# Live Feature Clustering in Video Using Appearance and 3D Geometry

Adrien Angeli a.angeli@imperial.ac.uk Andrew Davison ajd@doc.ic.ac.uk Department of Computing Imperial College London London SW7 2AZ

#### Abstract

We present a method for live grouping of feature points into persistent 3D clusters as a single camera browses a static scene, with no additional assumptions, training or infrastructure required. The clusters produced depend both on similar appearance and on 3D proximity information derived from real-time structure from motion, and clustering proceeds via interleaved local and global processes which permit scalable real-time operation in scenes with thousands of feature points. Notably, we use a relative 3D distance between the features which makes it possible to adjust the level of detail of the clusters according to their distance from the camera, such that the nearby scene is broken into more finely detailed clusters than the far background. We demonstrate the quality of our approach with video results showing live clustering of several indoor scenes with varying viewpoints and camera motions. The clusters produced are often consistently associated with single objects in the scene, and we forsee applications of our method both in providing cues for scene segmentation and labelling, and in building efficient 3D descriptors suitable for place recognition.

# **1** Introduction

Live 3D processing of the video from a standard single camera as it browses a natural scene has advanced significantly over recent years due to progress in processing resources and vision algorithms. The most notable success has been real-time Simultaneous Localisation and Mapping (SLAM) / Structure from Motion (SFM) algorithms which are able to track the camera's pose live while reconstructing a sparse but consistent map of natural features (e.g.  $[B, D, \square]$ ). Now that good solutions to camera pose tracking exist, attention naturally turns towards what additional 3D information can be extracted live from the same video stream. In applications such as advanced robotics where the aim is vision-based handling of general objects, or in augmented reality where virtual objects should ideally be able to interact physically with the real scene, dense surface modelling and understanding is the long-term aim, and systems with the ability to apply heavy computation to perform quality live dense reconstruction are starting to appear  $[\square]$ . However, in the interim there is still much more that can be achieved based on the feature-based reconstructions used by camera trackers.

In this paper our aim is to show that the application of sensibly formulated online feature clustering to the points generated by real-time monocular SLAM allows useful determination of groups of features which are closely related both in geometry and appearance. We consider this a problem in data-driven perceptual grouping, and the clusters may or may not correspond to individual objects in the scene; however they definitely correspond to characteristic, repeatable structure.

There exists a wide variety of approaches to perceptual grouping in computer vision, mostly applied to single images or operating on individual video frames, ranging from simple pixel-wise methods which group neighbouring pixels incrementally according to their similarity  $[\Box]$ , or those which rely on advanced energy minimisation techniques to cast the problem into a probabilistic framework  $[\Box]$ , to methods based on spectral analysis to find relevant clusters of image keypoints  $[\Box]$  in individual frames of a video sequence.

It is more challenging to integrate information from a whole video sequence. In the approach of  $[\square]$ , each pixel of each video frame is first mapped to a vector space encoding image position, RGB description and a time index, before a two-stage batch procedure is employed to cluster the mapped data points. The solution does not require any camera tracking, nor any estimation of the structure of the scene. In a very different approach based on camera tracking and geometry, Sturgess *et al.* [ $\square$ ] propose to analyse the sparse map of a scene to derive a semantic segmentation for every frame of the corresponding video sequence, based on domain-specific assumptions about road scenes. In particular, this work demonstrates that taking advantage of 3D scene structure and motion cues in addition to appearance information can lead to relevant dense partitioning of video frames.

In this paper, we propose to address for the first time the problem of *live* feature clustering in an incremental manner, by integrating appearance, structure and motion information 'on the fly', as the camera browses a scene. This poses new challenges, because information must be processed locally (*e.g.* at keyframes of the video sequence) for efficiency, while still ensuring global consistency over the whole video sequence. Our method is able to incrementally group features into clusters corresponding to natural objects by combining the efforts of two disjoint clustering algorithms: one for efficient efficient processing, and one for global integration. We demonstrate the quality of our approach with live clustering results from several indoor scenes, and focus on the consistency of the clusters generated.

### 2 Incremental Feature Clustering

Our key novelty is in the manner in which real-time clustering performance is enabled via a two-step local/global approach. The main enabling technology for our algorithm is real-time monocular SLAM (structure from motion), and it is desirable that this should operate at the frame-rate of the camera and provide accurate 3D positions estimates for as many point features as possible. We therefore use the monocular SLAM system described by Strasdat *et al.* in [12] which follows the main design of Klein and Murray's PTAM [11]. The output is a live camera pose estimate, and a continuously updated 'map' of the 3D positions of thousands of point features and poses of a representative set of historical keyframes.

We make direct use of these keyframes in our algorithm, as they are key to the two-stage manner in which we enable incremental, real-time clustering. First, the features from each single keyframe are clustered independently, using appearance and geometry information. Then, the output of each single keyframe clustering step is used to maintain global statistics about the co-occurrence of the features throughout the video sequence, and the obtained matrix of pairwise co-occurrence probabilities is used as input to a separate, *sequence* clustering algorithm. The idea underlying this approach is that features which appear relatively close and similar to each other in several keyframes will be clustered together by the keyframe clustering algorithm, leading to a high probability of co-occurrence for these features over the video sequence, hence favouring in the end their association in the same group by the sequence clustering algorithm.

In practise, the proposed sequential approach for incremental clustering offers several advantages over a 'traditional' batch-processing approach (where, for instance, a matrix of pairwise distances between the features is maintained over the video sequence and used as input to a clustering algorithm). First, single keyframe clustering can be done efficiently due to the limited number of features present in each keyframe (in general around 300). As a consequence, each keyframe can be processed in a few hundred milliseconds, in a separate thread of execution, without affecting the flow of the SLAM process. Second, the accumulated statistics lead to a very sparse matrix of probabilities, which can be processed efficiently by the sequence clustering algorithm, even when the scene contains a few thousands of features. The two clustering algorithms are now presented in detail.

### 2.1 Single keyframe clustering

The features in each keyframe are clustered independently, by a 'local' process which aims at grouping together features according to their appearance and 3D position. Each feature's appearance is characterised using a 1D RGB histogram (with all colour channels concatenated, and 16 bins per channel) computed over the  $15 \times 15$  pixel image patch extracted around the image position of that feature. Two histograms are compared using the  $\chi^2$  distance. For geometry information, we compare two features *i* and *j* with 3D positions  $x_i$  and  $x_j$  using the 3D distance between them normalised by the distance to the camera:

$$d_{\text{Geom}}(i,j)^2 = \frac{\|x_i - x_j\|^2}{\max(\|x_i - c\|^2, \|x_j - c\|^2)},$$
(1)

where ||x|| denotes the Euclidean norm, and *c* is the 3D position of the camera. The motivation for this choice of distance is to give the ability to adjust the level of cluster detail depending on the distance between the scene and the camera: when features are close to the camera,  $d_{\text{Geom}}(i, j)$  increases rapidly with  $||x_i - x_j||$ , whereas when the features are far from the camera the ratio of Equation 1 is dominated by the denominator, keeping  $d_{\text{Geom}}$  small.

Following the line of work of [**B**], we combine appearance and geometry information into a one dimensional similarity measure between features:

$$s(i,j) = \exp\left(-\frac{d_{\text{RGB}}(i,j)^2}{\sigma_{\text{RGB}}^2} - \alpha \frac{d_{\text{Geom}}(i,j)^2}{\sigma_{\text{Geom}}^2}\right), \qquad (2)$$

where  $d_{\text{RGB}}(i, j)$  is the  $\chi^2$  distance between the two histograms. This similarity measure can be interpreted as the probability that any two features *i* and *j* belong to the same cluster given their geometrical closeness and similarity in appearance. In the work of [**B**], all cues provide the same contribution in the exponential formulation of the similarity measure: the scaling parameters  $\sigma_{\text{RGB}}$  and  $\sigma_{\text{Geom}}$  are set to the same percentage (30%) of the range of the respective distance functions  $d_{\text{RGB}}$  and  $d_{\text{Geom}}$ . However, we found it important to give extra credit to the geometry cue via an additional scaling parameter  $\alpha$ . The clustering solution proposed in  $[\square]$  is based on a spectral analysis of the fullyconnected graph  $\mathscr{G}$  of pairwise similarities between all the features of a given frame. Graphbased clustering is ideal in situations, like ours, where only a graph (or the corresponding adjacency matrix) of pairwise similarity scores is available as input, while standard clustering methods assuming 'vector-valued' data points (like *k-means*, or *mean-shift*  $[\square]$ ) cannot be used directly. However, the complexity of spectral clustering methods currently prevents their use in live video analysis. We therefore need a different clustering algorithm, which should be efficient, unsupervised (*i.e.* the number of clusters should be determined automatically, online, and independently for each keyframe), and deterministic.

The information retrieval community has developed an abundance of clustering algorithms of all sorts. The problems of document classification and unsupervised document clustering in fact bear strong similarity with the task of visual feature clustering. According to a recent survey in those domains [IIG], the family of *Constrained Agglomerative Clustering* (CAC) methods provides the best compromise between efficiency and robustness. Efficiency comes from the hierarchical nature of the clustering process, whereas robustness is ensured by a two-stage procedure where early mis-associations due to outliers are avoided. However, the algorithm proposed in [IIG] is not deterministic, which is of crucial importance here.

We therefore designed a new clustering algorithm in the spirit of CAC. First, a divisive procedure separates the input data into a set of small yet very consistent clusters. Then an agglomerative algorithm merges the obtained clusters recursively, proceeding by pairs at each iteration until a suitable number of clusters is obtained. In [III], the merging operation is continued until all clusters form a single entity, and the ideal number of clusters (which corresponds to one particular level in the obtained hierarchy) is determined using ground-truth data. Because ground-truth data is not available in our case, we propose to determine the number of clusters automatically using model selection: the idea is to find the best compromise between the complexity of the model and the likelihood of the data given that model.

#### 2.1.1 The divisive clustering algorithm

We use as a divisive algorithm here the *medoid-shift* method [ $\Box$ ]. Put very simply, this is the analogue of the mean-shift algorithm when the input data is a pairwise distance matrix instead of vector-valued data points. Given a set of points, the medoid is the *real* point (*i.e.* it has to be one of the data points) which has the same properties as the mean: it minimises the sum of the distances to all the other points. The medoid-shift procedure is similar to the mean-shift procedure, except that medoids are considered instead of means: the idea is to recursively 'shift', until convergence, the medoids computed at every point towards regions of maximal density. In practise, the medoids can be computed once and for all at the beginning of the procedure, thus making the implementation of the medoid-shift algorithm very efficient on a GPU.

In mean-shift, the crucial parameter of the algorithm is the *radius*, which defines the size of the regions over which means are computed. In medoid-shift, a similar parameter is needed. In fact, the algorithm requires two matrices: a matrix **D** of pairwise distances between all the data points to cluster, and the corresponding matrix **S** of pairwise similarity scores. For instance, one can define  $\mathbf{S} = \exp(\mathbf{D}/\sigma)$ , using  $\sigma$  as the radius. In our case, **S** is the matrix of pairwise similarity scores computed according to Equation 2 between all the features of a given keyframe. Consequently, **D** is the matrix of pairwise distances with  $\mathbf{D}(i, j) = d_{\text{RGB}}(i, j)^2 + \alpha d_{\text{Geom}}(i, j)^2$ . The scaling parameters of Equation 2 can then be seen as the parameters adjusting the radius.

#### 2.1.2 The agglomerative clustering algorithm

Following the spirit of the CAC methods, the output of the medoid-shift is next subject to a recursive, agglomerative merging algorithm. The solution retained here is a simple *average linkage* algorithm: at each iteration, a similarity score for each pair of clusters (*i.e.* the linkage) is computed, and the two most similar clusters are merged. We use here the *MinMaxCut* linkage  $[\mathbf{G}] \ l(A,B) = sc(A,B)/(sc(A,A)sc(B,B))$ , where  $sc(A,B) = \sum_{i \in A, j \in B} s(i, j)$  denotes the similarity between clusters *A* and *B*.

It is well known [III] that simple recursive agglomerative algorithms based on linkage are prone to degenerate clusters in the presence of outliers in the data: incorrect associations can have a dramatic effect when they occur in the early stages of the procedure, when all or most of the points are not yet part of a cluster. CAC methods avoid these early mis-associations by first processing data 'globally', using the divisive algorithm, so that the entities used as input of the agglomerative procedure are already quite consistent.

#### 2.1.3 Determining the number of clusters

During the recursive agglomerative procedure, we need to automatically determine the level in the hierarchy corresponding to the optimal number of clusters. We propose a solution inspired by model selection, where the idea is to find a compromise between the complexity of the model (or the number k of its parameters), and the likelihood L of the data given that model. Such a compromise can be achieved by computing the Akaike Information Criterion (AIC) at each iteration n of the algorithm:

$$AIC_n = 2k_n - 2\ln(L_n) , \qquad (3)$$

where the subscript n denotes the iteration number. Here,  $k_n$  is the number of clusters at iteration n, and we approximate the negative logarithm of the likelihood as follows:

$$-\ln(L_n) = \sum_{C=1}^{k_n} \sum_{i \in C} -\ln(s(i, \text{medoid}(C))) .$$
(4)

Equation 4 approximates the negative logarithm of the likelihood as a residual: it is the sum of the squared distances to the medoids of the clusters obtained at iteration *n*. As the number of clusters increases, the residual monotonically decreases (in the limit, when we have one cluster per data point, the residual is 0, and the likelihood is maximal). In practise, we introduce a scaling parameter  $\beta$  to give more or less importance to the number of clusters: large values of  $\beta$  will favour configurations with few clusters, and vice versa.

#### 2.1.4 Summary of the single keyframe clustering method

To summarise, each time a new keyframe is available from SLAM, its features are clustered based on combined geometric distance and similarity measures, by first applying medoid-shift to create set of small and consistent clusters, and then merging these in a simple recursive agglomerative procedure, with the number of clusters determined automatically according to the AIC. Figure 1 shows example results.



Figure 1: The two stages of single keyframe clustering: raw output of the medoid-shift algorithm (left, centre left), and clusters obtained after merging (centre right, right). Clusters are shown in the image and 3D space. Note that this and the other figures in the paper are best viewed on-screen or in a colour print-out.

### 2.2 Sequence clustering

Single keyframe clustering is a local process, with each keyframe treated independently. Our goal here is to achieve live persistent clustering over a whole video sequence: we therefore need to combine all the individual results obtained for each keyframe into a globally consistent result. To achieve this we use the output of each single keyframe clustering process to maintain, over the whole video sequence, a matrix of pairwise co-occurrence probabilities between all the features in the scene: we increase the co-occurrence between any two features every time they are observed in the same single keyframe cluster. The idea is that if two features share similar appearance and are located close to each other, they should often be observed in the same single keyframe cluster between.

In the family of graph-based approaches to clustering, there exist methods based on the concept of *Random Walks* (RW) over a graph  $\mathscr{G}$ . A RW process is a walk over the nodes of a graph where at each step the node to visit next is selected randomly, with a probability depending on the weight of the edges leaving the current node: nodes connected by 'strong' edges are likely to be traversed one after another.

In practise, a RW process of t steps can be achieved by raising the Markov matrix **M** associated with  $\mathscr{G}$  to the power t. **M** is a *column stochastic* matrix: its columns sum to one, and each entry  $\mathbf{M}(i, j)$  is meant to represent the conditional probability p(i|j) of moving to node i in  $\mathscr{G}$  given that the current position is node j. **M** is usually computed by independently normalising the columns of the similarity matrix **S** encoding adjacency between the nodes of  $\mathscr{G}$  in terms of their relative similarity scores: the underlying conditional probabilities are therefore approximated by column-wise normalised relative similarity.

In our case, we have access to the matrix of pairwise co-occurrence probabilities p(i, j) between all the features of the scene. Therefore, computing the corresponding Markov matrix by column-wise normalisation of these probabilities would lead to the exact conditional probabilities p(i|j) required for a RW process, thus making clustering approaches based on RW the most suitable choice for sequence clustering. We opted for the *Markov Cluster* (MCL) algorithm [**D**]. The MCL procedure iteratively alternates between RW and re-scaling of **M**, pruning edges with negligible weights at each iteration, until convergence (*i.e.* when the resulting matrix  $\mathbf{M}_{res} \approx \mathbf{M}_{res}^2$ ). At convergence, the clusters naturally appear in the resulting graph as separate connected components, thus removing the need to specify the numbers of clusters beforehand.

The implementation<sup>1</sup> of MCL provided by Dongen [**D**] takes advantage of the sparseness

of the matrix of co-occurrence probabilities to achieve efficient sequence clustering, only requiring a couple of seconds even when the scene contains a few thousand features.

### **3** Experimental Results

Experimental results were obtained for several distinct indoor image sequences, using the same set of parameters.  $\sigma_{\text{RGB}}$  and  $\sigma_{\text{Geom}}$  were set to 40% of the range of their respective distance functions  $d_{\text{RGB}}$  and  $d_{\text{Geom}}$ . The scaling parameters  $\alpha$  and  $\beta$  were set to 10 and 5, respectively. Finally, the inflation parameter for the MCL algorithm was set to 2.

We first present results showing the consistency of the first stage of our algorithm, feature clustering for individual keyframes as a camera browses an indoor scene with several distinct objects (Figure 2).



Figure 2: Single keyframe clustering results for 3 different keyframes of an indoor video sequence. The frames are processed independently, so the colours of the clusters can differ from one keyframe to another. However, the general consistency of clustering is good.

We can see from Figure 2 that the obtained single keyframe clusters are consistent, and mostly correspond to the objects of the scene which are close to the camera. Moreover, we can observe that the algorithm has the desirable properly of grouping together features in the background which are far away from each other, even if they do not belong a single object, due to the normalised 3D distance introduced in Section 2.1. This adaptive level of detail has a real motivation in robot interaction tasks, where it is more important to get detailed scene understanding for things which are close to the viewing robot.

In a second experiment, we evaluate the consistency of the clustering algorithm across different camera motions: we browsed a scene with 4 different motions, and compared the obtained sequence clusters. The results are given in Figure 3: we can see that clustering a given scene with 4 different camera motions (of approximately the same magnitude) leads to comparable clusters, demonstrating the consistency of our solution. However, note that observing the same scene from very distinct perspectives will necessarily lead to different clusters, as expected given that the distance between features defined in Equation 1 is normalised with respect to camera position. We consider this a desirable property.

Finally, Figures 4 and 5 show additional clustering results for 4 other indoor video sequences. Notably, Figure 4 shows the consistency of the clustering of a cluttered scene across slightly different viewpoints: as the camera gets closer to the objects, more features are added to the already existing clusters associated with these objects.

All the results were obtained in 'live' conditions, *i.e.* as the camera was browsing the scene, on a standard laptop with a 2.93GHz Intel Core Duo processor and nVidia GeForce



Figure 3: Clustered 3D map of the same indoor scene with 4 different camera motions. The displacement of the camera can be seen at the bottom of each map. The object 'IG' is the initialisation grid used to bootstrap the SLAM system: it is therefore put in a different position for each experiment. Note that the scale of each map is independent.



Figure 4: Additional sequence clustering results for a cluttered scene, at different keyframes. Note the consistency of clustering as the camera gets closer and further away. Subfigures are ordered with respect to the time index of the keyframes in the video.



Figure 5: Additional clustering results for three other indoor scenes.



Figure 6: **Left:** keyframe computation time vs. number of features in the keyframe. The plot shows medoid-shift processing time (blue crosses), average linkage processing time (green triangles) and total keyframe processing time (sum of the previous two, red circles). **Right:** sequence clustering computation time vs. total number of features in the scene.

9600M GT graphics card. On average, single keyframe clustering took 99ms, while in the worst case sequence clustering, which is able to run amortized in the background, took 5855ms (with more than 2000 features in the scene). More detailed computation times are given in Figure 6 which gives keyframe and sequence clustering computation times with respect to the number of features per keyframe and in the scene. The maximum keyframe computation time never exceeded 314ms, which makes it possible to achieve real-time performance given that keyframes are usually dropped at 1–2Hz. Moreover, a significant part of the sequence clustering computation time is spent on the overhead of sending data to the MCL program which is currently only available as a separate binary executable. In fact, in all the experiments reported below, the MCL process itself (excluding data transfer costs) always took less than 3.5s.

# 4 Discussion

Our incremental clustering approach relies on a probabilistic interpretation of the similarity score (Equation 2) between two features. The weighting of the cues involved is of particular importance and has a dramatic effect on the quality of the clustering. If the weighting parameters are changed together, it is possible to adjust the coarseness of the clusters generated by the medoid-shift algorithm: small values of  $\sigma_{RGB}$  and  $\sigma_{Geom}$  tend to produce small and specific clusters, whereas larger values lead to coarse groups with high intra-class variance. And of course varying their relative values gives more or less importance to one cue or the other (by adjusting the parameter  $\alpha$ ). In our solution,  $\sigma_{RGB}$ ,  $\sigma_{Geom}$  and  $\alpha$  are fixed manually and have a global effect. Potentially, these weights could be learned beforehand from labelled training data to maximise the accuracy of the similarity scores, or adjusted locally. Despite its consistency across different camera motions, our algorithm suffers from some poor tolerance to 'unreliable' features, which only appear in a single keyframe, though this problem can be reduced by passing only repeatedly tracked features to the clustering process.

# 5 Conclusions

We have presented an incremental method for live feature clustering from video which combines both appearance and depth-adjusted relative 3D geometry information. A scalable, two level local/global approach allows us to achieve real-time performance in scenes containing thousands of features and generate repeatable clusters in real-time which often correspond to natural objects. Our approach was validated in several indoor video sequences, confirming that consistent clusters can be achieved with different camera motions observing the same scene. This live clustering algorithm has the potential to become part of the standard toolbox of live 3D camera tracking and reconstrution systems, and play a useful role either in guiding dense multi-label scene segmentation and labelling, though as the method is presently only a perceptual grouping approach this would require additional domain-specific heuristics or training. Alternatively, the method could be applied out of the box as the basis for efficient 3D descriptors for place recognition, aiming to bridge the gap between geometry estimation in systems such as  $[\square]$  and current place recognition systems which throw away geometry information to rely only on appearance (*e.g.*  $[\square]$ ).

### Acknowledgements

This work was supported by European Research Council Starting Grant 210346. We are very grateful to Francisco Estrada for useful advice, and to Hauke Strasdat, Margarita Chli and other colleagues at Imperial College London for discussions and software collaboration.

### References

- A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics* (*T-RO*), 24(5):1027–1037, 2008.
- [2] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proceedings of the International Conference* on Computer Vision (ICCV), volume 1, pages 105–112, 2001.
- [3] D. Chekhlov, M. Pupilli, W. W. Mayol, and A. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *Proceedings of the 2nd International Symposium on Visual Computing*, 2006.
- [4] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(5):603–619, 2002.
- [5] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (*PAMI*), 29(6):1052–1067, 2007.
- [6] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut for graph partitioning and data clustering. In *IEEE International Conference on Data Mining*, 2001.
- [7] S. Dongen. *Graph clustering by flow simulation*. PhD thesis, University of Utrecht, 2000.
- [8] F.J. Estrada, P. Fua, V. Lepetit, and S. Süsstrunk. Appearance-based keypoint clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

- [9] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.
- [10] G. Klein and D. W. Murray. Parallel tracking and mapping for small AR workspaces. In Proceedings of the International Symposium on Mixed and Augmented Reality (IS-MAR), 2007.
- [11] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [12] F. Perbet, B. Stenger, and A. Maki. Random forest clustering and application to video segmentation. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.
- [13] Y.A. Sheikh, E.A. Khan, and T. Kanade. Mode-seeking by medoidshifts. In Proceedings of the International Conference on Computer Vision (ICCV), 2007.
- [14] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale drift-aware large scale monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2010.
- [15] P. Sturgess, K. Alahari, L. Ladicky, and P.H.S. Torr. Combining appearance and structure from motion features for road scene understanding. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.
- [16] Y. Zhao and G. Karypis. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10:141–168, 2005.