Title

Efficient Dense Reconstruction Using Geometry and Image Consistency Constraints

Authors

Shashkov, Mikhail M
Mak, Jason
Recker, Shawn
et al.

Publication Date

2015-10-01

Peer reviewed

Efficient Dense Reconstruction Using Geometry and Image Consistency Constraints

Mikhail M. Shashkov, Jason Mak, Shawn Recker, Connie Nguyen, John Owens, Kenneth I. Joy
Institute for Data Analysis and Visualization
University of California - Davis
Davis, California 95616
Email: mmshash, jwmak, strecker, csnguyen, jowens @ ucdavis.edu

***Abstract*—We introduce a method for creating very dense reconstructions of datasets, particularly turn-table varieties. The method takes in initial reconstructions (of any origin) and makes them denser by interpolating depth values in two-dimensional image space within a superpixel region and then optimizing the interpolated value via image consistency analysis across neighboring images in the dataset. One of the core assumptions in this method is that depth values per pixel will vary gradually along a gradient for a given object. As such, turntable datasets, such as the dinosaur dataset, are particularly easy for our method. Our method modernizes some existing techniques and parallelizes them on a GPU, which produces results faster than other densification methods.**

I. Introduction

In the past several years the number of applications that benefit from dense and efficient multi-view reconstruction has surged. Robotics, for example, has experienced an increased interest in quadcopters and other personal drones due to their increased affordability. Similarly, consumer attention to 3D printing has symbiotically led to increased 3D scanning needs. Both of these areas, and classical applications such as surveillance and terrain modeling, are made possible by efficient, accurate, and dense modeling of the scenes and objects in their view. State-of-the-art algorithms [1]–[3] are based mainly on sparse feature detection and matching, utilizing the Scale-Invariant Feature Transform (SIFT) algorithm [4] and other feature trackers inspired by its concept. For general scenes, these algorithms provide reasonably accurate feature tracking, camera poses, and scene structure. However, a key observation is that most of the data for the previously mentioned applications consists of sequential images of a scene whose geometry varies gradually along a gradient, i.e., there are few sharp jumps in depth on a per pixel basis. Many prior algorithms have taken this into account by doing region-based calculations [5]–[9]. Our proposed method revisits this work by creating regions with a modern superpixel algorithm, SEEDS [10], interpolating iteratively for higher fidelity, and performing computation on multiple graphics processing units (GPUs). Specifically, it consists of a two-phase algorithm that iteratively solves for unknown pixels by interpolating and optimizing ray-distance values using a multi-image consistency check via the Colored SIFT (CSIFT) descriptor [11]. The accuracy of the resulting reconstruction is only limited by the accuracy and density of the initial known reconstruction and the accuracy of the camera parameters. For example, the *dinosaur* dataset has very accurater camera positions and an initial reconstruction just using SIFT features can be made very dense via our method as shown in Figure 1.

Fig. 1. An initial sparse reconstruction, with good camera estimation, on the left. It can be made into a very dense version, on the right, by our method.

Furthermore, it is demonstrated that prior results from state-of-the-art algorithms can be substantially improved. An overview of general and sequential multi-view reconstruction is provided in Section II. The proposed algorithm is detailed in Section III, followed by results (Section IV) and conclusions and future work (Section V).

II. Background

For scene reconstruction, the input is a set of images and, in some cases, camera calibration information, while the output is typically a 3D point cloud along with color and/or normal information, representing scene structure. Camera parameters include intrinsic parameters, such as focal length, skew and principal point, as well as extrinsic or pose parameters of absolute position and orientation, and radial distortion. Intrinsics and extrinsics can be encapsulated in 3×4 projection matrices for each camera [12]. For estimating the epipolar geometry [12] between views, camera calibration and scene structure, most pipelines, such as *Bundler* [2], made use of feature tracks between images. This and other reconstruction algorithms are based on SIFT feature detection and tracking [4], but there are a number of other sparse and dense methods in the literature. Dense tracking assigns a correspondence in a

978-1-4673-9558-8/15/$31.00

destination image to each source image position and can be computed through a variety of methods [13], such as optical flow. Dense approaches especially suffer from issues such as occlusions, repetitive patterns, texture-less regions, and illumination changes, which dramatically affect the quality of the tracks and reconstruction. To alleviate this problem, recent approaches utilize either luminance and/or geometric information from images to compute a descriptor for a given interest point. With SIFT [4] and its variants [11], [14], local gradient-orientation histograms for the same-scale neighboring pixels of an interest point are used as the key entries of the descriptor. All orientations are assigned relative to a dominant orientation of the interest point, making the descriptor invariant to object orientation. The stability to occlusion, partial appearance, and cluttered surroundings is achieved by local description of the interest points. An overview of different pose estimation methods based on feature tracking were given in Rodehorst et al. [15]. Scene structure can be computed from feature tracks and projection matrices using, for example, linear or optimal triangulation [12]. Once pose and structure estimates are available, a common fine-tuning step is to perform a bundle adjustment, where the total reprojection error of all computed 3D points in all cameras is minimized using non-linear techniques [16].

There are a number of successful general reconstruction algorithms in the literature, and comprehensive overviews and comparisons were given in Seitz et al. [17] and Strecha et al. [18]. As for classical general sequential reconstruction algorithms, Pollefeys et al. [19] provided a method for reconstruction from hand-held cameras, Nistér [20] dealt with reconstruction from trifocal tensor hierarchies, while Fitzgibbon et al. [21] provided an approach for turn-table sequences.

A. Existing Dense Reconstruction Methods

There are a quite a number of dense reconstruction algorithms in the literature [6]–[9], [22]–[34]. Perhaps the best known of these algorithms is *Patch-Based Multi-View Stereo* (PMVS) [22]. This algorithm creates quasi-dense reconstructions by enforcing photo-consistency constraints on patch matching. The upgraded *Clustering Views for Multi-view Stereo* (CMVS) version [23] provided higher efficiency by intelligently grouping sets of images, and does not have memory limitations, but still suffers from non-completeness and a lack of additional constraints. Both algorithms are part of the popular *VisualSfM* program [1] for 3D reconstruction. There is even a further improvement, *Tensor-Based Multi-view Stereo* (TMVS) [35], which suffers from the same problems.

Besides patch-based multi-view stereo, there are a number of image-based rendering methods in the literature and others that can provide a fully dense and watertight reconstruction. Examples of this, in chronographical order, are shape from silhouettes [36], voxel coloring [37], and space carving [38], [39]. In summary, shape from silhouettes [36] is a form of voxel labeling, in which the visual hull of the viewed shape is computed by intersecting the projected volumes of the object's silhouettes as they appear in each input image. Voxel coloring [37] differs in that it computes a photo-consistent 3D shape by voxel projection followed by correlation of pixel colors amongst the input images. Space carving [38], [39] uses a multi-pass sweep of a plane to eliminate voxels which violate the photo consistency constraint, as does plane sweeping [40]. The main issue with these algorithms is that they typically rely on an accurate knowledge of the viewed object's silhouette, and thus have a more restricted application space than multi-view stereo methods, which do not have this requirement. Furthermore, since plane sweeping is based on homographies, there could be 'drifting' of the obtained feature tracks, and inaccuracy in the 3D point. An advantage of space carving is that it doesn't depend on texture or color, and is capable of producing a dense, water-tight reconstruction by virtue of the approach. However, because of inaccuracies in the obtained silhouettes or input camera parameters, it is seldom accurate enough to capture very fine details.

There also exist a number of volumetric methods [26]–[28]. Given the recent advances in convex optimization, globally optimal formulations have been proposed for the multi-view reconstruction problem [26], [27]. However, this line of research has so far mainly focused on the optimization methods themselves. In order to obtain highly accurate reconstruction results, the data term in energy formulations is just as important. Even the best currently available approaches have major problems in low-textured image areas, leading to visible artifacts in the obtained reconstructions. In Kostrikov et al. [30], a formulation based on an analysis of the reasons why volumetric approaches have problems in specific challenging regions is derived. A probabilistically well-founded formulation for the labeling cost that is more robust to outliers and that achieves improved reconstruction results is provided. Though great results are obtained, there is an outlier removal step that affects completeness, and is still based on the use of a cost function.

A benchmark for comparing dense reconstruction algorithms was provided by the Middlebury Multi-View Stereo Evaluation [17]. This evaluation is based on completeness percentage and accuracy. According to the results, which are updated live through user input, PMVS/CMVS is still the top performing method overall as far as completeness, with the method by Guillemaut and Hilton [34] also performing very well. It is hard to see a clear trend in accuracy; both those methods plus Kostrikov et al. [30] perform well in most evaluations. As far as runtimes, most methods take several hours on the tested datasets, when runtimes are normalized to a 3.0 GHz processor frequency. By far the fastest overall are the methods by Zach [31], Merrell et al. [33] and Chang et al. [32], which take on the order of just a few seconds, but have a lower accuracy and completeness percentage overall than the top performing methods in those categories.

Given the problems with the current literature, it is desirable to find a method that is accurate, dense, efficient, and does not require additional image information, such as silhouettes. As will be described, this can be achieved with a deceptively simple, highly parallelizable method that isn't far from a brute-

force algorithm.

III. Methodology

This section presents a densification algorithm that begins with any initial reconstruction. The key behind the algorithm is that the distance along the ray from the position of the 3D structure to the camera (ray-distance) varies smoothly for image regions corresponding to the same object. Furthermore, these regions do not have to be computed on a per-object basis since objects can be over-segmented by a superpixel algorithm [10]. The details of the algorithm are presented in Section III-A, along with GPU implementation details in Section III-B.

A. Densification Algorithm

The goal of the densification procedure is to find a ray-distance value for each and every pixel in the image sequence. This value, along with the pixel coordinate, uniquely defines the 3D location in the scene for that pixel. The algorithm depends on an initial reconstruction (input images, camera projection matrices, and initial 3D structure), for which these distance values have been computed. A SuperPixel segmentation [10] of the input images is computed to roughly segment the image into regions corresponding to the same objects. The average and standard deviation of the number of known pixel-distance pairs are computed from the initial reconstruction. Once this information has been computed, the algorithm utilizes two major procedures (*interpolate* and *optimize*), which are invoked iteratively in two phases. Pseudocode for the densification algorithm is presented in Figure 1. The algorithm will use the known ray-distance values to compute interpolated pixel-distance pairs in eligible superpixels, optimize them, triangulate them, and add the 3D structure into the scene.

Algorithm 1 Densification Algorithm

```
1: procedure DENSIFY(images, init_recon)
2:     for each image ∈ images do
3:         known_Ds ← computeKnownDists(init_recon)
4:         for i ← 0 to NUM_ITERATIONS do
5:             for each pixel ∈ EligiblePixels do
6:                 guess_D ← interpolate(known_Ds, pixel)
7:                 dist ← optimize(guess_D, images)
8:                 current_Ds ← dist
9:                 point ← triangulate(dist)
10:            known_Ds ← updateKnownDists(current_Ds)
11:        EligiblePixels ← increaseEligiblePixels(i)
```

The algorithm proceeds as follows: first, an initial reconstruction is provided. The known ray-distances are computed from the initial reconstruction and used to interpolate a distance for the given pixel being solved. After the distance is optimized, the point is triangulated and the computed distance is added to the set of currently known distances. This process proceeds for some user defined number of iterations. Details of the interpolation, and the reasoning behind the multiple passes is provided in the following section.

1) Interpolation: Certain superpixels have more known pixel-distance pairs than others. The iterative process is designed to capitalize on superpixels that have a large number of known pairs. While the algorithm can solve all superpixels in one iteration, the option of multiple iterations allows us to solve for increasingly sparse superpixels per iteration, which results in more accurate interpolation estimates at the cost of computation. Solved pixel-distance pairs from the previous iteration are added to the known pixel-distance pairs for interpolation in the next iteration. In the first phase, a superpixel is eligible for reconstruction when the number of pixel-distance pairs within it exceed an adaptive threshold. This threshold is determined by dividing the distribution of distance counts into a number of regions equal to the number of phase-one iterations (i.e. one iteration solves for all superpixels, two iterations solves for superpixels with higher than average counts in the first iteration and the rest in the second, and so forth). In phase one, this threshold must always be greater than zero, but there may exist superpixels that have no known pixel-distance pairs. An optional phase two of the algorithm solves for empty superpixels neighboring non-empty superpixels per iteration. With enough phase-two iterations, all of the pixels in the image can potentially be added to the scene but each iteration adds increasingly dubious pixels depending on the quality and distribution of the initial reconstruction.

2) Optimization: The *optimize* function fine-tunes the ray-distance estimate from the interpolation procedure for the given iteration. This procedure defines a small ray-distance search space around the given interpolation estimate to search for the best possible value. The best distance is determined using image information from a window of neighboring images in the image sequence. The effect of window size is explored in Section IV-A.

Specifically, the search space is set to the initial distance estimate plus/minus a user-defined threshold. This space is quantized into n candidate distances. The density of this quantization is limited by the desired computation time. For each candidate ray-distance, the corresponding feature track is first computed for all images within the window. This is accomplished by simply traversing the ray generated from the pixel location for the candidate ray-distance. In other words, this procedure searches along epipolar lines in the image sequence.

To ensure the best possible match is obtained, the CSIFT [11] descriptor is evaluated at each feature track location for each candidate ray-distance. This evaluation is essentially checking for image consistency for a given pixel's candidate structure point. The given pixel provides a reference CSIFT descriptor, c_{ref}. The CSIFT descriptors at the feature track locations, c_i, (generated by the candidate ray-distances) should be very similar. The algorithm chooses the candidate ray-distance that minimizes the L_1 Euclidean distance between c_i and c_{ref}, for all i, across all the feature tracks generated from candidate values. Notice there could be occlusions present at the correct distance; images at which these occur increase the error value. However, the total error is not as high

as in cases where the wrong distance is being evaluated since *most* images will coincide.

Naturally, this optimization scheme works well when the camera parameters are perfect or very accurate, and the desired feature is actually present on the epipolar line along which the optimizer searched. To account for datasets where the camera parameters are not completely accurate, the search can be expanded to include a variable number of lines parallel to the initial epipolar line. The less accurate the camera parameters are, the more extra lines should be searched, but this increases the likelihood of false positives. A good number of extra lines to add could be congruent to the reprojection error of the initial reconstruction (i.e., 2 pixels of reprojection error should require searching 2 extra lines above and 2 extra lines below).

3) Cleaning: As one might imagine, there may be a considerable amount of noise produced in failure cases and also a large amount of redundancy. Luckily, since the algorithm has redundantly added the "same feature" to the image multiple times (as many times as it appears in unique images), noise can easily be removed by performing a statistical outlier removal and then merging close points to minimize the size of the reconstruction. This optional functionality was implemented using the Point Cloud Library [41]. Additionally, it might be necessary to perform background segmentation on images from a turntable dataset to avoid reconstructing background pixels.

B. GPU Implementation

The method solves for the depth of every pixel in every image, a computation workload that can be performed in parallel. To make runtime more tractable, the method is implemented on a GPU using the CUDA programming model. As an overview, CUDA programs are written in a language based on C++, but a single program runs in parallel on a group of *blocks*, with each block having up to 1024 *threads*. The GPU can run several blocks in parallel, but within each block, it runs a group of 32 threads, called a *warp*, in SIMD fashion. Threads that are part of the same block can also share data through a small *shared memory* that is many times faster than the much larger off-chip DRAM (*global memory*). Both global and shared memory, however, require memory accesses to be regular across threads to be efficient. Another type of memory that is read-only, texture memory, exists to enable a group of threads to have irregular memory accesses as long as these accesses are spatially local in one, two, or three dimensions. Our implementation works on multiple GPUs, as the pixel-distance pairs to solve can be split independently among multiple GPUs. OpenMP is used to manage the separate GPUs.

Between the two major stages of our method, *interpolate* and *optimize*, the more expensive is *optimize*. During this stage, hundreds of candidate ray-distances for each pixel are tested in the search for the most optimal one. For each candidate ray-distance, a descriptor is computed on the neighborhood patch surrounding the coordinates where the candidate ray-distance is projected onto an image. The absolute difference (L_1 Euclidean distance) between this descriptor and the reference descriptor is computed, and the smallest average of differences across all images in the image window determines the final depth that is most optimal. For simplicity, the optimization of each pixel-distance pair is assigned to one CUDA block. Each thread computes multiple descriptors, one for each image in the image window, and saves the average difference of the descriptors with respect to the reference descriptor. Next, finding the smallest average difference across threads is simply a blockwise parallel reduction that is done in shared memory. This reduction is implemented using the NVIDIA CUB library [42]. Figure 2 illustrates the work granularity of searching for the optimal depth of a single pixel.

Traversing the ray-distance for a pixel in the reference image corresponds to traversing an epipolar line in another image. This approach of searching for the best depth leads directly to descriptors being computed along diagonal epipolar lines in the images. Although image data is stored linearly in memory, this cannot be exploited due to a lack of predictable regular memory access patterns when reading along diagonal epipolar lines. WIth this in mind, images are processed in texture memory instead of global memory. The GPU's dedicated hardware for texture fetching and its texture cache are efficient for irregular groups of memory reads, as long as the reads have spatial locality. Adjacent GPU threads in our implementation compute descriptors along the same epipolar line, enabling the memory accesses to have 2D spatial locality. Additionally, descriptors are computed with image data at a subpixel level, and texture memory is optimized for fast subpixel interpolation.

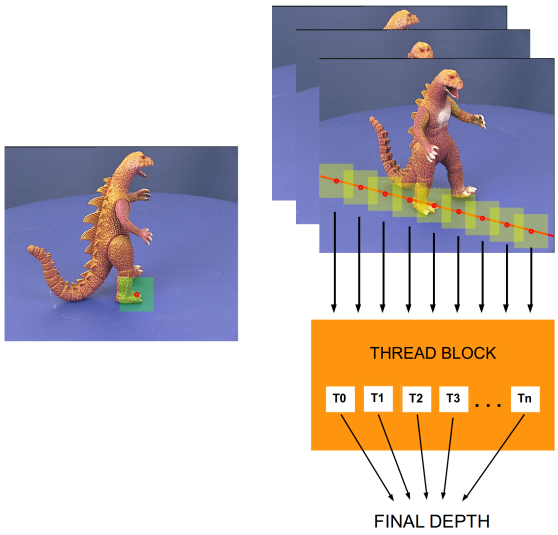


Fig. 2. A pixel whose depth is to be optimized is highlighted on the left. A reference descriptor is computed at this pixel. Candidate depths are projected to other images in the window of images. As the depth is varied during the search, its projection to other images traverses epipolar lines. Descriptors are computed on the projected points, and the L_1 Euclidean distance between each descriptor and the reference descriptor is computed. The work to compute descriptors and distances are assigned to threads within a block. A final blockwise reduction determines the smallest average distance and the best depth.

IV. Results

The proposed algorithm was analyzed for its general behavior and processing time on several real datasets. The effect of window size in the optimization routine was tested on datasets with ground-truth information available. The algorithm was implemented in C++, parallelized with CUDA, and all results were generated on an Ubuntu 12.04 Linux machine with an Intel Xeon E5-2637 and four K40C NVIDIA GPUs.

A. Window Size Justification

The optimization procedure of the algorithm performs a photo-consistency check across neighboring images in the sequence. It is necessary to experiment with the amount of neighboring images (the window size). A window size of two refers to analyzing one image before and one after the current image, a window size of four corresponds to analyzing the two images before and two after, and so on. First, the ray-distances were generated using the ground-truth 3D information and camera data. The standard deviation, σ, of the ground-truth ray-distance values, s_i, was computed. To analyze the robustness of the *optimize* procedure, noise was introduced into the ground-truth distances at varying levels, $\epsilon = [0, 1, 2, 3]$ and the *optimize* procedure was executed using the noisy estimate, s_i' and a specified window size. The noisy estimate is computed by sampling a uniform distribution in the range of $s_i' \in [s_i - \sigma \times \epsilon, s_i + \sigma \times \epsilon]$. Reported error values correspond to average distance error between computed and ground-truth 3D points, for a full reconstruction. Table I shows the results of the test when run on the ground-truth Oxford Dinosaur dataset [43]. For all tested window sizes, and low noise levels $\epsilon = 0$ and $\epsilon = 1$, similar error values overall were obtained. Therefore, using a window size of two is usually justified since it is less expensive to compute and provides the same results. For larger window sizes there is a risk of running into occlusions and other wide baseline effects that might affect scoring. For high noise levels, such as $\epsilon = 2$ and $\epsilon = 3$, errors were significantly higher, and relatively constant across window sizes. For our results on sequential images, we choose a window size of four to keep runtime small, while still obtaining a sufficient photo-consistency check.

TABLE I
Average 3D positional error at varying window sizes and noise levels.

Window size	2	4	6	8	34
Score $\epsilon = 0$	0.0013	0.0014	0.0016	0.0018	0.0093
Score $\epsilon = 1$	0.0014	0.0015	0.0017	0.0019	0.0086
Score $\epsilon = 2$	0.0287	0.0236	0.0207	0.0191	0.0239
Score $\epsilon = 2$	0.0815	0.0682	0.0612	0.0584	0.0758

B. Results on Real Datasets

To analyze the efficacy of the algorithm, the densification procedure was executed on the Middlebury Temple [17] dataset benchmark. Results, as displayed in Figure 3, show that the densification procedure outperforms PMVS/CMVS in runtime (1.5 minutes vs. 4.5 hours). We also found them to be similarly complete.

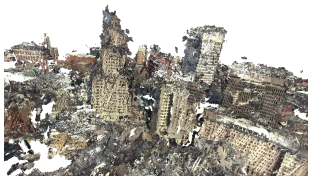
(a) Reference Image

(b) Our Result

Fig. 3. Reconstructions of the Template Dataset. Reference image of the Temple dataset (a). Complete dense reconstruction of the Temple dataset via the proposed method (b). For this result, 39 images of size 640x480 were used. The result of the proposed method outperforms PMVS/CMVS in runtime (1.5 minutes vs. 4.5 hours) and is similarly complete.

In addition, we tested our method on a dataset with aerial images, since aerial scenes can often resemble turntable sequences. Figure 4 shows a reference image from the Brown Site 22 dataset [44], and a dense reconstruction of the scene produced by our method. As shown in the figure, our result is noisy and unable to accurately capture many details in the scene. Like many other algorithms, our method struggles with aerial scenes due to their inherent challenges, including the difficulty of estimating accurate camera parameters and the low resolution of the images with respect to the relatively large scale of the scene in real life. The latter prevents our method from performing accurate region segmentation. However, considering the size of the images in the dataset (1280x720), we achieve a reasonable runtime of 32 minutes. We believe our method, with its use of GPUs, can enable the reconstruction of very large-scale aerial scenes. In the future, we hope to explore ways to improve accuracy, particularly by incorporating other sensor data, such as GPS, in the process to recover accurate camera parameters.

(a) Reference Image



(b) Our Result

Fig. 4. The result of our method on Brown22, an aerial dataset containing 243 images of size 1280x720.

As discussed earlier, one limitation to the final correction stage of the algorithm is that it can only be used for sequential

image streams, unlike PMVS/CMVS which is more general. However, there are a great number of relevant scenarios, spanning many important applications, where an accurate and dense reconstruction is necessary, and the proposed algorithm is capable at meeting these requirements. Finally, Figure 5 shows a number of dense reconstructions obtained with the proposed method.

V. Conclusions and Future Work

This paper presented an updated, efficient take on an old deceptively simple algorithm. It was modernized in many ways, including region segmentation via the SEEDS Super-Pixel algorithm [10], the use of an effective and fast descriptor, CSIFT [11], and parallelization on the GPU. Additionally, it proposes a two phase approach that allows for even denser reconstructions when initial reconstructions are well distributed in 2D space.

Ideally, future improvements on this algorithm would lower runtime by exploring data structures to enable more efficient memory access patterns on the GPU. Reconstruction of aerial scenes can be revisited by incorporating more sensor data. Accuracy improvements can also be made by applying geometric constraints from known geometries (such as the rectangular nature of buildings).

Acknowledgments

This work is supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, and by the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000. We would also like to thank NVIDIA for equipment donations. We give special thanks to our colleagues at the Institute for Data Analysis and Visualization for the useful discussions and support.

References

[1] C. Wu, "Towards linear-time incremental structure from motion," in *3D Vision - 3DV 2013, 2013 International Conference on*, June 2013, pp. 127–134, 10.1109/3DV.2013.25.

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3D," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*. New York, NY, USA: ACM, 2006, pp. 835–846.

[3] M. Goesele, N. Snavely, C. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *Proceedings of ICCV 2007*, 2007.

[4] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal On Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] H.-H. Vu, P. Labatut, J.-P. Pons, and R. Keriven, "High accuracy and visibility-consistent dense multiview stereo," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 5, pp. 889–901, May 2012, 10.1109/TPAMI.2011.172.

[6] J. Isidro and S. Sclaroff, "Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 1335–1342 vol.2, 10.1109/ICCV.2003.1238645.

[7] Y. Wei and L. Quan, "Region-based progressive stereo matching," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, June 2004, pp. I–106–I–113 Vol.1, 10.1109/CVPR.2004.1315020.

[8] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof, "Dense reconstruction on-the-fly," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1450–1457, 10.1109/CVPR.2012.6247833.

[9] Z. Zhang and Y. Shan, "A progressive scheme for stereo matching," in *Revised Papers from Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, ser. SMILE '00. London, UK, UK: Springer-Verlag, 2001, pp. 68–85. [Online]. Available: http://dl.acm.org/citation.cfm?id=646489.694937

[10] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "Seeds: Superpixels extracted via energy-driven sampling," in *Computer Vision ECCV 2012*, ser. Lecture Notes in Computer Science, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Springer Berlin Heidelberg, 2012, vol. 7578, pp. 13–26, 10.1007/978-3-642-33786-4-2. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33786-4-2

[11] A. E. Abdel-Hakim and A. A. Farag, "CSIFT: A SIFT descriptor with color invariant characteristics," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 1978–1983, 10.1109/CVPR.2006.95.

[12] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[13] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal On Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[14] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 3951, pp. 404–417, 10.1007/11744023.32. [Online]. Available: http://dx.doi.org/10.1007/11744023.32

[15] V. Rodehorst, M. Heinrichs, and O. Hellwich, "Evaluation of relative pose estimation methods for multi-camera setups," in *International Archives of Photogrammetry and Remote Sensing (ISPRS '08)*, Beijing, China, 2008, pp. 135–140.

[16] M. I. A. Lourakis and A. A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm," Institute of Computer Science – FORTH, Heraklion, Crete, Greece, Tech. Rep. 340, Aug. 2000.

[17] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 519–528.

[18] C. Strecha, W. von Hansen, L. J. V. Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *CVPR'08*, 2008.

[19] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, pp. 207–232, 2004.

[20] D. Nistér, "Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors," in *ECCV '00*. London, UK: Springer-Verlag, 2000, pp. 649–663.

[21] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D model construction for turn-table sequences," in *Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*. London, UK: Springer-Verlag, 1998, pp. 155–170.

[22] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[23] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski, "Towards internet-scale multi-view stereo," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 1434–1441.

[24] C. Esteban and F. Schmitt, "Silhouette and stereo fusion for 3D object modeling," in *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, Oct 2003, pp. 46–53.

[25] C. Hernandez, G. Vogiatzis, and R. Cipolla, "Probabilistic visibility for multi-view stereo," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–8.

[26] K. Kolev, M. Klodt, T. Brox, and D. Cremers, "Continuous global optimization in multiview 3D reconstruction," *International Journal of Computer Vision*, vol. 84, no. 1, pp. 80–96, 2009, 10.1007/s11263-009-0233-1. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-0233-1

Fig. 5. Results from the proposed method for the Dinosaur and Conch datasets. The first column shows an example input image, the second shows the initial sparse reconstruction used as input, the third shows results from the proposed method, and the last column shows the result of CMVS/PMVS as implemented by VisualSfM. For the Conch shell dataset with 216 images of size 640x480, our method takes 3 minutes. For the Dinosaur dataset with 36 images of size 720x576, our method takes 1.3 minutes.

[27] K. Kolev, T. Pock, and t. . A. b. . P. s. . E. y. . . i. . X. l. . H. p. . . n. . . u. . h. a. . . p. . S. a. . B. Cremers, D.

[28] G. Vogiatzis, P. Torr, and R. Cipolla, "Multi-view stereo via volumetric graph-cuts," in *Computer Vision and Pattern Recognition (CVPR), 2005 IEEE Conference on*, vol. 2, June 2005, pp. 391–398 vol. 2, 10.1109/CVPR.2005.238.

[29] S. Ricco and C. Tomasi, "Video motion for every visible point," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 2464–2471, 10.1109/ICCV.2013.306.

[30] I. Kostrikov, E. Horbert, and B. Leibe, "Probabilistic labeling cost for high-accuracy multi-view reconstruction," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1534–1541, 10.1109/CVPR.2014.199.

[31] C. Zach, "Fast and High Quality Fusion of Depth Maps," *Proc. ASP/UI Symp. Close-Range Photogrammetry*, pp. 1–18, 2008.

[32] J. Y. Chang, H. Park, I. K. Park, K. M. Lee, and S. U. Lee, "GPU-friendly multi-view stereo reconstruction using Surfel representation and graph cuts," *Comput. Vis. Image Underst.*, vol. 115, no. 5, pp. 620–634, May 2011, 10.1016/j.cviu.2010.11.017. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2010.11.017

[33] 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 *Computer Vision (ICCV), 2013 IEEE International Conference on*, Oct 2007, pp. 1–8, 10.1109/ICCV.2007.4408984.

[34] J.-Y. Guillemaut and A. Hilton, "Joint multi-layer segmentation and reconstruction for free-viewpoint video applications," *Int. J. Comput. Vision*, vol. 93, no. 1, pp. 73–100, May 2011, 10.1007/s11263-010-0413-z. [Online]. Available: http://dx.doi.org/10.1007/s11263-010-0413-z

[35] T.-P. Wu, S.-K. Yeung, J. Jia, and C.-K. Tang, "Quasi-dense 3D reconstruction using tensor-based multiview stereo," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 1482–1489, 10.1109/CVPR.2010.5539796.

[36] R. Szeliski, "Rapid octree construction from image sequences," *CVGIP: Image Underst.*, vol. 58, no. 1, pp. 23–32, Jul. 1993, 10.1006/ciun.1993.1029. [Online]. Available: http://dx.doi.org/10.1006/ciun.1993.1029

[37] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *Int. J. Comput. Vision*, vol. 35, no. 2, pp. 151–173, Nov. 1999, 10.1023/A:1008176507526. [Online]. Available: http://dx.doi.org/10.1023/A:1008176507526

[38] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *Int. J. Comput. Vision*, vol. 38, no. 3, pp. 199–218, Jul. 2000, 10.1023/A:1008191222954. [Online]. Available: http://dx.doi.org/10.1023/A:1008191222954

[39] S. Lazebnik, E. Boyer, and J. Ponce, "On computing exact visual hulls of solids bounded by smooth surfaces," in *Computer Vision and Pattern Recognition (CVPR), 2001 IEEE Conference on*, vol. 1, 2001, pp. I–156–I–161 vol.1, 10.1109/CVPR.2001.990469.

[40] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Computer Vision and Pattern Recognition (CVPR), 2007 IEEE Conference on*. IEEE, 2007, pp. 1–8.

[41] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[42] D. Merrill and NVIDIA-Labs, "Cuda unbound (cub) library."

[43] Oxford Visual Geometry Group, "Multi-view and Oxford Colleges building reconstruction," http://www.robots.ox.ac.uk/~vgg/, Aug. 2009.

[44] F. Calakli, A. O. Ulusoy, M. I. Restrepo, G. Taubin, and J. L. Mundy, "High resolution surface reconstruction from multi-view aerial imagery," in *Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, ser. 3DIMPVT '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 25–32, 10.1109/3DIMPVT.2012.54. [Online]. Available: http://dx.doi.org/10.1109/3DIMPVT.2012.54