HOPE: Hierarchical Object Prototype Encoding for Efficient Object Instance Search in Videos

Tan Yu[1], Yuwei Wu[1,2]*, Junsong Yuan[1]
[1] ROSE Lab, Interdisciplinary Graduate School, Nanyang Technological University, Singapore
[2] Beijing Laboratory of Intelligent Information Technology, Beijing Institute of Technology, Beijing
{tyu008, jsyuan}@ntu.edu.sg, wuyuwei@bit.edu.cn

Abstract

This paper tackles the problem of object instance search in videos. To effectively capture the relevance between a query and video frames and precisely localize the particular object, we leverage the object proposals to improve the quality of object instance search in videos. However, hundreds of object proposals obtained from each frame could result in unaffordable memory and computational cost. To this end, we present a simple yet effective hierarchical object prototype encoding (HOPE) model to accelerate the object instance search without sacrificing accuracy, which exploits both the spatial and temporal self-similarity property existing in object proposals generated from video frames. We design two types of sphere k-means methods, i.e., *spatially-constrained sphere k-means and temporally-constrained sphere k-means to learn frame-level object prototypes and dataset-level object prototypes, respectively. In this way, the object instance search problem is cast to the sparse matrix-vector multiplication problem. Thanks to the sparsity of the codes, both the memory and computational cost are significantly reduced. Experimental results on two video datasets demonstrate that our approach significantly improves the performance of video object instance search over other state-of-the-art fast search schemes.*

1. Introduction

With the advent of the era of Big Data, a huge body of visual resources can be easily found on the multimedia sharing platforms such as YouTube, Facebook, and Flickr. Object instance search is attracting considerable attention in the multimedia and computer vision literature. Given a specific object as query, it aims to retrieve *which* frames of videos contain a specific object instance and localize *where* is the object in retrieved frames. Designing an effective object instance search system, however, is a challenging task due to the query usually only occupies a small area in the reference frame and there is dense clutter around it. In this paper, we present a hierarchical object prototype encoding method which can achieve satisfactory search accuracy, search efficiency, and memory cost simultaneously.

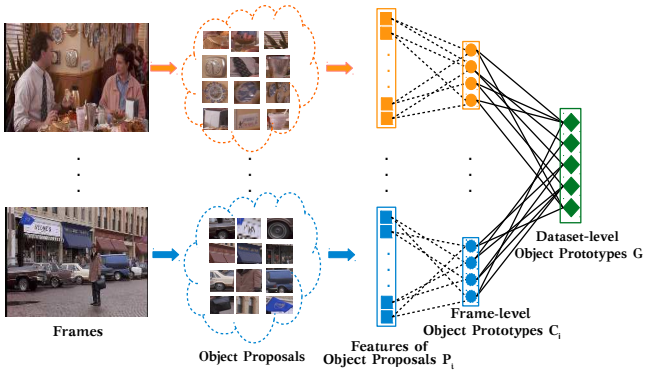


Figure 1. The framework of our hierarchical object prototype encoding. The features of the object proposals from frame f_i are first decomposed into the product of frame-level object prototypes $\mathbf{C}_i$ and the frame-level sparse codes $\mathbf{H}_i$ illustrated by dashed lines. The frame-level object prototypes of all the frames $[\mathbf{C}_1, ..., \mathbf{C}_n]$ are further decomposed into the product of dataset-level object prototypes $\mathbf{G}$ and the dataset-level sparse codes $[\mathbf{E}_1, ..., \mathbf{E}_n]$ depicted by solid lines. By exploiting our model, we can accelerate the object instance search without sacrificing accuracy, as validated in Section 4.

Different from the traditional image retrieval, the query object only occupies a small portion of a frame in the object instance search task. In this scenario, the relevance between the query object and the frame is not equivalent to the global similarity between the query object and the whole frame. Moreover, the object instance search task requires to precisely localize the object in the frames. Hence the global representation of the whole image/frame, *e.g.*, Vector of Locally Aggregated Descriptor (VLAD) [16], max pooling [27], cross-dimensional pooling [17] and bilinear pooling [9], might be neither effective to capture the relevance between the query object and frames nor applicable for the object localization.

*This work was done when Yuwei Wu was a research staff at NTU.

To tackle the drawbacks of global representation, many efforts [27, 22] have been devoted to utilizing the sliding windows which exhaustively crop patches at every location across multiple scales to conduct the spatial search in the reference frames. To precisely detect the object, it may require a large number of sliding windows especially when the size of the query is very small. Instead, Bhattacharjee *et al.* [4] exploited object proposals [28, 37] to enhance the efficiency of object search and achieved state-of-the-art performance in small object search. Nevertheless, hundreds of object proposals serve as candidate regions of the objects in each frame, which is still quite inefficient as it inevitably involves hundreds of times memory and computational cost. This means that the applicability of object proposals is limited for an efficient visual search. It motivates a question: how can we better advance the object proposals for robust visual search and achieve satisfactory search efficiency?

To achieve this goal, we start from the following observations. Hundreds of proposals from the same frame heavily overlap, leading to the fact that cropped patches perhaps are similar to each other. In addition, it is well known that the consecutive frames are also very similar with each other and thus they tend to share multiple object instances. In other words, object proposals generated from video frames possess the spatio-temporal self-similarity property and self-representation principle may well apply to them [36]. Thanks to the self-representation property, we propose a hierarchical object prototype encoding (HOPE) model to expedite the object proposal based object instance search.

Driven by the emergence of large-scale data sets and fast development of computation power, features based on convolutional neural networks (CNN) have proven to perform remarkably well on visual search [3, 2, 34, 35]. In this work, we exploit CNN features to represent object proposals. Given a query object represented by $\mathbf{q} \in \mathbb{R}^d$, directly computing the similarity between the query and m object proposals $\mathbf{P}_i \in \mathbb{R}^{d\times m}$ from the frame f_i is unaffordable because both the number of object proposals m and the number of the frames n are large. Therefore, we design two types of sphere k-means methods, *i.e.,* spatially-constrained sphere (SCS) k-means and temporally-constrained sphere (TCS) k-means to accelerate the object instance search without sacrificing the search accuracy.

In order to exploit the self-similarity property in the spatial domain, for the frame f_i, SCS k-means learns frame-level object prototypes $\mathbf{C}_i \in \mathbb{R}^{d\times t_1} (t_1 \ll m)$ and factorizes $\mathbf{P}_i$ into $\mathbf{C}_i\mathbf{H}_i$ where each column of $\mathbf{H}_i$ is the frame-level sparse code for the object proposal. Similarly, to exploit the self-similarity property in temporal domain, TCS k-means further factorizes all the frame-level object prototypes $[\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_n]$ into $\mathbf{G}[\mathbf{E}_1, \mathbf{E}_2, ..., \mathbf{E}_n]$, where $\mathbf{G} \in \mathbb{R}^{t_2} (t_2 \ll nt_1)$ denotes the dataset-level object prototypes and $\mathbf{E}_i$ denotes the dataset-level sparse codes for the frame-level object prototypes from frame f_i. Therefore, the object proposals $\mathbf{P}_i$ will be hierarchically decomposed into $\mathbf{G}\mathbf{E}_i\mathbf{H}_i$ as shown in Figure 1. In this scenario, rather than directly storing the features of all the object proposals of all the frames $[\mathbf{P}_1, ..., \mathbf{P}_n] \in \mathbb{R}^{d\times nm}$, we only need to store the dataset-level object prototypes $\mathbf{G} \in \mathbb{R}^{d\times t_2}$, the dataset-level sparse codes $\{\mathbf{E}_i\}_{i=1}^n$ and the frame-level sparse codes $\{\mathbf{H}_i\}_{i=1}^n$. Since $t_2 \ll nm$ and both $\mathbf{E}_i$ and $\mathbf{H}_i$ are sparse, the proposed HOPE model leads to a dramatic reduction in terms of memory and computational cost. In fact, as demonstrated in our experiments, the memory cost of storing the HOPE model is less than 1% of that of storing $\{\mathbf{P}_i\}_{i=1}^n$.

2. Related Work

Object Instance Search. A main challenge in object instance search [39, 41, 40, 43, 42] is that the query object only occupies a small area in the reference image and there may be dense clutters around it. In order to tackle this challenge, Tolias *et al.* [27] and Mohedano *et al.* [22] uniformly sampled tens of regions in reference images. The best-matched region of the reference image is considered as the search result. However, tens of sampled regions are not enough to capture the small object in the reference image. Bhattacharjee *et al.* [4] utilized the object proposals as candidate regions of the query object in reference images and achieved state-of-the-art performance in small object instance search. Nevertheless, hundreds of object proposals brought huge amount of memory and computational cost. In order to speed up the search, Meng *et al.* [20] selected the "keyobjects" from all the object proposals. To preserve the search precision, the selection ratio can not be too small, which limits its usefulness. Recently, Cao *et al.* [5] proposed a query-adaptive matching method. But it requires solving a quadratic optimization problem, which is computationally demanding. In contrast, our HOPE scheme only requires calculating the sparse matrix-vector multiplication, which is more efficient.

Efficient Search Methods. In object instance search tasks, the visual features, *e.g.*, VLAD and CNNs, are usually compressed to binary aggregated descriptors by hashing [6, 11, 31, 7, 24, 38] because the binarized one allows for fast Hamming distance computation as well as light storage of visual descriptors. However, the hashing algorithms are only able to produce a few distinct distances resulting in limited ability to describe the distance between data points. Parallelly, another scheme termed product quantization (PQ) [15, 10, 1, 33, 29, 19] is also widely used in fast nearest neighborhood (NN) search which decomposes the space into a Cartesian product of subspaces and quantizes each subspace individually. It is worth noting that PQ and its variants are closely related to sparse coding. They can be seen as a special case of block-wise sparse coding [32], where coefficients are only valued by 0 or 1. More re-

cently, inspired by sparse coding, Jain *et al.* [13] extended the product quantilization by adding the coefficients and achieved higher accuracy in NN search. As a contemporaneous work with [13], Iscen *et al.* [12] formulated the NN search as a matrix factorization problem also solved by sparse coding and achieved state-of-the-art performance. Inspired by the success of the sparse coding in NN search, the proposed HOPE model generates hierarchical sparse codes to exploit the self-similarity property among object proposals in videos.

3. Hierarchical Object Prototype Encoding

3.1. Problem Statement

Given a query represented by the ℓ_2-normalized feature $\mathbf{q} \in \mathbb{R}^d$ and a dataset consisting of n frames $\{f_i\}_{i=1}^n$, the object instance search is to retrieve all the relevant frames and localize the query object in the relevant frames. In most cases, the interested object normally takes up a small area in the whole frame. To capture the relevance of the frame with the query, for each frame f_i, we extract m object proposals serving as potential regions of the query object in the frame. We denote by $\mathbf{p}_{ij} \in \mathbb{R}^d$ the ℓ_2-normalized feature of the j-th object proposal from the i-th frame and denote by $\mathbf{P}_i = [\mathbf{p}_{i1}, ..., \mathbf{p}_{im}] \in \mathbb{R}^{d\times m}$ the ℓ_2-normalized features of all the object proposals from the i-th frame. In this scenario, the relevance score of the frame f_i with the query is determined by the similarity $S(\mathbf{q}, \mathbf{p}_{ij})$ between the query and the best-matched object proposal in the frame:

$$R(f_i) = \max_j S(\mathbf{q}, \mathbf{p}_{ij}) = \max_j \mathbf{q}^\top \mathbf{p}_{ij}. \tag{1}$$

Here, the best-matched object proposal is used to localize the query object in the relevant frame.

In order to obtain the relevance scores of all the frames, we need to compare the query with all the object proposals in all the frames, which is equivalent to calculating

$$\mathbf{s}^0 = \mathbf{q}^\top [\mathbf{P}_1, ..., \mathbf{P}_n]. \tag{2}$$

It should be emphasized that computing $\mathbf{s}^0$ in Eq. (2) requires $\mathcal{O}(nmd)$ complexity. In this case, the key problem of the object instance search is how to efficiently obtain $\mathbf{s}^0$.

3.2. Frame-level Object Prototype Encoding

As mentioned in Section 1, the proposals from the same frame tend to overlap with each other which brings a great amount of redundancy. Redundancy means that there exists self-similarity among the object proposals. To exploit the self-similarity property, we propose to factorize the features $\mathbf{P}_i$ of object proposals from the frame f_i using the frame-level object prototypes $\mathbf{C}_i$. The formulation is given by

$$\mathbf{P}_i \rightarrow \mathbf{C}_i \mathbf{H}_i, \tag{3}$$

where $\mathbf{C}_i = [\mathbf{c}_{i1}, ..., \mathbf{c}_{it_1}] \in \mathbb{R}^{d\times t_1}$ is generated by our spatially-constrained sphere k-means and $\mathbf{H}_i = [\mathbf{h}_{i1}, ..., \mathbf{h}_{im}] \in \mathbb{R}^{t_1\times m}$ consists of frame-level sparse codes of the object proposals generated from soft-assignment coding. We will introduce how to solve $\mathbf{C}_i$ and $\mathbf{H}_i$ in details.

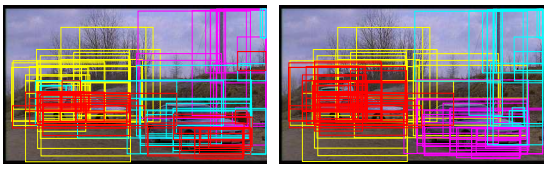
Spatially-constrained Sphere (SCS) K-means is an extension of sphere k-means [13] by taking the spatial information of the object proposals into consideration. The spatial relationship can provide valuable information. Intuitively, if two object proposals are heavily overlapped, they are inclined to share the objects. For convenience, we drop the index i in $\mathbf{P}_i, \mathbf{C}_i, \mathbf{H}_i$, which identifies a specific frame because the following spatially-constrained sphere k-means is applied to each frame independently. We denote by $\mathcal{B} = \{b_1, ..., b_m\}$ the bounding boxes of the object proposals. We develop the spatially-constrained sphere (SCS) k-means which iteratively updates the frame-level object prototypes (ℓ_2-normalized centroids of the clusters) and the assignment of the object proposals by

$$\begin{aligned} &Assign: A_u = \arg\max_k \mathbf{p}_u^\top \mathbf{c}_k + \frac{\lambda_s}{|\mathcal{A}_k|} \sum_{v\in\mathcal{A}_k} IoU(b_u, b_v), \\ &Update: \mathbf{c}_k = \sum_{v\in\mathcal{A}_k} \mathbf{p}_v / \| \sum_{v\in\mathcal{A}_k} \mathbf{p}_v \|_2, \end{aligned} \tag{4}$$

where A_u denotes the cluster index which u-th object proposal is assigned to, $\mathbf{c}_k$ denotes the k-th frame-level object prototypes, $\mathcal{A}_k$ denotes the set of object proposals assigned to the k-th cluster and $IoU(\cdot,\cdot)$ computes the intersection over union ratio of two bounding boxes. λ_s is the parameter controlling the weight of the spatial constraint. When $\lambda_s = 0$, the SCS k-means will degrade to the original sphere k-means. The iteration stops when the frame-level object prototypes do not change or the maximum iteration number is reached. Actually, our experiments show that SCS k-means converges within 20 iterations.

The spatial information is especially useful to handle the situation in which multiple different but similar instances appear in the frame. Figure 2 visualizes the clustering results of the object proposals from the sphere k-means and the spatially-constrained sphere k-means. It can be observed that the sphere k-means groups the object proposals containing different types of cars into the same clustering (red bounding boxes). In contrast, the spatially-constrained sphere k-means groups the object proposals containing different cars into different clusterings (red bounding boxes and purple bounding boxes).

Given an object proposal $\mathbf{p} \in \mathbb{R}^d$ and the frame-level object prototypes generated from spatially-constrained sphere k-means $\mathbf{C} \in \mathbb{R}^{d\times t_1}$, the soft-assignment coding (SAC) first finds z closest object prototypes of $\mathbf{p}$ which we denote



(a) Sphere k-means. (b) SCS k-means.

Figure 2. The comparison between sphere k-means and SCS k-means. Sphere k-means group two different types of cars into the same cluster (red bounding boxes). In contrast, SCS k-means groups different types of the cars in two clusters (red bounding boxes and purple bounding boxes).

by $\hat{\mathbf{C}} = [\mathbf{c}_{k_1}, .., \mathbf{c}_{k_z}] \in \mathbb{R}^{d \times z}$ and further obtains the codes $\mathbf{h} \in \mathbb{R}^{t_1}$ by

$$\hat{\mathbf{h}}(i) = \begin{cases} \exp(\beta \mathbf{c}_i^\top \mathbf{p}), \ i = k_1, ..., k_z \\ 0, \ otherwise \end{cases} \qquad \mathbf{h} = \hat{\mathbf{h}} / \hat{\mathbf{h}}^\top \mathbf{1}, \tag{5}$$

where β controls the softness of the assignment. The sparsity of the codes from SAC is strictly controlled by z. The soft-assignment coding was traditionally used as a feature extraction method for image classification [18] while we use it as a sparse coding method to achieve high efficiency for object instance search.

Complexity Analysis in Search Phase. Both the SCS k-means and the soft-assignment coding can be conducted offline, which does not affect the search time. In the search phase, the similarity scores for all the object proposals of all the frames can be efficiently achieved by

$$\mathbf{s}^1 = \mathbf{q}^\top [\mathbf{C}_1 \mathbf{H}_1, ..., \mathbf{C}_n \mathbf{H}_n]. \tag{6}$$

We denote by z_1 the number of non-zeros elements in each column of $\mathbf{H}_i$. Computing $\mathbf{s}^1$ in Eq. (6) only requires $\mathcal{O}(nt_1 d + nmz_1)$ complexity. This is because we can first compute $[\mathbf{x}_1, ..., \mathbf{x}_n] = \mathbf{q}^\top [\mathbf{C}_1, ..., \mathbf{C}_n]$ taking $\mathcal{O}(nt_1 d)$ complexity and then compute $\mathbf{s}^1 = [\mathbf{x}_1 \mathbf{H}_1, ..., \mathbf{x}_n \mathbf{H}_n]$ which is a sparse matrix-vector multiplication (SpMV) problem [30] taking $\mathcal{O}(nmz_1)$ complexity. In fact, $t_1 \ll m$ and $z_1 \ll d$, which means $nt_1 d + nmz_1 \ll nmd$. Therefore, computing $\mathbf{s}^1$ is much more efficient than directly computing $\mathbf{s}^0$ in Eq. (2). In addition, to store the sparse matrix $\mathbf{H}_i$, we only need to store its non-zero elements. Therefore, the memory cost for storing $\{\mathbf{C}_i\}_{i=1}^n$ and $\{\mathbf{H}_i\}_{i=1}^n$ is only $\mathcal{O}(nt_1 d + nmz_1)$ which is much less than that of storing $\{\mathbf{P}_i\}_{i=1}^n$. However, when n is huge considering the fact that a video consists of enormous frames, the computational cost of calculating $\mathbf{s}^1$ as well as the memory cost of storing $\{\mathbf{C}_i\}_{i=1}^n$ and $\{\mathbf{H}_i\}_{i=1}^n$ are still considerable. It motivates us to exploit the self-similarity across the frames further.

3.3. Dataset-level Object Prototype Encoding

It is well known that the consecutive frames are very similar to each other. To further speed up the object instance search in videos, we propose to encode the frame-level object prototypes again using the self-similarity property across multiple different frames. We denote by $[\mathbf{C}_1, ..., \mathbf{C}_n] \in \mathbb{R}^{d \times nt_1}$ the frame-level object prototypes of all the frames in the dataset. The dataset-level prototype encoding factorizes $[\mathbf{C}_1, ..., \mathbf{C}_n]$ into $\mathbf{GE}$, where $\mathbf{G} = [\mathbf{g}_1, ..., \mathbf{g}_{t_2}] \in \mathbb{R}^{d \times t_2}$ comprises dataset-level object prototypes generated from the proposed temporally-constrained sphere k-means and $\mathbf{E} = [\mathbf{E}_1, ..., \mathbf{E}_n] = [\mathbf{e}_{11}, ..., \mathbf{e}_{1t_1}, \mathbf{e}_{21}, ..., \mathbf{e}_{nt_1}] \in \mathbb{R}^{t_2 \times nt_1}$ is composed of the dataset-level sparse codes generated from the soft-assignment coding. In what follows, we show the whole encoding flow of the proposed HOPE model.

Algorithm 1 Hierarchical Object Prototype Encoding

Input: Object proposals from n frames $\{\mathbf{P}_i\}_{i=1}^n$, the number of frame-level object prototypes per frame t_1, the number of dataset-level object prototypes t_2, the number of non-zero elements in each frame-level code z_1, the number of non-zero elements in each dataset-level code z_2.

Output: Dataset-level object prototypes $\mathbf{G}$, dataset-level codes $\{\mathbf{E}_i\}_{i=1}^n$, frame-level codes $\{\mathbf{H}_i\}_{i=1}^n$.

for $i = 1$ ... n
  $\mathbf{C}_i \leftarrow \text{SCSKmeans}(\mathbf{P}_i, t_1)$ using Eq. (4)
  $\mathbf{H}_i \leftarrow \text{SAC}(\mathbf{P}_i, \mathbf{C}_i, z_1)$ using Eq. (5)
end
$\mathbf{G} \leftarrow \text{TCSKmeans}([\mathbf{C}_1, ..., \mathbf{C}_n], t_2)$ using Eq. (7)
for $i = 1$ ... n
  $\mathbf{E}_i \leftarrow \text{SAC}(\mathbf{G}, \mathbf{C}_i, z_2)$ using Eq. (5)
end
return $\mathbf{G}$, $\{\mathbf{E}_i\}_{i=1}^n$, $\{\mathbf{H}_i\}_{i=1}^n$

Temporally-constrained Sphere (TCS) K-means. In a video, the temporal relationship among the frames can also provide valuable information. One common sense is that the consecutive frames will be prone to contain the similar objects. We thus utilize the temporal information to weakly supervise the clustering.

We divide the frames into multiple groups according to their temporal location. For example, given a long video consisting of M frames, we can equally divide the M frames into τ groups. According to their group assignment, the frame-level object prototypes will be divided into multiple temporal chunks $\{\mathcal{S}_1, ..., \mathcal{S}_\tau\}$. Sometimes, the dataset also provides the shot information, *i.e.*, it indicates which frame is from which shot. In this case, we can directly arrange the frame-level object prototypes from the same video shot in the same temporal chunk. The temporally-

constrained sphere k-means iteratively updates the dataset-level object prototypes and the assignment of the frame-level object prototypes by

$$\begin{aligned} Assign: A_u &= \arg\max_k \mathbf{c}_u^\top \mathbf{g}_k + \lambda_t \frac{|\mathcal{A}_k \cap \mathcal{S}_{\mathbf{c}_u}|}{|\mathcal{A}_k|}, \\ Update: \mathbf{g}_k &= \sum_{v \in \mathcal{A}_k} \mathbf{c}_v / \| \sum_{v \in \mathcal{A}_k} \mathbf{c}_v \|_2, \end{aligned} \tag{7}$$

where $\mathcal{S}_{\mathbf{c}_u}$ denotes the temporal chunk containing the u-th frame-level object prototype $\mathbf{c}_u$. $\mathcal{A}_k \cap \mathcal{S}_{\mathbf{c}_u}$ denotes the set of the frame-level object prototypes assigned to the k-th cluster and $\mathcal{S}_{\mathbf{c}_u}$. λ_t is the parameter controlling the weight of the temporal constraint. The temporally-constrained sphere k-means will reduce to the original sphere k-means when λ_t is 0. The iteration stops when the dataset-level object prototypes do not change or the maximum iteration number is reached. Our experiments show that the iteration can converge within 100 iterations.

Complexity Analysis in Search Phase. Utilizing the generated dataset-level object prototypes $\mathbf{G}$, the dataset-level sparse codes $\{\mathbf{E}_i\}_{i=1}^n$ and the frame-level sparse codes $\{\mathbf{H}_i\}_{i=1}^n$, the similarity scores for all the proposals of all the frames can be obtained by

$$\mathbf{s}^2 = \mathbf{q}^\top \mathbf{G}[\mathbf{E}_1\mathbf{H}_1, ..., \mathbf{E}_n\mathbf{H}_n]. \tag{8}$$

We denote by z_2 the number of non-zero elements in each column of $\mathbf{E}_i$ and denote by z_1 the number of non-zero elements in each column of $\mathbf{H}_i$. We decompose computing $\mathbf{s}^2$ into following three steps:

$$\begin{aligned} &1): \mathbf{x} \leftarrow \mathbf{q}^\top \mathbf{G} \quad Complexity: \mathcal{O}(t_2 d) \\ &2): [\mathbf{y}_1, ..., \mathbf{y}_n] \leftarrow [\mathbf{x}\mathbf{E}_1, ..., \mathbf{x}\mathbf{E}_n] \quad Complexity: \mathcal{O}(nt_1 z_2) \\ &3): \mathbf{s}^2 \leftarrow [\mathbf{y}_1\mathbf{H}_1, ..., \mathbf{y}_n\mathbf{H}_n] \quad Complexity: \mathcal{O}(nmz_1) \end{aligned} \tag{9}$$

Both step 2) and step 3) in Eq. (9) are the sparse matrix-vector multiplication (SpMV) problem [30] which take $\mathcal{O}(nt_1z_2)$ and $\mathcal{O}(nmz_1)$, respectively. In total, the complexity of computing $\mathbf{s}^2$ is $\mathcal{O}(t_2d + nt_1z_2 + nmz_1)$. Since $t_2 \ll nt_1$ and $z_2 \ll d$, computing $\mathbf{s}^2$ is much more efficient than computing $\mathbf{s}^1$. Meanwhile, the total memory cost of storing $\mathbf{G}$, $\{\mathbf{E}_i\}_{i=1}^n$ and $\{\mathbf{H}_i\}_{i=1}^n$ is $\mathcal{O}(t_2d + nt_1z_2 + nmz_1)$, which is much less than that of storing $\{\mathbf{C}_i\}_{i=1}^n$ and $\{\mathbf{H}_i\}_{i=1}^n$ in computing $\mathbf{s}^1$.

3.4. Non-exhaustive Search

The above analysis shows that computing $\mathbf{s}^2$ takes $\mathcal{O}(t_2d + nt_1z_2 + nmz_1)$ complexity. In the practical cases, $t_1 \ll m$ and $t_2 d \ll nmz_1$. Therefore, the most time-consuming part of computing $\mathbf{s}^2$ is multiplying the sparse codes $\mathbf{h}_{ij}$ for each object proposal. To further improve the efficiency, we propose the non-exhaustive search scheme to avoid comparing the query with all the object proposals of all the frames. Since the frame-level object prototypes are the ℓ_2-normalized centroids of the clusters of object proposals, they represent different types of objects in the frame. In other words, they can be utilized as the representatives of the object proposals. Therefore, the relevance score of the frame with the query can be more efficiently calculated by the best-matched frame-level object prototype:

$$R(f_i) = \max_{j=1,...,t_1} (\mathbf{q}^\top \mathbf{c}_{ij}) \approx \max_{j=1,...,t_1} (\mathbf{q}^\top \mathbf{G}\mathbf{e}_{ij}). \tag{10}$$

We rank all the frames in the dataset through the frame-level relevance scores computed by Eq. (10). A candidate list of frames $\mathcal{F} = [f_{r1}, ..., f_{rs}]$ will be obtained by selecting a portion of frames having high rankings. In this case, we only need to conduct spatial localization of the query object in the candidate list $\mathcal{F}$ by

$$\mathbf{s}^3 = \mathbf{q}^\top \mathbf{G}[\mathbf{E}_{r1}\mathbf{H}_{r1}, ..., \mathbf{E}_{rs}\mathbf{H}_{rs}]. \tag{11}$$

The best-matched object proposal of each frame will serve as the detected region of the query object in the frame. Computing $\mathbf{s}^3$ only requires $\mathcal{O}(t_2d + nt_1z_2 + \alpha nmz_1)$ complexity, where α is the ratio of the frames for further localizing the query object. It is much more efficient than computing $\mathbf{s}^2$ as $\alpha \ll 1$. Overall, the frame-level object prototypes $\mathbf{C}_i$ serve two roles: (1) the representatives of object proposals to efficiently determine the relevance score of the frame. (2) the prototypes to encode the features of object proposals to speed up the object localization in the relevant frames. In addition, the frame-level codes $\{\mathbf{H}_i\}_{i=1}^n$ are only used for object localization in the frame and irrelevant with the relevance scores of the frames. Table 1 summarizes the complexity of different schemes.

Table 1. The complexity analysis of different schemes. m is the number of object proposals per frame, n is the number of frames. d is the feature dimension. $t_1 \ll m$ is the number of frame-level object prototypes per frame. $t_2 \ll nt_1$ is the number of dataset-level object prototypes. z_1 and z_2 denote the number of non-zero elements in frame-level codes and dataset-level codes, respectively.

Scheme	Complexity
$\mathbf{s}^0$ in Eq. (2)	$\mathcal{O}(mnd)$
$\mathbf{s}^1$ in Eq. (6)	$\mathcal{O}(nt_1d + nmz_1)$
$\mathbf{s}^2$ in Eq. (8)	$\mathcal{O}(t_2d + nt_1z_2 + nmz_1)$
$\mathbf{s}^3$ in Eq. (11)	$\mathcal{O}(t_2d + nt_1z_2 + \alpha nmz_1)$

4. Experiment

4.1. Settings and Datasets

In this paper, we adopt Edge Boxes [37] to generate object proposals serving as the potential regions of the object instance. For each proposal, we extract its feature by max-pooling the last convolutional layer of VGG-16 CNN model [25] pre-trained on the Imagenet dataset. The max-pooled 512-dimensional features are further post-processed

(a) Groundhong Day.

(b) NTU-VOI.

Figure 3. Query objects visualization.

by principle component analysis (PCA) and whitening in order to suppress the burstiness [14] but the dimension of the feature is kept as 512. The effectiveness of the proposed method is evaluated by mean average precision (mAP).

We conduct the systematic experiments on the Groundhog Day [26] and NTU-VOI [21] datasets. Groundhog Day dataset contains 5640 keyframes and six types of small query objects: Red clock, Microphone, Black clock, Frame sign, Phil sign and Digital clock. NTU-VOI dataset contains 37340 frames and we use five types of query objects: Ferrari, Kittyb, Kittyg, Maggi and Plane to evaluate the proposed method. Figure 3 visualizes the query objects.

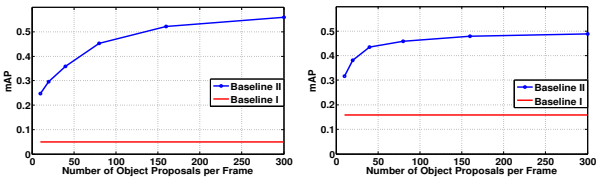
4.2. Effectiveness of Object Proposals

In this section, we evaluate the effectiveness of the object proposals. The first algorithm (Baseline I) extracts features for the whole frame and represents each frame with a global feature. In this scenario, the frames are ranked according to their cosine similarity to the query object. The second algorithm (Baseline II) exhaustively compares the query object with all the object proposals of all the frames, illustrated in Eq. (2). In Baseline II, the best-matched proposal of each frame, *i.e.*, the proposal with the highest similarity scores, determines the relevance between the query and the frame. In other words, the frames are ranked according to the similarity scores of their best-matched proposals.

Figure 4 shows the object instance search performance of the Baseline I and Baseline II. It can be observed that the performance of Baseline II is much better than Baseline I. At the same time, the mAP of baseline II improves as the number of object proposals n increases. However, the increase rate is slower and slower. It makes sense as in most cases, hundreds of object proposals are enough to capture the location of the object instance in the frames. To balance the effectiveness and efficiency, we set the default number of object proposals as 300. We will use Baseline II algorithm as the reference to further evaluate the proposed hierarchical object prototype encoding method.

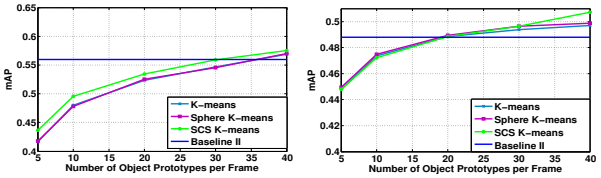
4.3. Frame-level Object Prototype Encoding

The proposed hierarchical object prototype encoding consists of two levels. We first evaluate the effectiveness of the frame-level object prototypes encoding and do not conduct the dataset-level object prototype encoding. We adopt s^1 scheme in Eq. (6) where we set the number of nonzero elements in frame-level sparse codes z_1 as 3.



(a) Groundhog Day. (b) NTU-VOI.

Figure 4. The mAP of two baseline algorithms on Groundhog day and NTU-VOI datasets. Benefited from object proposals, Baseline II is much better than Baseline I and the performance of Baseline II improves as the number of object proposals increases.

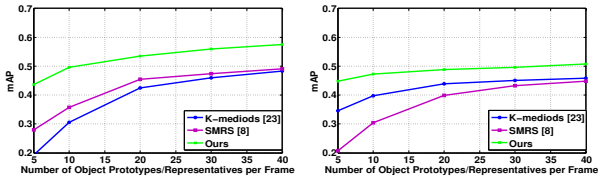


(a) Groundhog Day. (b) NTU-VOI.

Figure 5. The mAP comparison between K-means, Sphere K-means and SCS K-means on Groundhog Day and NTU-VOI datasets. SCS K-means achieves the best performance and it is better than Baseline II when the number of object prototypes per frame t_1 surpasses 30.

Figure 5 compares the performance of the frame-level object prototypes generated from k-means, sphere k-means and spatially-constrained sphere k-means, respectively. In our implementation, the object prototypes are generated from 300 object proposals and λ_s is fixed as 3 on both datasets. It can be observed that, in general, the sphere k-means is slightly better than k-means. In comparisons, the proposed spatially-constrained sphere k-means is better than the sphere k-means. Interestingly, when the number of frame-level object prototypes per frame t_1 is over 30, the performance achieved by the frame-level object prototypes is even better than baseline II. Although a larger number of frame-level object prototypes can bring higher mAP, the default number of the object prototypes is set as 30 to balance the accuracy and efficiency.

Comparison with Representative Selection. We compare the performance of the frame-level object prototypes with the object representatives selected from K-mediods [23] and the SMRS method [8] which select representative object proposals from hundreds of object proposals for each frame. In the representative selection scheme, we only need to compare the query object with the selected representative objects and the exhaustive search can be avoided. The fundamental difference of the proposed frame-level object prototypes from the object representatives is that the object representatives are selected from the original data samples whereas the object prototypes are generated from the cen-



(a) Groundhog Day. (b) NTU-VOI.

Figure 6. Comparison of the proposed frame-level object prototypes with the object representatives generated from K-mediods [23] and SMRS [8].

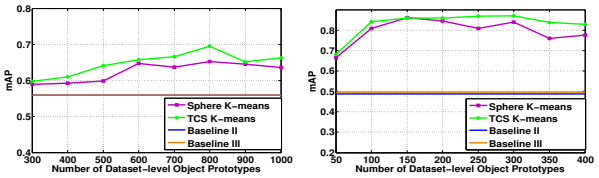
troids of the clusters which are more robust. From Figure 6, we can see that the performance of the proposed object prototypes scheme is much better than that from representative objects generated from SMRS and K-mediods.

4.4. Effectiveness of HOPE

In this section, we evaluate the whole HOPE model. We adopt the non-exhaustive search scheme in Eq. (11) where we set α as 0.05. We denote by Baseline III the implementation in Section 4.3 where the frame-level object prototypes are generated from the SCS k-means, the number of frame-level object prototypes $t_1 = 30$ and the number of non-zero elements in frame-level sparse codes $z_1 = 3$. We fix the settings of frame-level prototype encoding in HOPE model exactly same as Baseline III and focus on testing dataset-level object prototype encoding. On the Groundhog Day dataset, we equally divide the 5640 keyframes into 50 groups to generate temporal chunks required by the TCS k-means. On the NTU-VOI dataset, we directly use the shot information provided by the dataset to obtain temporal chunks. On both datasets, we set the $\lambda_t = 2$.

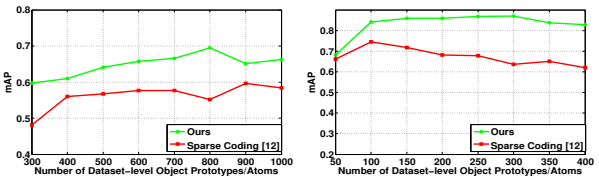
We first compare the performance of the dataset-level object prototypes using the sphere k-means and the TCS k-means, where the number of non-zero elements in the codes z_2 is set as 3. As depicted in Figure 7, the performance of dataset-level object prototypes generated from the TCS k-means is significantly better than that of generated from the sphere k-means. Surprisingly, Figure 7 shows that the mAP of from our HOPE model is much better than both Baseline II and Baseline III, which can be attributed to the denoising effect of the HOPE model. It is worth noting that 800 dataset-level object prototypes can achieve 0.70 mAP on the Groundhog day dataset. In this scenario, storing the HOPE model ($\{\mathbf{H}_i\}_{i=1}^n, \{\mathbf{E}_i\}_{i=1}^n, \mathbf{G}$) only requires 28.64 MB whereas the Baseline II costs 3.22 GB to store $\{\mathbf{P}_i\}_{i=1}^n$. On the NTU-VOI dataset, 100 dataset-level object prototypes can achieve 0.84 mAP, which only requires 176.47MB to store the HOPE model whereas the Baseline II costs 21.36 GB. On both datasets, the memory cost from HOPE is less than 1% of that of Baseline II.

Comparison with Sparse Coding. We compare our object prototype encoding with sparse coding method proposed in [12] to further validate the effectiveness of the



(a) Groundhog Day. (b) NTU-VOI.

Figure 7. The mAP comparison between sphere k-means and TCS k-means on Groundhog Day and NTU-VOI datasets.



(a) Groundhog Day. (b) NTU-VOI.

Figure 8. The performance comparison between sparse coding [12] and our dataset-level object prototype encoding.

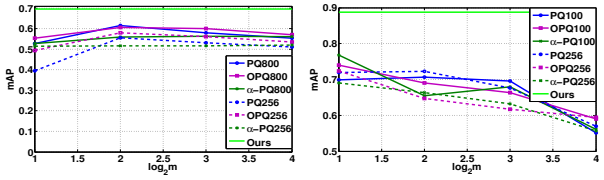
HOPE. To make a fair comparison, both the sparse coding and the dataset-level object prototype encoding are conducted on the exactly same frame-level object prototypes generated from SCS K-means and we fix z_2 as 3 for both sparse coding and dataset-level object prototype encoding. From Figure 8, we can see the proposed dataset-level object prototype encoding is much better than that of sparse coding used in [12].

Comparison with Quantization. In order to further validate the effectiveness of the dataset-level object prototype encoding, we compare it with product quantization (PQ) [15], optimized product quantization (OPQ) [10], and product quantization with coefficients (α-PQ) [13]. In the quantization scheme, we quantilize all the frame-level object prototypes into the dataset-level quantilizers and the relevance score of the frame-level object prototypes can be efficiently obtained from the look-up table. In the implementation of PQ, OPQ and α-PQ, we vary m, the number of sub-groups, from 2 to 16. We first set the number of clusters of each sub-group as 256 which is the standard settings of PQ. Then we set the number of clusters of each sub-group as 800 on the Groundhog day dataset and 100 on the NTU-VOI dataset which are the same settings as our dataset-level object prototype encoding. We can observe from Figure 9 that the proposed dataset-level object prototype encoding significantly outperforms PQ, OPQ and α-PQ.

Efficiency Comparison. We compare the memory and time cost of the proposed prototype encoding (ours) with Sparse Coding (SC) and Product Quantization (PQ) in Table 2. In implementation, we set $z_2 = 3$ and $t_2 = 800$ for ours and SC. In PQ, we set the number of sub-codebooks m as 4 and the sub-codebook size is 800. As we can see from Table 2 that ours significantly outperforms SC and PQ in precision with comparable memory and time cost.

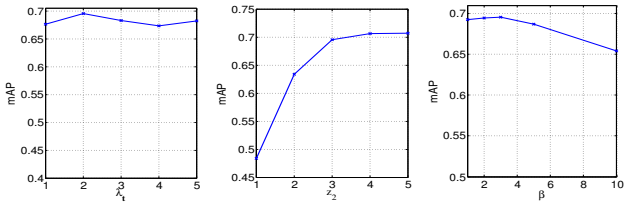
Method	Memory	Search Time	mAP
Ours	28.6MB	218 ms	0.70
SC	28.6MB	218 ms	0.55
PQ	27.0MB	212 ms	0.62

Table 2. Efficiency comparison on the Groundhog Day.



(a) Groundhong Day. (b) NTU-VOI.

Figure 9. Comparison of our dataset-level object prototype encoding with product quantization [15], optimized product quantization [10] and product quantization with coefficients [13].



(a) λ_t (b) z_2 (c) β

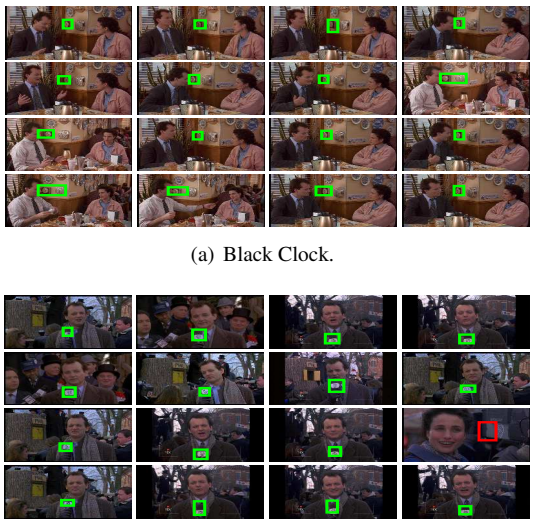
Figure 10. The influence of parameters on the Groundhog Day.

Parameters Sensitivity. Figure 10 shows the influence of the parameters. The mAP is relatively stable when $\lambda_t \in [1, 5]$. The bigger z_2 brings better mAP but higher memory and computation cost. Particularly, when $z_2 = 1$, the soft-assignment becomes hard-assignment and it achieves worse mAP than the soft-assignment when $z_2 > 1$. To balance the precision and efficiency, we set $z_2 = 3$. When $\beta = \infty$, the soft-assignment will become hard-assignment, so we should not set β to be too large. We set the default value of β as 3.

Search Results Visualization. Finally, we visualize the search results from the proposed HOPE model in Figure 11 and Figure 12. We can see that the query object only occupies a very small area surrounded by very dense clutter but our method can accurately find the frames containing the query object and precisely localize it in the frames.

5. Conclusion

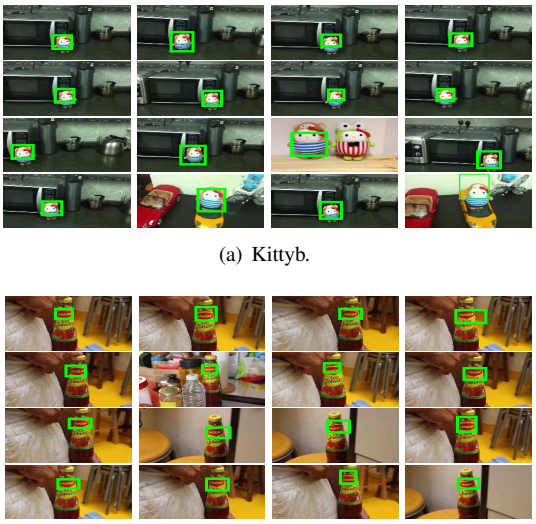
In this paper, we leverage the object proposals to improve the quality of object instance search in videos. To address the large memory and computational cost of using object proposals, we propose to formulate the search into a hierarchical sparse coding problem. By utilizing the spatial and temporal self-similarity property of object proposals, we present SCS K-means and TCS K-means to learn the frame-level object prototypes and dataset-level object prototypes, respectively. Experimental results on two video datasets demonstrate that our approach can achieve even better performance than exhaustive search using all object proposals within a fraction of complexity and significantly outperform other state-of-the-art fast methods.



(a) Black Clock.

(b) Microphone.

Figure 11. Visualization of top-16 search results on the Groundhog dataset with query black clock and microphone.



(a) Kittyb.

(b) Maggi.

Figure 12. Visualization of top-16 search results on the NTU-VOI dataset with query Kittyb and Maggi.

Acknowledgements: This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114. This research was carried out at the ROSE Lab at the Nanyang Technological University, Singapore. The ROSE Lab is supported by the National Research Foundation, Prime Ministers Office, Singapore, under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office. We gratefully acknowledge the support of NVAITC (NVIDIA AI Technology Centre) for their donation of a Tesla K80 and M60 GPU used for our research at the ROSE Lab.

References

[1] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 931–938, 2014.

[2] A. Babenko and V. Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1269–1277, 2015.

[3] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *European Conference on Computer Vision*, pages 584–599. Springer, 2014.

[4] S. D. Bhattacharjee, J. Yuan, Y.-P. Tan, and L.-Y. Duan. Query-adaptive small object search using object proposals and shape-aware descriptors. *IEEE Transactions on Multimedia*, 18(4):726–737, 2016.

[5] J. Cao, L. Liu, P. Wang, Z. Huang, C. Shen, and H. T. Shen. Where to focus: Query adaptive matching for instance retrieval using convolutional feature maps. *arXiv preprint arXiv:1606.06811*, 2016.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[7] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *European Conference on Computer Vision*, pages 219–234. Springer, 2016.

[8] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1600–1607. IEEE, 2012.

[9] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 317–326, 2016.

[10] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2953, 2013.

[11] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 817–824, 2011.

[12] A. Iscen, M. Rabbat, and T. Furon. Efficient large-scale similarity search using matrix factorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[13] H. Jain, P. Pérez, R. Gribonval, J. Zepeda, and H. Jégou. Approximate search with quantized sparse representations. In *European Conference on Computer Vision*, pages 681–696. Springer, 2016.

[14] H. Jégou and O. Chum. Negative evidences and cooccurences in image retrieval: The benefit of pca and whitening. In *European Conference on Computer Vision*, pages 774–787. Springer, 2012.

[15] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[16] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, 2012.

[17] Y. Kalantidis, C. Mellina, and S. Osindero. Cross-dimensional weighting for aggregated deep convolutional features. *arXiv preprint arXiv:1512.04065*, 2015.

[18] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *Proceedings of the International Conference on Computer Vision*, pages 2486–2493, 2011.

[19] J. Martinez, H. H. Hoos, and J. J. Little. Solving multi-codebook quantization in the gpu. In *European Conference on Computer Vision*, pages 638–650. Springer, 2016.

[20] J. Meng, H. Wang, J. Yuan, and Y.-P. Tan. From keyframes to key objects: Video summarization by representative object proposal selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2016.

[21] J. Meng, J. Yuan, J. Yang, G. Wang, and Y.-P. Tan. Object instance search in videos via spatio-temporal trajectory discovery. *IEEE Transactions on Multimedia*, 18(1):116–127, 2016.

[22] E. Mohedano, K. McGuinness, N. E. O'Connor, A. Salvador, F. Marques, and X. Giro-i Nieto. Bags of local convolutional features for scalable instance search. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ICMR '16, pages 327–331, New York, NY, USA, 2016. ACM.

[23] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009.

[24] X. Shi, F. Xing, J. Cai, Z. Zhang, Y. Xie, and L. Yang. Kernel-based supervised discrete hashing for image retrieval. In *European Conference on Computer Vision*, pages 419–433. Springer, 2016.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606, 2009.

[27] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of cnn activations. In *Proceedings of International Conference on Learning Representations*, 2016.

[28] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[29] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Supervised quantization for similarity search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[30] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, and J. Demmel. Optimization of sparse matrix–vector multipli-

cation on emerging multicore platforms. *Parallel Computing*, 35(3):178–194, 2009.

[31] Y. Xia, K. He, P. Kohli, and J. Sun. Sparse projections for high-dimensional binary codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3332–3339, 2015.

[32] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[33] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *Proceedings of the International Conference on Machine Learning*, pages 838–846, 2014.

[34] L. Zheng, S. Wang, L. Tian, F. He, Z. Liu, and Q. Tian. Query-adaptive late fusion for image search and person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1741–1750, 2015.

[35] L. Zheng, Y. Yang, and Q. Tian. Sift meets cnn: A decade survey of instance retrieval. *arXiv preprint arXiv:1608.01807*, 2016.

[36] P. Zhu, W. Zuo, L. Zhang, Q. Hu, and S. C. K. Shiu. Unsupervised feature selection by regularized self-representation. *Pattern Recognition*, 48(2):438–446, 2015.

[37] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014.

[38] W. Hong, J. Yuan, S. D. Bhattacharjee. Fried Binary Embedding for High-Dimensional Visual Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[39] V. Chandrasekhar, J. Lin, Q. Liao, O. Morere, A. Veillard, L. Duan, and T. Poggio. Compression of deep neural networks for image instance retrieval. *arXiv preprint arXiv:1701.04923*, 2017.

[40] Y. Jiang, J. Meng, and J. Yuan. Randomized visual phrases for object search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3100–3107. IEEE, 2012.

[41] Y. Jiang, J. Meng, J. Yuan, and J. Luo. Randomized spatial context for object search. *IEEE Transactions on Image Processing*, 24(6):1748–1762, 2015.

[42] T. Yu, Y. Wu, S. D. Bhattacharjee, and J. Yuan. Efficient object instance search using fuzzy objects matching. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[43] W. Zhang and C.-W. Ngo. Topological spatial verification for instance search. *IEEE Transactions on Multimedia*, 17(8):1236–1247, 2015.