Video Segmentation and Summarization for Persistent Robot Systems using Coresets

Author Names Omitted for Anonymous Review. Paper-ID [add your ID here]

***Abstract*—Information extraction from persistent robot systems that operate for long periods of time and collect vast amounts of data can enable new autonomous capabilities for robots. The collected data is valuable for mapping, situation awareness, and modeling behaviors, but requires efficient tools that can extract the right information at the right time efficiently. In this paper we propose a novel algorithm that provides for autonomous efficient and scalable segmentation of video streams. Our proposed framework allows fast segmentation cost computation as well as efficient computation of an approximately-optimal segmentation. We describe the proposed algorithm, demonstrate its theoretical and practical properties, and present experimental results of multiscale analysis of large streams of video sequences for mobile cameras.**

I. Introduction

We wish to create systems that benefit from long-term visual data collection. Applications include: robots that can quickly compare their current images against the image content of their past history to determine if they had visited a place, map completion, check whether anything changed since their last observation, summarize their collective visual experiences, identify activity patterns, etc.

Enabling such capabilities over visual data requires new algorithms for on-line segmentation and summarization of visual data streams. Currently, robots that operate for long periods of time and collect visual streams can not always take advantage of this data. One of the reasons is that the current algorithms for visual data segmentation and summarization can handle only short videos. Large video streams are usually handled by running local optimization on only small parts of the video.

In this paper we use a data reduction technique known as *coresets* [1, 7] to enable rapid content-base segmentation of visual data streams. A coreset (or core-set) C is problem dependent compression of the original data D, so that running algorithm A on the compression C yields a result $A(C)$ that provably approximates the result $A(D)$ of running the algorithm on the original data. If the coreset C is small and its construction is fast, then computing $A(C)$ is fast even if computing the result $A(D)$ on the original data is intractable.

We present a new coreset construction that provides provable approximations for the natural k-segmentation optimization problem. The running time and required memory of our algorithm is linear in both the number of observations n, their dimensionality d, and the number k of desired segments. Previous results have running time that is at least quadratic in d and cubic in k, and thus cannot be applied on long streaming video data which is usually high-dimensional and contain large number of scenes.

In this paper we use a coreset approach to enable rapid content-base segmentation of visual data streams. Coresets [7] allow efficient data extraction using a biased subsampling approach.

Coresets have been defined for a number of problems [7]. In this paper we develop a new coreset for segmentation. Unlike previous coresets, this coreset allows not just cost computation for data and video summarization, but also efficient segmentation with strong guarantees. The proposed coreset allows us to perform efficient and scalable segmentation of data streams of high dimensionality (including video over feature space). Combining it with video processing and analysis techniques, we are able to get video segmentation that is semantically-driven and efficient, with an objective function that relates directly to methods from video activity analysis, scene understanding, and place recognition.

A. Main Contribution

The main contributions of the paper are: (i) We demonstrate a new system for segmentation and compression of video data into a small number of scenes. Our approach allows summarization of the video streams in a way that preserves the semantic content of the aggregated video sequences, and is easily extendable. (ii) We propose a new provable approximation for the k-segmentation problem of d-dimensional signals. The new scheme allows fast segmentation over one pass over streaming data and support distributed computation. Unlike previous results, the insertion time per new observation and required memory is only linear in both the dimension of the data, and the number k of segments. The approximation is based on a compression scheme, known as coresets that can also be used to solve many variants of the problem; (iii) We conduct full-scale experiments to evaluate our system with respect to output size, running time and quality and compare our coresets to uniform and random sample compression. Our algorithm is both scalable and parallelizable, allowing us to deploy our system on Amazon cluster comprised of 255 machines to process a video stream of 256,000 frames, running at 9 times real-time rate, while have a provable approximation of the cost function. We show that the resulting segmentation yields a meaningful activity-based video summarization.

B. Related work

We build upon a multitude of prior works on efficiently gathering visual information from multiple sources for activity

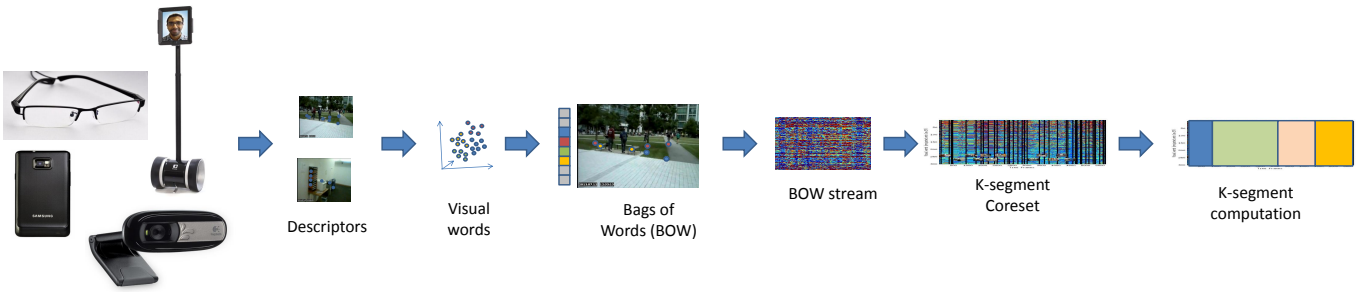


Fig. 1: System overview: a) Descriptor extraction. b) Bags of words computation for each frame. c) Add the new bag of words into a k-segment mean coreset. d) Compute k-segmentation over the coreset.

understanding of both robotic systems [6] and human users [3]. Sample applications of video stream analysis include mapping and navigation [19, 26, 15, 31], medical / assistive interaction [18, 5], monitoring robotic systems, and augmented-reality applications. Compressing the video is suboptimal for such tasks since compression is geared towards preserving video quality for all frames, and therefore stores semantically redundant content. We are looking instead for a summarization approach that allows us to represent the video content by a set of key segments. Application-aware summarization and analysis of ad-hoc video streams is a difficult task with many attempts aimed at tackling it from various perspectives [23, 3]. The problem is highly related to video action classification [32, 33, 27, 20], scene classification (see for example [34]), and object segmentation [23], among other topics. Significant attention has been given to low-level feature extraction from video sequences for these vision tasks (see [33] for example). These vary from simple measures based on brightness-, gradient- and optical flow-based descriptors to quite elaborate descriptions of scene structure and content. Maekawa et al. [24] used an HSV color codebook in order to represent activities from a wrist-based camera. Bandla and Grauman [3] use spatio-temporal features based on optical flow and HoG freatures, and Koppula et al. [21] combine descriptors and local transformation cues. Lu and Grauman [23] define the relative strength of inter-frame connections based on object cooccurrence. Directly learning features has also been attempted by deep learning techniques, see for example [2].

Developing scalable analysis schemes for summarizing large quantities of visual data is still lacking, especially as far as provably efficient and accurate algorithms are concerned. In this paper we propose a new scheme for compression of these data streams, with well-founded scalability properties, allowing fast near-optimal segmentation with strong error bounds.

Approximation Algorithms. One of the main challenges is providing provable guarantees for the segmentation size and quality, is that it requires global optimization. Most heuristics suggest ad-hoc local optimization of the streaming data. The closest recent works to ours involve algorithms that provide provable approximations are [25] and [9].

The result in [25] is the last in a line of research for video summarization in the context of robotics that translates each frame into a point and run approximated clustering algorithms such as k-center on the points; see [13, 14] for surveys. The resulting k centers of the clusters are the video summarization. The main disadvantage of these techniques is that (i) they partition the set of images into k clusters that do not provide k-segmentation over time. (ii) Computing the k-center takes time exponential in both d and k [17]. In [25] heuristics were used for dimension reduction, and in [14] a 2-approximation was suggested for the off-line case, which was replaced by a heuristic for the streaming case. (iii) We utilize a descriptor-based feature space that is more robust to nuisance factors such as illumination and viewpoint changes compare to the pixel (color) space that is used in [25]. Unlike previous result, this space can be updated on-line and is learned from the video using coreset for k-means clustering of the features seen so far.

k-segment mean. The k-segment mean problem can be solved exactly using dynamic programming [4]. However, this takes $O(dn^2k)$ time and $O(dn^2)$ memory, which is impractical. In [16, Theorem 8] a $(1+\varepsilon)$-approximation was suggested using $O(n(dk)^4 \log n/\varepsilon)$ time. In [9] an improved algorithm that takes $O(nd^2k+ndk^3)$ time was suggested. The algorithm is based on a coreset of size $O(dk^3/\varepsilon^3)$. Unlike the coreset in this paper, the running time of their coreset is super-linear in both d and k, and its size is linear in n.

The result in [9] is the last in a line of research for the k-segment mean problem and its variations; see survey in [8, 16, 12]. The application was segmentation of 3-dimensional GPS signal (time, latitude, longitude). The coreset construction in [9] and previous papers takes time and memory that is quadratic in the dimension d and cubic in the number of segments k. Our coreset construction takes time only linear in both k and d.

While the recent results suggest running time that is linear in n, and space that is near-logarithmic in n, the computation time is still cubic in k, the number of segments, and quadratic in d, the dimension. Since the number k represents the number of scenes, and d is the total number of possible features, such a running time is prohibitive.

We now turn to describe our algorithm in detail. We begin by giving an overview of our video segmentation pipeline in Section II. We proceed to describe the k-segmentation problem and the proposed coresets, their construction, and their

properties in Section III. We perform several experiments in order to validate the computational requirements of our system and demonstrate the aggregation and analysis of multiple long sequences of wearable user video in Section IV. Section V concludes the paper and briefly discusses future directions.

II. Video-Processing Pipeline

We now describe our system for video streams summarization and compression. The basis for video segmentation and summarization is content comparison across video frames. Different choices of distance measures between frames have been attempted in the context of activity understanding, scene classification, and localization. This usually involves either a vector space and standard metric, as in [29], or using a more elaborate definition of an interframe distance measure, as in [22, 23]. This dissimilarity measure is subjective and depends on our own definition of activities and segments in the video, and it may involve user interaction or input.

We chose as our vector space and dissimilarity measure color-augmented SURF features, and a weighted L_2 norm learned from a handful (~ 12) of user annotations from the beginning of the video stream. In a robot setting, such a small number of learning examples can be pre-trained, or selected by the system from high-confidence sequences in a self-reinforcing manner. The feature vectors are obtained by a running a SURF detector on the intensity image. The SURF descriptors of the intensity image, along with the saturation and hue channels of the image (a 66-dimensional space) are binned into 5000 bins. The data used to perform the *vector quantization* (VQ) is taken from the first $240,000$ images in the ILSVRC 2013 dataset. This dataset is chosen since it is standard, and because it differs sufficiently from our data (in the device type, resolution and object types) so as to preclude overfitting. In order to compute VQ on a large set of high dimensional vectors we use a coreset proposed in [7] and later in [10]. We gather about 3.5×10^8 vectors from the data by an online streaming coreset for k-means representation. The coreset contains $40,000$ weighted points which are then used to run a weighted k-means clustering. Most of the time is taken by image loading and descriptor computation, with k-means taking a few minutes on an Intel $i7$ CPU.

Given this representation of the frames, some notion of interframe dissimilarity must still be defined. We still expect the introduction of user-based context to be done in a straightforward and stable manner. In order to keep the geometric and extendable nature of the coreset framework, we take a metric learning approach, motivated by [28, 30], which provides a simple and elegant solution, while allowing domain-specific adaptation. The resulting descriptor histograms are subject to a linear transformation. The transformation matrix is

$$W = (\Sigma + \varepsilon I)^{\dagger}$$

The resulting vectors are filtered in an online manner, by a sliding window median filter, followed by a sliding window averaging filter. The resulting filtered signals are fed into an online coreset construction, as described in Section III. We then can perform segmentation using dynamic programming [4], or use the segments of D as an over-segmentation of the video, similar to superpixels in spatial image segmentation. The overall algorithm is visually shown in Figure 1.

Feeding these vectors into a scalable k-means clustering algorithm such as [25] will allow us to search for similar and/or repeating scenes in the video sequence. In this paper, since we focus on video summarization, we do not expand upon the relation between the two problems which is future work.

III. Coreset Construction for k-segment mean

We now describe the optimization problem and its proposed coreset, which is motivated by video segmentation.

A. Model assumptions

A streaming digital video can be represented by a set of points over time in an appropriate feature space, where every point in the feature space represents a frame or a small set of frames, in terms of the image content. The dimensionality of this space is often quite large (from tens to thousands of features), and the specific choice of feature space is application dependent – several such choices are described in Section II. The assumptions of our model are: (a) Each of the n input images is represented well by its d-dimensional features vector; (b) The content of the video includes at most k scenes that we wish to detect automatically; and (c) The images in each scene consist of similar "bag of features", in the sense that their corresponding feature vectors can be approximated well by a line, compared to images outside of the scene. This motivates the following problem definition.

B. Problem Statement

The k-segment mean problem optimally fits a given discrete time signal of n points by a set of k linear segments over time, where $k \geq 1$ is a given integer. That is, we wish to partition the signal into k consecutive time intervals such that the points in each time interval are lying on a single line; see Fig. 2(left) and the following formal definition.

Definition 1 (k-segment mean)**.** *A set P in $\mathbb{R}^{d+1}$ is a* signal *if $P = \{(1, p_1), (2, p_2), \cdots, (n, p_n)\}$ where $p_i \in \mathbb{R}^d$ is the point at time stamp i for every $i = [n] = \{1, \cdots, n\}$. For an integer $k \geq 1$, a* k-segment *is a k-piecewise linear function $f : \mathbb{R} \to \mathbb{R}^d$ that maps every time $i \in \mathbb{R}$ to a point $f(i)$ in $\mathbb{R}^d$. Hence, f represents k-segments over time. The* fitting error *at time t is the squared distance between p_i and its corresponding projected point $f(i)$ on the k-segments. The* fitting cost *of f to P is the sum of these squared distances,*

$$\text{cost}(P, f) = \sum_{i=1}^{n} \|p_i - f(i)\|_2^2, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean distance. The function f is a k-segment mean *of P if it minimizes* $\text{cost}(P, f)$.

For the case $k = 1$ the 1-segment mean is the solution to the linear regression problem. If we restrict each of the k-segments to be a horizontal segment, then each segment will

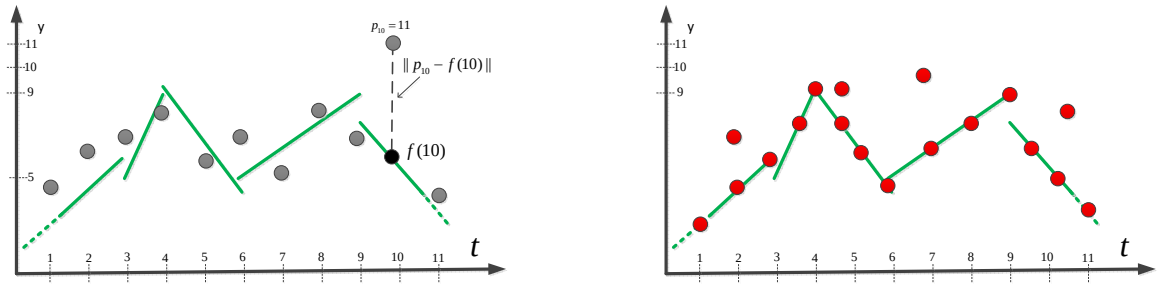


Fig. 2: For every k-segment f, the cost of input points (grey points) is approximated by the cost of the coreset (red points). Left: An input signal (grey points) and a 4-segment f (green). The regression distance from the 10th point $(t, p_{10}) = (10, 11)$ to its projection $f(t) = f(10)$ is $\|p_{10} - f(10)\| = |11 - 5| = 6$. The cost of f is the sum of these squared distances from all the input points. Right: The coreset (red points) consists of the projection of the input points onto few segments, and additional few points outside these segments.

be the mean height of the corresponding input points. The resulting problem is similar to the k-mean problem, except each of the voronoi cells is force to be a single region in time, instead of nearest center assignment.

C. 1-Segment Coreset

A $(1, \varepsilon)$-coreset approximates $\text{cost}(P, f)$ for every 1-segment f up to a factor of $1 \pm \varepsilon$ as defined below. It will be the main ingredient for constructing coresets for k-segments.

Definition 2 ($(1, \varepsilon)$-coreset)**.** *Let P and C be two sets in $\mathbb{R}^{d+1}$ and let $\varepsilon, w > 0$. The pair (C, w) is a* $(1, \varepsilon)$-coreset *for P, if for every 1-segment $f : \mathbb{R} \to \mathbb{R}^d$ we have*

$$(1 - \varepsilon)\text{cost}(P, f) \leq w \cdot \text{cost}(C, f) \leq (1 + \varepsilon)\text{cost}(P, f).$$

For example, (P, w) is a $(1, \varepsilon)$-coreset for P with $\varepsilon = 0$ and $w = 1$. However, a coreset is efficient if its size $|C|$ is much smaller than P.

It is easy to compute a $(1, 0)$-coreset of $d + 1$ points in $O(nd^2)$ time using SVD, as shown in the supplementary. This and running time might be too large, for example when d is in the order of n, or we are dealing with high dimensional space such as our videos, when $d \gg 1/\varepsilon$. For the rest of the paper the construction of $(1, \varepsilon)$-coresets suffices. Using recent results from [10] and [11], we construct in $O(nd/\varepsilon^4)$ time $(1, \varepsilon)$-coreset that consists of only $O(1/\varepsilon^2)$ points .

Theorem 3. *Let $P \subseteq \mathbb{R}^{d+1}$ and let $\varepsilon > 0$. A $(1, \varepsilon)$-coreset $C \subseteq \mathbb{R}^{d+1}$ for P of size $|C| = O(1/\varepsilon^2)$ can be computed in $O(nd/\varepsilon^4)$ time.*

The proof of Theorem 3 is given in the supplementary material.

D. k-Segment Coreset

We would like to compute a (k, ε)-coreset for our data. A (k, ε)-coreset D for a set P approximates the fitting cost of a any query k-segment to P up to a small multiplicative error of $1 \pm \varepsilon$. However, for a k-segment, $k > 2$, we cannot choose such a representative point set (we prove this in the supplementary material).

Instead, we define a more involved data structure D that represents the coreset, and define a new cost function $\text{cost}'(D, f)$ that approximates the cost of P to a k-segment f.

The set D consists of tuples of the type (C, g, b, e). Each tuple corresponds to a different time interval $[b, e]$ in $\mathbb{R}$ and represents the set $P(b, e)$ of points of P in this interval. The set C is a $(1, \varepsilon)$-coreset for $P(b, e)$. Our first observation is that if all the points of the k-segment f are on the same segment in this time interval, i.e, $\{f(t) \mid b \leq t \leq e\}$ is a linear segment, then the cost from $P(b, e)$ to f can be approximated well by C, up to $(1 + \varepsilon)$ multiplicative error.

The second observation is that if we project the points of $P(b, e)$ on their 1-segment mean g, then the projected set L of points will approximate well the cost of $P(b, e)$ to f, even if f corresponds to more than one segment in the time interval $[b, e]$. Unlike the previous case, the error here is additive. However, the third observation is that, since f is a k-segment there will be at most $k - 1$ time intervals $[b, e]$ that will intersects more than two segments of f, so the overall additive error is small . This motivates the following definition of D and cost'.

Definition 4 ($\text{cost}'(D, f)$)**.** *Let $D = \{(C_i, g_i, b_i, e_i)\}_{i=1}^m$ where for every $i \in [m]$ we have $C_i \subseteq \mathbb{R}^{d+1}$, $g_i : \mathbb{R} \to \mathbb{R}^d$ and $b_i \leq e_i \in \mathbb{R}$. For a k-segment $f : \mathbb{R} \to \mathbb{R}^d$ and $i \in [m]$ we say that C_i is* served by one segment *of f if $\{f(t) \mid b_i \leq t \leq e_i\}$ is a linear segment. We denote by $\text{Good}(D, f) \subseteq [m]$ the union of indexes i such that C_i is served by one segment of f. We also define $L_i = \{g_i(t) \mid b_i \leq t \leq e_i\}$, the projection of C_i on g_i.*

We define $\text{cost}'(D, f)$ to be

$$\text{cost}'(D, f) = \sum_{i \in \text{Good}(D,f)} \text{cost}(C_i, f) + \sum_{i \in [m] \setminus \text{Good}(D,f)} \text{cost}(L_i, f).$$

We will compute such a small structure D that approximates $\text{cost}(P, f)$ for every k-segment f using the above definition of $\text{cost}'(D, f)$. Such a set D will be called a (k, ε)-coreset as follows.

Algorithm 1: BALANCEDPARTITION(P, ε, σ)

Input: A set $P = \{(1, p_1), \cdots, (n, p_n)\}$ in $\mathbb{R}^{d+1}$ an error parameters $\varepsilon \in (0, 1/10)$ and $\sigma > 0$.
Output: A set D that satisfies Lemma 6.

$Q := \emptyset$; $D = \emptyset$;
p_{n+1}:= an arbitrary point in $\mathbb{R}^d$;
for $i := 1$ *to* $n + 1$ **do**
  $Q := Q \cup \{(i, p_i)\}$
  $f^* :=$ a linear approximation of Q
  $\lambda := \text{cost}(Q, f^*)$
  if $\lambda > \sigma$ **or** $i = n + 1$ **then**
    $T := Q \setminus \{(i, p_i)\}$
    $C :=$ a $(1, \varepsilon/4)$-coreset for T
    $g :=$ a linear approximation of T
    $b := i - |T|$
    $e := i - 1$
    $D := D \cup \{(C, g, b, e)\}$
    $Q := \{(i, p_i)\}$
return D

Definition 5 ((k, ε)-coreset)**.** *Let $P \subseteq \mathbb{R}^{d+1}$, $k \geq 1$ be an integer, for $\varepsilon > 0$ a small number. A set D, with a cost $\text{cost}'(\cdot)$ is a (k, ε)-*coreset *for P if for every k-segment f we have*

$$(1 - \varepsilon)\text{cost}(P, f) \leq \text{cost}'(D, f) \leq (1 + \varepsilon)\text{cost}(P, f).$$

Our coreset construction is based on an input parameter $\sigma > 0$ such that for an appropriate σ the output is a (k, ε)-coreset. Recall that for $\alpha, \beta > 0$, an (α, β)-approximation for the k-segment mean of P is a $(k \cdot \beta)$-segment g such that $\text{cost}(P, g) \leq \alpha \cdot \text{cost}(P, f^*)$. We show that using the value $\text{cost}(P, g)$ of such an approximation, even without knowing g, suffices to get a (k, ε)-coreset. In the next section we will compute such an (α, β)-approximation for small α and β.

The size of the resulting coreset depends on α and β. In particular, for $\alpha = \beta = 1$ the following lemma implies that there exists a (k, ε)-coreset of size $O(k/\varepsilon^2)$ for every input set P.

Lemma 6. *Let $P = \{(1, p_1), \cdots, (n, p_n)\}$ such that $p_i \in \mathbb{R}^d$ for every $i \in [n]$. Suppose that $h : \mathbb{R} \to \mathbb{R}^d$ is an (α, β)-approximation for the k-segment mean of P, and let σ be*

$$\sigma = \frac{\varepsilon^2 \text{cost}(P, h)}{100k\alpha}.$$

Let D be the output of a call to BALANCEDPARTITION(P, ε, σ)*; see Algorithm 1. Then D is a (k, ε)-coreset for P of size*

$$|D| = O(k) \cdot \left(\frac{\alpha}{\varepsilon^2} + \beta\right),$$

and can be computed in $O(dn/\varepsilon^4)$ time.

Proof: Let $m = |D|$ and f be a k-segment. We denote the ith tuple in D by (C, g_i, b_i, e_i) for every $i \in [m]$. For every $i \in [m]$ we have that C_i is a $(1, \varepsilon/4)$-coreset for a corresponding subset $T = T_i$ of P. By the construction of D we also have $P = T_1 \cup \cdots \cup T_m$.

Using Definition 4 of $\text{cost}'(D, f)$, $\text{Good}(D, f)$ and L_i, we thus have

$$\begin{aligned}
&|\text{cost}(P, f) - \text{cost}'(D, f)| \\
&= |\sum_{i=1}^{m} \text{cost}(T_i, f) - \Big(\sum_{i \in \text{Good}(D,f)} \text{cost}(C_i, f) \\
&+ \sum_{i \in [m] \setminus \text{Good}(D,f)} \text{cost}(L_i, f)\Big)| \\
&= |\sum_{i \in \text{Good}(D,f)} (\text{cost}(T_i, f) - \text{cost}(C_i, f)) + \\
&\sum_{i \in [m] \setminus \text{Good}(D,f)} (\text{cost}(T_i, f) - \text{cost}(L_i, f)) \\
&\leq \sum_{i \in \text{Good}(D,f)} |\text{cost}(T_i, f) - \text{cost}(C_i, f)| + \\
&\sum_{i \in [m] \setminus \text{Good}(D,f)} |\text{cost}(T_i, f) - \text{cost}(L_i, f)|,
\end{aligned} \tag{2}$$

where the last inequality is due to the triangle inequality. We now bound each term in the right hand side.

For every $i \in \text{Good}(D, f)$ we have that C_i is a $(1, \varepsilon/4)$-coreset for T_i, so

$$|\text{cost}(T_i, f) - \text{cost}(C_i, f)| \leq \frac{\varepsilon \text{cost}(T_i, f)}{4}. \tag{3}$$

For every $i \in [m] \setminus \text{Good}(D, f)$, we have

$$|\text{cost}(T_i, f) - \text{cost}(L_i, f)| \tag{4}$$

$$= \left| \sum_{(p,t) \in T_i} \|p - f(t)\|^2 - \sum_{t=b_i}^{e_i} \|g_i(t) - f(t)\|^2 \right|$$

$$= \left| \sum_{(p,t) \in T_i} \left(\|p - f(t)\|^2 - \|g_i(t) - f(t)\|^2\right) \right| \tag{5}$$

$$\leq \sum_{(p,t) \in T_i} \left| \|p - f(t)\|^2 - \|g_i(t) - f(t)\|^2 \right| \tag{6}$$

$$\leq \sum_{(p,t) \in T_i} \left(\frac{12\|g_i(t) - p\|^2}{\varepsilon} + \frac{\varepsilon \|p - f(t)\|^2}{2} \right) \tag{7}$$

$$= \frac{12\text{cost}(T_i, g_i)}{\varepsilon} + \frac{\varepsilon \text{cost}(T_i, f)}{2} \leq \frac{24\sigma}{\varepsilon} + \frac{\varepsilon \text{cost}(T_i, f)}{2}, \tag{8}$$

where (6) is by the triangle inequality, and (7) is by the weak triangle inequality (see [10, Lemma 7.1]). The inequality in (8) is because by construction $\text{cost}(T, f^*) \leq \sigma$ for some 2-approximation f^* of the 1-segment mean of T. Hence, $\text{cost}(T, g_i) \leq 2\text{cost}(T, f^*) \leq 2\sigma$.

Plugging (8) and (3) in (2) yields

$$\begin{aligned}
&|\text{cost}(P, f) - \text{cost}'(D, f)| \\
&\leq \sum_{i \in \text{Good}(D,f)} \frac{\varepsilon \text{cost}(T_i, f)}{4} + \sum_{i \in [m] \setminus \text{Good}(D,f)} \left(\frac{24\sigma}{\varepsilon} + \frac{\varepsilon}{2} \text{cost}(T_i, f) \right) \\
&\leq \left(\frac{\varepsilon}{4} + \frac{\varepsilon}{2}\right) \text{cost}(P, f) + \frac{24k\sigma}{\varepsilon},
\end{aligned}$$

where in the last inequality we used that fact that $|[m] \setminus \text{Good}(D, f)| \leq k-1 < k$ since f is a k-segment. Substituting σ yields

$$\begin{aligned}|\text{cost}(P, f) - \text{cost}'(D, f)| &\leq \frac{3\varepsilon}{4}\text{cost}(P, f) + \frac{\varepsilon \text{cost}(P, h)}{4\alpha} \\ &\leq \frac{3\varepsilon}{4}\text{cost}(P, f) + \frac{\varepsilon \text{cost}(P, f)}{4} \\ &= \varepsilon \text{cost}(P, f).\end{aligned}$$

Bound on $|D|$: Let $j \in [m-1]$, consider the values of T, Q and λ during the execution of Line 8 when $T = T_j$ is constructed. Let $Q_j = Q$ and $\lambda_j = \lambda$. The cost of the 1-segment mean of Q_j is at least $\lambda_j/2 > \sigma/2 > 0$, which implies that $|Q_j| \geq 3$ and thus $|T_j| \geq 1$. Since Q_{j-1} is the union of T_{j-1} with the first point of T_j we have $Q_{j-1} \subseteq T_{j-1} \cup T_j$. By letting g^* denote a 1-segment mean of $T_{j-1} \cup T_j$ we have

$$\text{cost}(T_{j-1} \cup T_j, g^*) \geq \text{cost}(Q_{j-1}, g^*) \geq \lambda_j/2 > \sigma/2.$$

Suppose that for our choice of $j \in [m-1]$, the points in $T_{j-1} \cup T_j$ are served by a single segment of h, i.e, $\{h(t) \mid b_{j-1} \leq t \leq e_j\}$ is a linear segment. Then

$$\begin{aligned}&\text{cost}(T_{j-1}, h) + \text{cost}(T_j, h) \\ &= \text{cost}(T_{j-1} \cup T_j, h) \geq \text{cost}(T_{j-1} \cup T_j, g^*) \\ &> \sigma/2.\end{aligned} \tag{9}$$

Let $G \subseteq [m-1]$ denote the union over all values $j \in [m-1]$ such that j is both even and satisfies (9). Summing (9) over G yields

$$\begin{aligned}\text{cost}(P, h) &= \sum_{j \in [m]} \text{cost}(T_i, h) \\ &\geq \sum_{j \in G} (\text{cost}(T_{j-1}, h) + \text{cost}(T_j, h)) \\ &\geq |G|\sigma/2.\end{aligned} \tag{10}$$

Since h is a (βk)-segment, at most $(\beta k) - 1$ sets among $T_1, \cdots, T_m$ are not served by a single segment of h, so $|G| \geq (m - \beta k)/2$. Plugging this in (10) yields $\text{cost}(P, h) \geq (m - \beta k)\sigma/4$. Rearranging,

$$m \leq \frac{4\text{cost}(P, h)}{\sigma} + \beta k = O\left(\frac{k\alpha}{\varepsilon^2}\right) + \beta k. \tag{11}$$

Running time: Theorem 3 shows how to compute a $(1, \varepsilon)$-coreset C in time $O(dn/\varepsilon^4)$ for n points using the algorithm in [11]. This algorithm is dynamic and supports insertion of a new point in $O(d/\varepsilon^4)$ time. Therefore, updating the 1-segment mean f^* and the coreset C can be done in $O(d/\varepsilon^4)$ time per point, and the overall running time is $O(nd/\varepsilon^4)$. ∎

For efficient k-segmentation we run a k-segment mean algorithm on our small coreset instead of the original large input. Since the coreset is small we can apply dynamic programming (as in [4]) in an efficient manner.

Unlike other coresets, our coreset is not a set of points in $\mathbb{R}^d$, but rather the structure D from Definition 4, and thus we cannot apply [4] directly. Instead, we use a straight-forward variant of this algorithm that is described in the supplementary material.

E. Parallel and Streaming Implementation

One major advantage of coresets is that they can be constructed in parallel as well as in a streaming setting. They can be constructed in what known as "embarrassingly parallel" due to the fact that the union of coresets is a coreset. More precisely, if a data set is split into ℓ subsets, and we compute a $(1+\varepsilon)$-coreset of size s for every subset, then the union of the ℓ coresets is a $(1+\varepsilon)$-coreset of the whole data set and has size $\ell \cdot s$. This is especially helpful if the data is already given in a distributed fashion on l different machines. Then, every machine will simply compute a coreset. The small coresets can be sent to a central node which approximately solves the optimization problem; see [10, Theorem 10.1] for more details and a formal proof.

In the streaming setting, data points arrive one by one into a system with a small memory that is incapable of storing more than ℓ points at a time. In a parallel setting, data points arrive at different nodes, and must me aggregated efficiently at a root node. More generally, we assume that both restrictions apply to our problem. In this work we present a general variant of "map and reduce" that allows our coreset to scale to both small-memory streaming systems, as well as massively parallel architectures.

We prove that we can re-compress a union of two coresets $D = D_1 \cup D_2$ again to obtain one coreset E of size $|D_1| = |D_2| = |D|/2$. This is the main result that is needed to use the merge-and-reduce approach that allows to use off-line coresets in the streaming and parallel model, as explained above. The construction of the coreset E is the same as in Algorithm 1, where each input point is replaced by a 3-tuple of D; see Definition 5. Plugging this in the merge-and-reduce framework ([10, Theorem 10.1]) yields the streaming version of our algorithm. We omit the details due to space limitations.

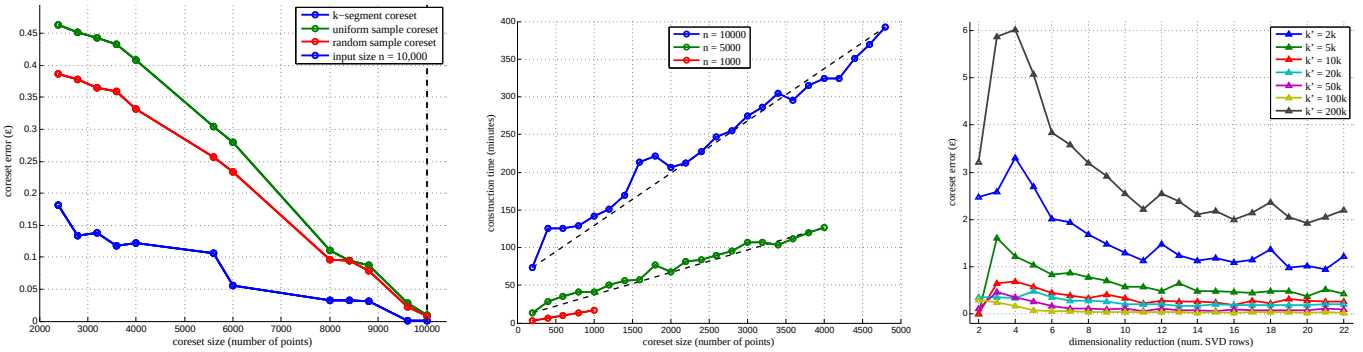
IV. Experimental Results

We now proceed to demonstrate the results of our algorithm, using data from several types of robots and mobile cameras.

A. Vector Quantization over Large Datasets

We first examine the empirical behavior of the algorithm on synthetic data. The purpose of these experiments is to verify (1) correctness of the approximation on a data with a known structure and ground truth, (2) efficiency (time and space), and (3) scalability of our algorithms. Our synthetic test data is obtained by generating a discrete k-segment P, and then corrupting the data with Gaussian and salt-and-pepper noise. We then benchmark the performance of a given (k, ε)-coreset D by comparing it against piecewise linear approximations with (1) a uniformly sampled subset of control points U and (2) a randomly placed control points R.

For a fair comparison between the (k, ε)-coreset D and the corresponding approximations U, R we require the linear



(a) Coreset size vs coreset error (b) (k, ε)-coreset size vs construction time (c) Coreset dimensionality vs coreset error

Fig. 3: Figure 3a shows the coreset error (ε) decreasing as a function of coreset size. The dotted black line indicates the point at which the coreset size is equal to the input size and thus can trivially obtain zero error simply by sampling every point. Figure 3b shows the coreset construction time in minutes as a function of coreset size. Trendlines show the linear increase in construction time with coreset size. Vertical gridlines illustrate the linear relationship between computation time and n. Figure 3c shows the reduction in coreset error as a function of the dimensionality of the 1-segment coreset, for a fixed input size (note that in practice dimensionality is often reduced down to $\mathbb{R}^2$.

approximations and the coreset to use the same number of coefficients overall. Coresets are evaluated by computing the fitting cost to a query k-segment Q that is constructed based on the a-priori parameters used to generate P.

a) Approximative Power: Figure 3a shows the aggregated fitting cost error for 1500 experiments on synthetic data. We generated data with $k = 20$ and varied the assumed k' segment complexity. In the plot we show how well a given k' performed as a guess for the true value of k. As can be seen, we significantly outperform the other approximation schemes. As the coreset size approaches the size P the error for all coresets decreases to zero as expected.

b) Coreset Construction Time: Figure 3b shows the linear relationship between input size and construction time of D for different coreset size. Figure 3c shows how a high dimensionality benefits coreset construction. This is even more apparent in real data which tends to be sparse, so that in practice we are typically able to further reduce the coreset dimension in each segment.

c) Scalability: The coresets presented in this work are parallelizable, as discussed in Section III-E. We demonstrate scalability by conducting very large scale experiments on both real and synthetic data. We employed the MathWorks distributed computing server running on a network of 255 Amazon EC2 cluster nodes. The nodes used were computation optimized 64-bit 32-core vCPU machines. A 256,000-frame VQ stream was created by looping a single test video sequence (approximately 3 hours of video at 20 frames per second). Using this parallel streaming architecture, the total coreset computation and merging time for this dataset was approximately 20 minutes, including overhead. For a single node running on a similar size dataset, we predict an estimated total running time of approximately 42 hours.

B. Semantic Video Segmentation

We now demonstrate efficient k-segment mean computation using our coreset. Data gathering is done as described in Section II. Our implementation ran on Matlab, on an Intel i7-3940XM CPU unless noted otherwise. Video processing was done with standard software in Matlab, which is not real time (i.e around 1fps), but descriptor computation and other operations can be done at real-time speeds even on low-power devices. We note that the dynamic programming allows us to compute k-segmentation for all k' values smaller than k, and we demonstrate a choice of k' that gives reasonable segmentation. We also note that the same 5000 VQ representatives are used for all of the video streams from all devices in our experiements, even though we have trained the VQ representatives from a significantly different images dataset (ImageNet/ILSVRC2013). The results obtained demonstrate the robustness of our method.

In Figure 5 we demonstrate segmentation of a video feed from a Double Robotics telepresence robot. The figure shows 4 minutes camera footage where robot was guided to follow a circular route with piecewise-linear segments, while its video feed was recorded. We visualize the first 300 eigenvectors of the bags-of-words representations for the video frames, and mark the segments suggested by the k-segment mean algorithm run on top of the proposed coreset. Inspecting the results, most of the segment transitions occur at scene and room changes. By construction of the feature vectors, this representation is robust to reasonable changes in the point of view and in the illumination of the environment. The resulting segmentation can be used so as to reason about the places traversed, activity of the robot, and unusual events. The piecewise-linear motion of the robot results in a video that easily summarized in 20 seconds using our Matlab implementation.

In Figure 4 we demonstrate segmentation of a video feed of 4 minutes taken from Google Glass. Coreset construction for this visually more complex video takes 38 seconds and allows us efficient segmentation of the video sequence with small preprocessing time.

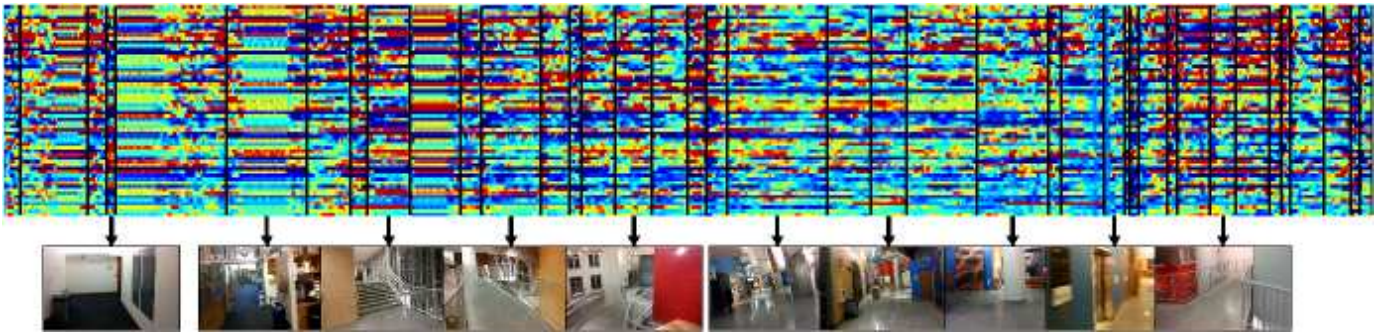


Fig. 4: Segmentation from Google Glass. Black vertical lines present segment boundaries, overlayed on top of the bags of word representation. Icon images are taken from the middle of each video segment.

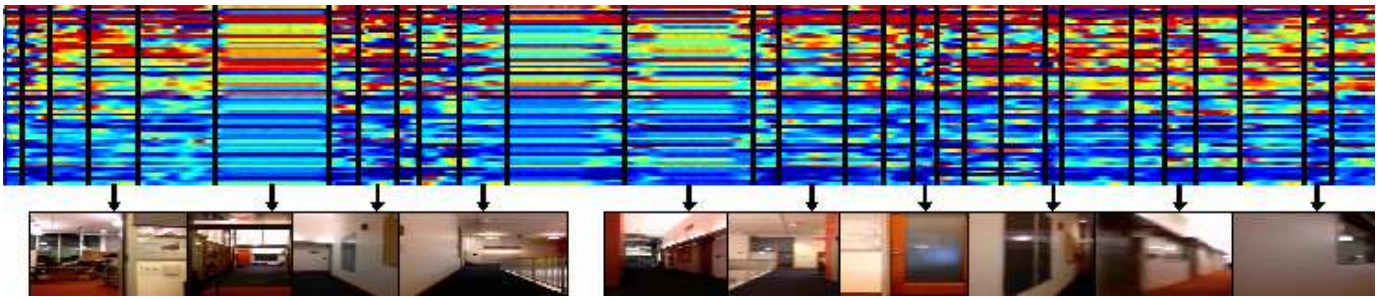


Fig. 5: Segmentation from a telepresence robot. Black vertical lines present segment boundaries, overlayed on top of the bags of word representation. Icon images are taken from the middle of each video segment.

C. Segmenting Multiple Video Streams

An additional property of our algorithm is the ability to handle segmentation based on several data streams, whose coresets are computed separately. Using the partition of segments in each D_i for several video streams indexed by i allowed us to obtain an over-segmentation marking events occuring in the scene and observed by one of the cameras. Since the computation time for the coreset is linear in n, aggregating the segment boundaries from all videostreams allowed us to compute the segmentation for each device separately without overhead. In our experiement, three video streams were taken (synchronized in time) from a web camera, an android device, and a glasses-mounted camera. The coreset was computed in an online fashion for each videostream, and the resulting segment end-points were unified. This resulted in a summarization of the scene capturing events from all devices.

V. Conclusions

In this paper we demonstrated a new framework for segmentation and event summarization of video data from robot cameras. We have shown the effectiveness and scalability of the algorithms proposed, and its applicability for large distributed video analysis with multiple devices. In the context of video processing, we demonstrate how using the right framework for analysis and clustering, even relatively straightforward representations of image content lead to a meaningful and reliable segmentation of video streams.

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximations via coresets. *Combinatorial and Computational Geometry - MSRI Publications*, 52: 1–30, 2005.

[2] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In Albert Ali Salah and Bruno Lepri, editors, *HBU*, volume 7065 of *Lecture Notes in Computer Science*, pages 29–39. Springer, 2011. ISBN 978-3-642-25445-1. URL http://dblp.uni-trier.de/db/conf/hbu/hbu2011.html#BaccoucheMWGB11.

[3] Sunil Bandla and Kristen Grauman. Active learning of an action detector from untrimmed videos. In *International Conference on Computer Vision*, 2013.

[4] Richard Bellman. On the approximation of curves by line segments using dynamic programming. *Commun. ACM*, 4(6):284, 1961.

[5] M Hughes N Caprani A. R. Doherty S. E. Hodges C Gurrin, Z Qiu and A. F. Smeaton. The smartphone as a platform for wearable cameras in health research. *American journal of preventive medicine*, 44(3):308–313, 2013.

[6] Winston Churchill and Paul Newman. Continually improving large scale long term visual navigation of a vehicle in dynamic urban environments. In *Proc. IEEE Intelligent Transportation Systems Conference (ITSC)*, Anchorage, USA, September 2012.

[7] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proc. 41th Ann. ACM Symp. on Theory of Computing (STOC)*, 2010. Manuscript available at arXiv.org.

[8] D. Feldman, A. Sugaya, and D. Rus. An effective coreset compression algorithm for large scale sensor networks. In *The 11th Intl. Conf. on Info. Proc. in Sensor Networks (IPSN)*, pages 257–268, 2012. ISBN 978-1-4503-1227-1.

[9] D. Feldman, C. Sung, and D. Rus. The single pixel gps: learning big data signals from tiny coresets. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 23–32. ACM, 2012.

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

[11] M. Ghashami and J. M. Phillips. In *Relative Errors for Deterministic Low-Rank Matrix Approximations (to appear)*, 2014.

[12] Anna C Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, S Muthukrishnan, and Martin J Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 389–398. ACM, 2002.

[13] Yogesh Girdhar and Gregory Dudek. Onsum: A system for generating online navigation summaries. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 746–751. IEEE, 2010.

[14] Yogesh Girdhar and Gregory Dudek. Efficient on-line data summarization using extremum summaries. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3490–3496. IEEE, 2012.

[15] Yogesh Girdhar, Philippe Giguere, and Gregory Dudek. Autonomous Adaptive Exploration using Realtime Online Spatiotemporal Topic Modeling. *International Journal of Robotics Research (Accepted)*, 2013.

[16] Sudipto Guha, Nick Koudas, and Kyuseok Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems (TODS)*, 31(1):396–438, 2006.

[17] Dorit S Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

[18] Svebor Karaman, Jenny Benois-Pineau, Remi Megret, Vladislavs Dovgalecs, Jean-Francois Dartigues, and Yann Gaestel. Human daily activities indexing in videos from wearable cameras for monitoring of patients with dementia diseases. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 4113–4116, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4109-9. doi: 10.1109/ICPR.2010.999. URL http://dx.doi.org/10.1109/ICPR.2010.999.

[19] Kurt Konolige, James Bowman, J.D. Chen, Patrick Mihelich, Michael Calonder, Vincent Lepetit, and Pascal Fua. View-based maps. *Int. J. Rob. Res.*, 29 (8):941–957, July 2010. ISSN 0278-3649. doi: 10.1177/0278364910370376. URL http://dx.doi.org/10.1177/0278364910370376.

[20] Hema Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[21] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *I. J. Robotic Res.*, 32(8):951–970, 2013.

[22] Yunpeng Li, David J. Crandall, and Daniel P. Huttenlocher. Landmark classification in large-scale image collections. In *ICCV*, pages 1957–1964, 2009.

[23] Zheng Lu and Kristen Grauman. Story-driven summarization for egocentric video. In *CVPR*, pages 2714–2721, 2013.

[24] Takuya Maekawa, Yutaka Yanagisawa, Yasue Kishino, Katsuhiko Ishiguro, Koji Kamei, Yasushi Sakurai, and Takeshi Okadome. Object-based activity recognition with heterogeneous sensors on wrist. In Patrik Floréen, Antonio Krüger, and Mirjana Spasojevic, editors, *Pervasive Computing*, volume 6030 of *Lecture Notes in Computer Science*, pages 246–264. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12653-6. doi: 10.1007/978-3-642-12654-3_15. URL http://dx.doi.org/10.1007/978-3-642-12654-3_15.

[25] (Names omitted for blind review). Visual precis generation using coresets. In *ICRA*. IEEE Press, 2014. accepted.

[26] Rohan Paul and Paul Newman. FAB-MAP 3D: Topological mapping with spatial and visual appearance. In *ICRA*, pages 2649–2656, Anchorage, Alaska, May 2010. 05.

[27] Michael S. Ryoo and Larry Matthies. First-person activity recognition: What are they doing to me? In *CVPR*, pages 2730–2737. IEEE, 2013. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2013.html#RyooM13.

[28] Greg Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology, 2005.

[29] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003. URL http://www.robots.ox.ac.uk/~vgg.

[30] Christoph Strecha, Alexander M. Bronstein, Michael M. Bronstein, and Pascal Fua. Ldahash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):66–78, 2012. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.103.

[31] Gian Diego Tipaldi, Daniel Meyer-Delius, and Wolfram Burgard. Lifelong localization in changing environments. 32(14):1662–1678, 2013.

[32] Pavan K. Turaga, Rama Chellappa, V. S. Subrahmanian,

and Octavian Udrea. Machine recognition of human activities: A survey. *IEEE Trans. Circuits Syst. Video Techn.*, 18(11):1473–1488, 2008.

[33] Heng Wang, Muhammad Muneeb Ullah, Alexander Kläser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference*, page 127, sep 2009. URL http://lear.inrialpes.fr/pubs/2009/WUKLS09.

[34] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010.