Camera-Clustering for Multi-Resolution 3–D Surface Reconstruction

Andrei Zaharescu[1], Cedric Cagniart[2], Slobodan Ilic[2], Edmond Boyer[1], Radu Horaud[1]

[1] Jean Kuntzmann Laboratories, Grenoble & INRIA Rhone-Alpes
[2] Technical University of Berlin & Deutsche-Telekom Laboratories

Abstract. In this paper we propose a framework for piecewise mesh-based 3D reconstruction from a set of calibrated images. Most of the existing approaches consider all available images at once. However, this is not tractable with very large sets of cameras. Therefore, we use subsets of images and evolve parts of the surface corresponding to those images. Our main contribution is an approach to partition the camera images, either semi-automatic, through clustering, or user guided, via a geometric modeling interface. The sub-parts of the surface corresponding to camera subsets are independently evolved at multiple mesh resolutions. This allows to handle large scenes and to increase the mesh resolution in surface parts containing high levels of detail at reduced memory and computational costs. We demonstrate the versatility of our approach on different data sets and with different camera layouts. Finally, comparing the piecewise and global reconstructions with groundtruth, we find no significant loss in the overall reconstruction quality.

1 Introduction

Recent advances in multi-view 3D reconstruction from a set of calibrated cameras produced impressive results. The visual and measured quality is getting comparable to that of the laser scans. An issue of interest that naturally arises in this field is how to efficiently deal with scenarios where there are lot of images and, due to memory requirements, they cannot all be processed at the same time. In order to reduce the volume of image data we need to access simultaneously, we use subsets of the original image set and evolve the parts of the surface corresponding to those images by maximizing photo-consistency. The main contribution of our method is an approach to partitioning of the camera images which can be either semi-automatic, through clustering, or user guided, via a geometric modeling interface. The sub-parts of the surface corresponding to the camera subsets are independently evolved at multiple mesh resolutions. This allows for an increase of the mesh resolution in surface parts containing high level of detail at reasonable memory and computational costs.

The problem of content-aware camera clustering and reconstruction by parts did not receive considerable attention in the past. Simon *et al.* [1] address an orthogonal problem to ours: scene summarization. In their scenario, they have a lot of images covering a scene and they are interested in the canonical views that can best describe it. They choose a representative exemplar from within each camera cluster, which is computed using visibility information for SIFT matches. There exist a a number of 3-D reconstruction methods [2–4] that can deal with large number of images, thus overcoming the apparent need for such a reconstruction by parts. As we shall see, they implicitly define heuristic camera clusters and they could benefit from the currently proposed algorithm. [4] casts the problem in a tracking framework and thus uses a temporal prior (sliding window). [2, 3] compute a set of sparse 3-D points from image correspondences, which are later on used to infer the full geometry. In order to reduce the search space for a given image/camera, the other image/camera is selected among the ones sharing the same viewing direction and rotation orientation. All these methods can benefit from our camera clustering method for special cases: revisiting the same sub-scene for [4]; camera panning scenarios for [2, 3].

We will provide a short review of the 3-D reconstruction methods. They can be categorized in *Dense multi-view stereo* algorithms , *Graph Cut* approaches and *Variational* methods. We will motivate the particular choice of the reconstruction algorithm, keeping in mind that the proposed camera clustering framework is very general and can thus be used in combination with any 3-D reconstruction method.

Dense multi-view stereo algorithms [3, 5, 6, 4] incrementally build up a point cloud of the environment during the reconstruction process. From such reconstructions, it is possible to build mesh representations using information such as points [7], or oriented points with normals [3] by triangulating them using available algorithms [8]. Such a reconstruction is constrained by the quality of the reconstructed data point clouds, which are in general noisy and contain outliers difficult to remove from the final mesh.

Graph-cut approaches [9] look for the closed surface maximizing the photometric consistency between the interior -source- and the exterior -sink- of an object over a regular grid. Recent advances [10] allow for an adaptive multi-resolution of the graph by using a tetrahedral volumetric mesh representation.

Variational methods can adopt either an implicit surface (Eulerian) representation [11–13] or a mesh-based (Lagrangian) [14–18] point of view. They look for a surface which minimizes a global photo-consistency error function. The level-set implicit representation [13] requires dense regular sampling on a grid of the initial bounding volume, thus fixing the mesh resolution to the cell grid size. One advantage of such representations is the straightforward handling of topology changes at the cost of increased memory requirements. Evolving meshes directly calls for more elaborate schemes to handle topology changes and self-intersections, but offers a much more compact representation and can have an adaptive resolution compared to the implicit representations. Due to the smoothing energy terms, they tend to offer better resistance to outliers than dense multi-stereo approaches. Recent advances in mesh-based methods [17] provide a solution to these problems and will be used in our method. As opposed to the other Lagrangian methods, it does not constrain meshes to a fixed resolution and it allows for faces of all sizes. This approach to mesh evolution, coupled with multi-resolution strategy on the surface parts, efficiently recovers objects of different complexity with the targeted precision on the more detailed surface parts.

We tested our approach on different data sets including single-compact objects, outdoor architectural sites filmed in high resolution, and a long synthetic sequence. Finally, we analysed quantitatively our results and compared our piecewise and global reconstructions to the laser scans, showing very little loss in the overall reconstruction quality.

In the reminder of the paper we will describe our method, show the results of the experimental evaluation and finally conclude, talking about future work.

2 Method

Our objective is to evolve the complete surface by parts. This is an important aspect, if we want to reduce memory costs and computational time imposed when using all images at once. We rely on the recent mesh-based evolution method of Zaharescu et al. [17], which efficiently handles mesh topological changes and allows meshes with variable facet sizes. The mesh is evolved in parts over time by minimizing the photo-consistency error function proposed by Pons et al. [13]. For more details, consults [17], [13]. The mesh parts to be evolved are defined according to the partitioning algorithm discussed below. The camera clustering method that will be presented can work in combination with any 3-D reconstruction algorithm.

Camera Clustering. In general, if the positioning of the cameras is arbitrary and the rough initial geometry is known we can cluster original camera set C into a given number k of camera subsets $C_m, m = 1..k$. To do this, we first recover the geometry of the object/scene at a coarse resolution from down-sampled images using all cameras $c_i, i = 1..N_c$. For each camera c_i, we name S_i the set containing all the vertices from the scene set S which are visible.

We then define an intuitive distance function between two cameras c_p and c_q as the cardinal of the symmetric difference of S_p and S_q:

$$\begin{aligned} d_S(c_p, c_q) &= |S_p \bigtriangleup S_q| && (1)\\ &= |(S_p \cup S_q) \setminus (S_p \cap S_q)| && (2) \end{aligned}$$

In practice we use OpenGL depth maps [19] to evaluate the visibility and accumulate the information into the $N_v \times N_c$ visibility matrix defined as:

$$\Delta = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,N_c} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,N_c} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N_v,1} & \beta_{N_v,2} & \cdots & \beta_{N_v,N_c} \end{bmatrix}$$

where $\beta_{i,j}$ is 1 or 0 depending on whether the j^{th} vertex is visible in the camera c_i.

Instead of using a coarse mesh, one could also use potential SIFT matches in the image and accumulate them in the visibility matrix Δ, as it has been proposed by [1].

Camera-based clustering consists of performing k-means clustering [20] on the columns of Δ, where k represents the number of desired camera sub-sets. Each of these columns represent one camera, encapsulating visibility information for all the mesh vertices. Using these binary vectors, computing the distance function we defined in (2) is equivalent to computing the sum of squared differences :

$$d_S(c_p, c_q) = ||\Delta(:, p) - \Delta(:, q)||_E^2 \qquad (3)$$

Note that each 3-D surface point has its contribution in the distance function, based on whether it is visible in both cameras. This simple formulation takes into account the geometry of the object and the layout of the cameras implicitly, by using the visibility information. We present some of the clustering results in Figure 1. Please note how the clustering correctly delineates the parts of the objects that share less visibility information (the two sides of the dinosaur, or the facets of the temple).

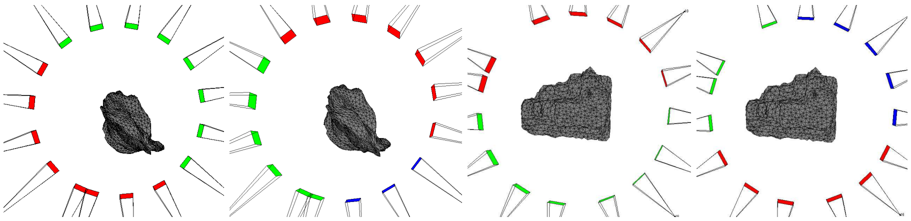


Fig. 1. Illustration of *camera-based camera clustering* using two data sets, dino (first 2 images) and temple (next 2 images), with two and three clusters.

Geometry-based clustering Alternatively, one could address the dual problem and perform clustering on the rows of the matrix Δ, thus on the geometry of the scene. [3] Once the vertex clusters have been obtained, the set of the most discriminant cameras for each cluster has to be selected. This is done in practice by a voting method, imposing a minimum camera voting threshold of α times the average score among the camera with positive votes within each cluster. Using this dual formulation implies some tuning the α parameter, but has the great advantage of allowing potential camera overlaps, meaning that the same camera might be used by different vertex clusters. We present some of the clustering results in Figure 2. We have chosen $\alpha = 0.90$ in the dino case and $\alpha = 0.70$ in the temple case.

Part-Based Surface Reconstruction. For each of the obtained clusters we run the algorithm described in [17], allowing only the vertices visible in the current camera cluster to evolve. In practice, we impose a minimum vertex visibility threshold γ. In order to avoid the issues related to merging partial reconstructions, we run one camera cluster at a time. The output of algorithm for one cluster is used as the input for the subsequent cluster. However, this approach comes at the expense of being unable to parallelize the approach in the current formulation. Alternatively, we could use algorithms such as [8] to merge the reconstructions and process all the clusters in parallel.

3 Results

We demonstrate the possibility of manually selecting vertices in a geometric modeling interface, in order to recover the surface regions of interest in high resolution. We

[3] The normalized point coordinates and the normal information can be added to the Δ matrix in order to take more geometric information into account.

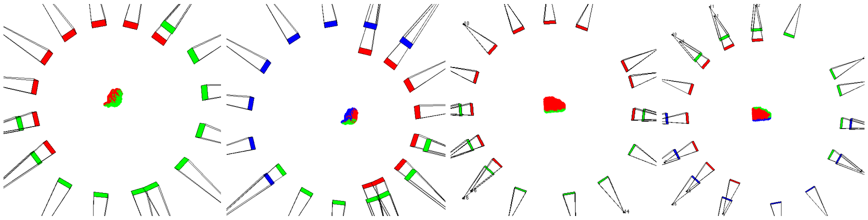
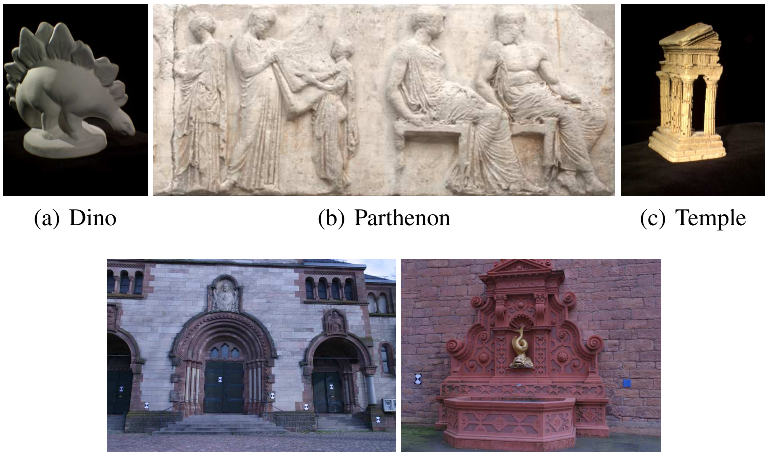


Fig. 2. Illustration of *geometry-based camera clustering* using two data sets, dino (first 2 images) and temple (next 2 images), with two and three clusters.

also present the results of 3D reconstructions using our camera partitioning method. To demonstrate the versatility of our approach, we use different data sets, shown in Figure 3. When minimizing each subset, only the visible parts of the mesh are being sub-divided and minimized, while the others are blocked. We impose a minimum vertex visibility of $\gamma = 3$ in all cases.



(a) Dino (b) Parthenon (c) Temple

(d) Herz-Jesu (e) Fountain

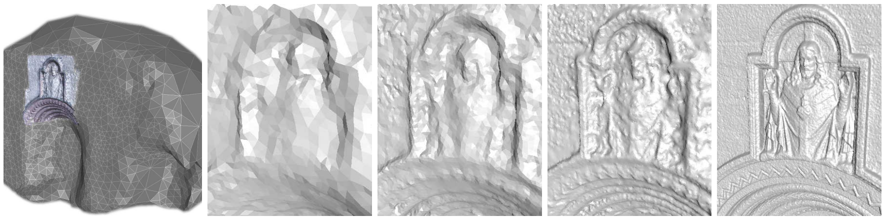
Fig. 3. Original images from the datasets used in our experiments.

3.1 User Guided Multi Resolution Scenario

The *User Guided Multi Resolution results* are presented on the ”Herz Jesu” and the ”Fountain” sequences of [21]. These experiments make use of user-defined regions of interest to increase the mesh resolution.

These sequences illustrate the interest of evolving directly a mesh representation of the reconstructed geometry when assisting a 3D artist in the task of visual modeling. The user can manually select a region of interest by selecting the corresponding vertices in the current approximation of the geometry and then ask the system for a further improvement of this part of the mesh. The higher resolution part can then be automatically evolved to maximize the photo-consistency accross the input image set. Virtual cameras are thus generated, representing the relevant input image sub-parts. In practice, cropping an image at coordinates (x_1, y_1, x_2, y_2) modifies the associated camera projection matrix by translating optical center by $(x_1 y_1)$.

The Herz Jesu Sequence consists of 8 high resolution (3072 x 2048 pixels) pictures. The general view of the coarse reconstruction can be found in Figure 4(a). The scene was very interesting in the validation of our algorithm, because it involved different parts which had very different levels of detail. The wall can be represented by a coarser resolution mesh, whereas the door and the sculpted representation of Jesus above it are regions of interest that can benefit from a higher resolution reconstruction. The sculpture, in particular, is a region that a user might want to recover, but would not be able to quickly model it using simple geometric primitives. Our method allows to rapidly select the corresponding vertices and to let the algorithm maximize the photo-consistency. In addition, we have also obtained from the author groundtruth data for a part of the reconstruction, which was acquired via laser-scanning. The error measurements are presented in Table 1.



(a) 3-D view (b) level 1 (c) level 2 (d) level 3 (e) groundtruth

Fig. 4. The Herz-Jesu sequence

The Fountain Sequence consists of 11 pictures of size 3072 x 2048 pixels, taking up 64.4 Mb in compressed format. It involves very fine 3D details and was therefore a good stress test. We pushed the algorithm to a very high resolution of 4 pixels per triangle. We ran two tests on this dataset. The first test was to reconstruct the whole fountain at a high level of detail, leaving only the wall behind in a coarser state. The algorithm needed 894 minutes to finish. We then ran the algorithm on the fish sculpture only and got a result after 83 minutes. This validates our approach in the sense that evolving a subpart of the reconstructed geometry independently from the rest allowed us to stay away from swapping and other memory problems. In both cases, we started the algorithm from

Level	Avg. DistanceError	Completeness (0.05m)	Avg.Edge Size	Avg.Edge Size	No. Triang.
Level 1	0.0270m	83.36%	0.1347m	21.04 pixels	2,136
Level 2	0.0177m	92.90%	0.0703m	10.98 pixels	8,592
Level 3	0.0164m	94.17%	0.0232m	3.62 pixels	82,490
Groundtruth	0.0000m	100.00%	0.0064m	1 pixel	1,693,914

Table 1. Information about Herz Jesu reconstructions. The errors are measured in meters. The completeness is measured with respect to a threshold of 0.05m.1 pixel corresponds to an edge size of 0.0064m.

a coarser reconstruction that was performed with all images at half the original image size. The results are presented also in Table 2.

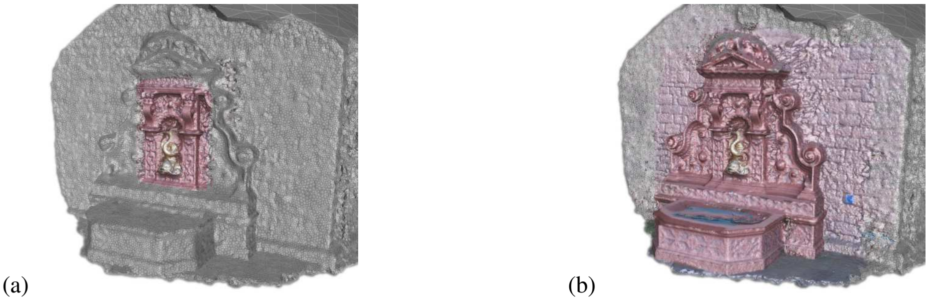


Fig. 5. The Fountain sequence. The parts shown in color are reconstructed in higher resolution. (a) Further minimization on the fish; (b) Further minimization on the whole fountain.

Level	Img. Input Size	Avg.Edge	Avg.Edge	No. Triang.	Time
Coarse	64.4Mb	0.0750 m	20.83 pixels	75,904	129 mins.
Close-up Fish	5.4Mb	0.0161 m	4.44 pixels	151,564	129 + 83 = 212 mins.
Close-up Fountain	40.6Mb	0.0160 m	4.44 pixels	660,540	129+894 = 1,023 mins.

Table 2. Information about Fountain reconstructions. 1 pixel corresponds to an edge size of 0.0036 m. The image input size value represents the total compressed size of the input images, which can be further sub-sampled, depending upon the resolution used.

3.2 Camera Partitioning

The camera partitioning algorithm is presented on two very different types of sequences. We first validate the method on a typical turntable situations, where the object bounding volume projects inside all the images of the sequence. The Dino and Temple datasets are presented. We then present the Parthenon dataset, which involves one long image sequence covering a large object. In this case, each image only contains a small portion of the reconstructed geometry. We have used the camera-based clustering in all results shown below. The point-based camera clustering leads to very similar results. Due to the inherent overlap between views, we decided to use the simplest method.

Dino and Temple Sequence. These sequences were obtained from the Middleburry Multi-View Stereo dataset [22]. It consists of 47 images of size 640x480. The coarse surfaces were evolved from the visual hull using all down-sampled images at 320x240 resolution. The reconstruction results for two clusters are shown in Figure 6 and Table 3 (see [4] for more). Our proposed method does not lose significant accuracy with respect to the original method [17] (which uses all the cameras), while reducing the memory requirements in half and maintaining comparable time processing times.

Dataset	Temple Ring				Dino Ring			
Paper	Acc.	Compl.	Mem.	Time	Acc.	Compl.	Mem.	Time
Zaharescu et al [17]	0.55mm	99.2%	1031MB	60min	0.42mm	98.6%	962MB	43min
Our method - cluster 1	0.62mm	98.5%	468MB	36 min	0.5mm	98.5%	483MB	33min
Our method - cluster 2			472MB	42 min			476MB	35min

Table 3. Middleburry 3-D Rec. Results. Accuracy: the distance d in mm that brings 90% of the result R within the ground-truth surface G. Completeness: the percentage of G that lies within 1.25mm of R. Memory: the amount or RAM used by the program. Time: the duration for the program to finish.

The parthenon sequence consists of 200 images of size 640x480. Each of these cameras covered only about 1/10th of the overall structure. This sequence is synthetic and was generated using Blender[5] and the textured models obtained from the Parthenon scuplture gallery website[6]. We have employed various camera cluster sizes, with $k = 2, 4, 8, 20$. The camera path can be observed in Figure 7 where we also show the camera clusters in different colors. The original surface was a parallelepiped of 4472 facets. In Figure 8 we show reconstruction results throughout the evolutions of different clusters.

We measured the reconstruction precision with respect to the groundtruth as shown in Table 4. As it can be observed, there is negligible loss in precision of 5mm when performing subset-based reconstruction versus when using all the cameras at the same time. One has to bear in mind that the laser error for the given distance is also around 5mm.

Virtual Fountain Sequence In the fountain sequence , since we are dealing with very high resolution images (3072 x 2048), we have generated virtual cameras such that the

[4] http://vision.middlebury.edu/mview/eval/

[5] http://www.blender.org/

[6] http://projects.ict.usc.edu/graphics/parthenongallery/index.html

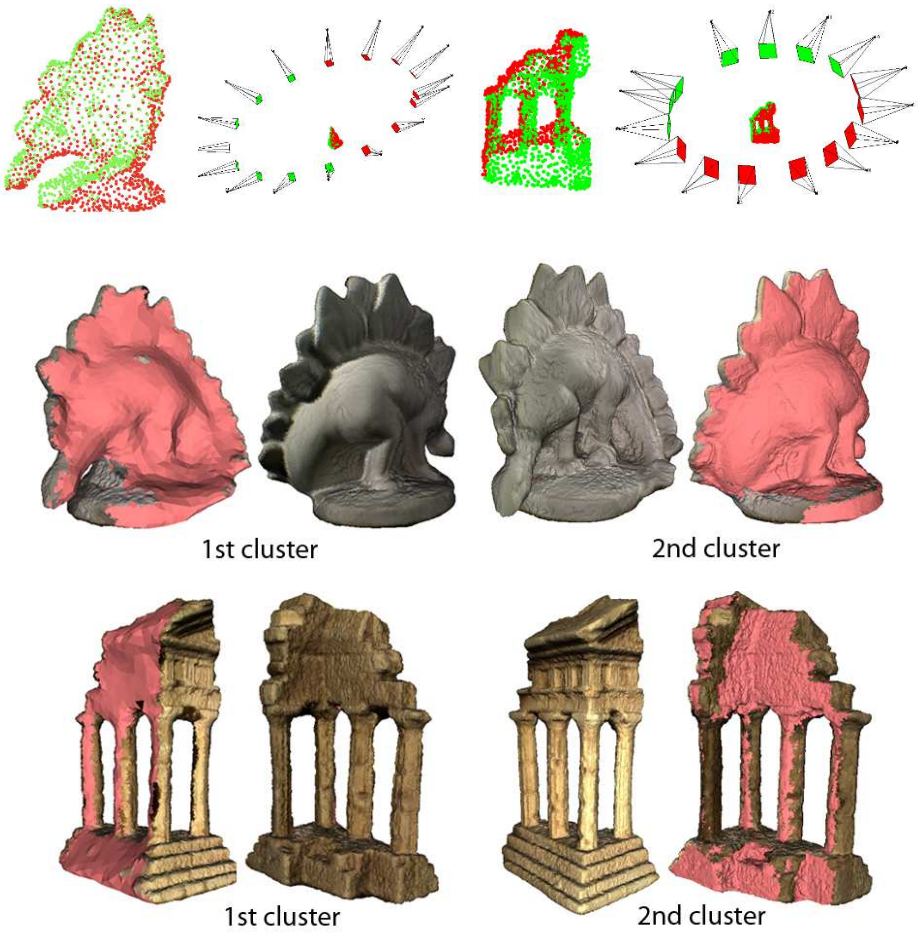


Fig. 6. Dino and Temple Sequence reconstruction results. Top row. Camera partitioning. Middle row. Partial reconstructions of the ”dino” using two clusters. Bottom row: Partial reconstruction of the ”temple” using two clusters. The invisible vertices within each cluster are coloured in light red. The reconstructions are made at 5 pixels per edge size.

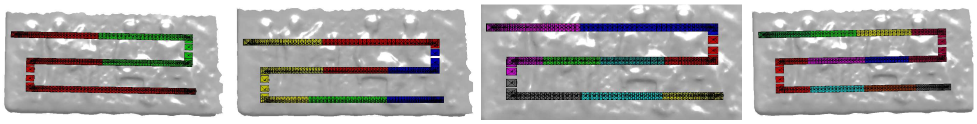


Fig. 7. Different camera clustering for Parthenon sequence with the number of clusters being $k = 2, 4, 8, 10$. Cameras belonging to the same cluster are colored in the same color.

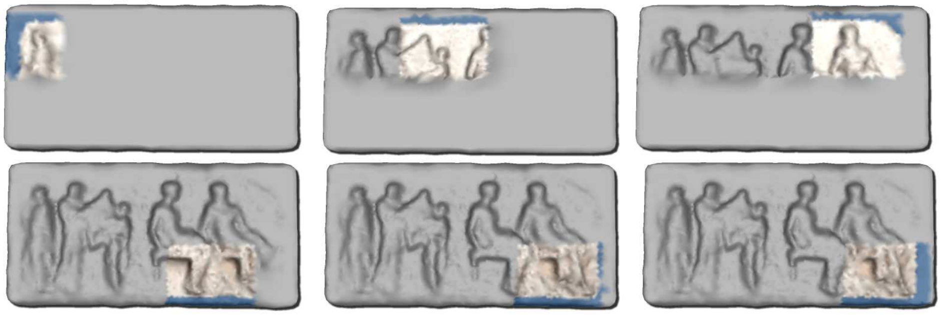


Fig. 8. Parthenon reconstruction using 200 cameras and 20 clusters of cameras. We show partial reconstructions where parts of the surface are reconstructed using cameras belonging to one cluster at a time.

Level	Avg. Dist. Err.	Completeness (0.05m)	Avg.Edge (m)	Avg.Edge (pixels)
20 Clusters - Low Res.	0.0344 m	76.98%	0.2528 m	16.20 pixels
All Cameras - Low Res.	0.0265 m	84.83%	0.2885 m	18.49 pixels
2 Clusters - High Res.	0.02006 m	92.50%	0.1629 m	10.44 pixels
4 Clusters - High Res.	0.0209 m	91.59%	0.1641 m	10.52 pixels
8 Clusters - High Res.	0.0201 m	92.35%	0.1599 m	10.25 pixels
20 Clusters - High Res.	0.0205 m	92.36%	0.1331 m	8.53 pixels
All Cameras - High Res.	0.0153 m	95.73%	0.1102 m	7.20 pixels
Groundtruth	0.0000m	100.00%	0.0841m	5.39 pixels

Table 4. The Parthenon reconstructions error measures, compared to the laser scan ground truth. The errors are measured in meters. The completeness is measured with respect to a threshold of 0.05m. 1 pixel corresponds to an edge size of 0.0156m. Note there is negligible loss in precision when performing subset-based reconstruction versus when using all the cameras at the same time.

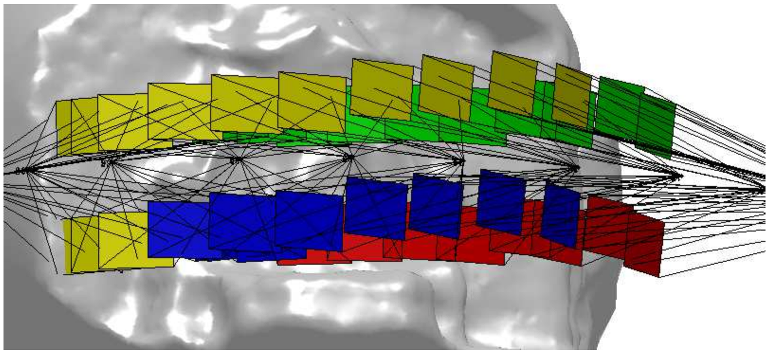


Fig. 9. 4 Cluster View for the Virtual Fountain dataset.

original image is cropped into a 2x2 grid (hence 4 virtual cameras for each real camera). We are pleased to report that, performing camera-subset clustering, the 4 correct subsets were found. Results can be observed in Figure 9. The reconstruction results per cluster are presented in Figure 10.

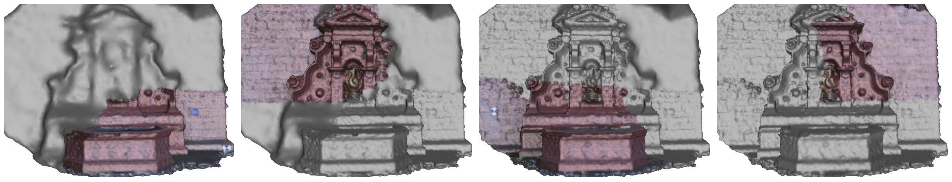


Fig. 10. Results for the partial reconstructions in the virtual fountain scene.

Future Work. One other possible scenario that we plan on investigating is, instead of pre-generating virtual cameras and performing clustering, to generate the virtual cameras post-clustering, limiting the virtual cameras to the bounding boxes. Also, we plan on exploring automatic mesh segmentation methods that take into account more mesh properties, which will in turn allow for the selection/generation of the proper cameras. Finally, we plan on integrating a photo-consistency based threshold for adaptive mesh resolution. It would adaptively determine if a facets represents the geometry well enough, based on the reprojection error measure. It was not currently implemented due to time constraints and do to the fact that in practice we calculate only the derivative of the photo-consistency measure, not the measure itself.

4 Conclusion

In this paper we addressed the problem of piecewise 3D surface reconstruction from multiple calibrated cameras. We showed that, starting from the coarse initial geometry, the original set of cameras can be partitioned into a number of camera subsets, each of which is observing a part of the surface to reconstruct. Independent reconstructions of surface parts require less memory than when using all cameras as in global approaches. We also showed the possibility of using these techniques in a graphical modeling interface, when regions of interest have to be reconstructed in high resolutions. We have demonstrated that the proposed method does not lose significant accuracy with respect to global methods, while offering several advantages with respect to the time and to the memory requirements.

References

1. Simon, I., Snavely, N., Seitz, S.: Scene summarization for online image collections. In: Proceedings of International Conference on Computer Vision. (2007) 1–8

2. Habbecke, M., Kobbelt, L.: A surface-growing approach to multi-view stereo reconstruction. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (2007)
3. Furukawa, Y., Ponce, J.: Accurate, dense and robust multi-view stereopsis. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (2007)
4. Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J.M., Yang, R., Nister, D., Pollefeys, M.: Real-time visibility-based fusion of depth maps. In: Proceedings of International Conference on Computer Vision. (2007) 1–8
5. Pollefeys, M., Gool, L.V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. International Journal of Computer Vision **3** (2004) 207–232
6. Fua, P.: From multiple stereo views to multiple 3-d surfaces. International Journal of Computer Vision **24** (1997) 19–35
7. Labatut, P., Pons, J.P., Keriven, R.: Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In: Proceedings of International Conference on Computer Vision. (2007)
8. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Symposium on Geometry Processing. (2006) 61–70
9. Vogiatzis, G., Torr, P., Cipolla, R.: Multi-view stereo via volumetric graph-cuts. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (2005)
10. Sinha, S.N., Mordohai, P., Pollefeys, M.: Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (2007)
11. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer (2003)
12. Duan, Y., Yang, L., Qin, H., Samara, D.: Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. In: Proceedings of European Conference on Computer Vision. Volume 3. (2004) 238–251
13. Pons, J.P., 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
14. McInerney, T., Terzopoulos, D.: T-snakes: Topology adaptive snakes. Medical Image Analysis **4** (2000) 73–91
15. Lachaud, J.O., Taton, B.: Deformable model with adaptive mesh and automated topology changes. In: Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling. (2003)
16. Jung, W., Shin, H., Choi, B.K.: Self-intersection removal in triangular mesh offsetting. Computer-Aided Design and Applications **1** (2004) 477–484
17. Zaharescu, A., Boyer, E., Horaud, R.P.: Transformesh: a topology-adaptive mesh-based approach to surface evolution. In: Proceedings of Asian Conference on Computer Vision. Volume II of LNCS 4844. (2007) 166–175
18. Pons, J.P., Boissonnat, J.D.: Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (2007) 1–8
19. Schreiner, D., Woo, M., Neider, J., Davis, T.: OpenGL Programming Guide. 5 edn. Addison-Wesley (2006)
20. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
21. Strecha, C., von Hansen, W., Gool, L.V., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (2008)
22. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Volume 1. (2006) 519–526