2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Visual Precis Generation using Coresets

Rohan Paul Dan Feldman Daniela Rus Paul Newman

Abstract— **Given an image stream, our on-line algorithm will select the semantically-important images that summarize the visual experience of a mobile robot. Our approach consists of data pre-clustering using coresets followed by a graph based incremental clustering procedure using a topic based image representation. A coreset for an image stream is a set of representative images that semantically compresses the data corpus, in the sense that every frame has a similar representative image in the coreset. We prove that our algorithm efficiently computes the smallest possible coreset under natural well-defined similarity metric and up to provably small approximation factor. The output visual summary is computed via a hierarchical tree of coresets for different parts of the image stream. This allows multi-resolution summarization (or a video summary of specified duration) in the batch setting and a memory-efficient incremental summary for the streaming case.**

I. Introduction

Imagine a mobile robot operating for long periods, day-after-day accruing visual information. Given an incessant stream of images recording its traversal, how does a robot summarize its visual experience capturing what was ordinary as well as perplexing? We explore the task of extracting a set of canonical images that provide a succinct topical representation of its operating workspace. In particular, we address the problem of summarizing continuous or incremental image streams possessing high degree of redundancy and perceptual similarity. This scenario is common for robot applications requiring high fidelity or large scale appearance-based mapping such as [1] and [2]. A direct organization of such data streams using proposed approaches like [3] and [4] can quickly become impractical even for a moderately-sized traversal. Further, due to the nature of data collection in real-life robotic applications, we seek an incremental online approach that scales well with time and saliency of data.

In this work, we present a hierarchical data-driven, algorithm that combines a semantic-compression or pre-clustering with a graph organization algorithm for generating visual summaries. Additionally, we leverage probabilistic topic models providing a thematic representation for image data [3]. The central idea is to compute online a semantically compressed subset, called coreset, for the image stream, possessing the property that every frame in the original stream has a similar representative in the coreset (under natural definitions of similarity). As computing the smallest possible coreset is NP-hard, we present an efficient algorithm for computing coresets whose size larger than the optimal by a provable small factor using memory poly-logarithmic in the size of the input. Importantly, the result obtained by running a summarisation algorithm on the coreset yields approximately the same results as running on the original stream. Since the coreset is significantly smaller than the original input, the overall running time is expected to be faster. Further, coresets can be constructed efficiently in merge-reduce style which allows our system to run in the streaming setting as well as in parallel. Combined with the incremental graph clustering, coreset trees allow generation of multi-resolution summaries as well as incremental summaries for streaming data with bounded memory use, see Figure 1. In summary, the main contributions of this paper are:

- An algorithm for near linear-time construction of small coresets for star clustering based summarisation in batch as well as the streaming setting.
- Theoretical guarantees for the existence, size and construction of coresets as defined above, and a proof that the star clustering algorithm applied to a coreset is within epsilon approximation to the equivalent result on the full data set.
- Construction of coreset trees in a map-reduce style combined with the incremental star clustering algorithm, yielding offline multi-resolution summaries and online bounded-memory summaries.
- Evaluation of the technique on publicly available data sets from a mobile platform demonstrating significant timing speed-ups with minimal or no loss of performance with summaries on coresets.

This paper is structured as follows. The next section reviews related efforts. Section III discusses image representation as a point in a low-dimensional topic space and the incremental star clustering approach [3]. Section IV presents the construction algorithm and theoretical guarantees for coresets. Section V discusses the multi-resolution and incremental summary generation. The experimental evaluation appears in Section VI and Section VII concludes this paper.

II. Related Work

Within mobile robotics, Ranganathan et. al. [5] use Bayesian *surprise* for identifying salient landmarks for topological mapping with vision and laser features. Girdhar and Dudek [4] extract k-most novel images from a visual map using set-theoretic surprise and bag-of-words image model. In [6] they used a classic 2-approximation for the k-center problem in order to extract such images. However, they could not obtain a provably correct streaming version for their algorithm and it is not clear how to run the algorithm in parallel. Our paper deals with these issues using coresets and their ability to run in a merge-and-reduce fashion.

Rohan Paul and Paul Newman are with the Mobile Robotics Group, University of Oxford, UK. e-mail: {rohanp,pnewman}@robots.ox.ac.uk.
Dan Feldman and Daniela Rus are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, USA. e-mail: {dannyf,rus}@csail.mit.edu.

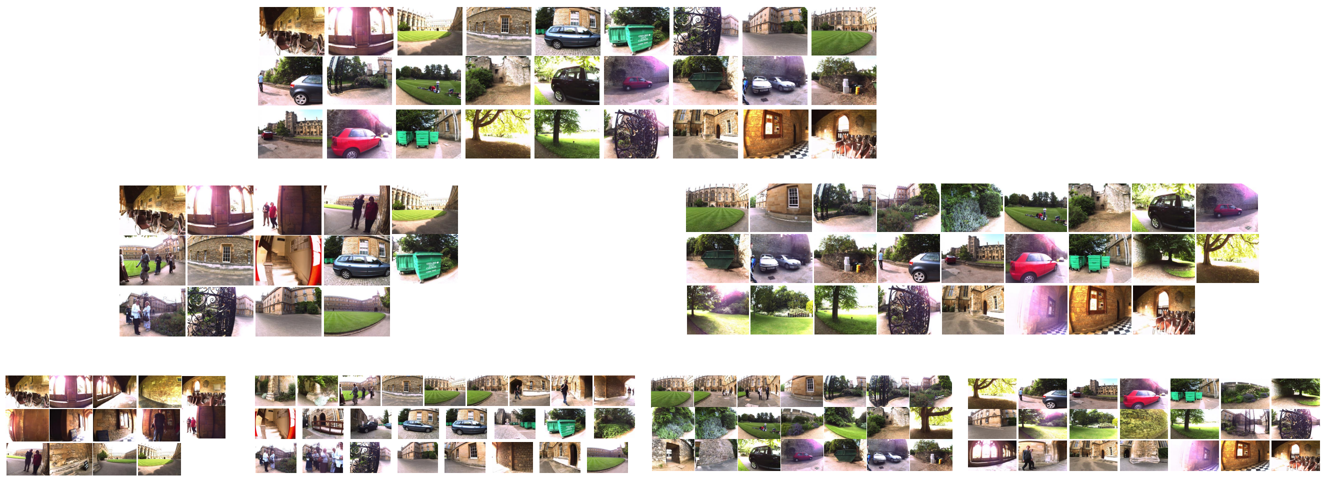


Fig. 1: **Visual summary computed on a binary coreset tree.** The figure shows three highest levels with leaf size 300 on the New College data set. Examine the lowest level (from left to right), the initial summaries consist of representative images from the cloisters area leading into the quad section. Later summaries contain images from the parklands, car park before returning to the cloisters area. Summary at the parent level is computed by merge-reduce operation on coresets at the lower level, representing scenes from a larger traversal sequence. This allows multi-resolution summaries in the batch setting and memory efficient summarization in the streaming setting.

In addition, the guarantee of our compression is $(1 + \varepsilon)$ approximation rather than 2. Unlike previous algorithms that take k as the input parameter, our main algorithm gets the error parameter as input and computes the appropriate k that achieves this bound.

The existence and properties of coresets has been actively studied within computational geometry. The problem of computing an r-coreset for a set of points in $\mathbb{R}^d$ is closely related to the classic k-center problem [7], where we wish to minimize the maximum distance from a set of n clients to a set of k servers. Gonzalez [8] suggested a 2-approximation for the k-center problem that takes time $O(nk)$. Exact or even $(1 + \varepsilon)$-approximations to these optimization problems generally require exponential computation time in k and have given rise to many approximation methods including *coresets*, or sparse representative sets [9]. This paper uses the off-line coreset from [10] and the streaming framework from [11], using $\log^{O(d)} n$ memory and update time per point, where the error ε and the number of centers k are constants.

III. Visual Summary Generation

We present a brief overview of the visual summarisation system used in this work [3]. For a mobile robot continually accruing images, the task is to produce a story board of canonical images summarizing the robot's visual experience. It captures the ordinary as well as novel appearances and evolves incrementally with acquired data.

Topic Space Representation. An acquired image is described as a vector of visual words [12] and further encoded as a mixture of latent visual topics [13], leading to a thematic representation in topic vector space. By mapping images to a low-dimensional thematic representation, images with common topics can get associated even if they have few words in common. Topic models based on Latent Dirichlet Allocation [13] are used for estimating the topic distributions and topic proportions per image vector. Inference is carried out using an MCMC Gibbs sampling procedure [14] in the state space of topic labels for observed visual words.

Incremental organisation using Star Clustering. After obtaining a suitable representation of images in topic space, the collected imagery is organised incrementally using an efficient graph clustering technique. For each cluster in the graph, the cluster center acts as its exemplar. The set of cluster centers forms a visual summary of the robot's experience. The graphical organisation evolves over time as new imagery is collected, yielding an ever-improving workspace summary. The star clustering algorithm [15] is used for topic-driven clustering. The algorithm aims to maximise similarity within intra-cluster elements using the cosine similarity metric. The number of clusters is not specified *a-priori*. Instead, it is naturally induced from the desired intra-cluster similarity, σ. Procedurally, the star algorithm constructs a pairwise similarity graph thresholded on the similarity σ specified. The thresholded graph is covered by a minimum cover of maximally sized star subgraphs containing a central vertex linked to satellite vertex nodes. In essence, the image corpus is represented as points on a unit sphere. The similarity between two points p, q on the sphere of angle θ between them is defined to be $\cos(p, q) := |\cos(\theta)|$. The algorithm seeks the minimum number of centers or *stars* such that every image point is associated with a center with similarity at least σ, for a given threshold $\sigma > 0$.

Scalability of Summarisation using Star Clustering. While we can run a summarisation system based only on the star algorithm and the topic model, the running time is dominated by the graph clustering algorithm that scales as $O(n^2 log^2 n)$ for a graph of n vertices. This can be prohibitive for real applications such as summarisation of large scale topological maps [16] or high frame-rate data sets [17] with overlaps and redundancy. In this work, we develop a scalable data-driven approach for summarisation. Instead of sending all collected imagery for star organisation, a pre-clustering or data reduction step is undertaken before running star organisation only on the representative samples. The central idea is to efficiently extract a small representative *coreset* from a given image corpus that further *guarantees* that any star clustering result obtained using the coreset is a bounded approximation to the equivalent result expected on the entire data set, which we discuss next.

IV. Coresets and Semantic Compression

In this section, we introduce coresets for visual summaries. We present a polynomial time algorithm for extracting coresets and provide theoretical guarantees for the existence of coresets. We prove that the result of the star clustering algorithm on the coreset is within a provably close bound of the equivalent result that uses full data set. Efficient construction of coresets in a map-reduce style is discussed in the following section.

A. Coreset Construction

A coreset can be considered a semantic compression of the data. Formally, we model every image in the streaming video as a point in a low d-dimensional topic space. A coreset is defined as:

Definition 4.1: For a given threshold $r > 0$, a set C is an r-coreset for the set P of points (frames, images) seen so far in a stream, if each $p \in P$ has a representative c in C of Euclidean distance at most r, i.e, $\text{dist}(p, c) \leq r$.

The coreset construction algorithm is given in Algorithm 1. Our coreset algorithm computes an r-coreset of size that is larger by a small factor as compared to the smallest possible r-coreset of the stream at every given moment. This property is summarised by the following theorem:

Theorem 4.2: Let P be a set of n points in $\mathbb{R}^d$. Let $r > 0$, and let $k = k(P, r)$ denote the size of the smallest r-coreset for P. Then Algorithm 1 returns, with probability at least $1 - 1/n$, an εr-coreset C for P of size $|C| = O(k \log^3 n/\varepsilon^d)$. The running time is $(nk + k/\varepsilon^d) \cdot (\log n)^{O(1)}$.

Note that since two points on the unit sphere that are very close to each other, also have a small angle between them, this includes the optimisation function of the star algorithm is as follows.

Observation 4.3: Let P be a set of n points in $\mathbb{R}^d$, and $\sigma, \varepsilon > 0$. Suppose that S is a set of star images (points on the unit sphere), that has similarity of at least σ to each image in P. Then S has similarity of at least $\sigma(1 - \varepsilon)$ to each point in an $(\varepsilon\sigma)$-coreset of P.

Our algorithm computes r-coreset of size that is larger by a factor of $\beta = (\log n)^{O(d)}$ compared to the smallest possible r-coreset of the stream in every given moment. While theoretical lower bounds [18], [19] imply that significantly better approximations do not exist, we show that our implementation yields a much better approximation than β in practice. This is probably due to the fact that real video streams contain much more structure than the worst case (and syntectic) examples that are used in the analysis.

For example, P is a coreset of itself for every $r \geq 0$. However, in order to get an efficient compression for P, we wish to compute a *small* r-coreset for P; see Fig. 3(a). In other words, we wish to cover the points of P by a small number of balls, each of radius r. The centers of these balls

Algorithm 1: Eps-Coreset(P, k, ε)

```
Input: A set $P \subseteq \mathbb{R}^d$, an integer $k \geq 1$ and an error
       parameter $\varepsilon \geq 0$.
Output: A set $C \subseteq P$ that satisfies Theorem 4.2.
1  $C \leftarrow \emptyset$
2  while $|P| \geq 1$ do
3      $n \leftarrow |P|$
4      Set $c$ to be a sufficiently large constant whose exact
       value can be determined from the proof of
       Theorem 4.2
5      Pick a uniform random sample $S$ of $\left\lceil \frac{ck \log^2 n}{\varepsilon^2} \right\rceil$ i.i.d.
       points from $P$
6      Set $Q$ to be the $\lceil n/2 \rceil$ points $p \in P$ with the smallest
       value $\text{dist}(p, S)$
7      Set $r \leftarrow \text{dist}(Q, S) = \max_{p \in Q} \text{dist}(p, S)$
8      for each $s \in S$ do
9          $G_s \leftarrow$ a squared grid of side length $2r$ that is
           centered at $s$ and contains cells with side length
           $\varepsilon r$
10         for each cell $\Delta$ of $G_s$ do
11             $q \leftarrow$ an arbitrary point in $\Delta$
12             $C \leftarrow C \cup \{q\}$
13     $P \leftarrow P \setminus Q$
14 return $C$
```

will be the desired r-coreset. Given a set of centers $C \subseteq \mathbb{R}^d$, we denote the needed radius to cover P by balls that are centered at these centers as

$$\text{dist}(P, C) := \max_{p \in P} \text{dist}(p, C) = \max_{p \in P} \min_{c \in C} \text{dist}(p, c).$$

Since the coreset size is only poly-logarithmic in n, running the star algorithm on the coreset would also take poly-logarithmic time. Since the coreset construction takes time linear in n, the overall running time on the coreset is thus linear in n, rather than $O(n^2 \log^2 n)$.

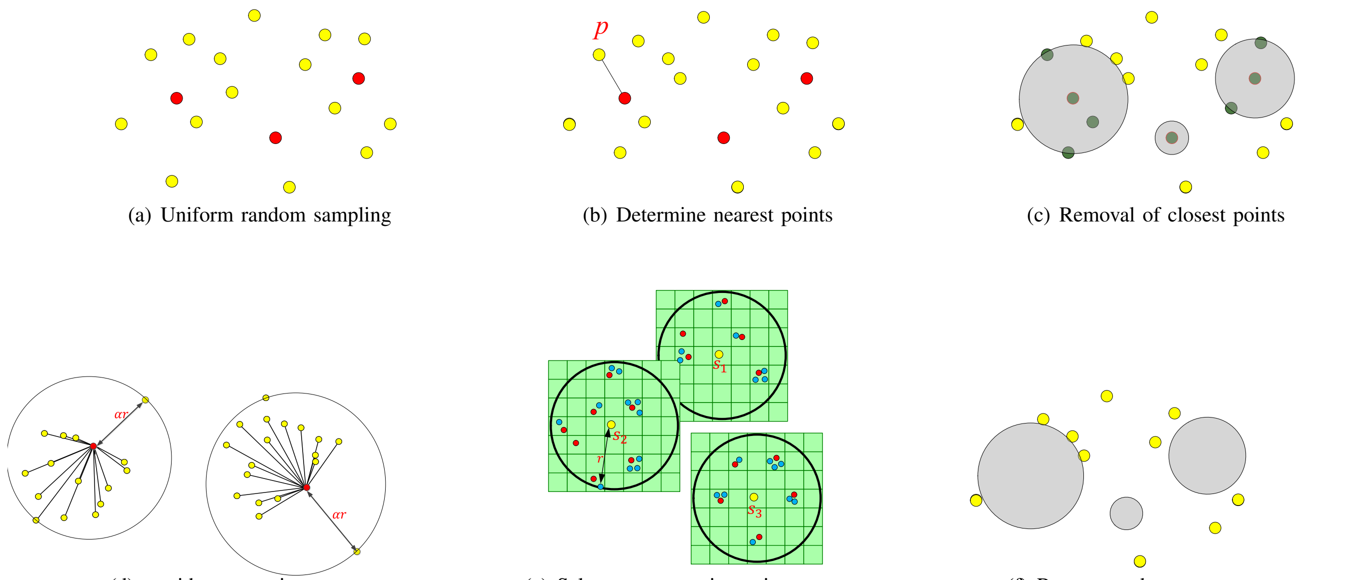
B. Overview of Algorithm

Figure 2(a) illustrates the key stages of the coreset construction algorithm, Eps-Coreset(P, k, ε). The algorithm gets as input a set P of n points in $\mathbb{R}^d$, the size $k = k(r)$ of the smallest r-coreset for P, and $\varepsilon > 0$. It returns an (εr)-coreset $C \subseteq P$ that satisfies Theorem 4.2.

We first pick a uniform random sample S of $\Theta(k \log^2 n/\varepsilon^2)$ points from P; see Fig. 2(a). Next, we determine the set Q of the half closest input points to S and remove them from P; see Fig. 2(b)-2(c). The next steps compute semantic compression C_Q for the set Q.

Let $r = \text{dist}(Q, S)$ denote the maximum distance from a point in Q to its closest point in S. For each $s \in S$, construct a square grid G_s that is centered at s. The side length of G_s is $2r$ and each of its cells is a square whose side length is εr; see Fig 2(d). Hence, there are $O(1/\varepsilon^d)$ squares in G_s and the union of the grids covers Q.

For every $s \in S$, pick a representative point $p \in Q$ from each cell in G_s. Hence, every point of Q has a



(a) Uniform random sampling (b) Determine nearest points (c) Removal of closest points

(d) ϵ-grid construction (e) Select representative points (f) Remove and recurse

Fig. 2: **Corset construction.** An illustration of Algorithm 1 with an example data set. (a) A small uniform random sample S (red points) from the input points P (in yellow) is picked (*line 4*). (b) For each input point $p \in P$ (in yellow) the distance $\text{dist}(p, S)$ to its nearest sample point in S (red) is determined (*line 5*). (c) The subset Q containing half of the input points closest to the red sample points in S is computed (*line 5*). (d) A single representative point (in red) from each cell in the grid G_s is added to the output coreset (*lines 8–10*). (e) The grid G_s is computed and the coreset C (in red) is updated for each cluster of input points (blue) that were closest to the same sample point (in yellow) see (*lines 7– 11*). (f) The algorithm is repeated recursively from (a), with the remaining points $P \setminus Q$ (that were not covered by the grids) as the input (*lines 1–12*).

```
Algorithm 2: $r$-CORESET$(P, r, \varepsilon)$
  Input: A set $P \subseteq \mathbb{R}^d$, a threshold $r \geq 1$, and error
         parameter $\varepsilon > 0$.
  Output: A set $C_k \subseteq P$ that satisfies Proposition 4.4.
1  $i_{\min} \leftarrow 1$; $i_{\max} \leftarrow n$
2  while $i_{\min} \leq i_{\max}$ do
3      $k \leftarrow \left\lceil \frac{i_{\min} + i_{\max}}{2} \right\rceil$
4      $C_k \leftarrow$ EPS-CORESET$(P, k, \varepsilon)$
       /* See Algorithm 1                                */
5      $r_k \leftarrow \text{dist}(P, C_k)$
       /* Where $\text{dist}(P, C_k) := \max_{p \in P} \min_{c \in C_k} \|p - c\|$.
          */
6      if $r_k > r$ then
7          $i_{\max} \leftarrow k - 1$
8      else
9          $i_{\min} \leftarrow k + 1$
11 return $C_k$
```

representative at distance of at most $\sqrt{d}\varepsilon r$, so the union C_Q of representatives is an $(\varepsilon r\sqrt{d})$-coreset for Q; see Fig. 2(e)

We now remove Q from P, and recursively compute coresets for the rest of the points by running the algorithm again for the remaining set of points $P \setminus Q$ instead of P; see Fig. 2(f). We repeat the iterations until there are no points left in P. The algorithm returns $C' = \bigcup C_Q$, the union of coresets that were constructed during all the iterations.

Note that Algorithm 1 requires k the size of the smallest possible coreset as an input. This is obtained via a binary search on the value of k using Algorithm 2. We discuss this in the next sub-section.

C. From k-Center to Coresets

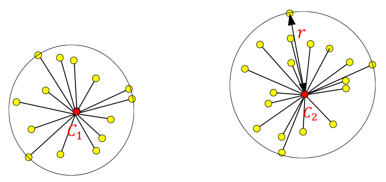
The problem of computing the smallest r-coreset is a variant of the well known k-center problem, where, for a given integer k, we wish to cover P by k identical balls whose radius is minimal. That is, compute a set that minimizes $\text{dist}(P, C)$ over every set $C \subseteq \mathbb{R}^d$ of size $|C| = k$. Suppose that the optimal (smallest) r-coreset has size k. That is, k is the minimum number of balls of radii r that are needed to cover P. Hence, the smallest radii that is needed to cover P by k balls is at most r. In particular, for the k-center C^* of P we have $\text{dist}(P, C) \leq r$, implying that C^* is an r-coreset for P.

From the previous paragraph, we conclude that given the size k of the smallest r-coreset, the k-center C^* of the input set is the desired coreset. Since we do not know the value of k in advance. Instead, we can simply compute the k-center C_k of P for all the possible values of k from 1 till n, and return the smallest k such that $\text{dist}(P, C_k) \leq r$. A better implementation would be to use the fact that $\text{dist}(P, C_k)$ can only decrease with larger values of k, and therefore use binary search over k, as shown in Algorithm 2 that evaluate the k-center for only $O(\log n)$ number of iterations. We summarize this result in Proposition 4.4.

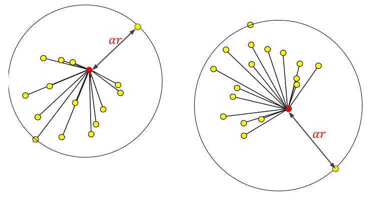
Proposition 4.4: Let C^* denote the k-center of P. Suppose that for every $k \geq 1$, EPS-CORESET(P, k, ε) *returns a set C such that*

$$\text{dist}(P, C) \leq \text{dist}(P, C^*).$$

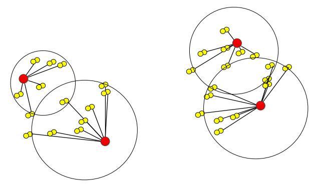
*Then for every $r > 0$, the output $C = r$-*CORESET$(P, r,$ K-CENTER$)$ *of Algorithm 2 is an r-coreset for P. Moreover, if also $|C| = k$ then C is the smallest among all possible r-coreset of P.*



(a) Optimal coreset



(b) α-approximation



(c) (α,β)-approximation

Fig. 3: **Optimal coresets and approximations.** (a) The smallest r-coreset C^* is the minimum number $k(r)$ of balls of radii r that are needed to cover the input points in $\mathbb{R}^d$. The points are on the plane ($d = 2$), and the $k(r) = 2$ centers $C^* = \{c_1, c_2\}$ of the balls are red. The radius of each ball is $r = \text{dist}(P, C^*)$. A set C of centers is an: (b) α-approximation for the smallest r-coreset C^* if it has the same size $|C| = |C^*| = k(r)$ as the smallest r-coreset, but the maximum distance from each point to its closest center in C is only larger by a factor of at most α, i.e, C is an αr-coreset. In the general case, α cannot be smaller than one. Each of the $|C| = 2$ balls around the red points has radius at most αr. (c) (α,β)-approximation for the smallest r-coreset C^* if it consists of *more* centers than C^* by a factor of β, and the maximum distance between an input point to its center is only αr, i.e., C is an αr-coreset of size at most $\beta \cdot |C^*|$. Here $|C| = 4 = \beta|C^*|$ so $\beta = 2$. Since C uses more centers than the smallest r-coreset C^*, it might be that $\alpha \ll 1$.

A similar reduction works for the opposite direction: using an algorithm that computes an r-coreset for P, we can compute the k-center of P for a given $k \geq 1$ by running binary search on the value of r.

D. Approximations for k-Center

Since the k-center problem is NP-hard for non-fixed k, i.e, has running time that is at least exponential in k, we conclude the following from Proposition 4.4.

Proposition 4.5: Every algorithm that computes the smallest r-coreset C^* for P has running time at least exponential in the size of C^*.

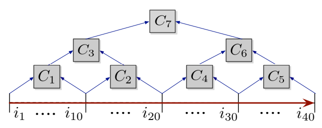
The last proposition suggests the use of approximation algorithms or heuristics for computing r-coresets. Let $k(r)$ denote the size of the smallest r-coreset of P. It is left to show that we can actually compute such a small εr-coreset C. The algorithm Eps-Coreset(P, k, ε) constructs such a coreset under the assumption that the size $k = k(r)$ of the smallest r-coreset for P is known and given as input. It returns an εr-coreset whose size is proportional to k. We will remove this assumption and replace the input parameter $k(r)$ by r using binary search on k, as shown in Algorithm 2.

V. Coreset Tree and Scalable Summarisation

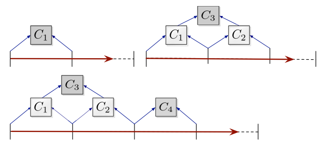
A key advantage of coresets is that they can be constructed in parallel with batch access to data or in a streaming setting where data arrives incrementally. Coresets satisfy the following composition properties.

- ***Merge***. *If C_1 is an (εr)-coreset for P_1, and C_2 is a (εr)-coreset for P_2. Then $C_1 \cup C_2$ is an (εr)-coreset for $P_1 \cup P_2$.*
- ***Reduce***. *If C is an (εr)-coreset for P, and D is an (εr)-coreset for C. Then D is a $(2\varepsilon r)$-coreset for P.*

The *merge* operation leads to an increase in the coreset size but does not increase the approximation error. The *reduce* operation does increase the error but yields a resultant coreset of a smaller size. A naive reduction strategy can cause and exponential increase in the approximation error. However, this is prevented by organising the *merge* and *reduce* operations as a binary tree of height $O(\log n)$ where each level contains coresets for the previous level of coresets. Using the compositional properties and the reduction explored earlier in [10], [11], [20] the following theorem can be obtained.


C_7
C_3
C_6
C_1
C_2
C_4
C_5
i_1 ···· i_{10} ···· i_{20} ···· i_{30} ···· i_{40}

(a) Tree of coresets



(b) Incremental construction

Fig. 4: **Batch and online construction of coreset trees.** Blue arrows indicate *merge-and-reduce*. Image sequence (red arrow) is chunked according to a specified leaf-size. (a) Batch operation C_1, C_2, C_4 and C_5 are constructed in parallel, followed by C_3 and C_6, finally resulting in C_7. (b) Incremental data stream: the (intermediate) coresets $C_1, \ldots, C_7$ are enumerated in the order of construction, only $O(logn)$ coresets are retained in memory (gray).

Theorem 5.1: An (εr)-coreset C of size $O(k \log^3 n/\varepsilon^d)$ for a set of n points in a stream of images can be computed using $O(k \log^d n/\varepsilon^d)$ memory and update time per point insertion.

We use the parallel coreset construction property to generate multi-resolution summaries of a data set (that may be stored distributed) where batch access is possible. Further, the streaming construction allows incremental visual summary generation in the scenario where memory is bounded and not all data points can be stored. This is presented next.

A. Multi-resolution Summaries with Parallel Coresets

In the batch setting, the data is partitioned into sets and coresets are constructed for each set independently. We then *merge* in parallel the two coresets, and compute a single coreset for every pair of such coresets. Continuing in this manner yields a process that takes $O(\log n)$ iterations of parallel computation, Fig. 4. Applying the star clustering algorithm to the coreset at each level in the constructed tree leads to a hierarchical summarization for the entire data set. We assume that the coreset-tree is saved in memory or disk in the batch setting. The user can browse the summary generated (or video) in a multi-resolution manner specified as a time interval or equivalently as an error bound. For example, the root of the tree represents the semantic compression of all the captured images. The left child of the root contains summary of images for the first half of the time interval, while the right child contains summary for the second half. The user can thus zoom in/out time intervals to focus on the relevant compression via browsing the coresets tree organized as star clusters.

B. Bounded-memory Summaries using Streaming Coresets

In the streaming setting, an incremental summary is generated keeping only a small subset of $O(\log n)$ coresets in memory (each of a small size). Each data element received is inserted into the star clustering. Coresets are constructed and saved in memory for each data block. Two coresets in memory, we can *merge* them (resulting in an εr-coreset) and *reduce* them by computing a single coreset from the merged coresets to avoid an increase in the coreset size. In effect, points in the merged set, not part of the coreset can be deleted from the star clustering graph while retaining the representative coreset points in the graph. See Fig. 4 for the online construction process. Note that star clustering supports efficient *insert* and *delete* operations asymptotically scaling $O(n^2 \log^2 n)$ for a sequence of n operations.

C. Relation with other approaches

For simplicity, consider the simple distance function where each point represents the pixels in the image and their colour intensities. For example, in a video stream of 30 frames per second there will be several images that are similar in this sense. Instead of uniform sampling, our compression expects to return more images when the camera moves at high speed and less images when the camera stands still. Generalisations, distance functions and experiments of compression using more involved vision-based feature spaces can be found in [6]. Another limitation is that the k-center problem deals with points in space, not over time. However, our streaming tree produces a coreset of different resolutions over time. Another approach is to use the time dimension is to add it as additional coordinate to each point, with an appropriate weight, so images that related to similar time stamps will have smaller distance. However, the appropriate weighting factor is still to be obtained.

VI. Results

A. Data sets and Pre-processing

A data set traversing streets and park areas was collected consisting of 2874 images, recorded 10m apart and perpendicular to the robot's motion. Image samples were non-overlapping, excluded loop closures pairs and hence approximated independent samples from the observation distribution used for vocabulary and topic learning. A visual vocabulary [12] of approximately 11k visual words was generated by clustering SURF features [21] extracted from this data set. Topic distributions were estimated using a Gibbs sampling [14]. The Markov chain was randomly intialized and was run till convergence to the target distribution using data log-likelihood as a measure. The appropriate number of topics was selected by maximizing the data log-likelihood for the corpus given topics, $P(\mathbf{w}|T)$ which was found maximum using 50 topics [14] for the data set. Each image was converted to a vector of visual words representation by first extracting SURF features and then quantizing against the learnt vocabulary. This was followed by estimating topic proportions mapping to a topic space.

The visual summarization system was tested on two public data sets [1]. The New College data set consists of 1355 images from a 2.1km traversal within a typical college in Oxford consisting of medieval buildings and parkland areas. The second data set City Centre data set is 2km in length, possessing 1683 images from parks and city roads. These data sets did not overlap with the Urban data set used for learning topic distributions.

B. Pre-clustering using Coresets

An extracted coreset induces a pre-clustering for the original data set with clusters obtained by associating each data point to the nearest element in the coreset. Figure 5 visualizes *three* representative clusters from the New College data set induced via a coreset of size 497 (37% of the original data) using parameters $\beta = 0.5$ and $\epsilon = 0.5$. The coreset element (indicated in *red*) is shown first, followed by the associated data set images in decreasing order of similarity. The clustered views display a high degree of similarity indicating an effective compression and redundancy removal using coresets.

C. Visual Summaries using Coreset Trees

Figure 1 visualizes summaries obtained using a binary coreset tree constructed for the New College data set with a leaf size of 300. Three levels are displayed. The summary images are obtained at a similarity threshold of $\sigma = 0.6$. Examining the lowest level (from *left* to *right*), the initial summaries consist of representative images from the cloisters area leading into the quad section. Later summaries contain images from the parklands, car park before returning to the cloisters area. Summary at the parent level is computed by a merge-reduce operation on the coresets at the lower level, representing scenes from a larger traversal sequence. Note that this allows multi-resolution summaries in the batch setting and memory efficient summaries in the incremental case. Figure 6 presents the visual summary obtained on the coreset and on the entire data set. Note that the summaries are very similar, each possessing representative images from the traversal including cloisters, quad and parkland areas, capturing different modes of visual appearance of the workspace.

D. Approximation Error for Coreset-based Summaries

Next, we empirically compare the performance of the star clustering algorithm when applied to the coreset and the entire data set. For comparison, we define k-fraction error metric computed as follows:

1) Let C be the coreset computed for point set P. Given the star clustering similarity threshold σ, run the star clustering on C resulting in k_c star centers.
2) Determine the minimum distance threshold r_c such that balls of radius $\frac{r_c}{(1-\epsilon)}$ can cover set P using k_c centers.
3) Apply the star clustering on P using similarity threshold σ_c corresponding to distance r_c to obtain k' centers. The k-fraction is defined as $(\frac{k_c}{k'})$.

Intuitively, for a given threshold σ, the k-fraction metric expresses the relative number of centers that can cover coreset C compared to point set P using an equivalent threshold σ_c. This is based on the fact that the algorithm output on the coreset is guaranteed to be a $(1 \pm \varepsilon)$-approximation to the equivalent result on the full data set.

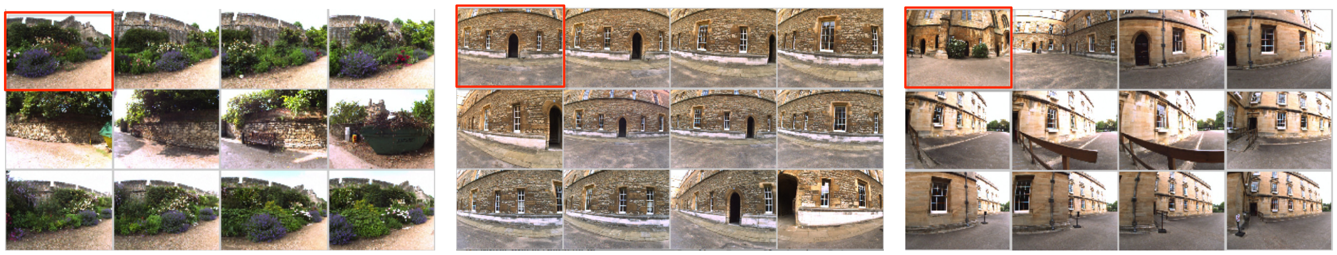


Fig. 5: **What do coresets cluster?** Visualisation of coreset clusters obtained by assigning data set points to the nearest element in the coreset. Coreset point is indicated in red followed by associated data set images arranged in decreasing order of similarity. Highly similar views are clustered together indicating that the coreset is effective in removing redundancy in the data. New College data set. Coreset size 497 (37% of original data) with $\beta = 0.5$ and $\epsilon = 0.5$.

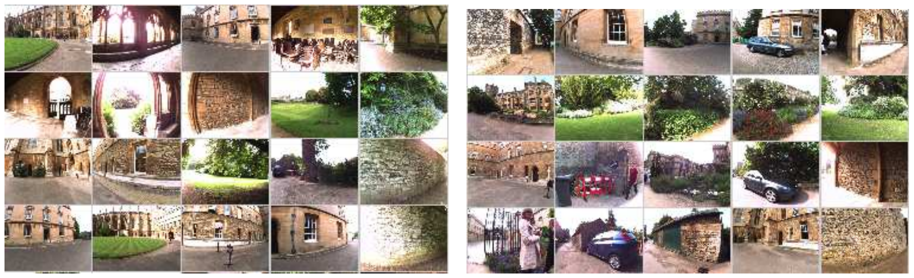

(a) Using coreset (497 images) (b) Using full data set (1355 images)

Fig. 6: **Comparing visual summaries.** Summary obtained at the end of the traversal using (a) coreset (497 images, $\beta = 0.5$ and $\epsilon = 0.5$) and (b) on the full data set (1355 images) for the New College data set. Note the two summaries contain very similar views, each capturing images from cloisters, quad and parkland areas, representing different modes of visual appearance. Images were picked randomly from the final summary set for display in this figure.

Figure 8(a-b) plots the k-fraction error for varying coreset sizes for similarity threshold σ ranging from 0.4 till 0.9 for the City Center data set, using the coresets obtained by varying the parameters as discussed above. The k-fraction error plots are observed to lie below *one* for a majority of the values, except for few runs with $\sigma = 0.9$ where the error is marginally higher, indicating that the star clustering algorithm on the coreset is a good approximation to the resulting output on the full data set.

Note that coresets often yield *better* results (k-fraction error below 1) while running heuristic algorithms like star clustering. Intuitively, coresets smooth data, removing noise, and reduce the number of bad local minima. Further, low error values are observed even for relatively small coreset sizes. A worse error is observed for runs with a higher similarity threshold. Using a higher similarity threshold yields more numerous clusters (with higher intra-cluster similarity) inducing a cover with balls with a smaller radii. Consequently, compression using coresets is less effective in this scenario as extra centers are required to cover the data points with a smaller radius induced by the higher similarity threshold. The result for the New College data set were found to be similar.

E. Coreset Size

The size of the resulting coreset depends on the length parameter, ϵ for the ϵ-grid and the removal fraction of points in each iteration during the construction process, Algorithm Eps-Coreset(P, k, ε). Figure 7 plots the coreset sizes obtained by varying parameters logarithmically as $\{0, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, 1\}$. Using a finer ϵ-grid or removing a higher removal fraction of points in each iteration leads to a larger coreset size.

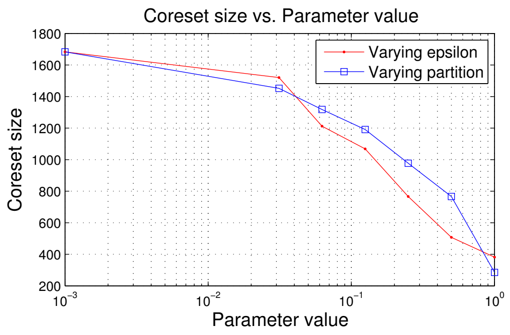


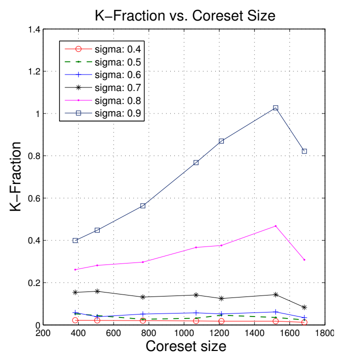
Fig. 7: **Coreset size.** Varying ϵ-grid size and removal fraction (partition) per iteration logarithmically as $\{0, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, 1\}$ for the City Centre data set. Coreset size decreases with a coarser ϵ-grid or a higher removal fraction

F. Runtime Efficiency

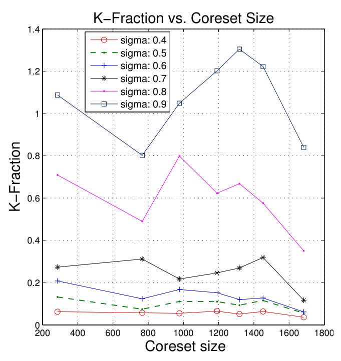
Figures 8(c-d) plot the fraction of the time taken to run the summarisation algorithm on the coreset compared to the full data set. For a majority of the runs a significant speed-up is observed. However, for few runs with higher similarity threshold the run time was found to be greater. This can be attributed to a large number of stars re-arrangements due to the sequence in which the coreset points are incrementally organised. Note that we used the incremental version of the star clustering algorithm in our experiments.

VII. Conclusions

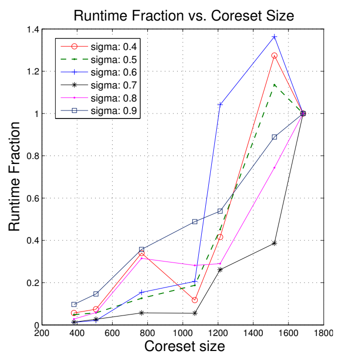
In this paper we described a scalable approach to generating visual summaries from streams of images acquired by long-duration traversal of a mobile robot. Current methods do not scale well for natural traversals particularly with high redundancy and perceptual similarity in image data. Our approach identifies a sequence of images that thematically summarize the robot's visual experience. Our solution has



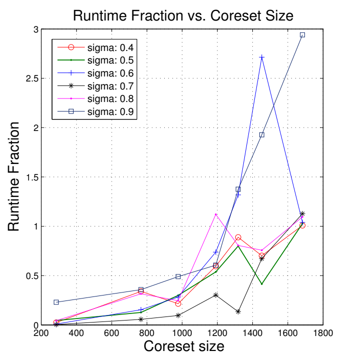
(a) Varying epsilon



(b) Varying removal fraction



(c) Varying epsilon



(d) Varying removal fraction

Fig. 8: ***k*-fraction coreset error for star clustering. (a-b)** The error lies below *one* for majority of values except for some runs for $\sigma = 0.9$ where a marginally higher error is observed. Even relatively *small* coresets perform well with low error values indicating a good compression of the full set. Worse errors are seen for higher thresholds where numerous clusters are obtained. **Timing efficiency. (c-d)** Fractional running time of the summarization algorithm on the coreset compared to the full data set. Significant speed-up is observed for a majority of runs. For few runs with higher similarity threshold show increased running time, attributable to large number of stars re-arrangements dependent on the sequence in which coreset points appear. Experiments used the City Centre data set. Coresets were obtained by varying ϵ-grid size and removal fraction (partition) per iteration logarithmically as $\{0, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, 1\}$.

two algorithmic components. We demonstrate how coresets of the image stream greatly reduce the amount of data to be processed in order to generate such summaries and provide a method for choosing the "right data" from the large data stream with provable performance guarantees. By running the star clustering algorithm on the image coreset tree we generate a hierarchy of visual summaries for the robot's entire visual experience facilitating multi-resolution viewing or video synthesis or a desired duration. In the streaming setting, the method allows summaries with bounded memory using a logarithmic number of coresets. Our experiments indicate that the coreset-star-clustering approach to visual summaries is practical, enabling a new capability for robots.

VIII. Acknowledgements

This project was supported in part by the Future Urban Mobility project, Singapore-MIT Alliance for Research and Technology (SMART) Center, with funding from Singapore's National Research Science Foundation, by the Foxconn Company, by ONR MURI Grants N00014-09-1-1051 and N00014-09-1-1031, and by NSF award IIS-1117178. Paul Newman was supported by an EPSRC Leadership Fellowship, EPSRC Grant EP/I005021/1.

References

[1] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[2] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, 2009.

[3] R. Paul, D. Rus, and P. Newman, "How was your day? online visual workspace summaries using incremental clustering in topic space," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4058–4065.

[4] Y. Girdhar and G. Dudek, "Online navigation summaries," in *IEEE International Conference on Robotics and Automation (ICRA), 2010*. IEEE, 2010, pp. 5035–5040.

[5] A. Ranganathan and F. Dellaert, "Bayesian surprise and landmark detection," in *IEEE International Conference on Robotics and Automation (ICRA), 2009*. IEEE, 2009, pp. 2017–2023.

[6] G. Yogesh and G. Dudek, "Efficient on-line data summarization using extremum summaries," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3490–3496.

[7] D. S. Hochbaum, "Easy solutions for the k-center problem or the dominating set problem on random graphs," in *Analysis and Design of Algorithms for Combinatorial Problems*. North-Holland, 1985, vol. 109, pp. 189 – 209.

[8] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[9] P. Agarwal, S. Har-Peled, and K. Varadarajan, "Geometric approximation via coresets," *Discrete and Computational Geometry*, vol. 52, 2005.

[10] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data." in *Proc. 41th Ann. ACM Symp. on Theory of Computing (STOC)*, 2011.

[11] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *Proc. 36th Ann. ACM Symp. on Theory of Computing (STOC)*, 2004, pp. 291–300.

[12] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1470–1477.

[13] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[14] T. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences*, vol. 101, no. Suppl 1, p. 5228, Jan 2004.

[15] J. Aslam, E. Pelekhov, and D. Rus, "The star clustering algorithm for static and dynamic information organization," *Journal of Graph Algorithms and Applications*, vol. 8, no. 1, pp. 95–129, 2004.

[16] M. Cummins and P. Newman, "Highly scalable appearance-only slam-fab-map 2.0," in *Proceedings of Robotics: Science and Systems (RSS)*, vol. 5, 2009.

[17] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy *et al.*, "Navigating, recognizing and describing urban spaces with vision and lasers," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1406–1433, 2009.

[18] P. Agarwal and R. Sharathkumar, "Streaming algorithms for extent problems in high dimensions," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2010, pp. 1481–1489.

[19] P. Agarwal, S. Har-Peled, and K. Varadarajan, "Geometric approximation via coresets," *Combinatorial and computational geometry*, vol. 52, pp. 1–30, 2005.

[20] D. Feldman, M. Schmidt, and C. Sohler, "Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering," *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

[21] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," vol. 13, May 2006, pp. 404–417.