2017 IEEE International Conference on Computer Vision

# Compressive Quantization for Fast Object Instance Search in Videos

Tan Yu, Zhenzhen Wang, Junsong Yuan
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore
{tyu008, zwang033, jsyuan}@ntu.edu.sg

## Abstract

*Most of current visual search systems focus on image-to-image (point-to-point) search such as image and object retrieval. Nevertheless, fast image-to-video (point-to-set) search is much less exploited. This paper tackles object instance search in videos, where efficient point-to-set matching is essential. Through jointly optimizing vector quantization and hashing, we propose compressive quantization method to compress $M$ object proposals extracted from each video into only $k$ binary codes, where $k \ll M$. Then the similarity between the query object and the whole video can be determined by the Hamming distance between the query's binary code and the video's best-matched binary code. Our compressive quantization not only enables fast search but also significantly reduces the memory cost of storing the video features. Despite the high compression ratio, our proposed compressive quantization still can effectively retrieve small objects in large video datasets. Systematic experiments on three benchmark datasets verify the effectiveness and efficiency of our compressive quantization.*

## 1. Introduction

Given an image of the query object, the video-level object instance search [27, 28] is to retrieve all relevant videos in the database that contain the query object. Retrieving small objects in big videos has many applications in social media, video surveillance, and robotics, but is much less explored compared with object search in images.

Despite the recent progress of object instance search in images [15, 24, 11, 25, 31, 23, 22, 6, 3, 5, 19, 18, 37], video-level object instance search [22, 38, 36] remains a difficult problem due to the following challenges: **(1)** The query object can be a small one and appear only in a small portion of frames of a video. A direct matching between the small query object with the whole video usually cannot bring satisfactory search performance. **(2)** Different from image search which mainly addresses point-to-point matching in the feature space, object-to-video search is a point-to-set matching problem, where the query object requires to match a set of object candidates in the video to determine its relevance to that video. Considering the big video corpus of video clips, a fast point-to-set search is critical.

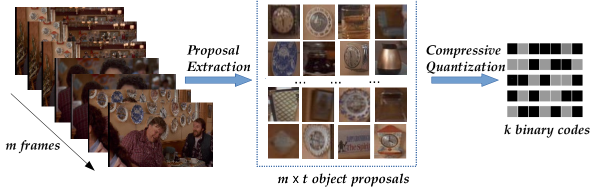


Figure 1. The overview of the proposed algorithm. Given a video of $m$ frames, we extract $t$ object proposals for each frame and further compress the $m \times t$ object proposals into $k$ binary codes using the proposed compressive quantization.

To address the above challenges, in this work, we propose to utilize the spatio-temporal redundancy of the video-data to enable fast object-to-video (*i.e.*, point-to-set) search. For each video clip in the database, we can represent it as a collection of object proposals generated from its frames. However, instead of storing all the object proposals from one video, we propose compressive quantization to compress the set of object proposals into only a few binary codes, where each binary code is a compact representative of the object proposals. As a compact representation of a set of object proposals, our compressive quantization targets at **(1)** quantizing the set of object proposals into fewer codewords and **(2)** hashing the codewords into binary codes. Nevertheless, instead of following a two-step optimization, we propose a joint optimization framework that can simultaneously optimize the distortion errors from quantization and hashing. It directly compresses $M$ object proposals into $k$ binary codes, where $k \ll M$.

As our compressive quantization simultaneously performs vector quantization and hashing, it is different from either conventional vector quantization or hashing methods. On one hand, different from conventional vector quantization [20], our compressive quantization constrains the codewords to be binary codes to achieve higher compression ratio. On the other hand, unlike hashing methods [8, 32, 10, 14] mapping one real-value vector to one binary

2380-7504/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCV.2017.85

code, our compressive quantization summarizes many real-value vectors into only a few binary codes, leading to a more compact representation of the set compared with traditional hashing methods.

As illustrated in Figure 1, using the proposed compressive quantization, a video consisting of $m \times t$ object proposals can be compactly represented by $k$ ($\ll m \times t$) binary codes. To measure the relevance of a video with the query, we only need to compare the binary code of the query with $k$ binary codes representing the video. It not only reduces the number of distance computations ($m \times t \rightarrow k$) but also enables fast search leveraging binary codes, which boosts the efficiency in both time and memory.

To evaluate the performance of our proposed compressive quantization, we perform systematic experiments on three benchmark datasets. It validates that our compressive quantization can not only provide a high compression ratio for video features, but also maintain excellent precision of searching small objects in big videos. Particularly, on the CNN2h dataset [1] consisting of 72000 frames, exhaustively comparing the query object with all the object proposals of all the videos takes $10.48$ s and $2.060$ GB memory to achieve $0.856$ mAP. In contrast, the proposed compressive quantization achieves $0.847$ mAP using only $0.008$ s and $4.39$ MB memory. It achieves $1310\times$ speed-up and $480\times$ memory reduction with comparable search precision.

## 2. Related Work

**Object Instance Search.** In the past decade, the problem of object instance search has been widely exploited on image datasets [15, 11, 30, 29, 31, 6] , while few works have focused on video datasets. A pioneering work for object instance search in videos is Video Google [28], which treats each video keyframe independently and ranks the videos by their best-matched keyframe. Some following works [39, 34] also process the frames individually and ignore the redundancy across the frames in videos. Until recently, Meng *et al.* [22] create a pool of object proposals for each video and further select the keyobjects to speed up the search. It is closely related with our work, but ours can achieve higher compression ratio since the codewords generated from the proposed compressive quantization are binary codes whereas the keyobjects from [22] are represented by high-dimension real-value vectors.

**Fast Nearest Neighbour Search.** In the past decade, we have witnessed substantial progress in fast nearest neighbour (NN) search, especially in visual search applications. We review two related methods: hashing-based binary codes and non-exhaustive search based on inverted-indexing.

Locality-Sensitive hashing [8] motivates the hashing-based binary codes. Some following works include spectral hashing [32], iterative quantization (ITQ) [10], bilinear projections (BP) [9], circulant binary embedding (CBE) [35] and sparse projection binary encoding (SPBE) [33]. The generated binary codes from hashing not only reduce the memory cost but also speed up the search process by fast Hamming distance calculation. Our work is closely related with the hashing-based compact codes. The difference is that, the hashing-based binary code maps each real-value vector into a bit vector whereas our work maps $M$ real-value vectors into $k$ bit vectors, where $k \ll M$. The compression ratio of our compressive quantization is higher since it not only makes each vector more compact but also reduces the number of vectors.

When the scale of the dataset is large, the exhaustive search is mostly infeasible to achieve satisfactory efficiency even if we adopt hashing-based binary codes. Thus, many researchers have resorted to the non-exhaustive search. The current mainstream non-exhaustive search system is based on inverted-indexing. Some representative works include IVFADC [17] and its followers such as IMI [2] and GNO-IMI [4]. IVFADC divides the data space into multiple cells through k-means and each cell will be represented by its centroid. When the query comes, IVFADC only considers the points from the cells which are close to it. In essence, IVFADC conducts the point-to-cell search to filter out some unrelated data points in order to avoid the exhaustive search. Our work is closely related with inverted-indexing method. But our method can achieve higher compression ratio since the codewords from the proposed compressive quantization are bit vectors whereas the cell centroids used in inverted-indexing are real-value vectors.

**Point-to-set Matching.** Zhu *et al.* [40] exploit the point-to-set matching problem and propose the point-to-set distance (PSD) for visual classification. However, computing PSD requires to solve a least square regression problem which is computationally demanding. The point-to-set matching has also been exploited in face video retrieval task [21]. Given a face image of one person, face video retrieval is to retrieve videos containing this person. Li *et al.* [21] aggregate the features of the frames of the video shots into a global covariance representation and learned the hashing function across Euclidean Space and Riemannian Manifold. Their method is supervised by the labelled image-video matching pairs. However, the labelled image-video matching pairs are not available in real applications. In contrast, our proposed compressive quantization method is fully unsupervised.

## 3. A Baseline by Quantization + Hashing

We denote by $\mathcal{V}_i$ the $i$-th video in the database containing $m$ frames $\{f_i^1, \cdots, f_i^m\}$. For each frame $f_i^j$, we crop $t$ object proposals. Therefore, the video $\mathcal{V}_i$ will be cropped into a set of $m \times t$ object proposals $\mathcal{S}_i = \{\mathbf{o}_i^1, \cdots, \mathbf{o}_i^{m \times t}\}$. Given a query $\mathbf{q}$, the distance between $\mathcal{V}_i$ and $\mathbf{q}$ can be determined by the best-matched object proposal in $\mathcal{S}_i$:

$$D(\mathbf{q}, \mathcal{V}_i) = \min_{\mathbf{o} \in \mathcal{S}_i} \|\mathbf{q} - \mathbf{o}\|_2. \tag{1}$$

To obtain $D(\mathbf{q}, \mathcal{V}_i)$ in Eq. (1), it requires to compare the query with all the object proposals in $\mathcal{S}_i$. Nevertheless, it ignores the redundancy among the object proposals in $\mathcal{S}_i$ and leads to heavy computation and memory cost.

**Quantization**. To exploit the redundancy among the object proposals for reducing the memory cost and speeding up the search, one straightforward way is to quantize $m \times t$ object proposals in $\mathcal{S}_i$ into $k$ codewords $\mathcal{Z}_i = \{\mathbf{z}_i^1, \cdots, \mathbf{z}_i^k\}$. In this case, we only need to compare the query $\mathbf{q}$ with codewords rather than all the object proposals in $\mathcal{S}_i$. One standard vector quantization method is k-means, which seeks to minimize the quantization errors $Q_i$:

$$Q_i = \sum_{u=1}^{k} \sum_{v \in C_i^u} \|\mathbf{o}_i^v - \mathbf{z}_i^u\|_2^2, \tag{2}$$

where $C_i^u$ denotes the set of indices of the object proposals in $\mathcal{S}_i$ that are assigned to the $u$-th cluster. Then the distance between $\mathbf{q}$ and $\mathcal{V}_i$ can be calculated by

$$D(\mathbf{q}, \mathcal{V}_i) = \min_{\mathbf{z} \in \mathcal{Z}_i} \|\mathbf{q} - \mathbf{z}\|_2. \tag{3}$$

Since $|\mathcal{Z}_i| \ll |\mathcal{S}_i|$, computing Eq. (3) is much faster than computing Eq. (1).

**Hashing**. In practice, due to the number of videos $n$ in the dataset is huge and the codewords are represented by high-dimensional real-value vectors, it is still time consuming to compare the query with all the codewords. To further speed up the search, we seek to map the codewords to Hamming space by hashing. One popular hashing method is ITQ [10], which learns the binary codes $[\mathbf{b}_1^1, \mathbf{b}_1^2, \cdots, \mathbf{b}_n^k] \in \{-1, 1\}^{l \times nk}$ and the transform matrix $\mathbf{R} \in \mathbb{R}^{d \times l}$ through minimizing the distortion error $H$:

$$H = \sum_{i=1}^{n} \sum_{u=1}^{k} \|\mathbf{z}_i^u - \mathbf{R}^\top \mathbf{b}_i^u\|_2^2, \text{ s.t. } \mathbf{R}^\top \mathbf{R} = \mathbf{I}. \tag{4}$$

We denote by $\mathcal{B}_i = \{\mathbf{b}_i^1, \cdots, \mathbf{b}_i^k\}$ the set of binary codes for the video $\mathcal{V}_i$ and denote by $\mathbf{b}_q = sign(\mathbf{Rq})$ the binary code of the query $\mathbf{q}$. In this scenario, the distance between $\mathbf{q}$ and the video $\mathcal{V}_i$ can be determined by the Hamming distance ($\| \cdot \|_H$) between the query's binary code $\mathbf{b}_q$ and the nearest binary code in $\mathcal{B}_i$:

$$D(\mathbf{q}, \mathcal{V}_i) = \min_{\mathbf{b} \in \mathcal{B}_i} \|\mathbf{b}_q - \mathbf{b}\|_H. \tag{5}$$

**Limitations.** The above two-step optimization sequentially minimizes the quantization errors of object proposals caused by quantization and the distortion errors of the codewords caused by hashing. Nevertheless, the above two-step optimization can only yield a suboptimal solution for the following reasons: **(1)** The distortion error $H$ caused by the hashing is not taken into consideration when optimizing quantization errors $\{Q_i\}_{i=1}^n$. **(2)** The quantization errors $\{Q_i\}_{i=1}^n$ are not considered when minimizing the distortion error $H$. This motivates us to jointly minimize the total errors caused by quantization and hashing in order to achieve a higher search precision.

## 4. Compressive Quantization

One straightforward way to jointly optimize the quantization and hashing is to optimize a weighted sum of the quantization errors caused by the quantization and the distortion errors caused by hashing given by:

$$J = \sum_{i=1}^{n} Q_i + \lambda H, \tag{6}$$

where $\lambda$ is the parameter controlling the weight of distortion error $H$ with respect to the quantization errors $\{Q_i\}_{i=1}^n$. However, the choice of $\lambda$ is heuristic and dataset dependent. Meanwhile, the accumulated distortion error from two separate operations is not equivalent to a simple summation of errors from each operation.

The above limitations motivate us to propose the compressive quantization scheme which directly minimizes the total distortion error $J$:

$$\begin{aligned} J = \sum_{i=1}^{n} \sum_{u=1}^{k} \sum_{v \in C_i^u} \|\mathbf{o}_i^v - \alpha \mathbf{R}^\top \mathbf{b}_i^u\|_2^2, \\ \text{s.t. } \mathbf{R}^\top \mathbf{R} = \mathbf{I}, \;\; \mathbf{b}_i^u \in \{-1, +1\}^l, \end{aligned} \tag{7}$$

where $C_i^u$ denotes the set of indices of the object proposals assigned to $u$-th cluster of $\mathcal{S}_i$; $\mathbf{b}_i^u$ denotes the binary codewords representing $C_i^u$; $\alpha \in \mathbb{R}^+$ denotes the scaling factor and $\mathbf{R} \in \mathbb{R}^{d \times d}$ denotes the rotation matrix.

Intuitively, we fix the binary codewords as the vertices of the hypercube. As illustrated in Figure 2, we rotate and stretch the hypercube to minimize the sum of the squares of the Euclidean distance from each object proposal to the corresponding vertex of the hypercube. Note that, in current formulation, we set the rotation matrix $\mathbf{R}$ as the square matrix, *i.e*, we fix the code length $l$ to be equal to the original feature's dimension $d$. Later on, we will discuss how to extend the current algorithms to the scenario when $l \neq d$.

It is easy to see the above optimization problem is highly non-convex and difficult to solve. But it is tractable to solve the problem by updating one variable with others fixed:

1. Fix $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\mathbf{R}$ and $\alpha$, update $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$.

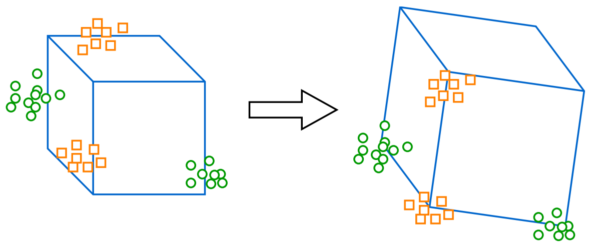


Figure 2. The green circles denote the object proposals generated from the video $\mathcal{V}_1$ and the orange squares denote the object proposals from the video $\mathcal{V}_2$, we rotate and stretch the blue hypercube to minimize the distance from each object proposal to the corresponding vertex of the hypercube the object proposal is assigned to. After the optimization, the corresponding vertices of the hypercube will be the binary codewords representing $\mathcal{V}_1$ and $\mathcal{V}_2$.

2. Fix $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$, update $\mathbf{R}$.
3. Fix $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\mathbf{R}$ and $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$, update $\alpha$.
4. Fix $\mathbf{R}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$, update $\{C_i^u\}_{i=1,u=1}^{n,k}$.

### 4.1. Fix $\{C_i^u\}_{i=1,u=1}^{n,k}$, R and $\alpha$, update $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$.

When $C_i^u$, $\mathbf{R}$ and $\alpha$ are fixed, the optimal value of $\mathbf{b}_i^u$ can be obtained by solving the below optimization problem:

$$\begin{aligned} \mathbf{b}_i^u &= \underset{\mathbf{b}\in\{-1,1\}^{d\times 1}}{\operatorname{argmin}} \sum_{v\in C_i^u} \|\mathbf{o}_i^v - \alpha\mathbf{R}^\top\mathbf{b}\|_2^2 \\ &= \underset{\mathbf{b}\in\{-1,1\}^{d\times 1}}{\operatorname{argmin}} \sum_{v\in C_i^u} \|\alpha^{-1}\mathbf{x}_i^v - \mathbf{b}\|_2^2, \end{aligned} \tag{8}$$

where $\mathbf{x}_i^v = \mathbf{R}\mathbf{o}_i^v$. We denote by $b_i^u(t)$ the value of $t$-th bit of the above optimized $\mathbf{b}_i^u$ and denote by $x_i^u(t)$ the value of $t$-th dimension of $\mathbf{x}_i^u$. Below we will prove that

$$b_i^u(t) = sign(\sum_{v\in C_i^u} x_i^v(t)). \tag{9}$$

Note that $b_i^u(t)$ has only two possible values $\{-1,+1\}$. We denote by $l_{+1}(t) = \sum_{v\in C_i^u}(\alpha^{-1}x_i^v(t)-1)^2$ the distortion error when the $t$-th bit of $\mathbf{b}_i^u$ is set to be $+1$ and denote by $l_{-1}(t) = \sum_{v\in C_i^u}(\alpha^{-1}x_i^v(t)+1)^2$ the distortion error when the $t$-th bit of $\mathbf{b}_i^u$ is set to be $-1$ . We can further obtain

$$l_{+1}(t) - l_{-1}(t) = -2\alpha^{-1}\sum_{v\in C_i^u} x_i^v(t). \tag{10}$$

Since $\alpha > 0$,

$$\min\{l_{+1}(t), l_{-1}(t)\} = \begin{cases} l_{-1}(t), & \text{if } \sum_{v\in C_i^u} x_i^v(t) < 0 \\ l_{+1}(t), & \text{if } \sum_{v\in C_i^u} x_i^v(t) > 0 \end{cases} \tag{11}$$

Thereafter,

$$b_i^u(t) = \begin{cases} -1, & \text{if } \sum_{v\in C_i^u} x_i^v(t) < 0 \\ +1, & \text{if } \sum_{v\in C_i^u} x_i^v(t) > 0 \end{cases} \tag{12}$$

### 4.2. Fix $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$, update $\mathbf{R}$.

We arrange the object proposals of all the videos $\{\mathbf{o}_1^1, \mathbf{o}_1^2, \cdots, \mathbf{o}_n^{mt}\}$ as columns in matrix $\mathbf{O} \in \mathbb{R}^{d\times nmt}$. We denote by $\mathbf{B} = [\mathbf{b}_1^{A(\mathbf{o}_1^1)}, \mathbf{b}_1^{A(\mathbf{o}_1^2)}, \cdots, \mathbf{b}_n^{A(\mathbf{o}_n^m)}]$ the corresponding binary codes of the object proposals, where $A(\mathbf{o}_i^v)$ denotes the index of the cluster which $\mathbf{o}_i^v$ is assigned to. When $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$ are fixed, optimizing $\mathbf{R}$ is an orthogonal procrustes problem [12] solved by

$$\mathbf{R} = \mathbf{V}\mathbf{U}^\top, \tag{13}$$

where $\mathbf{U}$ and $\mathbf{V}$ are obtained by the singular value decomposition (SVD) of $\mathbf{O}\mathbf{B}^\top$. Note that computing $\mathbf{O}\mathbf{B}^\top$ takes $\mathcal{O}(nmtdl)$ complexity. Since $mt$, the number of object proposals per video, is relatively large, this step is considerably time consuming. Below we optimize our algorithm to speed up the process of computing $\mathbf{O}\mathbf{B}^\top$. We rewrite $\mathbf{O}\mathbf{B}^\top$ into:

$$\begin{aligned} \mathbf{O}\mathbf{B}^\top &= \sum_{i=1}^{n}\sum_{v=1}^{m} \mathbf{o}_i^v \mathbf{b}_i^{A(\mathbf{o}_i^v)}{}^\top \\ &= \sum_{i=1}^{n}\sum_{u=1}^{k}\sum_{v\in C_i^u} \mathbf{o}_i^v \mathbf{b}_i^{u\top} \\ &= [\sum_{v\in C_1^1} \mathbf{o}_i^v, \cdots, \sum_{v\in C_i^u} \mathbf{o}_i^v][\mathbf{b}_1^1, \cdots, \mathbf{b}_n^k]^\top \\ &= [\mathbf{y}_1^1, \cdots, \mathbf{y}_n^k][\mathbf{b}_1^1, \cdots, \mathbf{b}_n^k]^\top, \end{aligned} \tag{14}$$

where $\mathbf{y}_i^u = \sum_{v\in C_i^u} \mathbf{o}_i^v$ has already been obtained in Eq. (9). Therefore, we can calculate $\mathbf{O}\mathbf{B}^\top$ through $[\mathbf{y}_1^1, ..., \mathbf{y}_n^k][\mathbf{b}_1^1, ..., \mathbf{b}_n^k]^\top$, which takes only $\mathcal{O}(nkdl)$ complexity. Considering the number of codewords ($k$) per video is significantly smaller than the number of object proposals ($mt$) per video, the complexity is largely reduced.

### 4.3. Fix $\{C_i^u\}_{i=1,u=1}^{n,k}$, R and $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$, update $\alpha$.

We directly compute the first-order and second-order derivative of $J$ with respect to $\alpha$:

$$\begin{aligned} \frac{\partial J}{\partial \alpha} &= 2\sum_{i=1}^{n}\sum_{u=1}^{k}\sum_{v\in C_i^u} (-\mathbf{R}^\top\mathbf{b}_i^u)^\top(\mathbf{o}_i^v - \alpha\mathbf{R}^\top\mathbf{b}_i^u), \\ \frac{\partial^2 J}{\partial^2 \alpha} &= 2\sum_{i=1}^{n}\sum_{u=1}^{k}\sum_{v\in C_i^u} (-\mathbf{R}^\top\mathbf{b}_i^u)^\top(-\mathbf{R}^\top\mathbf{b}_i^u). \end{aligned} \tag{15}$$

Since $\partial^2 J/\partial^2\alpha = 2\sum_{i=1}^{n}\sum_{u=1}^{k}\sum_{v\in C_i^u}\|\mathbf{R}^\top\mathbf{b}_i^u\|_2^2 > 0$, by setting $\partial J/\partial\alpha = 0$, we can obtain the optimal value of the scaling coefficient $\alpha$ which minimizes $J$:

$$\begin{aligned}\alpha =&\frac{\sum_{i=1}^{n}\sum_{u=1}^{k}\mathbf{b}_i^{u\top}\sum_{v\in C_i^u}\mathbf{R}\mathbf{o}_i^v}{\sum_{i=1}^{n}\sum_{u=1}^{k}\sum_{v\in C_i^u}\mathbf{b}_i^{u\top}\mathbf{R}\mathbf{R}^\top\mathbf{b}_i^u}\\ =&\frac{\sum_{i=1}^{n}\sum_{u=1}^{k} sign(\sum_{v\in C_i^u}\mathbf{R}\mathbf{o}_i^v)^\top\sum_{v\in C_i^u}\mathbf{R}\mathbf{o}_i^v}{\sum_{i=1}^{n}\sum_{u=1}^{k}\sum_{v\in C_i^u}\mathbf{b}_i^{u\top}\mathbf{b}_i^u}\\ =&\frac{\sum_{i=1}^{n}\sum_{u=1}^{k}\|\sum_{v\in C_i^u}\mathbf{R}\mathbf{o}_i^v\|_1}{nmt}.\end{aligned} \tag{16}$$

### 4.4. Fix $\mathbf{R}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$, update $\{C_i^u\}_{i=1,u=1}^{n,k}$.

When $\mathbf{R}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$ are fixed, the optimal $\{C_i^u\}_{i=1,u=1}^{n,k}$ can be obtained by simply assigning each object proposal $\mathbf{o}_i^v$ to its nearest vertex of hypercube by

$$A(\mathbf{o}_i^v) = \underset{u}{\operatorname{argmin}}\,\|\mathbf{o}_i^v - \alpha\mathbf{R}^\top\mathbf{b}_i^u\|_2. \tag{17}$$

### 4.5. Initialization and Implementation Details

We initialize $\{C_i^u\}_{i=1,u=1}^{n,k}$ by conducting k-means clustering for $\{\mathcal{S}_i\}_{i=1}^n$ and initialize the rotation matrix $\mathbf{R}$ using the centroids of $\{C_i^u\}_{i=1,u=1}^{n,k}$ through ITQ. The whole optimization process is summarized in Algorithm 1.

**Algorithm 1** Compressive Quantization

```
Input: $n$ sets of object proposals $\{\mathcal{S}_i\}_{i=1}^n$ generated from
    videos $\{\mathcal{V}_i\}_{i=1}^n$, number of codewords $k$ per video.
Output: The rotation matrix $\mathbf{R}$, the binarized codewords
    $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$.
Initialize the clusters $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\mathbf{R}$, $\alpha$.
repreat
    for $t$ = 1 ... $T$
        Update $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ using Eq. (9).
        Update $\mathbf{R}$ using Eq. (13).
    end
    Update $\alpha$ using Eq. (16).
    Update $\{C_i^u\}_{i=1,u=1}^{n,k}$ using Eq. (17).
until converge
return $\mathbf{R}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$.
```

Our algorithm iteratively updates $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\mathbf{R}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$. Note that there is an inner loop within the outer loop, we set the number of iterations of the inner loop $T = 10$. The aim of the inner loop is to ensure that $\mathbf{R}$ and $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ are able to reach their local optimality before updating $\{C_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$. Since each update of $\{C_i^u\}_{i=1,u=1}^{n,k}$, $\mathbf{R}$, $\{\mathbf{b}_i^u\}_{i=1,u=1}^{n,k}$ and $\alpha$ can strictly reduce the distortion error $J$, our algorithm can ensure the convergence. Our experiments show that it can converge within 50 iterations of outer loops on three datasets. Like other alternative-optimization algorithms, our algorithm only achieves a local optima and is influenced by the initialization.

**Discussion**: In the above formulation, we fix $\mathbf{R}$ as a square matrix, *i.e.*, we constrain the length of code $l$ to the dimension of the original feature $d$. However, we still can deal with the situation when $l \neq d$ with pre-processing operations.

Particularly, when $l < d$, we will pre-process the original features using principle component analysis (PCA) to reduce their dimensionality from the $d$ to $l$. After that, we can conduct the proposed compressive composition using the reduced features. It will obtain $l$-bit binary codes by the square rotation matrix $\mathbf{R} \in \mathbb{R}^{l\times l}$. On the other hand, if $l > d$, we project the original feature $\mathbf{o}$ to $l$-dimension feature $\hat{\mathbf{o}}$ by a random orthogonal matrix $\mathbf{P} \in \mathbb{R}^{l\times d}$ :

$$\hat{\mathbf{o}} = \mathbf{P}\mathbf{o}, \ \text{ s.t. } \mathbf{P}^\top\mathbf{P} = \mathbf{I}. \tag{18}$$

After projection, the Euclidean distance between two points in original feature space $\|\mathbf{o}_1 - \mathbf{o}_2\|_2$ can be preserved in the projected feature space since

$$\|\hat{\mathbf{o}}_1 - \hat{\mathbf{o}}_2\|_2^2 = (\mathbf{o}_1 - \mathbf{o}_2)^\top\mathbf{P}^\top\mathbf{P}(\mathbf{o}_1 - \mathbf{o}_2) = \|\mathbf{o}_1 - \mathbf{o}_2\|_2^2. \tag{19}$$

We can further conduct the proposed compressive quantization using the projected data samples to obtain $l$-bit binary codes by the square rotation matrix $\mathbf{R} \in \mathbb{R}^{l\times l}$.

### 4.6. Relation to Existing Methods

**K-means** [20] is a widely used vector quantization method. It minimizes the quantization errors by iteratively updating the assignments of the data samples and the centroids of the clusters. Our compressive quantization also seeks to minimize the quantization errors, but the codewords generated from our method are binary codes.

**Iterative Quantization** [10] is a popular hashing method. It learns the rotation matrix by minimizing the distortions between the data samples with their rotated binary codes. Like other hashing methods, it maps a real-value vector $\mathbf{x}$ to a binary code $\mathbf{b}$. The proposed compressive quantization also learns a rotation matrix, but we map a set of real-value vectors $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$ to another set of binary codes $\mathcal{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_k\}$, where $|\mathcal{B}| \ll |\mathcal{X}|$.

**K-means Hashing** [13] is another hashing method, which minimizes a weighted summation of the quantization errors cause by k-means and the affinity errors caused by hashing. Its formulation is very similar to Eq. (6), which is different from ours formulated as Eq. (7). Meanwhile, K-means Hashing maps one real-value vector to a binary code. In contrast, our compressive quantization maps $M$ real-value vectors to $k$ ($\ll M$) binary codes, which can achieve significantly higher compression ratio.

(a) Groundhog Day

(b) CNN2h

(c) Egocentric

Figure 3. Examples of query objects.

# 5. Experiment

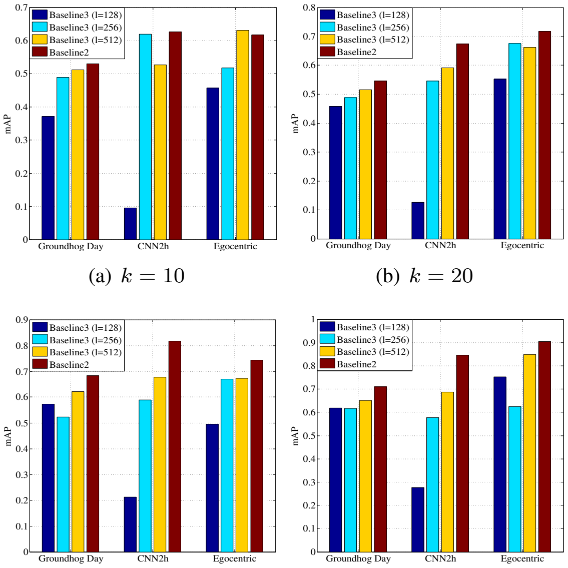
## 5.1. Settings, Dataset and Evaluation Metric

We adopt Edge Boxes [41] to generate the object proposals for each frame of the videos. For each frame, we extract 300 object proposals. For each object proposal, we further extract its feature by max-pooling the last convolutional layer of VGG-16 CNN model [26] pre-trained on Imagenet dataset. The max-pooled 512-dimensional features are further post-processed by principle component analysis (PCA) and whitening in order to suppress the burstiness [16] and the dimension of the feature is fixed as 256. Observing the object proposals from the same frame heavily overlap with each other, we use k-means to group 300 object proposals into 30 clusters. We select the centroids of the cluster as the compact object proposals. Given a video of $n$ frames, we will obtain $30n$ object proposals.

We conduct the experiments on Groundhog Day [28], CNN2h [1] and Egocentric [7] datasets. The Groundhog Day contains 5640 keyframes. We equally divide them into 56 short videos. The CNN2h dataset contains $72,000$ frames and we equally divide them into 720 short videos. The Egocentric dataset consists of 19 long videos. We uniformly sample the keyframes per 20 frames, obtain 51804 keyframes and further equally divide all the keyframes into 517 videos. The Groundhog Day dataset provides six query objects. On CNN2h and Egocentric datasets, we use eight query objects and skip those scene query images. Figure 3 visualizes the query objects of three datasets. The effectiveness of the proposed method is evaluated by mean average precision (mAP).

## 5.2. Baseline Evaluations

Before evaluating the effectiveness of the proposed compressive quantization, we first evaluate three baselines. We denote by **Baseline1** the exhaustive matching scheme defined as Eq. (1). It is considerably memory demanding and computationally costly since it requires to compare the query with all the object proposals of the videos.



(a) $k = 10$

(b) $k = 20$

(c) $k = 40$

(d) $k = 100$

Figure 4. mAP comparison between Baseline2 and Baseline3.

We denote by **Baseline2** the algorithm based on k-means defined as Eq. (3). We vary the number of codewords $k$ per video among 10, 20, 40 and 100. As shown in Table 1, the precision obtained by Baseline2 improves as the number of codewords $k$ per video increases. Meanwhile, Baseline2 achieves comparable mAP with Baseline1 with significantly less computation and memory cost. For instance, on the CNN2h dataset, the mAP achieved by Baseline1 is 0.856 using 10.48 seconds and 2.060GB memory, whereas Baseline2 achieves 0.817 mAP when $k = 40$ using only 0.147 seconds and 28.12 MB memory. The promising results achieved by Baseline2 is expected since there exists heavy redundancy among the object proposals from the same video.

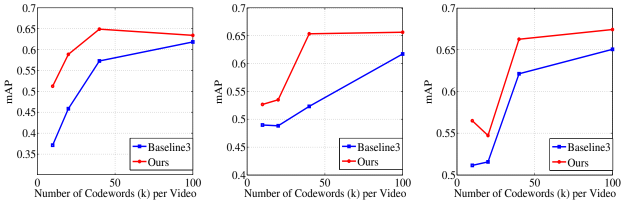
We denote by **Baseline3** the algorithm based on k-means followed by iterative quantization defined as Eq. (5). Note that Baseline3 conducts the quantization and hashing sequentially. We vary the number of codewords $k$ per video among 10, 20, 40 and 100 and vary the code length $l$ of the binary codewords among 128, 256 and 512. As we can see from Figure 4, the mAP achieved by Baseline3 is considerably lower than that of Baseline2. This is expected since the ITQ brings additional distortion errors, which can significantly deteriorate the search precision if the accumulated distortion errors from k-means and ITQ are large. Particularly, on the CNN2h dataset, when $k = 100$, Baseline2 achieves 0.845 mAP but Baseline3 ($l$ = 512) only achieves 0.687 mAP.

## 5.3. Evaluations of Compressive Quantization

In this section, we evaluate the performance of the proposed compressive quantization. To demonstrate its effectiveness, we directly compare it with Baseline2 and Baseline3, respectively.

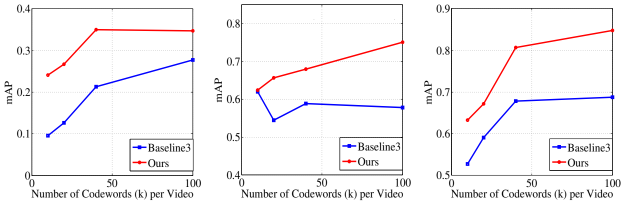
|  |  | Groundhog Day |  |  | CNN2h |  |  | Egocentric |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
|  |  | Memory | Time | mAP | Memory | Time | mAP | Memory | Time | mAP |
| Baseline1 |  | 165.2 MB | 1.398 s | **0.724** | 2.060 GB | 10.48 s | **0.856** | 1.518 GB | 8.826 s | **0.974** |
| Baseline2 | k = 10 | 0.547 MB | 0.004 s | 0.530 | 7.031 MB | 0.035 s | 0.625 | 5.049 MB | 0.026 s | 0.617 |
|  | k = 20 | 1.094 MB | 0.006 s | 0.545 | 14.06 MB | 0.069 s | 0.674 | 10.10 MB | 0.049 s | 0.717 |
|  | k = 40 | 2.188 MB | 0.012 s | 0.682 | 28.12 MB | 0.147 s | 0.817 | 20.20 MB | 0.084 s | 0.744 |
|  | k = 100 | 5.469 MB | 0.031 s | **0.709** | 70.31 MB | 0.314 s | **0.845** | 50.49 MB | 0.226 s | **0.905** |

Table 1. The memory, time and mAP comparisons of Baseline1 and Baseline2 on three datasets.



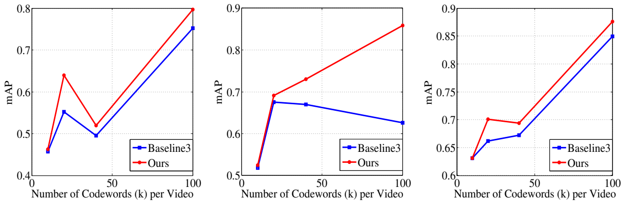
(a) $l = 128$ (b) $l = 256$ (c) $l = 512$

Figure 5. mAP comparison of the proposed compressive quantization (ours) with Baseline3 on the Groundhog Day dataset.



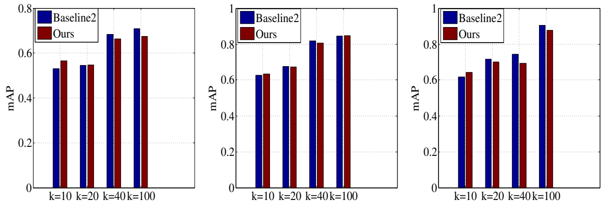
(a) $l = 128$ (b) $l = 256$ (c) $l = 512$

Figure 6. mAP comparison of the proposed compressive quantization (ours) with Baseline3 on the CNN2h dataset.



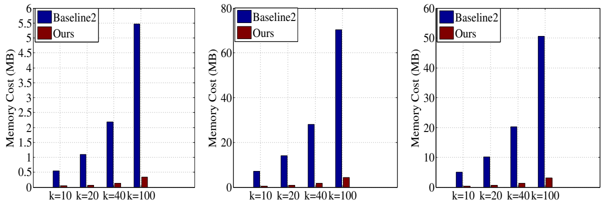
(a) $l = 128$ (b) $l = 256$ (c) $l = 512$

Figure 7. mAP comparison of the proposed compressive quantization (ours) with Baseline3.



(a) Groundhog Day (b) CNN2h (c) Egocentric

Figure 8. mAP comparison between the proposed compressive quantization (ours) and Baseline2.



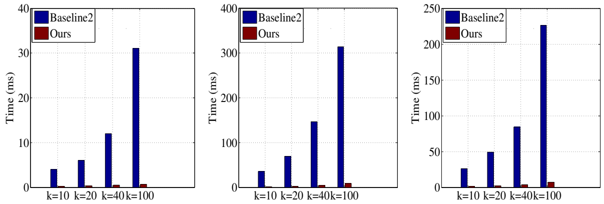
(a) Groundhog Day (b) CNN2h (c) Egocentric

Figure 9. Memory comparison between the proposed compressive quantization (ours) and Baseline2.

Figure 5, 6, 7 compare the performance of the proposed compressive quantization with Baseline3 on Groundhog Day, CNN2h and Egocentric datasets. In our implementation, we vary the code length ($l$) among 128, 256 and 512 and vary the number of codewords ($k$) per video among 10, 20, 40 and 100. As we can see from Figure 5, 6, 7 that, our compressive quantization significantly outperforms Baseline3 when the memory and computation cost are the same. Particularly, on the Groundhog Day dataset, when $k = 40$ and $l = 256$, our compressive quantization achieves 0.654 mAP whereas Baseline3 only obtains 0.523 mAP. On the CNN2h dataset, when $k = 40$ and $l = 256$, our compressive quantization achieves 0.751 mAP whereas Baseline3 only obtains 0.578 mAP.

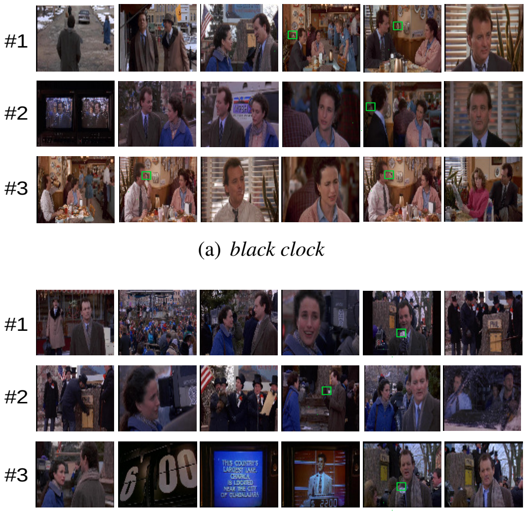
Figure 8 compares the mAP of the proposed compressive quantization with Baseline2 on Groundhog Day, CNN2h and Egocentric datasets. In our implementation, we fix the code length ($l$) as 512 and vary the number of codeword ($k$) per video among 10, 20, 40 and 100. We can observe from Figure 8 that, the precision achieved by the proposed compressive quantization is comparable with that from Baseline2 implemented by k-means. For example, on the CNN2h dataset, when $k = 100$, our compressive quantization has a mAP of 0.847, which is as good as Baseline2 which achieves 0.846 mAP. Meanwhile, it is also comparable with Baseline1 (0.856 mAP). The slightly worse precision of ours is expected because of the inevitable distortion errors caused by the quantization and the binarization steps.

Figure 9 compares the memory cost of the proposed compressive quantization with Baseline2 on three datasets. As shown in Figure 9, the proposed compressive quantization requires much less memory than Baseline2. In fact, Baseline2 requires to store 256-dimension real-value codewords whereas the codewords from the proposed compressive quantization are only 512-bit. The memory cost of our compressive quantization is only $1/16$ of that of Baseline2. Particularly, on the CNN2h dataset, when $k = 100$,



(a) Groundhog Day (b) CNN2h (c) Egocentric

Figure 10. Time comparison between the proposed compressive quantization (ours) and Baseline2.



(a) *black clock*

(b) *microphone*

Figure 11. Top-3 search results of *black clock* and *microphone*.

the memory cost from Baseline2 is 70.32 MB, whereas our compressive quantization only takes 4.39 MB. Compared with Baseline1 using 2.060 GB, the proposed compressive quantization brings 480× memory reduction.

Figure 10 compares the time cost of the proposed compressive quantization with Baseline2 on three datasets. As shown in Figure 10, the search based on our compressive quantization is significantly faster than that of Baseline2. This is attributed to the efficiency of Hamming distance computation. Particularly, on the CNN2h dataset, when $k = 100$, Baseline2 requires 314 milliseconds to conduct the search, whereas our compressive quantization only takes 8 milliseconds, which is around 39× faster. Meanwhile, compared with Baseline1 which takes 10.48s, the acceleration of the proposed compressive quantization is around 1310×. Note that the reported time cost does not consider the encoding time of the query, since it can be ignored compared to the search time when dealing with huge datasets.

In Figure 11, we visualize top-3 retrieved videos from the proposed compressive quantization method using queries *black clock* and *microphone* on the Groundhog Day dataset. For each video, we shows 6 keyframes uniformly sampled from it. We use the green bounding boxes to show the location of the query object. As shown in Figure 11, although the object only occupies a very small area in relevant frames, we are still able to retrieve the correct videos.

| | Groundhog Day | | | CNN2h | | |
|---|---|---|---|---|---|---|
| | 128 | 256 | 512 | 128 | 256 | 512 |
| LSH [8] | 0.393 | 0.526 | 0.533 | 0.313 | 0.376 | 0.473 |
| SH [32] | 0.512 | 0.632 | 0.615 | 0.242 | 0.731 | 0.771 |
| BP [9] | 0.576 | 0.586 | 0.617 | 0.317 | 0.691 | 0.645 |
| CBE [35] | 0.491 | 0.590 | 0.665 | 0.289 | 0.607 | 0.791 |
| SPBE [33] | 0.578 | 0.536 | 0.603 | 0.295 | 0.599 | 0.696 |
| Ours | **0.635** | **0.656** | **0.674** | **0.347** | **0.751** | **0.847** |

Table 2. Comparison of our method with k-means followed by other hashing methods.

## 5.4. Comparison with Other Hashing Methods

We also compare our method with k-means followed by other hashing methods, *e.g.*, Locality-Sensitivity Hashing (LSH) [8], Spectral Hashing (SH) [32], Bilinear Projections (BP) [9], Circulant Binary Embedding (CBE) [35], and Sparse Projection Binary Encoding (SPBE) [33]. All of the above methods are conducted on the codewords obtained from k-means. We fix the number of codewords $k$ per video as 100 and vary the code length $l$ among 128, 256 and 512. As shown in Table 2, the mAP achieved by ours outperforms all above-mentioned hashing methods. When $l = 512$, ours achieves 0.847 mAP, whereas the second best mAP is only 0.791 achieved by CBE. The excellent performance of our compressive quantization is attributed to the joint optimization mechanism, since all other hashing methods optimize the hashing and quantization independently.

# 6. Conclusion

To remedy the huge memory and computation cost caused by leveraging object proposals in video-level object instance search task, we propose the compressive quantization. It compresses thousands of object proposals from each video into tens of binary codes but maintains good search performance. Thanks to our compressive quantization, the number of distance computations are significantly reduced and every distance calculation can be extremely fast. Experimental results on three benchmark datasets demonstrate that our approach achieves comparable performance with exhaustive search using only a fraction of memory and significantly less time cost.

**Acknowledgements:** This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114 and Tier 1 RG27/14. This research was carried out at the ROSE Lab at the Nanyang Technological University 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 our research at the ROSE Lab.

## References

[1] A. Araujo, M. Makar, V. Chandrasekhar, D. Chen, S. Tsai, H. Chen, R. Angst, and B. Girod. Efficient video search using image queries. In *ICIP*, pages 3082–3086, 2014.

[2] A. Babenko and V. Lempitsky. The inverted multi-index. In *CVPR*, pages 3069–3076, 2012.

[3] A. Babenko and V. Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, pages 1269–1277. IEEE, 2015.

[4] A. Babenko and V. Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *CVPR*, pages 2055–2063, 2016.

[5] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *ECCV*, pages 584–599. Springer, 2014.

[6] S. D. Bhattacharjee, J. Yuan, W. Hong, and X. Ruan. Query adaptive instance search using object sketches. In *Proc. of the ACM on Multimedia Conference*, pages 1306–1315, 2016.

[7] V. Chandrasekhar, W. Min, X. Li, C. Tan, B. Mandal, L. Li, and J. Hwee Lim. Efficient retrieval from large-scale egocentric visual data using a sparse graph representation. In *CVPR Workshops*, pages 527–534, 2014.

[8] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. of the 21th Annual Symposium on Computational geometry*, pages 253–262. ACM, 2004.

[9] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, pages 484–491, 2013.

[10] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011.

[11] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *ECCV*, pages 241–257. Springer, 2016.

[12] J. C. Gower. Procrustes methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):503–508, 2010.

[13] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, pages 2938–2945, 2013.

[14] W. Hong, J. Yuan, and S. D. Bhattacharjee. Fried binary embedding for high-dimensional visual features. In *CVPR*, 2017.

[15] A. Iscen, T. Giorgos, A. Yannis, F. Teddy, and C. Ondrej. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representation. In *CVPR*, 2017.

[16] H. Jégou and O. Chum. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *ECCV*, pages 774–787. Springer, 2012.

[17] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. In *ICASSP*, pages 861–864, 2011.

[18] Y. Jiang, J. Meng, and J. Yuan. Randomized visual phrases for object search. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3100–3107. IEEE, 2012.

[19] 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.

[20] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on PAMI*, 24(7):881–892, 2002.

[21] Y. Li, R. Wang, Z. Huang, S. Shan, and X. Chen. Face video retrieval with image query via hashing across euclidean space and riemannian manifold. In *CVPR*, pages 4758–4767, 2015.

[22] J. Meng, H. Wang, J. Yuan, and Y.-P. Tan. From keyframes to key objects: Video summarization by representative object proposal selection. In *CVPR*, pages 1039–1048, 2016.

[23] E. Mohedano, A. Salvador, K. McGuinness, F. Marques, N. E. O'Connor, and X. Giro-i Nieto. Bags of local convolutional features for scalable instance search. *arXiv preprint arXiv:1604.04653*, 2016.

[24] F. Radenović, G. Tolias, and O. Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *ECCV*, pages 3–20. Springer, 2016.

[25] A. Salvador, X. Giró-i Nieto, F. Marqués, and S. Satoh. Faster r-cnn features for instance search. In *CVPR Workshops*, pages 9–16, 2016.

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

[27] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, volume 2, pages 1470–1477, 2003.

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

[29] R. Tao, E. Gavves, C. G. Snoek, and A. W. Smeulders. Locality in generic instance search from one example. In *CVPR*, pages 2091–2098, 2014.

[30] R. Tao, A. W. Smeulders, and S.-F. Chang. Attributes and categories for generic instance search from one example. In *CVPR*, pages 177–186, 2015.

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

[32] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009.

[33] Y. Xia, K. He, P. Kohli, and J. Sun. Sparse projections for high-dimensional binary codes. In *CVPR*, pages 3332–3339, 2015.

[34] Y. Yang and S. Satoh. Efficient instance search from large video database via sparse filters in subspaces. In *ICIP*, pages 3972–3976. IEEE, 2013.

[35] F. Yu, S. Kumar, Y. Gong, and S.-f. Chang. Circulant binary embedding. In *ICML*, pages 946–954, 2014.

[36] T. Yu, J. Meng, and J. Yuan. Is my object in this video? reconstruction-based object search in video. In *IJCAI*, 2017.

[37] T. Yu, Y. Wu, S. D. Bhattacharjee, and J. Yuan. Efficient object instance search using fuzzy objects matching. In *AAAI*, pages 4320–4326, 2017.

[38] T. Yu, Y. Wu, and J. Yuan. Hope: Hierarchical object prototype encoding for efficient object instance search in videos. In *CVPR*, 2017.

[39] C.-Z. Zhu and S. Satoh. Large vocabulary quantization for searching instances from videos. In *Proc. of the ACM ICMR*, page 52, 2012.

[40] P. Zhu, L. Zhang, W. Zuo, and D. Zhang. From point to set: Extend the learning of distance metrics. In *ICCV*, pages 2664–2671. IEEE, 2013.

[41] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405. Springer, 2014.