Accurate Multi-View Reconstruction Using Robust Binocular Stereo and Surface Meshing

Derek Bradley
University of British Columbia
bradleyd@cs.ubc.ca

Tamy Boubekeur
TU Berlin
boubek@cs.tu-berlin.de

Wolfgang Heidrich
University of British Columbia
heidrich@cs.ubc.ca

Abstract

This paper presents a new algorithm for multi-view reconstruction that demonstrates both accuracy and efficiency. Our method is based on robust binocular stereo matching, followed by adaptive point-based filtering of the merged point clouds, and efficient, high-quality mesh generation. All aspects of our method are designed to be highly scalable with the number of views.

Our technique produces the most accurate results among current algorithms for a sparse number of viewpoints according to the Middlebury datasets. Additionally, we prove to be the most efficient method among non-GPU algorithms for the same datasets. Finally, our scaled-window matching technique also excels at reconstructing deformable objects with high-curvature surfaces, which we demonstrate with a number of examples.

1. Introduction

Multi-view stereo (MVS) algorithms have seen a surge of interest in recent years. Much progress has been made, both in terms of precision and in terms of performance, although methods with a combination of both high efficiency and high quality remain elusive.

In this paper, we revisit one of the most simple approaches to multi-view reconstruction, namely that of merging multiple depth maps estimated using binocular stereo. We show that, with careful design, this simple approach can yield some of the highest quality reconstructions of any multi-view stereo algorithm, while remaining highly scalable and offering excellent performance.

MVS algorithms that rely on depth maps can typically be divided into two separate processes. First, a depth map is computed for each viewpoint. Secondly, the depth maps are merged to create a 3D model, and a triangle mesh is generated. In this paper we introduce new methods for both of these stages. A state-of-the-art surface meshing algorithm allows us to reliably remove outliers and high frequency noise. This in turn enables the use of a simple and fast binocular stereo algorithm that produces very dense, but potentially unreliable points. In particular, our MVS algorithm has the following characteristics:

- The binocular stereo algorithm makes use of *scaled window matching* to improve the density and precision of depth estimates. Filtering and constraints are used to further improve the robustness.
- Our surface reconstruction algorithm uses adaptive, point-based filtering and outlier removal of the joint depth information from multiple views. This point-based formulation of the filtering process avoids resampling and quantization artifacts of other approaches.
- The meshing algorithm works in a lower dimensional space, resulting in high performance and well-shaped triangles.

All geometry processing algorithms work only on local neighborhoods, and have a complexity of $O(k \log k)$, where k is the number of points processed by the respective algorithm. Since the binocular stereo part is linear in the number of images, the complete algorithm remains highly scalable despite the high quality reconstructions it produces.

In the following, we will first review related work (Section 2) and provide a more detailed overview of our method (Section 3). We then discuss the individual stages of our method, including the binocular stereo matching (Section 4) and the surface reconstruction (Section 5). We conclude with results and a discussion in Sections 6 and 7.

2. Related Work

As mentioned, the MVS problem has received a lot of attention recently, yielding a variety of reconstruction algorithms. Following the taxonomy of Seitz et al. [33], MVS algorithms can be categorized into four classes: 3D volumetric approaches [21, 39, 40, 35, 34, 23], surface evolution techniques [9, 30, 14, 45], algorithms that compute and merge depth maps [37, 16, 36, 44, 25], and techniques that grow regions or surfaces starting from a set of extracted feature or seed points [17, 24, 12, 19]. Our algorithm falls into

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

the third category, and thus our discussion of earlier work focuses on other techniques using a similar approach. We refer the reader to [33] and the MVS evaluation website [26] for a more thorough discussion of the other techniques.

A multi-view framework for computing dense depth estimates was first proposed by Szeliski [37], who formulates the problem as a global optimization over the unknown depth maps. Szeliski also recovers motion estimates.

Strecha et al. [36] propose to jointly solve for depth and visibility using a generative model, where input images are assumed to be generated by either an inlier process or an outlier process. Depth and visibility are modeled as a Hidden Markov Random Field in conjunction with the Expectation-Maximization algorithm. Computation times are comparatively low for a sparse set of viewpoints, however they do not scale well. In addition, the focus of their work is to obtain only the depth map and outlier estimation for each view, and so they do not discuss merging the data to create a 3D scene.

A complimentary algorithm is presented by Zach et al [44], which takes as input a set of depth maps and volumetrically integrates them to create a 3D model using total variation regularization and an L^1 norm to measure data fidelity. Merrell et al. [25] also address the problem of merging depth maps to produce a 3D surface with a real-time GPU technique. They recursively merge depth maps from adjacent viewpoints by minimizing violations of visibility constraints. Two different approaches are presented, one that favors stability and one that is based on confidence. The fused depth maps are then converted to a consistent triangular surface with a multi-resolution quad-tree.

Our work is most similar to that of Goesele et al. [16], who showed that simple modifications to original window-based stereo algorithms can produce accurate results. In their algorithm, depth maps are computed by backprojecting the ray for each pixel into the volume and then reprojecting each discrete location along the ray onto neighboring views where window-based correlation is performed with sub-pixel accuracy. They choose only the points that correlate well in multiple views, and thus reconstruct only the portion of the scene that can be matched with high confidence. Finally, depth maps are merged with an off-the-shelf volumetric technique [8]. Although their method is simple to implement, their models suffer from a large number of holes and very long processing times. In contrast, our algorithm is very efficient and achieves very high accuracy combined with high density, when compared to other state-of-the-art MVS techniques.

3. Algorithm Overview

Our multi-view reconstruction algorithm takes as input a set of calibrated images, captured from different viewpoints around the object to be reconstructed. We assume that a segmentation of the object from the background is provided, so that the visual hull is represented as a set of silhouette images. As mentioned in the introduction, our MVS method is performed in two steps, binocular stereo on image pairs, followed by surface reconstruction. Figure 1 shows a diagram of the individual stages.

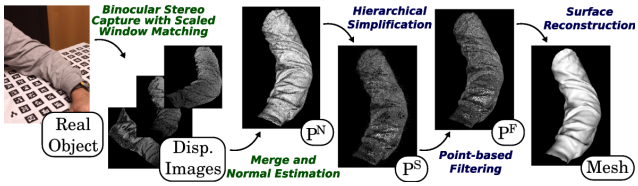

Figure 1. Acquisition pipeline: the binocular stereo algorithm generates a 3D point cloud that is subsequently processed and converted to a triangle mesh.

The binocular stereo part of our algorithm creates depth maps from pairs of adjacent viewpoints. We first rectify the image pairs, and then observe that the difference in projection between the views causes distortions of the comparison windows. We compensate for the most prominent distortions of this kind by employing a *scaled-window matching* technique, which improves the quality especially in high curvature regions and for sparse viewpoints (i.e. large baselines). The depth images from the binocular stereo pairs are converted to 3D points and merged into a single dense point cloud.

The second part of the algorithm aims at reconstructing a triangular mesh from the initial point cloud. It consists of three steps:

1. **Downsampling**: The point cloud is usually much denser than required for reproducing the amount of actual detail present in the data. Our first step is thus to downsample the data using *hierarchical vertex clustering* [5, 31, 32].
2. **Cleaning**: The simplified point cloud remains noisy. While some methods integrate the noise removal in the meshing algorithm [29, 22], we believe that this important data modification must be controlled explicitly, prior to any decision concerning the mesh connectivity.
3. **Meshing**: The final step is to generate a triangle mesh without introducing excessive smoothing. We build on *lower dimensional triangulation* methods [6, 18], which are fast and run locally, ensuring scalability and good memory-computational complexity.

In the following sections, we elaborate on the two main steps of our algorithm.

4. Stereo Matching

The first step of our MVS algorithm involves estimating depth maps for each camera view using binocular stereo

with one of the neighboring cameras as a reference view. For each image pair, the views are first rectified [13], such that corresponding scanlines in the primary and reference view correspond to epipolar lines. For each pixel in the rectified primary view we then find the closest matching pixel on the corresponding epipolar (scan)line in the rectified reference view. Specifically, we assume brightness constancy, and use *normalized cross correlation* (NCC) on square pixel regions as a metric for the best match:

$$NCC(v_0, v_1) = \frac{\sum_{j=1}^{N^2}(v_0(j) - \overline{v_0}) \cdot (v_1(j) - \overline{v_1})}{\sqrt{\sum_{j=1}^{N^2}(v_0(j) - \overline{v_0})^2 \cdot \sum_{j=1}^{N^2}(v_1(j) - \overline{v_1})^2}}, \tag{1}$$

where v_0 and v_1 are local neighborhoods of size $N \times N$ in the primary and the reference view, and $\overline{v_0}$ and $\overline{v_1}$ represent the intensity averages over the same neighborhoods. We ignore very bad matches by thresholding the NCC metric to the range of [0.5 - 1]. Because some uncertainty remains in the resulting matches, we further remove outliers with constraints and filtering, as discussed below.

Scaled window matching. As observed previously [10, 27, 28, 17], the difference in projection between the primary and reference view can distort the matching window, such that a square window in the primary view is non-square in the reference view. In our rectified setup, the vertical scale is identical in both views, since corresponding scanlines represent corresponding epipolar lines. The horizontal scale, however, depends on the 3D surface orientation.

A first-order differential model of this effect can be described as follows. Consider a differential surface patch dA in the tangent plane of an object point with normal n (see Figure 2). Let dw denote the (differential) length of the intersection of dA and the epipolar plane. This line segment dw projects to a segment $dw_1 = (dw \cdot \cos\phi_1)/(\cos\psi_1 \cdot z)$ on the image plane of the primary view (I_P), and to a segment $dw_2 = (dw \cdot \cos\phi_2)/(\cos\psi_2 \cdot z)$ on the image plane of the reference view (I_R). Here, z is the perspective depth, the ϕ_i correspond to the angles between the viewing rays and the projection of n into the epipolar plane, and the ψ_i correspond to the incident angles of the viewing rays on the respective image planes (see Figure 2). As a first-order effect, we therefore expect that any window in the primary view will appear scaled horizontally by a factor of

$$\frac{\cos\psi_1}{\cos\psi_2} \cdot \frac{\cos\phi_2}{\cos\phi_1}. \tag{2}$$

This horizontal scaling is particularly pronounced for wide camera baselines, i.e. "sparse" MVS setups with few cameras, as well as horizontally slanted surfaces, common when imaging high curvature geometry. To improve robustness in those settings, we employ a similar approach to Ogale and Aloimonos [27, 28], by performing window matching between the rectified primary view and a number

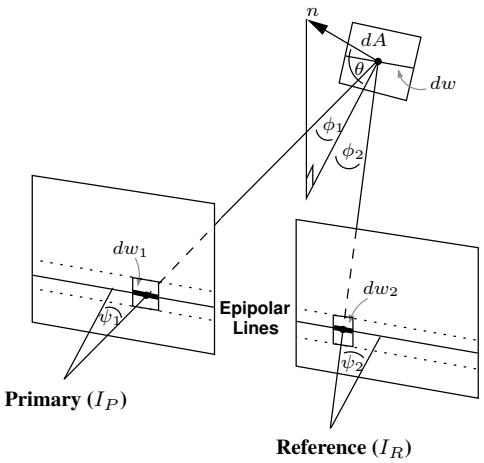

Figure 2. Corresponding pixel windows in the primary and reference view can have different width depending on the orientation of the 3D surface.

of versions of the rectified reference view with different horizontal scale factors. The best match is then defined as the largest NCC of any $N \times N$ neighborhood on the epipolar line in any of the differently scaled reference views.

We experimentally found that scale factors of $\frac{1}{\sqrt{2}}$, 1, and $\sqrt{2}$ yield excellent results for most datasets. Larger scale factors do not generate a significant number of reliable matches since view dependent effects start to dominate. For strongly asymmetric camera configurations, we first pre-scale the reference image with a global average of $\cos\psi_1/\cos\psi_2$, and then use window matching at the three scales mentioned above.

The above first order approximation is valid for small window sizes and surface orientations where the normal n is close to parallel to the epipolar plane. Larger angles of θ introduce a shear along the horizontal direction in addition to the horizontal scaling. We analyzed this issue, and determined that the shear is negligible for moderate camera baselines (up to 45° between viewing rays), and θ up to about 60°. We therefore decided against also using sheared versions of the reference view for the block matching.

Subpixel Optimization. The result of the window matching is a highly quantized disparity image for each viewpoint. We compute subpixel disparity values by performing a local search at the subpixel level. For each pixel p_P and its matched pixel p_R we perform correlation at a number of linearly interpolated positions between $p_R - 1$ and $p_R + 1$, that is, between the previous and next pixel in the scanline. With $2n$ additional correlation evaluations per pixel we achieve nominal subpixel precision of $1/n$ pixels. The calculations can be optimized by pre-computing an interpolated version of I_R which is $n \cdot width(I_R) \times height(I_R)$. In our experiments, we found that $n = 10$ gives us good precision; larger values have a diminishing return.

Constraints. In order to improve accuracy and reduce the number of outliers in the point cloud, we incorporate two constraints when performing window matching. Like many previous algorithms, we use the visual hull of the reconstruction object as a first constraint. We then use a disparity ordering constraint [4] for scenes where this assumption is valid, including the datasets in this paper.

Filtering. As a final step, outliers in the disparity images are removed with a median-rejection filter. If a disparity value is sufficiently different from the median of its local neighborhood, then it is rejected. To remove some high-frequency noise already in image space, disparity images are then smoothed with a trilateral filter, which is a standard bilateral filter [38] weighted by the NCC value that corresponds to the best match for each disparity pixel. The NCC value is a good indication of the confidence of each pixel, and we incorporate this into the smoothing function so that points of low confidence have less influence on their neighbors. Further denoising is later performed in 3D, as described in the next section.

At the end of the binocular stereo stage, all depth maps are backprojected into 3D, and merged into a single point cloud.

5. Surface Reconstruction

In this section, we describe our method for generating a triangle mesh from the unorganized point cloud obtained with binocular stereo. In addition to a position p_i, the algorithm requires a normal estimate n_i for each point. We generate a mesh M from the point-normal sampling

$$P^N = \{\{p_0, n_0\}, ...\{p_m, n_m\}\}$$

by **downsampling**, **cleaning** and **meshing** (see Figure 1).

Normal Estimate. One way to compute an approximate normal vector at each point is to use image-space gradient calculations on the individual depth maps. For better quality, taking into account several depth maps, we rather use a Principal Component Analysis (PCA) [20] in 3D space, once the individual depth maps have been merged into a point cloud. In this approach, the normal is given as u_i, the eigen vector associated with the smallest eigen value of the covariance matrix of the k-nearest-neighborhood of p_i. In practice, we choose $k \in [20, 50]$, and use a kD-Tree to efficiently compute the k-neighborhood queries.

In our case, the sign of the normal can be resolved using additional information available from the acquisition process: the vector c_i from the surface point to the camera always points from the surface to the outside of the object. Thus, we get

$$n_i = \frac{\overline{n_i}}{||\overline{n_i}||} \text{ with } \overline{n_i} = \begin{cases} u_i & \text{if } u_i \cdot c_i > 0 \\ -u_i & \text{otherwise} \end{cases}.$$

Note that n_i is only an estimate, with a smoothness controlled by k. In order to increase the quality of this estimate for later stages of the reconstruction pipeline, the normals are re-estimated after simplification and cleaning.

5.1. Downsampling

After merging of the depth maps, the resulting point cloud P^N contains large amounts of redundant information due to oversampling of smooth regions, as well as duplicate reconstructions of parts of the geometry from multiple views. In order to quickly and adaptively remove the redundant information, we apply a *hierarchical vertex clustering* to P^N. This recursive approach does not require connectivity and works as follows:

1. compute a bounding box B around P^N;
2. instantiate a hierarchical space partitioning structure H in B;
3. partition P^N recursively in H until each subset of P^N respects a given error metric;
4. replace each subset by a single representative sample.

Here, we choose H as a Volume-Surface Tree [5] for its hybrid octree-quadtree structure, which is almost as fast as a simple octree while performing fewer splits for a given error bound. We combine it with a linear L_2 error metric computed over the subsets of P^N at each level (bounded to 10^{-3} of the diagonal of B in our experiments). We choose this metric in order to trade quality for efficiency, however it can be seamlessly replaced by alternative ones, such as the $L^{2,1}$ [7] or the Quadratic [15] error metrics. The representative samples are computed as simple averages of positions and normals in the cluster. The union P^S of these representative samples forms the set of points we process in the following stages of the pipeline.

5.2. Cleaning

Even after simplification, P^S remains a noisy model, and thus needs to be filtered. In practice, we can identify two kinds of noise:

- **outliers**, which are samples located far away from the surface, usually grouped in small isolated clusters.
- **small scale high frequency noise**, generated as in any physical acquisition process.

We address the former using an iterative classification. We compute the *Plane Fit Criterion* proposed by Weirich et al. [42], remove the detected outliers, and restart with a quadratically decreasing bound until a user-defined threshold. Our experiments show that the number of iterations can be fixed to 3.

The high frequency noise is removed using a point-normal filtering inspired by Amenta and Kil [3] and is basi-

cally a simplification of the Moving Least Square projection [1] using the per-sample normal estimate. This filtering process is based on an iterative projection procedure, which can be made local: considering a point $q = \{p_q, n_q\} \in P^S$, we compute the projection of q onto the plane $H_q = \{c(q), n(q)\}$, where

$$c(q) = \frac{\sum_i \omega(||p_q - p_i||)p_i}{\sum_i \omega(||p_q - p_i||)},$$

and

$$n(q) = \frac{\sum_i \omega(||n_q - n_i||)n_i}{||\sum_i \omega(||n_q - n_i||)n_i||}$$

are computed over a local neighborhood of points near q. For efficiency reasons, we use Wendland's [41] compactly supported, piecewise polynomial kernel function

$$\omega(t) = \begin{cases} (1 - \frac{t}{h})^4(\frac{4t}{h} + 1) & \text{if } 0 \leq t \leq h \\ 0 & \text{if } t > h \end{cases},$$

where h controls the size of the support (i.e. smoothness). This procedure converges very quickly in the vicinity of P^S, a conservative condition that always holds in our case, since the only points we filter are actually the samples of P^S. Therefore, we fix the number of iterations to 3. Note that $\omega(t)$ has compact support, so that the filtering process is local and allows us to process the data with only knowledge of a bounded and small neighborhood.

5.3. Meshing

The last part of our surface reconstruction pipeline generates the connectivity between filtered samples P^F in order to obtain a reliable triangle mesh. As we have already filtered the point samples, we seek an interpolation of P^F, rather than an approximation. Thus we discard dynamic and implicit surface reconstruction approaches. We propose to use a fast interpolating meshing approach based on the Delaunay triangulation. This combinatorial structure offers a canonical way to define a connectivity over a set of points. Basically, it provides a n-D simplicial connectivity for a set of samples in an n-D space, which means that, applied on the 3D sampling P^F, we obtain a tetrahedral mesh, from which we have then to extract a 2-manifold made of triangles. Unfortunately, this last procedure is very slow [2] and prone to artefacts when the surface sampling ratio is unknown, which is the case here. In fact, the Delaunay triangulation can, in practice, only be performed efficiently with small and low dimensional point sets. Therefore, we build on [18] and [6] and design an algorithm that solves the problem using the following principles:

- **locality**: in order to replace a global triangulation by a collection of local ones, we split P^F into clusters using a density-driven octree (we bound the density to 100 points in our experiments). A Delaunay triangulation is then performed independently in each cluster.
- **dimension reduction**: clusters are split more finely if they violate a predicate indicating that the local set of points can be projected onto a plane without folding, i.e. if the cluster is not a height field. This approach allows us to project the samples in a leaf cluster onto a least-squares plane and to perform the Delaunay triangulation in the 2D space defined by this plane.

When processing the local clusters, this lower-dimensional Delaunay triangulation leads to a collection of disjoint mesh patches. To establish a single connected mesh, one efficient method is to generate overlap between neighboring clusters prior to triangulation. As such, we volumetrically *inflate* each cluster by adding points from neighboring octree cells, then perform the 2D triangulation in the cluster, and finally *remove* overlapping triangles afterwards using the following aggressive classification of the triangles:

- **outside**: a triangle that lies completely outside the original (i.e. pre-inflation) cluster it was generated for;
- **redundant**: more than one instance of the triangle is present in the overlapping zone (i.e. perfect overlapping, which happens frequently with Delaunay triangulations);
- **dual pair**: the triangle forms, with a triangle sharing a common edge, the dual configuration of two triangles present in a neighboring partition;
- **valid**: in all other cases.

Outside, *redundant* and *dual pair* triangles are removed. This process creates well-shaped triangles and rarely fails at avoiding non-manifold configuration thanks to the local uniqueness of the Delaunay triangulation. Artefacts appear when the sampling-to-curvature ratio exceeds a certain threshold, but fortunately our binocular stereo process produces dense enough sample sets to avoid such problems. Our performance-driven approach processes larger clusters, minimizing the number of lower-dimensional projections. Additionally it allows the algorithm to straightforwardly run in parallel.

Rapid Height-Field Detection. In order to estimate if a given (inflated) cluster can be orthogonally projected in the lower dimension (i.e. single-valued bivariate function), we use Boubekeur's aggressive predicate [6, 5]. In cluster i with k_i samples, we fit a plane $H_i = (c_i, n_i)$. Then, the cluster can be projected if:

$$\forall j \in [0, k_i] \begin{cases} n_{ij} \cdot n_i & > \delta_n \text{with } \delta_n \in [0, 1], \text{ and} \\ \frac{|(p_{ij} - c_i) \cdot n_i|}{\max_{k_i}(||p_{ik_i} - c_i||)} & < \delta_d \text{with } \delta_d \in [0, 1] \end{cases}$$

The value δ_n bounds the cone of normals of the cluster while the δ_d bounds the displacement from the tangent plane leading to more planar clusters, more suitable for triangulation. We use $\delta_n = \delta_d = 1/6$ in our experiments.

5.4. Performance

All three steps in the geometry reconstruction stage are designed for high performance, as well as high quality. For each input point, each algorithm only processes a small local neighborhood of fixed size. Such neighborhoods can be found in $\log N$ time using kD-Trees. Thus, the asymptotic complexity of each algorithm is $O(N \log N)$, where N is the number of input points of the respective algorithm.

In absolute terms, the algorithms are very fast, each taking no more than a few seconds on a single CPU for the examples in the paper. For example, the mesh generation algorithm processes $100k$ samples in less than 5 seconds. These times are negligible compared to the binocular stereo stage, which takes several minutes on typical datasets.

6. Results

We demonstrate our multi-view stereo algorithm with a number of reconstructions. First, we quantitatively evaluate our results using the *Middlebury* datasets [26, 33]. The data consists of two objects, a dinosaur and a temple (see Figures 3 and 4), and three different sets of input images for each one, with viewpoints forming a sparse ring, a full ring, and a full hemisphere around the object.

Our method is well-suited for viewpoints arranged in a ring setup since the grouping of cameras into binocular pairs is straightforward. Thus we perform our evaluation on the *dinoRing* (48 images), *dinoSparseRing* (16 images), *templeRing* (47 images), and the *templeSparseRing* (16 images) inputs. The quantitative results of our algorithm are shown in Table 1. For each of the datasets our method is compared to a number of the state-of-the-art techniques. Algorithms are evaluated on the accuracy (Acc) and completeness (Cmp) of the final result with respect to a ground truth model, as well as processing time (Time). We highlight the best performing algorithm for each metric. Our technique proves to be the most accurate for the sparse-viewpoint datasets. Additionally, ours is the most efficient among non-GPU methods for all four datasets. For reference, we include a short list of methods that make use of the GPU. These results are of much lower quality. We show our reconstruction of the dinoSparse dataset in Figure 3, including an image of the points generated by the stereo matching step. Note that it would be possible to use our method in camera setups other than rings. For instance, a full spherical setup could be handled by computing a Delaunay triangulation of the camera positions, and then computing a binocular image for each edge in that triangulation. This approach would still scale linearly in the number of cameras, since the number of edges in a Delaunay triangulation is linear in the number of vertices.

We highlight the effectiveness of the scaled-window matching technique (described in Section 4) by re-coloring

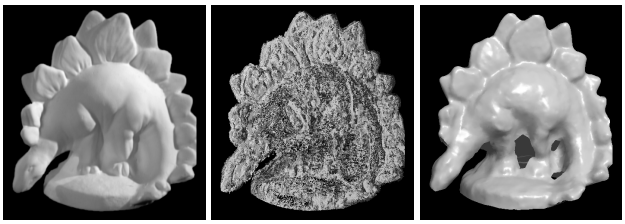

Figure 3. Dino reconstruction from sparse (16) viewpoints. Left: one input image (640x480 resolution). Middle: 944,852 points generated by the binocular stereo stage. Right: reconstructed model (186,254 triangles).

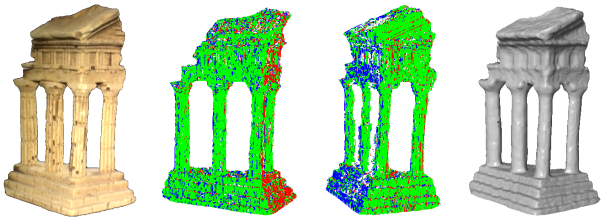

Figure 4. Scaled window matching on the templeRing dataset. Left: one of 47 input images (640x480 resolution). Middle: two disparity images re-colored to show which horizontal window scale was used for matching. $Red = \sqrt{2}$, $Green = 1$, and $Blue = \frac{1}{\sqrt{2}}$. Right: reconstructed model (869,803 triangles).

two of the recovered disparity images from the templeRing model, showing which of the three horizontal window scales were used to recover each depth estimate. In this visualization (shown in Figure 4) green pixels represent the un-scaled window, red pixels represent matches using $\sqrt{2}$ scaling, and blue pixels correspond to $\frac{1}{\sqrt{2}}$ scaling.

In addition to being accurate and efficient, our algorithm works particularly well for objects with deformable surfaces, which we demonstrate by reconstructing a shirt sleeve (Figure 5), a crumpled blanket (Figure 6), and a bean-bag chair (Figure 7). These datasets were captured using a Canon Digital SLR camera calibrated using the technique of Fiala and Shu [11]. For the shirt sleeve we include a zoom region to show the high-resolution mesh reconstruction that captures the individual wrinkles. Previously, a result of this visual quality has only been achieved by placing colored markers directly on the shirt [43]. For the blanket model, we include a window-scale image similar to Figure 4. Here we can see the different horizontal window scales that are used in a single binocular pair as a result of the high-curvature surface of the blanket. Finally, we reconstruct a bean-bag chair showing deformation before and after placing a heavy weight on it. With a number of synchronized video cameras, we could reconstruct this deformation over time by applying our MVS technique at each time-step. Since typical video sequences of deforming objects can have hundreds or even thousands of frames, the efficiency of our method makes it very suitable for such applications. A list of parameters used in all experiments is shown in Table 2.

	dinoSparseRing			dinoRing			templeSparseRing			templeRing		
Method	Acc	Cmp	Time	Acc	Cmp	Time	Acc	Cmp	Time	Acc	Cmp	Time
Our technique	0.38	94.7	7	0.39	97.6	23	0.48	93.7	4	0.57	98.1	11
Furukawa [12]	0.42	99.2	224	0.33	99.6	544	0.62	99.2	213	0.55	99.1	363
Zaharescu [45]	0.45	99.2	20	0.42	98.6	42	0.78	95.8	25	0.55	99.2	59
Gargallo [14]	0.76	90.7	18	0.60	92.9	35	1.05	81.9	35	0.88	84.3	35
Goesele [16]	0.56	26.0	843	0.46	57.8	2516	0.87	56.6	687	0.61	86.2	2040
Hernandez [9]	0.60	98.5	106	0.45	97.9	126	0.75	95.3	130	0.52	99.5	120
Strecha [36]	1.41	91.5	15	1.21	92.4	146	1.05	94.1	6	0.86	97.6	57
GPU Methods												
Sormann [35]				0.81	95.2	5				0.69	97.2	5
Merrell_stab [25]				0.73	73.1	0.5				0.76	85.2	0.4
Merrell_conf [25]				0.84	83.1	0.3				0.83	88.0	0.3
Zach [44]				0.67	98.0	7				0.58	99.0	7

Table 1. Results for the Middlebury Datasets. The top performing algorithm in each category is highlighted. *Accuracy* is measured in millimeters, *Completeness* as a percentage of the ground truth model, and *Normalized Running Time* is in minutes.

Parameter	**Value**
Stereo window matching size	5×5
NCC lower threshold	0.5
Sub-pixel matching precision	$1/10$ pixels
Median-rejection filter (size, threshold)	$15 \times 15, 2 \times median$
Tri-lateral filter (size, $\sigma_{domain}, \sigma_{range}$)	$15 \times 15, 15/4, 10$
k-neighborhood size	50
L_2 error bound for downsampling	10^{-3}
Outlier removal iterations	3
Filtering support size h	5.10^{-3}
Filtering iterations	3
Cell density for meshing	100
δ_n and δ_d for height-field predicate	$1/6$

Table 2. Summary of parameters used in the experiments (top: stereo matching, bottom: surface reconstruction). Surface reconstruction values are rescaled in the unit cube.

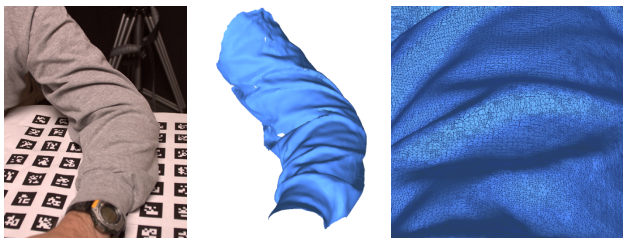

Figure 5. Sleeve reconstruction. Left: cropped region of 1 of 14 input images (2048x1364). Middle: final reconstruction (290,171 triangles). Right: zoom region to show fine detail.

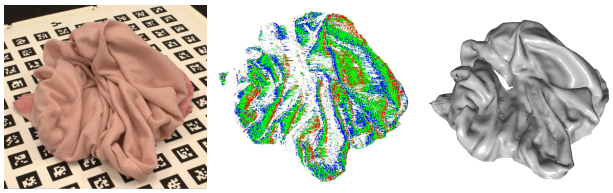

Figure 6. Crumpled blanket. Left: 1 of 16 input images (1954x1301). Middle: re-colored disparity image to show window scales. Right: final reconstruction (600,468 triangles).

7. Discussion

In this paper, we have introduced a novel multiview stereo algorithm based on merging binocular depth maps with a state-of-the-art meshing algorithm. By employing an efficient, yet high-quality point-based processing pipeline, we are able to effectively remove outliers and suppress high-frequency noise. This robustness allows us to use relatively simple but efficient methods for the binocular stereo part, without paying too much attention to the reliability of the stereo matches. In addition we increase the number of stereo matches by employing a scaled window matching approach that, in turn, provides the meshing stage with additional information. It is this combination of a simple stereo algorithm producing lots of matches with sophisticated point-based post-processing that allows us to create high-quality reconstructions very efficiently.

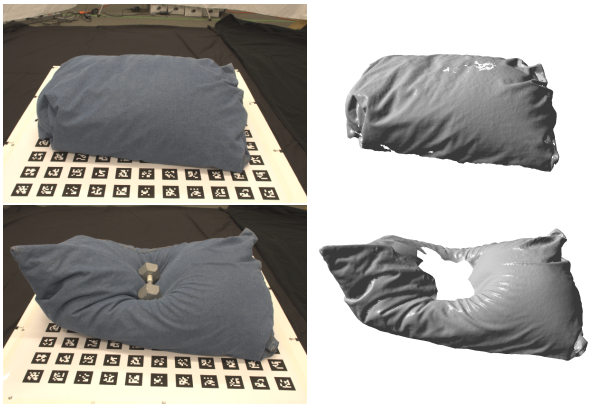

Figure 7. Bean-bag chair reconstruction, before (top) and after (bottom) deformation. Left: 1 of 16 input images (1954x1301). Right: final reconstruction (4,045,820 and 3,370,641 triangles).

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *IEEE Visualization*, 2001.

[2] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry Processing*, 2007.

[3] N. Amenta and Y. J.Kil. Defining point-set surfaces. In *ACM SIGGRAPH*, 2004.

[4] H. Baker and T. Binford. Depth from edge and intensity based stereo. pages 631–636, 1981.

[5] T. Boubekeur, W. Heidrich, X. Granier, and C. Schlick. Volume-surface trees. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2006)*, 25(3):399–406, 2006.

[6] T. Boubekeur, P. Reuter, and C. Schlick. Visualization of point-based surfaces with locally reconstructed subdivision surfaces. In *Shape Modeling International*, June 2005.

[7] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *ACM SIGGRAPH*, 2004.

[8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[9] C. H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *Comput. Vis. Image Underst.*, 96(3):367–392, 2004.

[10] O. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level set methods and the stereo problem. *IEEE Trans. on Image Processing*, 7(3):336–344, 1998.

[11] M. Fiala and C. Shu. Fully automatic camera calibration using self-identifying calibration targets. Technical Report NRC-48306 ERB-1130, National Research Council of Canada, 2005.

[12] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. In *CVPR*, 2007.

[13] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Mach. Vision Appl.*, 12(1):16–22, 2000.

[14] P. Gargallo, E. Prados, and P. Sturm. Minimizing the reprojection error in surface reconstruction from images. In *ICCV*, 2007.

[15] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH*, 1997.

[16] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *CVPR*, 2006.

[17] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007.

[18] M. Gopi, S. Krishnan, and C. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Eurographics*, 2000.

[19] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *CVPR*, 2007.

[20] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *ACM SIGGRAPH*, 1992.

[21] A. Hornung and L. Kobbelt. Hierarchical volumetric multiview stereo reconstruction of manifold surfaces based on dual graph embedding. In *CVPR*, 2006.

[22] M. Kazhdan, M. Bolitho, , and H. Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, 2006.

[23] K. Kolev, M. Klodt, T. Brox, and D. Cremers. Propagated photoconsistency and convexity in variational multiview 3d reconstruction. In *Workshop on Photometric Analysis for Computer Vision*, 2007.

[24] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *ICCV*, 2007.

[25] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, 2007.

[26] mview. http://vision.middlebury.edu/mview/.

[27] A. S. Ogale and Y. Aloimonos. Shape and the stereo correspondence problem. *Int. J. Comp. Vis.*, 65(3):147–162, 2005.

[28] A. S. Ogale and Y. Aloimonos. A roadmap to the integration of early visual modules. *Int. J. Comp. Vis.*, 72(1):9–25, 2007.

[29] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H. Seidel. Multi-level partition of unity implicits. In *SIGGRAPH*, 2003.

[30] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *CVPR*, 2005.

[31] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scenes. *Modeling in Computer Graphics*, 1993.

[32] S. Schaefer and J. Warren. Adaptive vertex clustering using octrees. In *Geometric Design and Computing*, 2003.

[33] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006.

[34] S. N. Sinha, P. Mordohai, and M. Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *ICCV*, 2007.

[35] M. Sormann, C. Zach, J. Bauer, K. Karner, and H. Bischof. Watertight multi-view reconstruction based on volumetric graph-cuts. In *SCIA*, 2007.

[36] C. Strecha, R. Fransens, and L. V. Gool. Combined depth and outlier estimation in multi-view stereo. In *CVPR*, 2006.

[37] R. Szeliski. A multi-view approach to motion and stereo. In *CVPR*, 1999.

[38] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[39] S. Tran and L. Davis. 3d surface reconstruction using graph cuts with surface constraints. In *ECCV*, 2006.

[40] G. Vogiatzis, C. H. Esteban, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. In *PAMI*, 2007.

[41] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(4):389–396, 1995.

[42] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. *Eurographics Symposium on Point-Based Graphics*, 2004.

[43] R. White, K. Crane, and D. Forsyth. Capturing and animating occluded cloth. In *SIGGRAPH*, 2007.

[44] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l^1 range image integration. In *ICCV*, 2007.

[45] A. Zaharescu, E. Boyer, and R. Horaud. Transformesh: A topology-adaptive mesh-based approach to surface evolution. In *ACCV*, 2007.