Scale Robust Multi View Stereo

Christian Bailer, Manuel Finckh, and Hendrik P. A. Lensch

Computer Graphics, Tübingen University
72076 Tübingen, Germany

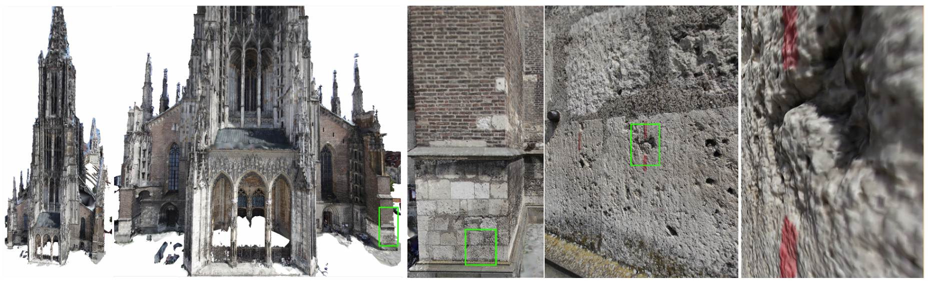


Fig. 1. Reconstructed 3D model of the Ulm Minster (German: *Ulmer Münster*) dataset with very large resolution differences, created by our method. Note the huge scale difference from the left most image (about 160m hight) to the right most image (about 3cm).

Abstract. We present a Multi View Stereo approach for huge unstructured image datasets that can deal with large variations in surface sampling rate of single images. Our method reconstructs surface parts always in the best available resolution. It considers scaling not only for large scale differences, but also between arbitrary small ones for a weighted merging of the best partial reconstructions. We create depth maps with our GPU based depth map algorithm, that also performs normal optimization. It matches several images that are found with a heuristic image selection method, to a reference image. We remove outliers by comparing depth maps against each other with a fast but reliable GPU approach. Then, we merge the different reconstructions from depth maps in 3D space by selecting the best points and optimizing them with not selected points. Finally, we create the surface by using a Delaunay graph cut.

1 Introduction

Thanks to fast camera calibration methods [1,2] today, it is possible to automatically calibrate thousands of images in space, i.e., to determine relative capturing position and camera pose for each image. An interesting application for such calibrated datasets is dense 3D reconstruction as this allows for automatically creating accurate 3D models of large objects or even whole city parts out of common photographs. For famouse locations, these photographs can even simply be taken from image portals like Flickr. However, there are still very few methods that can deal with such datasets, even if there are over 50 different Multi View Stereo approaches today. For example, this can be seen in the submission list of the Middlebury evaluation portal [3]. Many methods, which are based on voxels [4,5,6] or visual hulls [7,8,9] are not flexible enough to reconstruct big open datasets. Others, mainly those based on multiple depth maps or patches [10,11,12,13] are more flexible, but they usually use simple heuristics to find compatible images for depth map or patch creation and ignore the fact that images can show objects in strongly differing resolutions. Thus, they are primarily suited for structured datasets, where e.g. neighboring images in the image stream are also neighbors in space. To be able to reconstruct more general datasets, where image views can be arbitrarily scattered in space, an approach additionally needs some kind of image selection method that determines which images are worthwhile to compare because they show the same objects. Good image selection is important for accuracy but not trivial, because it depends on the surface distance (Figure 3(a)) and not the most similar images are the best choice. Furthermore, an algorithm should also consider if there are scale differences in the dataset, i.e., if objects can be seen in different images from different distances in different scales. Otherwise this usually leads to loss of reconstruction detail. Our approach provides solutions for both problems. For image selection we extend the image selection approach of Goesele et al. [14]. Scale robustness is achieved by considering scale in all stages of this work. These are besides image selection, multi depth map filtering, point selection, point optimization and meshing. Multi depth map filtering is an efficient, GPU parallelizable way to remove outliers by comparing depth map against each other and point selection and optimization selects the assumed best points in space and optimizes them in position with worse not selected points. This is only possible if we carefully consider their assumed accuracy.

2 Related Work

One of the most constrained methods to reconstruct huge datasets is reconstruction from street level video. Pollefeys et al. [15] demonstrated that this can even be done in realtime. Frahm et al. [16] developed an algorithm which is able to deal with millions of images. However, their method seems to be primarily interesting for image preselection and calibration and not for massive stereo reconstruction, as they demonstrated this only locally for single objects.

Jancosek et al. [17] proposed a scalable patch based method which can deal with huge unstructured image datasets, but it is not scale robust and has only a simple image selection heuristic. Goesele et al. [14] were probably the first that proposed a robust image selection method for huge unstructured datasets, with large scale differences. This allowed them to create good depth maps for such datasets. However, they didn't consider point accuracy for meshing. Thus, the overall method is not scale robust. They addressed this problem in recent publications where they presented scale robust meshing methods for depth maps [18,19]. Furukawa et al. [20] clustered huge unstructured datasets into clusters of similar visibility and scale. These clusters can then be independently processed by simple Multi View Stereo algorithms without image selection, like their previous work [12]. Afterwards, they refine the point clouds of different clusters by comparing them against each other. Thereby, they also consider point accuracies and remove points from clusters if a surface part is represented by another cluster in clearly higher resolution. However, they don't demonstrate meshing for their point clouds.

3 Overview

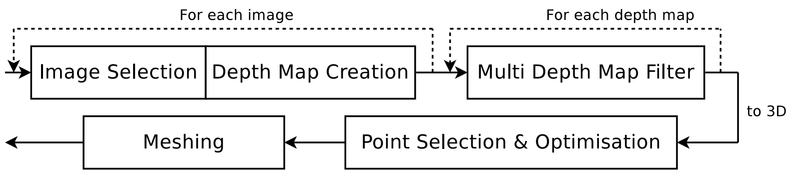
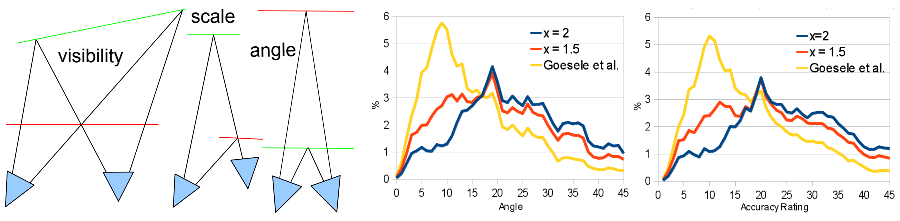


Fig. 2. The pipeline of our approach.

This section gives an overview over the proposed pipeline. Before our approach can work with raw image datasets, these must be calibrated in space at first. We use the publicly available Bundler software [1] for this task. Bundler delivers rotation matrices, translation vectors, distortion coefficients and focal lengths for all images, that could successfully be calibrated and a sparse set of surface points. This data is the input to our pipeline which is outlined in Figure 2. We create depth maps out of the perspective of each image, by matching it to other images that are selected as described in Section 4. The selection process prefers images with similar scale, large view angles and a big overlap in field of vision to the reference image. Additionally, we demand that overlaps and view angles are not too similar in different selected images. Our depth map algorithm which is described in Section 5 is GPU based and can perform normal optimization similar to region growing based approaches. It further filters out outliers. After creating and independently filtering all depth maps, they are compared against each other and points in depth maps that have not enough support by other depth maps or have too many visibility conflicts are filtered out by the approach in Section 6. Successively, points are transformed into the 3 dimensional space and ordered into a kd-tree. Because depth maps often have redundancy to one another the point cloud is very dense in some regions. Thus, the best points are selected and the rest of the points are only used to optimize the selected points in position as described in Section 7. Thereby, the redundancy can be used to remove surface noise. Selected local outliers are also removed in the optimization process by pulling them onto the surface. Finally, the mesh is created by a Delaunay graph cut described in Section 8.



(a) (b)

Fig. 3. a) Good image selection depends on the surface distance. Green indicates a good image distance. b) Our angular weighting scheme reduces the influence of close-by views, here demonstrated on the Ulm Minster data set.

4 Image Selection

In this section we present an image selection heuristic that considers scale and robust stereo reconstruction. To find suitable images for stereo matching with diversity, Goesele et al. [14] demand view angles above 10 degrees between all images including the reference image to which the other images are matched for depth map creation, if possible. However, works like [21] showed that 10 degrees are by far not enough for accurate stereo matching. Thus, we extend their approach by demanding especially big view angles between the reference and the other images. Furthermore, our approach prioritizes images, where regions of the reference image are visible that are hardly or not visible in already selected images.

To find suitable images I_s for stereo matching with a reference image R we use sparse Structure From Motion feature points f, delivered by Bundler [1]. These points have already visibility information and are necessary, because knowledge of the surface distance is essential for a good selection (Figure 3(a)). To rate an image, we use similar to Goesele et al. a weighted sum of all feature points visible in R and the tested image:[1]

$$g_R(I) = \sum_{f \in F_R \cap F_I} w_a(f) w_s(f) w_c(f) \qquad (1)$$

where I is an image, w_a is the angle weighting, w_s the scale weighting, w_c the covering weighting and F_X the set of feature points visible in image X. Our angle

[1] Please consult [14] for a detailed motivation for using a weighted sum.

weighting scheme differs from the reference and the already selected images:

$$w_a(f) = min\left(\frac{\angle_{I,R}(f)}{\alpha_{max}}, 1\right)^x \prod_{J|J\in I_s \cap f\in F(J)} min\left(\frac{\angle_{I,J}(f)}{\beta_{max}}, 1\right)^y , \qquad (2)$$

where$\angle_{X,Y}(f)$ is the view angle difference at point f for rays from the center of projection of image X and Y and I_s the set of already selected images. As I_s increases over time all images must be rerated after each selection of the best image. We used mostly $\alpha_{max} = 35°$, $\beta_{max} = 14°$, $x = 1.5$ and $y = 1$. Our coverage weighting, which favors feature points that are only sparsely covered by already selected images is given by:

$$w_c(f) = \frac{r_I^*(f)}{r_I^*(f) + \sum_{J|J\in I_s \cap p\in P(J)} r_J^*(f)} . \qquad (3)$$

A simple choice for r_X^* that ignores scale would be $r_X^* = 1$. However, we don't want images with good scales be covered by images with bad scales, so we set it to $r_X^*(f) = min\left(s_R(f)^2/s_X(f)^2, 1\right)$. This can be especially important at depth discontinuities. There, it might not be possible to select images with good scale on both sides of the discontinuity. We also changed the scale weighting of Goesele et al. to be more strict, as we don't want big angles with bad scale to be better rated than small angles with good scale. If $s_X(f)$ is the scale at f in an image and $r = s_R(f)/s_I(f)$ our scale weighting is calculated as:

$$w_s(f) = \begin{cases} 0 & r > 1.8 \\ r^2 & 1 < r \leq 1.8 \\ 1 & 1/1.6 < r \leq 1 \\ \left(\frac{1.6}{r}\right)^2 & else \end{cases} . \qquad (4)$$

As Figure 3(b) shows the provided angles are often smaller than 35°. Anyway, for $x = 1.5$ we get more than twice as much large angles than Goesele et al. Furthermore, our approach also performs better for the accuracy weighting $\angle max(1, 1/r)$. Overall we get better view angles not at the cost of worse scaling.

5 Depth Map Creation

After having selected images, we now can compute depth maps and filter them on the GPU. In contrast to common GPU based methods our algorithm can perform normal optimization like region growing based approaches [14,12] to get high quality depth maps. Our method is based on the stochastic PatchMatch algorithm [22] for texture synthesis. Our depth map estimation pipeline is illustrated in Figure 4.

First, we initialize the depth map by inserting feature points provided by bundler as depth values. Then, we propagate depths into neighboring pixels if there is no depth value or the matching error of the existing depth is bigger than the matching error of the propagated depth. The original algorithm propagates

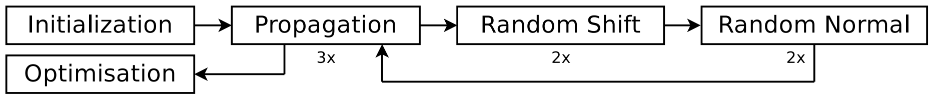


Fig. 4. The depth map creation pipeline

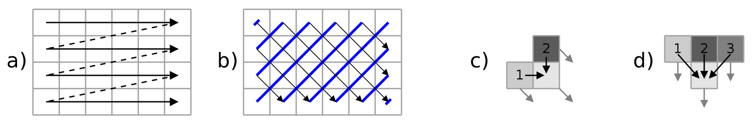


Fig. 5. Ways to propagate depth values through the depth map.

values for each pixel from the left and from the top like in Figure 5(c), i.e., $(x-1, y) \rightarrow (x, y)$ and $(x, y-1) \rightarrow (x, y)$. If pixels are processed in the order of Figure 5(a), good values can propagate arbitrary far to the right and to the bottom in only one pass. We found out, that this approach can directly be parallelized by processing the blue lines in Figure 5(b) in parallel, which leads to similar results as Figure 5(a). However, the parallelism depends on the position. Thus, we use horizontal processing lines and the three propagation directions of Figure 5(d). We call this *downward propagation*. Additionally, we perform *upward propagation* at the same time. Up and downward propagation is performed in all uneven propagation steps. Left and rightward propagation in all even steps.

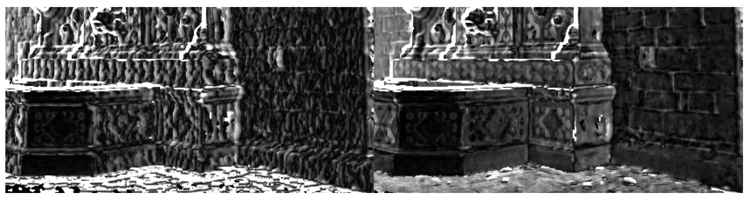


Fig. 6. Difference to ground truth. Left: No Normal Tuning Right: Tuning with random normals. Artifacts disappear nearly completely after the second propagation step.

After propagation, depth values are randomly shifted in depth and the new depth is kept if the matching error of the shifted depth is smaller. We perform three shifts with uniform distributions with $16\frac{d}{f}$, $8\frac{d}{f}$ and $4\frac{d}{f}$ maximum shift distance, where d is the depth value and f the focal length of the image. Depth values have initially no normals, hence, we assign normals with uniform random direction and surface angle to them if the matching error decreases:

$$n_x = sin(\beta) \cdot tan(\alpha), \; n_y = cos(\beta) \cdot tan(\alpha) \; , \tag{5}$$

where n_x and n_y are the gradients of the tangent plane in x and y direction. As Figure 6 shows, random normals are sufficient as the good ones get distributed over the depth map in the next propagation step. Allover, we perform 3 propagation steps as this usually almost leads to convergence. Even with two steps we mostly retrieved good results. Important for fast convergence of sloped surfaces is the usage of normal information to propagate into tangent direction in propagation step 2 and 3. Finally, we optimize the depth values and normals with a gradient descent similar to region growing based approaches. As matching error E over all selected images I_S to the reference image R we use a self weighted average over all 1 *minus Normalized Cross Correlation* $(1 - NCC)$ scores:

$$E = \frac{\sum_{I \in I_S} 1 - NCC_{R,I} \frac{1}{1-NCC_{R,I}}}{\sum_{I \in I_S} \frac{1}{1-NCC_{R,I}}} = \frac{\overline{I_S}}{\sum_{I \in I_S} \frac{1}{1-NCC_{R,I}}} \ . \tag{6}$$

The error function was occlusion robust in our tests, as it strongly prefers small errors. It performed even at non occluded surface parts better than a common weighting.[2] For details about matching with normals please consult [12].

Depth Map Filtering To filter erroneous values in the depth map, we discard all depth values that have a $(1 - NCC)$ matching error above a threshold in at least n images:

$$n = max(c, min(3, \overline{I_v}/2)) \ , \tag{7}$$

where $\overline{I_v}$ is the number of images that can see the point if there is no occlusion, i.e., the point lies not outside of the image area if it is projected into the image. We use $c = 2$ in most of our tests and $c = 1$ for sparse datasets.

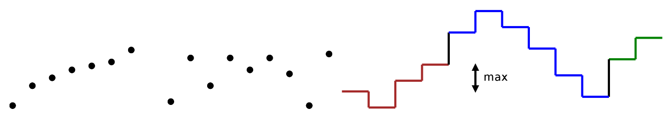


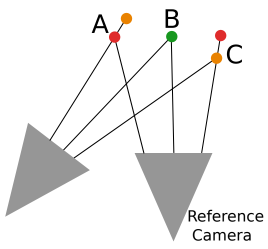
Fig. 7. Left: Points don't resemble a surface. Right: Segmentation of our filter.

Additionally, we filter outliers if they don't resemble a surface (see Figure 7). This is done by segmenting the images by depth discontinuities. If there is a way from one point in the depth map to another point without crossing a depth discontinuity above a threshold, both are in the same region. Thereafter, we delete all regions which have less than 15 points. The depth threshold is set to $2\frac{d}{f}$, where d is the current depth and f the focal length of the image.

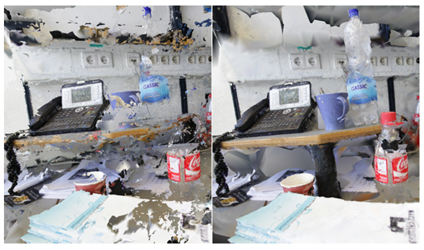
6 Multi Depth Map Filtering

In this section we describe how outliers and other inconsistent points are removed in a fast GPU parallelizable but very reliable way that works for arbitrary

[2] For tests in this section we used the datasets of [23] with ground truth depth maps of [24]. See: http://cvlab.epfl.ch/alumni/tola/daisy.html.



(a)



(b)

Fig. 8. a) Depth maps can be consistent (B), inconsistent (A, C) or show different non conflicting surfaces (not shown). b) Difficult scene for MVS, because of textureless and transparent objects and bad camera calibrations. Left: Even the Delaunay Graph cut cannot correct the unfiltered depth maps. Right: With our additional multi depth map filtering we can recover a lot of surface parts.

datasets. The idea is to compare depth maps against each other as proposed by Merrell et al. [25] and to filter out points in depth maps that are inconsistent to other depth maps. To determine which depth maps should be compared to a reference depth map, that we want to filter, we use our image selection algorithm. Especially image selection makes the approach powerful because the depth maps it finds show not only the same surface parts as the reference depth map, but are also very unlikely to have the same outliers, as they differ in view angle and thus in the set of images they were created from.

Figure 8(a) shows the different cases that can occur when corresponding points in two depth maps are compared to each other. In order to compare them, we assign a support value to each depth value. A depth value gains support for similarity (case B in Figure 8(a)) and looses support for visibility conflicts (case A and C). Hereby, we take only the difference of case A and C conflicts as negative support into account like in [25], because a point can not be too near and too far at the same time. The support for a depth value is defined as:

$$S_R(x,y) = \sum_{D \in D_s} B_{R,D}(x,y) r_b - \left| \sum_{D \in D_s} A_{R,D}(x,y) - \sum_{D \in D_s} C_{R,D}(x,y) \right|, \qquad (8)$$

where $A_{R,D}$,$B_{R,D}$ and $C_{R,D}$ are booleans that determine the case, A, B, or C between the reference depth map R and depth map D. r_b is a constant, which is set to 2 in this work, because we think case A and C are despite the difference still stronger affected by outliers in other depth maps than case B. After all depth maps were processed and points which meet the condition $S_R(x,y) < 1$ are filtered out.

We test for conflicts similar to [25], but when we render a depth map for case C into the view of the reference depth map, we render each depth value into the 4 closest pixels[3] instead of only one to avoid gaps (see Figure 9) that can

[3] Depth values can only be overwritten by smaller ones.

otherwise strongly occur for large view angles and scale differences. There is a conflict if a depth value of the rendered depth map is significantly smaller than the value of the reference depth map.

Case B could also be decided like case C. However, we handle it like case A because we think this is more accurate. This is done by transforming points of the reference depth map into the view of another depth map and comparing depth values there on the rays of the other view. Thereby, direct comparisons are possible without rendering. For the acceptance threshold we consider for case B only the accuracy of the referenced depth map. For conflicts we take the worse of both accuracies:

$$T_A = T_C = c \cdot max(d_R(X), d_{D_i}(X)),\ T_B = \frac{c}{2} \cdot d_R(X)\ , \tag{9}$$

where c is a constant we set to 1.6, X is the 3D representation of the tested point in the reference depth map, R the reference depth map, D_i another depth map, and d the sampling distance of a depth map at a point, i.e., the depth the point would have in that depth map over the focal length of the depth map. We do not test for conflicts if $1.6 \cdot d_{D_i}(X) > d_R(X)$ to avoid border effects at depth discontinuities, because of inaccurate borders in low resolution depth maps. After multi depth map filtering, we again filter depth maps with the segmentation filter of Figure 7 to remove single surviving outliers.

The approach can also be used to filter depth values in small scale depth maps. Therefore, the selection algorithm can be modified to prefer depth maps with 1.6 times or higher sampling rates as the reference depth map and depth values are filtered out if such a depth map is found for a pixel position. This is not necessary for scale robustness, but speeds up the rest of our approach. Practically, we use the filter but don't select extra high resolution depth maps.

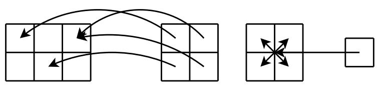


Fig. 9. When rendering a depth map into another view some pixels are covered multiple times, others never. This can be avoided by rendering points into 4 pixels at once.

7 Point Selection and Optimization

After all depth maps are filtered the remaining depth values are transformed into the 3 dimensional space. We transform only one depth value per 2x2 pixel window, as depth maps from stereo usually have no pixel accuracy. Alternatively, only half sized depth maps could be created. Anyway, we created full depth maps and used a small bilateral filter on the depth map to keep the extra information. In 3D space, each point gets, besides position and color, a normal, that is already delivered by some depth map algorithms and a scale value and influence radius that we assume as equal for the moment: $I = S = \frac{2z}{f}$, where z is the depth value in the depth map, f the focal length of the depth map and the factor of 2 is set because we used only one value per 2x2 pixel window. Then, all points are defined as primary points that have no primary points with smaller I in their own influence radius. This is done by testing the points with the smallest I first. Neighbors of a point in the same depth map can slightly lie inside the influence radius and are ignored to keep the regular depth map structure. Primary points then represent the surface approximately in the resolution of the best depth map and are most likely the most accurate points. Nevertheless, they can still be improved in position by using other points with similar scale to remove surface noise. We use a modified minimum least square algorithm [26] for this:

$$p^*_{i+1} = \frac{\sum_{q \in P_i} w(||p_i - q||)(S_p/S_q)^2 q}{\sum_{q \in P_i} w(||p_i - q||)(S_p/S_q)^2} \tag{10}$$

$$n_{i+1} = \frac{\sum_{q \in P_i} w(||p_i - q||)(S_p/S_q)^2 n_q}{\sum_{q \in P_i} w(||p_i - q|| n_q||)(S_p/S_q)^2} \tag{11}$$

$$p_{i+1} = p_i + n_i \left((p^*_{i+1} - p_i) n_i\right) \tag{12}$$

$p_0 = p$ is an unoptimized primary point, P_i contains all unmodified points that

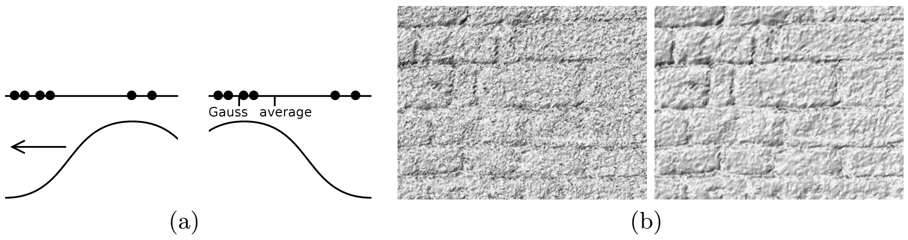


Fig. 10. (a): Local outlier avoidance with iteration (b) Left: Unoptimized primary points. Right: Optimized primary points

are in the radius of $2I_{p_i}$ and have a scale that is smaller than 1.6 times the scale of p. According to our simulations, $(S_p/S_q)^2$ is the optimal weighting function for all medium free noise functions. Of course, this is not fulfilled between depth maps of different scales, because rough depth maps can not have information about fine details. That is why we don't include inaccurate points to P_i. n_i is the local normal. p_i is only allowed to move along the normal and not along the tangent direction for p_{i+1}. p_i is optimized to p_{i+1} until the process fails or converges. If it fails because it does not terminate after 20 iterations, $|P_i| < 3$, or the point moves out of its original influence radius the point is discarded. The process converges if:

$$||p_n - p_{n+1}|| < min(\epsilon, ||p_n - p_{n-1}||) \ , \tag{13}$$

with a small $\epsilon > 0$. As weighting function we use

$$w(x) = \frac{1}{|c(x/I_p)^3| + 1} \ . \tag{14}$$

We use this function to improve the effect in Figure 10(a) to pull local outlier points onto the surface. The often used Gauss function falls outwards too fast to take much advantage of this effect. To speed up the optimization process we start two iterations for a point. One with $c = 5$ and one with $c = 30$. Local outliers can slide between other points on the surface, so, we discard primary points if there are better primary points in 0.8x their influence radius after optimization. Finally, we use Formula 10 with the color c_q instead of q to improve the color of an optimized point. In Figure 10(b) the difference between unoptimized and optimized primary points can be seen. Details are kept while noise is removed.

However, noise removal is only possible if there are enough samples with similar scale in the influence radius. Otherwise, we must increase I before point selection and decouple it from S to avoid noisy surfaces. We estimate the noise N as

$$1/N = \sum_{q \in P_i} w(||p_i - q||)(I_p/I_q)^2 \ . \tag{15}$$

I is increased as long as $1/N < Z \cap I < 3S$. We set Z to 2 and c in w to 10. Practically, it depends on the quality of the depth maps and, hence, the datasets.

8 Meshing

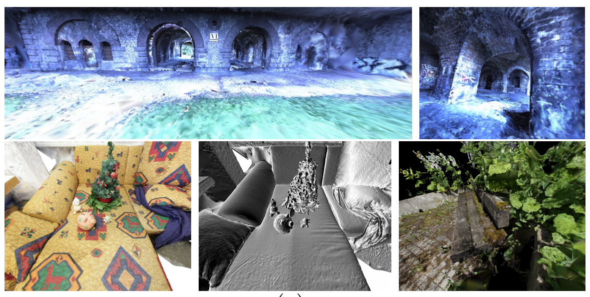
For meshing of our optimized primary points we use the Delaunay graph cut of Labatut et al. [27], because it is robust to changes in point density and considers visibility, which helps to reconstruct difficult surface parts. In their work they also showed that their approach is very robust against outliers. We got similar results with random noise and dense surface point clouds. However, as can be seen in Figure 8(b) outlier avoidance in real datasets can be more difficult.

For their visibility constraint we consider not only the reference image, from which that point was created, but we also add the reference image of each secondary point to the nearest optimized primary point. Additionally, we bind their σ constant to the scale of the primary point, to handle ray accuracy depending on the point accuracy and not the ray length:

$$\sigma_p = \sigma_{const} S_p \ . \tag{16}$$

9 Results

We tested our approach on several datasets. Results can be seen in Figure 1, 11 and 12. Our memory peak in the Ulm Minster dataset was mainly because of the


(a)

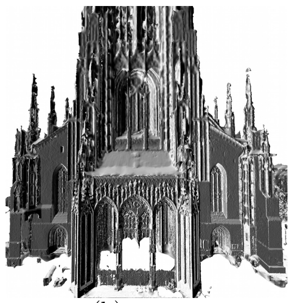

(b)

Fig. 11. a) The upper row shows reconstruction from a Ladybug video dataset. Through short Ladybug videos it is possible to reconstruct huge scenes with our approach. The lower row shows our sofa and garden dataset. Thanks to point optimization, the sofa dataset is very smooth but anyhow accurate at locations where many depth maps overlap. b) Ulm Minster surface.

Delaunay graph cut 12.8 GB for about 18 Million triangles and 55 Million points for point selection and optimization. Point processing took single core about 20 minutes and meshing 1 hour. Compared to [18,19] our approach is clearly faster and needs clearly less memory, as far as it is comparable on different datasets. Furthermore, it is probably also more accurate as we perform careful continuous optimizations instead of just using voxels of different scales. Depth map creation took in average 35 seconds, out of 9 images, for the 333 1863x1236 pixel depth maps of the Ulm Minster dataset. This is also very fast compared to run times of region growing based approaches in [3]. Multi depth map filtering took 192 seconds, altogether. Most time was needed for loading depth maps from the harddisk. We got 6.8% more 3D points and 5.3% more mesh points with our covering weighting on the garden dataset, than without. On the Ulm Minster dataset with less occlusions it were 3.4/1.8%. Figure 12(a) shows some differences because of the weighting.

10 Conclusion

We presented a scale robust Multi View Stereo approach that can process arbitrary image datasets. Our core contribution for scale robustness is surely the point selection and optimization method that delivers refined point clouds that can easily be processed by Delaunay based meshing methods. Nevertheless, our image selection method is also important for high quality depth maps and, thanks to image selection, the depth map based outlier filter is a powerful and fast method to remove outliers.

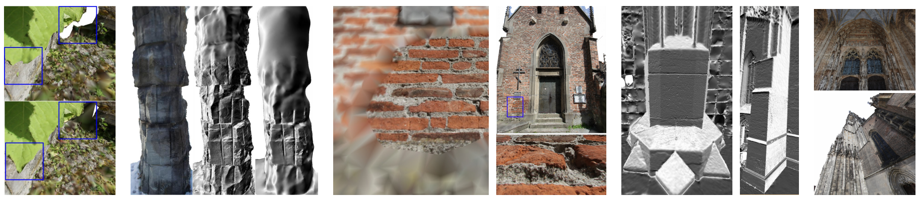


Fig. 12. a) Some differences with (bottom) and without (top) covering weighting on the garden dataset. b) Columns. The right most reconstruction was strongly reduced to demonstrate the high resolution. c) Some parts of the chapel were photographed in especially high resolution compared to the other parts. Thus, there is a strong resolution discontinuity at the border of the close reconstruction, that is clearly visible in the left image. The left image was like the column in (b) size reduced. The full reconstruction can be seen on the right. d) and e) Further results for the Ulm Minster dataset. The left column in d) has only pixel size in Figure 11 b).

Acknowledgments

This work has been partially funded by the DFG Emmy Noether fellowship (Le 1341/1-1) and by an NVIDIA Professor Partnership Award.

References

1. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring photo collections in 3d. In: SIGGRAPH Conference Proceedings, New York, NY, USA, ACM Press (2006) 835–846 `http://phototour.cs.washington.edu/bundler/`.
2. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building rome in a day. In: Computer Vision, 2009 IEEE 12th International Conference on, Ieee (2009) 72–79
3. Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Volume 1., IEEE (2006) 519–528 `http://vision.middlebury.edu/mview/`.
4. Pons, J., Keriven, R., Faugeras, O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. International Journal of Computer Vision **72** (2007) 179–193
5. Vogiatzis, G., Hernández Esteban, C., Torr, P.H.S., Cipolla, R.: Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. IEEE Trans. Pattern Anal. Mach. Intell. **29** (2007) 2241–2246
6. Kolev, K., Pock, T., Cremers, D.: Anisotropic minimal surfaces integrating photo-consistency and normal information for multiview stereo. Computer Vision–ECCV 2010 (2010) 538–551
7. Sinha, S., Pollefeys, M.: Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. (2005)
8. Furukawa, Y., Ponce, J.: Carved visual hulls for image-based modeling. Computer Vision–ECCV 2006 (2006) 564–577

9. Song, P., Wu, X., Wang, M.: Volumetric stereo and silhouette fusion for image-based modeling. The Visual Computer **26** (2010) 1435–1450
10. Gallup, D., Frahm, J., Mordohai, P., Yang, Q., Pollefeys, M.: Real-time plane-sweeping stereo with multiple sweeping directions. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE (2007) 1–8
11. Campbell, N., Vogiatzis, G., Hernández, C., Cipolla, R.: Using multiple hypotheses to improve depth-maps for multiview stereo. ECCV08 (2008) 766–779
12. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE transactions on pattern analysis and machine intelligence (2009) 1362–1376
13. Hiep, V., Keriven, R., Labatut, P., Pons, J.: Towards high-resolution large-scale multi-view stereo. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 1430–1437
14. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S.: Multi-view stereo for community photo collections. In: Proc. ICCV, Citeseer (2007) 1–8
15. Pollefeys, M., Nistér, D., Frahm, J., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S., Merrell, P., et al.: Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision **78** (2008) 143–167
16. Frahm, J., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y., Dunn, E., Clipp, B., Lazebnik, S., et al.: Building rome on a cloudless day. Computer Vision–ECCV 2010 (2010) 368–381
17. Jancosek, M., Shekhovtsov, A., Pajdla, T.: Scalable multi-view stereo. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, IEEE (2009) 1526–1533
18. Mücke, P., Klowsky, R., Goesele, M.: Surface reconstruction from multi-resolution sample points. In: Proceedings of Vision, Modeling, and Visualization. (2011) 105–112
19. Fuhrmann, S., Goesele, M.: Fusion of depth maps with multiple scales. In: ACM Transactions on Graphics (TOG). Volume 30., ACM (2011) 148
20. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Towards internet-scale multi-view stereo. In: CVPR 2010, IEEE (2010) 1434–1441
21. Rumpler, M., Irschara, A., Bischof, H.: Multi-view stereo: Redundancy benefits for 3d reconstruction. In: Proceedings of 35th AAPR/OAGM 2011. (2011)
22. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: Patchmatch: A randomized correspondence algorithm for structural image editing. In: ACM Transactions on Graphics (TOG). Volume 28., ACM (2009) 24
23. Strecha, C., Von Hansen, W., Van Gool, L., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. (2008) http://cvlab.epfl.ch/~strecha/multiview/denseMVS.html.
24. Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide-baseline stereo. IEEE transactions on pattern analysis and machine intelligence (2009) 815–830
25. Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J., Yang, R., Nistér, D., Pollefeys, M.: Real-time visibility-based fusion of depth maps. In: Computer Vision, 2007. ICCV 2007. IEEE 11th Int. Conf. on, IEEE (2007) 1–8
26. Cuccuru, G., Gobbetti, E., Marton, F., Pajarola, R., Pintus, R.: Fast low-memory streaming mls reconstruction of point-sampled surfaces. In: Proceedings of Graphics Interface 2009, Canadian Information Processing Society (2009) 15–22
27. Labatut, P., Pons, J., Keriven, R.: Robust and efficient surface reconstruction from range data. In: Computer Graphics Forum. Volume 28., Wiley Online Library (2009) 2275–2290