Pacific Graphics 2009
S. Lee, D. Lischinski, and Y. Yu
(Guest Editors)

Volume 28 (2009), Number 7

An Intuitive Interface for Interactive High Quality Image-Based Modeling

Martin Habbecke and Leif Kobbelt

Computer Graphics Group, RWTH Aachen University, Germany

Abstract

We present the design of an interactive image-based modeling tool that enables a user to quickly generate detailed 3D models with texture from a set of calibrated input images. Our main contribution is an intuitive user interface that is entirely based on simple 2D painting operations and does not require any technical expertise by the user or difficult pre-processing of the input images. One central component of our tool is a GPU-based multi-view stereo reconstruction scheme, which is implemented by an incremental algorithm, that runs in the background during user interaction so that the user does not notice any significant response delay.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Methodology and Techniques—Interaction techniques, Image Processing and Computer Vision [I.4.8]: Scene Analysis—Surface fitting

1. Introduction

The reconstruction of realistic 3D models from photos and videos is a very common problem and over the last decades many different techniques have been proposed. However, most of the classical approaches utilize automatic algorithms that are controlled by a number of, more or less, intuitive parameters such as thresholds and weight coefficients. Due to numerous sources for errors such as image noise, lack of foreground segmentation, miscalibration, and ambiguous visibility, none of these algorithm can use one single default set of parameters for all inputs. Instead the user needs to adjust them accordingly. As a consequence, the application of such algorithms requires a certain level of technical expertise from the user and normally the parameters have to be adjusted in a trial and error process.

Recently, interactive reconstruction techniques have come into focus. Using these techniques, the user sketches rough hints through a graphical user interface, which are used by the system to adjust parameters and as boundary constraints for the reconstruction. In this paper we are proposing a new system which follows this concept. Our system takes a set of calibrated images as input and provides a simplistic and intuitive graphical user interface to the user. The output is a textured, high quality 3D model that on average is obtained after just a few minutes of interaction. The major challenges in the implementation of such a system are:

Intuitive control: We want to keep the interface as simple as possible, without requiring any technical knowledge from the user. Hence, we restrict the interaction to 2D as much as possible and provide only a minimum number of different modes (panning, zooming, painting). The only 3D interaction is the rotation of the (partially) reconstructed model.

Sketch-based interface: The system does not require precise user input such as picking feature points or lines. Moreover, it is not necessary to segment foreground from background. When the user paints over the region to be reconstructed he can stay away safely from the object silhouette and choose another image where this surface part is not near the silhouette for its reconstruction.

Fluent workflow: With existing algorithms, it can take several minutes or even hours of computation time to reconstruct an object of moderate complexity. This is what makes the parameter tuning for automatic reconstruction algorithms so tedious: the response times when changing a parameter are too long to give the impression of direct control. In our system we implemented an incremental reconstruction scheme that runs in parallel to the user activity. By doing so, computation times are effectively "hidden" within the interaction dialog. As a consequence, the user does not notice any significant delay.

The central motivation and justification for our interactive reconstruction technique is that putting the user into the loop enables a better streamlined workflow leading to shorter overall process times from raw input data to the final result. Even if user time is usually much more expensive than CPU time, we still have to consider the overall process time in applications where the result is needed quickly.

So-called automatic reconstruction algorithms require careful tuning of a set of parameters before the reconstruction runs automatically. If the result is not satisfactory, the parameters have to be adjusted and the reconstruction re-run. Hence, the overall process could also be considered "interactive" because the system computes in between two manual parameter changes. Our claim is that the 2D painting user interface that we are proposing is much more intuitive to handle than the parameters of existing multi-view stereo algorithms.

Practical experience shows that it is not easier to fix a broken, wrongly reconstructed model after a possibly fully automatic reconstruction process. The artifacts one encounters are not just small holes that can easily be filled, but rather surface parts that deviate from the true surface due to, for instance, local matching errors. In a post-process, with a standard surface editing tool, the input images are not available anymore. It hence is not possible to determine the correct position of the surface. In contrast, our system allows for the easy and immediate validation and correction of the surface with the help of the input images and as an integral part of the actual reconstruction process. Moreover, our user interface, with a simple 2D painting metaphor, requires less user skill than a 3D polygon mesh modeling tool.

2. Related Work

The idea to interactively generate 3D models from digital images has been explored in several earlier publications and software systems. The Facade system by Debevec et al. [DTM96] generates 3D models by manually building geometry proxies and linking related edges in several images. Similarly, the commercial software package PhotoModeler by Eos Systems and the PhotoMatch component of Google's SketchUp allow for the manual creation of 3D models based on images. None of these systems makes use of pre-computed structure or camera poses, they instead shift most of the work to the user.

Recently, interactive image-based 3D modeling systems have been proposed that exploit pre-computed structure-from-motion information. VideoTrace by van den Hengel et al. [vdHDT*07] and the architectural modeling system by Sinha et al. [SSS*08] allow for easy, interactive generation of 3D models by first sketching polygons in a user-selected image and then manually adjusting the positions of projected vertices and edges. Both systems use scene points or automatically detected vanishing points and lines to guide the user while editing. Their main limitation is the inability to reproduce fine surface structure and geometric detail since the reconstructed model consists of a coarse collection of planar polygons only. In contrast, our system is based on a photo-consistency driven mesh deformation approach and allows for the recovery of fine surface detail. Thormählen and Seidel [TS08] have taken a different approach by generating orthographic images from a calibrated input sequence. Users are then supposed to load these images into their modeling package of choice and do the actual modeling manually. While this works well for mechanical and especially symmetric objects, it is difficult to create models where the symmetry is less apparent or for entire scenes. The level of surface detail is completely up to the user's manual effort. The single view modeling approaches by Zhang et al. [ZDPSS02] and Prasad et al. [PZF06] also follow the idea of user-guided reconstruction. However, since these methods are limited to single input images, the user interactions are more complex than the simple 2D painting we employ.

Our approach is also related to earlier work in the field of multi-view stereo reconstruction and depth-map recovery from images based on explicit surface representations by triangle meshes. Zhang and Seitz [ZS01] as well as Isidoro and Sclaroff [IS03] deform a mesh by moving single vertices according to an energy functional; Esteban and Schmitt [ES04] add surface smoothness and silhouette constraints. More recently, Delaunoy et al. [DPG*08] have presented a mesh-based multi-view stereo formulation that rigorously integrates visibility information into the gradients of the error terms. None of the above methods has been designed for interactivity; they rather function as black boxes with prohibitively long computation times for an interactive system. In addition, a good initialization of the complete surface or even exact image silhouettes are required, which can be difficult to acquire in uncontrolled setups. Our method does not rely on any preprocessing or initial surface and is hence very flexible with respect to the input data.

Recent region growing reconstruction methods by Furukawa and Ponce [FP07], Goesele et al. [GSC*07], and Habbecke and Kobbelt [HK07] introduce the idea of extending a known part of the surface into unknown regions. The main benefit is that known surface parts serve well as initialization for the recovery of unknown surface regions. We integrate this idea into our interactive framework by enabling the user to actively extend the reconstructed surface through simple 2D painting interactions. While they yield results of high quality, surface growing approaches usually have two disadvantages. First, they generate seeds on a regular grid of image positions since there is no way to automatically determine which parts of a scene are supposed to be reconstructed. This results in a large number of seeds that have to be discarded and requires long computation times. In ad-

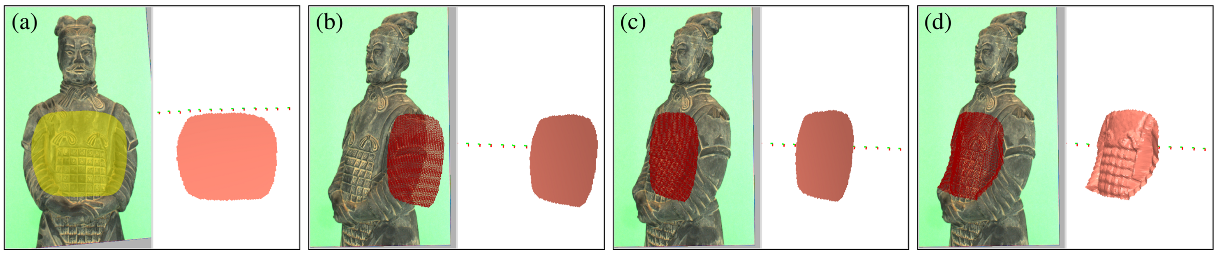


Figure 1: *Surface initialization: For a user-painted image region (a) the modeling system generates an initial surface patch (b). Optionally and only for the initial patch, the user can adjust the position of the patch by dragging its projection in the 2D viewer (c), after which the automatic matching procedure aligns the patch with the scene (d).*

dition, they fit surface elements individually rather than integrating a global regularization term. Our patch-based approach overcomes both weaknesses by only reconstructing what the user desires and by incorporating a geometrically meaningful surface smoothness term. It hence gains robustness – especially in the case of regions with little or no texture.

Zach [Zac08] has presented a depth map fusion algorithm that is related to our work since it generates reconstruction results of comparable quality and speed due to a GPU implementation. However, its main focus does not lie on the actual reconstruction from images, the results hence largely depend on the quality of the input depth maps. Furthermore, since it is based on a volumetric approach implemented with a flat memory layout on the GPU, the achievable resolution is rather limited.

Recent advances in interactive image editing and processing tools have shown that even complex problems can be made accessible by simple 2D user interfaces. In addition to the above mentioned modeling tools, particular examples are an interactive image completion system [PSK06], unwrap mosaics for video editing [RAKRF08] and an interactive image matting approach [WAC07], to name a few. While not related technically, our system follows the idea of approaching a difficult problem with a very simple interface.

3. Interactive Modeling System

The user interface of our modeling system consists of a 2D *image* viewer and a 3D *object* viewer (cf. Figure 1). Both viewers are synchronized such that panning or zooming a 2D image triggers the corresponding transformation in the 3D viewer. When rotating the object displayed in the 3D viewer, the 2D viewer switches to the input image that best matches the current viewing direction and performs a 2D rotation and scaling according to the orientation of the camera (see accompanying video). Hence the 3D viewer can be considered an image selection tool similar to the photo viewer of [SGSS08]. However, unlike in their system we do not apply perspective distortions to the input images to keep them as true 2D entities. This is done to limit the user interactions to 2D painting on a fronto-parallel plane and thereby keep the interface as simple as possible.

In addition to the image selection by panning, zooming, and rotation, the system provides a simple stroke-based interface with the modes *paint* and *erase* (un-paint) in the 2D image viewer for the actual surface reconstruction. For each user-painted region in a 2D image, the system generates a 3D surface patch by reconstructing a depth map. Since painting is merely activating pixels in an image, the user can easily switch back to a previous image and extend or trim the corresponding patch. Extending an existing image region yields a seamlessly enlarged surface patch. Painting in a new, unpainted image triggers the generation of a new, individual patch. The system overlays the input images with 2D projections of the already recovered surface which enables the user to easily spot uncovered regions. During an interactive modeling session, the user hence incrementally paints the object or scene to be reconstructed with simple brush strokes in 2D, thereby guiding the surface reconstruction algorithm.

The raw input images are calibrated by 2d3's *Boujou* and are loaded into the system without any further preprocessing such as foreground segmentation. The reconstruction algorithm runs in parallel to the user interaction. Whenever the user paints a stroke, the system starts reconstructing the depth values of the corresponding pixels right away. As a consequence, the maximum number of depth values that have to be computed simultaneously is limited by the maximum stroke size. For such small problems, our hierarchical reconstruction algorithm (cf. Section 5) converges in a fraction of a second, which is about the time the user needs to draw the next stroke. Hence, the user does not notice the delay caused by the computation, as he is busy with the next stroke. This gives the impression of a fluent workflow similar to traditional modeling or photo editing systems.

The precision requirements in the painting mode are not very strict since no special features have to be picked. Moreover, no precise painting along the object silhouette is required since the surface region that is close to the silhouette in one image can be reconstructed by painting on another image where this region is sufficiently far away from the silhouette.

Manual depth initialization: When a new patch is created, its initial depth values are estimated from the depth information of neighboring patches or by intersecting the viewing direction vectors of nearby images. In some cases, this initialization may fail and the user has to provide an additional hint (similar to VideoTrace [vdHDT*07]) by switching to a different, nearby image and dragging the projection of the 3D patch to a better initial position (cf. Figure 1b,c).

Watertight mesh generation: When the user is satisfied with the visual quality of the reconstructed collection of surface patches, they are turned into a solid triangle mesh using the method of Kazhdan et al. [KBH06]. Finally, our system automatically generates a texture atlas for the reconstructed mesh using the painted image regions (cf. Section 6). For a complete modeling session using our system, please see the accompanying video.

4. Patch Representation

Surface patches are represented as 2D triangle meshes with per-vertex depth values attached and embedded in a reference image. Since images are calibrated, each such 2D mesh induces a 3D surface. For an efficient implementation of the depth reconstruction algorithm (Section 5), we store a hierarchy of triangle meshes with different resolutions for each input image. For a given resolution we overlay a regular mesh of equilateral triangles over the entire image. When the user selects a certain region of the image, we activate all mesh vertices that lie within the painted region and all triangles that contain at least one active vertex. The reconstruction algorithm is then applied to the active parts of the mesh only.

In order to propagate the depth information from coarse to fine levels in the mesh hierarchy for each vertex in the fine level we store the barycentric coordinates with respect to the coarse-level triangle into which it falls. This provides the necessary data for a prolongation operator based on piecewise linear interpolation. By default, we use three hierarchy levels with edge lengths of 5, 10, and 15 pixels.

5. Depth Reconstruction

The reconstruction algorithm takes as input a reference image I_0, a set of comparison images $I_1 \dots I_m$, and a 2D triangle mesh M embedded in I_0. The goal is to recover a depth value d for each vertex in M such that the resulting 3D triangle mesh M' approximates the part of the scene visible in the region of I_0 covered by M. We extend the two-view matching method described in [SO07] to multiple views and add visibility terms as well as a regularization term. The latter term improves convergence and smooths the resulting surface to compensate for inevitable image noise.

For each image I_j, a 3×4 projection matrix $(\mathbf{P}_j | -\mathbf{P}_j \mathbf{c}_j)$ is given where $\mathbf{c}_j$ is the projection center. We pre-transform the object space such that $(\mathbf{P}_0 | -\mathbf{P}_0 \mathbf{c}_0) = (\mathbf{I}|\mathbf{0})$. By this transform, the relation between a point $\mathbf{x}$ in object space (i.e., a vertex of M'), its projection $\mathbf{p}$ into the reference image (i.e., the associated vertex in M) and the corresponding depth value d simplifies to $\mathbf{x} = d\mathbf{p}$ where $\mathbf{p} = (u, v, 1)$ is given in extended coordinate notation.

For an object space triangle S we measure photo-consistency by comparing the pixel colors in the projection of this triangle into reference image I_0 with the projections into comparison images I_j. Since the mesh M' is parametrized by a depth field over I_0 there is a one-to-one correspondence between triangles in M' and M. Hence, we can start with a triangle $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) \in M$, un-project it to $S = (d_1\mathbf{p}_1, d_2\mathbf{p}_2, d_3\mathbf{p}_3) \in M'$ and then map it to some comparison image I_j. The complete map from I_0 to I_j via S can be written as

$$\mathbf{H}_j(S) \;=\; \mathbf{P}_j - \mathbf{P}_j \mathbf{c}_j \mathbf{n}(S)^T$$

where $\mathbf{n}(S)$ is the normal vector of S, scaled such that the equation of the embedding plane becomes $\mathbf{n}(S)^T \mathbf{x} = 1$.

Let $\mathbf{x}_i = d_i(u_i, v_i, 1)$ be the corners of S, then the normal vector can be derived from the plane equation by

$$\mathbf{n}(S) \;=\; \begin{pmatrix} u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \\ u_3 & v_3 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1/d_1 \\ 1/d_2 \\ 1/d_3 \end{pmatrix}.$$

Our multi-view objective function sums over all comparison images I_j and all triangles $T \in M$ the pixel color differences between T and its re-projections, i.e.,

$$E_1 \;=\; \sum_{j=1}^{m} \sum_{T \in M} \sum_{\mathbf{p} \in T} \left(I_0(\mathbf{p}) - I_j(\mathbf{H}_j(T)\,\mathbf{p}) \right)^2.$$

Sugimoto and Okutomi [SO07] minimize E_1 by applying a Gauss-Newton optimization, i.e., by computing the Jacobian $\mathbf{J}$ of E_1 and by solving $\mathbf{J}^T\mathbf{J}\Delta = \mathbf{J}^T\mathbf{e}$ for parameter updates Δ. Here $\mathbf{e}$ denotes the vector of per-pixel intensity differences. In our work we employ a full Levenberg-Marquardt optimization by augmenting the linear system to $(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\Delta = \mathbf{J}^T\mathbf{e}$ [NW06]. This implies an algorithm that iteratively updates initial depth estimates by solving a sparse linear system for the per-vertex depth values.

We further extend the original approach by integrating a visibility term for each face of the mesh M. The binary weight $z_{j,T}$ is determined by OpenGL rendering the 3D mesh M' and all other previously reconstructed surface patches into I_j, and the continuous confidence weight $c_{j,T}$ is computed by the cosine of the angle between the face normal and the viewing direction:

$$E_2 \;=\; \sum_{j=1}^{m} \sum_{T \in M} z_{j,T}\, c_{j,T} \sum_{\mathbf{p} \in T} \left(I_0(\mathbf{p}) - I_j(\mathbf{H}_j(T)\,\mathbf{p}) \right)^2. \quad (1)$$

Finally, we add a surface regularization term E_{smooth} based on a discrete Laplace operator for triangle meshes

$$E_{\text{smooth}} = \sum_{\mathbf{x} \in M'} \mathbf{L}(\mathbf{x})^T \mathbf{L}(\mathbf{x}), \quad \mathbf{L}(\mathbf{x}) := \sum_{\mathbf{x}_i \in N(\mathbf{x})} (\mathbf{x}_i - \mathbf{x})$$

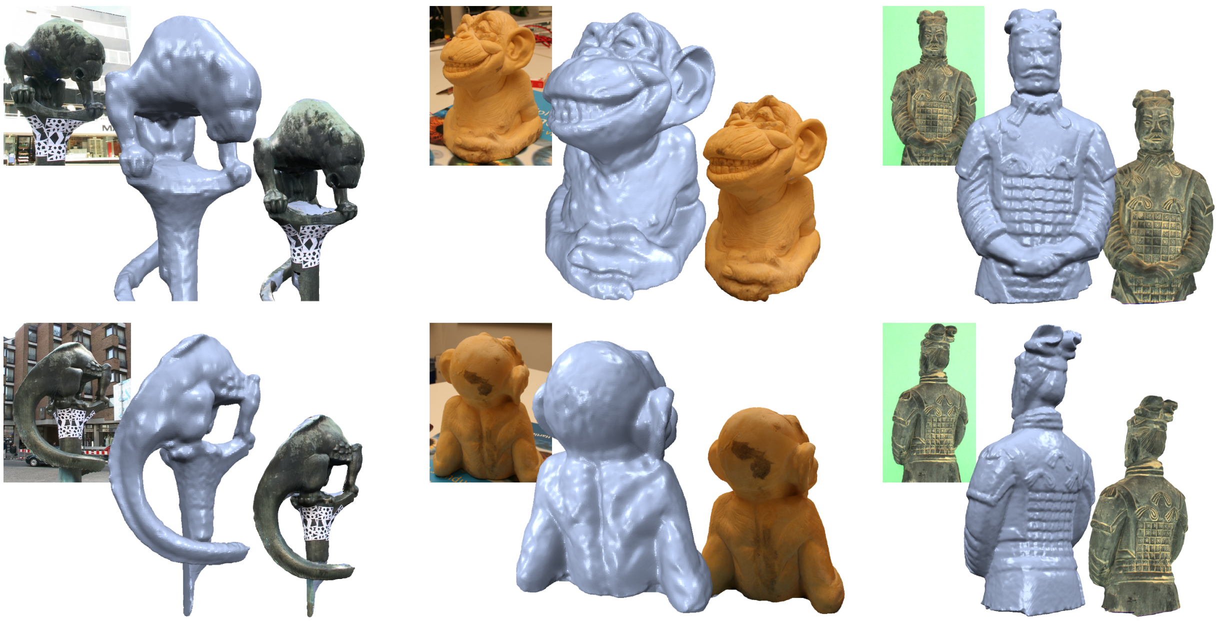


Figure 2: *Left: reconstruction of an outdoor statue with difficult topology and changing lighting conditions due to the shiny material illuminated by bright sunlight. The images were captured with a low quality, hand-held consumer video camera. Center: monkey sculpture modeled from a sequence of images taken with a high-res digital SLR. Right: Detailed reconstruction of a Chinese Warrior statue that demonstrates the high surface quality our method is able to achieve. Note that no foreground segmentation or any other image preprocessing has been applied during our experiments.*

where $N(\mathbf{x})$ denotes the set of 1-ring neighbors of $\mathbf{x}$ in M. We then end up with the complete objective function

$$E_3 \;=\; E_2 + \alpha E_{\text{smooth}}.$$

We choose the global weight α as

$$\alpha \sim m \cdot \left|\bigcup T\right| / (|V| \cdot e_{\text{avg}}^2),$$

where m is the number of comparison images, $|\bigcup T|$ is the total number of pixels covered by projected triangles T in I_0, and $|V|$ is the number of vertices in M. The unknown scale of the scene is compensated for by the average edge length e_{avg} of M' in object space. Since α also depends on the quality of the input images it is difficult to set it fully automatically. However, by choosing the weight according to the above heuristic, in our experiments it only had to be slightly adjusted by a constant factor that was kept fixed for each individual image sequence in our experiments.

In order to significantly accelerate the convergence of the iterative solver we run a hierarchical cascading scheme that first computes the best fit for a coarse mesh M_0, prolongates this solution to the next finer level M_1, and continues iterating. Since the sparse linear system solver takes most of the computation time, we only reduce the mesh resolution but not the image resolution. Our experiments have shown that it has a positive effect on the overall performance to reduce the number of comparison images m on coarse levels and only use the complete set of images on the finest level.

Depth values are initialized by propagating the depth information from neighboring, previously reconstructed patches. This information is obtained by rendering all front-facing patches into the reference image I_0 and reading out the z-buffer. Initial depth values are then propagated to neighboring vertices in M which are not covered by the rendering of a previously reconstructed surface patch. In case none of the vertices of a new patch overlaps with an existing part of the surface, our system falls back to a simple depth estimation heuristic that intersects viewing rays of the current and a nearby camera.

To compensate for non-Lambertian lighting conditions, we apply a simple intensity normalization by subtracting the average per-triangle intensities $\mu_{j,T}$. The inner term of E_2 in (1) is hence extended to

$$\sum_{\mathbf{p}\in T} \Big((I_0(\mathbf{p}) - \mu_{0,T}) - (I_j(\mathbf{H}_j(T)\,\mathbf{p}) - \mu_{j,T}) \Big)^2 .$$

Additional division by per-triangle intensity standard deviation has not shown to improve results in our experiments.

We have implemented the complete evaluation of the data term (1) and the computation of its partial derivatives with respect to the vertex depth values in CUDA. The main difficulty is the irregularity of the triangle mesh patches: Since patch boundaries can be arbitrary, it is not possible to find a regular layout for face *and* vertex data that enables coalesced memory accesses. We hence introduced a level of indirection

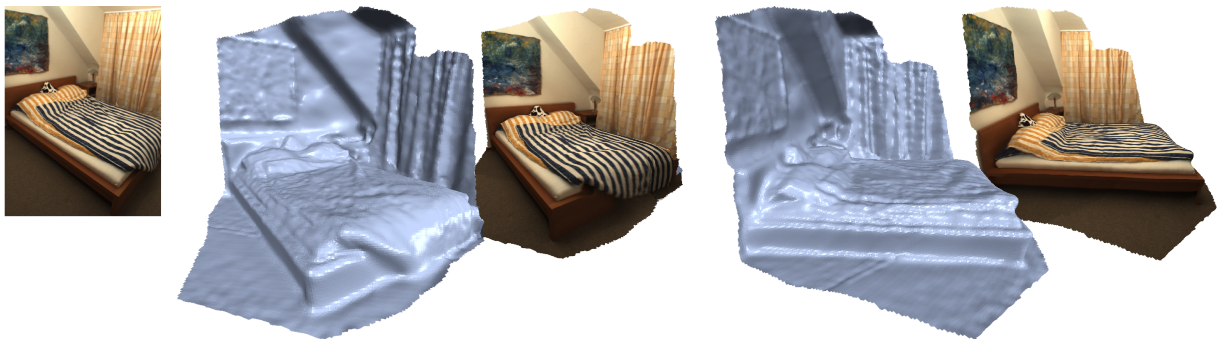


Figure 3: *Partial reconstruction of an indoor scene that demonstrates the versatility of our approach. In contrast to many previous MVS systems, our approach does not rely on initial surfaces derived from, e.g., object silhouettes. It hence easily copes with inside-out capturing scenarios as in this example.*

L	#ci	#f	#v	#s	CUDA	CPU	Solve
1	10	1000	548	18	1.13	57.34	1.86
2	2	226	135	67	0.70	9.20	0.20
3	2	91	60	148	1.32	8.26	0.07
1	10	8000	4130	18	6.17	464.91	30.00
2	2	1934	1032	67	1.59	79.26	4.36
3	2	832	459	148	1.91	76.35	4.36
1	10	16000	8224	18	12.34	936.84	70.02
2	2	3903	2063	67	3.04	161.40	10.03
3	2	1679	913	148	3.43	154.39	3.39

Table 1: *Computation time (in milliseconds) required by the major components of our system for three surface patches of different sizes. Please see text for details.*

by uploading a map from face indices to the three respective vertex indices of each face. All required face and vertex data can then simply be stored as linear arrays. Although this introduces incoherent memory accesses, we found our CUDA implementation to outperform a similar CPU implementation by a large margin (see Section 7 for detailed timings).

6. Texture Generation

By painting image regions in selected reference images, the user has explicitly specified which part of the surface is best seen in each of the input images. Similar to the approach of Sinha et al. [SSS*08], our system generates textures from the user-painted regions and hence ensures that no occluded or otherwise invalid image region is used for texturing. Since the Poisson fusion of the surface patches does not preserve the relation between surface regions and their respective reference image, we project the surface patches onto the final mesh in normal direction. Small regions on the mesh without reference information are closed by a few breadth-first propagation steps. We then find connected components of triangles with the same reference image, project them to the respective images and generate a texture atlas and appropriate texture coordinates.

Model	Resolution	Images	Time
Bahkauv	720×576	325	8 min
Monkey	2496×1664	107	10 min
Room	780×580	200	3 min
Warrior	1024×768	127	3 min
Dino	640×480	363	8 min
Temple	640×480	312	16 min

Table 2: *Details of the image sequences and complete modeling times for all reconstructed models.*

7. Results

All experiments have been performed on an Intel Core 2 Duo E6750 system. CUDA was run on a GeForce 8800 GTX graphics card. Table 1 summarizes the times required by the computation of partial derivatives of (1) as well as the solution of the sparse Levenberg-Marquardt system: “CUDA” denotes the partial derivative computation on the GPU, “CPU” an equivalent implementation on the host processor, and the “Solve” column contains the time required by the CHOLMOD sparse linear system solver [CDHR06]. L, #ci, #f, #v, and #s denote the resolution level, the number of comparison images, the number of faces and vertices of the surface patch, and the maximal number of per-face sample points in the reference image, respectively. The measured times show that the CUDA implementation is faster than the CPU implementation by a factor of up to 75, and that most of the time is spent solving the linear system.

Figure 4 shows reconstruction results for the Middlebury [SCD*06] *Dino* and *Temple* datasets. In both cases the full datasets with more than 300 images each were used. Our measurement results are: Temple: 90% within 0.6mm, 98.4% within 1.25mm. Dino: 90% within 0.52mm, 99.1% within 1.25mm. With these numbers our method takes an average rank among all methods that have participated in the full dataset benchmark (please see [Mid09] for comparison). The result for the Dino demonstrates that the MVS component of our system is capable of robustly reconstructing almost completely textureless surfaces due to the integra-

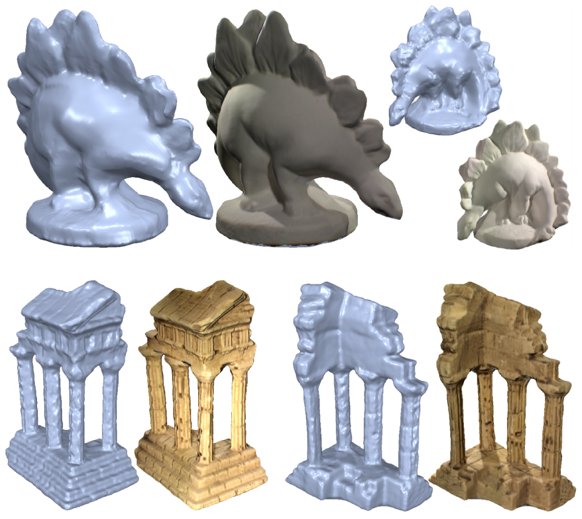
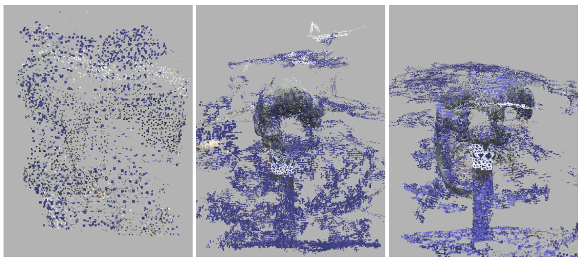


Figure 4: *Reconstruction results for the Middlebury [Mid09] Dino and Temple datasets, generated in about 8 minutes and 16 minutes, respectively.*

tion of a large number of input images and the geometrically meaningful regularization term. Regarding the computation times (cf. Table 2), our system outperforms almost all other methods, most of them by a large margin. At the time of writing only the depth map fusion method of Zach [Zac08] was faster than our interactive system (again, please see [Mid09] for details). Our results in the Middlebury benchmark underline that the proposed approach is able to solve one of the problems of current MVS systems: What matters most in many real applications is the time it takes to convert raw image data into a textured 3D model. Even if several hours of computing time (see most of the automatic methods in the Middlebury benchmark) might be less expensive than a few minutes of human interaction time, it does not help if a result is required as quickly as possible.

The *Bahkauv* statue, shown in Figure 2 (left) has been reconstructed from a sequence taken with a low quality hand held consumer camera. An noted above, the raw input images for this and the remaining experiments were calibrated by 2d3's *Boujou* but remained otherwise unmodified. In spite of the specular surface and the changes in lighting conditions, it is an easy task to recover a faithful reconstruction with our interactive system. The full modeling session is shown in the accompanying video. The images of the *Monkey* sculpture (cf. Figure 2 (center)) were taken with a digital SLR camera. Due to the higher resolution (2496$\times$ 1664), our method was able to recover much of the fine surface detail. The *Chinese Warrior* shown in Figure 2 (right) also demonstrates the high quality that our interactive modeling system is able to generate in just 3 minutes of total interactive reconstruction time for this example. Please note that no foreground segmentation has been applied to any of the input images during our experiments.



Run	Time	num	A	fullin	thresh	size	minImg
1	14 hrs	325	16	0	0.7	5	3
2	10 hrs	82	5	0	0.5	5	3
3	12 hrs	82	5	0	0.5	5	6

Figure 5: *Application of the fully automatic MVS system of Furukawa and Ponce [FP07] to the* Bahkauv *sequence. Parameters* num, A, fullin, threshold, size, *and* minImage *are named as they occur in the publicly available software. The resulting sets of reconstructed surface patches for three runs of the algorithm with adjusted parameters are shown from left to right.*

In the last example (cf. Figure 3) our system has been applied to an indoor scene. Inside-out capturing scenarios as in this case (in contrast to outside-in capturing for objects) pose a severe problem to many existing reconstruction systems that rely on, e.g., the visual hull for surface initialization. Our system, however, does not require a surface initialization or image pre-processing of any form and hence is flexible enough to cope with inside-out captured image sequences.

To substantiate our claim that putting the user into the loop significantly reduces the overall reconstruction time and improves the resulting quality, we have compared our approach to an existing, fully automatic MVS system. We chose the method of Furukawa and Ponce [FP07] since it has been made publicly available and since it is one of the best-rated methods in the Middlebury benchmark. We applied the system three times to the *Bahkauv* sequence with different parameter settings, starting with the default parameters. The required computation times, the actual parameters we used, and the resulting sets of reconstructed surface patches are shown in Figure 5. After more than 36 hours of total computation time we have not been able to find suitable parameters that enable the automatic reconstruction of the Bahkauv statue. Of course, given the right parameter settings and precise object silhouettes, the system of Furukawa and Ponce will certainly generate a higher quality solution. What we want to illustrate with this experiment is that the search for the right parameter settings in fully automatic reconstruction techniques can be quite time consuming. Furthermore, specifying object silhouettes for complex input images usually requires more precise manual work than the rough sketches in our system.

8. Discussion and Future Work

Our interactive image-based modeling system works best for a densely sampled sequence of input images. We do not, however, consider this a limitation: With today's capturing hardware and state-of-the-art structure-from-motion software it is an easy task to quickly generate calibrated sequences with hundreds of images. In contrast to other systems that are often limited in the number of input images due to both memory and computation time constraints, our system is able to handle an arbitrarily large number of input images. The number of input images does not influence the overall computation time.

Several previous methods ([vdHDT*07] and [SSS*08], for instance) rely on 3D points from the structure-from-motion process to assist the actual interactive reconstruction. We chose not to do so since these points may be very sparse for certain regions of the surface or may not be given at all, as is the case for the Middlebury data. While the mesh-based depth recovery has several advantages, like its robustness against image noise or slight miscalibration, its main limitation is the geometric resolution. Thin structures below the size of the triangle faces cannot be reconstructed. Furthermore, due to the simple photo-consistency measure, our current implementation is only able to handle scenes that do not deviate too much from the Lambertian reflectance model.

A limitation of the reconstruction workflow is the traditional 2-step process: Images are first captured and calibrated and only then passed on to the actual reconstruction. Only after the reconstruction is finished can missing images and unseen surface regions be detected, which often makes capturing of a completely new sequence inevitable. Though this is a problem common to all traditional multi-view stereo systems, it becomes even more apparent with our approach since the user gets an impression of the quality and coverage of the input images much more quickly. We hence believe that the integration of a rapid reconstruction approach like the one presented in this paper with the actual capturing process is a very important area of future work.

Acknowledgments

This project was funded by the DFG Cluster of Excellence on Ultra-high Speed Information and Communication (UMIC), German Research Foundation grant DFG EXC 89, http://www.umic.rwth-aachen.de/.

References

[CDHR06] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: *Algorithm 8xx: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate*. Technical Report TR-2006-005, University of Florida, 2006.

[DPG*08] DELAUNOY A., PRADOS E., GARGALLO P., PONS J.-P., STURM P.: Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *Proc. BMVC* (2008).

[DTM96] DEBEVEC P. E., TAYLOR C. J., MALIK J.: Modeling and rendering architecture from photographs. In *Proc. ACM SIGGRAPH* (1996).

[ES04] ESTEBAN C. H., SCHMITT F.: Silhouette and stereo fusion for 3d object modeling. *CVIU 96*, 3 (2004), 367–392.

[FP07] FURUKAWA Y., PONCE J.: Accurate, dense, and robust multi-view stereopsis. In *Proc. CVPR* (2007).

[GSC*07] GOESELE M., SNAVELY N., CURLESS B., HOPPE H., SEITZ S. M.: Multi-view stereo for community photo collections. In *Proc. ICCV* (2007).

[HK07] HABBECKE M., KOBBELT L.: A surface-growing approach to multi-view stereo reconstruction. In *CVPR* (2007).

[IS03] ISIDORO J., SCLAROFF S.: Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *Proc. ICCV* (2003), pp. 1335–1342.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proc. of SGP* (2006), pp. 61–70.

[Mid09] MIDDLEBURY: Middlebury multi-view stereo evaluation results. http://vision.middlebury.edu/mview, 2009.

[NW06] NOCEDAL J., WRIGHT S.: *Numerical Optimization*, 2nd ed. Springer, 2006.

[PSK06] PAVIC D., SCHÖNEFELD V., KOBBELT L.: Interactive image completion with perspective correction. *The Visual Computer 22*, 9-11 (2006), 671–681.

[PZF06] PRASAD M., ZISSERMAN A., FITTZGIBBON A.: Single view reconstruction of curved surfaces. In *Proc. of CVPR* (2006).

[RAKRF08] RAV-ACHA A., KOHLI P., ROTHER C., FITZGIBBON A. W.: Unwrap mosaics: a new representation for video editing. *ACM Trans. Graph. 27*, 3 (2008).

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. CVPR* (2006), pp. 519–528.

[SGSS08] SNAVELY N., GARG R., SEITZ S. M., SZELISKI R.: Finding paths through the world's photos. *ACM Trans. Graph. 27*, 3 (2008).

[SO07] SUGIMOTO S., OKUTOMI M.: A direct and efficient method for piecewise-planar surface reconstruction from stereo images. In *Proc. CVPR* (2007).

[SSS*08] SINHA S. N., STEEDLY D., SZELISKI R., AGRAWALA M., POLLEFEYS M.: Interactive 3d architectural modeling from unordered photo collections. *ACM Trans. Graph. 27*, 5 (2008).

[TS08] THORMÄHLEN T., SEIDEL H.-P.: 3d-modeling by ortho-image generation from image sequences. *ACM Trans. Graph. 27*, 3 (2008).

[vdHDT*07] V. D. HENGEL A., DICK A. R., THORMÄHLEN T., WARD B., TORR P. H. S.: Videotrace: rapid interactive scene modelling from video. *ACM Trans. Graph. 26*, 3 (2007).

[WAC07] WANG J., AGRAWALA M., COHEN M. F.: Soft scissors: an interactive tool for realtime high quality matting. *ACM Trans. Graph. 26*, 3 (2007).

[Zac08] ZACH C.: Fast and high quality fusion of depth maps. In *Proc. of 3DPVT* (2008).

[ZDPSS02] ZHANG L., DUGAS-PHOCION G., SAMSON J.-S., SEITZ S. M.: Single view modeling of free-form scenes. *Journal of Visualization and Computer Animation 13*, 4 (2002), 225–235.

[ZS01] ZHANG L., SEITZ S. M.: Image-based multiresolution shape recovery by surface deformation. In *Proc. SPIE* (2001), pp. 51–61.