Efficient Retrieval from Large-Scale Egocentric Visual Data Using a Sparse Graph Representation

Vijay Chandrasekhar, Wu Min, Xiao Li, Cheston Tan, Bappaditya Mandal, Liyuan Li, Joo Hwee Lim
Institute for Infocomm Research
{vijay,minwu,xlli,cheston-tan,bmandal,lyli,joohwee}@i2r.a-star.edu.sg

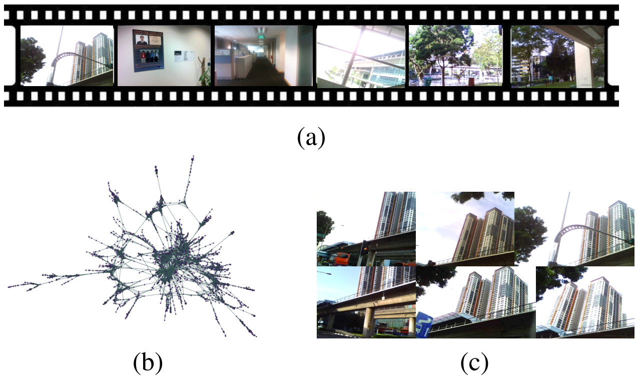
Abstract

We propose representing one's visual experiences (captured as a series of ego-centric videos) as a sparse-graph, where each node is an individual frame in the video, and nodes are connected if there exists a geometric transform between them. Such a graph is massive and contains millions of edges. Autobiographical egocentric visual data are highly redundant, and we show how the graph representation and graph clustering can be used to exploit redundancy in the data. We show that popular global clustering methods like spectral clustering and multi-level graph partitioning perform poorly for clustering egocentric visual data. We propose using local density clustering algorithms for clustering the data, and provide detailed qualitative and quantitative comparisons between the two approaches. The graph-representation and clustering are used to aggressively prune the database. By retaining only representative nodes from dense sub graphs, we achieve 90% of peak recall by retaining only 1% of data, with a significant 18% improvement in absolute recall over naive uniform subsampling of the egocentric video data.

1. Introduction

First-person-view systems will become popular with devices like Google Glass. These systems will enable new applications for visual search and augmented reality, exploiting what a person has seen in the past. Applications like summarization, scene understanding, object recognition and activity recognition are being explored in the context of egocentric visual data [26, 5, 18].

We envision a system where one's entire visual memory is captured, stored and indexed. We believe that such systems will have a wide range of applications in search, understanding and navigation. With visual search, such a system could be used for answering interesting questions like: Have I seen this object before ? When and where did I last see this person or object ? How often do I visit a place (e.g., a restaurant) ? Where am I right now ? or when was I here last ?


(a)

(b)

(c)

Figure 1. Figure *(a)* shows snapshots from an autobiographical egocentric video collected over a week. Figure *(b)* illustrates a sparse graph representation using Graphviz [4] of a 2-hour subset of the data (20K frames). Nodes correspond to individual frames and edges correspond to frames that match. Figure *(c)* illustrates images from a dense sub-graph cluster.

But before such a system can provide relevant and meaningful assistance to our queries, there is a genuine and pressing need to develop efficient ways to organize such voluminous visual data. In this work, we explore how egocentric visual memory should be stored and represented. We consider the most general (and typical) case where egocentric visual data is not labelled, and no GPS information is available. We explore a sparse-graph based representation of the data, and show how large-scale graph-processing algorithms can be used for efficient object retrieval. We summarize our contributions as follows:

- We propose representing one's visual experiences (captured as a series of ego-centric videos) as a sparse-graph, where each node is an individual frame in the video. Frames (or nodes) that have a valid geometric transform between them are connected by edges. The constructed graph is massive with hundreds of thousands of nodes, and millions of edges. We use graph visualization techniques to provide insight into typical subgraphs (substructures) found in such data, and show

how to use the graph based representation for efficient object retrieval.

- We show that popular global clustering methods like spectral clustering and *Graclus* [8] perform poorly for clustering egocentric graph data. We propose using local density-based clustering algorithms for clustering the data, and provide detailed qualitative and quantitative comparisons of different clustering schemes.
- We demonstrate object and scene retrieval from visual memory using our proposed graph-based representation and clustering, which are used to exploit redundancy in the data. By retaining only representative nodes from dense sub-graphs, we show how we can aggressively prune the data by an order of magnitude with only a small loss in recall. We achieve 90% of peak recall by retaining only 1% of data: a 18% improvement over a naive scheme like uniformly subsampling the egocentric video data.

2. Related Work

Graph-based representations of image collections have been used for detecting images of landmarks, label propagation, and 3-D reconstruction [7, 9, 17]. A graph over the entire image collection can be constructed efficiently using state-of-the-art content based image retrieval techniques [7]. In [9], Simon et al. start from connected components in a sparse-graph representation, and use clustering techniques on visual features to find representative views for scenes. In [17], Philbin and Zisserman find connected components in a large graph to identify similar images of individual landmarks. The authors also apply spectral clustering techniques on small connected components (tens of nodes) to seperate image-clusters that might have merged. In [11], the PageRank algorithm is used to find representative images in sets of images returned by text-based searches. In [7], Heath et al. discuss how to improve global connectivity when constructing graph-based representations of image collections from Flickr.

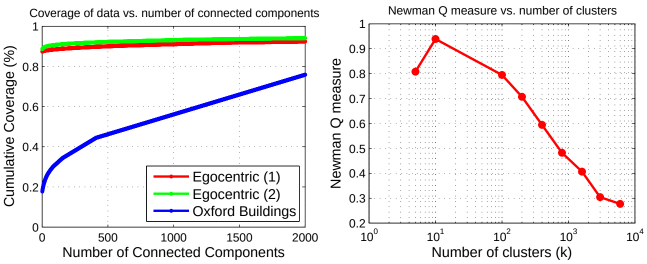
In our work, we consider graph-based representations of egocentric videos, and aim to find interesting patterns in the data. We collect over 20 hours of data over a week (over 1 million frames): capturing time at work, commuting to work, at lunch, in meetings and shopping. The nature of such data is highly different from that of image-collections. Sparse graph representations of image collections typically consist of small groups of connected components (typically tens of images), which correspond to individual objects or scenes. As a result, most of the work on sparse-graph representations stop at detecting connected components in the graph [9, 17, 7, 21]. In our case, connected component analysis would reveal little insight into patterns in the data. The connected components detected in our data are several orders of magnitude larger than those detected in image collection [17]. E.g., a single connected component covers over 90% of the 1M frames collected in egocentric video data over a week, as discussed in Figure 2(a). This is partly because of the contiguous nature of video data, but more importantly, because of the high levels of redundancy in visual data, i.e, we tend to see the same objects and scenes over and over again in our lives.

Graph clustering has been studied extensively in the literature (see [20] for a survey). Regardless of whether all nodes are assigned to a cluster or not, clustering algorithms can be broadly categorized into two categories: global and local. Popular global methods are based on spectral clustering [23], which are “cut-based” methods which partition data using eigenvectors of the adjacency matrix. Other examples of popular global clustering methods include multi-level partitioning approaches like *Metis* [12] and *Graclus* [8]. Local clustering algorithms, on the other hand, typically start from individual nodes and build dense clusters bottom-up by examining adjacency lists. Examples of such algorithms can be found in [13]. From the computational perspective, approximation algorithms and heuristic solutions are used for graph clustering, as the problem is NP-hard, and no constant-factor approximation algorithms are known [6].

In spite of the rich literature on graph clustering, selecting the appropriate algorithm is not straightforward and requires understanding of the underlying graph. [19] illustrates just one such example, where blindly applying traditional clustering methods like spectral clustering result in poor performance. In our work, we show why standard clustering methods, specifically, spectral clustering [23] and multi-level graph partitioning *(Graclus)* [8], perform poorly and are not suited to the problem. Instead, we propose using local density-based clustering algorithms for detecting dense subgraphs in the egocentric data.

3. Data Set

Existing egocentric data sets are inadequate for our application scenario, in that they either consist of clips with a single activity (e.g., cooking) [18], clips that are too short or consist of several short segments collected by multiple individuals [15], or clips with limited variety of content [5]. We are interested in egocentric data, that is autobiographical in nature i.e. capturing the activities of a single individual over extended periods of time (life-logging). Further, none of the existing data sets have a notion of “query” and “database” data, which we require for object-retrieval experiments. In our work, the “database” consists of the video sequences captured by the person, while queries are snap-shots of objects or scenes of interest seen by the person, which can be retrieved from the database. Our video database is collected by two different users: we name the data sets as *Egocentric (1)* and *Egocentric (2)* here-on. We use a *BH-906* spy-camera for collecting data, which is worn



(a) (b)

Figure 2. (a) Percentage of data covered as the number of connected components increases. Note that a single connected component covers over 90% of the *Egocentric* data-sets. (b) Newman Q measure for *Graclus* as the number of clusters k is increased. Typically, k corresponding to values between 0.3 and 0.7 are chosen.

over the ear like a blue-tooth ear piece. The *BH-906* has a short battery life: as a result, each video segment is typically less than half an hour. The camera has limited field of view, and captures data at VGA resolution (640 × 480) at 30 frames per second. Each data set consists of over 10 hours of video data captured over a week. To avoid long segments of video with no activity (as would be typical of a work-day spent in a cubicle), we select 1-2 hours of interesting activity per day, capturing a wide variety of content. Typical activities include commuting from home to work (bus or train), walking around the office, eating food at the pantry, manipulating objects of interest, shopping and meeting colleagues. The data consists of plenty of rapid motion, and captures a typical week in a person's life. The data is highly redundant, as would be typical of such autobiographical data. To the best of our knowledge, this is one of the largest autobiographical video data sets currently available, with over 1M frames per user, and captured over one week.

For queries, we took images of 100 objects or scenes of interest, seen by each user. We restrict our query data set to rigid objects. The queries are collected at a completely different time from the database videos, using a different capture device (the *iPhone4*). Queries consist of typical objects of interest like milk cartons, buildings, scenes at the work place, posters on the wall, restaurant signs, book covers, etc, which are seen one or more times by the user. The queries are collected independently of the database data. Some examples of database and query images can be seen in Figure 10. All collected data will be made available on our website [1].

4. Graph Representation and Visualization

We use Content Based Image Retrieval (CBIR) techniques [7, 16] for building a graph based representation. Each node in the graph denotes a video frame, and two nodes are connected if they have a geometric transform between them. We sub-sample the video data by 10× (resulting in 3 frames per second), as it suffices to capture rapid motion in the data. Each data set consists of ∼100K frames after subsampling.

Data Set	Number of nodes	Number of edges
Egocentric (1)	103,596	6,822,822
Egocentric (2)	115,585	18,690,994

Table 1. Details of graphs of egocentric data sets

The technique used for graph construction is similar to [7]. For local features, we extract Difference-of-Gaussian (DoG) interest points, and SIFT feature descriptors [22]. Since the data set is large, it is not feasible to perform pairwise comparison between all pairs of frames. As a result, we use a standard Bag-of-Words (BoW) retrieval pipeline [16, 2] for discovering matching frames. Geometric Consistency Checks (GCC), using RANSAC with a homography model are used to eliminate false positives. Up to 500 images are considered in the GCC step. The post GCC threshold is set to 12, which results in very low false positives.

For constructing the graph, each of the frames is queried into the BoW framework. Edges are added to the graph as matches are found. Setting edge weights using a normalized measure based on the number of matching features, did not lead to any significant improvement in our experiments. As a result, we set edge weight to 1 or 0, based on whether or not frames match. Such a graph is typically very sparse, compared to the total number of possible edges $O(N^2)$, where N is the number of nodes. The number of nodes and edges found are listed in Table 1. Note there are millions of edges in each graph. Also, note that different users' data can produce different graph statistics, as expected, based on their movement patterns.

To highlight the key difference between our data sets and typical image collections, we perform connected component analysis on the constructed graph data. In Figure 2(a), we plot the percentage of data covered, against the number of connected components, for the popular *Oxford* data set [16] (5K images), and the *Egocentric (1)-(2)* data (100K frames). The largest connected component in both egocentric data sets has more than 90K nodes. We note that the *Oxford* data set has small groups of connected components, while a single connected component covers over 90% of the *Egocentric* data-sets. From Figure 2(a), it is clear that connected component analysis suffices for discovering objects in image collections like *Oxford*, while yields little insight for our data. This motivates the need for more sophisticated algorithms to detect dense subgraphs from the data.

We also use *Graphviz* [4] for visualizing our graph-data. Typical subgraphs and substructures in the underlying data are shown in Figure 3(a)-(e). Figure 3(a) represents a clique, which corresponds to a static object or scene in the video, or an object that appears repeatedly. Figure 3(b) cor-

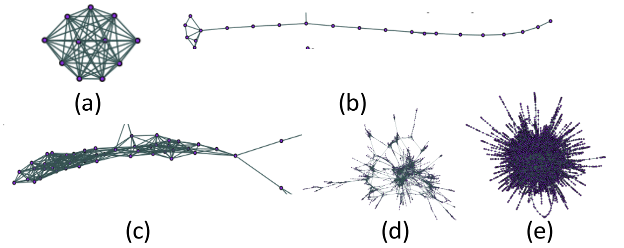


Figure 3. (a) corresponds to a clique-like structure representing a long static scene, or a repetitive scene or object, (b) corresponds to a chain representing rapid motion, (c) corresponds to a path taken multiple times. (d) and (e) show the largest connected component in 2 hours and 10 hours of data respectively.

responds to a chain, which represents rapid motion along a path, where only adjacent frames are similar to each other. Figure 3(c) corresponds to a path taken repeatedly by the user, which results in multiple chains merging together. Figure 3(d) and (e) are visualizations of the largest connected component in 2 hours of data, and the entire 10-hour data set, respectively. Figure 3(e) contains close to 90K nodes: the visualization of subgraphs provide intuition on typical structures present in the underlying data. Next, we discuss how to perform effective clustering on the graph, which we subsequently use for pruning the database.

5. Graph Clustering

In [21], the authors show how to prune an image-database by giving higher importance to images that belong to connected components, while giving less importance to singleton images, in a graph-representation. In analogous fashion, for egocentric video: chains are less important than dense subgraphs and clique-like structures, which typically correspond to salient objects or scenes, e.g., important static or repetitive scenes, or objects that are manipulated in daily activities. With graph clustering, our goal is to cluster frames of the same scene or object together. Ideally, we would like each cluster to correspond to an object in the real world. First, we discuss why traditional global clustering techniques do not work well for our data. Second, we use a local density clustering algorithm, that overcomes the short-comings of "cut" based approaches. Third, we provide detailed qualitative and quantitative comparisons of the different clustering algorithms.

5.1. Why not Global Clustering?

We first discuss two popular techniques for graph clustering.

- **Spectral Clustering.** Spectral clustering works by operating on eigen-vectors of the Laplacian matrix of the graph [23]. We use the implementation in [3].
- **Graclus Clustering.** *Graclus*, a multi-level graph partitioning algorithm, is based on repeated coarsening and refinement with emphasis on balanced cuts. *Graclus* is a popular replacement to spectral clustering, as it is faster and avoids the expensive eigenvector computation step.

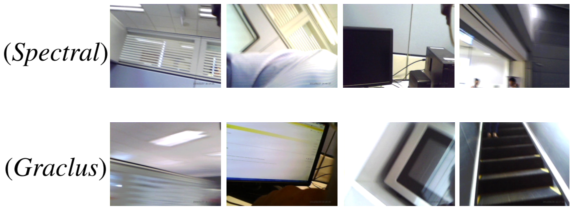


Figure 4. Examples of parts of clusters found with spectral clustering and *Graclus* clustering. Several clusters found have little internal coherence. Often, objects connected through long chains end up in the same cluster.

First, we note that eigenvector computation is the bottleneck in spectral clustering methods, making it quickly infeasible, as the size of the graph grows [3].For huge graphs of size in Table 1, spectral clustering is not practical. Hence, to illustrate problems with spectral clustering, we consider a graph with only the first 10% of the data in *Egocentric (1)*.

Also, another major problem with these approaches is choosing the number of clusters k apriori. Popular techniques for choosing k are based on the Newman Q measure [14]. Two popular approaches for choosing k are (1) repeat the clustering for different values of k and choosing the one with the highest Newman Q measure [14], or (2) choose k that corresponds to Newman Q values in the range of 0.3 to 0.7 [14]. In Figure 2(b), we plot the Newman Q measure for *Graclus* clusters for the *Egocentric (1)* data set. A similar trend is obtained for spectral clustering. We note that the number of clusters, which maximizes the Newman Q measure, is in the order of 10, which is too small to be useful. We pick $k \sim 1000$: Newman Q of close to 0.5.

While both the above approaches detect some clusters corresponding to a single object or scene, we find several problematic clusters when applying such top-down techniques to our data. First, we find plenty of clusters which lack any coherence. Second, objects connected through long chains often end up in the same cluster. Chains are highly common in the data, as shown in Figure 3, due to the nature of the egocentric video data. Third, disparate object clusters that might be connected through weak links end up in the same cluster. More generally, often clusters are under-segmented, or over-segmented, inspite of using the Newman Q measure for choosing k. All these can be attributed to the cut-based metric used for partitioning. Some examples of poor clusters are shown in Figure 4. Finally, note that even *Graclus* would not scale well, if we were to increase the data size by another order of magnitude. To overcome these issues, we propose using a local density-based clustering algorithm.

5.2. Local Density-based Clustering

In contrast to global clustering approaches, local clustering algorithms typically start from individual nodes and build dense clusters bottom-up by examining adjacency lists. A survey of subgraph detection algorithms can be found in [13, 25]. We surveyed the literature of local density clustering algorithms and build on the clustering algorithm proposed in [25]. We choose [25] as the starting point, as it was one of the top-performing clustering algorithms used for detecting dense sub-graphs in networks, as summarized in the recent survey [10], outperforming several other schemes based on Markov clustering, spectral clustering, partition-based clustering and density-based clustering.

The Parallel Local Density Clustering (PLDC) algorithm consists of two steps, namely (1) identifying preliminary clusters, and (2) refining them to obtain the final set of clusters. The set of preliminary clusters are chosen in a greedy fashion, based on local neighborhood graphs. These preliminary clusters contain nodes that have higher degree compared to nodes in their neighborhood. The preliminary clusters are typically very dense (clique-like): so an additional step is included to expand clusters in a greedy fashion. A cluster density parameter (t) in the algorithm is used to generate clusters with varying density. Precise definitions of cluster density and the exact algorithm used are presented in Appendix A.

The local clustering scheme in [25] is typically designed for small graphs with thousands of edges, while the PLDC algorithm presented in Appendix A can scale to large-graphs with tens of millions of edges. The preliminary clustering step in Algorithm 1 is the most time-consuming and we show how it can be parallelized for speed-up, i.e., we can divide the graph data (nodes and their adjacency lists) into blocks for computing preliminary clusters.

Finally, we observe that images in the same PLDC cluster tend to be of the same scene or object. Since egocentric data are highly redundant, we are motivated to prune the database further by selecting representative nodes in each cluster. For a cluster, we define its representative nodes as the minimum node cover [24]. The minimum node cover is a minimal subset of nodes, to which the remaining nodes in the graph are connected. The minimum node cover is a well-known NP-complete problem, and the heuristic used is discussed in the Appendix. The number of representative nodes in typical clusters ranges from a few nodes to tens of nodes.

5.3. Comparing Global and Local Density Clustering

Effect of changing PLDC parameter The parameter t represents how dense individual clusters are. The effect of changing t is shown in Figure 5 for data set *Egocentric (1)*. A higher t results in clusters with higher density, and a lower coverage of the entire graph.

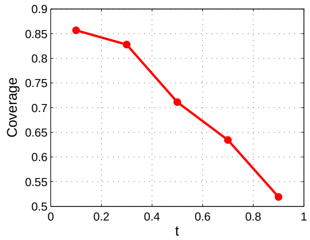
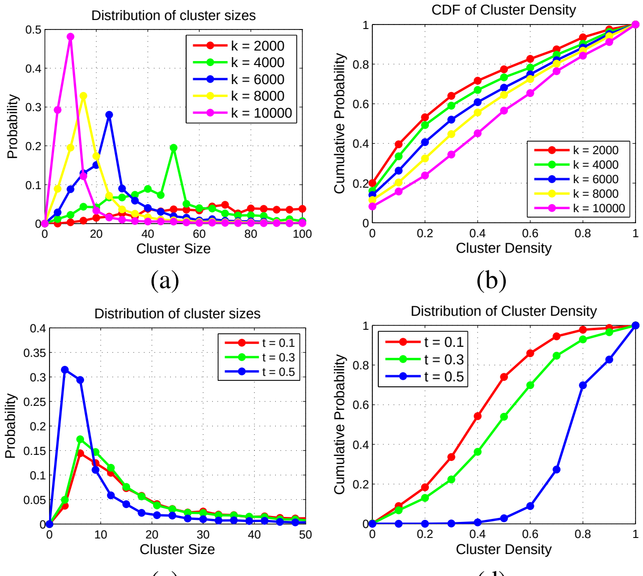


Figure 5. For PLDC, higher t results in clusters with higher density and smaller coverage of the entire graph.



(a)

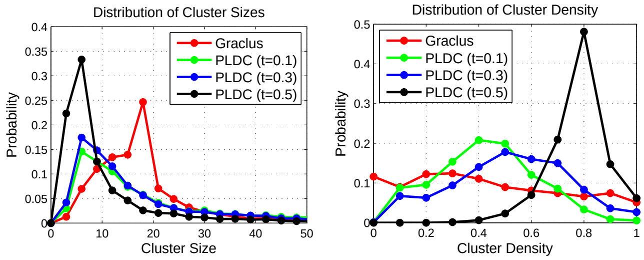
(b)

(c)

(d)

Figure 6. Comparing *Graclus* (a)-(b), and PLDC (c)-(d) as clustering parameters are varied. For PLDC, there are ∼6K clusters. From CDF of cluster density in (b) and (d), we note that PLDC is effective at finding dense subgraphs in the data, while increasing the number of clusters in *Graclus* is not effective.

Comparing distribution of cluster density and cluster sizes. In Figure 6, we describe how the distribution of cluster sizes and cluster density vary as clustering parameters are varied for *Graclus* and PLDC for *Egocentric (1)* data. For *Graclus*, clustering density is increased by increasing the number of partitions k. For PLDC, we increase clustering density by increasing t. For *Graclus*, it is intuitive that the distribution of cluster sizes shifts left (smaller average cluster size) as k increases from 2000 to 10000 in Figure 6(a). However, the Cumulative Distribution Function (CDF) of cluster density shifts right only gradually, i.e., even with $k = 10000$, more than half the clusters have less than 0.5 cluster density, as seen in Figure 6(b). This shows that a majority of clusters still contain chains or are weakly bound, due to the cut-based approach of *Graclus*. On the other hand, for PLDC, as t increases (number of clusters is



(a) (b)

Figure 7. Comparing *Graclus* and PLDC for the same number of clusters (∼6K). We note that distribution of cluster density for *Graclus* is almost uniform, while the distribution of cluster density for PLDC is skewed toward higher values.

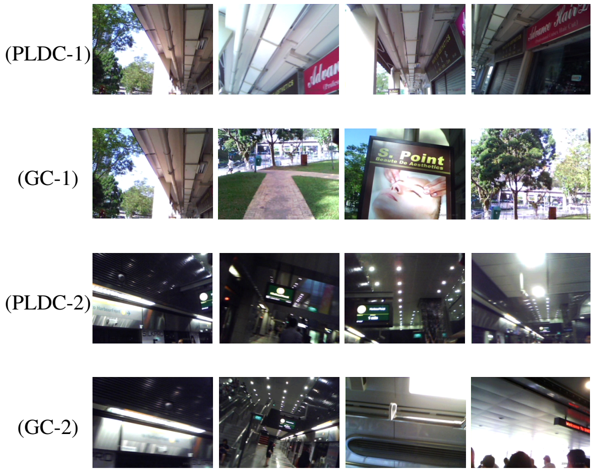


Figure 8. Examples of images from *Graclus* (GC) and PLDC clusters containing the same scene. (1):store-front, (2):train station. PLDC clusters belong to the same scene, while the corresponding *Graclus* clusters (GC) contains a wide range of scenes (connected through chains).

∼6000), we obtain clusters which are highly dense as seen from the CDF of cluster density for $t = 0.5$ in Figure 6(d). Note the difference in CDF distributions of cluster-density for PLDC with fewer (∼6000) clusters compared to 10000 *Graclus* clusters, in Figure 6(b) and (d).

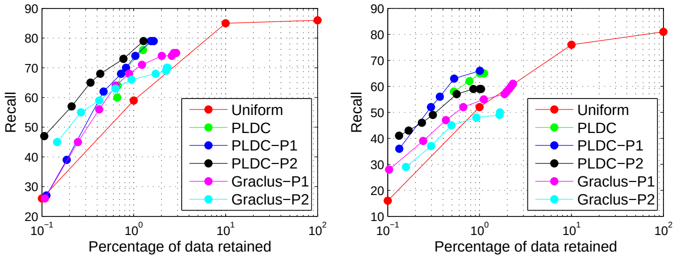
In Figure 7, we compare the distribution of cluster sizes and cluster density, for the same number of clusters: For *Graclus* ($k = 6000$), and for PLDC (∼6000 clusters). For *Graclus*, all nodes in the graph are labelled as determined by the algorithm, while for PLDC, a high percentage of 0.7 to 0.9 graph coverage is obtained as seen in Figure 5. We note in Figure 7(b), that the distribution of cluster density for *Graclus* is almost uniform, while PLDC results in smaller clusters, with cluster density distributions skewed towards higher values (increasing as t increases). In conclusion, PLDC is more effective in finding dense sub-graphs, than merely increasing the number of partitions in *Graclus* (or spectral clustering).

Qualitative comparisons. For *Graclus*, we choose a typical operating point corresponding to Newman Q measure of 0.5, while for PLDC, we set $t = 0.5$. We note that PLDC clusters typically correspond to the same object or scene, while *Graclus* often results in weak clusters. Two typical failure examples in Figure 8 show the case for *Graclus*, where multiple scenes connected by long chains end up in the same cluster. Finally, we discuss how the proposed clustering scheme can be used for efficient image retrieval.

6. Retrieval Experiments

Since egocentric data is highly redundant, we would like to aggressively prune the database while maintaining high recall. By retaining the most salient information in dense sub-graphs, we can prune the database and exploit redundancy in the data. We query 100 objects or scenes, with the collected egocentric videos as the database, using the same pipeline described in Section 4. Example query images and matching database frames are shown in Figure 10. For querying, we follow the same two-step pipeline described in Section 4, where we extract DoG features, shortlist 500 candidate images using an Inverted File System, and perform Geometric Consistency Checks (GCC) on the top ranked images, to eliminate false positives. We wish to retain a small percentage of data in the database, while maintaining recall. In Figure 9, we plot recall against the percentage of data retained for different pruning schemes:

- *Uniform*. The simplest scheme to prune the database is to uniformly sample the data. The points $10^{-1}, 1, 10, 100$ in Figure 9 correspond to picking every 1000^{th}, 100^{th}, 10^{th} and every frame (entire database) respectively. The point 1 corresponds to using the entire data-set of 1M frames.
- *PLDC*. We use *PLDC* with parameters discussed in Section 5. The number of clusters for datasets (1) and (2) is ∼6000 and ∼5000 respectively. We only retain the representative nodes in each cluster.
- *PLDC-P1*. Starting from *PLDC*, we prune clusters based on the cluster density measure. We retain representative nodes in the highest ranked clusters by cluster density, in increments of 1000 till all representative nodes are chosen.
- *PLDC-P2*. In addition to *PLDC-P1* pruning, we prune (1) clusters that are small (threshold=5) (2) clusters where all frames are closely spaced in time, by considering a threshold on the standard deviation of timestamps normalized by cluster size (threshold=0.6 works well).
- *Graclus-P1*. We apply *Graclus* clustering with 10K partitions. Similar to *PLDC-P1*, we retain representative nodes in the highest ranked clusters by cluster density



Egocentric (1) *Egocentric (2)*

Figure 9. Comparing different pruning schemes. Pruning based on the proposed PLDC scheme outperforms other schemes, with a significant improvement over naive uniform sampling. For *(1)*, we achieve 90% of peak recall by retaining only 1% of the data.

in increments of 1000, till all representative nodes in the 10000 clusters are chosen.

- *Graclus-P2*. In addition to *Graclus-P1*, we prune clusters based on criteria used in *PLDC-P2*.

First, we note that the maximum performance is achieved for the right-most point on the *Uniform* curve, corresponding to the entire database. There is a small drop in performance (2-5%) when we consider every 10^{th} frame, but performance drops drastically when we sub-sample by a factor of 10× and 100×. Second, we note that the *PLDC* based pruning schemes outperform their *Graclus* counterparts consistently, while also performing significantly better than *Uniform*. The *P2* pruning helps for dataset (1), but not for dataset (2), which suggests that clusters pruned in step *PLDC-P2* might still be useful. Third, compared to *Graclus*, *PLDC* schemes provide up to a 15% improvement in performance for both data sets.

Finally, consider peak performance on the *PLDC* curves: we can achieve 90% and 80% of peak recall of *Uniform*, while retaining only 1% of data: an increase of 18% and 14% in recall compared to *Uniform* for datasets (1) and (2) respectively. The *PLDC* scheme is effective in finding salient data in the database: *PLDC* finds dense subgraphs and clique-like structures, which typically correspond to salient objects or scenes, while discarding chains. In conclusion, the proposed approach can be used for maintaining an online cache of the most important database data, and can be used to significantly speed up retrieval time, compared to querying the entire database.

7. Conclusion

In this work, we show how a large sparse-graph representation of egocentric visual data can be used for efficient object retrieval. We show that popular methods like spectral clustering and *Graclus* perform poorly for clustering egocentric visual data, and propose using a density-based clustering algorithm for overcoming their shortcomings. By keeping only representative data from dense sub-graphs, we are able to prune the database by an order of magnitude with only a small drop in retrieval performance. In future work, we plan to consider databases that are 1-2 orders of magnitude larger than current size.

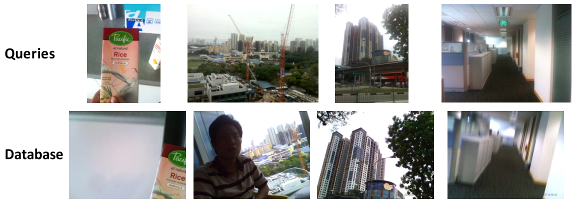


Figure 10. Example of query and matching images.

A. Parallel Local Density Clustering

The PLDC algorithm consists of two steps, namely (1) identifying preliminary clusters, and (2) refining them to obtain the final set of clusters. Before introducing the two steps, we briefly introduce some basic graph terminologies. Given a graph $G = (V, E)$, the degree of a node $v \in V$ is the number of edges connected to v in G (i.e., the number of v's neighbors), written as $deg(v, G)$. The density of G, denoted as $den(G)$, is defined as $den(G) = \frac{2\times|E|}{|V|\times(|V|-1)}$. For a vertex $v \in V$, the neighborhood graph of v consists of v, all its neighbors (i.e., $N_v = \{u|u \in V, (u, v) \in E\}$) and edges between them. It is defined as $G_v = (V', E')$, where $V' = \{v\} \cup N_v$, and $E' = \{(u_i, u_j)|(u_i, u_j) \in E, u_i, u_j \in V'\}$. Additionally, given two node sets V_A and V_B, $Sim(V_A, V_B)$ is defined as $Sim(V_A, V_B) = \frac{|V_A \cap V_B|^2}{|V_A|\times|V_B|}$: a measure of similarity or redundancy between node sets.

A.1. Preliminary clustering step

First, we find a set of preliminary clusters based on local neighborhood graphs. We define a node $u \in G_v$ (i.e., v's neighborhood graph) as a core node if u's degree in G_v is larger than or equal to G_v's average degree. Core nodes have higher degree than other nodes in G_v and thus are more likely to be part of dense clusters. We define the core graph of G_v as the subgraph consisting of all the core nodes and their corresponding edges.

A preliminary cluster is chosen and stored in a set PC if it satisfies the following constraints: (1) it is a subgraph of the core graph, i.e., all its nodes are core nodes, (2) it is dense (with density $\geq$ 0.7) in [25]) and (3) it is maximal, i.e., none of its super-graphs satisfy the first two constraints. We apply the core-removal algorithm in [25] to detect preliminary clusters starting from each node's neighborhood graph.

A.2. Refinement step

The set of preliminary clusters, denoted as PC, may overlap with each other, and we need to remove redundancy in the clusters. We construct a redundancy graph, denoted as $RG = (V_{RG}, E_{RG}, W)$, where the super vertex set $V_{RG} = \{C_i | C_i \in PC\}$ is the set of preliminary clusters. The super edge set $E_{RG} = \{(C_i, C_j) | C_i, C_j \in V_{RG}, Sim(C_i, C_j) \geq \omega\}$ links pairs of super vertices if their preliminary clusters overlap and satisfy the similarity criterion. E.g., setting ω to 0 results in non-overlapping clusters. The vertex weighting function W denotes the density of each preliminary cluster, i.e., $W(C_i) = den(C_i)$.

Both preliminary and refinement steps of PLDC are illustrated in Algorithm 1. We iteratively select the cluster with highest density (in Line 5), and remove the cluster and its neighbors (in Line 6) until the redundancy graph RG is empty. At this point, we have a set of clusters FC, where there is no overlap between clusters. Since our clusters are very dense, we also include a step to expand clusters (in Lines 9-11). Given a cluster $C_i \in FC$, we also include nodes v into C_i if $|N(v, C_i)|/|C_i| \geq t$, where $|N(v, C_i)|$ is the number of nodes in C_i connected to v, and $|N(v, C_i)|/|C_i|$ is the fraction of nodes in C_i connected to v. E.g., for $t = 0.5$, we expand the cluster C_i to include nodes that connect to more than half the nodes in C_i. By varying parameter t, we can generate clusters with varying density. Finally, the exact algorithm for computing representative nodes is presented in Algorithm 2.

Algorithm 1 Computing PLDC clusters

```
Input: Graph of egocentric data $G$.
Output: $FC$, final set of clusters.
  // preliminary clustering step
1: identify preliminary clusters $PC$ from $G$
2: construct $RG = (V_{RG}, E_{RG}, W)$ from $PC$
  // refinement step
3: $FC = \phi, i = 0, T_i = RG$ ;
4: while $V(T_i) \neq \phi$ do
       // select vertex $v_i$ with the highest density
5:     $v_i = \arg \max_{u \in V(T_i)} W(u)$;
       // update $T_i$: remove $v_i$ and its neighbors
6:     $T_{i+1} = T_i - (N_{T_i}(v_i) \cup \{v_i\}), i = i + 1$;
7:     $FC = FC \cup \{v_i\}$;
8: end while
  // expansion step
9: for each $C_i \in FC$ do
10:    $\forall v$, if $|N(v, C_i)|/|C_i| \geq t$, do $C_i = C_i \cup \{v\}$;
11: end for
```

Algorithm 2 Finding Representative Nodes

```
Input: $C$, a cluster detected from egocentric data;
Output: $Rep$, the set of representative nodes for $C$.
  // initialize the status for all nodes
$\forall x \in V, f(x) = 0$;
while($\exists x \in V, f(x) = 0$)
    // find the node with maximal degree
    $v^* = \arg \max_{f(v)=0} deg(v, C)$;
    // update node status
    $f(v^*) = 2$, insert $v^*$ into $Rep$;
    $\forall x \in N_{v^*}, f(x) = 1$;
    // mark subset of covered nodes as uncovered
    for $\forall x \in V$ and $f(x) = 0$
        $u^* = \arg \max_{u \in N_x, f(u)=1} deg(u, C)$;
        if $deg(u^*, C) > deg(x, C)$ do
            $f(u^*) = 0, f(x) = 1$;
        end if
    end for
end while
```

References

[1] *Egocentric Video Dataset*. http://www1.i2r.a-star.edu.sg/~vijay. 3

[2] Description of Test Model Under Consideration. *ISO/IEC JTC1 SC29 WG11 N12367*, December 2011. 3

[3] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):568–586, 2011. 4

[4] J. Ellson, E. Gansner, L. Koutsofios, S. North, G. Woodhull, S. Description, and L. Technologies. Graphviz open source graph drawing tools. In *Lecture Notes in Computer Science*, pages 483–484. Springer-Verlag, 2001. 1, 3

[5] A. Fathi, X. Ren, and J. M. Rehg. Learning to Recognize Objects in Egocentric Activities. In *Proceedings of CVPR*, Providence, RI, June 2011. 1, 2

[6] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29:2001, 1999. 2

[7] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image Webs: Computing and Exploiting Connectivity in Image Collections. In *Proceedings of CVPR*, SFO, California, June 2010. 2, 3

[8] Y. G. I. Dhillon and B. Kullis. Weighted Graph Cuts without EigenVectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, November 2007. 2

[9] N. S. I. Simon and S. Seitz. Scene Summarization for Online Image Collections. In *Proceedings of ICCV*, Rio de Janeiro, Brazil, October 2007. 2

[10] J. Ji, A. Zhang, C. Liu, X. Quan, and Z. Liu. Survey: Functional Module Detection from Protein-Protein Interaction Networks. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1–, November 2012. 5

[11] Y. Jing and S. Baluja. VisualRank: Applying PageRank to Large-Scale Image Search. *IEEE Transactions on PAMI*, 30(11):1877–, November 2008. 2

[12] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998. 2

[13] V. Lee, N. Ruan, R. Jin, and C. Aggarwal. A Survey of Algorithms for Dense Subgraph Discovery. In C. C. Aggarwal and H. Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 303–336. Springer US, 2010. 2, 5

[14] M. Newman. Fast algorithm for detecting community structure in networks. In *Physical Review E*, 2004. 4

[15] J. S. O. Aghazadeh and S. Carlsson. Novelty Detection from an Egocentric Perspective. In *Proceedings of CVPR*, Colorado, June 2011. 2

[16] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Proceedings of CVPR*, pages 1–8, Minneapolis, Minnesota, June 2007. 3

[17] J. Philbin and A. Zisserman. Object Mining using a Matching Graph on Very Large Image Collections. In *Proceedings of ICVGIP*, 2008. 2

[18] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *Proceedings of CVPR*, 2012. 1, 2

[19] B. Prakash, M. Seshadri, A. Sridharan, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *ICDMW*, pages 290–295, 2009. 2

[20] S. E. Schaeffer. Survey: Graph clustering. *Comput. Sci. Rev.*, 1(1):27–64, Aug. 2007. 2

[21] P. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *Proceedings of ICCV Workshops*, Kyoto, Japan, September 2009. 2, 4

[22] A. Vedaldi. An Implementation of SIFT Detector and Descriptor. In *Technical Report, University of California at Los Angeles*, pages 49–58, 2002. 3

[23] U. von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4):395–416, 2007. 2, 4

[24] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2001. 5

[25] M. Wu, X. Li, C.-K. Kwoh, and S.-K. Ng. A core-attachment based method to detect protein complexes in ppi networks. *BMC bioinformatics*, 10(1):169, 2009. 5, 7

[26] J. G. Y.J.Lee and K. Grauman. Discovering Important People and Objects for Egocentric Video Summarization. In *Proceedings of CVPR*, June 2012. 1