Yi Xie, Hui-min Yu, Roland Hu, 2014. Probabilistic hypergraph based hash codes for social image search. *Journal of Zhejiang University-SCIENCE C (Computers & Electronics),* **15**(7):537-550. [doi:10.1631/jzus.C1300268]

Probabilistic hypergraph based hash codes for social image search

Key words: Hypergraph Laplacian, Pribabilistic hypergraph, Hash codes, Image search

Corresponding author: Yi Xie, Hui-min Yu
E-mail: yixie@zju.edu.cn; yhm2005@zju.edu.cn

Motivation

- In the last decade, images and video have been widely stored and shared on the Internet. Billions of images pose a challenge for conventional image search methods.

- Exploring new representations and approaches to search large-scale datasets has become an urgent research issue.

Proposed approach (I)

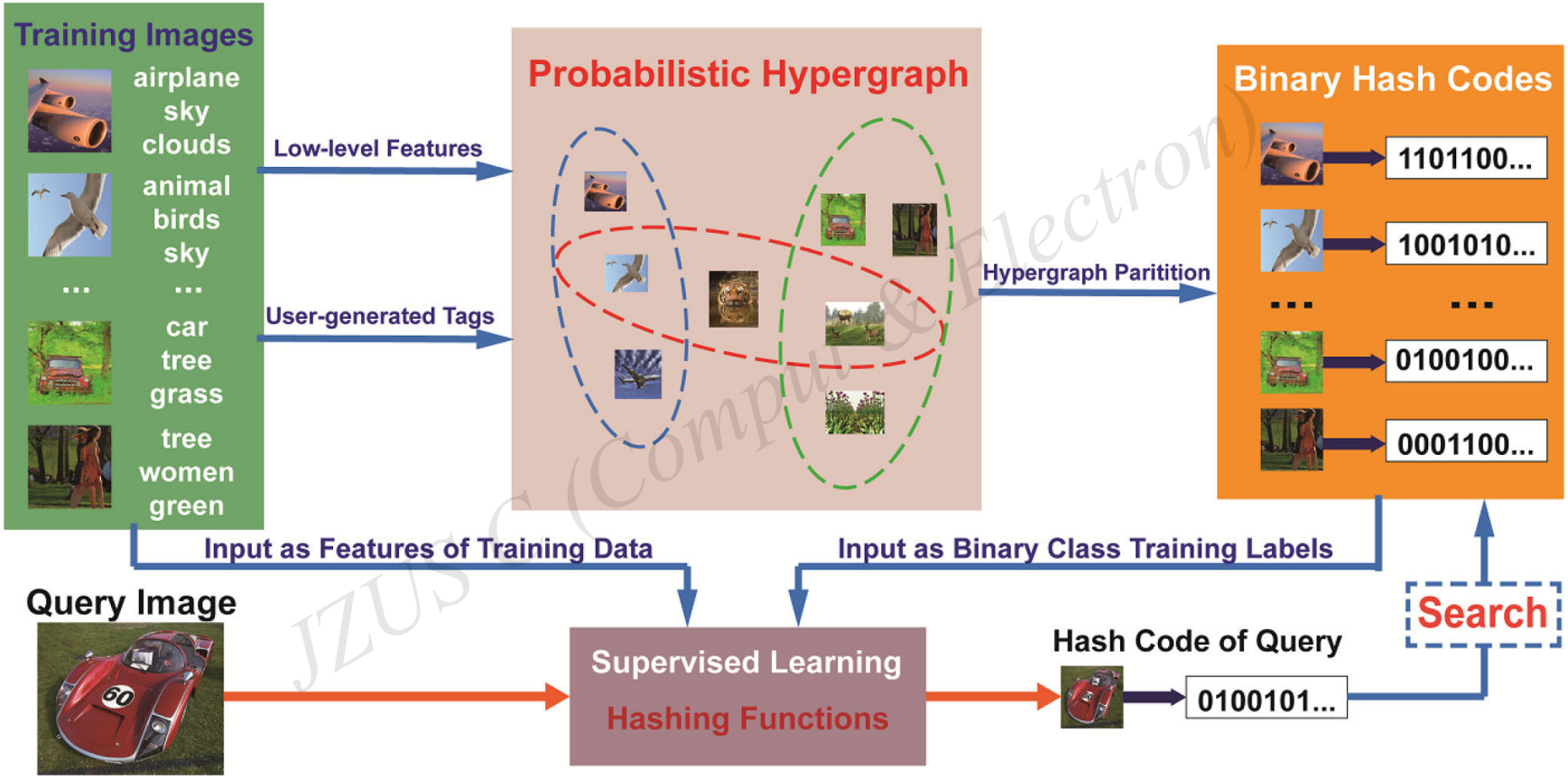

Fig. 1 Schematic of the probabilistic hypergraph based hashing algorithm

Proposed approach (II)

Algorithm Probabilistic hypergraph based hashing

Input: Social images $v_1, v_2, \ldots, v_n$.

Output: Binary codes $\boldsymbol{F}$ and hashing function $\Phi(\boldsymbol{x})$.

1 Generate bag-of-visual-words $\boldsymbol{y}_i^{\text{bow}}$ and tag information vector $\boldsymbol{y}_i^{\text{tag}}$ for each social image v_i in the dataset;

2 Construct probabilistic hypergraph $G=(V, E, w)$;

3 Compute the affinity coefficients for all the vertices in each content-based hyperedge using Eq. (7);

4 Assign affinity coefficients for vertices in tag-based hyperedges to 0.8 or 1;

5 Compute the hyperedge weight by Eqs. (8) and (9);

6 Compute the incidence matrix $\boldsymbol{H}_{\text{p}}$ using Eq. (2) and diagonal matrices $\boldsymbol{D}_{\text{v}}$, $\boldsymbol{D}_{\text{e}}$ using Eqs. (3) and (4);

7 Compute $\Delta = \boldsymbol{I} - \boldsymbol{D}_{\text{v}}^{-1/2}\boldsymbol{H}\boldsymbol{W}\boldsymbol{D}_{\text{e}}^{-1}\boldsymbol{H}^{\text{T}}\boldsymbol{D}_{\text{v}}^{-1/2}$

8 Generate $\boldsymbol{F}$ by computing the k eigenvectors of $\boldsymbol{\Delta}$ with minimal eigenvalues;

9 Threshold $\boldsymbol{F}$ to 1 and −1;

10 **for** $i \leftarrow 1$ to k **do**

11 Use the ith column of $\boldsymbol{F}$ as binary labels to train the SVM classifier $\Phi^i(\boldsymbol{x})$;

12 **end for**

13 Concatenate $\Phi^1(\boldsymbol{x}), \Phi^2(\boldsymbol{x}), \ldots, \Phi^k(\boldsymbol{x})$ to build the hashing function $\Phi(\boldsymbol{x})$.

Major results

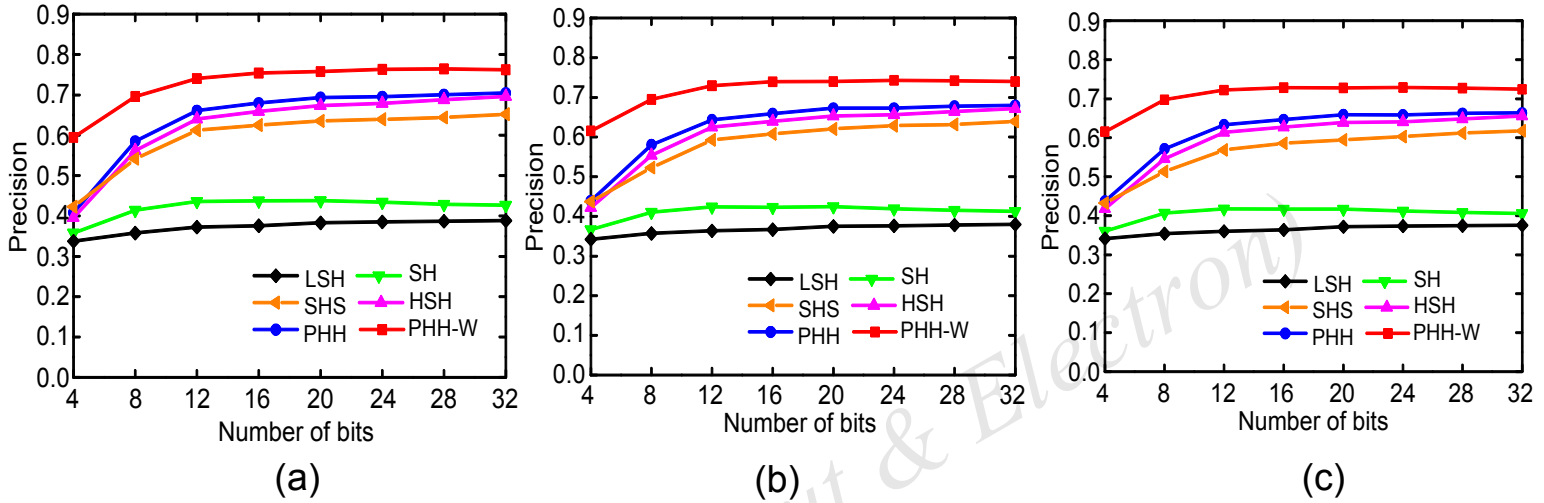

Fig. 5 Precision of the top 50 (a), 100 (b), and 150 (c) neighbors at different numbers of bits

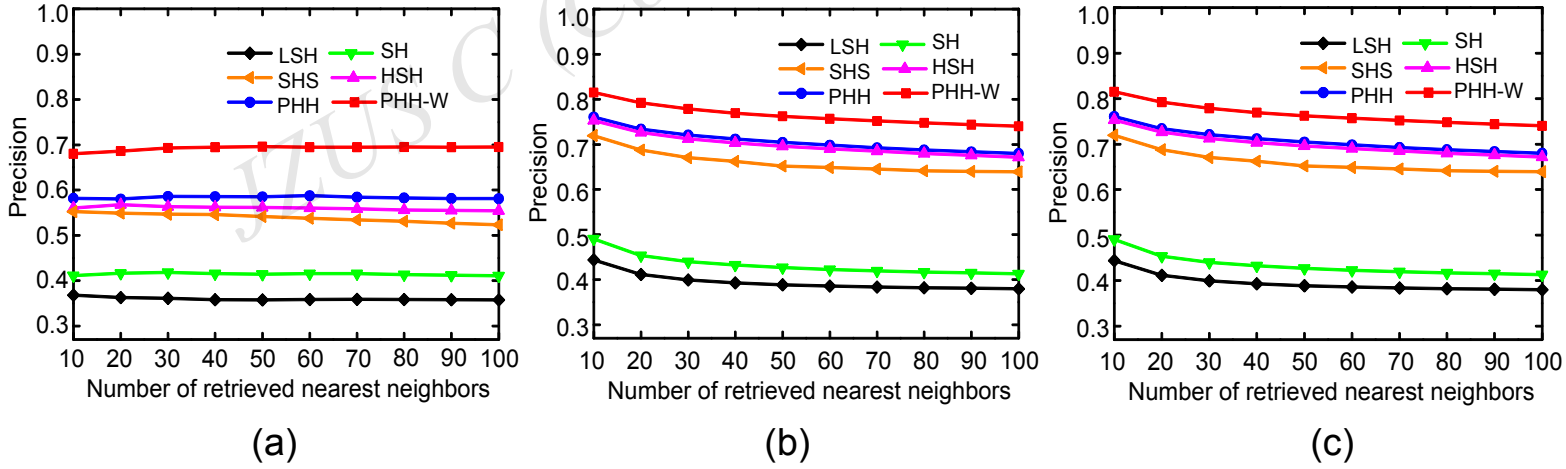

Fig. 6 Precision under different numbers of retrieved nearest neighbors at 8-bit (a), 16-bit (b), and 32-bit (c)