FSSR*, each depth map is 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*.

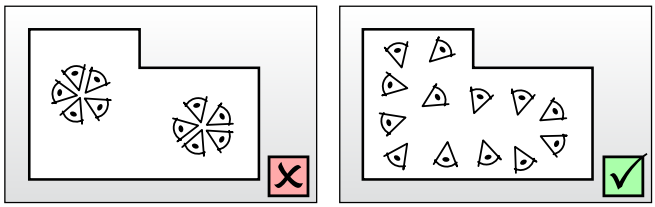

Figure 7: In room-like environments, it is important to move between every photo and to avoid capturing photos in a panorama fashion. Moving between the images ensures parallax and well conditioned 3D points.

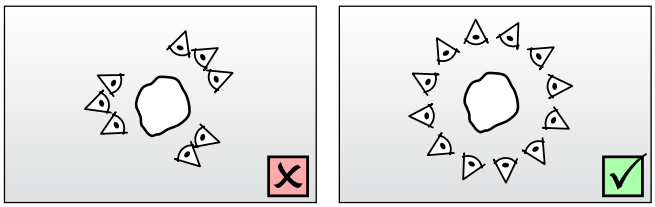

Figure 8: A densely sampled spiral around compact objects with a large overlap between the photos leads to the best results. A sparse sampling can lead to disconnected components in SfM or holes in the MVS reconstruction.

FSSR inserts all samples into an octree hierarchy. The octree is built dynamically, expanding and growing as more samples are added. Each sample is inserted in an appropriate level where the sample's scale is approximately as large as the side-length of the octree node. Voxels are then generated in the corners of the octree leaf nodes and a hierarchical, signed implicit function is constructed from the samples and evaluated at the voxel positions. The final surface is extracted as the zero-level set of the implicit function using a hierarchical variant of the *Marching Cubes* algorithm [25], see Figure 6.

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. Because scale values are known, *FSSR* is capable of analyzing whether an increased sample density is caused by an increase in surface resolution, or by sample redundancy, e.g., because many depth maps are overlapping in a certain region. This property allows *FSSR* to use the available redundancy for noise reduction, and prevents fitting to the noise in the input data.

2.4. Surface Texturing

Surface texturing uses the input photos with the associated camera parameters in order to create a globally consistent texture for the surface, see Figure 6 (right). This process usually generates one or several texture atlases and assigns texture coordinates to every vertex of the mesh. Generating a globally consistent and seamless texture is challenging due to changes in illumination, exposure time or white balancing in the input photos. Furthermore, the photos may contain objects in the foreground, such as tourists or cars, occluding the object of interest.

There mainly exist two different lines of work in texturing literature. The first line of work uses blending of multiple views in order to generate a final texture for every surface element [26, 27]. The blending, however, can lead to blurring of surface details. The other line of work uses a single input photo to texture a surface region [28, 29], which avoids blurring but may lead to visible seams between the regions caused by exposure or white balance differences. The latter approaches thus employ a color adjustment step to every texture region to eliminate the seams.

We use Waechter et al.'s work [30] (*Let There Be Color! Large-Scale Texturing of 3D Reconstructions*), which uses a single view to texture surface regions. This work is mainly based on an approach by Lempitsky and Ivanov [28] but uses an improved view selection that avoids blurred, out-of-focus image regions and employs a photo-consistency check to detect occluders in the input images. In a two-step color adjustment, colors are first globally adjusted on a per-vertex basis followed by local per-pixel adjustment using Poisson Image Editing [31].

3. Practical Aspects

In this section we discuss some aspects that should be considered when using our image-based reconstruction system. We present some guidelines that can help users to capture better input data in order to facilitate high quality results. We also discuss some limitations of the presented approaches, which do not only apply to our reconstruction system but more generally to these types of algorithms.

3.1. Capturing Photos

The capture session is a vital part of the reconstruction process which fundamentally influences the final quality and coverage of the scene. A very common problem is that too few photos are captured, which leads to a sparse coverage of the scene and can result in failures of both the SfM and the MVS reconstruction.

Visual Overlap. In order to robustly estimate the 3D position of any point on the surface of the scene, it has to be observed by at least five different cameras. This requirement originates from the MVS algorithm that tries to find for every pixel in the reference view a correspondence in at least four other views. Although a reconstruction with fewer neighboring views is in theory possible, it will reduce robustness and may lead to more noisy results. More overlap also leads to a denser and more accurate SfM reconstruction that is less likely to fail or leave out isolated views. Usually more photos will not hurt quality, but there is a trade-off between quality and the required reconstruction time. As a rule of thumb, it is a good idea to take twice as many photos as one might think is enough.

Camera Parallax. Besides a large overlap in the photos, parallax is required for a stable triangulation. The camera should be re-positioned for every photo and parallax in both horizontal and vertical direction is desirable. (This is exactly opposite to how panoramas are captured, where parallax in the images

Figure 9: Two sides of the *Arc de Triomphe* in Paris. Front and back are so similar that feature matching produces false correspondences connecting the two sides. Photo credits: Flickr users *trawets1* and *skding*.

must be avoided.) This is again important for SfM and MVS: Triangulating a feature track or the depth of a pixel with insufficient parallax results in a small triangulation angle and a poorly conditioned 3D position. Figure 7 and 8 illustrate good and bad camera distributions.

Light and Color. While overlap and parallax are mainly important for accurate surface estimation, the lighting in the scene and the camera settings can also influence the final result. Considerable changes in exposure and white balance will degrade the quality of the per-vertex color produced by surface reconstruction and also the final textured result. The camera settings should be kept as constant as possible over the capture session. As the scene illumination is baked into the photos and ultimately also into the textures, changing the lighting relative to the scene will also have a negative effect on the result. This happens, e.g., if an object is placed on a turntable and rotated during acquisition while keeping the lights constant. Hard shadows, e.g., if the scene is subject to direct sunlight, can produce unpleasant transitions on the object, and it is preferable to capture on an overcast day in order to avoid these shadow boundaries.

3.2. General Limitations

Structure from Motion. The feature matching step of SfM is designed to find correspondences between pairs of images. Each feature is expressed in terms of the image gradient, which is affected by both lighting and, more importantly, texture. Weakly textured objects will lead to fewer features and can result in fewer correspondences and more outliers. If feasible, placing additional textured targets in the scene often improves the results. Another common problem are repetitive structures in the scene, such as buildings with identical windows or walls. Thus features become ambiguous and may lead to false matches. See Figure 9 for an architectural object with repetitive appearance.

Multi-View Stereo. MVS estimates geometry by finding corresponding pixels in neighboring images. These correspondences are established by using photometric consistency measures on small image patches. For weakly textured surfaces the image patches are likely to contain insufficient color variation to identify unique correspondences. As a result, geometry estimation

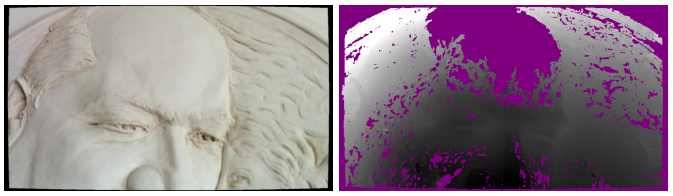

Figure 10: One image of a dataset with a weakly textured relief, and the corresponding depth map. The depth map contains a big hole in the homogeneous region because visual correspondences cannot be established reliably.

can fail in flat regions with uniform color, see Figure 10 for an example. The photometric consistency also assumes that the object's appearance is independent of the viewing direction, which is only true for diffuse surface materials. In case of specular materials, highlights move depending on the camera position and correspondences become inaccurate. Usually, this leads to more noise in the reconstruction.

Texturing. The texturing algorithm adjusts colors under the assumption of Lambertian materials and static lighting. Varying lighting situations (e.g., moving shadows, day/night illumination, colored lights) will have a negative effect on the generated texture. Surface specularity is problematic for the same reason, because it introduces local appearance changes. To some extent, the photo consistency check, which was originally designed to detect occluders, can help with these local variations. However, it is still required that the majority of the views observe a consistent surface.

4. Reconstruction Results

In the following, we show results on a few datasets we acquired over time. We selected a variety of scenarios to show the broad applicability of our system.

Duck. The first dataset, called *Duck*, was captured in a controlled studio environment and contains 160 images of a small, diffuse ceramic duck figurine, 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.

Trevi Fountain. Next, we reconstruct Rome's *Trevi Fountain* from 871 images downloaded from the Internet. We demonstrate that our pipeline is well suited even for uncontrolled Internet images: The features we use are invariant to many artifacts in the images, such as changing illumination. The MVS algorithm [23] uses a color scale to compensate for changing image appearance and is well suited for community photo collections. The surface reconstruction [9] handles the unstructured viewpoints well. The texturing, however, does not produce a particularly good result because the original images have very non-uniform appearances. In Figure 12 the right side shows a lot of different, inconsistent colors from the uncontrolled images.

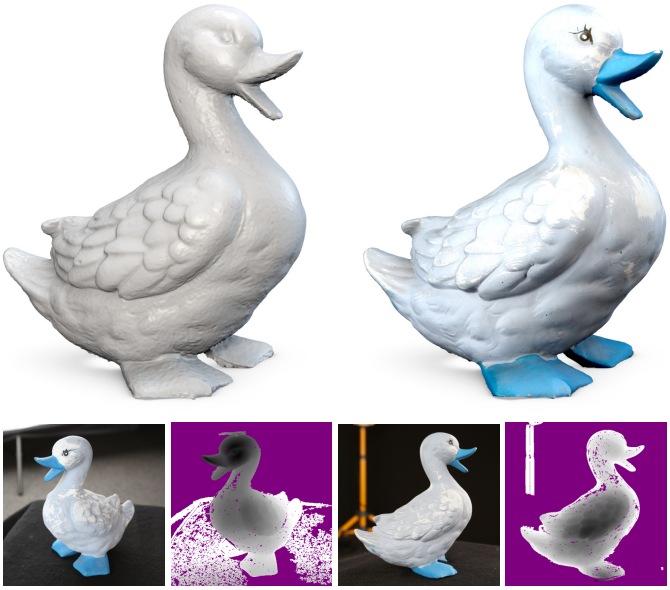

Figure 11: The *Duck* dataset. The bottom row shows 2 out of 160 input images and the corresponding depth maps. The top row shows the reconstruction with shading (left) and texture (right).

Figure 12: The *Trevi Fountain* dataset. The bottom row shows 3 images of a total of 871 input images. The top row shows the reconstruction rendered with shading (left) and with texture (right). Photo credits: Flickr users *Vince O'Sullivan*, *Andy Hay*, *Ecyrd*, Creative Commons License.

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

Actual runtime performance and memory consumption depends on various factors. For example, SfM runtime is dominated by the number of image features, MVS mostly depends on the image size, *FSSR* on the amount of surface samples and octree structure, and Texturing on the number of faces and size of the input images.

In Table 1 we present timings for all datasets in this paper. We decimated most meshes before texturing as the algorithm runs much faster if fewer faces need to be processed. However, decimating multi-scale datasets like the *Citywall* is significantly harder compared to our other datasets because high-resolution geometry can be destroyed in the process. We therefore textured this dataset at full resolution.

The reconstructions have been computed on an Intel Xeon Dual CPU system with 8×2.6 GHz per CPU. Usually 4 GB 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 multi-scale surface reconstruction is quite demanding). Our system is neither optimized for runtime performance nor memory consumption but most parts of the pipeline are parallelized and multiple CPUs will considerably improve the computation time. Currently, we do not perform computations on the GPU as only a few steps of our pipeline would benefit from GPU acceleration.

5. 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, including the ability to handle quite uncontrolled photos. It is thus suitable for reconstruction amateurs. Our focus on multi-scale data allows 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.

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). We strive for a user-friendly API and to keep the code size and library dependencies at a maintainable minimum. Our GUI application requires (aside from our own libraries) the widely used Qt framework for the user interface. We ship our software with com-

Dataset	Images	SfM [min]	MVS [min]	FSSR [min]	Texturing [min]
Duck	160	2+4	22	7	2
Citywall	363	130+29	132	134	58
Citywall*	363	363+28	131	133	56
Bronze	437	291+37	166	108	6
Trevi	871	304+107	149	197	6

Table 1: Runtime performance for various datasets. The SfM timings are broken down into feature detection with matching and incremental SfM. The *Citywall** row shows the timing using exhaustive matching, which is considerably slower than our accelerated matching procedure. All meshes except *Citywall* have been decimated before texturing.

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.

mand line applications for the entire pipeline to support computation on server machines without a graphical interface. MVE is tested on Linux, MacOS X and Windows. The source code is available from our website `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), the DFG Emmy Noether fellowship GO 1752/3-1 as well as the Intel Visual Computing Institute (Project RealityScan).

References

[1] N. Snavely, S. M. Seitz, R. Szeliski, Photo Tourism: Exploring Photo Collections in 3D, Transactions on Graphics 25 (3) (2006) 835–846.

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

[3] P. Moulon, P. Monasse, R. Marlet, and others, OpenMVG, `https://github.com/openMVG/openMVG`.

[4] Y. Furukawa, J. Ponce, Accurate, Dense, and Robust Multi-View Stereopsis, Transactions on Pattern Analysis and Machine Intelligence (PAMI) 32 (8) (2010) 1362–1376.

[5] M. Kazhdan, H. Hoppe, Screened Poisson Surface Reconstruction, Transactions on Graphics 32 (3) (2013) 29:1–29:13.

[6] S. Fuhrmann, F. Langguth, M. Goesele, MVE – A Multi-View Reconstruction Environment, in: Eurographics Workshop on Graphics and Cultural Heritage (GCH 2014), 2014.

[7] P. Mücke, R. Klowsky, M. Goesele, Surface Reconstruction from Multi-Resolution Sample Points, in: Vision, Modelling and Visualization (VMV), 2011.

[8] S. Fuhrmann, M. Goesele, Fusion of Depth Maps with Multiple Scales, in: SIGGRAPH Asia, 2011, pp. 148:1 – 148:8.

[9] S. Fuhrmann, M. Goesele, Floating Scale Surface Reconstruction, in: SIGGRAPH, 2014.

[10] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010.

[11] M. Armstrong, A. Zisserman, P. A. Beardsley, Euclidean Reconstruction from Uncalibrated Images, in: British Machine Vision Conference (BMVC), 1994, pp. 509–518.

[12] M. Pollefeys, R. Koch, L. V. Gool, 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.

[13] M. Pollefeys, L. V. Gool, M. Vergauwen, K. Cornelis, F. Verbiest, J. Tops, 3D Recording for Archaeological Field Work, Computer Graphics and Applications (CGA) 23 (3) (2003) 20–27.

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

[15] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-Up Robust Features (SURF), Computer Vision and Image Understanding (CVIU) 110 (3) (2008) 346–359.

[16] Q.-T. Luong, O. Faugeras, The Fundamental Matrix: Theory, Algorithms, and Stability Analysis, International Journal of Computer Vision (IJCV) 17 (1995) 43–75.

[17] C. Wu, S. Agarwal, B. Curless, S. Seitz, Multicore Bundle Adjustment, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 3057–3064.

[18] L. Kneip, D. Scaramuzza, R. Siegwart, A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

[19] S. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2006.

[20] K. Kolev, T. Brox, D. Cremers, 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.

[21] Y. Furukawa, B. Curless, S. M. Seitz, R. Szeliski, Towards Internet-scale Multi-view Stereo, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[22] H.-H. Vu, P. Labatut, J.-P. Pons, R. Keriven, High Accuracy and Visibility-Consistent Dense Multiview Stereo, Transactions on Pattern Analysis and Machine Intelligence (PAMI) 34 (5) (2012) 889–901.

[23] M. Goesele, N. Snavely, B. Curless, H. Hoppe, S. Seitz, Multi-View Stereo for Community Photo Collections, in: International Conference on Computer Vision (ICCV), 2007, pp. 1–8.

[24] B. Curless, M. Levoy, A Volumetric Method for Building Complex Models from Range Images, in: SIGGRAPH, 1996, pp. 303–312.

[25] M. Kazhdan, A. Klein, K. Dalal, H. Hoppe, Unconstrained Isosurface Extraction on Arbitrary Octrees, in: Eurographics Symposium on Geometry Processing (SGP), 2007.

[26] M. Callieri, P. Cignoni, M. Corsini, R. Scopigno, Masked Photo Blending: Mapping Dense Photographic Dataset on High-Resolution Sampled 3D Models, Computers & Graphics 32 (2008) 464–473.

[27] C. Allène, J.-P. Pons, R. Keriven, Seamless Image-Based Texture Atlases using Multi-band Blending, in: International Conference on Pattern Recognition (ICPR), 2008, pp. 1–4.

[28] V. Lempitsky, D. Ivanov, Seamless Mosaicing of Image-Based Texture Maps, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–6.

[29] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, D. Cohen-Or, Seamless Montage for Texturing Models, Computer Graphics Forum 29 (2010) 479–486.

[30] M. Waechter, N. Moehrle, M. Goesele, Let There Be Color! — Large-Scale Texturing of 3D Reconstructions, in: European Conference on Computer Vision (ECCV), 2014.

[31] P. Pérez, M. Gangnet, A. Blake, Poisson Image Editing, Transactions on Graphics 22 (3) (2003) 313–318.