Point-Based 3D Reconstruction of Thin Objects

Benjamin Ummenhofer and Thomas Brox
Computer Vision Group
University of Freiburg, Germany
{ummenhof, brox}@cs.uni-freiburg.de

Abstract

3D reconstruction deals with the problem of finding the shape of an object from a set of images. Thin objects that have virtually no volume pose a special challenge for reconstruction with respect to shape representation and fusion of depth information. In this paper we present a dense point-based reconstruction method that can deal with this special class of objects. We seek to jointly optimize a set of depth maps by treating each pixel as a point in space. Points are pulled towards a common surface by pairwise forces in an iterative scheme. The method also handles the problem of opposed surfaces by means of penalty forces. Efficient optimization is achieved by grouping points to superpixels and a spatial hashing approach for fast neighborhood queries. We show that the approach is on a par with state-of-the-art methods for standard multi view stereo settings and gives superior results for thin objects.

1. Introduction

Image-based 3D reconstruction is the problem of inferring the surface of real world objects solely from visual clues. In the past years many algorithms have been developed to address this problem. To the best of our knowledge, none of them addresses the reconstruction of very thin objects, such as the street sign in Fig. 1. The sign has almost no volume compared to the size of its surface.

Such thin objects are very problematic for contemporary reconstruction methods. Most methods represent the scene as an implicit function defined at regular grid points [3, 14, 13, 26, 12, 24]. Grids provide an effective structure to integrate information from multiple views and allow for the approximation of derivatives by simple finite differences. However, grids cannot properly represent objects thinner than the voxel size, and the fixed grid comes with high memory requirements, which severely limits the resolution. In the case of an arbitrary thin object, the resolution required to represent the object leads to extreme memory requirements. Using adaptive grids, such as octrees [20], mitigates this problem but cannot solve it.

Figure 1. Dense point cloud reconstruction of a street sign. **Top:** Two renderings of the reconstruction when ignoring opposed surfaces (left and center) and a photo of the scene (right). Many points are on the wrong side of the object. **Bottom:** Reconstruction result of our approach explicitly modeling opposed surfaces. All images are rendered with front- and back-faces visible. Our approach resolves collisions between points that represent different sides of thin objects. Almost all points from the correct side pass the depth test (left and center) and therefore lie on the correct side. The approach preserves the thin structure of the objects, as seen in the view from the top (right).

Another popular surface representation is by triangle meshes [8, 9, 5]. In contrast to voxel grids, they only model the surface rather than the whole scene volume, and different parts of the scene can be represented by triangles of different size. This makes mesh based algorithms suitable for large scale reconstructions [9], and potentially also allows to handle thin objects. However, mesh representations typically have problems with change in topology during surface evolution. For this reason, Vu *et al*. [9] create a Delaunay tetrahedral mesh where tetrahedra are labeled as inside

or outside. The initial surface triangles are the tetrahedron faces that connect two tetrahedra with opposite labels. In case of a thin sheet, none of the tetrahedra would be labeled as inside the object and the triangulated surface would miss the object.

We argue that the best representation for thin objects is a point cloud representation with reference to a set of registered depth maps. Point clouds are the output of many 3D sensors, such as laser scanners and a very traditional way to represent 3D scenes [23, 4, 15]. They are very common in robotics, yet have been less popular in computer vision in recent years as they do not make the topology of the scene explicit. However, regularity can be integrated by means of pairwise forces. Similar to Szeliski and Tonnesen [23] and Fua [4] we use forces to manipulate the orientation and position of points. Usually the design of the forces is physically motivated and uses for instance the Lennard-Jones potential to avoid a clustering of the points. We can avoid the latter, because we keep a reference of the points to the depth maps from which they originated and allow only for motion of points along their projection rays. Related to this, Merrell *et al*. [17] use multiple depth maps to model the scene. Finally, they generate a subset of high quality depth maps by fusing the information from multiple neighboring depth maps. A limitation of depth maps is the affinity to a specific camera. Objects that are occluded for the specific point of view cannot be represented. In contrast, our approach treats the values of all depth maps as a point cloud and jointly optimizes all points, improving all depth maps at the same time.

The PMVS approach of Furukawa and Ponce [6] uses a patch representation similar to a point cloud and potentially can deal with thin objects. The patches originate from feature matching and are added and removed in an iterative way. Like in our approach, the depth and the normal of the patches is optimized. However, the result obtained with PMVS is not dense everywhere when rendered from one of the camera viewpoints and lacks regularization, which shows in the experiments.

A common challenge of point cloud representations is computational efficiency because, in contrast to voxel grids or meshes, the neighborhood structure is not explicit and may change. We use efficient data structures and a coarse-to-fine optimization based on superpixels to handle large point clouds with millions of points.

Moreover, we explicitly deal with a problem that is specific to thin objects: if the object is regarded from opposite viewpoints, points from different surfaces basically share the same position but have opposite normals. Noise in the measurements will lead to contradictive results, where invisible points occlude visible ones. We call this the problem of *opposed surfaces* and introduce a coupling term in our energy model that deals with this problem.

We provide a complete reconstruction system that takes as input a video sequence with known intrinsic camera parameters. An initial point cloud is computed via incremental bundle adjustment and a variant of semi-global matching [10]. The heart of the approach is an energy model that regularizes the point cloud and pulls the points to common surfaces with normals that are consistent with the viewing direction. We also iteratively refine the camera parameters via dense alignment of the image and the corresponding rendered view of the reconstruction.

2. Initial depth maps and camera parameters

The initialization of our algorithm consists of a set of depth maps and the corresponding camera projection matrices. The input is a video sequence taken with a camera with known intrinsics. Point correspondences are obtained with the optical flow tracker of Sundaram *et al*. [22], and the external camera parameters are estimated with incremental bundle adjustment. To capture potential loop closures we add additional SIFT [16] correspondences.

For the dense reconstruction we automatically select an equally spaced subset of about 50 images. For each we compute a depth map with semi-global matching [10]. The matching cost takes into account the surrounding frames provided by the video sequence, as proposed by Newcombe *et al*. [19]. We accumulate a simple sum of absolute differences photometric error over 14 neighboring images for 128 depth labels. Our SGM implementation uses 32 directions and an adaptive penalty for large depth label changes steered by the gradient magnitude of the image.

Camera parameters and depth values yield a coarse estimate of the scene. The depth maps contain a large amount of outliers and noise.

3. Energy model

We represent the surfaces of a scene by a set of oriented points $\mathcal{P}$. The points are initially given by the dense depth maps, *i.e*., each point $P_i \in \mathcal{P}$ corresponds to a pixel in one of the input images. It contains the surface position $\mathbf{p}_i \in \mathbb{R}^3$ and its normal vector $\mathbf{n}_i$ at this point. This way the sampling of the surface is defined by the image resolution. Surfaces covered by many pixels in the image are automatically represented at a higher resolution in the reconstruction

Points generated from different depth maps are unlikely to agree on the same surface due to noise, wrong measurements and inaccurate camera poses. We treat the point cloud as a particle simulation and define an energy that pulls close points towards a common surface:

$$E = E_{\text{smooth}} + \alpha E_{\text{data}} + \beta E_{\text{collision}}. \qquad (1)$$

E_{data} keeps the points close to their measured position $\mathbf{p}^0$, E_{smooth} and $E_{\text{collision}}$ define pairwise forces that pull the

points to a common surface and push the points to resolve self intersections, respectively. The parameters $\alpha \geq 0$ and $\beta \geq 0$ steer the importance of the data term and the collision forces respectively.

Each point in the point cloud corresponds to a depth map. As we know the (approximate) camera parameters of this view, we can reconstruct the projection ray. The point may only move along the direction of this projection ray $\mathbf{d}, |\mathbf{d}| = 1$. This way a point stays always associated with the pixel of its corresponding image. We denote the distance that the point P has been moved away from its original position $\mathbf{p}^0$ by u and optimize this quantity together with the surface normal $\mathbf{n}$ associated with this point.

The data term penalizes points that diverge from their initial position:

$$E_{\text{data}} = \sum_{P_i \in \mathcal{P}} \|\mathbf{p}_i^0 - \mathbf{p}_i\|_2 = |u_i|. \tag{2}$$

We use the non-squared error norm because it is more robust to outliers than the squared norm. The derivative with respect to u is constant and does not depend on the distance to the initial position, *i.e.*, there is a constant force that draws the point towards its initially measured position.

The energy E_{smooth} defines pairwise interactions of points and reads

$$E_{\text{smooth}} = \sum_{P_i \in \mathcal{P}} \sum_{P_j \in \mathcal{P} \setminus \{P_i\}} \eta_{\mathbf{n}_i, \mathbf{n}_j} \frac{1}{\rho_i} w_{ij} |\langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{n}_i \rangle|, \tag{3}$$

where $\langle , \rangle$ denotes the dot product. The energy measures a weighted unsigned distance of P_i to the surfaces defined by the neighboring points P_j. The energy for two points P_i and P_j is minimal if the points lie on the respective planes defined by their position and normal. The weight w_{ij} accounts for the importance of the neighbors and is defined as

$$w_{ij} = W(\|\mathbf{p}_j - \mathbf{p}_i\|, r) \frac{|\langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{n}_i \rangle|}{\|\mathbf{p}_j - \mathbf{p}_i\|}. \tag{4}$$

W is a smoothing kernel also referred to as Poly6 in the particle simulation literature:

$$W(l, r) = 0 \text{ for } l > r, \quad \text{else } \frac{315}{64\pi r^9}(r^2 - l^2)^3. \tag{5}$$

The support of the function is limited and allows to cut off neighbors with a distance greater than a distinct radius r. Neighbors close to P_i are considered more important. The second term in (4) weights the angle between the normal $\mathbf{n}_i$ and the neighboring point's position $\mathbf{p}_j$. Points directly behind or in front of a point P_i should have a high influence as they promote a different position for the surface described by $\mathbf{p}_i$ and $\mathbf{n}_i$, while a point near the tangent plane describes a similar surface at a different position. Fig. 2 shows the value of w_{ij} for varying positions of p_j. The choice of the smoothing radius r defines the size of the neighborhood and therefore directly influences the runtime as well as the topology of the reconstruction. *E.g.* two distinct surfaces with the same orientation may be joined when r is chosen too large. The radius r also relates to the depth uncertainty of the initial depth maps and should be chosen accordingly.

The function η restricts the computation of the smoothness force to points that belong to the same surface. Points with normals pointing in different directions shall not influence each other; hence we define

$$\eta_{\mathbf{n}_i, \mathbf{n}_j} = \begin{cases} \langle \mathbf{n}_i, \mathbf{n}_j \rangle, & \text{if } \langle \mathbf{n}_i, \mathbf{n}_j \rangle > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

We use the density ρ_i to normalize the energy and make it independent of the point density. The density at position p_i is

$$\rho_i = \sum_{P_j \in \mathcal{P}} W(\|\mathbf{p}_j - \mathbf{p}_i\|, r). \tag{7}$$

A special problem that arises for the reconstruction of thin objects are inconsistencies between the front-face and the back-face of a thin object. Due to noise, points with normals pointing in different directions may occlude each other. To resolve this opposed surface problem, we introduce a penalty force:

$$E_{\text{collision}} = \sum_{P_i \in \mathcal{P}} \sum_{P_j \in \mathcal{P} \setminus \{P_i\}} \eta_{-\mathbf{n}_i, \mathbf{n}_j} \frac{1}{\rho_i} w_{ij} \max(0, \langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{n}_i \rangle) \tag{8}$$

The energy measures the truncated signed distance of points P_i to the surfaces defined by the neighboring points P_j. The energy becomes non-zero if the distance of the points is positive and the normals have different directions (the dot product of the normals is negative). Point pairs P_i, P_j with this configuration are in conflict because they occlude each other but belong to opposite surfaces of the object.

4. Point cloud optimization

The gradient of the global energy (1) defines the forces that are used in an iterative scheme to optimize the position and normal of the points. The energy is non-convex due to the non-convex dependency of the weights w on the variables u_i and $\mathbf{n}_i$. We assume that a sufficient number of points is close enough to the actual surface to find a good local minimum.

We use gradient descent for fixed values of w and ρ to optimize the points and update w and ρ after each iteration, which yields a fixed point iteration scheme for w and ρ. Computing the gradient of Eq. (1) using a regularized norm $\|x\|_\epsilon = \sqrt{x^2 + \epsilon^2}$ is straightforward.

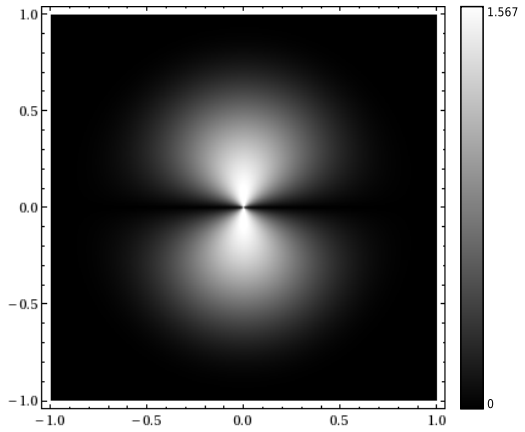

Figure 2. Weight w_{ij} with radius $r = 1$ for varying positions of $\mathbf{p}_j$ relative to $\mathbf{p}_i$. The normal $\mathbf{n}_i$ is pointing in positive y-direction. The weight is low (black) when $\mathbf{p}_j$ is far away and when the point is 'beside' $\mathbf{p}_i$ describing a different part of the surface.

The update scheme is

$$\begin{aligned} u_i^{t+1} &= u_i^t - \tau \partial_{u_i^t} E \\ \mathbf{n}_i^{t+1} &= \mathbf{n}_i^t - \tau \partial_{\mathbf{n}_i^t} E, \end{aligned} \tag{9}$$

where $\mathbf{n}_i$ and $E_{\mathbf{n}_i}$ are parameterized with spherical coordinates in an appropriate local coordinate frame.

The gradient descent scheme is very slow since the time step size τ must be chosen small to achieve convergence. We found that a mixture of coordinate descent and gradient descent significantly speeds up convergence. In coordinate descent, each variable is optimized separately.

We update $\mathbf{n}_i$ using a standard approach for normal estimation as described in [1]. It comes down to computing the covariance matrix for the neighborhood of each point. The normal estimate is the smallest eigenvector of the matrix. The sign ambiguity is resolved by the fact that the surface must point towards the camera that observes it.

The variables u_i are updated using line search along the viewing ray of each point. The energy (1) with fixed density ρ and weight w is a sum of weighted and possibly truncated $\|\|_2^1$-norms, thus the minimum is located at a position on the ray where one of the summands becomes zero. While it is not possible to evaluate the derivative at the possible minima, we can compute the derivative in the intervals between. Sorting these intervals with respect to the coordinate u_i allows us to quickly compute the minimum. The sorting can be aborted as soon as the sign of the derivative changes and the minimum is found.

Let $\hat{u}_i^t$ be the position on the ray where the energy for the point is minimal. The updated value u_i^{t+1} is

$$u_i^{t+1} = (1 - \omega) u_i^t + \omega \hat{u}_i^t. \tag{10}$$

For the coordinate descent to converge, $\omega \leq 1$. We track the minimum and maximum values of u over time for all points and decrease ω by the factor $\frac{1}{2}$ when the minimum and maximum is not altered for 80% of the points in the last iteration. If ω falls below $\frac{1}{4}$, we switch to the gradient descent scheme. Gradient descent is stopped as soon as the same criterion as for decreasing ω applies.

To resolve remaining collisions we add a last iteration using the coordinate descent scheme for the variables u with $\omega = 1$. The penalty forces defined by the energy in Eq. (8) only act *after* a collision occurs. The line search of the coordinate descent scheme allows to find a state free of collisions for points where the penalty forces act too late.

4.1. Runtime optimization

Processing point clouds with millions of points is computationally expensive. The time complexity for updating a point cloud with N fully connected points is in $O(N^2)$. Fortunately, due to the limited support of the smoothing kernel (5), the complexity can be reduced to $O(N)$ since only neighboring points within a radius r need to be considered. To this end, we use a modified implementation of the spatial hashing scheme proposed by Ihmsen *et al*. [11] to accelerate neighborhood queries. Instead of a chained hash table we implemented the open addressing scheme from [7]. The hash table allows for a fast query of the neighboring spatial cells and the particles inside these cells in $O(1)$.

To further reduce the runtime, we subsample the neighboring points. For each queried cell, three random points are sampled. We found that this subsampling hardly influences the quality of the reconstruction.

Moreover, we employ a hierarchical approach where we group multiple pixels to superpixels using the approach of Weikersdorfer *et al*. [25]. This reduces the problem size at the beginning of the optimization. We compute 5000 superpixels per image as seen in Fig. 3 and store the assignment of the original pixels, so that we can reverse the operation and return to the full resolution later on. We optimize the superpixel point cloud until convergence and transfer the positions and normals to the original point cloud. The optimization result of the superpixel point cloud yields a good approximation of the solution and greatly reduces the number of iterations spent on the original problem with N points.

4.2. Outlier removal

Due to erroneous depth maps, the initial point cloud may contain a large number of outliers, i.e., points that do not describe an actual surface of the scene. Some of them even persist after regularization.

To detect these outliers, we compute for each point P_i how many points from other images support the corresponding surface. A point P_j supports a point P_i if the position $\mathbf{p}_i$ is close to the tangent plane defined by $\mathbf{p}_j$ and $\mathbf{n}_j$. We

Figure 3. The use of superpixels reduces the size of the point cloud and significantly speeds up the optimization. **Left:** Original image. **Right:** Rendering with 5000 superpixels. The sampling density adapts to the scene depth to create superpixels with approximately equal size in space.

define the support as

$$S(P_i) = \sum_{\mathcal{N}} (1 - \delta_{I_i,I_j}) \max\left(1 - \frac{|\langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{n}_i \rangle|}{s_j}, 0\right). \tag{11}$$

s_j is the disk radius that we also use for rendering the point. The disk radius is simply computed as $s_j = \frac{\xi}{f}$, where ξ is the depth of the point and f is the focal length. The Kronecker delta is used to make sure that the support is computed only when P_i and P_j originate from different images I. The neighborhood $\mathcal{N}$ contains only points within a radius s_i. We remove points with a support of $S < 3$ after optimizing the point cloud with respect to the energy (1).

5. Point rendering and camera refinement

An important quality of a scene representation is the ability to render images from arbitrary views. We use the surface splatting approach of Botsch and Kobbelt [2] to render the point cloud. Our CUDA implementation of the algorithm allows us to render the scenes shown in Fig. 4 at about 50Hz.

We use a full image alignment as described in Newcombe *et al.* [19] to improve the initial camera parameters. The camera refinement is applied in an optional outer iteration loop that also includes updating the depth maps. Hence, it is quite costly, but it can improves details in the reconstruction. The improvement usually saturates after three iterations.

6. Results

We compare the reconstructions obtained with our approach to the PMVS algorithm of Furukawa and Ponce [6] and the KinectFusion algorithm by Newcombe *et al.* [18]. PMVS is one of the top performing reconstruction systems on the Middlebury benchmark [21], and it uses a patch-based representation, thus it potentially can deal with thin objects. KinectFusion acquires depth maps from an RGB-D camera and fuses them using a voxel grid. The voxel grid

Figure 6. Two of the 50 input images of the street sign scene. The highlighted pixels contribute to the reconstruction and are represented in the point cloud. The total number of points is about 5 million.

stores an implicit function, the zero level set of which represents the surface. The limited resolution of the grid cannot deal with very thin objects.

Datasets and test system. We used two challenging scenes that show thin as well as conventional objects. Fig. 4 depicts the scenes and their corresponding reconstruction. We use different values for the radius of the neighborhood and the parameters α, β for the two scenes. For both datasets we set the radius r to three times the disk radius of a pixel at the average depth of the scene. For the street sign scene we set parameter α to 0.05 for the superpixels level and 0.001 for the full resolution. For the toy ship we set α to 0.1 for the superpixels level and 0.001 for the full resolution. The strength of the collision forces is set to $\beta = 1$ for both datasets and both hierarchy levels of the point cloud.

The test system is an Intel Xeon X5690 3.4 GHz six core processor with an NVIDIA GeForce GTX Titan GPU. We used the GPU for computing the depth maps, the spatial hash table, and for rendering the point cloud during camera refinement. The point cloud is optimized on the CPU.

Comparison with PMVS. The reconstruction results of our system and PMVS are depicted in Fig. 5. PMVS was run on images with a resolution of 1280x720 to increase the number of reconstructed patches. Our method uses downsampled images with a resolution of 640x360 to reduce the number of points. Both approaches produce a reasonable reconstruction of the scene. The reconstruction of the sign is not consistent without an explicit collision handling. PMVS

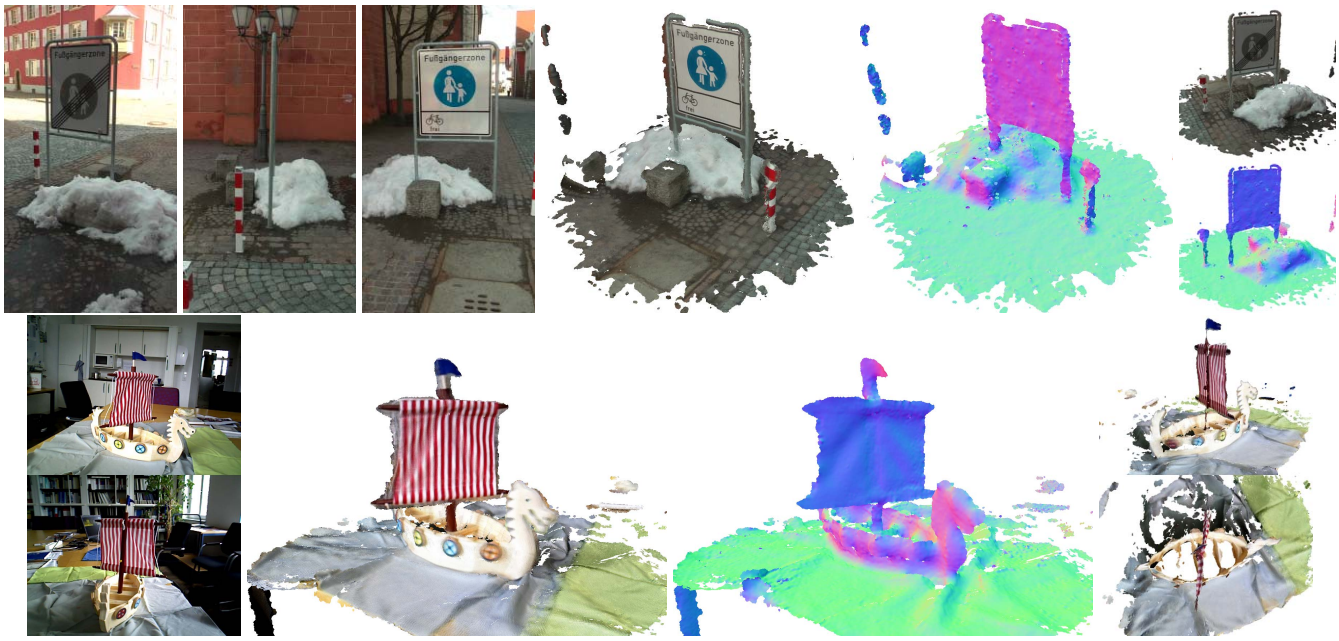

Figure 4. Test sequences with thin objects and their reconstructions. **Top:** The street sign sequence was captured with a commodity handheld camera. Beside the sign, the scene contains non-thin objects like the stone cube and the snow patch. Our reconstruction captures most of the thin sign and the majority of the surrounding objects. **Bottom:** The viking ship was acquired using an Asus Xtion depth camera. The reconstruction faithfully renders the thin sail.

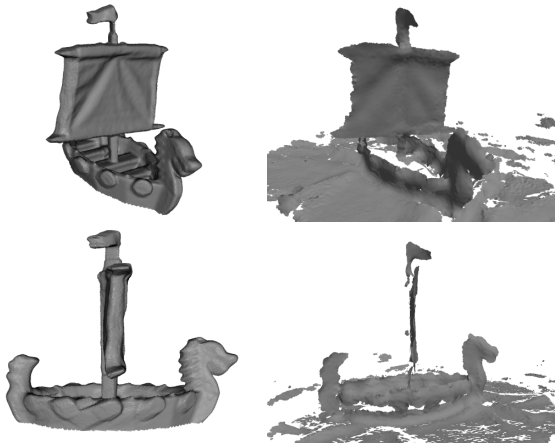

Figure 7. **Left:** The toy ship reconstructed with the KinectFusion approach by Newcombe *et al*. [18]. The implicit surface function is sampled on a regular grid and cannot represent the thin sail. The reconstruction is too thick as seen in the bottom image. **Right:** Our approach can deal with the sail but smooths away some of the details. We use a single video sequence taken with the same depth camera for the reconstruction process.

places points on the wrong side of the sign. Our method behaves similarly, if we disable the collision forces. Regions without structure, such as the plain white parts of the sign, are not reconstructed by PMVS, whereas the depth maps generated by the semi-global matching algorithm can infer the depth value from nearby edges. Hence, our reconstruction of the sign is denser.

The images in the right column of Fig. 5 show the surface normals. The normals obtained with our method are smoother and show less noise than the normals obtained with PMVS. On the other hand, our method tends to oversmooth some object borders, as can be seen from the stone cube below the sign.

The number of patches in the PMVS reconstruction is 102711. Our reconstruction is a point cloud with 5.2 million points. Each point originates from a pixel of one of the input images. Fig. 6 highlights in two input images the pixels that are used in the reconstruction.

Comparison with KinectFusion. We used the open source KinectFusion implementation available on `http://pointclouds.org/` for our experiments. KinectFusion is a real-time approach and allows for interactive steering of the acquisition of new data while the reconstruction is running. We scanned the object several times to get a good coverage. As expected, the reconstruction of the sail poses a problem to KinectFusion. We tried to scan the sail repeatedly from the front and the back side, which moves the position of the sail but does not change its thickness.

To compare our approach with KinectFusion we recorded a video sequence with the same depth camera. Instead of computing depth maps from the color images, we

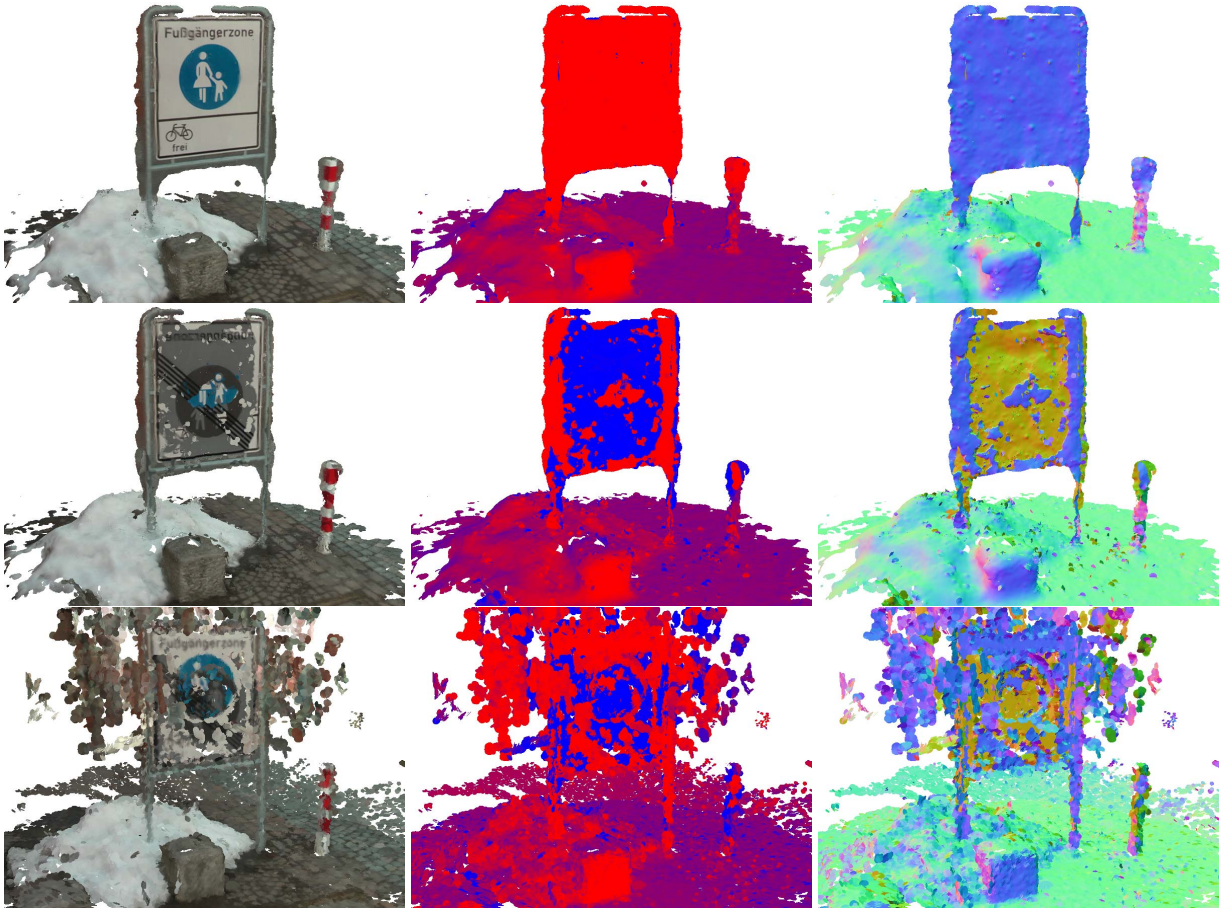

Figure 5. Reconstruction of a street sign with and without collision handling and comparison with PMVS [6]. Images from left to right: (i) rendered image with front and backfaces, (ii) color coded front and backface, (iii) surface normals. The street sign shows self-intersections when collisions are not handled. **Top:** In our reconstruction, all points are on the right side of the object, as can be seen by the uniform color in the middle image. The surface normals are consistent and smooth. **Middle:** Result of our approach without collision handling ($\beta = 0$). The front and the back side of the sign penetrate each other. Despite the collisions, the majority of the normals remain smooth. The normals of the ground show some disturbances. **Bottom:** The PMVS algorithm of Furukawa and Ponce [6] does not handle collisions. It creates many patches lying on the wrong side of the sign. The reconstruction models a larger area of the ground but also contains more clutter. The normals are noisy and some disturbances are visible. Especially the surface of the snow patch suffers from the lack of a strong regularization in the PMVS algorithm.

directly use the depth maps generated by the camera. Our system is an offline approach and does not allow to influence the acquisition. Fig. 7 shows our results compared to KinectFusion. KinectFusion yields the reconstruction with the better overall quality but also uses more data. Our system yields the better reconstruction for the sail. The quality of the normals of both reconstructions is similar.

6.1. Runtime

Most of the runtime of our system is spent on the optimization of the point cloud. The gradient descent and the coordinate descent scheme allow for a parallel implementation of this step.

Table 1 gives an overview of the runtimes for the presented datasets. The runtime mainly depends on the number of points. We can also observe that the distribution of the points influences runtime. Iterations take longer in the beginning, when the point cloud still contains a large amount of noise.

	Image size	Points	Time
Street sign	640x360	5.2 million	2h 17m
Toy ship	640x480	3.8 million	1h 24m

Table 1. Runtimes on Xeon X5690@3.4GHz + GTX Titan

7. Conclusion

We have proposed an approach for reconstructing arbitrarily thin objects. The presented method fuses the infor-

mation from multiple depth maps and produces dense surfaces. The fusion process explicitly handles intersections of opposing surfaces, a problem that has been neglected by many previous methods. The reconstructed point cloud can be rated as a dense surface representation, as each point corresponds to a pixel in the input images.

Acknowledgements We gratefully acknowledge partial funding by the ERC Starting Grant VideoLearn.

References

[1] J. Berkmann and T. Caelli. Computation of surface geometry and segmentation using covariance techniques. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(11):1114–1116, 1994. 4324

[2] M. Botsch and L. Kobbelt. High-quality point-based rendering on modern GPUs. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, pages 335–343. IEEE, 2003. 4325

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM. 4321

[4] P. Fua. Reconstructing complex surfaces from multiple stereo views. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1078–1085. IEEE, 1995. 4322

[5] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. *International Journal of Computer Vision (IJCV)*, 81(1):53–67, Jan. 2009. 4321

[6] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1362–1376, 2010. 4322, 4325, 4327

[7] I. García, S. Lefebvre, S. Hornus, and A. Lasram. Coherent parallel hashing. *ACM Trans. Graph.*, 30(6):161:1–161:8, Dec. 2011. 4324

[8] C. Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004. 4321

[9] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1430–1437. IEEE, 2009. 4321

[10] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 807–814. IEEE, 2005. 4322

[11] M. Ihmsen, N. Akinci, M. Becker, and M. Teschner. A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum*, 30(1):99–112, 2011. 4324

[12] K. Kolev, T. Brox, and D. Cremers. Fast joint estimation of silhouettes and dense 3D geometry from multiple images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):493–505, 2012. 4321

[13] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 82–96. Springer, 2002. 4321

[14] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision (IJCV)*, 38(3):199–218, July 2000. 4321

[15] M. Lhuillier and L. Quan. Quasi-dense reconstruction from image sequence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 125–139. Springer, 2002. 4322

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, Nov. 2004. 4322

[17] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nistér, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007. 4322

[18] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136. IEEE, 2011. 4325, 4326

[19] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE, 2011. 4322, 4325

[20] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps. In *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, pages 205–211. IEEE, 1997. 4321

[21] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528. IEEE, 2006. 4325

[22] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 438–451. Springer, 2010. 4322

[23] R. Szeliski and D. Tonnesen. *Surface modeling with oriented particle systems*. SIGGRAPH '92. ACM, New York, NY, USA, 1992. 4322

[24] B. Ummenhofer and T. Brox. Dense 3D reconstruction with a hand-held camera. In *Pattern Recognition*, pages 103–112. Springer, 2012. 4321

[25] D. Weikersdorfer, D. Gossow, and M. Beetz. Depth-adaptive superpixels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2087–2090. IEEE, 2012. 4324

[26] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L1 range image integration. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007. 4321