COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

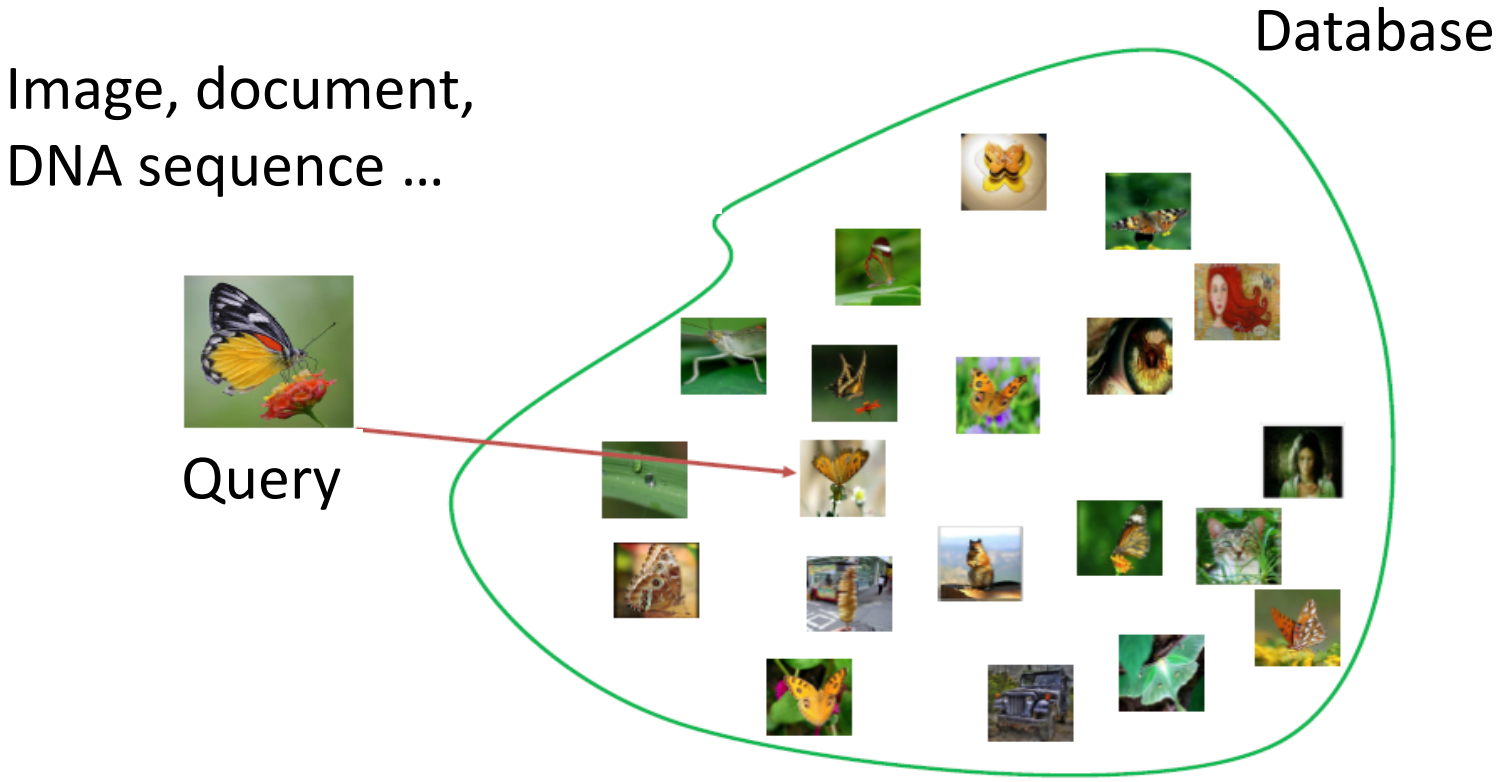
Recent Advances of Compact Hashing for Large-Scale Visual Search

Shih-Fu Chang
Columbia University
January 2015

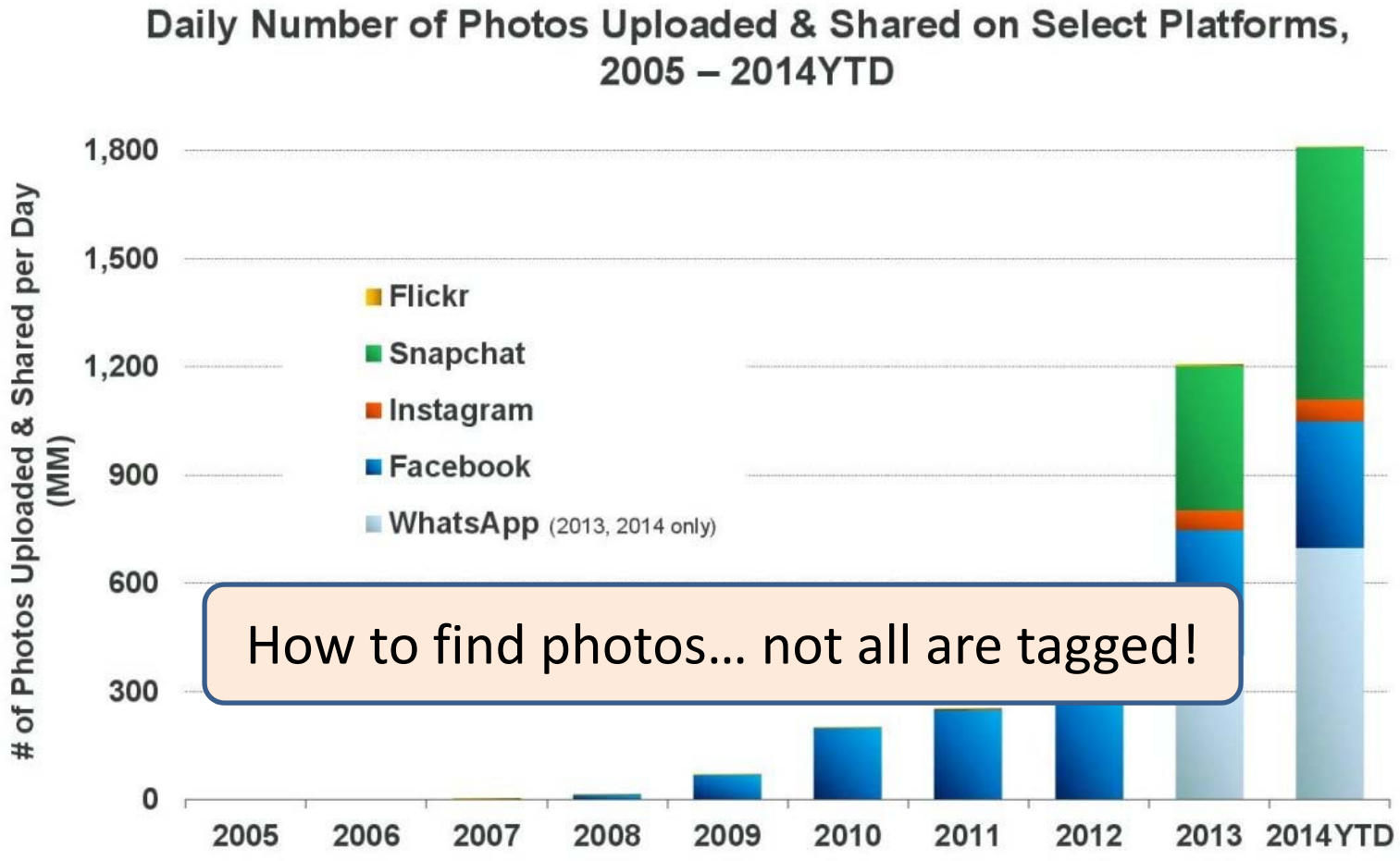
Joint work with Sanjiv Kumar (Google),
Wei Liu, Jun Wang (IBM Research), Junfeng He (Facebook)
Felix Yu, Guangnan Ye, Dong Liu (Columbia U)

Large-Scale Similarity Search

- Need for image search, document retrieval, biology data matching, network construction, …
- Scale up to billions or trillions of samples



The Explosive Data Growth
-- 1.8B photos uploaded/shared per day



@KPCB Source: KPCB estimates based on publicly disclosed company data, 2014 YTD data per latest as of 5/14.

Visual Search Example

TinEye
Reverse Image Search

GIF, 479x536, 20.8 KB

141 Results

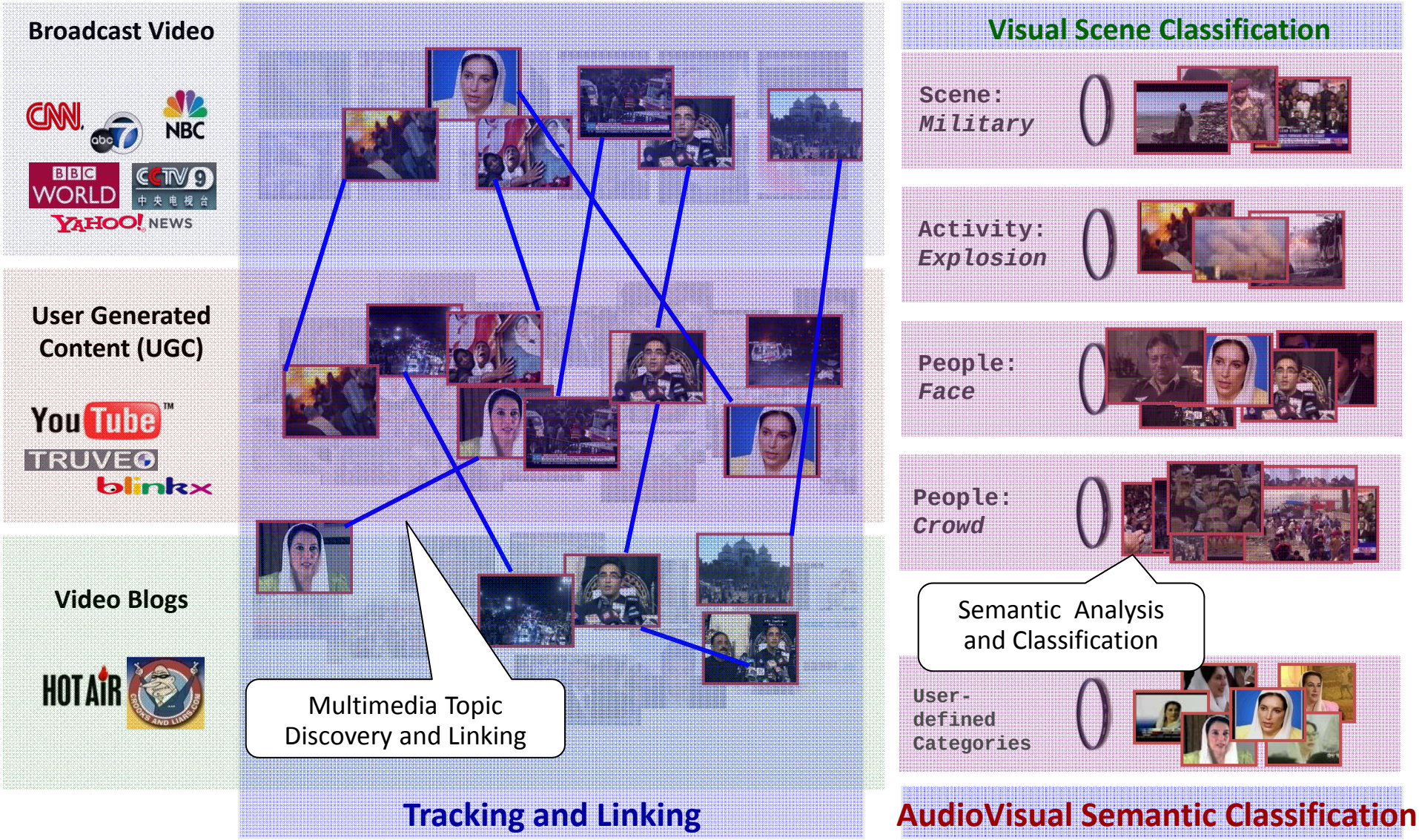
Searched over 8.463 billion images in 0.008 seconds.

for file: http://content.sportslogos.net/logos/35/882/full/2671.gif

- These results expire in 72 hours. Why?
- Share a success story!
- TinEye is free to use for non-commercial purposes.

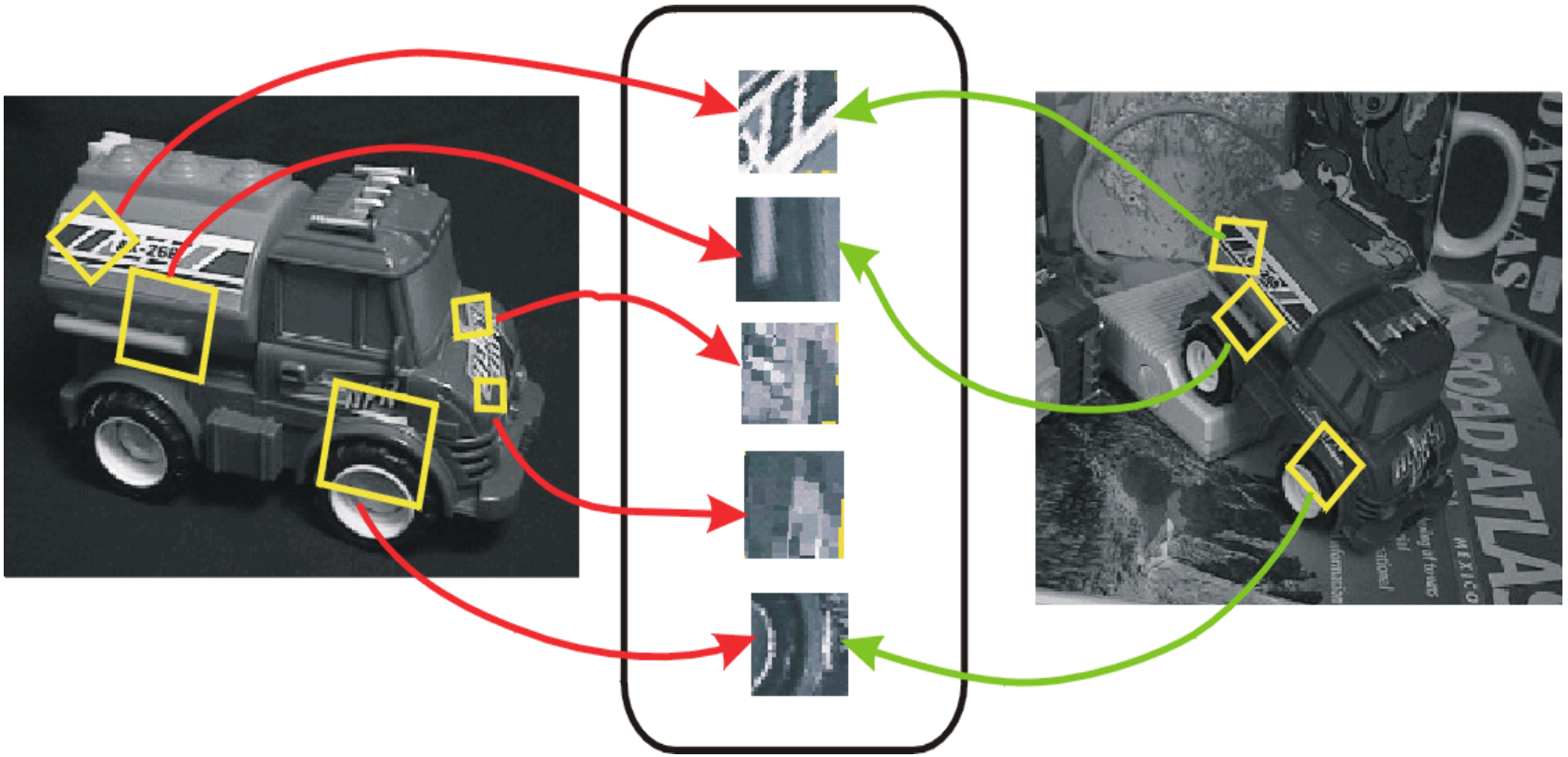
Columbia NewsRover System

linking video over 100+ channels (10 millions images per hour)



Fine-Grained Visual Match

- Image patches (e.g., SIFT, SURF) often used in image search: invariant to geometric and photometric transformations
- Scale grows up to billions or trillions



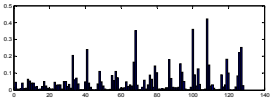
Slide: David Lowe

Estimate the Complexity

- 500 image patches per photo
  - Feature size ~128 K Bytes
- Database
  - 5 billions patches for 10 million images
  - Finding matched patches becomes challenging
- Also hard to do this over mobile devices …

Mobile Visual Search

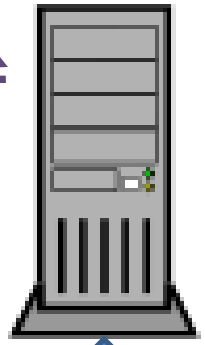
1. Take a picture

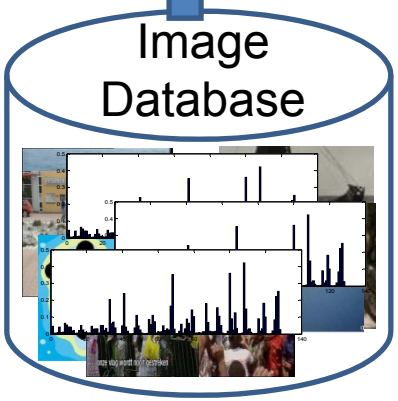


2. Send image or features

3. Send via mobile networks

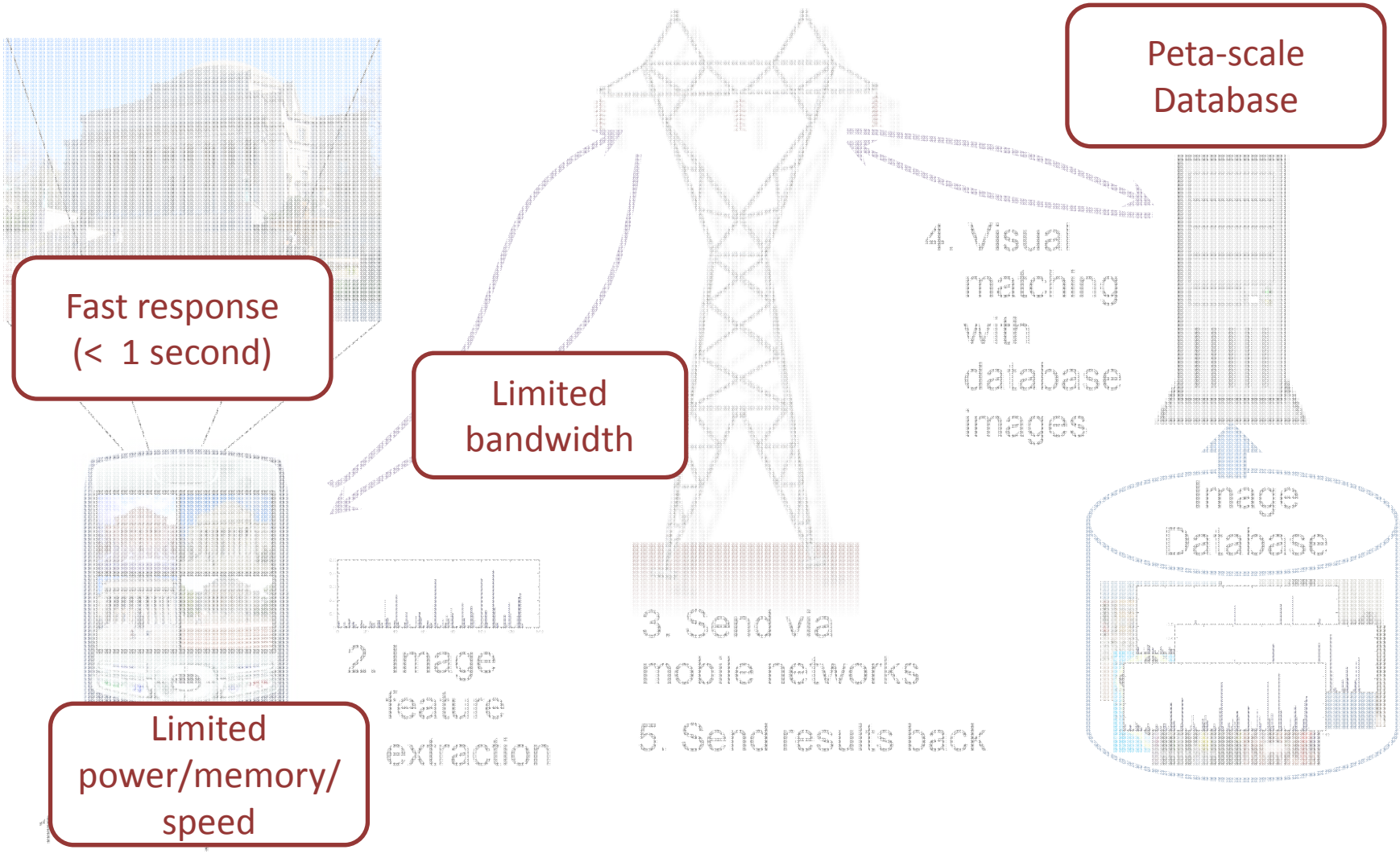
4. Visual matching with database images





5. Send results back

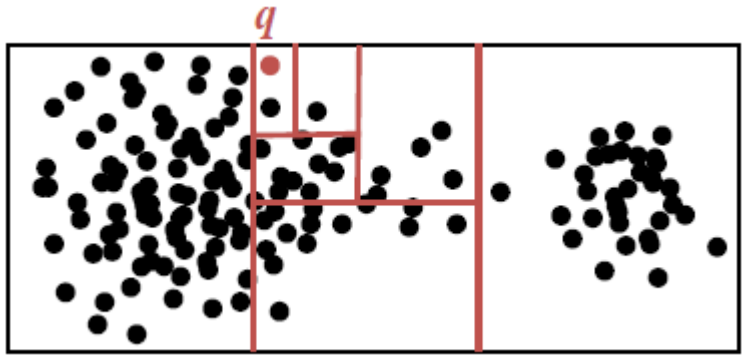
Challenges of MVS

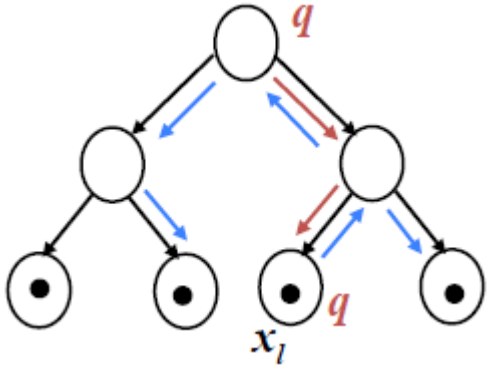


Needs of Fast Scalable Search

- Fast index code generation
  - Minimize online query processing time
- Avoid *O(N)* complexity of exhaustive approach
  - Sublinear or constant search complexity over database
  - Efficient storage

Traditional Indexing: K-D Tree

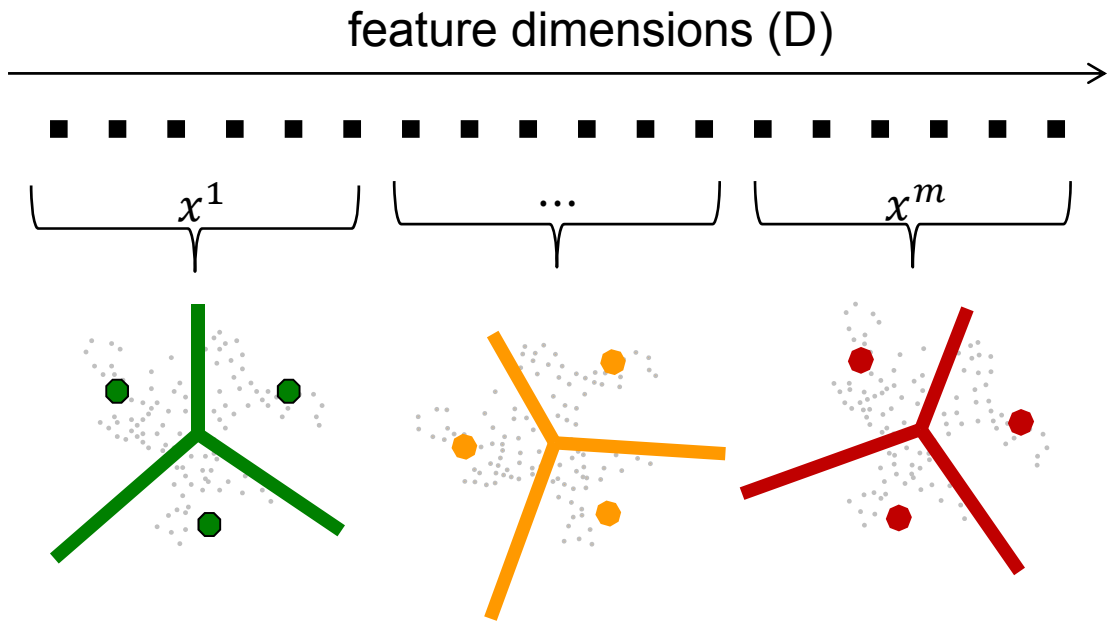




- Popular Public Source Vlfeat, FLANN
- Threshold in max variance or random dimension at each node
- Search: best-fit-branch-first, backtrack when needed
- Search time cost: $O(c*\log n)$
- But backtrack prohibitive when dimension is high (Curse of dimensionality)
- Another issue: exponential storage

Product Quantization

Jegou, Douze, Schmid, PAMI 2011



divide to
m subvectors

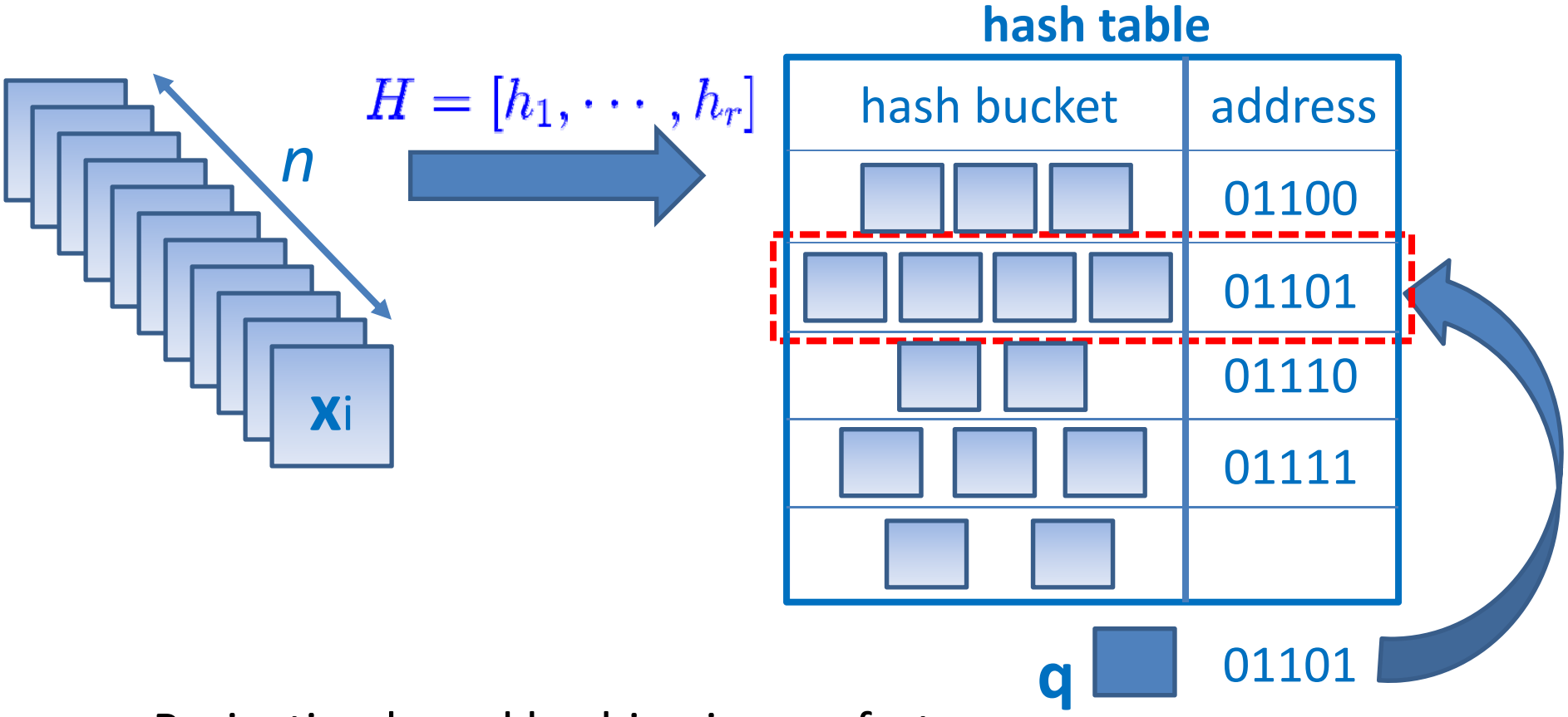
$k^{1/m}$ clusters in
each subspace

- Avoid exponential codebook size by product of subspace codebooks
- Efficient storage, only $mk^{1/m}$ codewords (3,000, m=3, k=1 billion)
- Exhaustive scan of codewords possible

$$d(q, w_i) \cong d\left(q, {w_i}^1\right) + d\left(q, {w_i}^2\right) + d(q, {w_i}^3)$$

- Drawback: high query processing cost

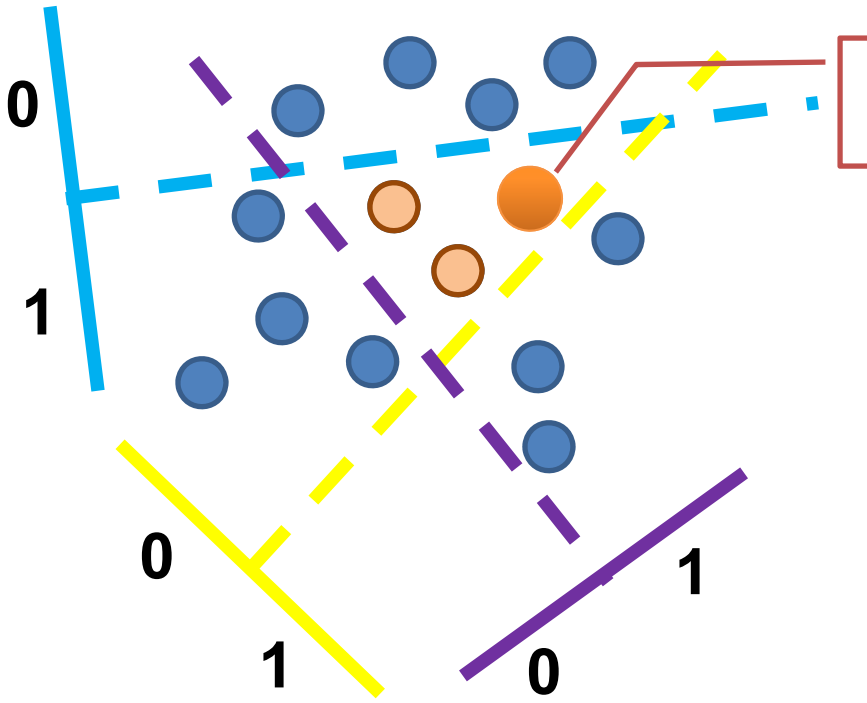
Hash Table based Search



- Projection based hashing is very fast
- *O*(1) search time by table lookup
- Optional step to include items in Hamming neighborhoods

Locality-Sensitive Hashing

[Indyk and Motwani 1998] [Datar et al. 2004]



101 **Index by compact code**

hash function

$$h(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^\top \mathbf{x} + b)$$

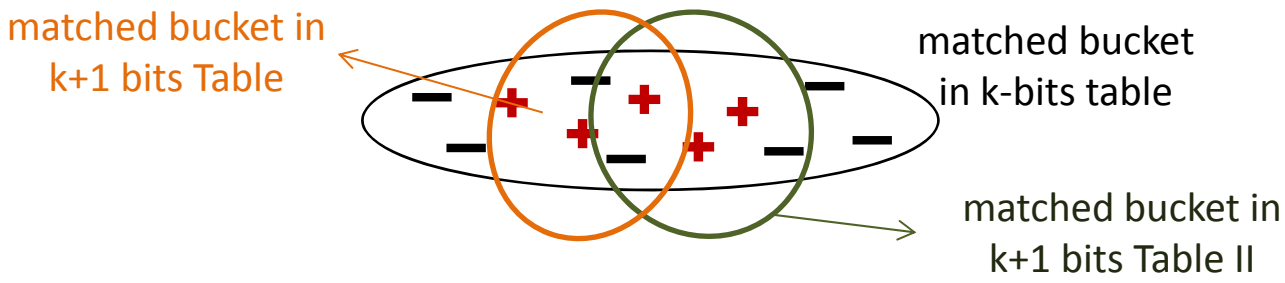
random

$$P\{H(\mathbf{x}) = H(\mathbf{y})\} = l \cdot \left[1 - \frac{cos^{-1}\mathbf{x}^\top \mathbf{y}}{\pi}\right]^K$$

- collision probability proportional to original similarity
  l: # hash tables, K: hash bits per table

Multi-Table Hashing

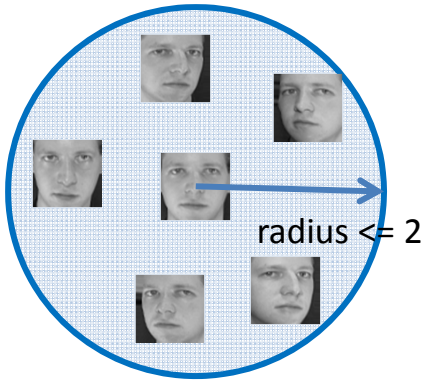
- Longer table increases precision but degrades recall

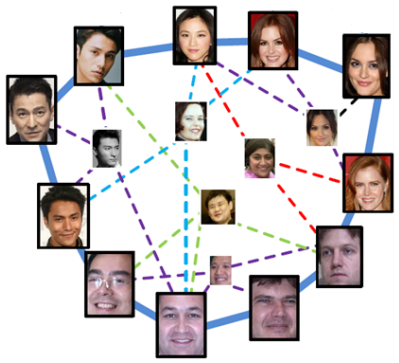


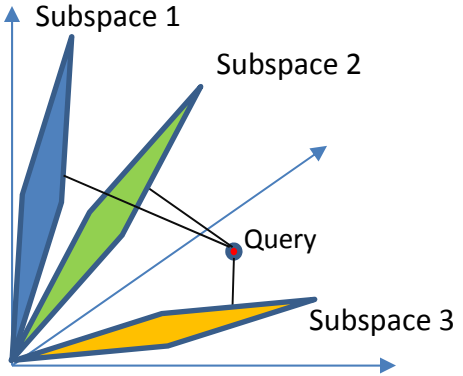
- Common practice: multi-table hashing
- Union of multi-table results increases precision and keeps recall
- But the number of hash bits 2X: bad for mobile

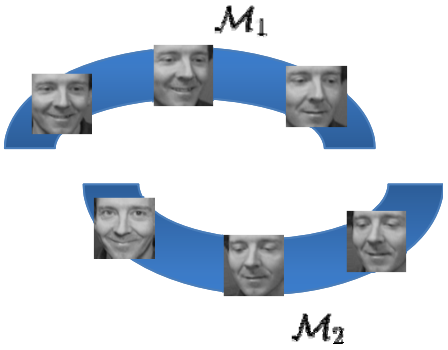
Beyond Point-to-Point Search

- Diverse Data: feature vectors, graphs, subspace, manifolds, dictionaries, etc.
- Search: range search, graph search, point-to-subspace, manifold to manifold, etc.



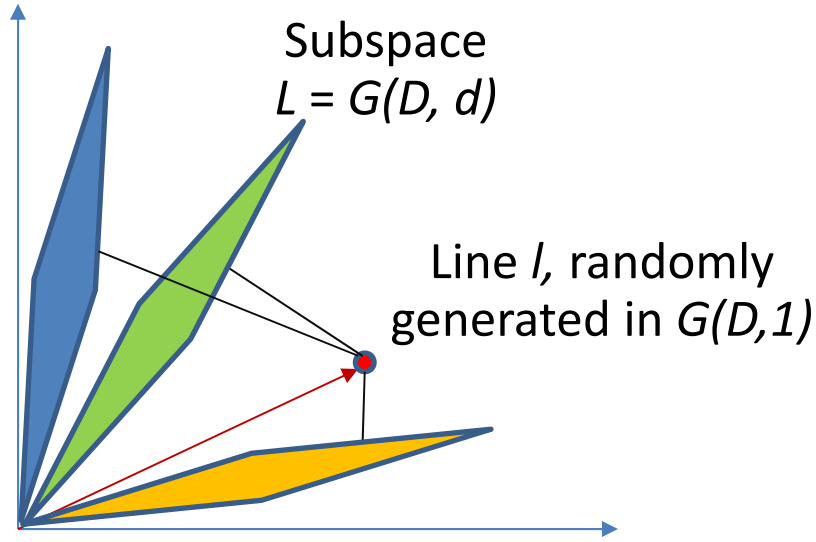






Locality-Sensitive Hashing for Subspace

(Wang et al, ICCV '13)

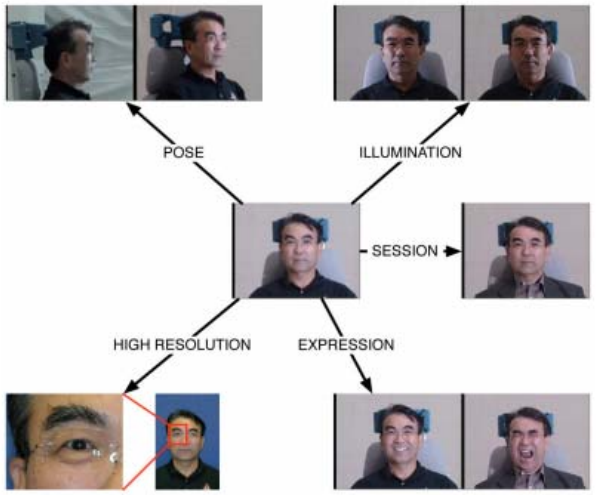


Generate K random lines, each producing a hash bit:

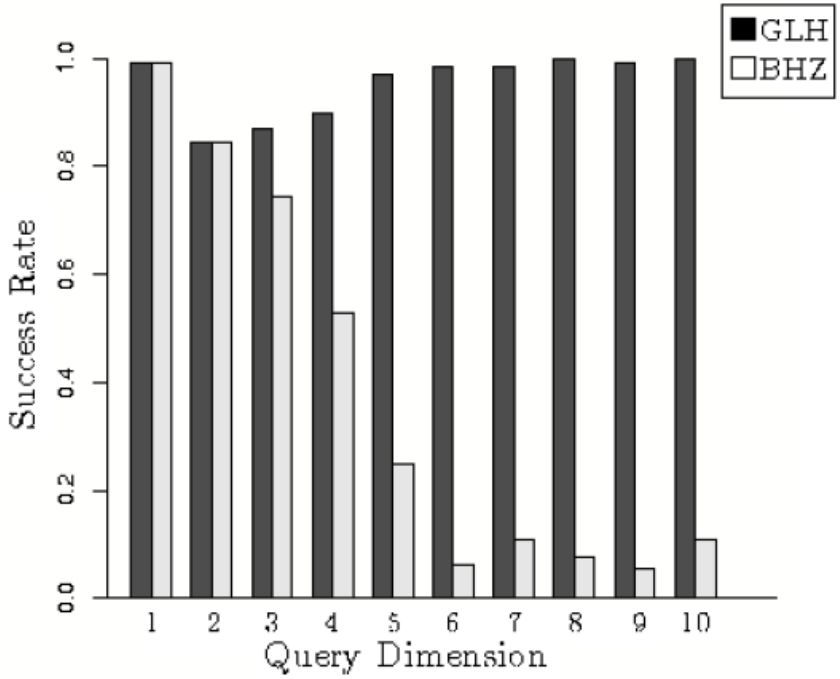
$$h_{l,\theta_0} = \begin{cases} 0, dist_G(l, L) > \theta_0 \\ 1, dist_G(l, L) \leq \theta_0 \end{cases}$$

- Preserve locality sensitivity, derive probability of Approximate NN
- Matching by hash table look-up (*O(1)* complexity)
- Fast index code generation

Subspace Hashing over Face Set Database



Multi-PIE data: 750,000 face images of 337 people under pose-illumination-expression variations

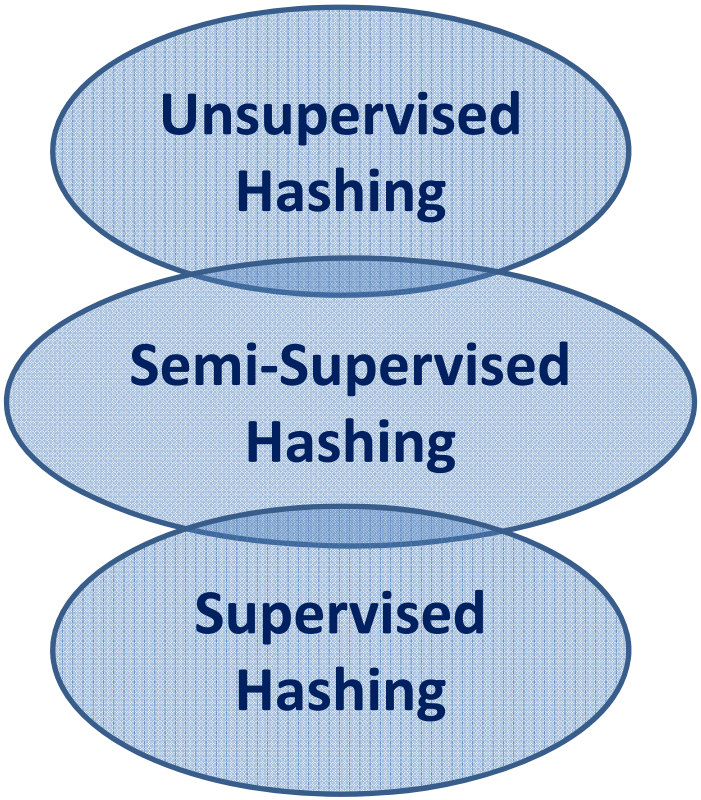


- Treat image set of the same person as a subspace
- 60 hash bits per subspace
- Given a single image or image set as query, find the right subspace

So far, these are random projections…

Focus: Learning-Based Hashing

Fit data distribution and structure



SH ‘08, KLSH ‘09,
AGH ’10, PCAH, ITQ ’11,
DGH ‘14

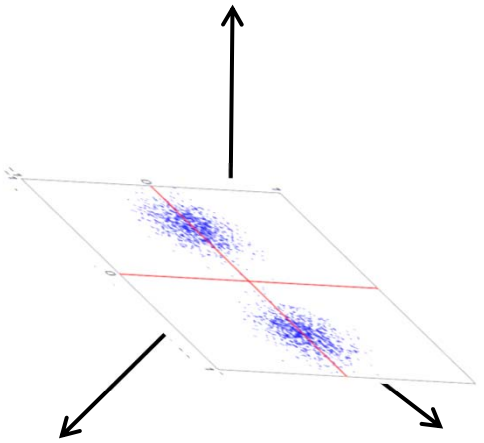
SSH ‘10, WeaklySH ‘10

Explore additional information

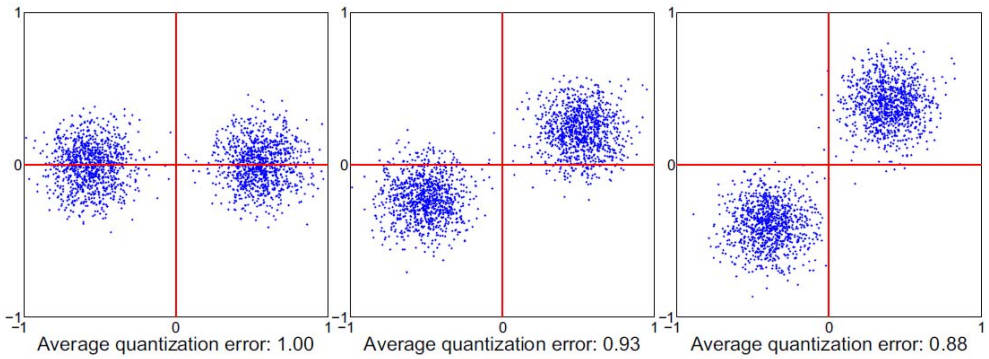
RBM ‘09, BRE ‘10,
MLH ‘11, LDAH ’11,
ITQ ‘11, KSH ’12, PHC’13,
VH’14

A Very Simple Form of Unsupervised Learning

- Find PCA bases as hash projection functions
- Maximize variance (discrimination) in each hash bit



- Rotate projections to minimize quantization errors (Gong&Lazebnik '11)

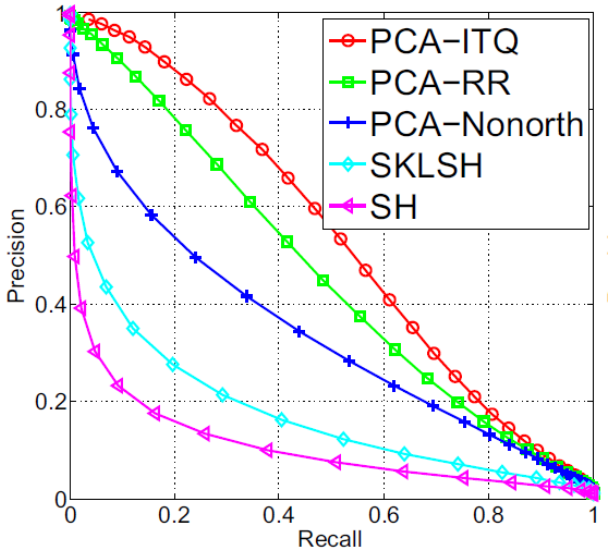


(a) PCA aligned. (b) Random Rotation. (c) Optimized Rotation.

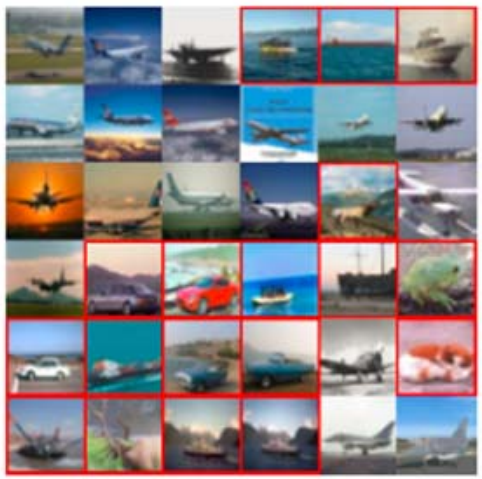
Data-dependent vs. Random Hashing

- 580K tiny images

PCA-ITQ, Gong&Lazebnik, CVPR 11



(b) Recall precision curve@64 bits.



Precision: 0.5

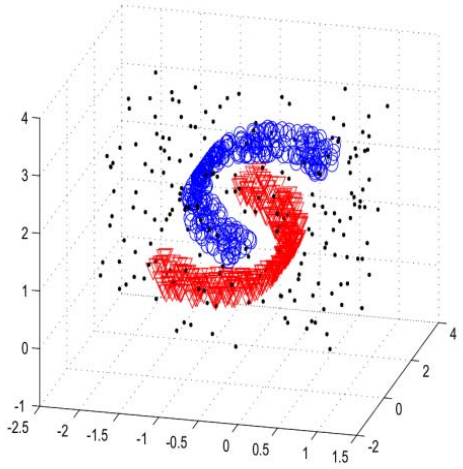
PCA-random rotation

Precision: 0.58

PCA-ITQ optimal alignment

Find Additional Structures in Data

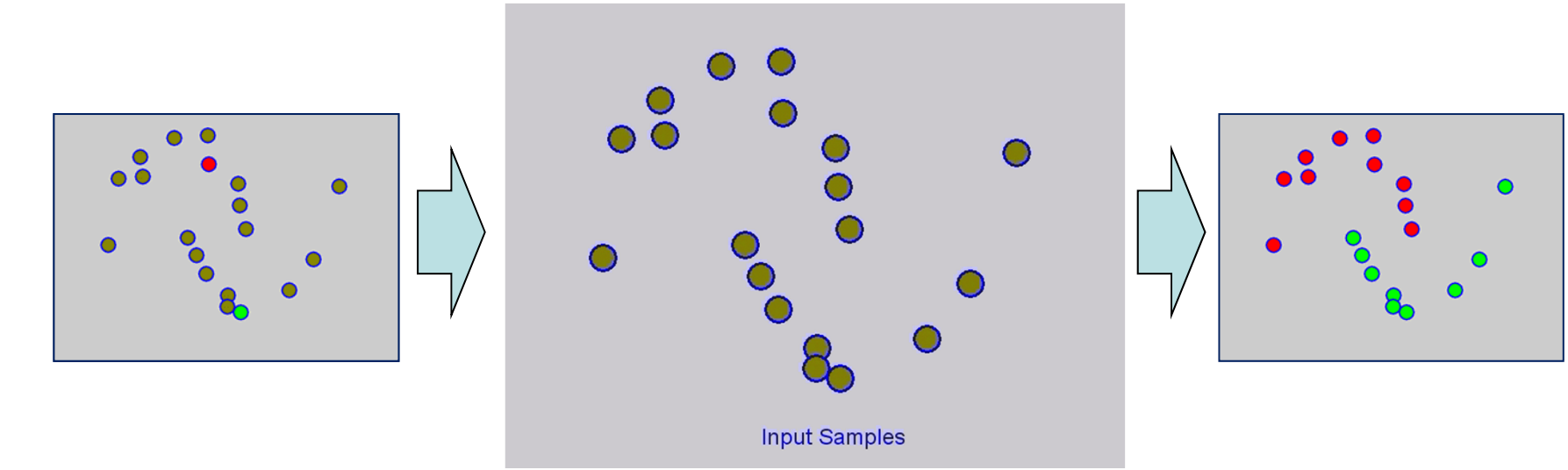
- Images are not random in the feature space
  - Explore (nonlinear) Structures



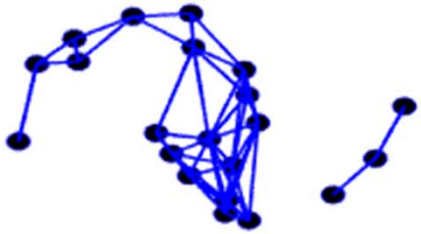
- Such structures are useful for
  - visualization, retrieval, label propagation

Tool: Sparse Graphs

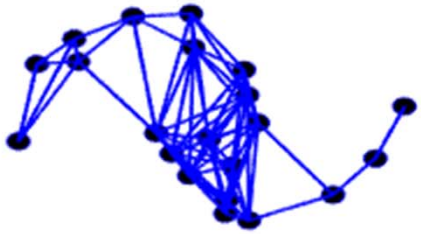
- Build sparse graphs with **local** connectivity
- Use it to find approximate NN and propagate labels



Active Research: Constructing Sparse Graphs

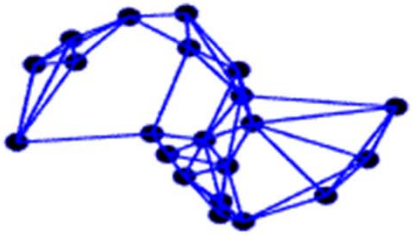


(a)

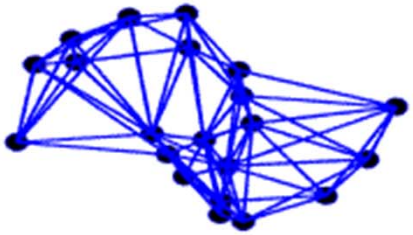


(b)

Distance threshold

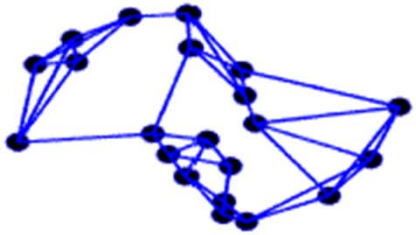


(c)

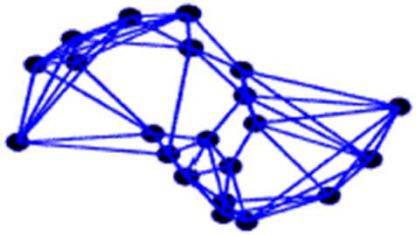


(d)

Rank threshold (kNN)



(e)



(f)

B-Match

k,b=4

k,b=6

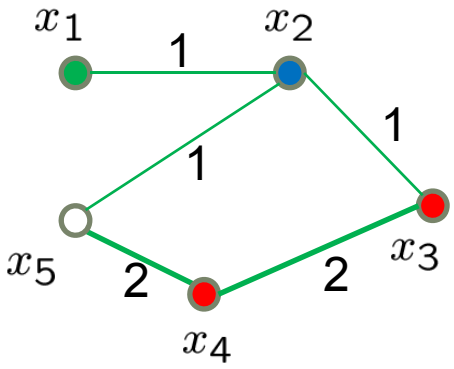
$$\max_P \sum_{ij} P_{ij} W_{ij}$$

$$s.t. \sum\nolimits_j P_{ij} = b, P_{ii} = 0, P_{ij} = P_{ji}, \forall i, j \in 1, \ldots, n$$

(Huang and Jebara, AISTATS 2007)

(Jebara, Wang, and Chang, ICML 2009)

Graph Laplacian



Affinity matrix

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 1 & 0 & 2 & 0 \end{bmatrix}$$

Degree Matrix

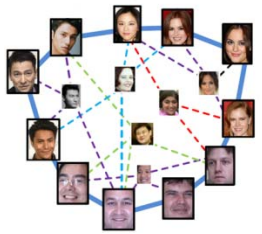
$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

- Graph Laplacian $\Delta = \mathbf{D} - \mathbf{W}$, and $\mathbf{L} = \mathbf{D}^{-1/2}\Delta\mathbf{D}^{-1/2}$
- smoothness of function *f* over graph

$$< f, \mathbf{L}f > = f^T \mathbf{L} f$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} \left\| \frac{f(x_i)}{\sqrt{D_{ii}}} - \frac{f(x_j)}{\sqrt{D_{jj}}} \right\|^2$$

Graph Hashing

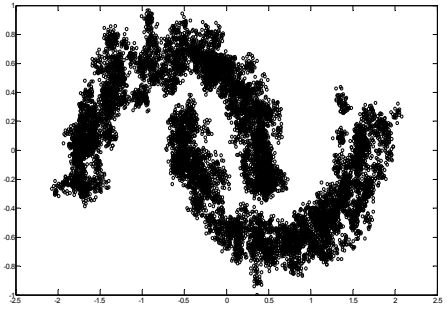


Weiss '08,
Liu '11, '14

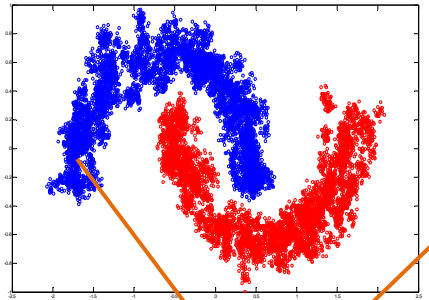
$$< h, \mathbf{L}h > = h^T \mathbf{L} h = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \left\| h(x_i) - h(x_j) \right\|^2$$

- Eigenvectors of Laplacian ***L*** give smooth functions over points
- Can be used to discover structures approximately

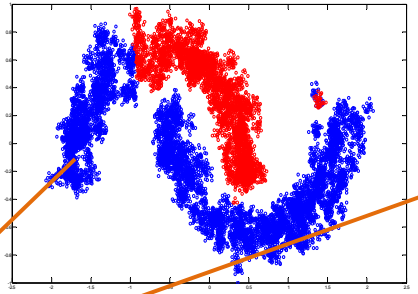
Example:
Graph (12K points)



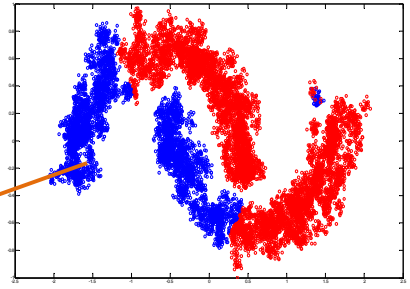
1st Eigenvector
(sign: blue: +1, red: -1)



2rd Eigenvector



3rd Eigenvector



Hash code: [1, 1, 1]

- Locality Sensitivity over the Graph Structure

Challenge: Scale Up to Large Graph

- When graph size is large (million – billion)
  - Hard to construct/store graph (kN^2)
  - Hard to compute eigenvectors (N^3)
- Also lacks a fast function to generate index code for novel data

Idea: Low-rank anchor graph

(Liu, He, Chang, ICML10)

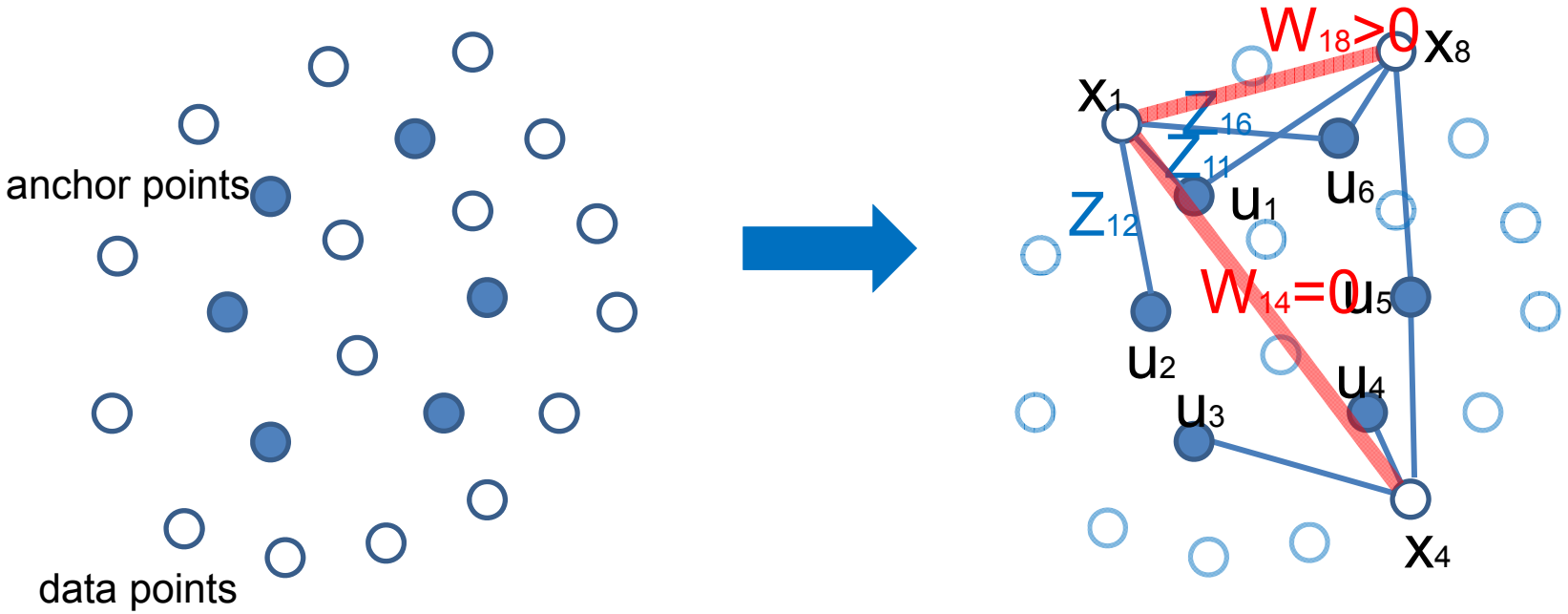
- Use anchor points to "abstract" the graph structure
- Compute data-to-anchor similarity: sparse local embedding

$$Z \in \mathbb{R}^{n \times m}, m \ll n$$

- Data-to-data similarity *W* = inner product in the embedded space

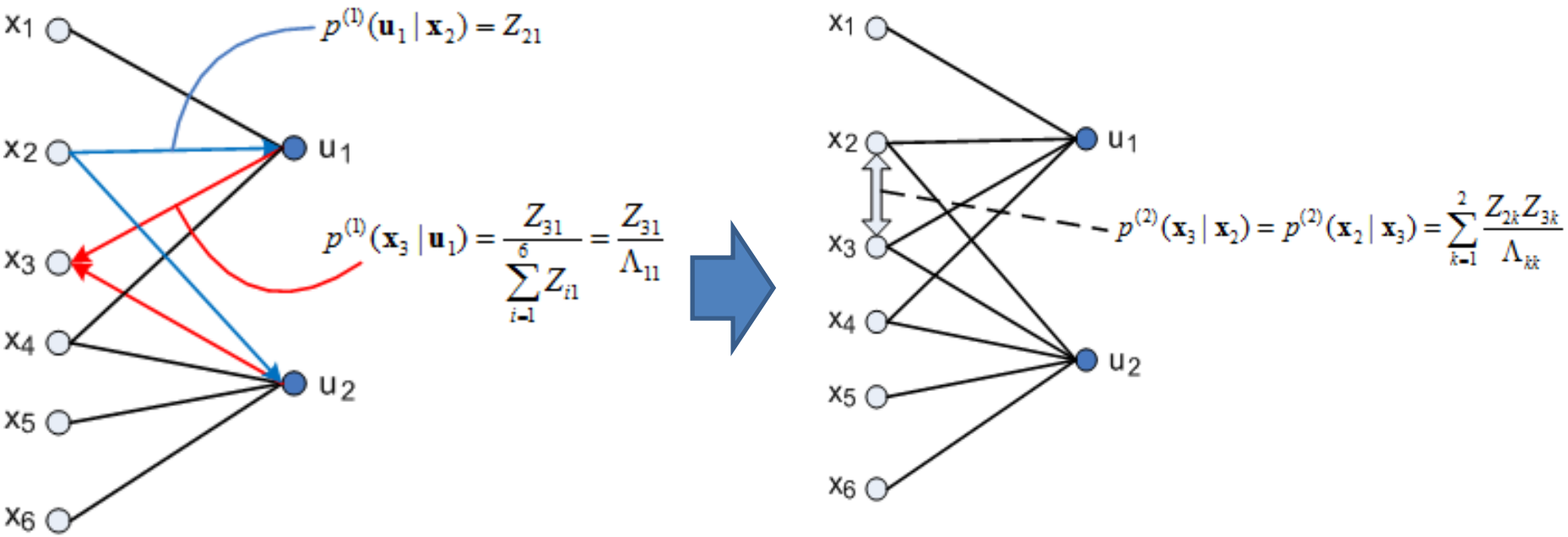
$$W_{ij} = \sum_{k=1}^{m} Z_{ik} Z_{jk} = Z_{i.} Z_{j.}^{T}$$

$$\min_{Z_{ik}} \|x_i - \sum_{k \in \langle i \rangle} Z_{ik} u_k\|^2 \quad s.t. \quad \sum_{k \in \langle i \rangle} Z_{ik} = 1, Z_{ik} \geq 0$$



Probabilistic Intuition

- Affinity between samples i and j, W_{ij}
  = probability of two-step Markov random walk



$$W = Z\Lambda^{-1}Z^{\top}, \text{where } \Lambda = \text{diag}(\mathbf{1}^{\top}Z).$$

AnchorGraph: sparse, positive semi-definite

Anchor Graph

$$W = Z\Lambda^{-1}Z^{\top}, \text{where } \Lambda = \text{diag}(\mathbf{1}^{\top}Z).$$

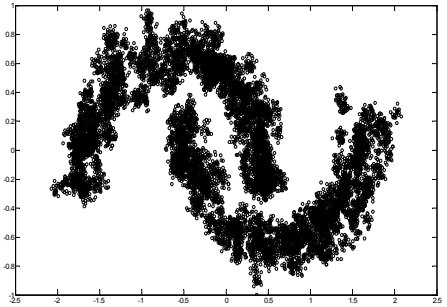
- Affinity matrix W is sparse, positive semi-definite, and low rank
- Eigenvectors of graph Lapalcian can be solved efficiently over low-rank matrix $O(m^2 n)$

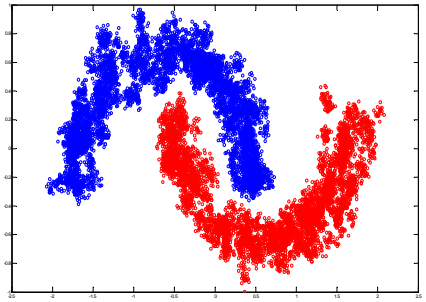
$$E = \{e_1, \cdots, e_K\} \in R^{m \times K}$$

- Fast hash function for new data: sgn(Z(x)E)

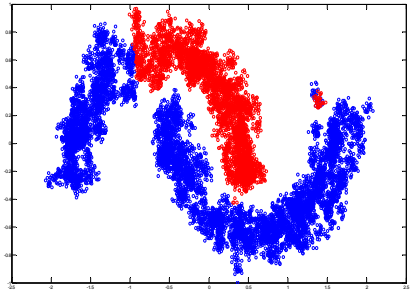
Example of Anchor Graph Hashing

Original Graph (12K points)

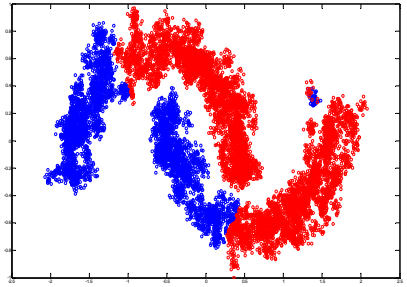




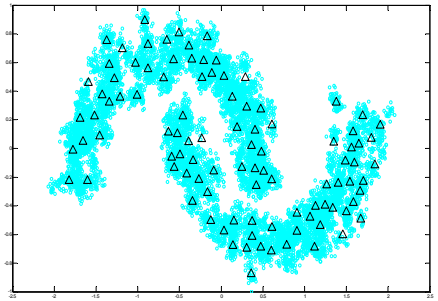
1st Eigenvector
(blue: +1, red: -1)

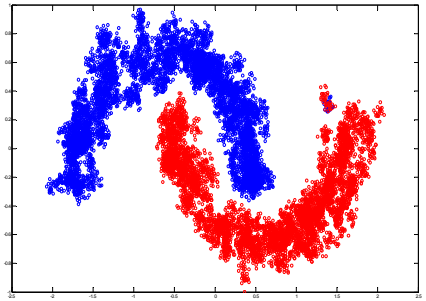


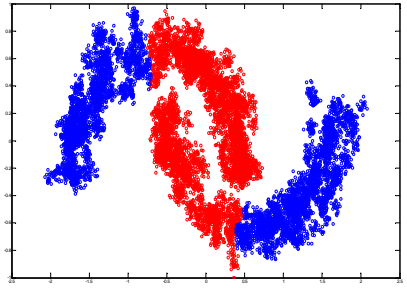
2rd Eigenvector

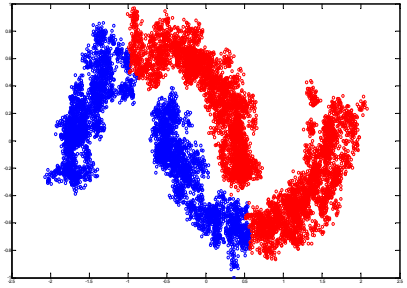


3rd Eigenvector







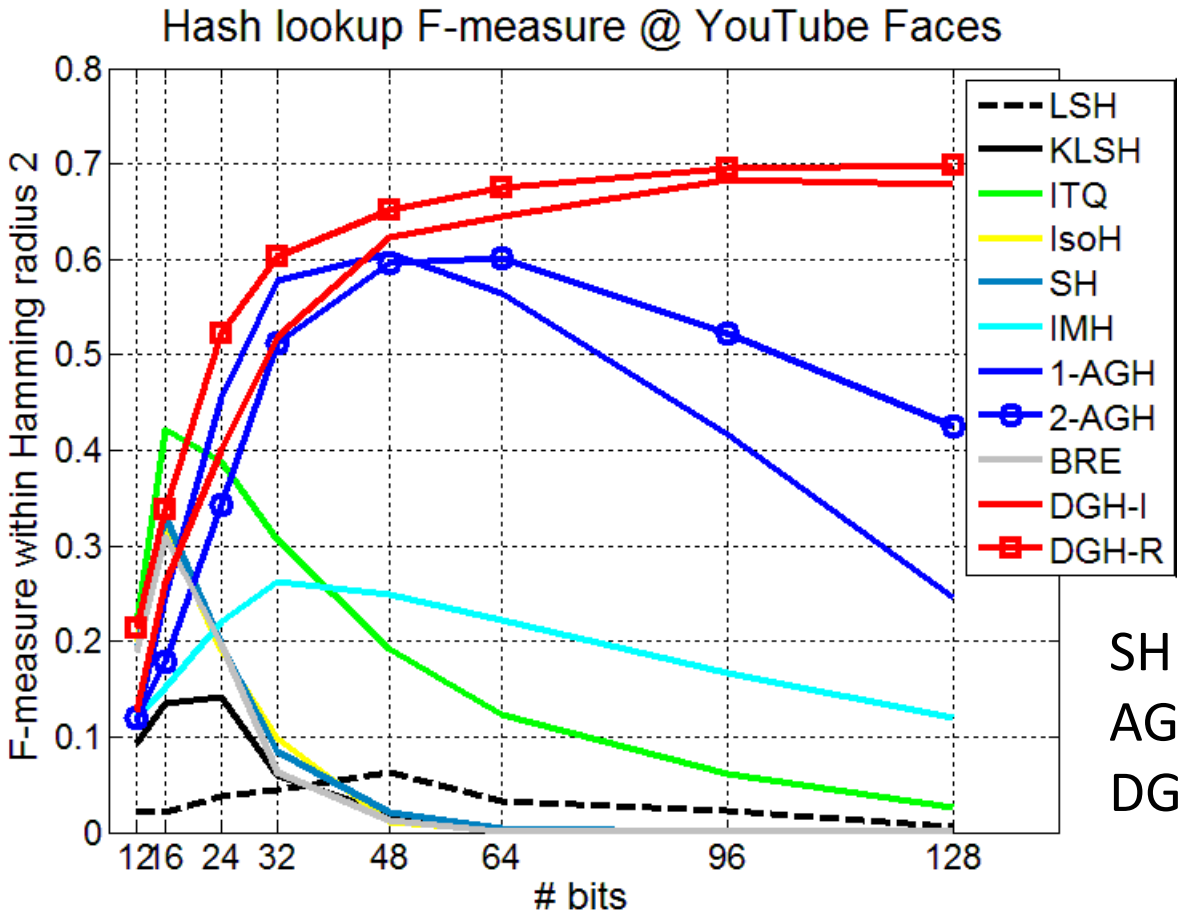


Anchor Graph (m=100 anchors)

- Approximate well the exact eigenvectors of the original graph

YouTube Face Dataset

Wolf *et al.* CVPR'11. 370K face images from 340 people, 3.8K images from 38 people as queries.



Correct: faces from the same person.

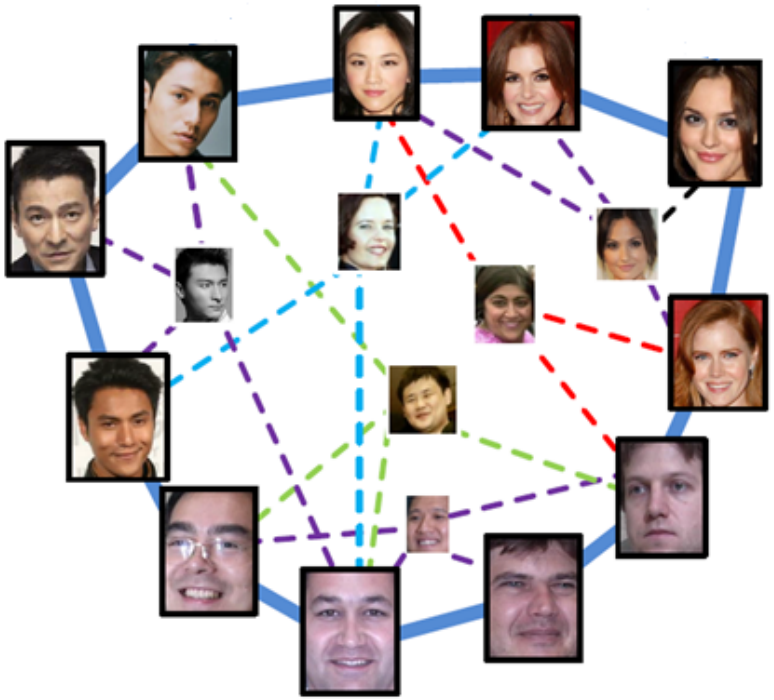
$$F\text{-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

SH = Spectral Hashing
AGH = Anchor Graph Hashing
DGH = Discrete Graph Hashing

Recent work: Direct Graph Hashing

(Liu, Mu, Kumar, Chang, NIPS14)

Face Image Graph



- Instead of finding eigenvectors and quantize, directly find binary hash codes
- direct discrete optimization

$$\begin{aligned} \min_H \quad & \sum_{i,j=1}^{n} \|H_i - H_j\|^2 W_{ij} \\ \text{s.t.} \quad & H \in \{1,-1\}^{n\times r} \\ & \mathbf{1}^\top H = 0 \quad \text{(balanced)} \\ & H^\top H = nI_{r\times r} \quad \text{(orthogonal)} \end{aligned}$$

W : graph adjacency matrix

Direct Graph Hashing Framework

Anchor Graph Laplacian $L = I - A$

relax to a mixed formulation, and learn by alternate maximization

$$\begin{aligned} \min_B \ & \operatorname{tr}(B^\top L B) + \tfrac{\rho}{2}\operatorname{dist}^2(B, \Omega) \\ \text{s.t.} \quad & B \in \{1, -1\}^{n \times r} \end{aligned}$$

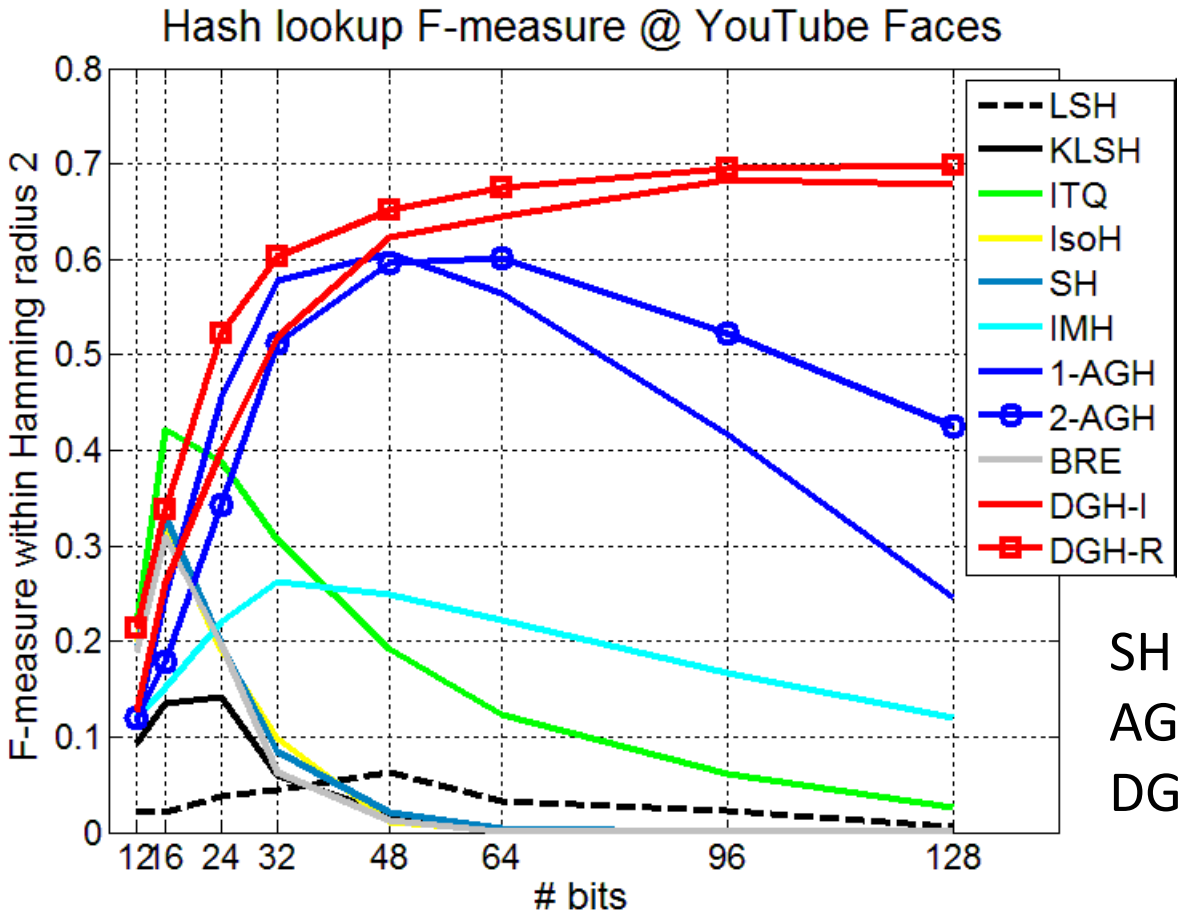
where

$$\Omega = \left\{ Y \in \mathbb{R}^{n \times r} \middle| \mathbf{1}^\top Y = 0, Y^\top Y = n\mathbf{I}_{r \times r} \right\},$$
$$\operatorname{dist}(B, \Omega) = \min_{Y \in \Omega} \|B - Y\|_{\mathrm{F}}.$$

- Generate *nearly* balanced and uncorrelated (controlled by the parameter ρ) hash codes B.

YouTube Face Dataset

Wolf *et al.* CVPR'11. 370K face images from 340 people, 3.8K images from 38 people as queries.



Correct: faces from the same person.

$$F\text{-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

SH = Spectral Hashing
AGH = Anchor Graph Hashing
DGH = Discrete Graph Hashing

Preserve Manifold Structure in Hash codes

Irie, Li, Wu, Chang, CVPR '14

1. Capture locally linear structures

2. Preserve locally linear structures in Hamming space

$$\min_{\mathbf{w}_i} \lambda\|\mathbf{s}_i^\top \mathbf{w}_i\|_1 + \frac{1}{2}\|\mathbf{x}_i - \sum_{j\in\mathcal{N}_E(\mathbf{x}_i)} w_{ij}\mathbf{x}_j\|^2$$

$$\text{s.t.: } \mathbf{w}_i^\top \mathbf{1} = 1,$$

$$\mathbf{s}_i = (s_{i1}, \ldots, s_{in})^\top \quad s_{ij} = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sum_{j\in\mathcal{N}_E(\mathbf{x}_i)} \|\mathbf{x}_i - \mathbf{x}_j\|}$$

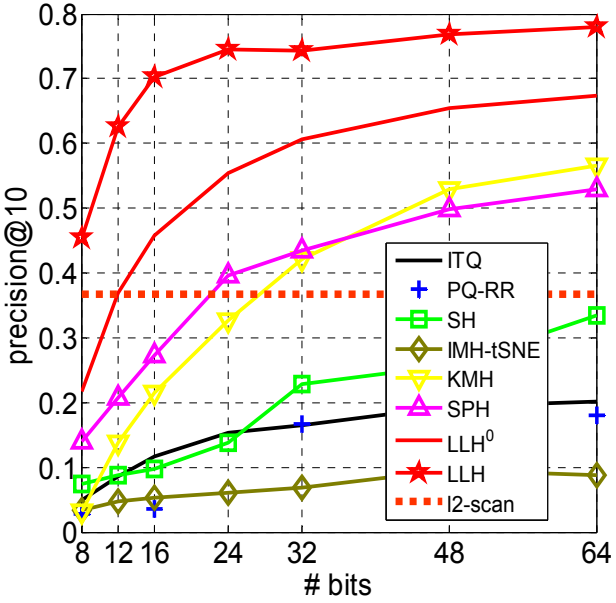
$$\min_{\mathbf{y}_1,\ldots,\mathbf{y}_n\in\mathcal{H}} \sum_i \|\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j\|^2$$

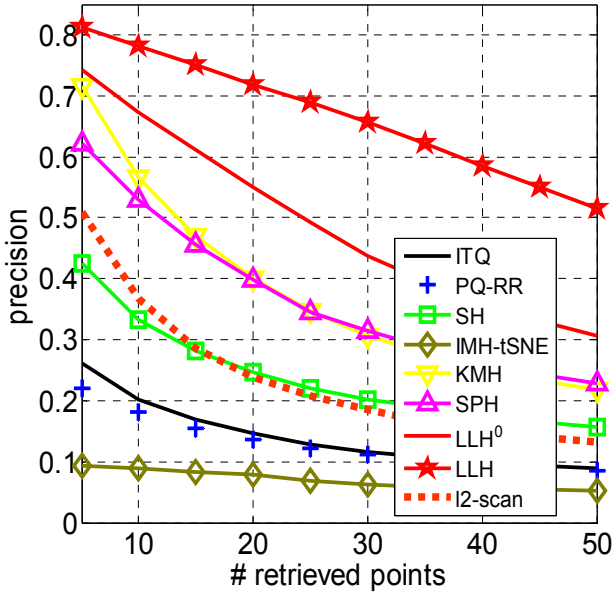
Experiments: Locally Linear hash

(a) Query

(b) Our LLH

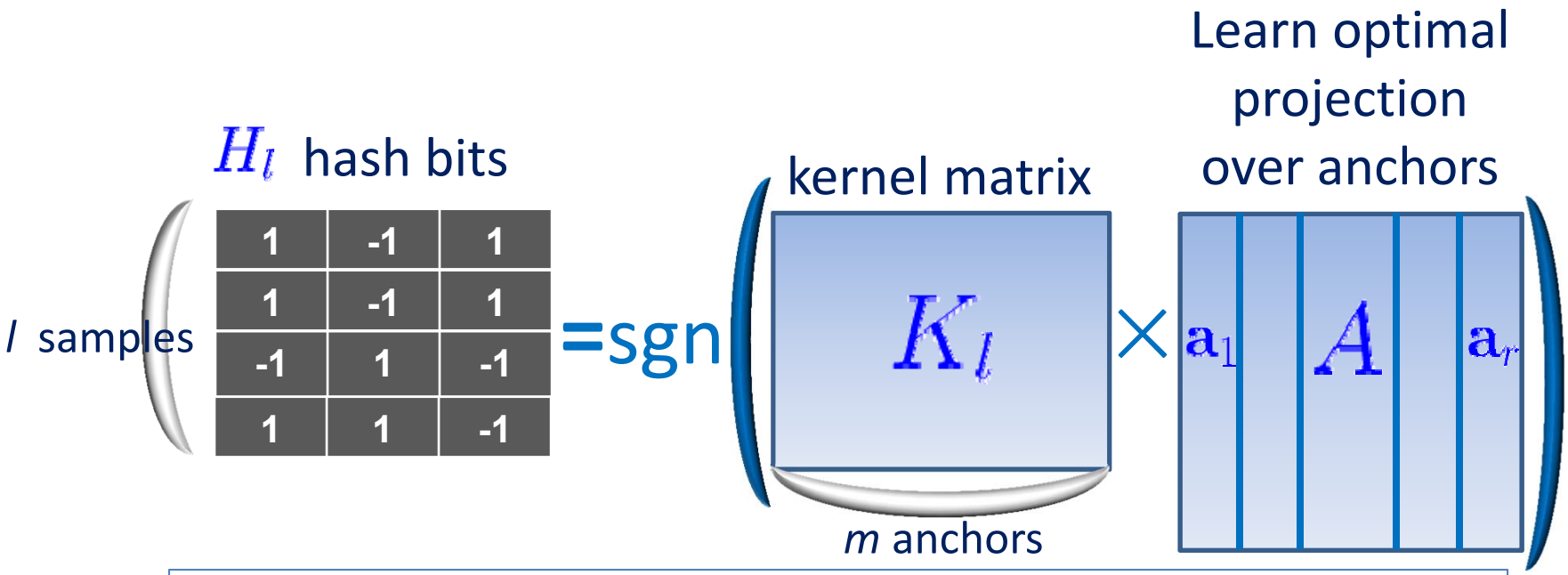
(c) SH [24]





Up to 6-7X performance gain
Over Yale face datset

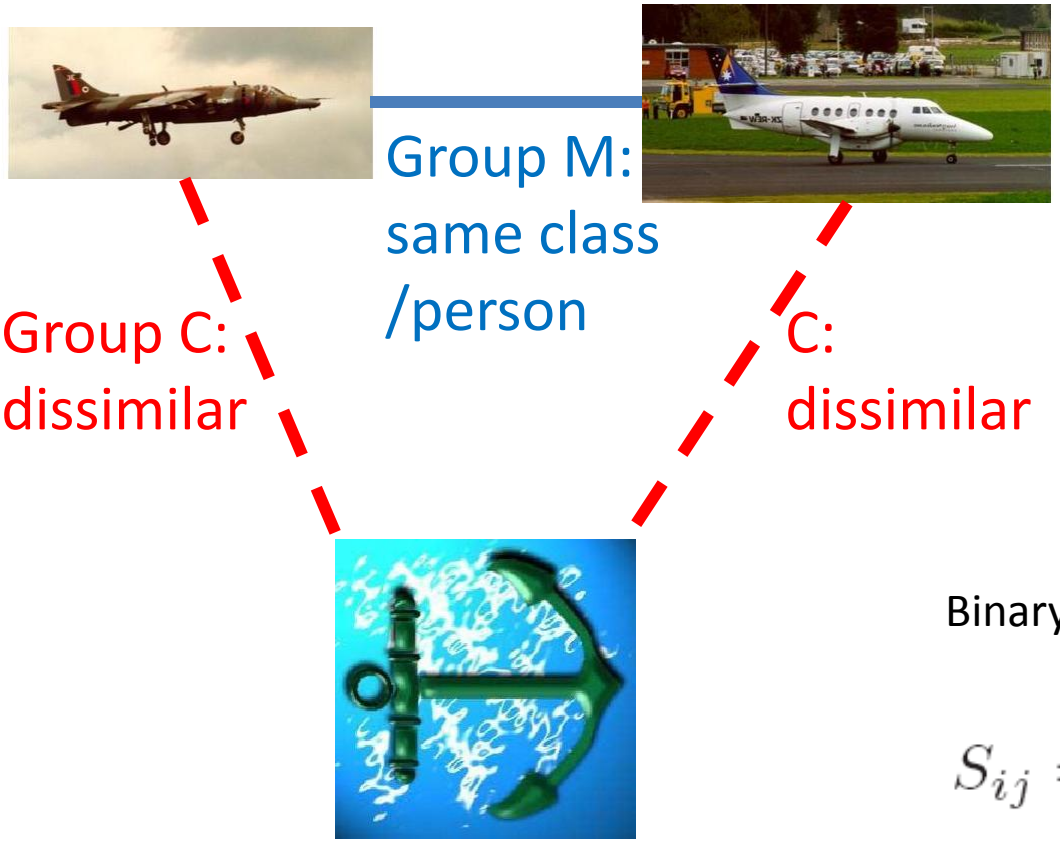
Extend Anchor Idea to Kernel Hashing



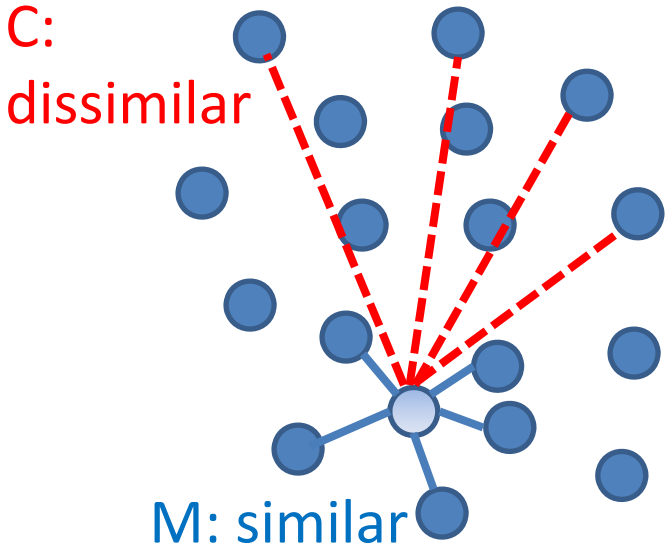
Kernel:
can be any nonlinear kernel, RBF, Gaussian, etc.

Supervised Learning: Explore Additional Side Information

E.g., Binary Supervision



Metric Supervision



Binary Relations:

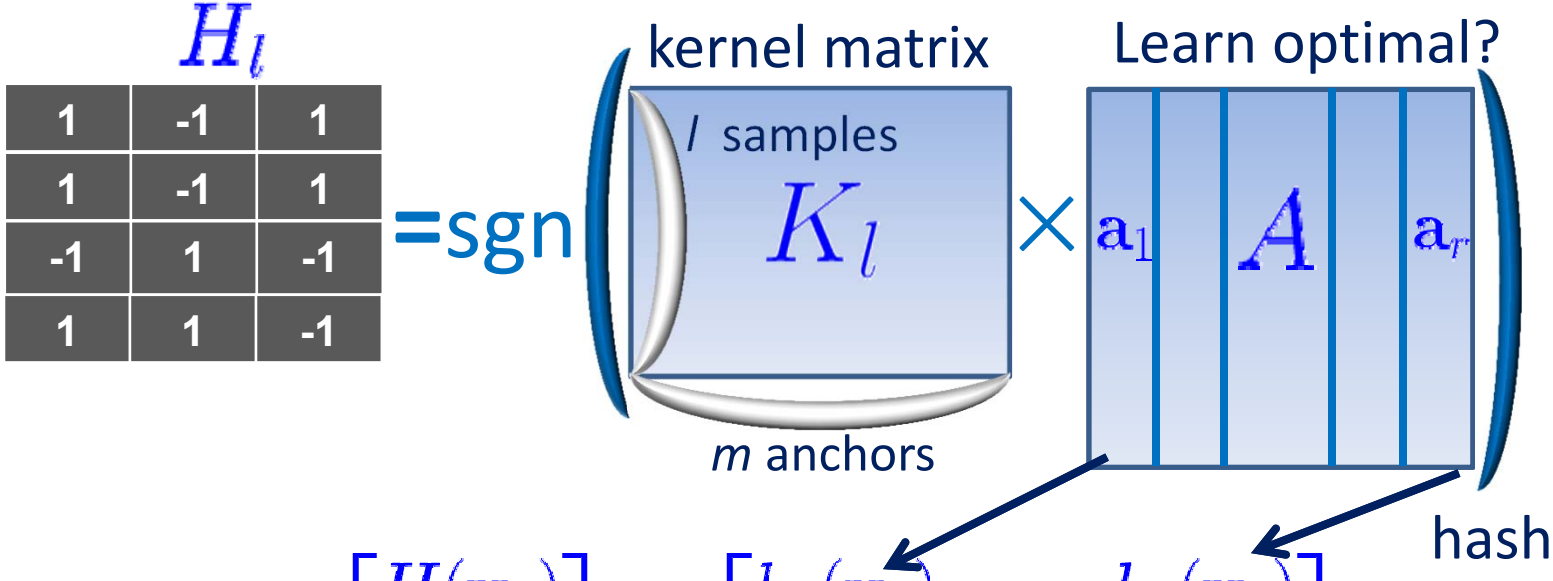
$$S_{ij} = \begin{cases} 1 & : & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ -1 & : & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ 0 & : & otherwise. \end{cases}$$

Liu, et al CVPR'12

Kernel Supervised Hashing

H_l

1	-1	1
1	-1	1
-1	1	-1
1	1	-1



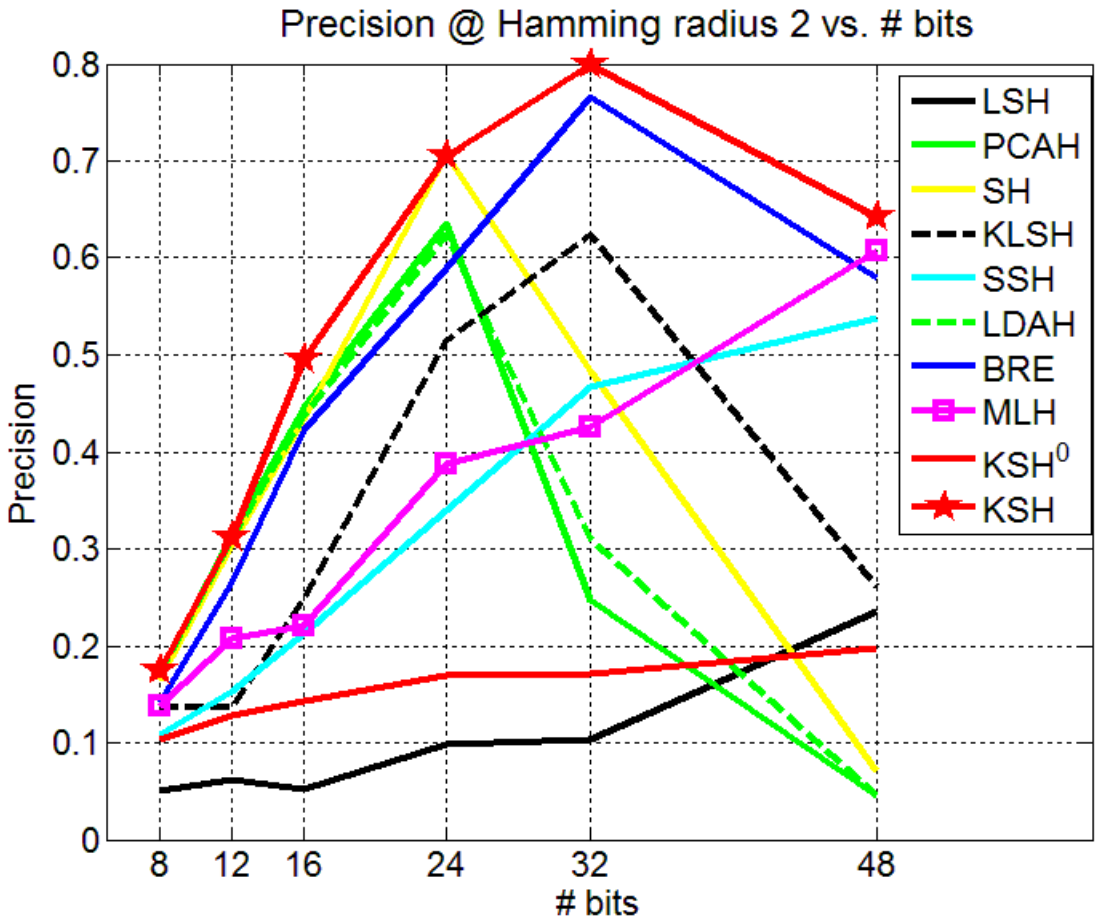
$$H_l = \begin{bmatrix} H(\mathbf{x}_1) \\ \cdots \\ H(\mathbf{x}_l) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1), \cdots, h_r(\mathbf{x}_1) \\ \cdots\cdots \\ h_1(\mathbf{x}_l), \cdots, h_r(\mathbf{x}_l) \end{bmatrix}$$

$$\min_{H_l \in \{1,-1\}^{l \times r}} \mathcal{Q} = \left\| \frac{1}{r} H_l H_l^\top - S \right\|_{\mathrm{F}}^2,$$

S: supervised Information

1M Tiny Image Dataset

2K query images + 1M database images. 5K (0.5%) pseudo-labeled positives are used for the supervised label matrix *S*.



KLSH (unsupervised) and KSH (supervised) use the same RBF kernel.

Comparison: Hash vs. Tree indexing

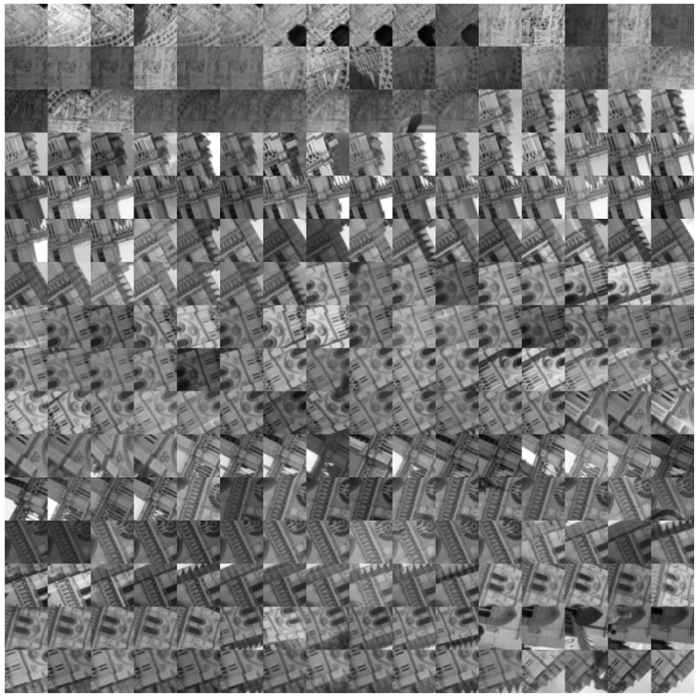
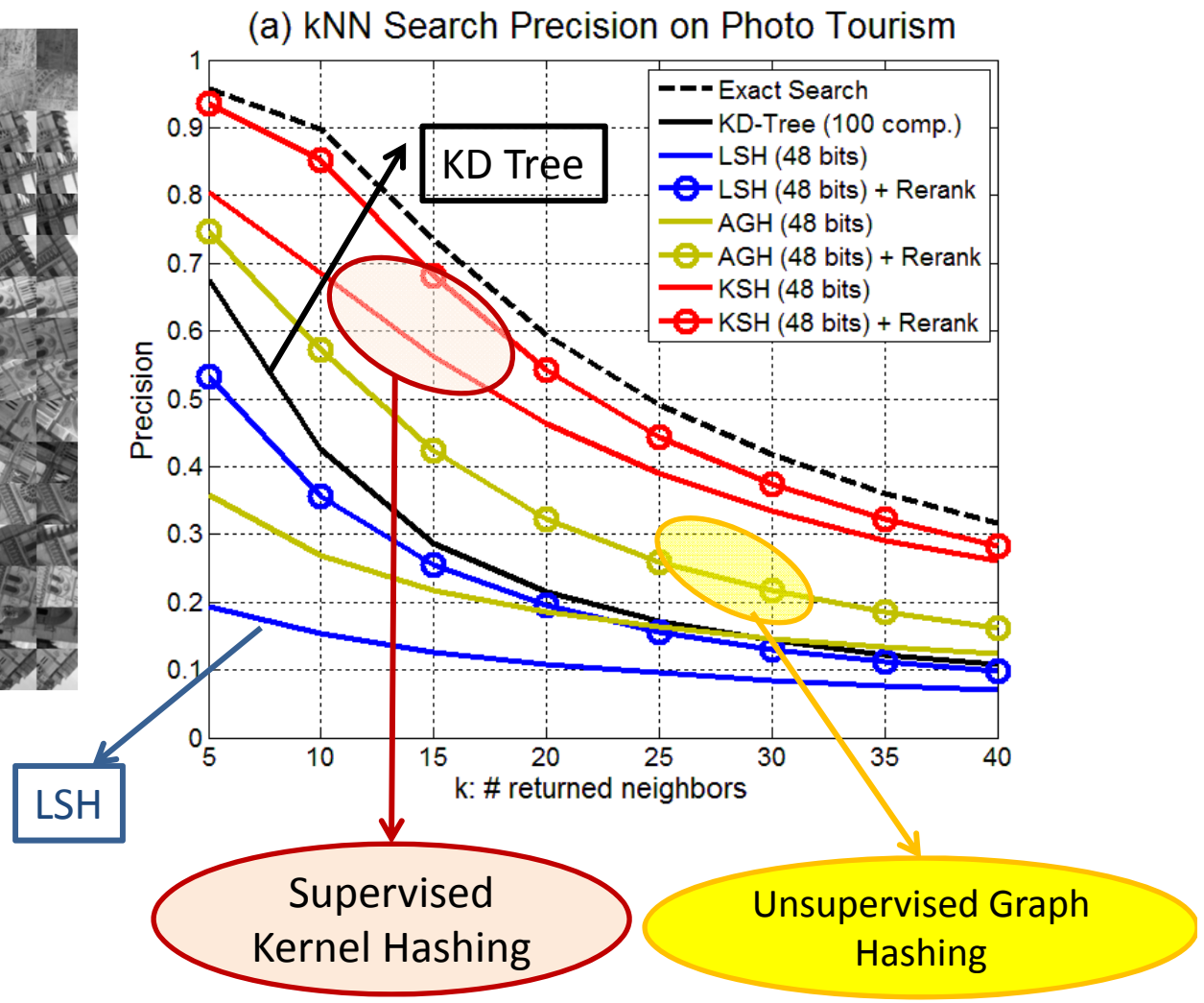
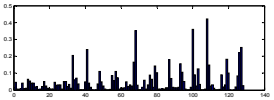


Photo Tourism Patch set
(103,000 samples)
512-dim SIFT feature



Back to Visual Search Applications

1. Take a picture

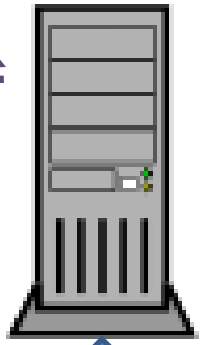


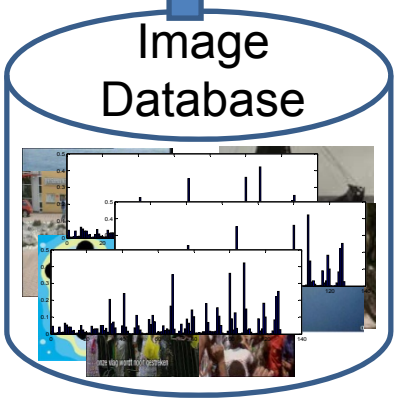
2. Send image or features

3. Send via mobile networks

5. Send results back

4. Visual matching with database images



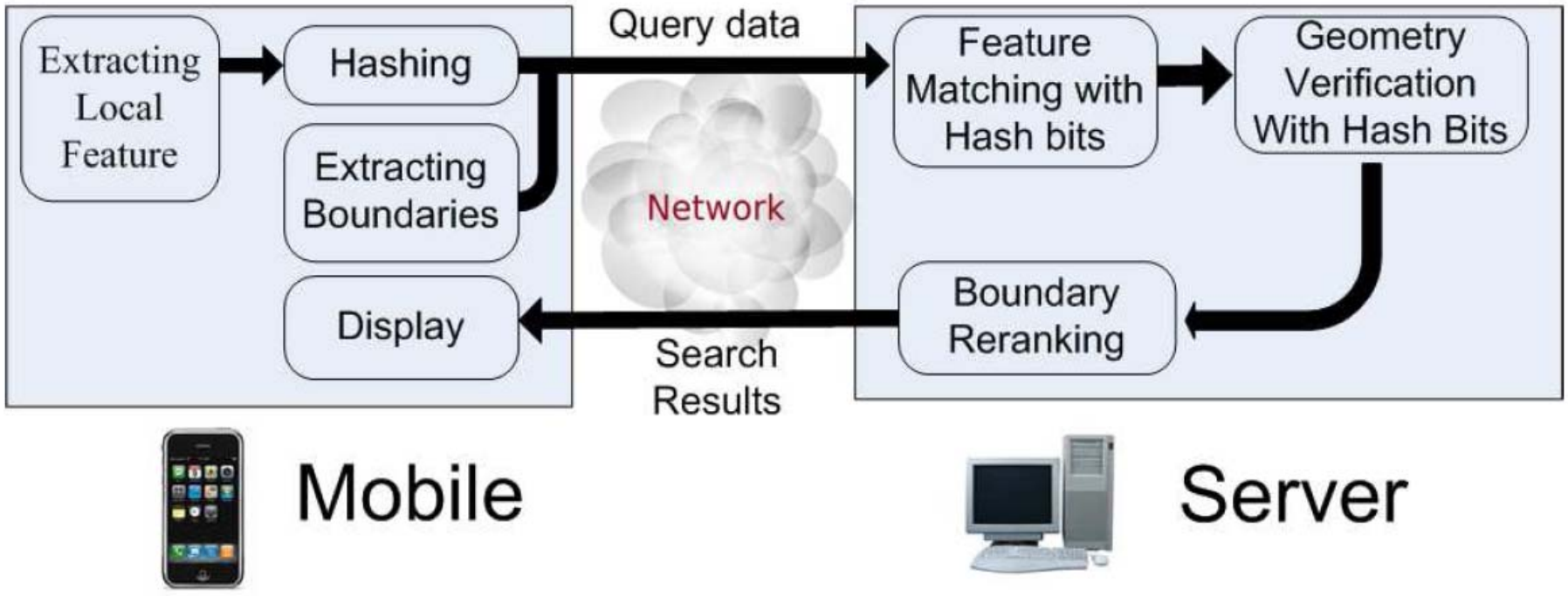


Mobile Visual Search using Bags of Hash Bits (BoHB)

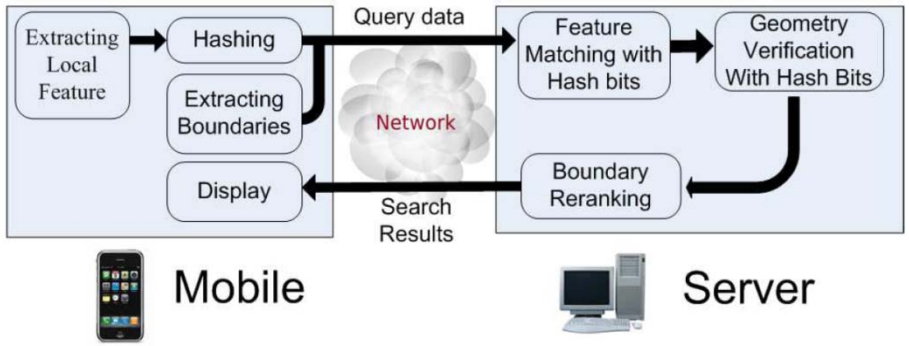
Fast Hashing

Compact Code

Search Large DB



Columbia MVS System:
Bags of Hash Bits and Boundary features



Server:

- **400,000 product images crawled from Amazon, eBay and Zappos**
- **Hundreds of categories; shoes, clothes, electrical devices, groceries, kitchen supplies, movies, etc.**

Speed

- **Feature extraction: ~1s**
- **Transmission: 80 bits/feature, 1KB/im**
- **Serer Search: ~0.4s**
- **Download/display: 1-2s**

video demo (50")

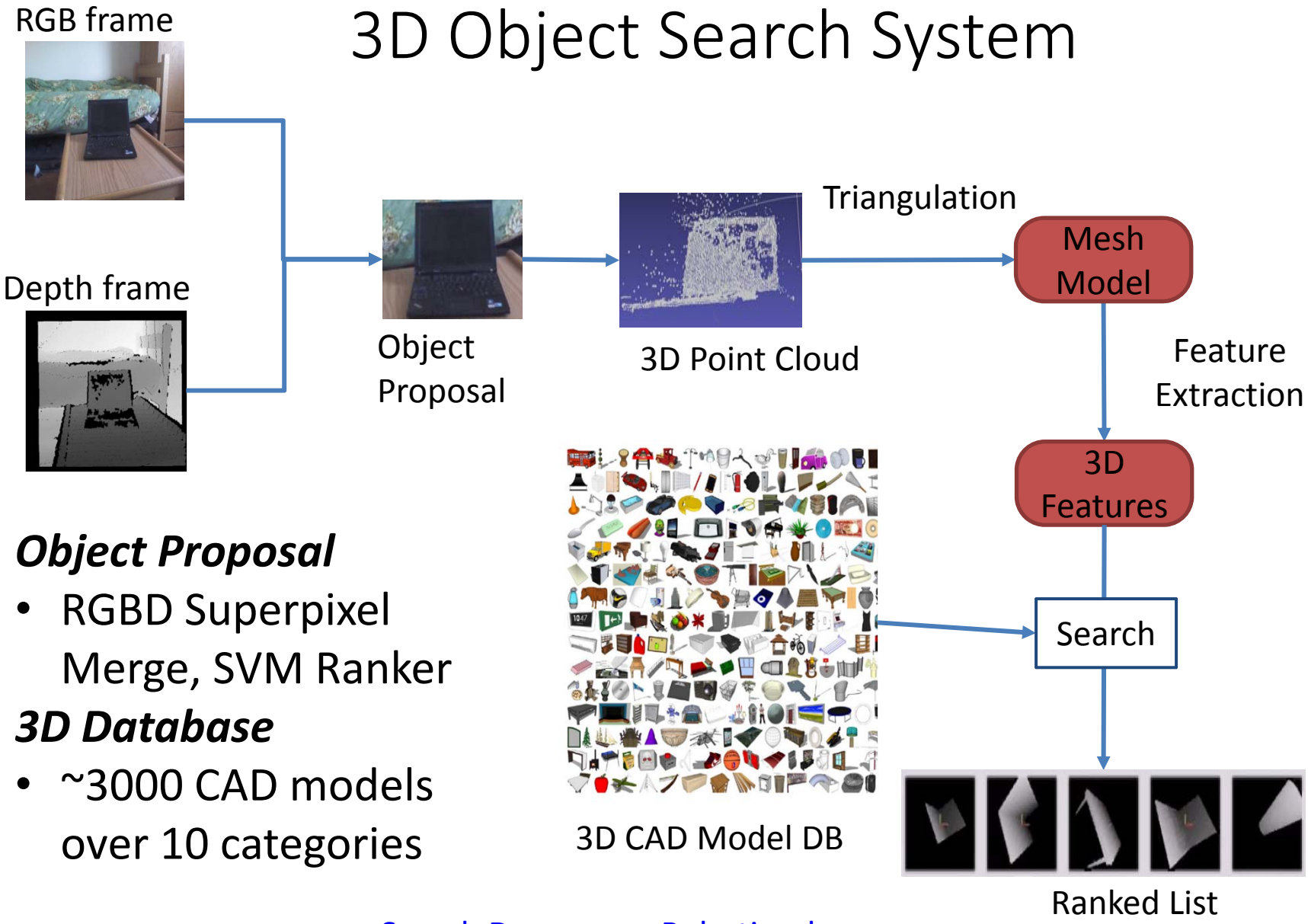
Extension to 3D Model Search

3D query

3D CAD Model DB

Application: 3D printing, Robotics, etc.

3D Object Search System



Object Proposal

- RGBD Superpixel Merge, SVM Ranker

3D Database

- ~3000 CAD models over 10 categories

Search Demo Robotics demo

Other Hashing Forms

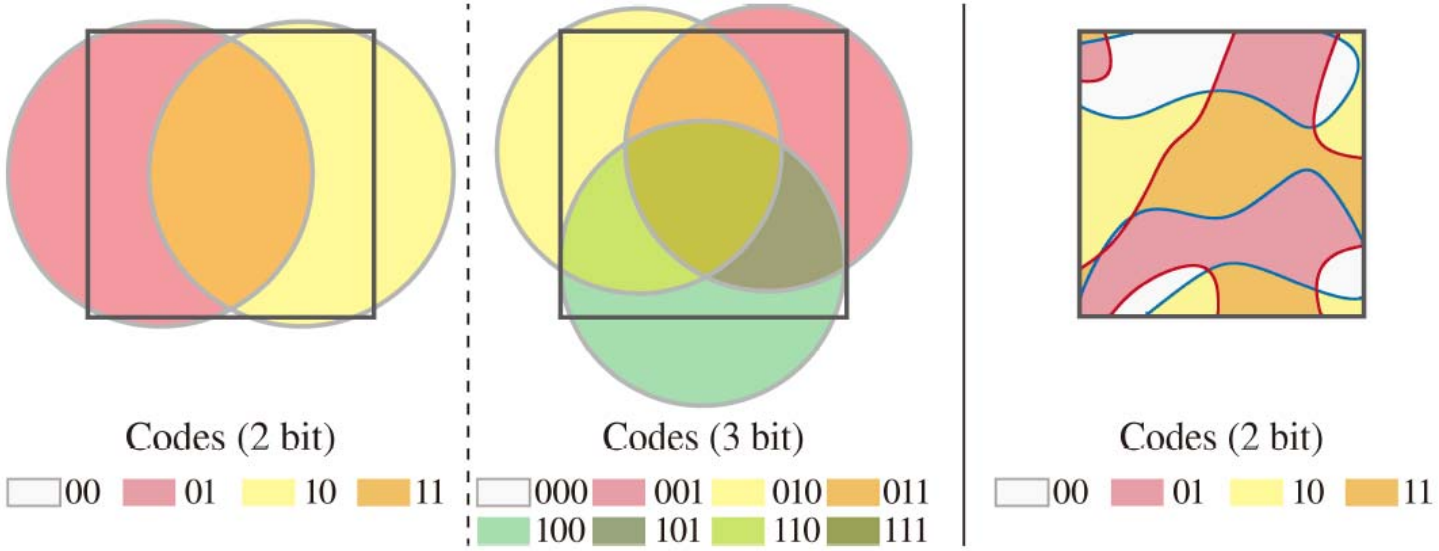
Spherical Hashing

Heo, Lee, He, Chang, Yoon, CVPR 2012

- linear projection -> spherical partitioning

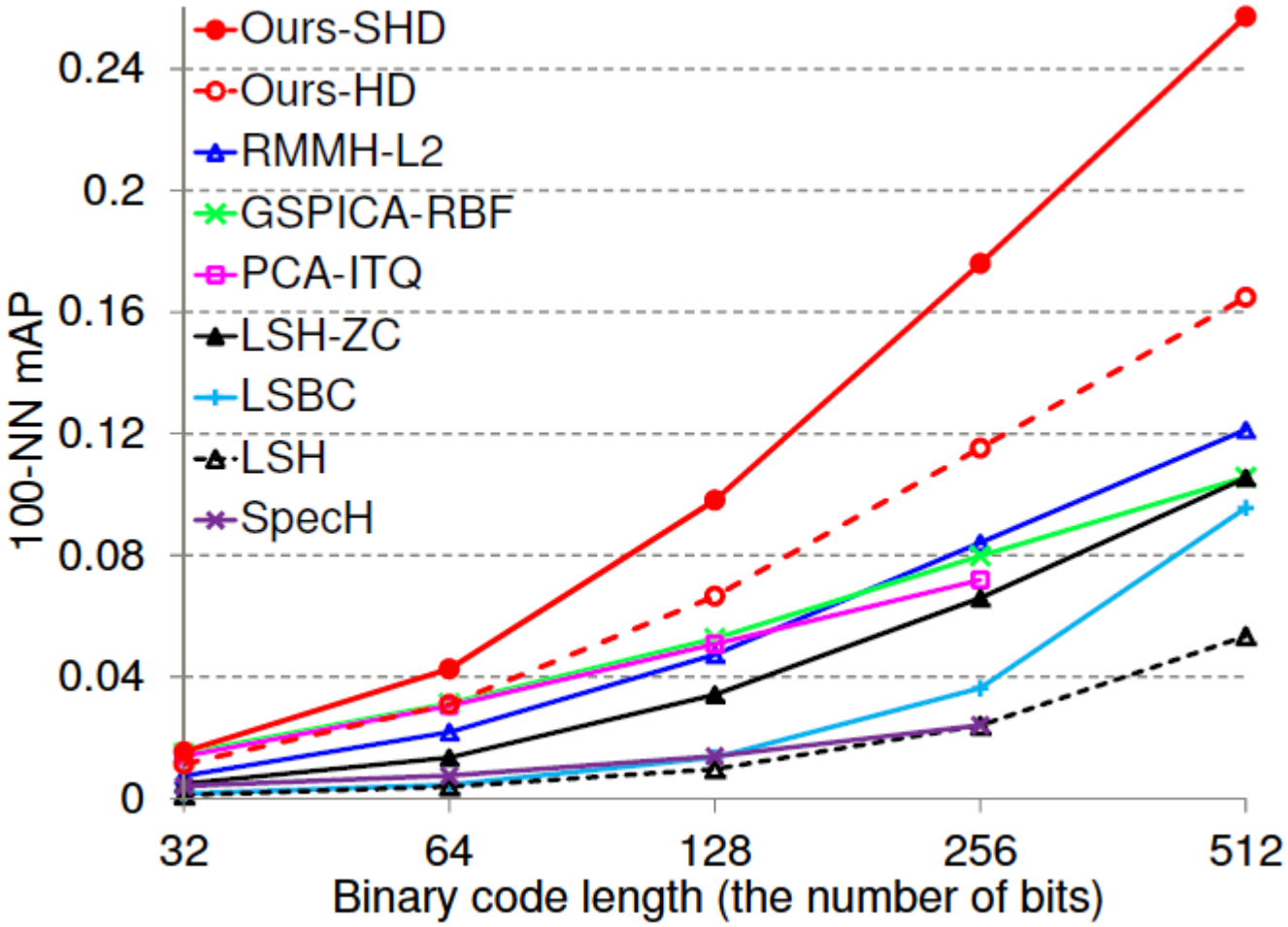
$$h_k(x) = \begin{cases} -1 & \text{when } d(p_k, x) > t_k \\ +1 & \text{when } d(p_k, x) \leq t_k \end{cases}$$

- Asymmetrical hash bits: tighter regions for +1
- Learning: find optimal spheres (center, radius) in the space

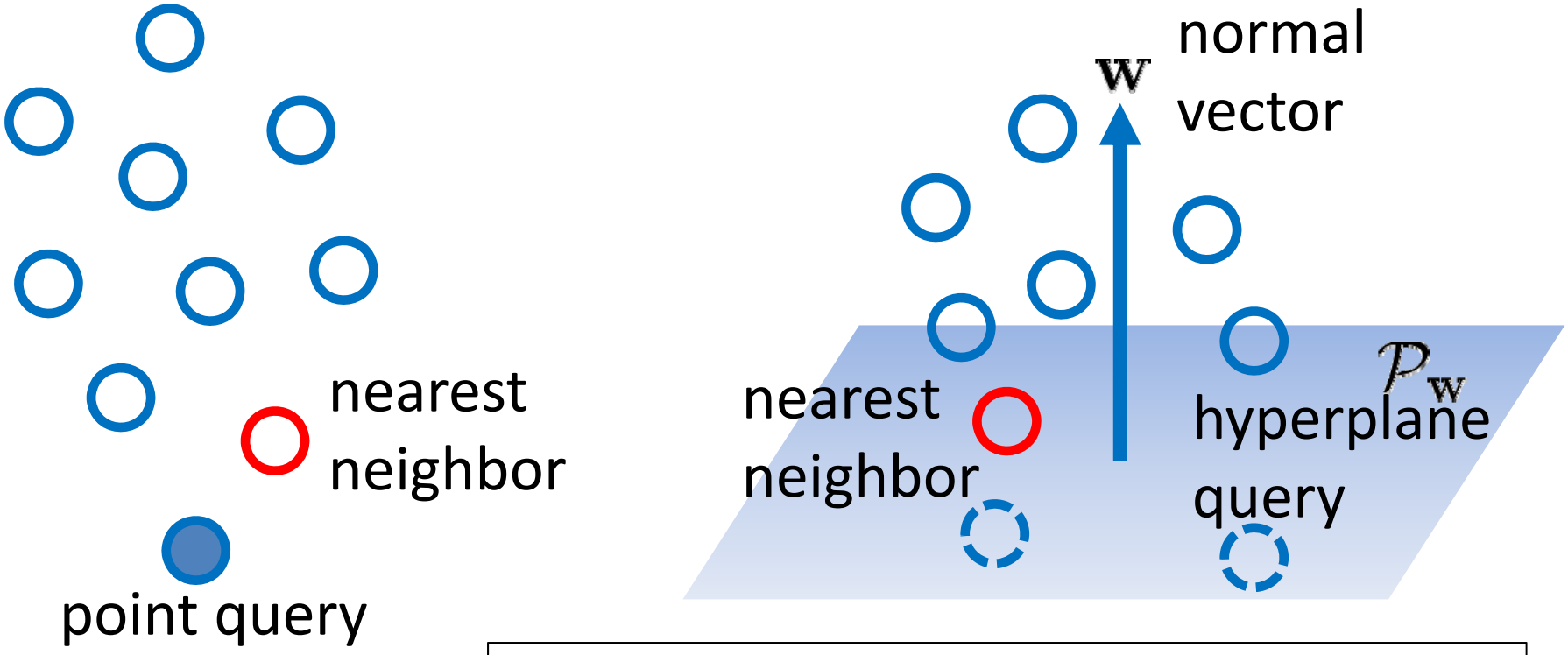


Spherical Hashing Performance

- 1 Million Images: GIST 384-D features

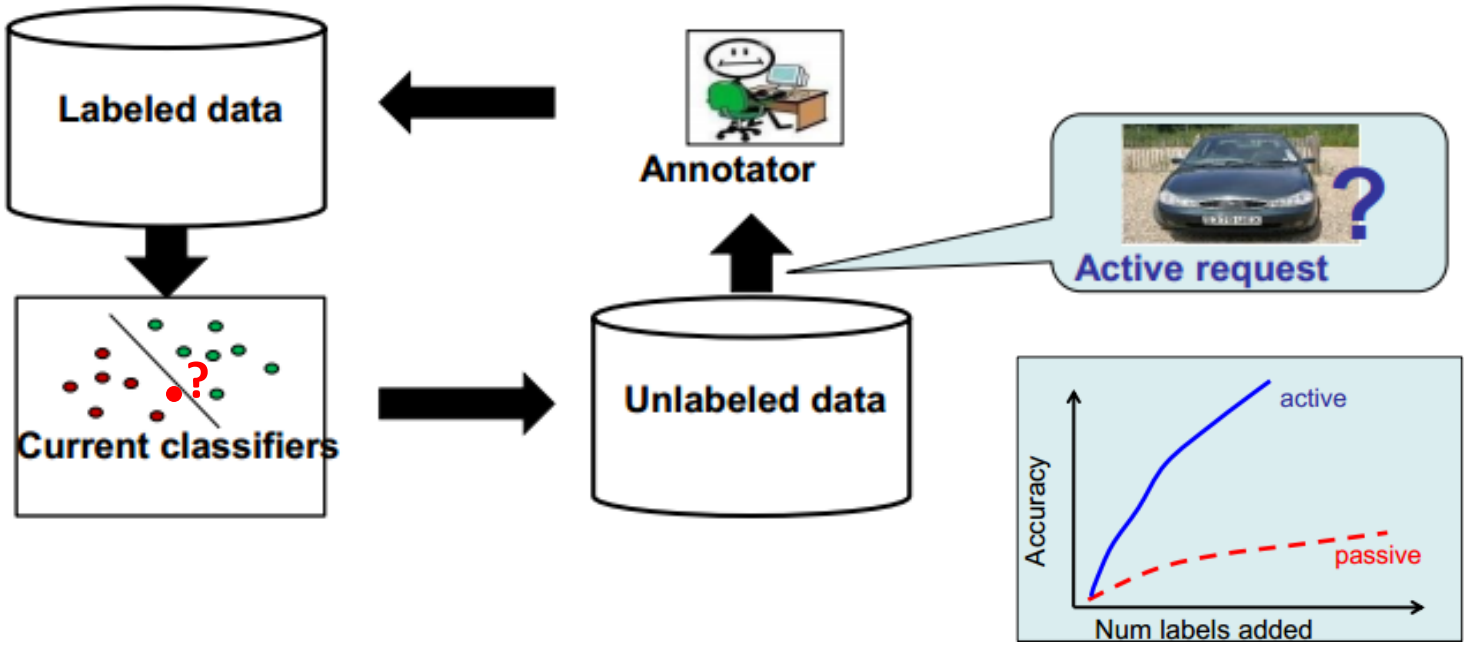


Point-to-Hyperplane Search



Find points closest to the hyperplane

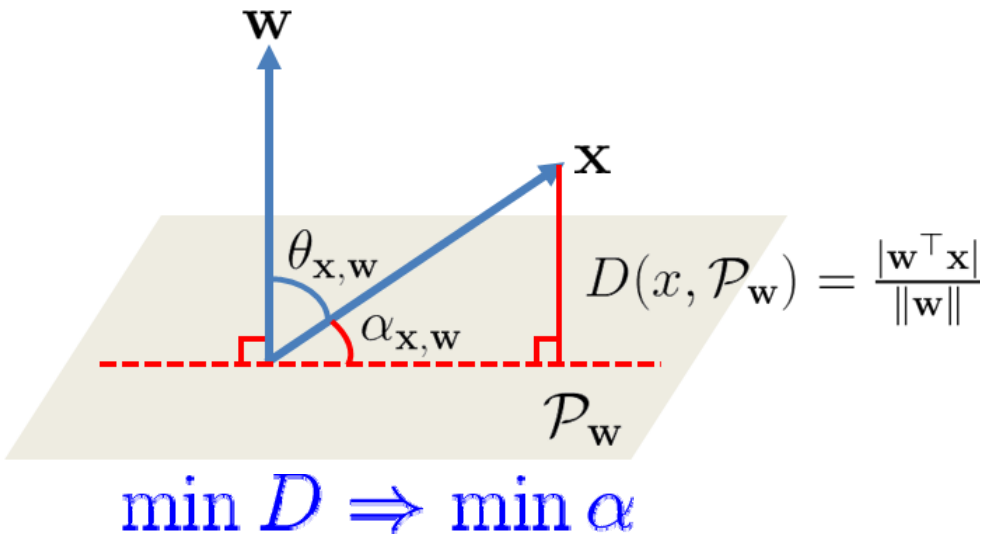
Active learning for image annotation

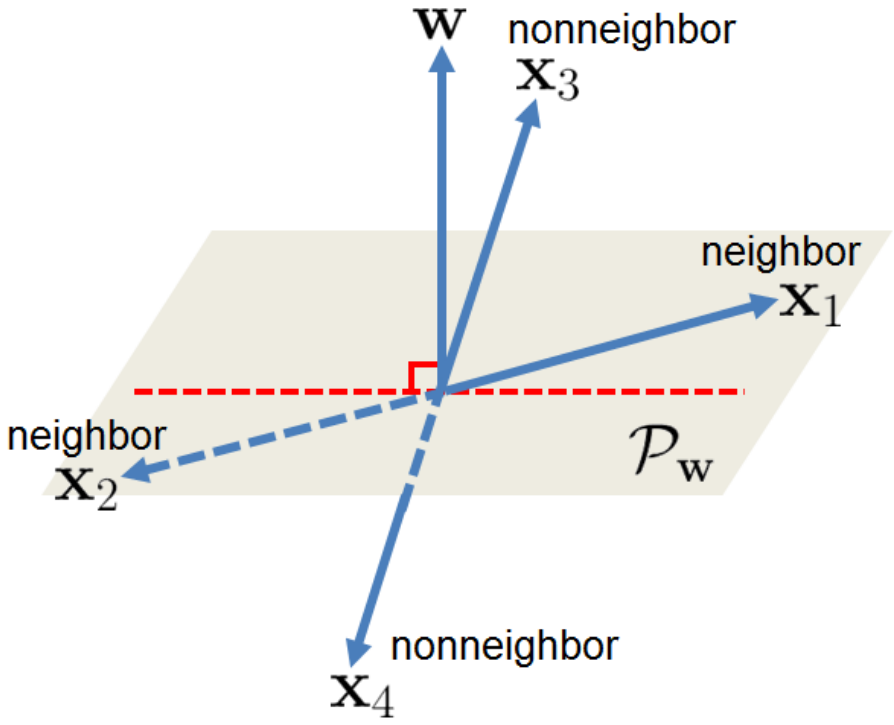


- Choose the sample closet to the SVM classifier hyperplane and ask human annotator

Kristen Grauman

Hashing Principle: Point-to-Hyperplane Angle





The ideal neighbors $\perp$ **w**

Bilinear Hashing

Liu, Jun, Kumar, Chang, ICML12

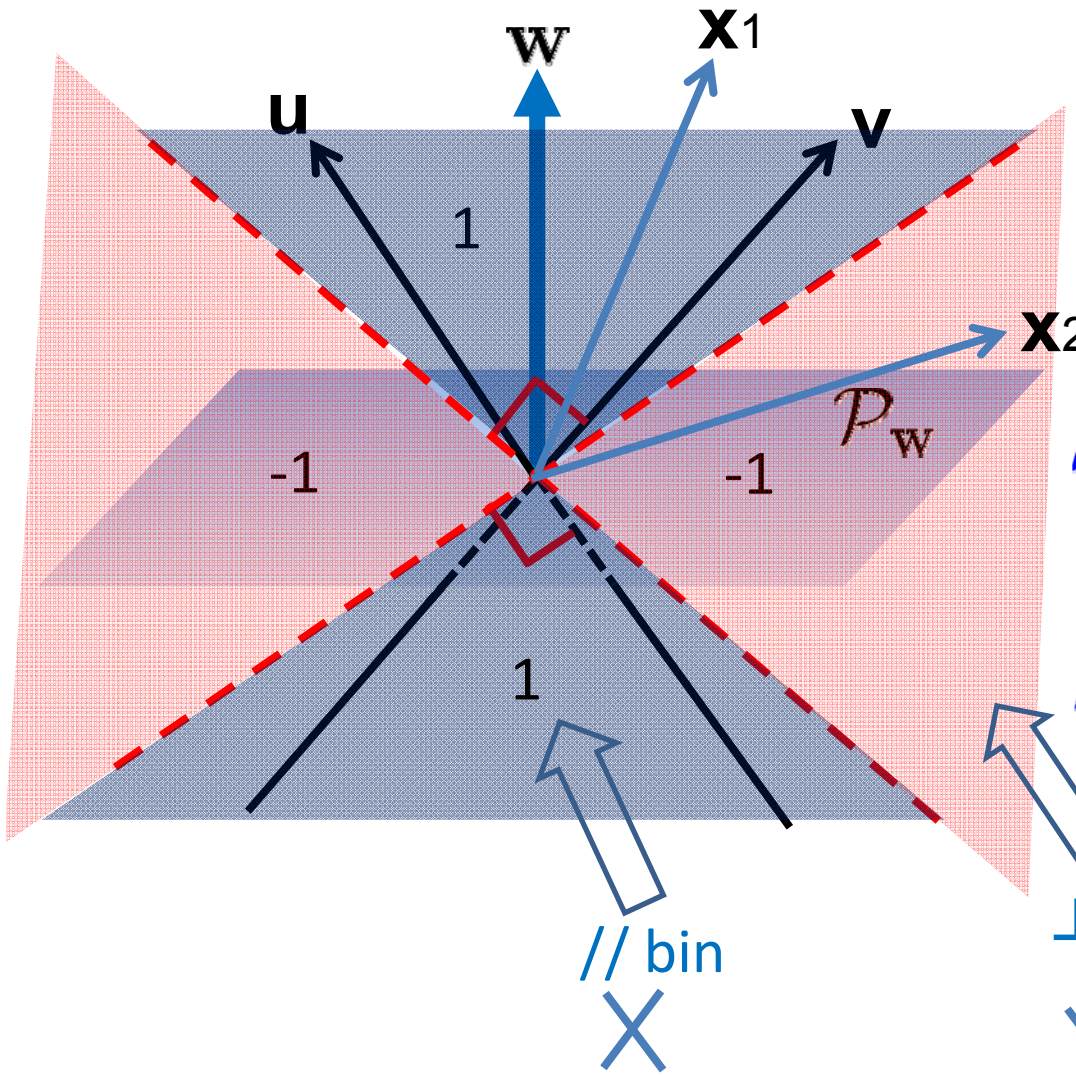
Bilinear-Hyperplane Hash (BH-Hash)

$$h^{\mathcal{B}}(\mathbf{z}) = \mathrm{sgn}(\mathbf{u}^\top \mathbf{z}\mathbf{z}^\top \mathbf{v}), \text{ i.i.d. } \mathbf{u}, \mathbf{v} \sim \mathcal{N}(0, I_{d\times d}).$$

2 random projection vectors

- **bilinear** hash bit: +1 for || points, -1 for $\perp$ points

A Single Bit of Bilinear Hash



$$h^{\mathcal{B}}(\mathbf{w}) = \mathrm{sgn}(\mathbf{u}^\top \mathbf{w}\mathbf{w}^\top \mathbf{v}) = \mathrm{sgn}(\mathbf{u}^\top \mathbf{w})\mathrm{sgn}(\mathbf{v}^\top \mathbf{w}) = 1 \cdot 1 = 1$$

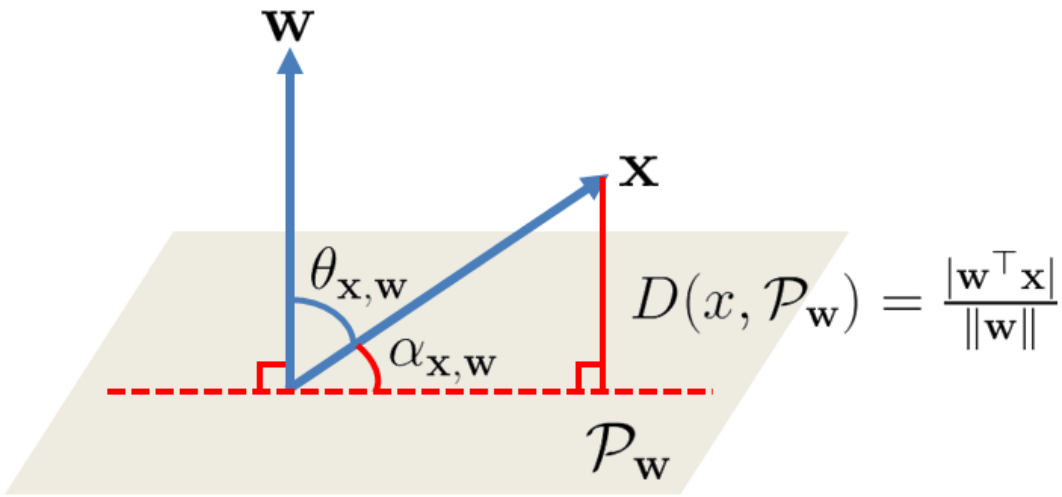
$$h^{\mathcal{B}}(\mathbf{x}_1) = \mathrm{sgn}(\mathbf{u}^\top \mathbf{x}_1)\mathrm{sgn}(\mathbf{v}^\top \mathbf{x}_1) = 1 \cdot 1 = 1$$

$$h^{\mathcal{B}}(\mathbf{x}_2) = \mathrm{sgn}(\mathbf{u}^\top \mathbf{x}_2)\mathrm{sgn}(\mathbf{v}^\top \mathbf{x}_2) = -1 \cdot 1 = -1$$

Locality Sensitivity

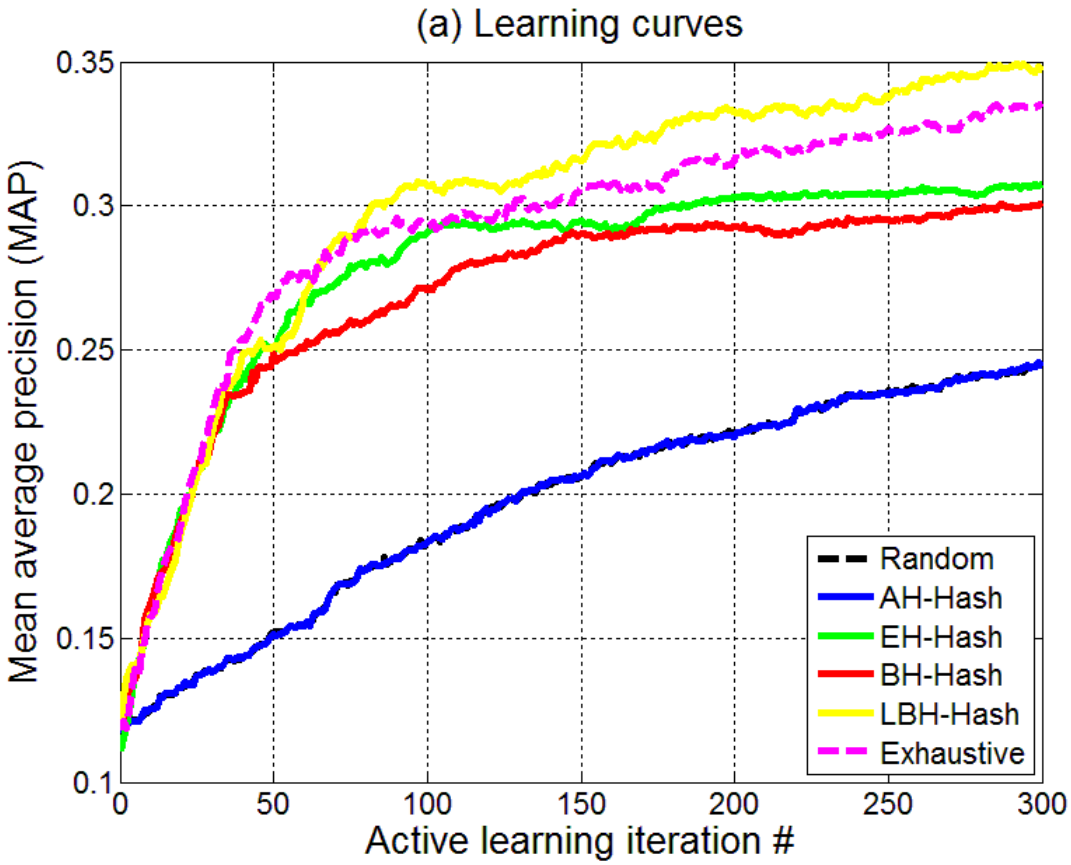
$$\Pr\left[h^{\mathcal{B}}(\mathbf{w}) \neq h^{\mathcal{B}}(\mathbf{x})\right] = \frac{1}{2} - \frac{2(\theta_{\mathbf{x},\mathbf{w}} - \frac{\pi}{2})^2}{\pi^2} = \frac{1}{2} - \frac{2\alpha_{\mathbf{x},\mathbf{w}}^2}{\pi^2}$$

highest collision probability for active hashing



Active SVM Learning with Hyperplane Hashing

- SVM Learning over 1 million data points
  - actively select the most decisive training point in each iteration



Summary and Open Issues

- Locality Sensitive Compact Hashing
  - Applications in large-scale search, active learning, etc.
- Properties
  - Locality Sensitive
  - Fast code generation
  - Compact size: 20-64 bits per point
  - Efficient search: O(1) or sublinear cost
- Novel formulations
  - Graph hash, Kernel hash, Hyperplane hash, spherical
- Open Issues
  - Adaptive learning given new data
  - Incorporate high-order relations (e.g., spatio-temporal)
  - Hashing for heterogeneous multimodal features

Selected References

(Hashing with Graphs)

W. Liu, J. Wang, S. Kumar, S.-F. Chang. Hashing with Graphs, ICML 2011.

W. Liu, C. Mu, S. Kumar, S.-F. Chang. Discrete Graph Hashing. NIPS, 2014.

(Iterative Quantization)

Y. Gong and S. Lazebnik, Iterative Quantization: A Procrustean Approach to Learning Binary Codes, CVPR 2011.

(Manifold Hashing)

G. Irie, Z. Li, X.-M. Wu, S.-F. Chang. Locally Linear Hashing for Extracting Non-linear Manifolds. CVPR, 2014.

(Supervised Kernel Hash)

W. Liu, J. Wang, R. Ji, Y. Jiang, and S.-F. Chang, Supervised Hashing with Kernels, CVPR 2012.

(Hash Based Mobile Product Search)

J. He, T. Lin, J. Feng, X. Liu, S.-F. Chang, Mobile Product Search with Bag of Hash Bits and Boundary Reranking, CVPR 2012

(Semi-Supervised Hash)

J. Wang, S. Kumar, S.-F. Chang. Semi-Supervised Hashing for Scalable Image Retrieval. CVPR 2010.

(Circular Hashing)

F. Yu, S. Kumar, Y. Gong, S.-F. Chang. Circulant Binary Embedding. ICML, 2014.