Spectral Camera Clustering

Alexander Ladikos Slobodan Ilic Nassir Navab

Chair for Computer Aided Medical Procedures (CAMP)
Technische Universität München, Germany

Abstract

We propose an algorithm for clustering large sets of images of a scene into smaller subsets covering different parts of the scene suitable for 3D reconstruction. Unlike the canonical view selection of [13], we do not focus only on the visibility information, but introduce an alternative similarity measure which takes into account the relative camera orientations and their distance from the scene. This allows us to formalize the clustering problem as a graph partitioning and solve it using spectral clustering. The obtained image clusters bring down the amount of data that has to be considered by the reconstruction algorithms simultaneously, thereby allowing traditional algorithms to take advantage of large multi-view data sets processing them significantly faster and at smaller memory costs compared to using the full image datasets. We tested our approach on a number of multi-view data sets and demonstrated that the clustering we obtain is suitable for 3D reconstruction and coincides with what a human observer would consider as a good clustering.

1. Introduction

The availability of high quality digital cameras at reasonable prices allows an easy acquisition of large numbers of high-quality images. This motivated Computer Vision researchers to intensify their research on 3D reconstruction problems, proposing new solutions which exploit those rich sources of information. The recent advances in static 3D reconstructions from calibrated cameras exploit the spatial redundancy among the images and produce high-quality reconstruction results [10, 5, 9, 6, 7]. This usually requires using all available images of the objects of interest at once. This is reasonable when a limited number of images is used, but becomes prohibitive when the number of available images is large and they are high-resolution. This is especially the case with large scenes such as those of Figure 4. For this reason breaking the collection of images into meaningful clusters, ideally covering different parts of the object, has to be addressed. The community has already addressed the problem of handling huge collections of images available on the internet [15]. This required addressing several problems such as structure from motion [16], summarizing images by finding canonical views [13] or finding

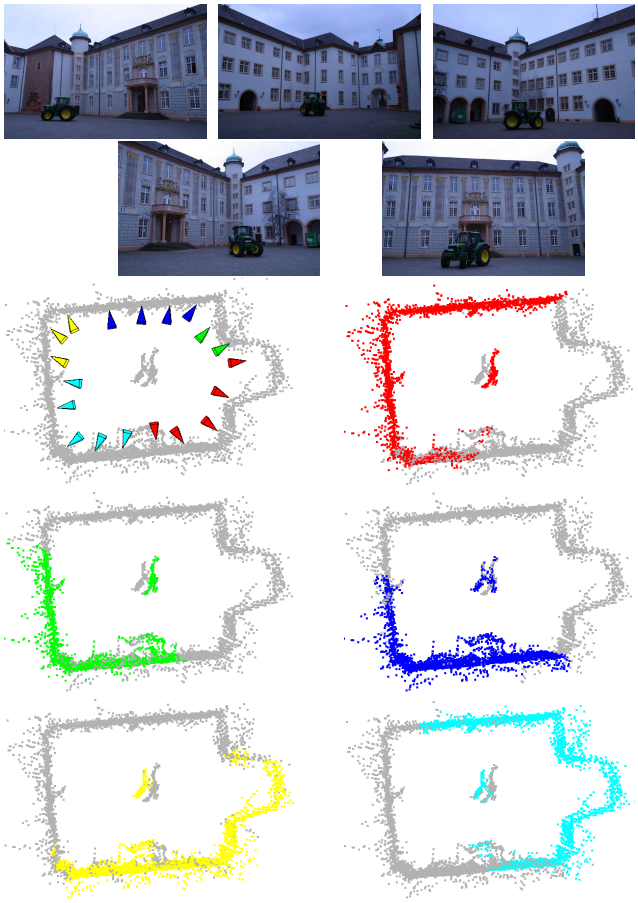


Figure 1. Clustering on the Castle sequence. The first two rows contain images showing representative views for each cluster. In the first image of the third row the camera clusters found by our algorithm are depicted. The following images show the object points visible in each cluster.

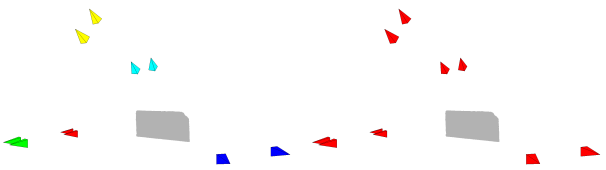
paths through the photos and turning them into controls for image based rendering [14]. Unlike these approaches, and similar to [19], our objective is to cluster a collection of images of a particular object into smaller image subsets suitable for 3D reconstruction as shown in Figure 1. In contrast to [13] and [19], we do not focus only on the visibility of the scene in the cameras, but introduce an alternative similarity measure which takes into account the relative camera orientations and their distance from the scene. This measure prefers camera configurations suitable for 3D reconstruction, i.e. those with a short baseline and similar distance from the scene. We represent the collection of images as a fully connected graph with the nodes corresponding to the cameras and the edges between them corresponding to the introduced similarity measure. This formulation of the problem allows us to define the clustering problem as a graph partitioning and solve it using spectral clustering. The clusters we obtain bring down the amount of data that has to be considered by the reconstruction algorithms simultaneously. We tested our approach on a number of multi-view data sets and demonstrated that the clusters we obtain are suitable for 3D reconstruction and coincide with what a human observer would consider as a good clustering. We also show that the total reconstruction time spent when using the camera clusters to reconstruct the scene is significantly smaller than the time spent when using all images at once. In addition, it is obvious that the memory requirements are much smaller when clusters of cameras are used, compared to using the full image dataset with all cameras.

The remainder of the paper is structured as follows: In section 2 we discuss previous work dealing with camera clustering. Section 3 introduces our proposed clustering approach, while results on clustering in general and multi-view reconstruction in particular are presented in section 4. We conclude with section 5.

2. Related Work

The problem of processing large sets of images for the purpose of 3D reconstruction has in particular been brought to the attention of the community through the Photo Tourism project [15]. This led to improved solutions for many related problems, such as structure from motion from unorganized image collections [16], scene navigation [14], object segmentation in the scene [12] and scene summarization [13]. In all those problems camera clustering has been considered. The SFM problem of [16] concentrates on reducing a full set of available images to a skeletal subset which is sufficient to provide a full sparse reconstruction, where other image views can be quickly added using pose estimation. The skeletal image set is computed by a maximum leaf t-spanner [1] of the graph with cameras at its nodes and the edges being joint position covariances between the pairs of cameras. In the other works, especially the one of Simon *et al*. [13], the goal is typically not finding a good partitioning of the cameras, but instead the selection of canonical views. These views are desired to be orthogonal to each other, while at the same time showing a representative view of the object. The proposed approach of [13] was used for scene summarization through a set of orthogonal canonical views. They assume the availability of 3D scene points together with visibility information and use an ad-hoc greedy method to automatically find their canonical views. While this seems to give good results they only make use of the point visibility, disregarding information about camera configurations. In our implementation of their greedy algorithm the number of clusters is highly dependent on the choice of some weighting parameters proposed also in the original algorithm. In [4] Denton *et al*. propose a method to find canonical views for a set of silhouette images of an object, in order to use them for view-based 3D object recognition. To solve the problem the authors use semidefinite programming (SDP) on a graph which models the similarity between the views. In the context of robot localization in [2], the environment is represented by a large number of collected images stored in a database. To speed up the search of the most similar image, the database is reduced to the set of representative canonical views. This is done using a graph representation of the database images with weights measuring image similarity. The problem is solved using the graph pruning technique known as the Connected Dominant Set problem in graph theory. While we also use a graph for representing the relation between different views, we are looking for a partitioning of the graph and not for canonical views. In addition we use spectral clustering [18, 11] based on spectral graph theory, which is significantly easier to implement than related approaches and yields good results. Zaharescu *et al*. [19] also consider the topic of camera clustering. Similar to our approach they directly find the clusters. However, they only use visibility information and apply the k-means algorithm for clustering requiring advance knowledge about the number of clusters. Hornung *et al*. [8] also select images for multi-view reconstruction, but their focus is on removing less useful views instead of clustering them. To be more precise they add views maximally contributing to the quality of the initial rough reconstruction.

Compared to previous work, we perform camera clustering based on geometric information about both the scene and the camera positions. We do not need to know the number of clusters in advance and only need a rough proxy geometry, which can be obtained either by a sparse bundle adjustment or the use of the visual hull computed from camera images. Especially in the context of multi-view reconstruction silhouettes are typically available, making our algorithm easy to integrate into an existing reconstruction pipeline.



(a) Spectral Camera Clustering (b) Canonical view selection [13]

Figure 2. Comparison of the clustering given by our method (left) and the one given by the canonical views algorithm [13] (right). The canonical views fail in this case, since the scene is planar and all cameras see the same points. Our method takes the positions of the cameras into account and therefore performs a more reasonable clustering.

3. Camera Clustering

We represent the camera clustering problem as a graph partitioning problem by modeling the relationship between the cameras as a graph. This allows us to apply powerful methods from graph theory to find a solution to our problem. Each vertex in the graph represents a camera, while the edges connecting two vertices represent the similarity between these two views. Hence, the first step in our algorithm is to define an appropriate similarity measure. While some existing clustering algorithms only use point visibility, we also incorporate the viewing angle between camera pairs and the distance to the scene to obtain more meaningful clusters. This is demonstrated in Figure 2 where we compare the performance of our method to that of the canonical views algorithm [13], which we extended to provide clusters instead of only canonical views by assigning each view to its most similar canonical view. Since the scene is mostly planar and every camera can see it, we only obtain one cluster using the canonical views. The cameras are spatially quite well clustered, but due to the use of the non-discriminate visibility information the algorithm cannot take advantage of this. Our method on the other hand successfully finds the clusters, since it also takes the camera geometry into account.

Before we introduce our similarity measure, let us first define some notation. For each camera we have its projection matrix P_i, a set of surface points X and the visibility matrix V, which collects the information about the visibility of X in each view, i.e. the row-vector V_i contains the visibility of the scene points with respect to camera i. In addition we also use V_i to represent the set of points which are visible in camera i. The 3D scene points X can be obtained in one of two ways. The first possibility is to run a bundle-adjustment procedure (e.g. [5] or [15]). However, this can be time consuming, especially for large scenes. In order to speed this process up and since we only need a rough object proxy geometry we downsample the images and use a sparse sampling (as it is possible in the Furukawa method). The second possibility is to make use of silhouette images when available by building the object's visual hull and then using the mesh vertices for X.

Algorithm 1 Method Outline

1: Compute the graph laplacian $L = D - W$.
2: Solve the generalized eigenvalue problem $Lv = \lambda Dv$.
3: Select the eigenvectors corresponding to the k smallest non-zero eigenvalues after which there is a visible jump in the eigenvalues, disregarding the eigenvector corresponding to $\lambda = 0$.
4: Construct the matrix E from the k smallest non-zero eigenvectors and perform a clustering on its rows using the mean-shift algorithm.

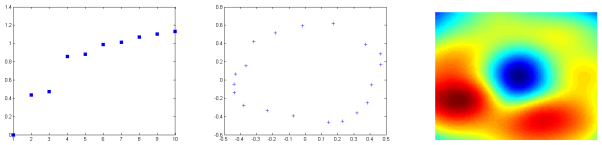
Once this information has been obtained we can compute our scene similarity measure. We want to be able to take into account both scene geometry and camera configuration. This is achieved by using the visibility information (scene geometry) and the viewing angles and scene distances for each camera-point pair (camera geometry). Our similarity measure between view i and j is given by:

$$S_{ij} = \sum_{X \in (V_i \cap V_j)} \frac{\|V_i \cap V_j\|}{\alpha S_{angle} + \beta S_{distance}}$$

$$S_{angle} = \arccos\left(\frac{(C_i - X)^\top (C_j - X)}{\|C_i - X\| \|(C_j - X\|}\right)$$

$$S_{distance} = \text{abs}(\|C_i - X\| - \|C_j - X\|)$$

where C_i and C_j are the camera centers of view i and j respectively. The term $\|V_i \cap V_j\|$, computed as the dot product between rows i and j of the visibility matrix, gives the number of scene points visible in both cameras i and j at the same time. The parameters α and β are weights to control the influence of the angular and the distance term (both



(a) (b) (c)

Figure 3. Eigenvalues and the clustering in the feature space of the Castle sequence of Figure 1. (a) The 10 smallest eigenvalues. Note the jump between eigenvalues λ_3 and λ_4 indicating that only two eigenvectors are needed for classification. (b) Plot of the entries of the eigenvectors corresponding to the two smallest eigenvalues (excluding the one being zero). (c) Same as (b) but overlayed with the Gaussian kernel used in mean-shift. Note that the layout of (b) resembles the camera positions, and that the maxima of (c) represent the clusters obtained by the mean-shift algorithm.

α and β are set to 1 in our experiments). The angular term is representing the angle between the viewing directions of the two cameras C_i and C_j with respect to the point X. The smaller it is, the less perspective distortion will occur between the points seen in the two cameras. This in turn will allow a more accurate and stable reconstruction. The distance term is meant to help with scene parts seen at different scales. Since many image-based similarity measures are not very robust with respect to scale changes, we try to group the cameras in such a way, that their distances to the scene are similar. This avoids unnecessary resampling of the images leading to more accurate reconstruction results. The nominator normalizes the score with respect to the number of common points visible in both cameras, in order to avoid any bias towards camera pairs which have a lot of points in common. This is desirable, since the proxy geometry we use might be more densely sampled (i.e. contain more points) in certain scene regions. By normalizing the similarity measure we are independent of the sampling density in the proxy geometry.

Our goal is to find the optimal partitioning of the graph, so that the nodes of the graph, i.e. the cameras, are separated in different groups according to their similarity. In other words, we want to find the partitioning of the graph such that the edges between the different graph parts have very small weight (similarity), and the edges within the same part have high weight, i.e. the views are very similar. This problem is NP-hard, but there exist approximative algorithms based on spectral graph theory, such as the normalized cuts [11]. It can be shown that the graph partitioning can be computed based on the solution to the generalized eigenvalue problem of the graph Laplacian [18]. Let W be the symmetric weighting matrix describing the edge weights, i.e. $W_{ij} = S_{ij}$, and D a diagonal matrix with $D_{ii} = \sum_{j=1}^{N} W_{ij}$. The Graph Laplacian is then defined as $L = D - W$. We solve the generalized eigenvalue problem $Lv = \lambda Dv$ and take the eigenvectors $u_1, u_2, \ldots u_k$ associated to the k smallest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$. Typically k is determined by observing all eigenvalues and taking the k smallest of them after which there is a visible jump in the eigenvalues (see Figure 3). This is known as the eigengap heuristic. Note that the smallest eigenvalue is zero and its eigenvector is constant, since every row of L adds up to zero. Therefore, we count from the first non-zero eigenvalue. In practice, we use $k = 2$ or $k = 3$ depending on when the jump in the eigenvalues occurs. We then use the eigenvectors corresponding to those eigenvalues, as shown in Figure 3 for the sequence of Figure 1, and perform a clustering on them. As it can be seen this representation resembles the real 3D positions of the cameras and already indicates the existence of separate clusters in this representation. Let E be the matrix containing the eigenvectors corresponding to the k smallest non-zero eigenvalues as its columns. Then, usually, the final step in the literature is to perform k-means clustering on the rows of E. However, this would require us to know the number of clusters in advance. Therefore we chose to adapt the mean-shift algorithm [3], which does not need to know the number of clusters in advance. The only parameter we have to choose is the width of the kernel. To choose the best value we applied a heuristic approach, i.e. we choose the kernel width as the mean of the k smallest eigenvalues. This heuristic gives very reasonable results. Our method is summarized in algorithm 1.

4. Results

We performed several experiments on real and synthetic datasets to validate our approach. Since there is no commonly agreed on definition of what constitutes a "good" clustering, we can only evaluate the obtained clusters in terms of their suitability for 3D reconstruction. Therefore, we desire the clusters to be coherent and local with respect to the part of the scene they are viewing. In addition, we check our results by reconstructing some datasets using the obtained clusters and compare the results to a full reconstruction using all the images.

The first dataset we used is the castle sequence provided by Christoph Strecha [17] consisting of 19 high-resolution images. We obtained a proxy geometry by reconstructing the scene using the method of Furukawa [5] after downsampling the images in order to speed up the run-time of the procedure. Our clustering algorithm finds five clusters as shown in Figure 1. To help assess the quality of the clustering we also colored the scene points. A point is considered to belong to a cluster when it is seen by at least two cameras in the cluster. This gives a good visual indication of which parts of the scene can be reconstructed using a cluster. The clusters exhibit a fair amount of overlap in the scene points which is important in order to obtain a good fitting between the reconstructed scene parts.

We also ran tests on the well-known dino and temple sequences from the Middlebury multiview evaluation [10] as shown in Figure 6. Here we used the visual hull, shown in the images as a proxy geometry. This is significantly faster than performing a multiview stereo reconstruction. For the dino we obtain three clusters, one covering each side and one looking at the head of the dino. This makes sense because the head part contains high curvatures and therefore should be reconstructed independently from the sides which are mostly planar. We reconstructed the dino model using the Furukawa method [5] and obtained a runtime of 59 minutes for the full reconstruction and 42 minutes for reconstructing the three clusters independently. This is a significant increase in performance while a visual comparison of the two reconstruction results shows no significant differences. For the temple we also obtain three clusters, one covering the front, one for the back and one for the narrow

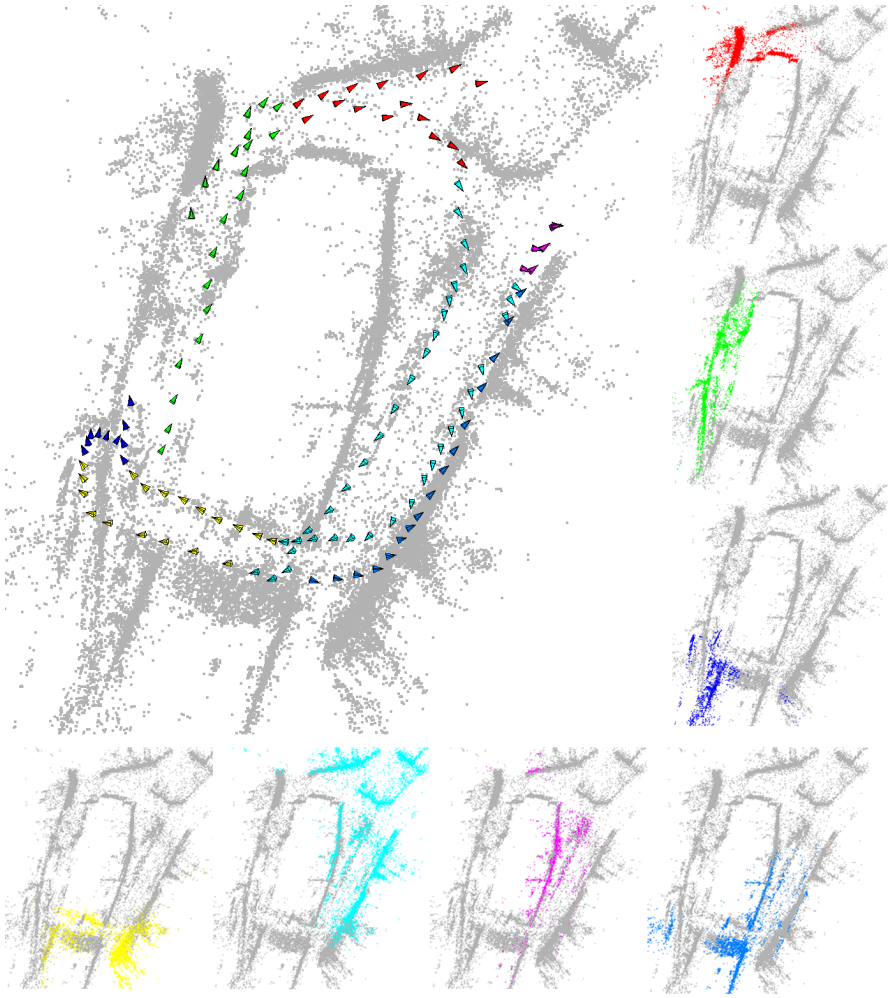


Figure 4. Clustering on the La Sarraz dataset. The scene is clustered into 7 separate clusters, which mostly follow the streets. See Figure 5 for representative views of each cluster and our reconstruction results.

side.

To show the validity on large scale scenes, we used a sequence of 125 high-resolution images provided by Christoph Strecha who also provides a multiview stereo reconstruction of the scene which we use as our proxy geometry. The sequence shows streets in the town of La Sarraz in Switzerland, where some streets have been traversed in opposite directions as indicated by the camera orientations in Figure 4. As you can see from our results in Figure 4 and Figure 5 we get a reasonable clustering, which mostly follows the streets. This result would not have been possible to obtain with the canonical views algorithm, since there is no point overlap between the first and the last views of most clusters, so that they would not have been clustered together. Also the running time of the canonical views is prohibitive due to the large amount of points in the scene (50433 points). After 15 minutes of computation it still hadn't converged to a solution. Our method on the other hand took only a few seconds to cluster the scene. We also performed a reconstruction on this dataset using the method of Furukawa [5] as shown in Figure 5. We reconstructed each cluster we found independently and merged the results. This took a total of 624 min. Reconstructing the full sequence on the other hand took over 25 hours and yielded an incomplete reconstruction with a lot of outliers. We suspect that this is due to the large number of images which lead to a much higher probability of having incorrect matches during the feature matching stage of the algorithm. This clearly shows that the use of good clusters not only speeds up the reconstruction times significantly, but can also help to obtain better results without having to deal with the higher complexity of the reconstruction task at the level of the reconstruction algorithm.

(a) Cluster 1 (red)

(b) Cluster 2 (green)

(c) Cluster 3 (blue)

(d) Cluster 4 (light blue)

(e) Cluster 5 (purple)

(f) Cluster 6 (turquoise)

(g) Cluster 7 (yellow)

Figure 5. Representative images for each cluster in the La Sarraz dataset shown in Figure 4 together with our reconstruction results.

5. Conclusion

We presented a method for camera clustering of large image sets into subsets. We modeled the camera clustering as a graph partitioning problem and introduced a new similarity measure which is used to weight the edges in the graph. It is based on the relative orientations between the cameras and their distance to the scene, and naturally favors configurations with a short baseline and similar distance from the scene. These properties are important for obtaining a good 3D reconstruction. We then use spectral clustering to partition our graph into suitable camera clusters. We showed that our algorithm provides reasonable results and runs very fast on large data sets. Even though the clustering is not intended to be used by any specific 3D reconstruction algorithm, we plan, in the future, to integrate it with a 3D reconstruction algorithm, so that we can alternate the clustering and the reconstruction to obtain high resolution reconstruction results in a reasonable time.

References

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9(1):81–100, 1993.

[2] O. Booij, Z. Zivkovic, and B. Kröse. Sparse appearance based modeling for robot localization. In *Proc. Intl. Conf. on Intelligent Robots and Systems*, 2006.

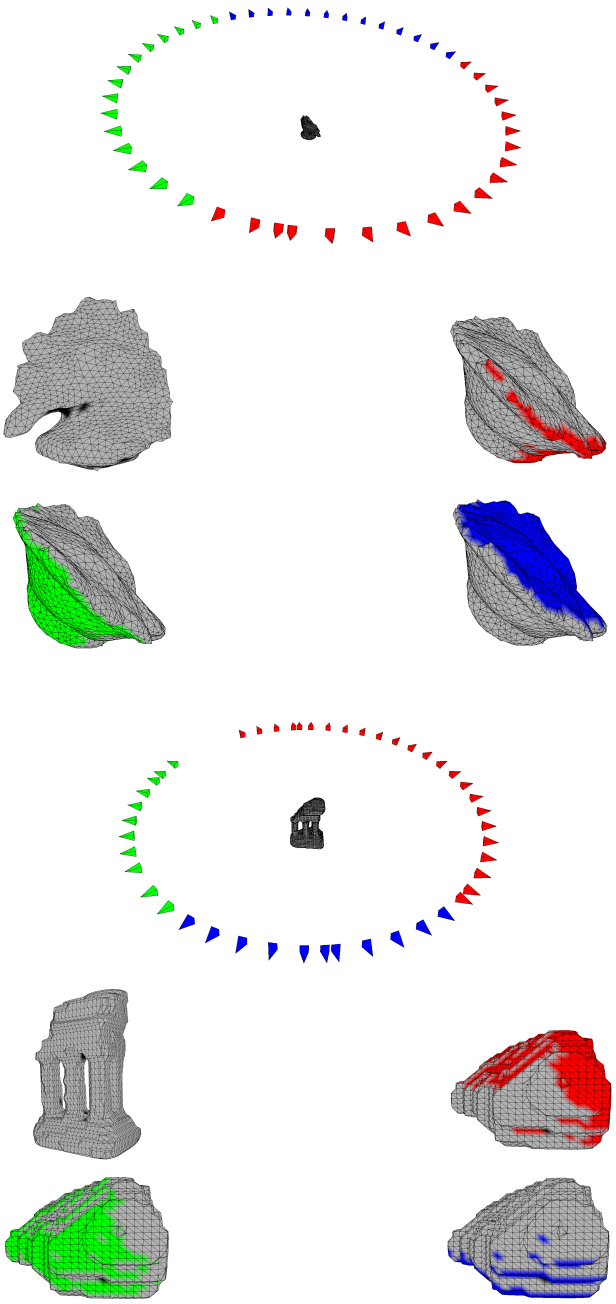


Figure 6. Clustering on the dinoRing and templeRing dataset from the Middlebury multiview evaluation. In this example we use the visual hull as a proxy geometry.

[3] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[4] T. Denton, M. F. Demirci, J. Abrahamson, A. Shokoufandeh, and S. J. Dickinson. Selecting canonical views for view-based 3-d object recognition. In *International Conference on Pattern Recognition*, pages 273–276, 2004.

[5] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[6] C. Hernandez and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[7] A. Hornung and L. Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[8] A. Hornung, B. Zeng, and L. Kobbelt. Image selection for improved multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2008.

[9] J. P. Pons, R. Keriven, and O. Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72(2):179–193, 2007.

[10] S. Seitz, B. Curles, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[11] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[12] I. Simon and S. M. Seitz. Scene segmentation using the wisdom of crowds. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 541–553, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[14] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, 27(3):11–21, 2008.

[15] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[16] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2008.

[17] 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 *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2008.

[18] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[19] A. Zaharescu, C. Cagniart, S. Ilic, E. Boyer, and R. P. Horaud. Camera clustering for multi-resolution 3-d surface reconstruction. In *ECCV 2008 Workshop on Multi Camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008.