COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

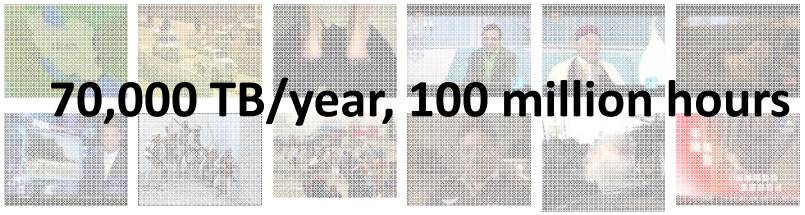
Large Scale Mobile Visual Search

Shih-Fu Chang

June 2012

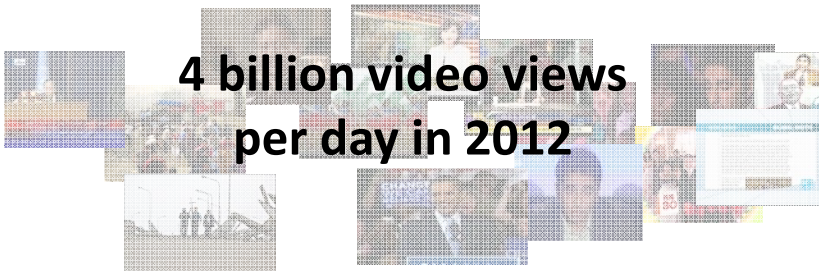
The Explosive Growth of Visual Data

broadcast



Social portals

video blogs



- Many domains broadcast, entertainment, social media
- 1 month YouTube > 60 years video of 3 major TV networks

(Slide from Lexing Xie)

Many research & commercial search engines

informedia
digital video understanding

Autonomy
VIRAGE

MediaMill
Semantic Video Search Engine

Sixth Sense

QBIC™

VideoGoogle

CuZero

bing

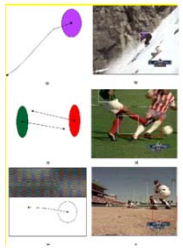
VisualSEEk

IBM
Multimedia Analysis and Retrieval System

snaptell

TALKMINER

VideoQ



blinkx

2008

Google goggles
labs

Google™
Video BETA

IQ Engines

VisionGo

TinEye

kooaba
IMAGE RECOGNITION

SINGING FISH
the audio/video search engine

like.com
visual search

leafsnap

VIDEOSURF

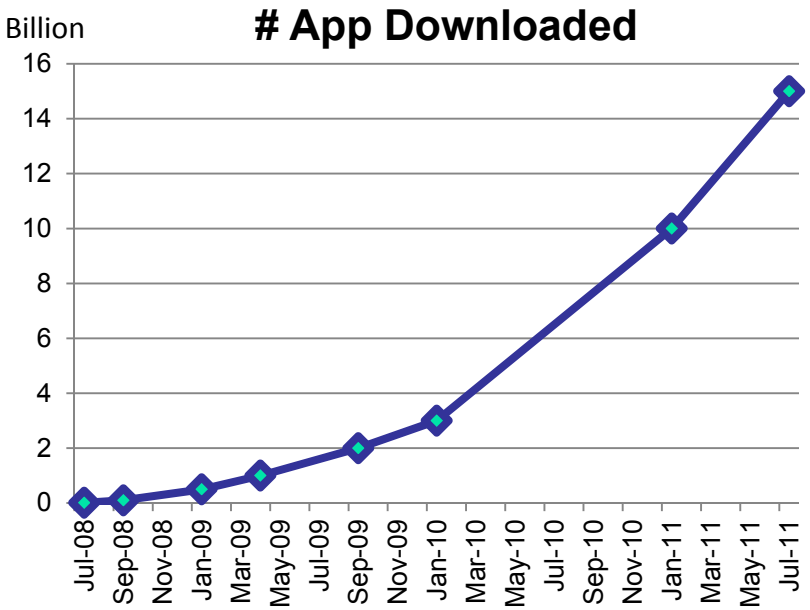
Stanford Mobile Visual Search

ALIPR™
Automatic Photo Tagging and Visual Image Search

Mealsnap

Explosion of Mobile Apps

- July 2008 – 10 million apps downloaded in the first weekend
- Jan 2011 – 10 billion apps downloaded (1000 apps every 3 seconds)
- July 2011 – 15 billion apps downloaded



Jan. 2009, askiphone.net

Mobile meets Visual Search

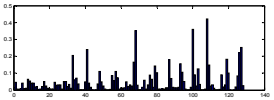
- Expanded visual sense

- Expanded audio sense

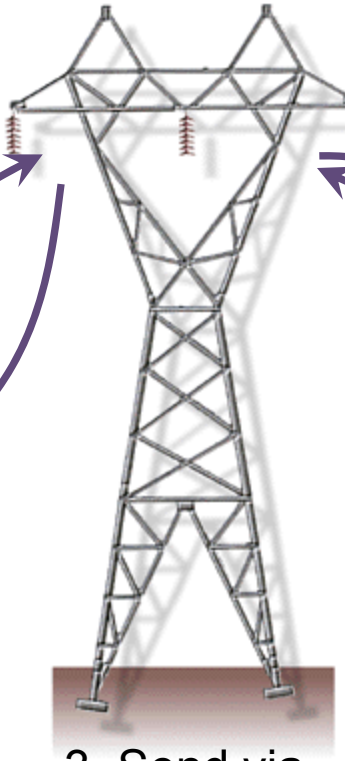
- Expanded food sense

How does Mobile Visual Search work?

1. Take a picture



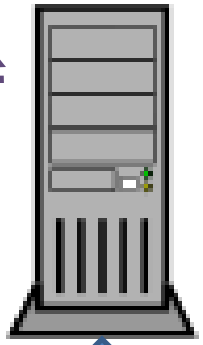
2. Send image or features

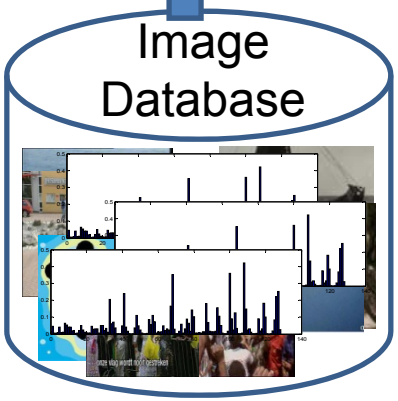


3. Send via mobile networks

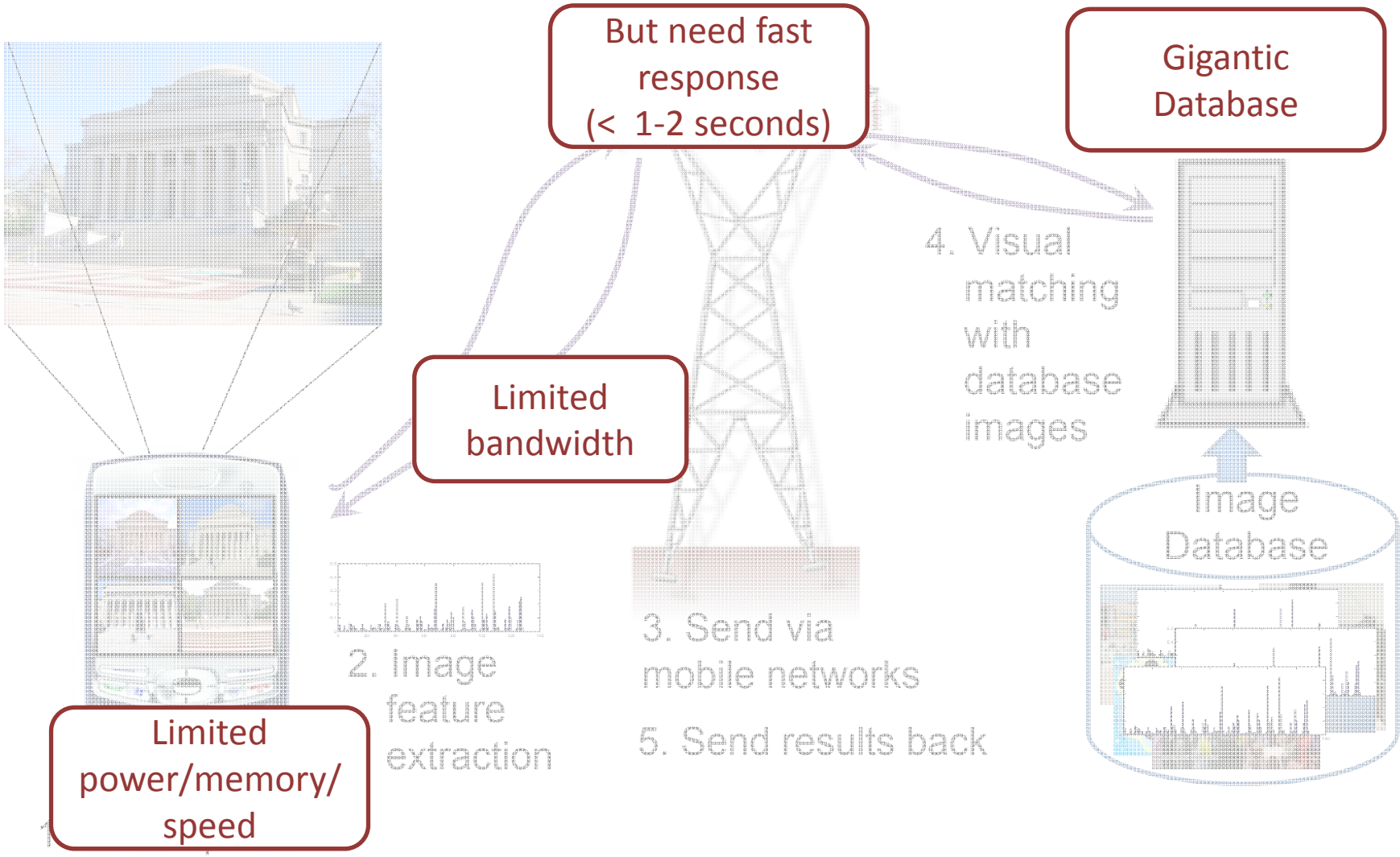
5. Send results back

4. Visual search on server database



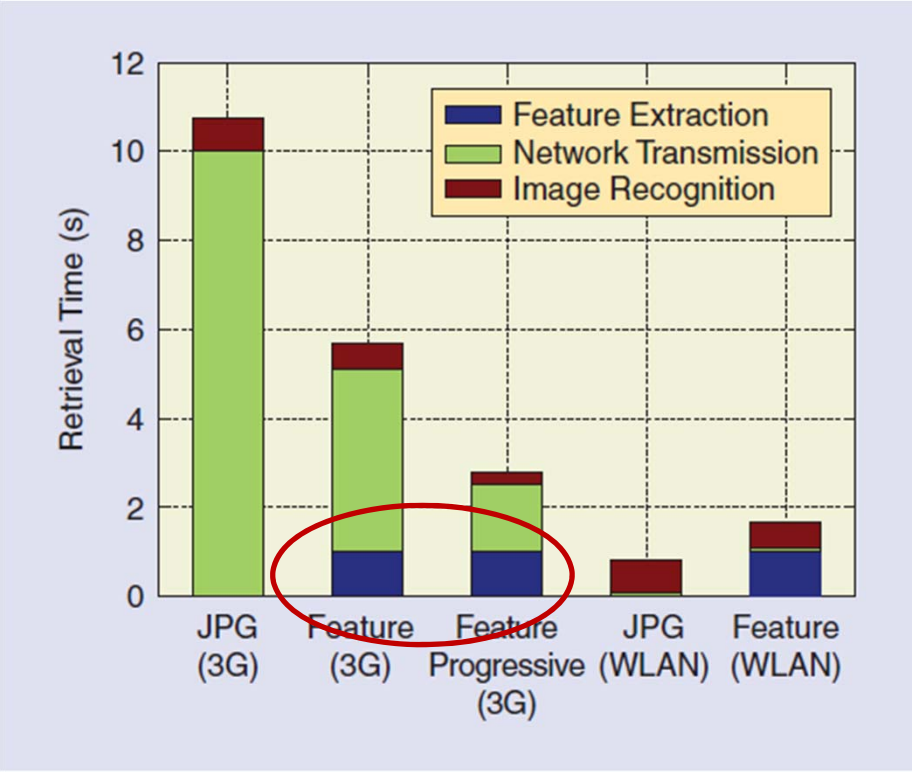


Challenges for MVS



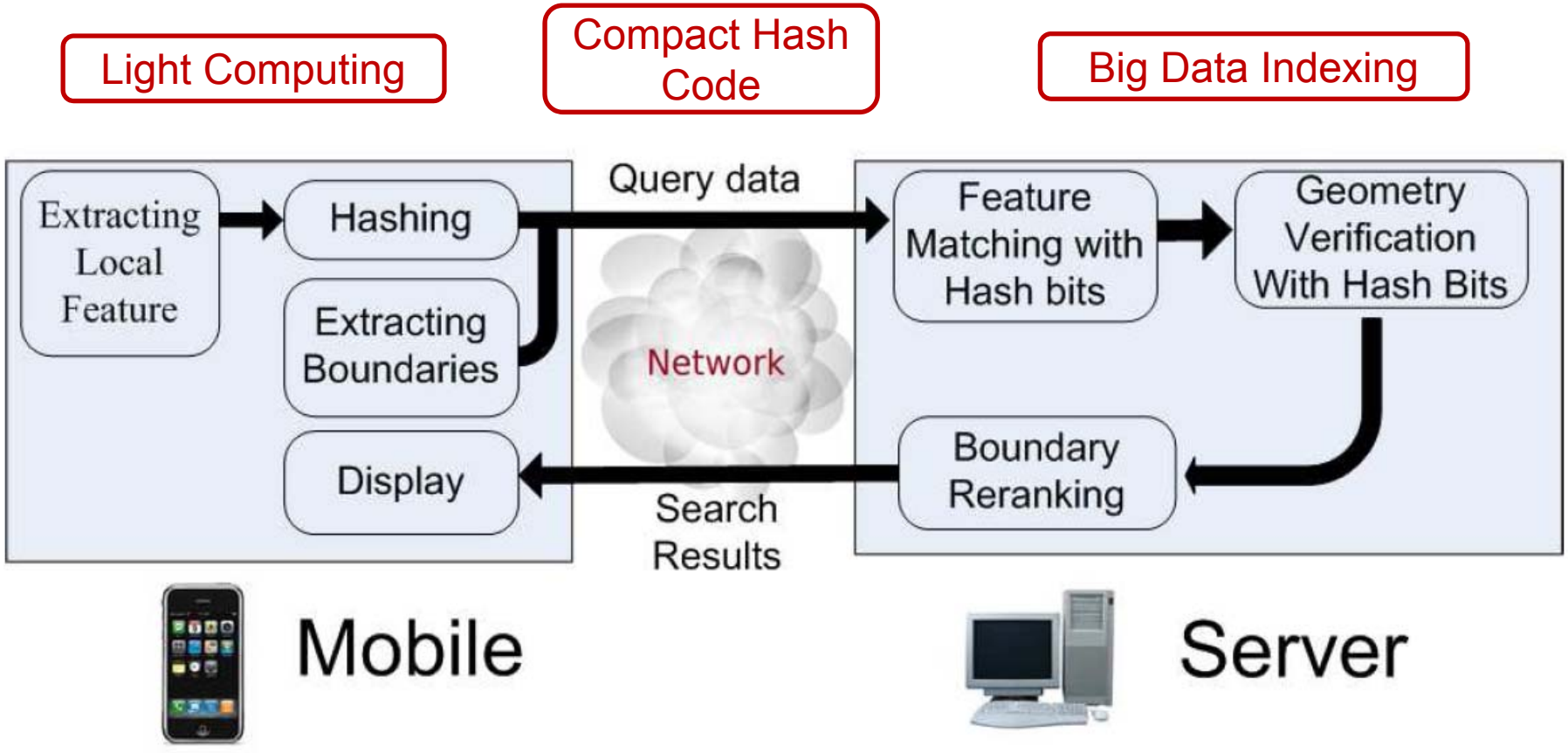
MVS calls for Distributed Optimization

Case Study (MVS, Girod *et al*, 2011)



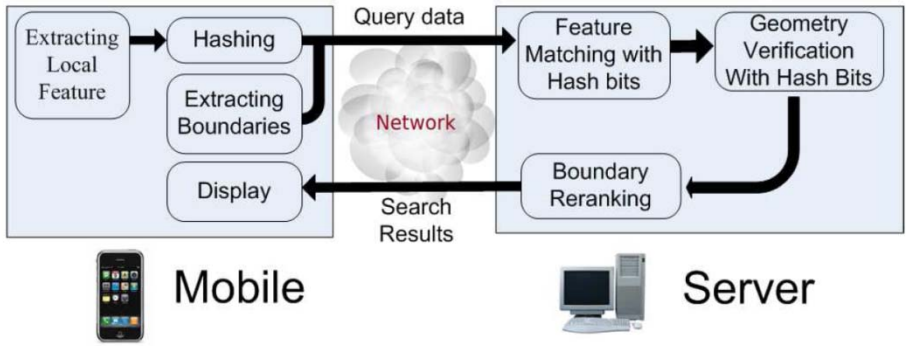
- Client:
  fast feature extraction
- Radio :
  transmit compact codes
- Server:
  scalable indexing over millions/billions

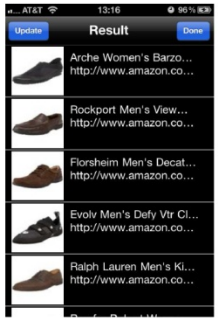
Mobile Search System by Hashing



He, Feng, Liu, Cheng, Lin, Chung, Chang. *Mobile Product Search with Bag of Hash Bits and Boundary Reranking*, CVPR 2012.

Columbia MPS System:
Bags of Hash Bits and Boundary features





Server:

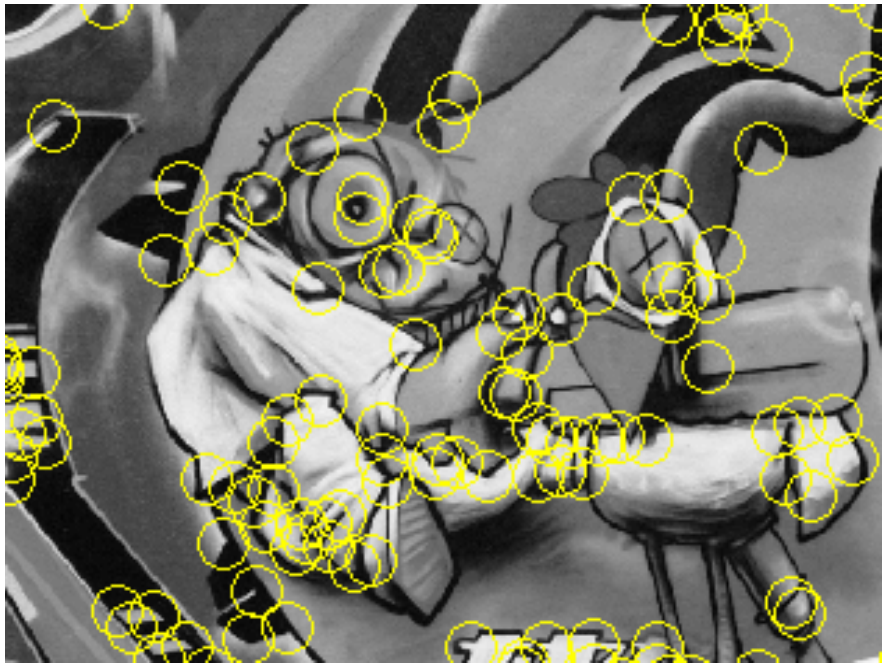
- **1 million product images crawled from Amazon, eBay and Zappos**
- **Hundreds of categories; shoes, clothes, electrical devices, groceries, kitchen supplies, movies, etc.**

Speed

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

video demo

Brief Review of Image Features

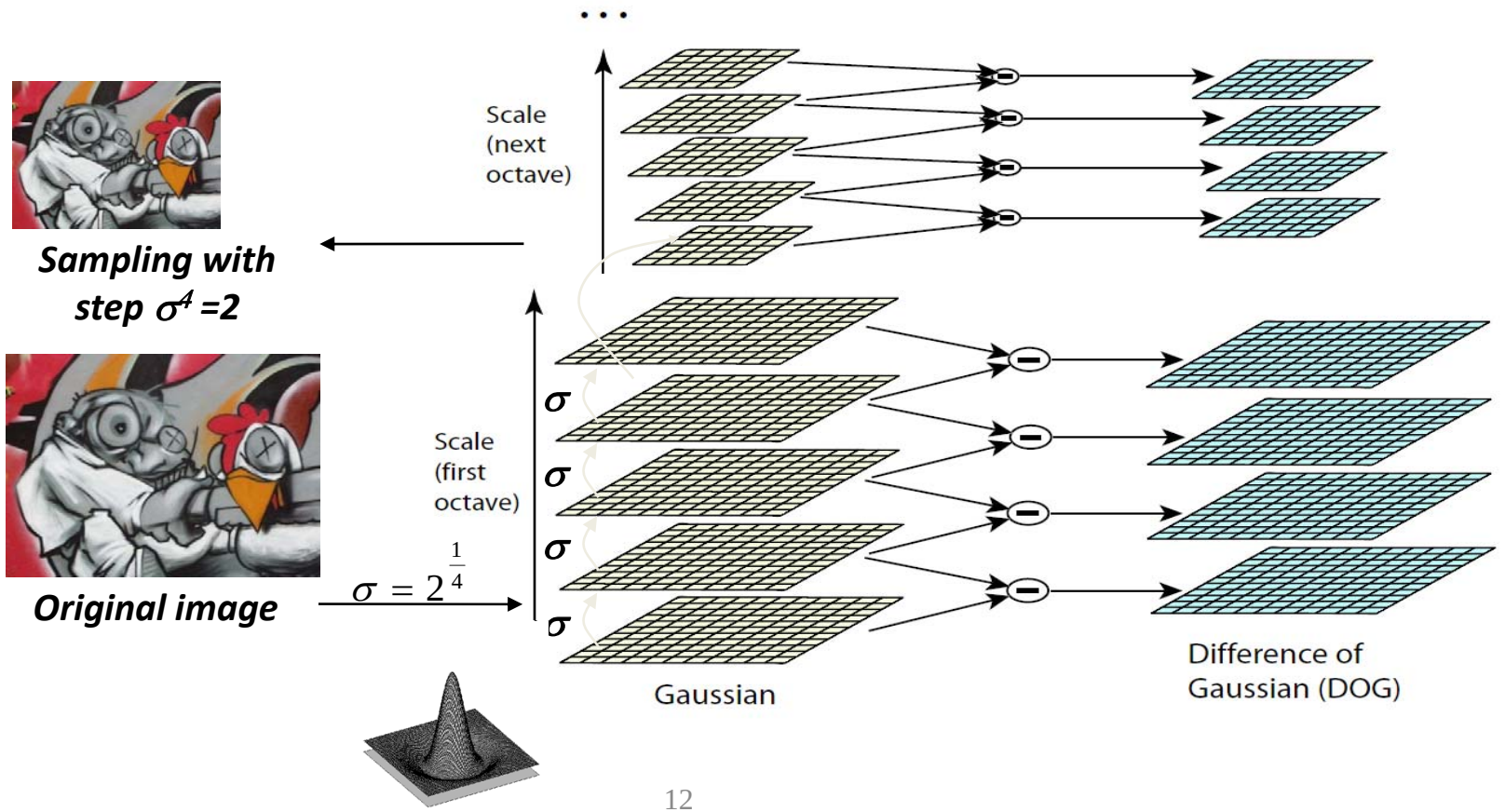


- Characterize visual content by local features (keypoints):
  - Interesting content
  - Precise localization
  - Repeatable detection under variations of scale, rotation, etc

(Slide of K. Grauman)

Example: keypoint detection

- Compute image Gaussian scale pyramid
- Keypoints from local maxima in scale space
- Many solutions: SIFT, SURF, MSER, BRIEF

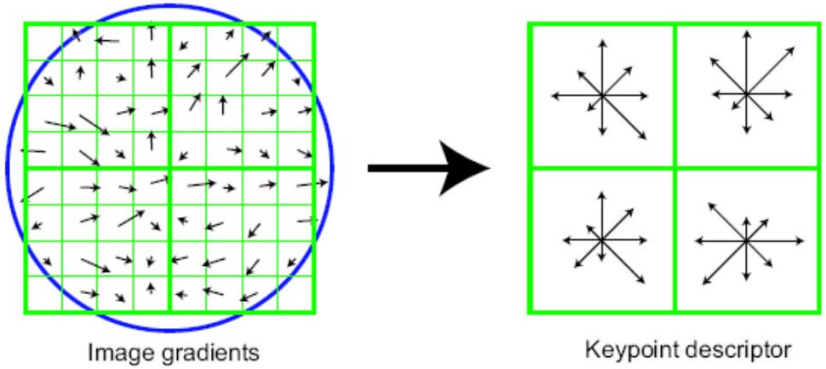


(Slide of K. Grauman)

Describe Appearance of Local Features

[Lowe, ICCV 1999]

Compute gradient
in a local patch

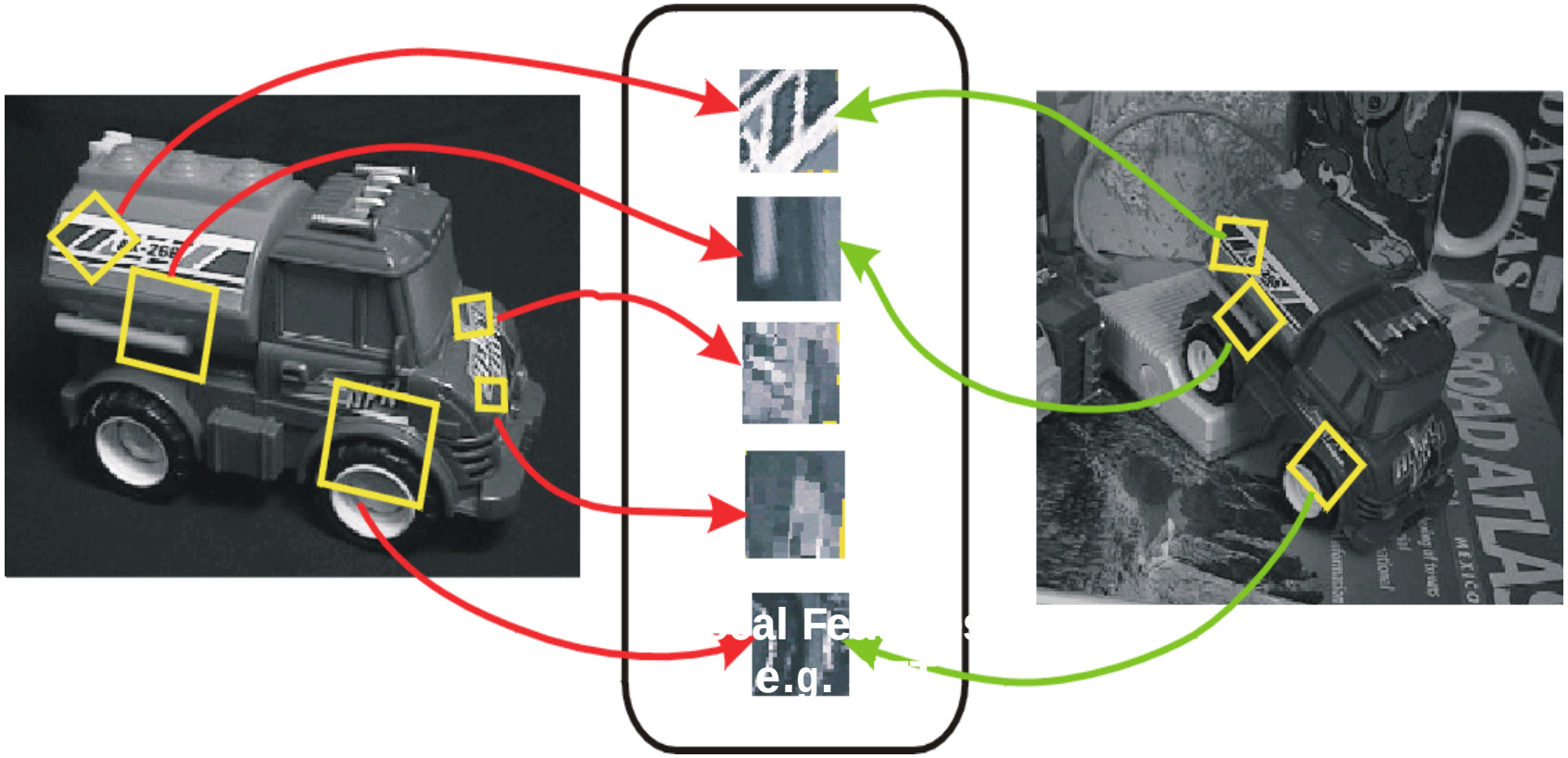


SIFT: Histogram of oriented gradients over local grids

- rotation invariant by orientation alignment
- scale invariant by scale space detection

Matching with Local Features

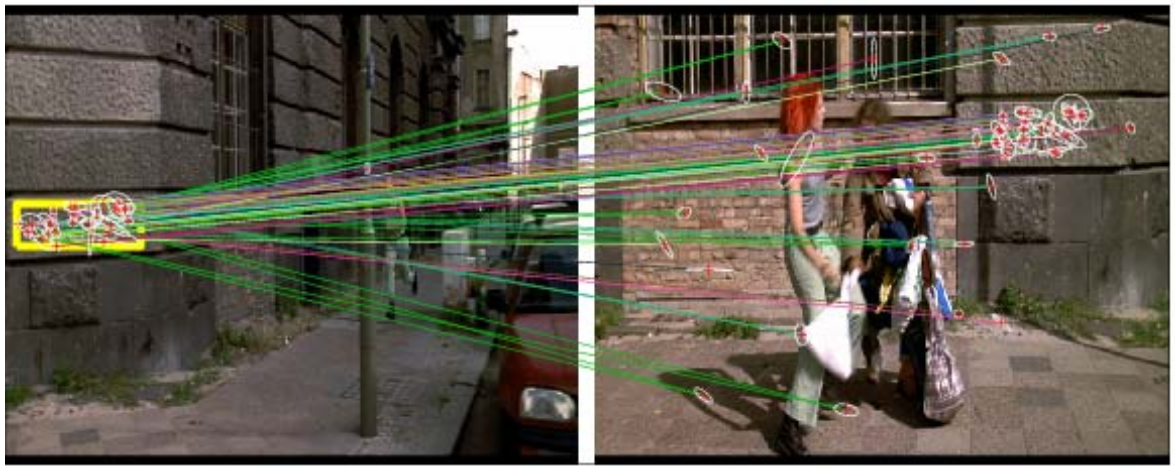
- local features facilitate robust matching over geometric and photometric transformations



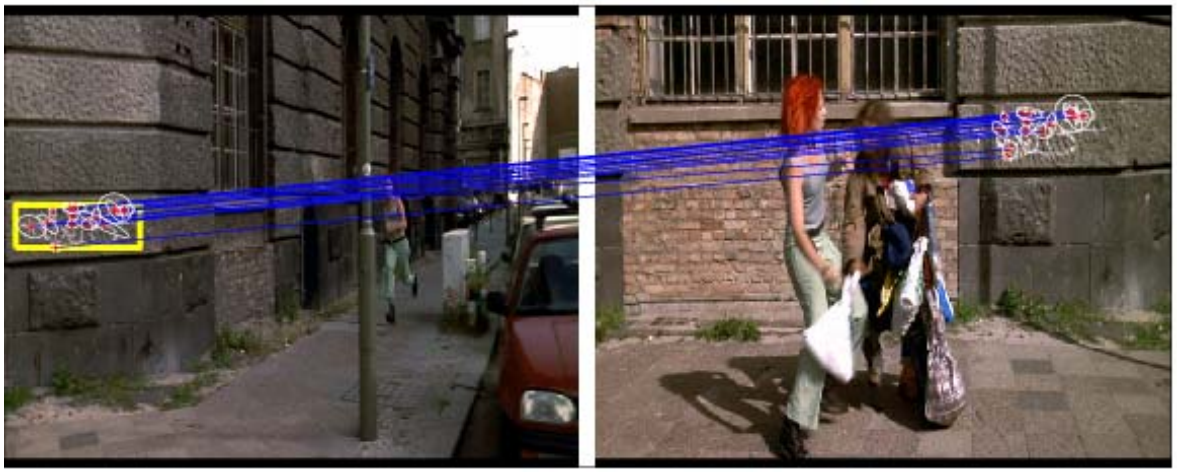
Slide: David Lowe

Example

Initial matches



Spatial consistency required



Slide : J. Sivic

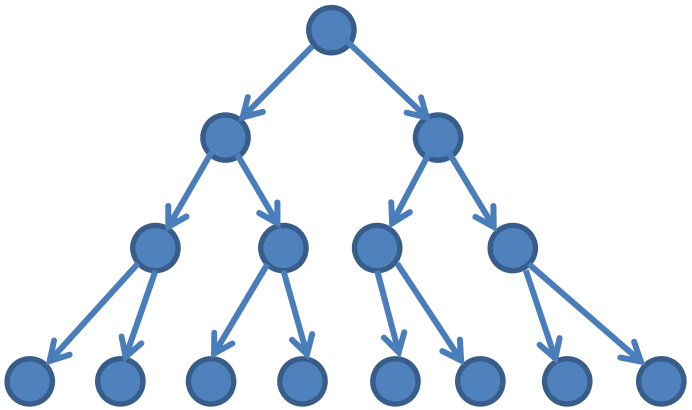
Estimate the Complexity

- 500 local features per image
  - file size ~128 Kbytes
  - more than 10 seconds for transmission over 3G
- Database indexing
  - 10 millions images need 5 billions local features
  - Finding matched features becomes challenging
- Idea:
  directly compute compact index codes on mobile devices

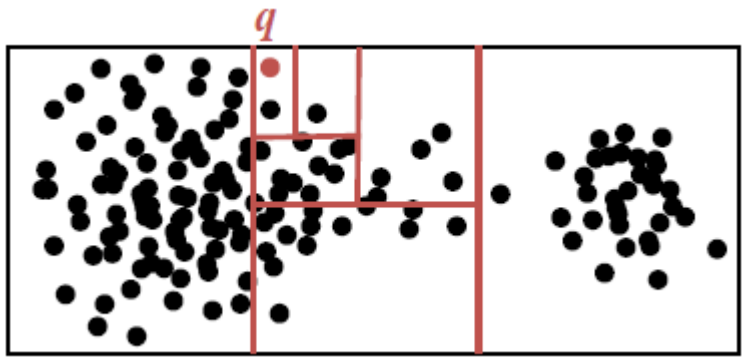
Standard Approach: Tree-Based Indexing

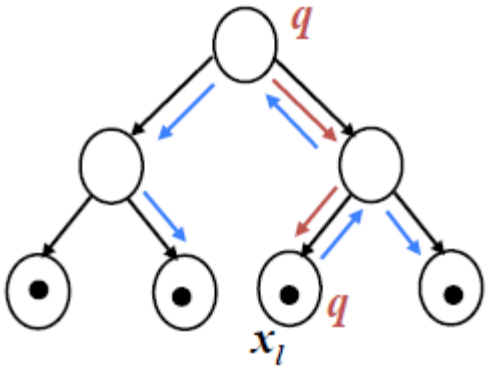
- $O(\log n)$ search time (20 bits for 1 million nodes)
- But "curse of high dimensionality" problem
- Hard to store on mobile devices

tree



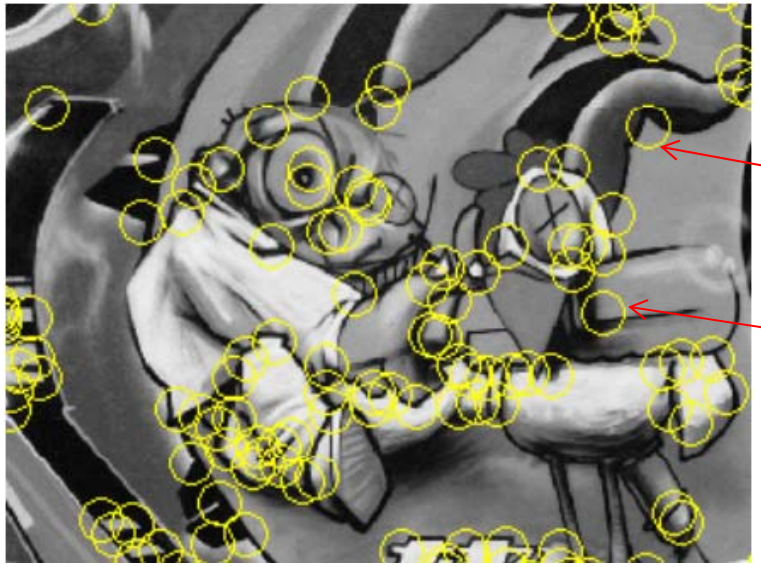
KD-tree





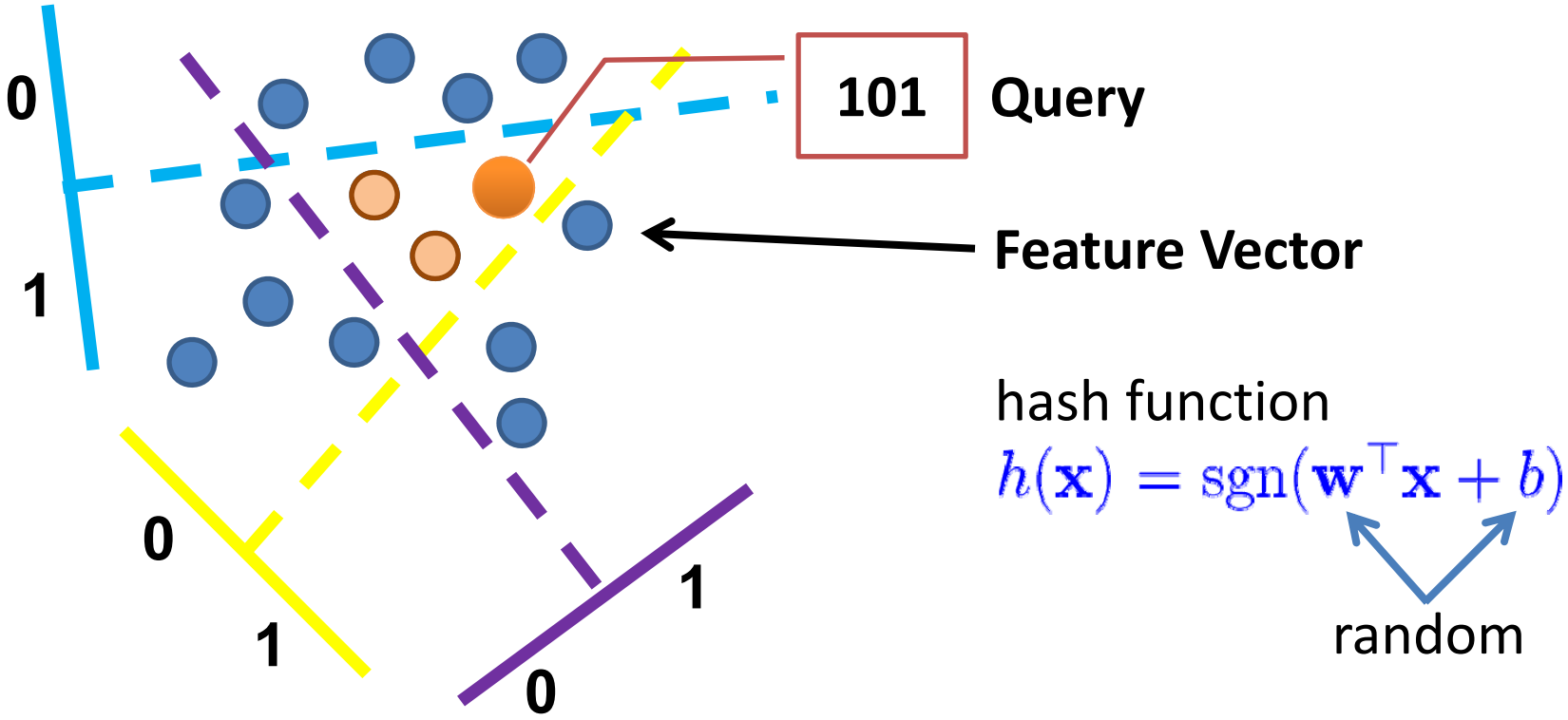
A Different Approach: hashing

- Each local feature coded as hash bits
  - locality sensitive, efficient for high dimensions
- Each image is represented as Bag of Hash Bits



Locality-Sensitive Hashing

[Gionis, Indyk, and Motwani 1999] [Datar et al. 2004]

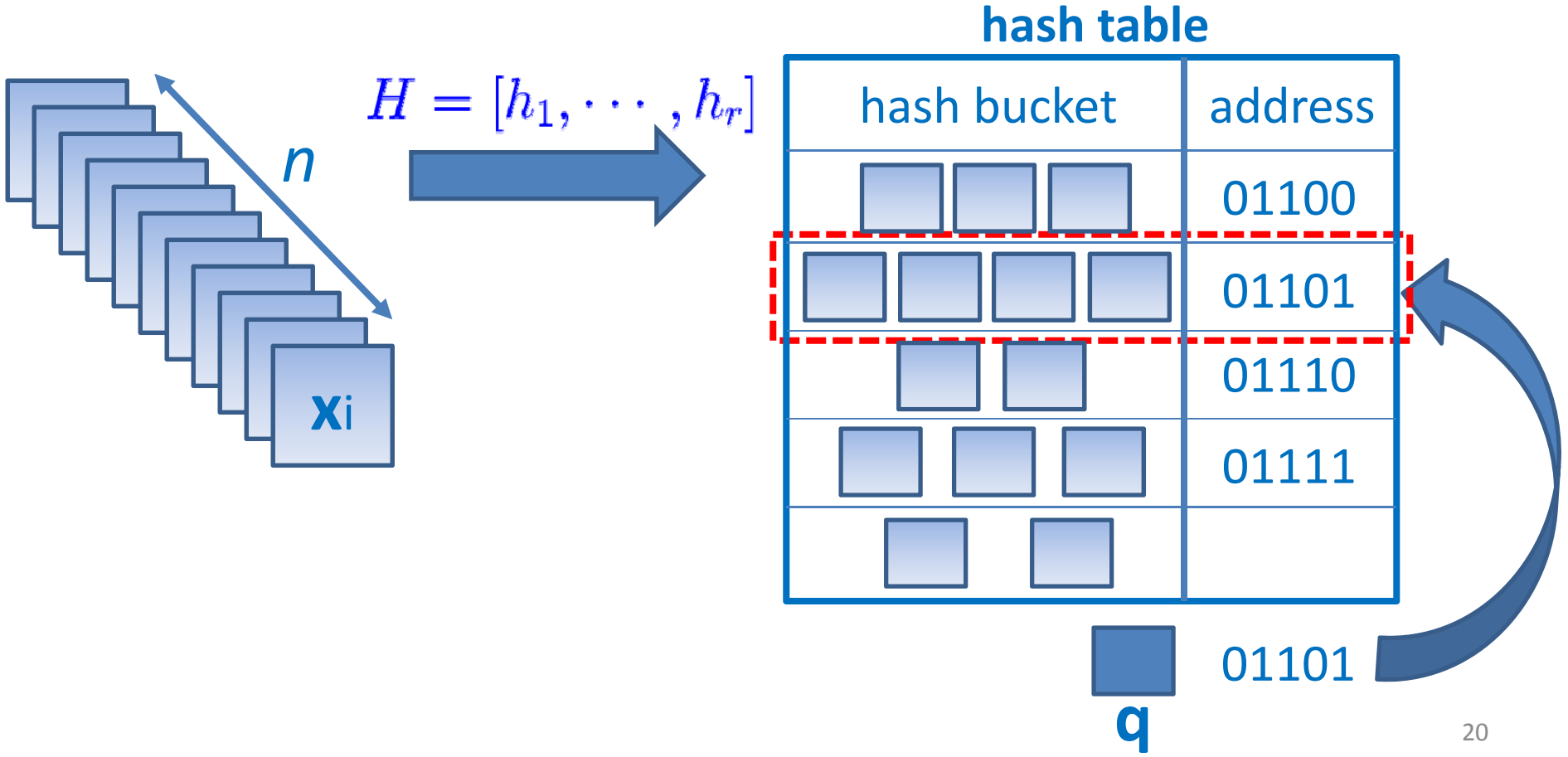


- Sublinear search time $O(n^{\frac{1}{1+\epsilon}})$ for ε-approximate NN search.

  x is an ε-approximate NN if $D(q, x) \leq (1+\varepsilon) D(q, x_{nn})$

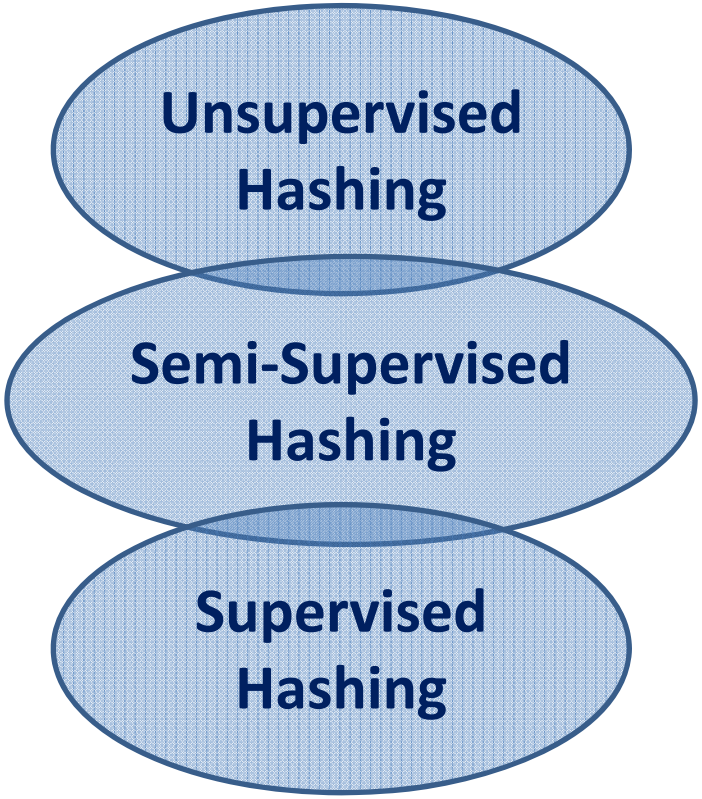
Efficient Search by Hash Table

- *O*(1) search time with short bits (<=50) and a single table.
- Both time and storage efficient.



Hashing: Active Research Topic

- Several categories published in KDD, ICML, CVPR, ICDM



LSH, PCAH, ITQ, KLSH, SH, AGH

SSH, WeaklySH

RBM, BRE, MLH, LDAH

Unsupervised hash

- Principle – explore data distributions
  - Similar hash codes for similar points (accuracy)
  - Balanced and non-redundant hash bits (time)

SPICA Hash, He et al, CVPR 11

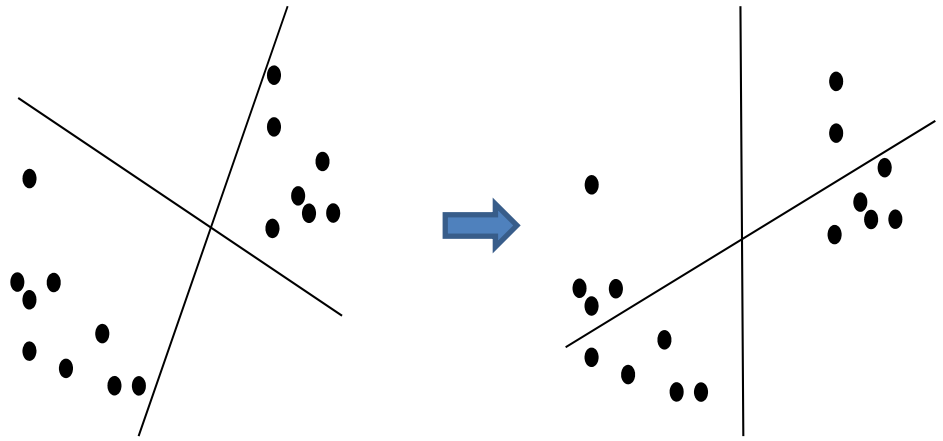
$$D(Y) = \sum_{p,q=1}^{N} W_{pq} \| Y_p - Y_q \|^2 \leq \eta$$

Search accuracy

$$\min I(y_1, \ldots, y_k, \ldots, y_M)$$

$$\text{while } E(y) = \sum_{p=1}^{N} Y_p = 0$$

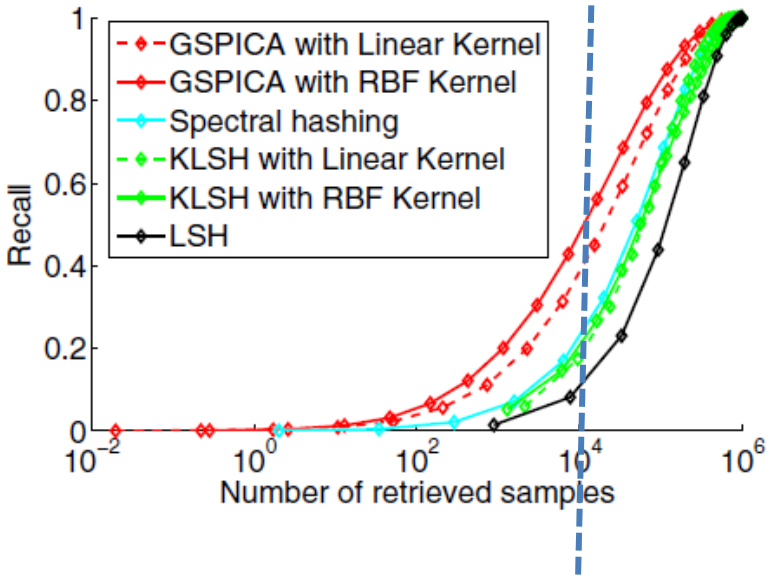
Balanced bucket size



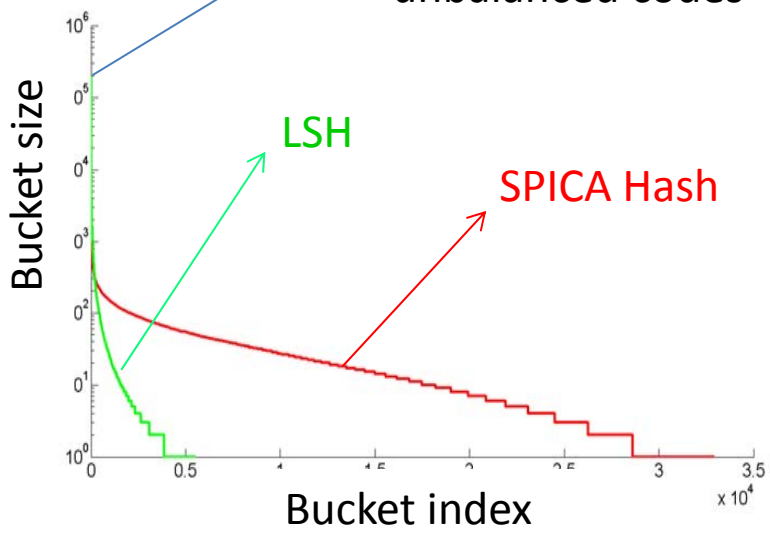
Learning Based Hashing vs. Random Hashing

- SPICA Hash: jointly optimize search accuracy & time

Reduce candidate set from
1Million to 10K @ 50% recall

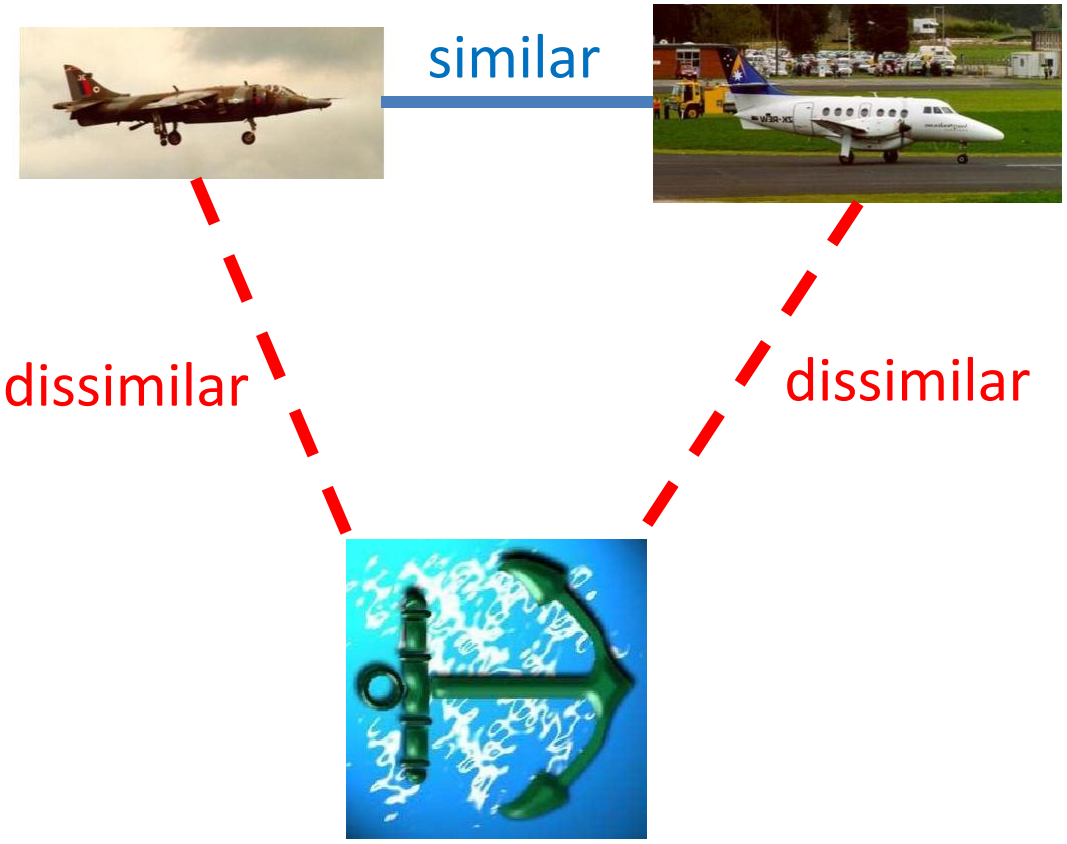


Random LSH often leads to
unbalanced codes

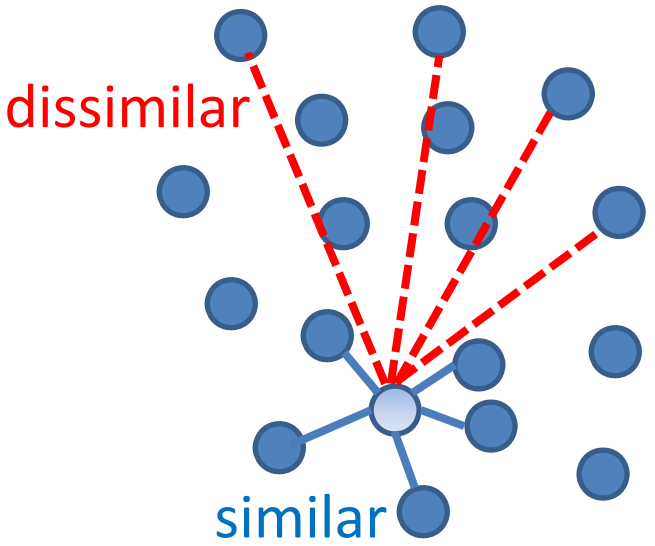


If there is supervised information

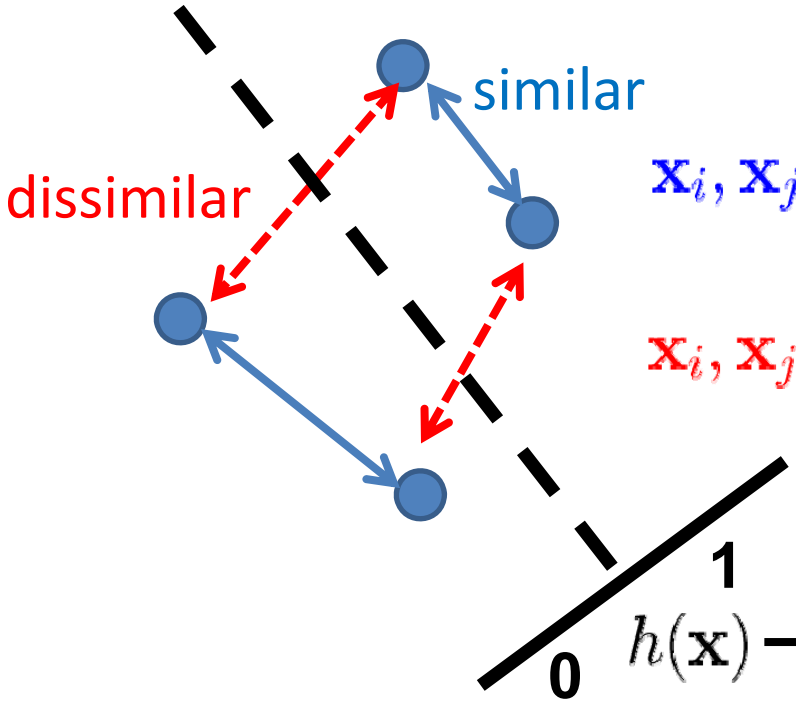
Semantic Supervision



Metric Supervision



Design Hash Codes to Match Supervised Information



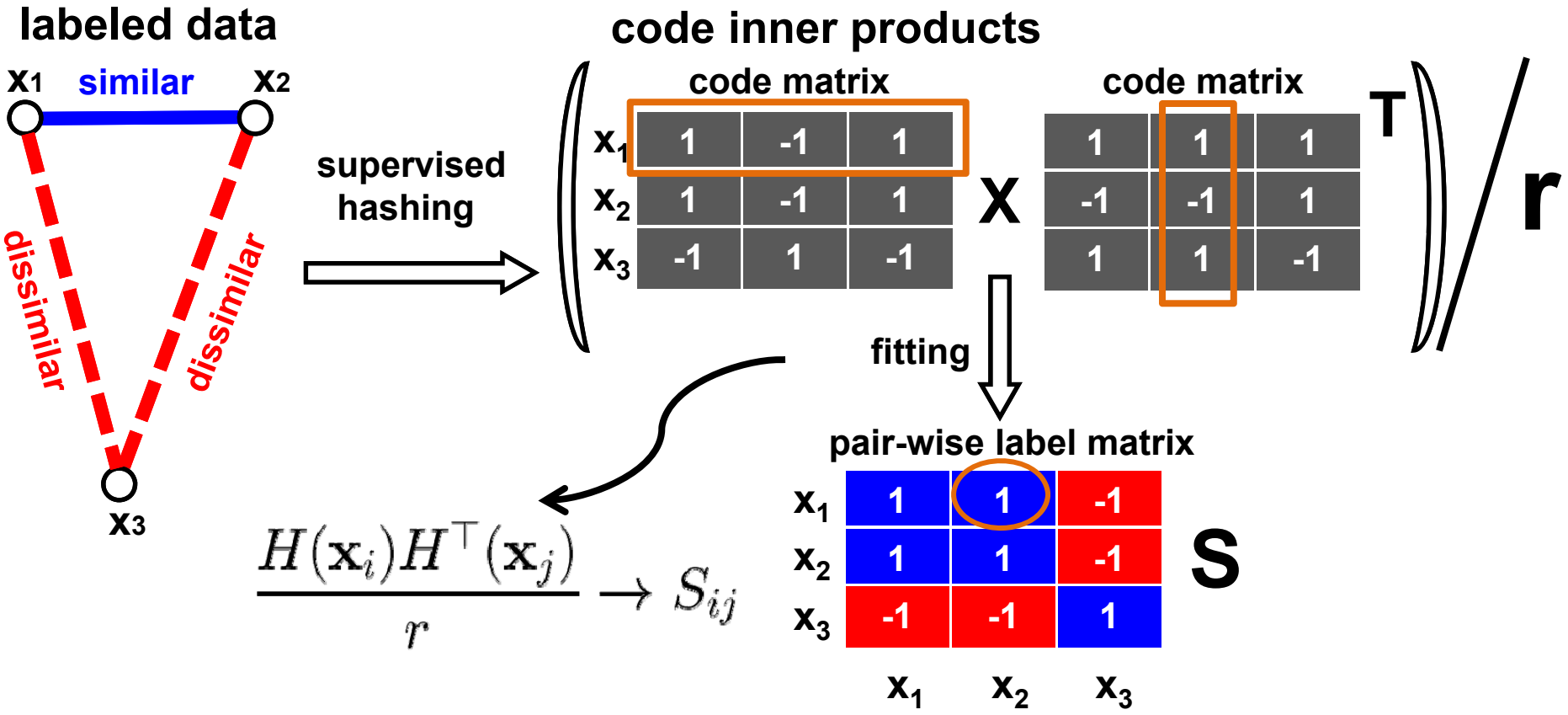
$\mathbf{x}_i, \mathbf{x}_j$ are similar $\Rightarrow h(\mathbf{x}_i) = h(\mathbf{x}_j)$

$\mathbf{x}_i, \mathbf{x}_j$ are dissimilar $\Rightarrow h(\mathbf{x}_i) \neq h(\mathbf{x}_j)$

$h(\mathbf{x}) \longrightarrow$ Preferred hashing

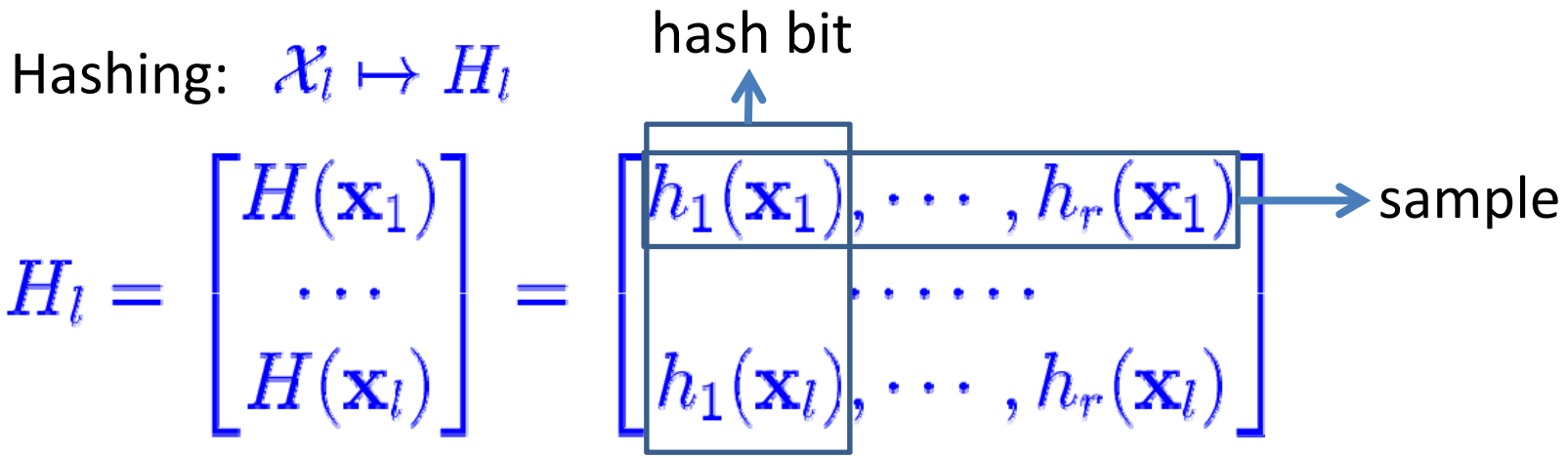
Use Code Inner Products to Match Supervised Labels

Liu, Wang, Ji, Jiang, Chang, CVPR2012



Learning Supervised Hash

Hashing: $\mathcal{X}_l \mapsto H_l$



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

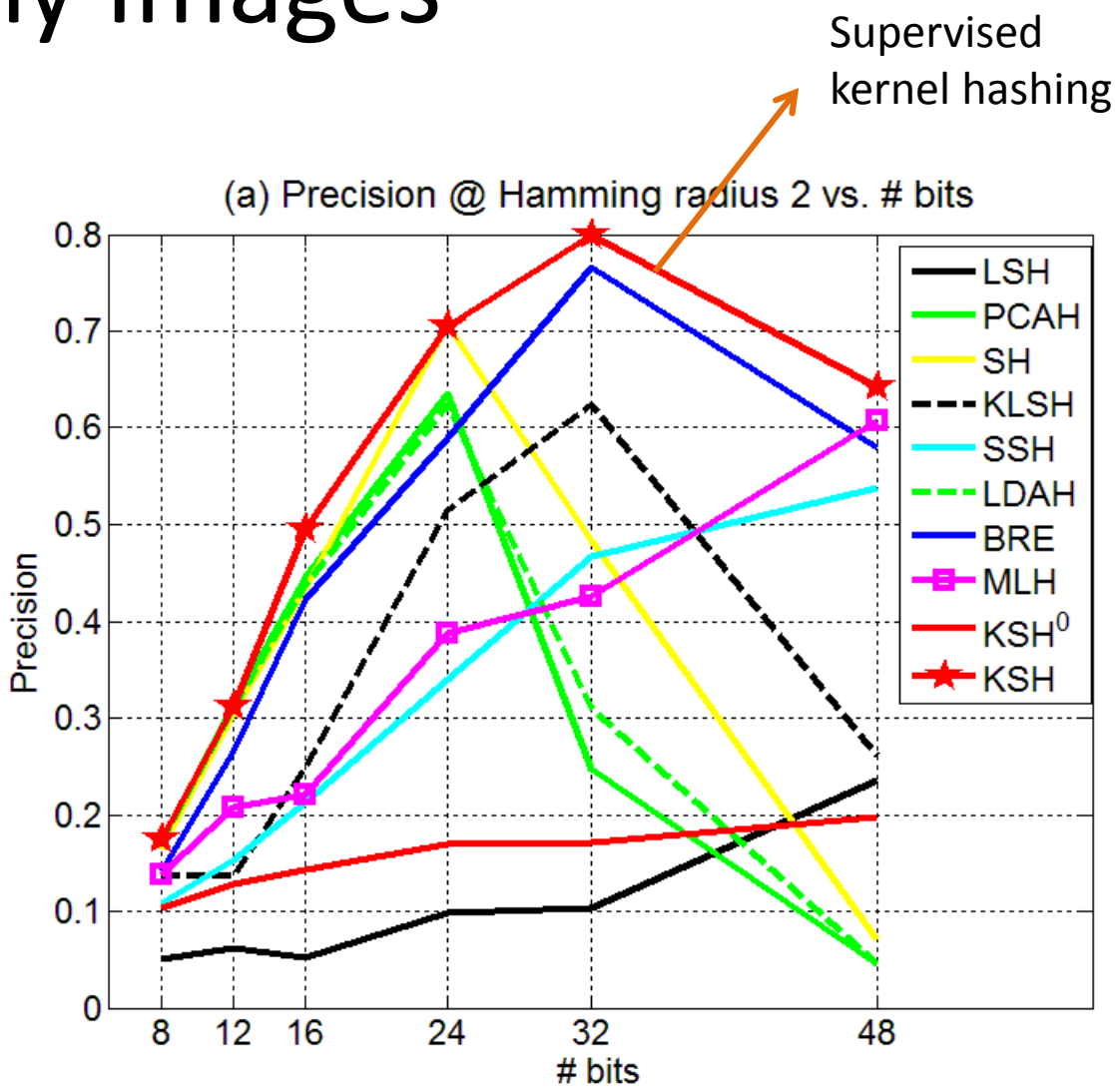
Design hash codes to match supervised information

- Easy to optimize and extend to kernels
- Sequential learning

1 Million Tiny Images

Torralba and Fergus, TPAMI 2008

- Search 1 million images from Web
- 2000 random images as queries
- Top 5000 nearest samples as consistent pairs



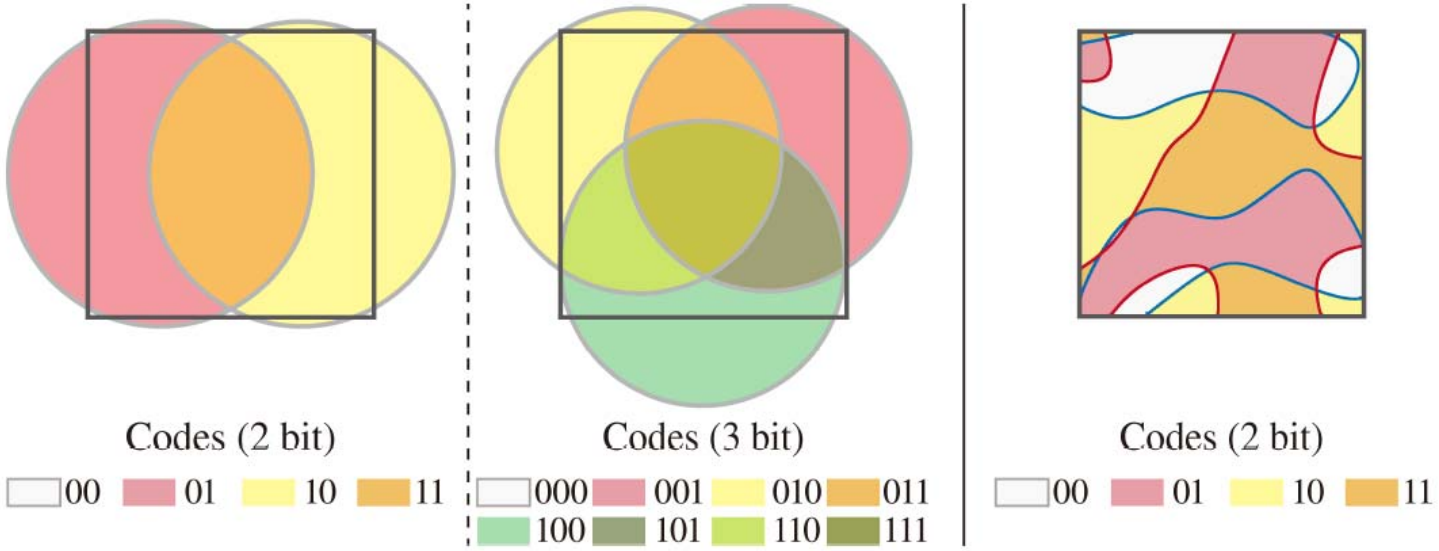
Spherical Hashing

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

- linear projection -> spherical partitioning

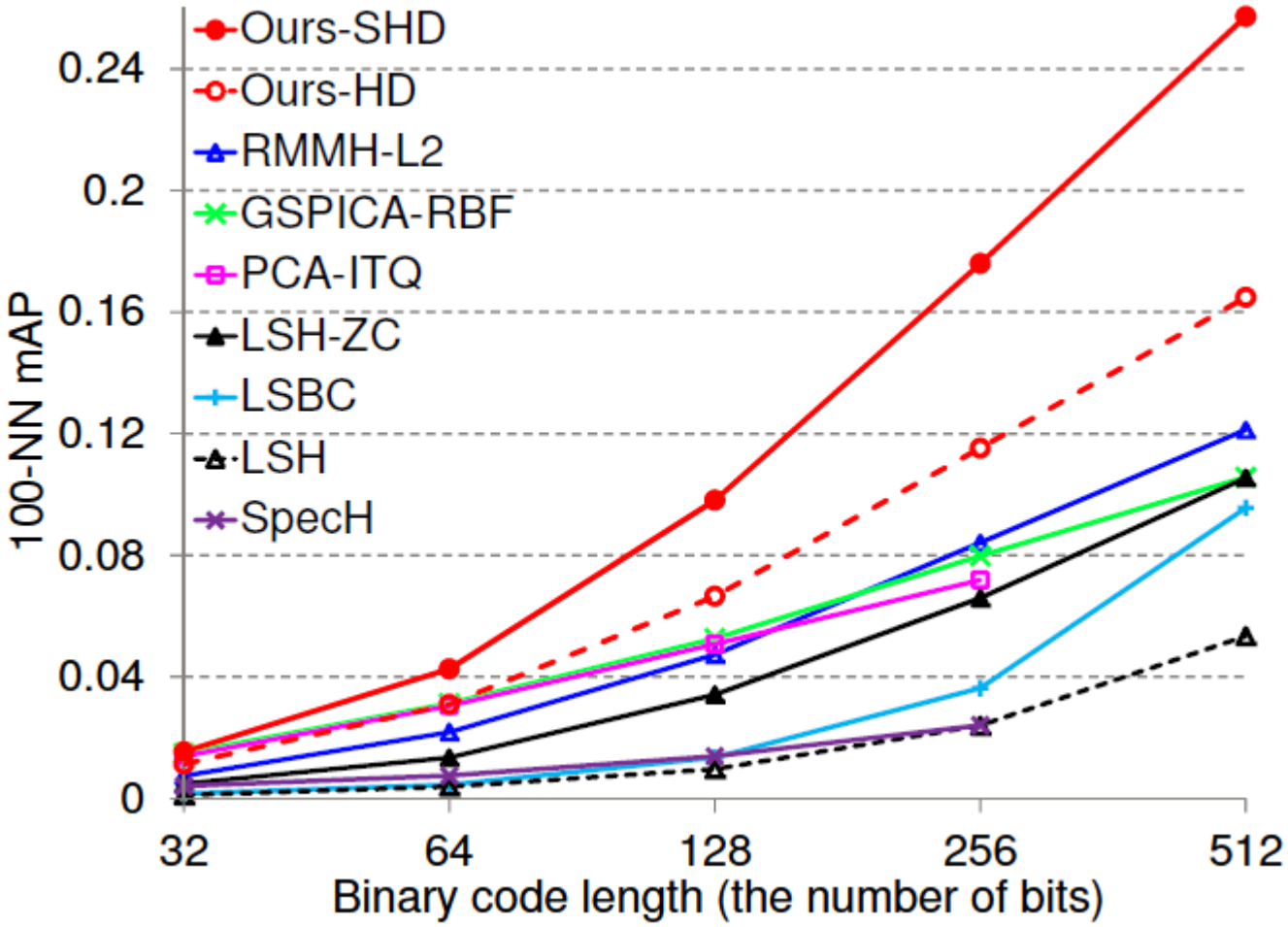
$$h_k(x) = \begin{cases} -1 & \text{when } d(p_k, x) > t_k \\ +1 & \text{when } d(p_k, x) \leq t_k \end{cases}$$

- Asymmetrical hash bits: tighter regions for +1
- Learning: find optimal spheres in the space

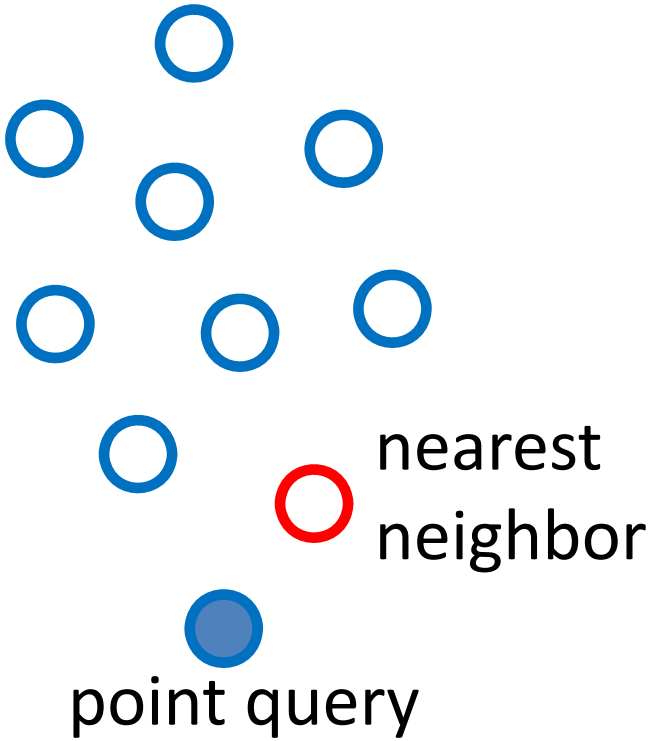


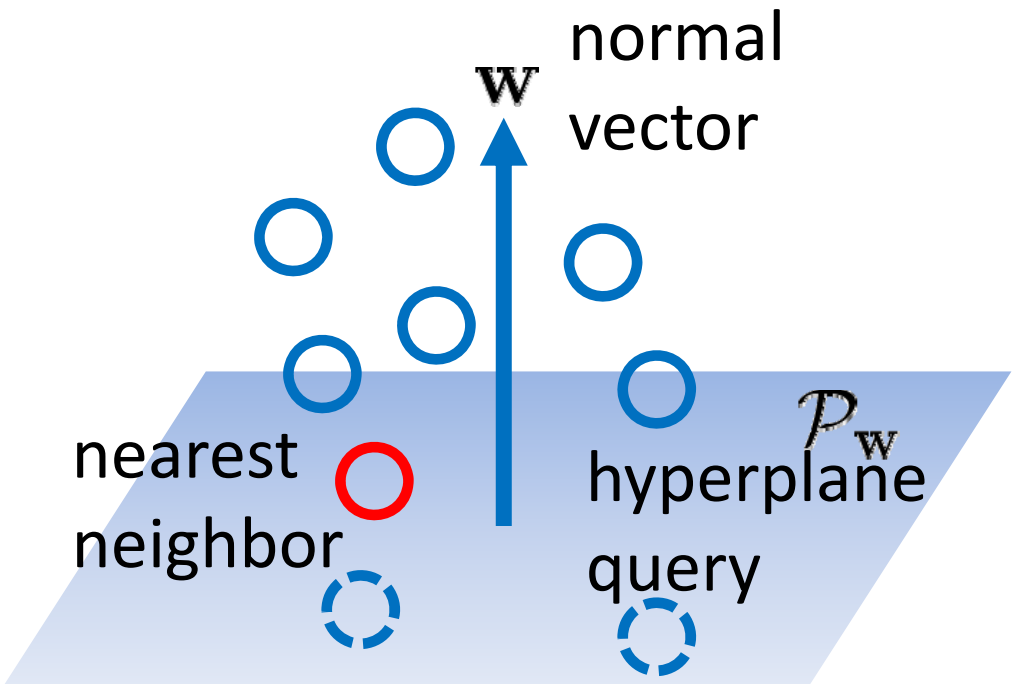
Spherical Hashing Performance

- 1 Million Images: GIST 384-D features

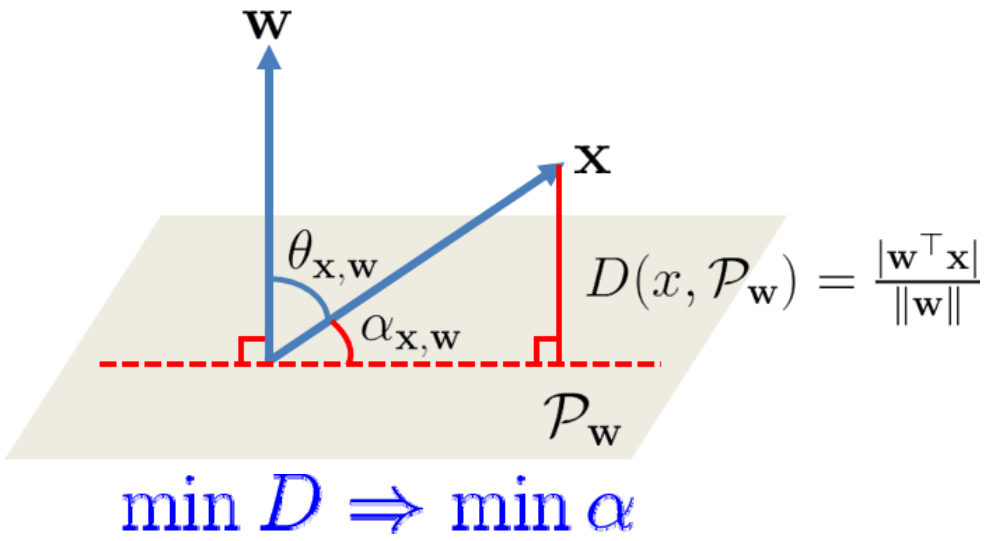


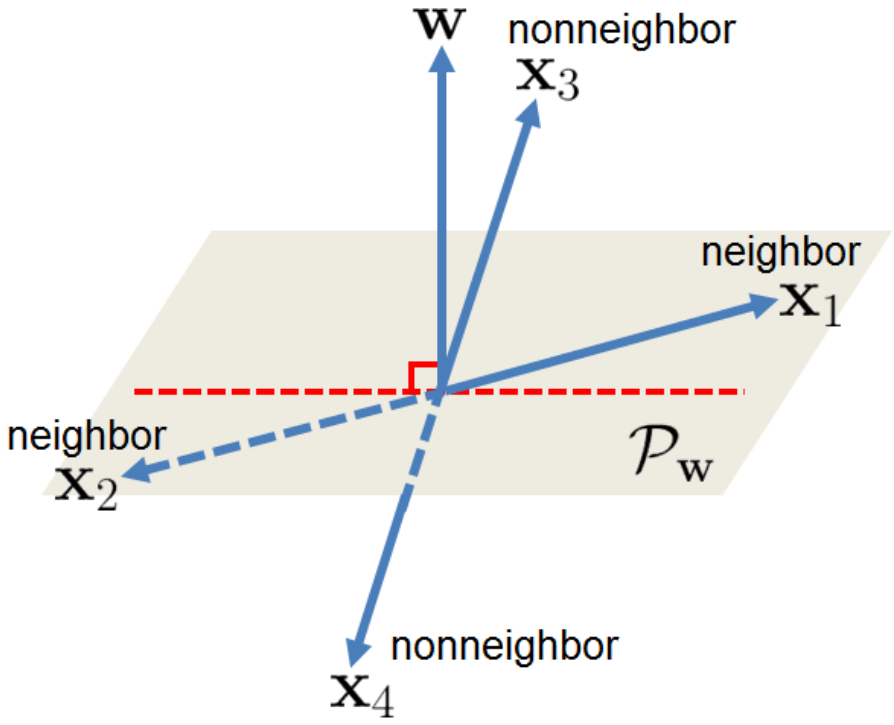
Point-to-Point Search vs. Point-to-Hyperplane Search





Hashing Principle: Point-to-Hyperplane Angle





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

Bilinear Hashing

Liu, Jun, Kumar, Chang, ICML12

Bilinear-Hyperplane Hash (BH-Hash)

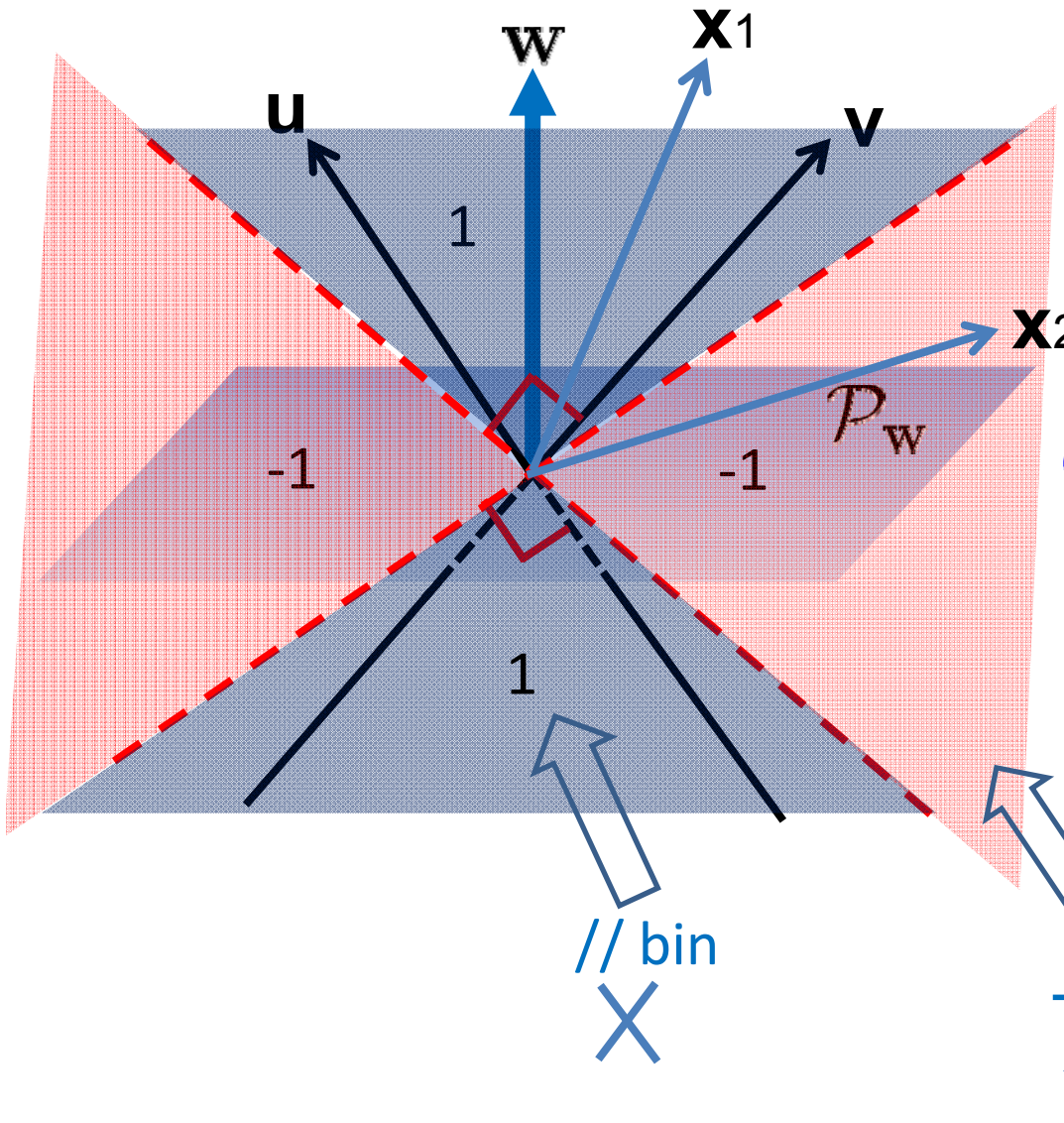
$$h^{\mathcal{B}}(\mathbf{z}) = \mathrm{sgn}(\mathbf{u}^\top \mathbf{z}\mathbf{z}^\top \mathbf{v}), \text{ i.i.d. } \mathbf{u}, \mathbf{v} \sim \mathcal{N}(0, I_{d\times d}).$$

query normal **w** or database point **x**

2 random projection vectors

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

A Single Bit of Bilinear Hash



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

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

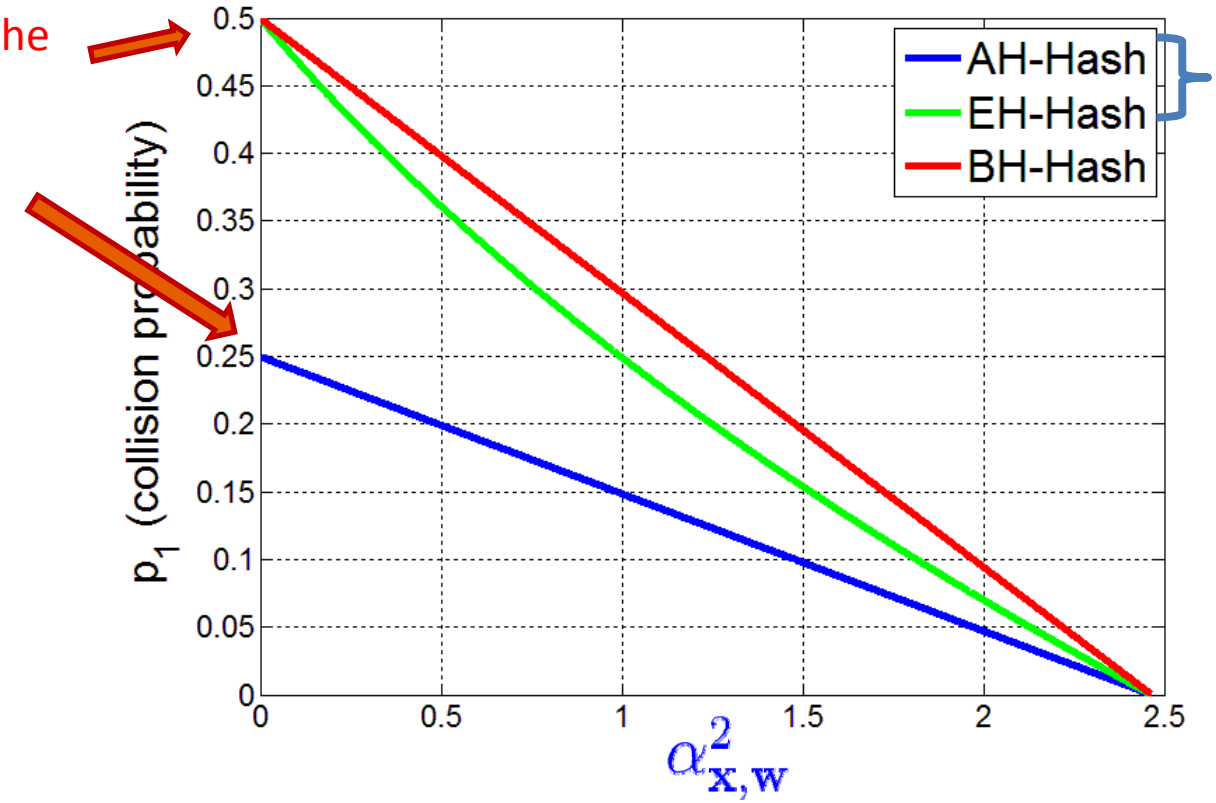
$$h^{\mathcal{B}}(\mathbf{x}_2) = \mathrm{sgn}(\mathbf{u}^\top \mathbf{x}_2)\mathrm{sgn}(\mathbf{v}^\top \mathbf{x}_2) = -1 \cdot 1 = -1$$

Theoretical Collision Probability

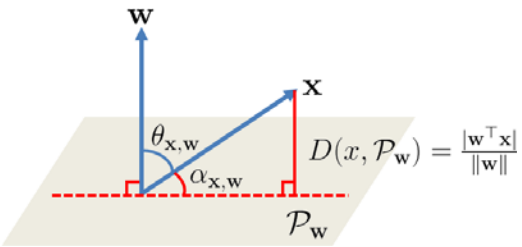
$$\mathbf{Pr}\left[h^{\mathcal{B}}(\mathbf{w}) \neq h^{\mathcal{B}}(\mathbf{x})\right] = \frac{1}{2} - \frac{2(\theta_{\mathbf{x},\mathbf{w}} - \frac{\pi}{2})^2}{\pi^2} = \frac{1}{2} - \frac{2\alpha_{\mathbf{x},\mathbf{w}}^2}{\pi^2}$$

highest collision probability for active hashing so far

Double the collision prob

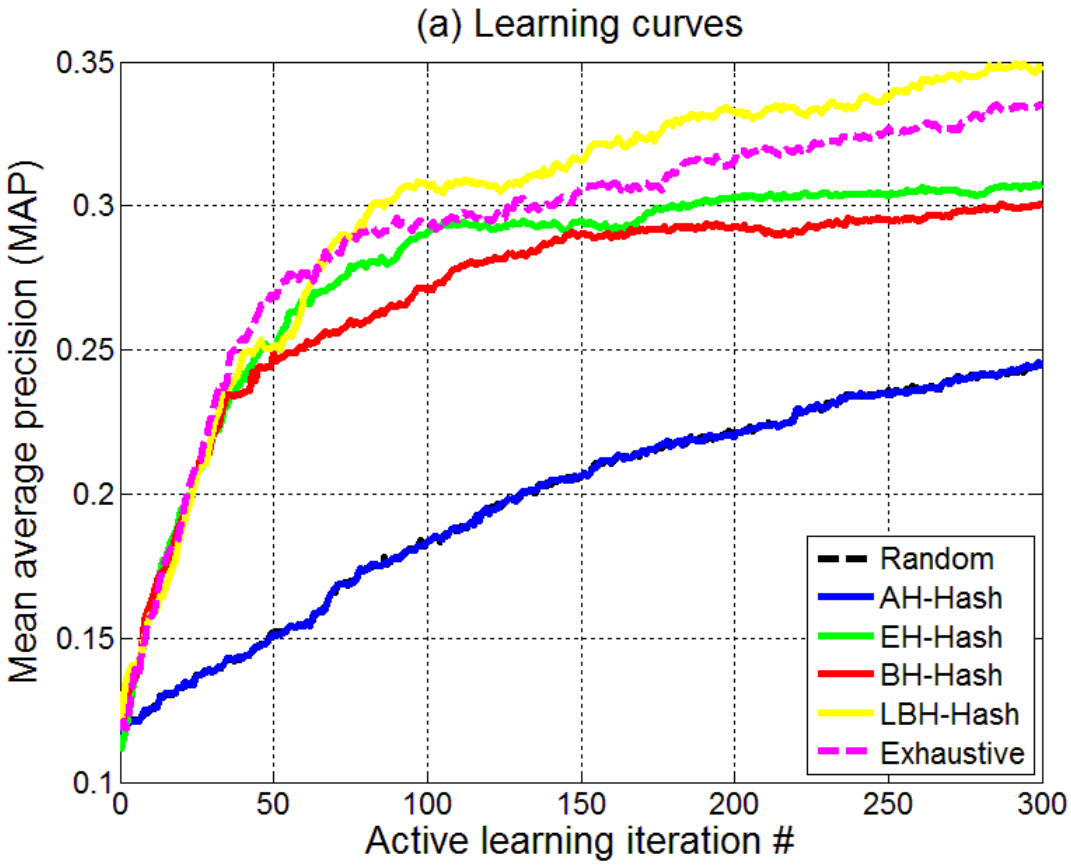


Jain et al. ICML 2010



Active SVM Learning with Hyperplane Hashing

- Linear SVM Active Learning over 1 million data points

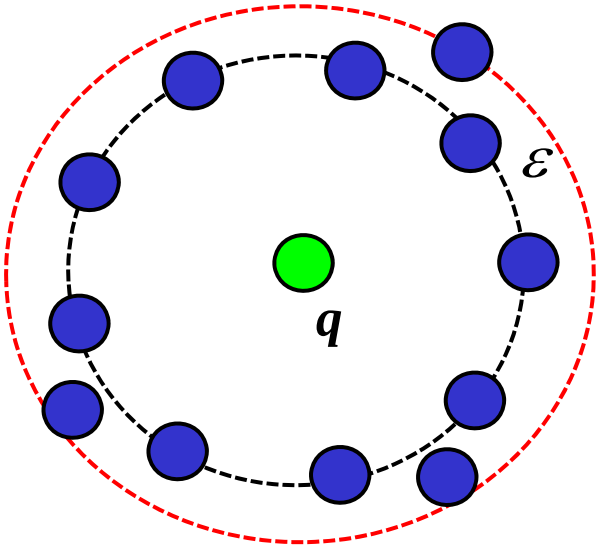


Understand Difficulty of Approximate Nearest Neighbor Search

He, Kumar, Chang, ICML 2012

- How difficult is approximate nearest neighbor search in a dataset?

Toy example



Search not meaningful!

x **is an** ε**-approximate NN if** $D(q,x) \le (1+\varepsilon) D(q,x_{nn})$

A concrete measure of difficulty of search in a dataset?

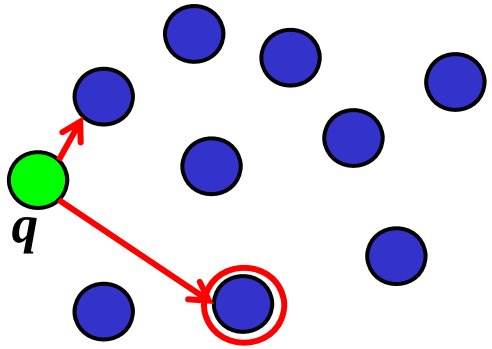
Relative Contrast

He, Kumar, Chang, ICML 2012

- A naïve search approach: Randomly pick a point and return that to be the NN

Relative Contrast

$$C_r = \frac{D_{random}}{D_{nn}} = \frac{E_x[D(q,x)]}{D(q,x_{nn})}$$



$$C_r = \frac{E_{q,x}[D(q,x)]}{E_q[D(q,x_{nn})]}$$

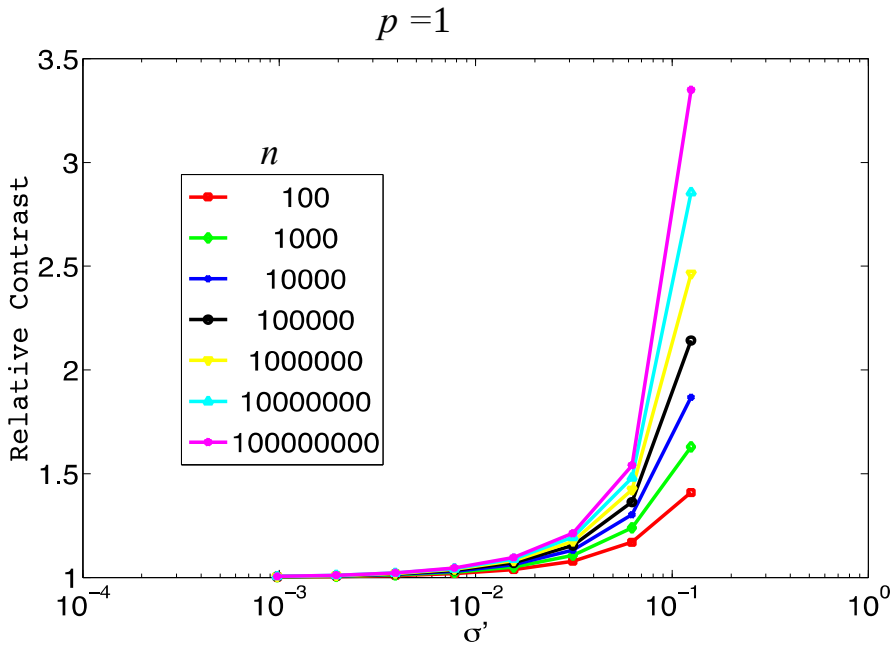
- **High Relative Contrast → easier search**
- **If $C_r \to 1$, search not meaningful**

Estimation of Relative Contrast

- With CLT, and binomial approximation

$$C_r = \frac{D_{random}}{D_{nn}} \approx \frac{1}{[1+\phi^{-1}(\phi(-1/\sigma')+1/n)\sigma']^{1/p}}$$

ϕ - standard Gaussian cdf

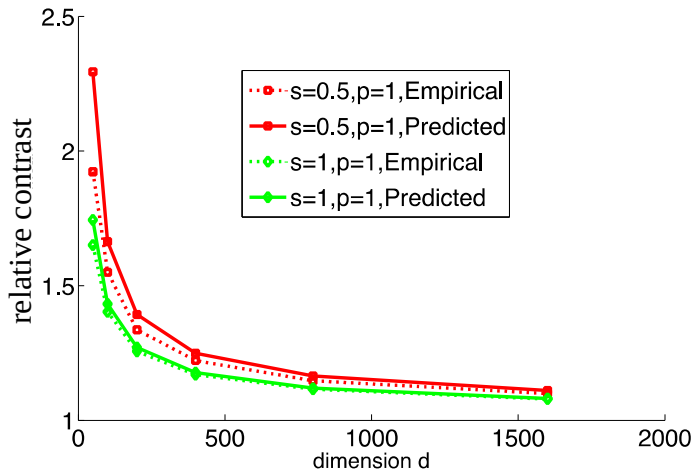


σ' – a function of data properties
e.g., dimensionality and sparsity

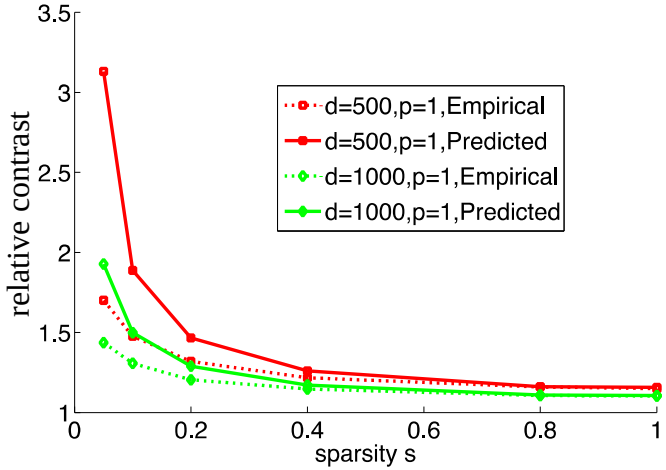
$$d \to \infty \Rightarrow \sigma' \to 0 \Rightarrow C_r \to 1$$

Synthetic Data

- Data sampled randomly from $U[0,1]$



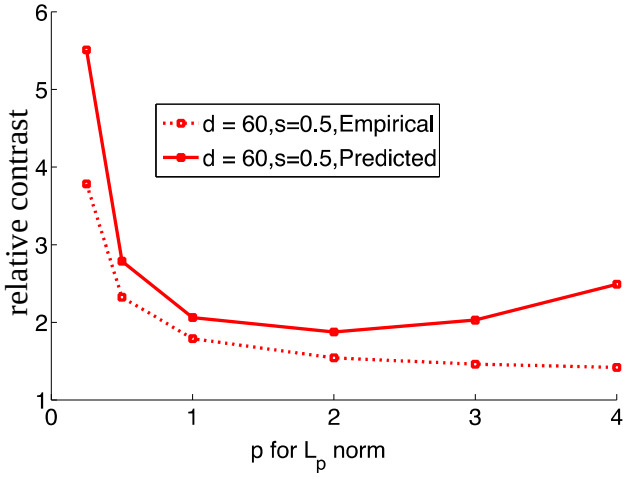
higher dimensionality → bad



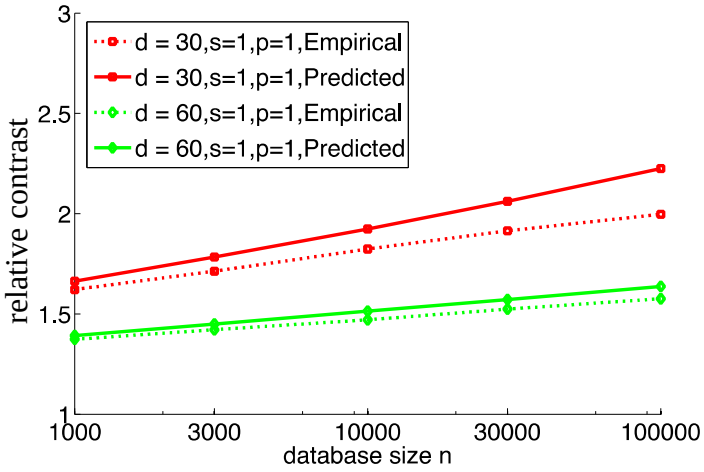
sparser vectors → good

Synthetic Data

- Data sampled randomly from $U[0,1]$



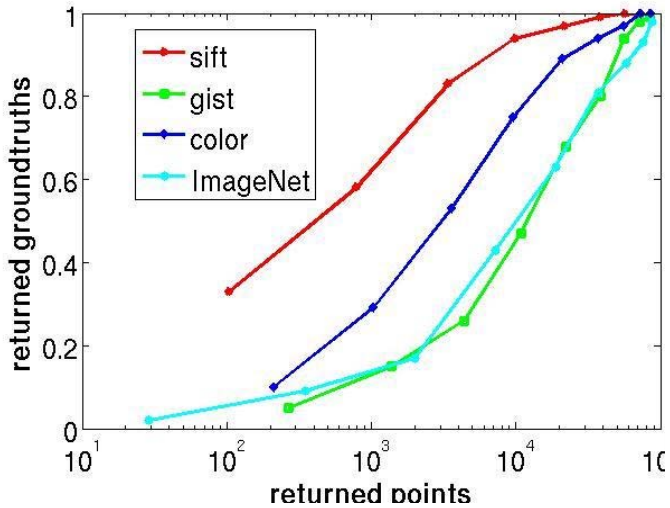
lower p → good



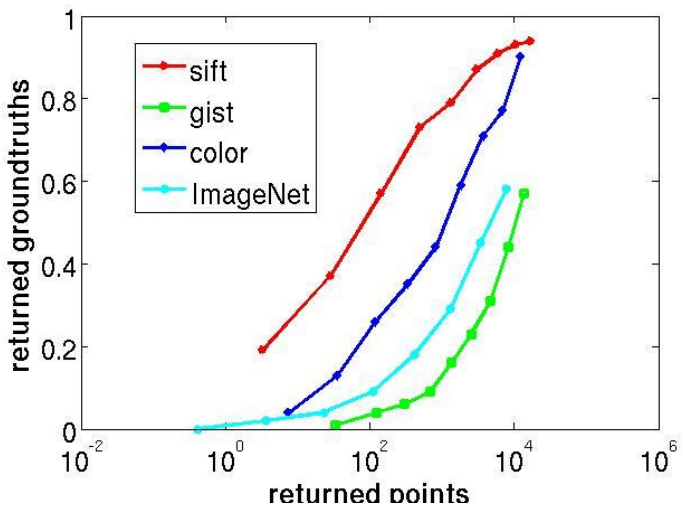
Larger database → good

Predict Hashing Performance of Real-World Data

Dataset	Dimensionality (d)	Sparsity (s)	Relative Contrast (C_r) for $p = 1$
SIFT	128	0.89	4.78
Gist	384	1.00	1.83
Color Hist	1382	0.027	3.19
Imagenet BoW	10000	0.024	1.90



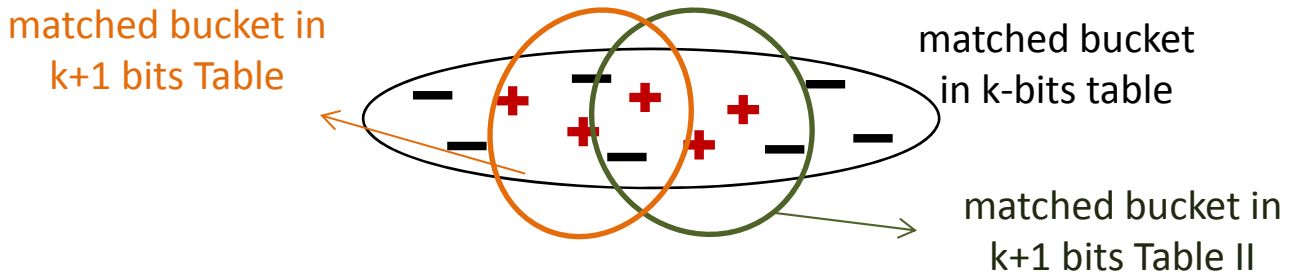
28 bits



16 bits

Multi-Table Hashing

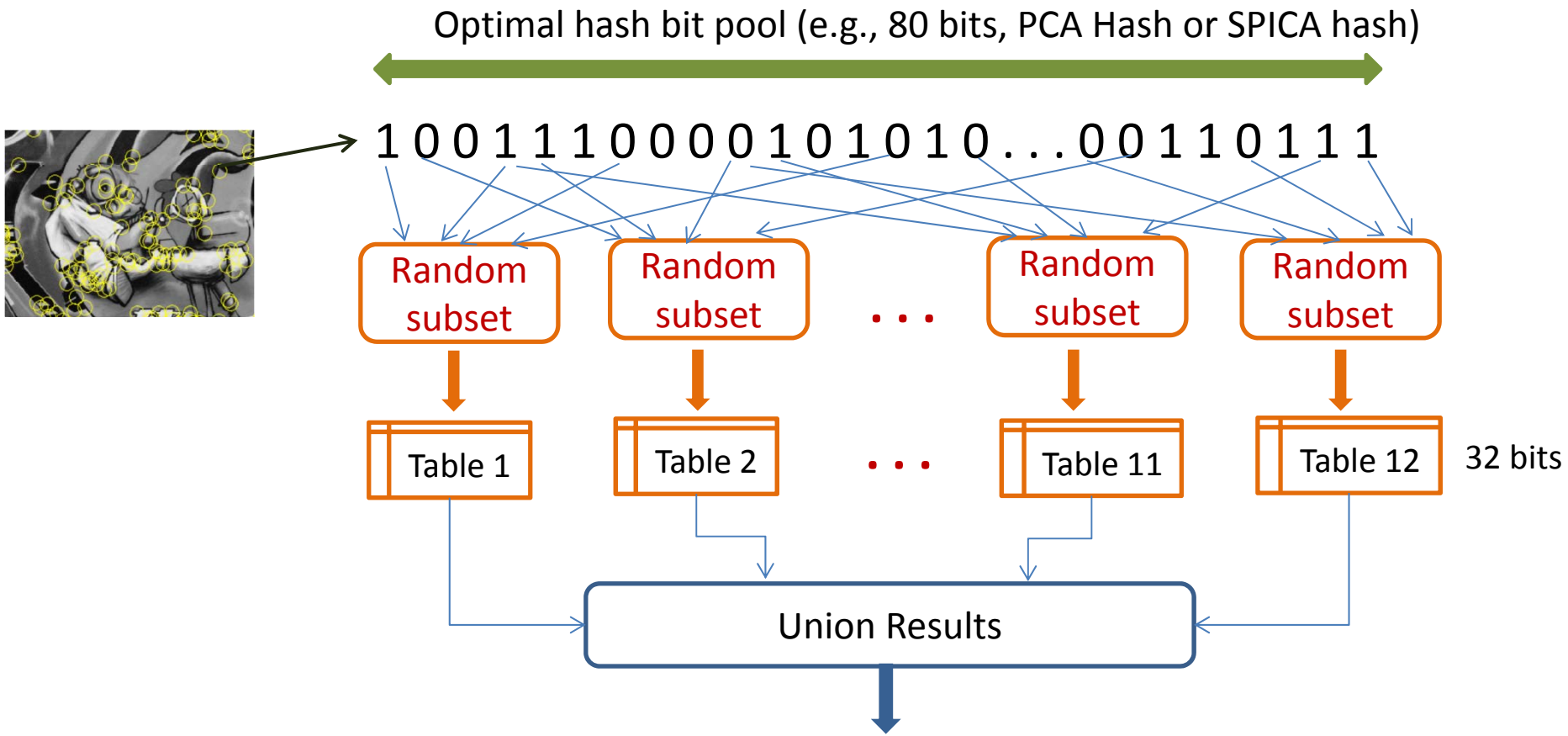
- Larger table increases precision but degrades recall
- Common trick: multi-table hashing



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

Bit Reuse for Multi-Table Hashing

- To reduce transmission size
  - Reuse top optimal hash bits by random sampling



Data Sets

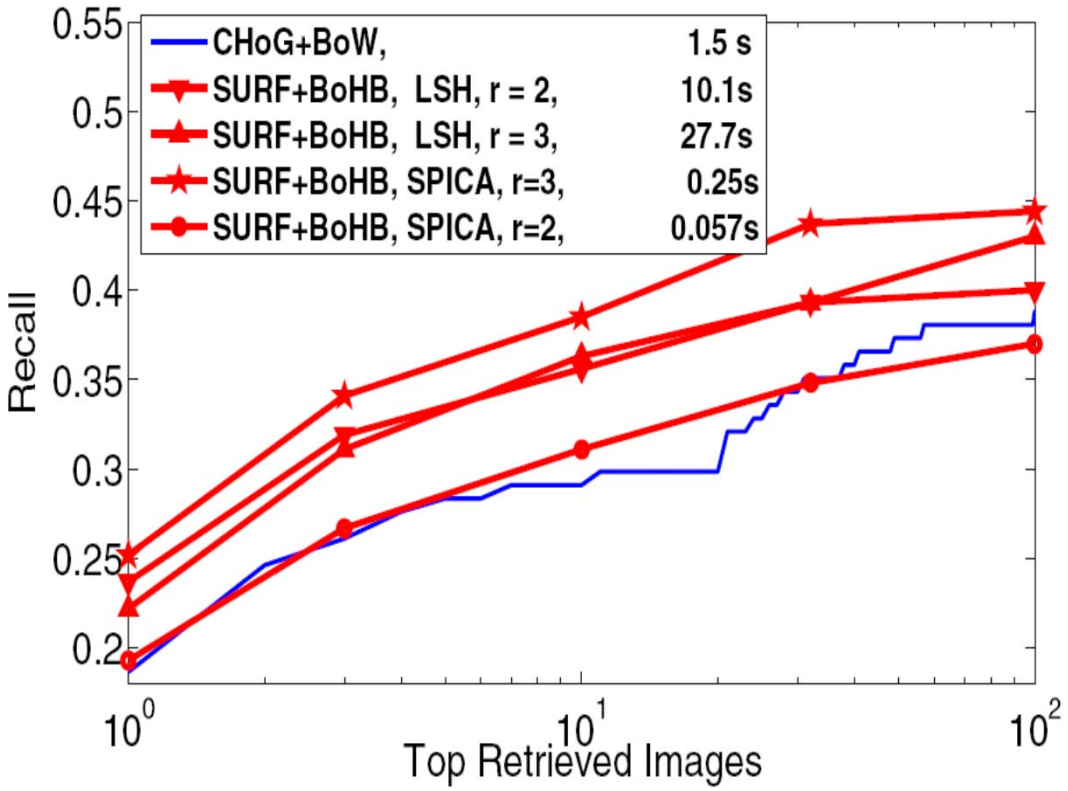
He, Feng, Liu, Cheng, Lin, Chung, Chang. *Mobile Product Search with Bag of Hash Bits and Boundary Reranking*, CVPR 2012.

- Data set 1: 400K products crawled from ebay, zappos;
  - more than 100 diverse categories
  - 205 queries, each has one GT in database
- Data set 2: 300K product images crawled from amazon
  - 20 categories, mainly shoes, home supplies
  - 135 queries, each has one GT in database
- On average, 100-200 local features(LF) for each image

Example queries and groundtruths for data set 1

Performance

- CHoG approach [V. Chandrasekhar et al 2009]:
  Compress local features with CHoG on mobile + BoW with VocTree (1M codewords) on server

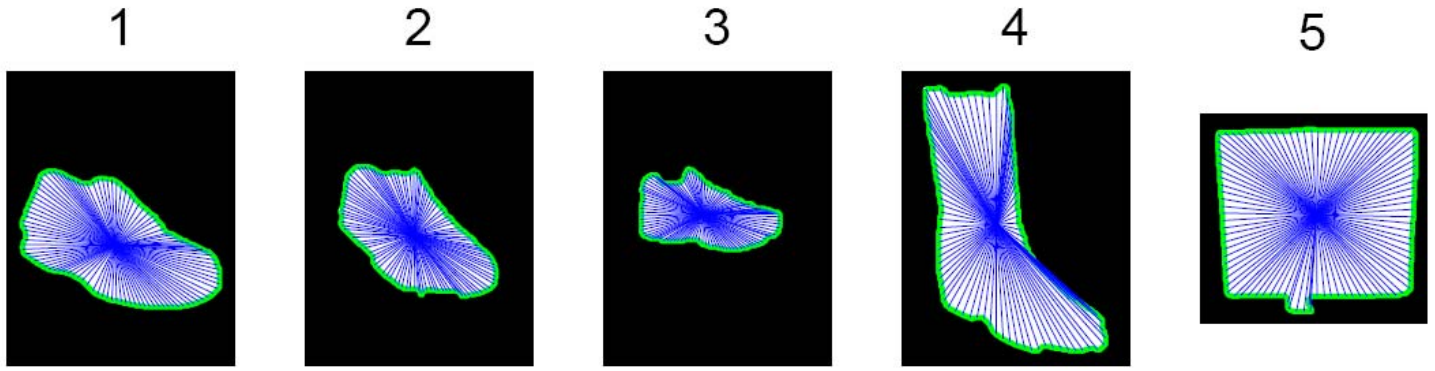


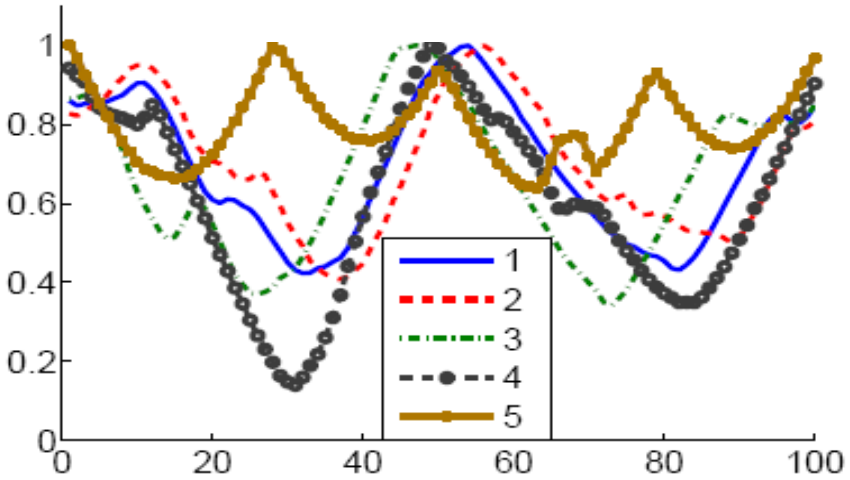
30% higher recall and
6X-30X search speedup

Rerank Results with Boundary Features

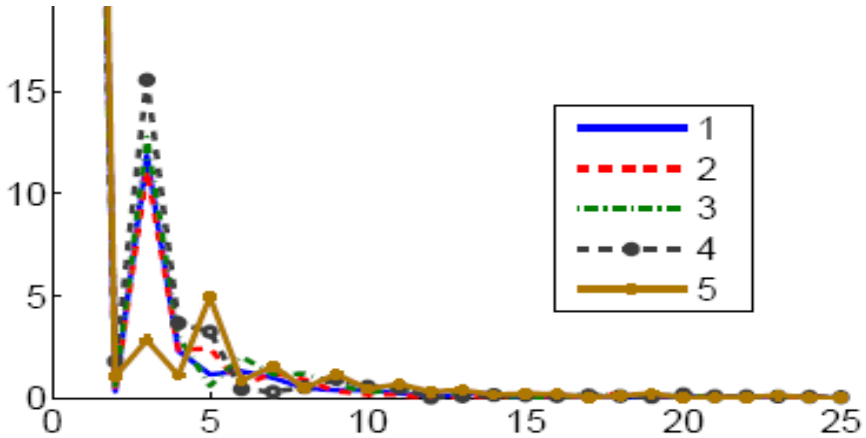
- Use automatic salient object segmentation for every image in DB [Cheng et al, CVPR 2011]
- Compute boundary features:
  normalized central distance, Fourier magnitude
- Invariance: translation, scaling, rotation

Boundary Feature – Central Distance





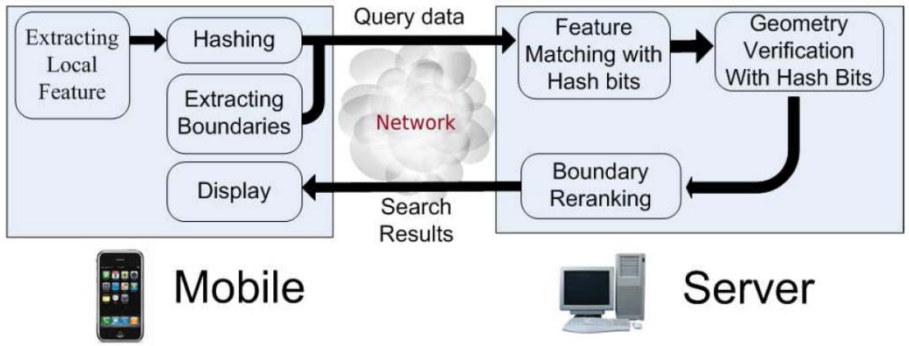
Distance to Center D(n)



FFT: F(n)

Reranking with boundary feature

Columbia MPS System:
Bags of Hash Bits and Boundary features

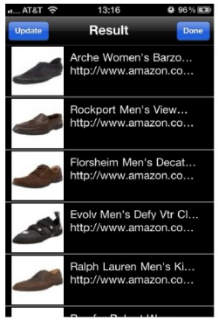


Server:

- **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

He, Feng, Liu, Cheng, Lin, Chung, Chang. *Mobile Product Search with Bag of Hash Bits and Boundary Reranking*, CVPR 2012.

Multi-View Challenge

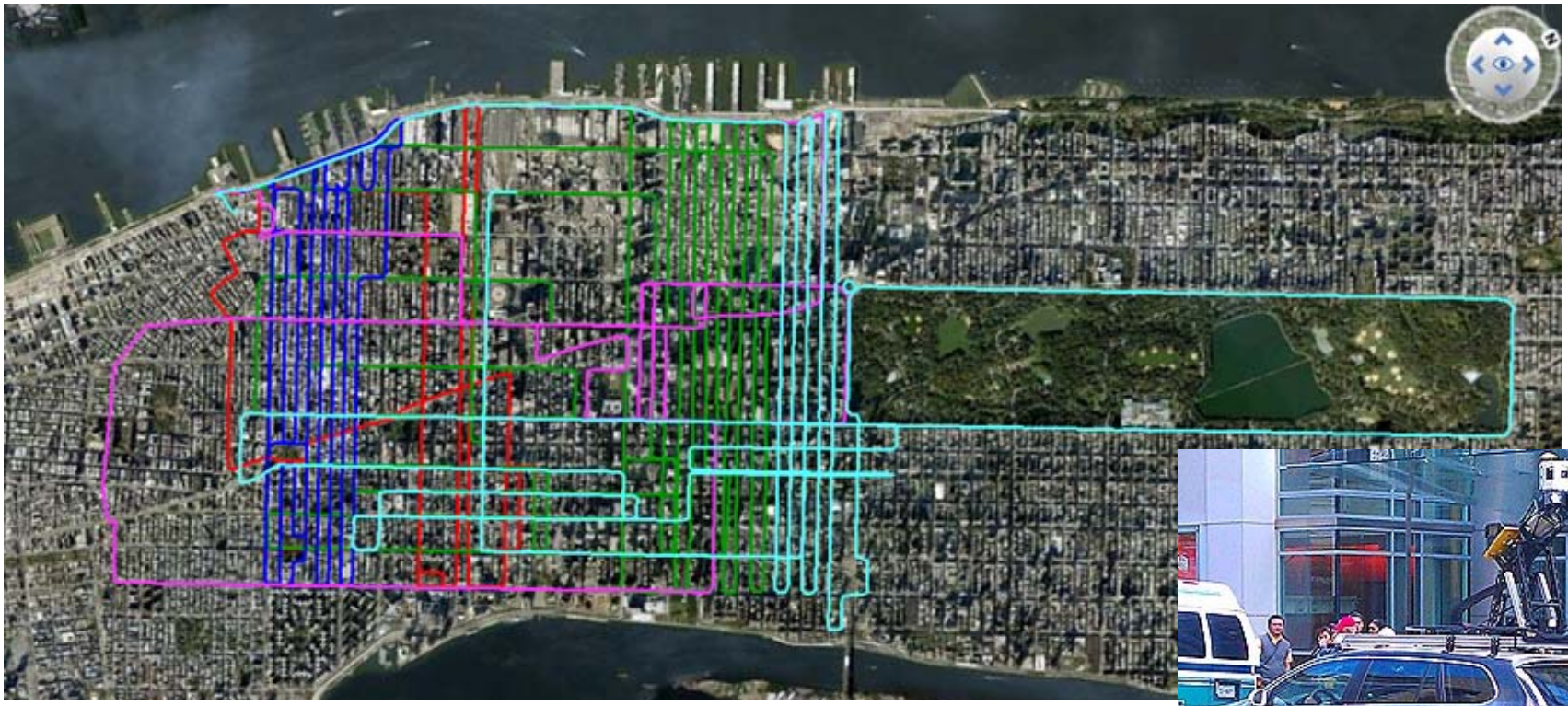
How to guide the user to take a successful mobile query?

- Which view will be the best query?
  - For example, in mobile location search:

  - Or in mobile product search:

Mobile Location Search

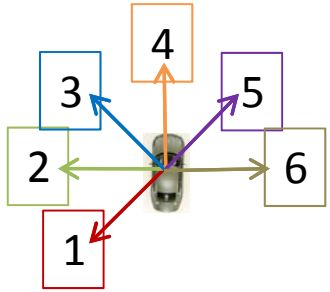
- 300,000 images of 50,000 locations in Manhattan
- Collected by the NAVTEQ street view imaging system



Geographical distribution

NAVTEQ 0.3M NYC Data Set

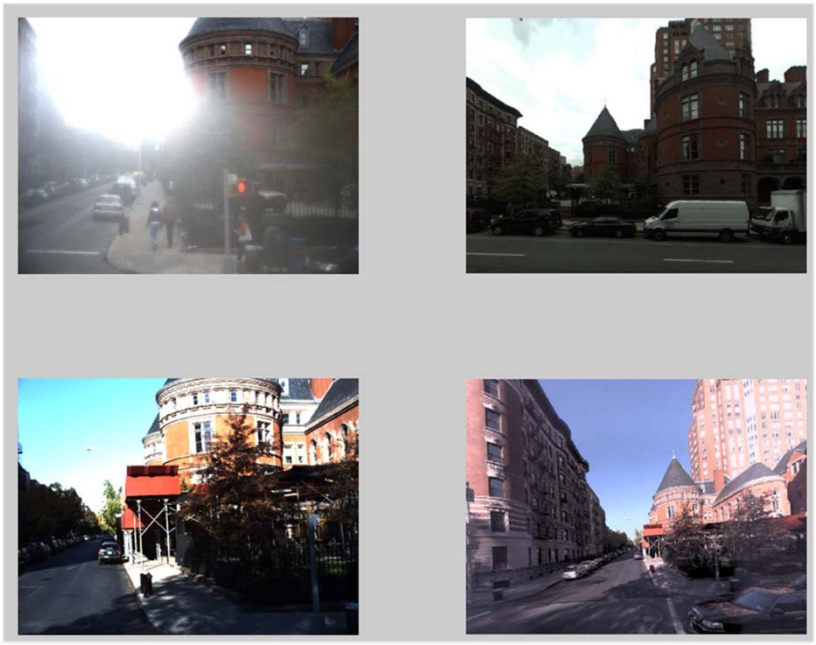
- Location Sampling
  - Locations are imaged at a four-meter interval on average
  - Six camera views for each location separated by 45º
- Visual Data Organization
  - Six views (images)
  - Also provide panorama (used for visualization in this work)

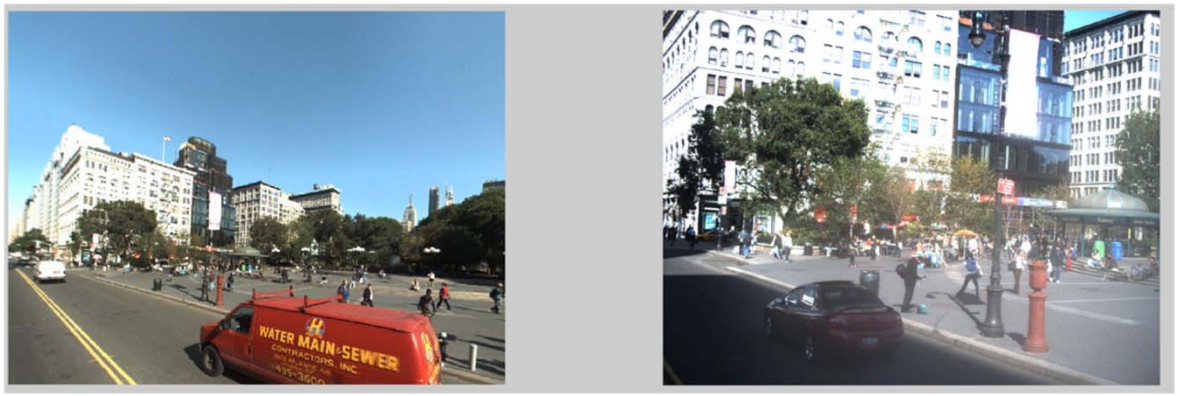


Images from Navteq

More Challenges on Mobile Clients

- Image quality variations
  - Exposure
  - Shadow
  - Distance
  - Obstruction
  - Blur
  - Weather
  - Day/Night





Navteq NYC Data

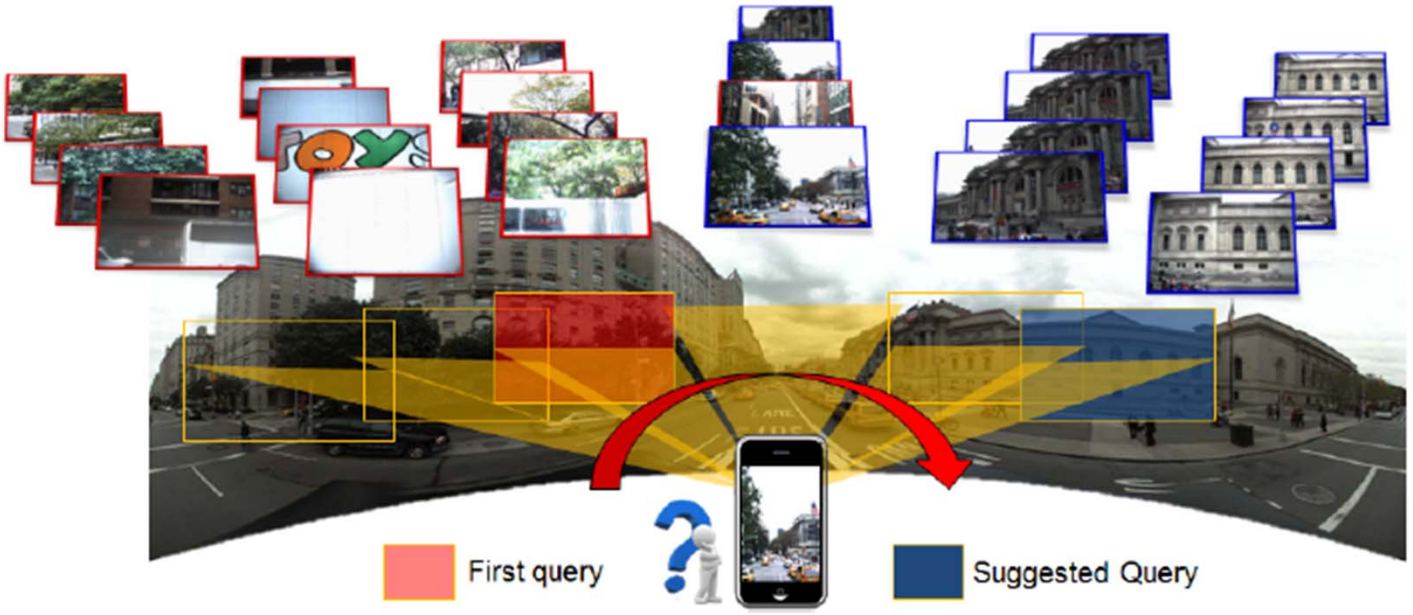
Not every view is equally good for search

- Recognition accuracy far from perfect
  - Less than 50% visual location searches successful initial tests *[Columbia Visual Location Search, '11]*

Subsample 200 locations to "# of searchable views" using cropped Google street view

Solution: Active Query Sensing

- Guide User to a More Successful Search Angle
  - Active Query Sensing [Yu, Ji, Zhang, and Chang, ACMMM, 2011]

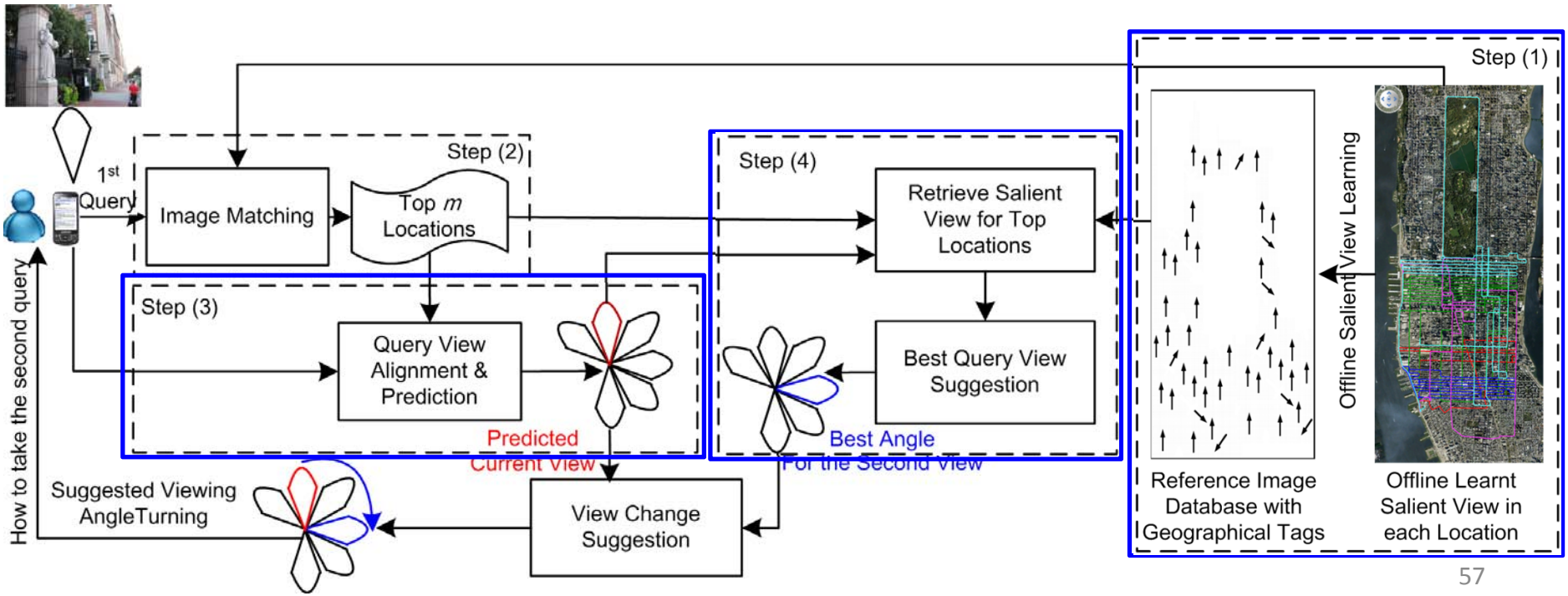


Video demo

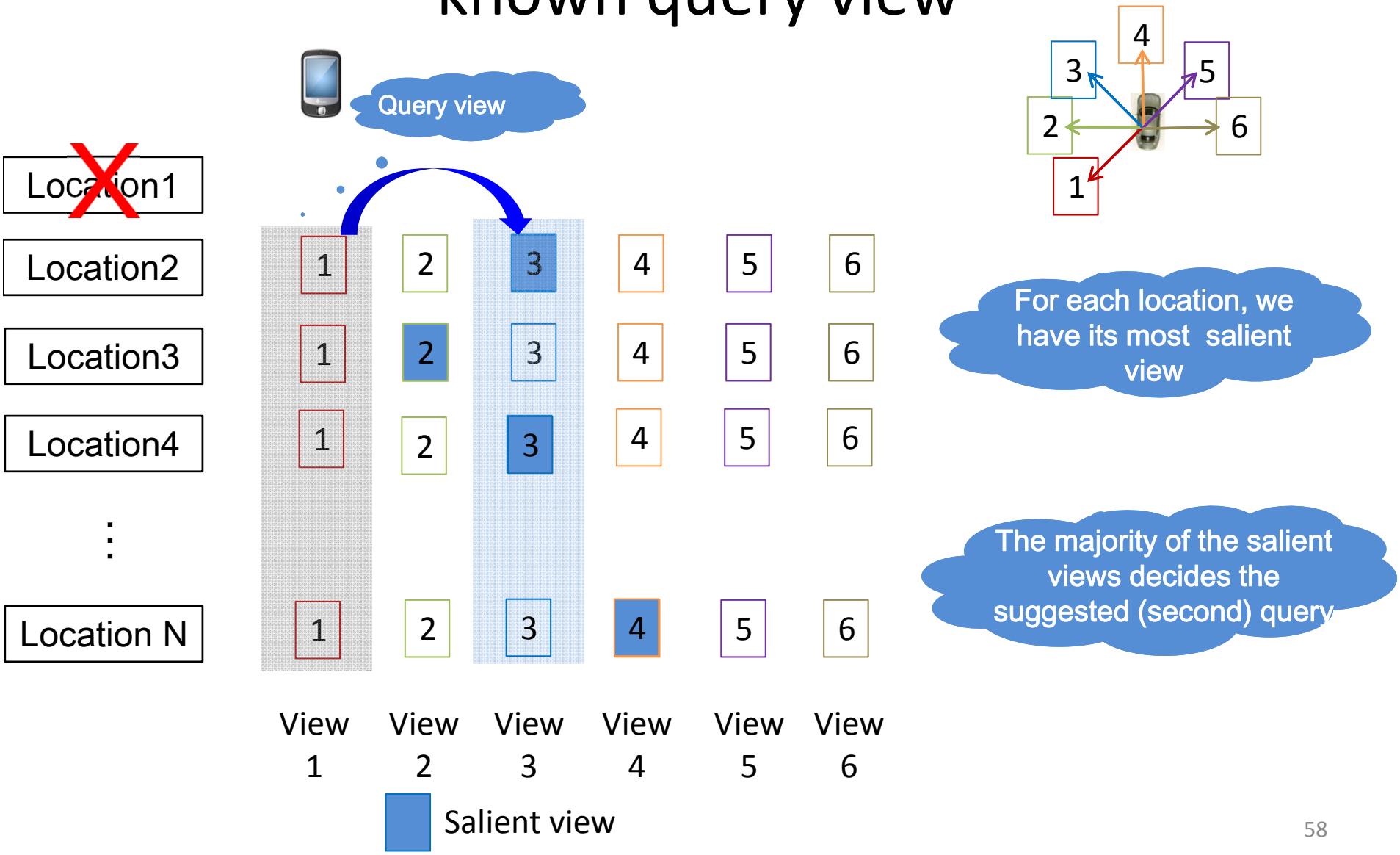
Mobile App Demo

Active Query Sensing System

- Offline
  - Salient view learning for each reference location
- Online
  - Viewing angle prediction of the first query
  - Suggest new views by majority voting



Active Query Sensing (case 1) known query view



What if query view is unknown?

- Step I: Predict the view angle of the first query

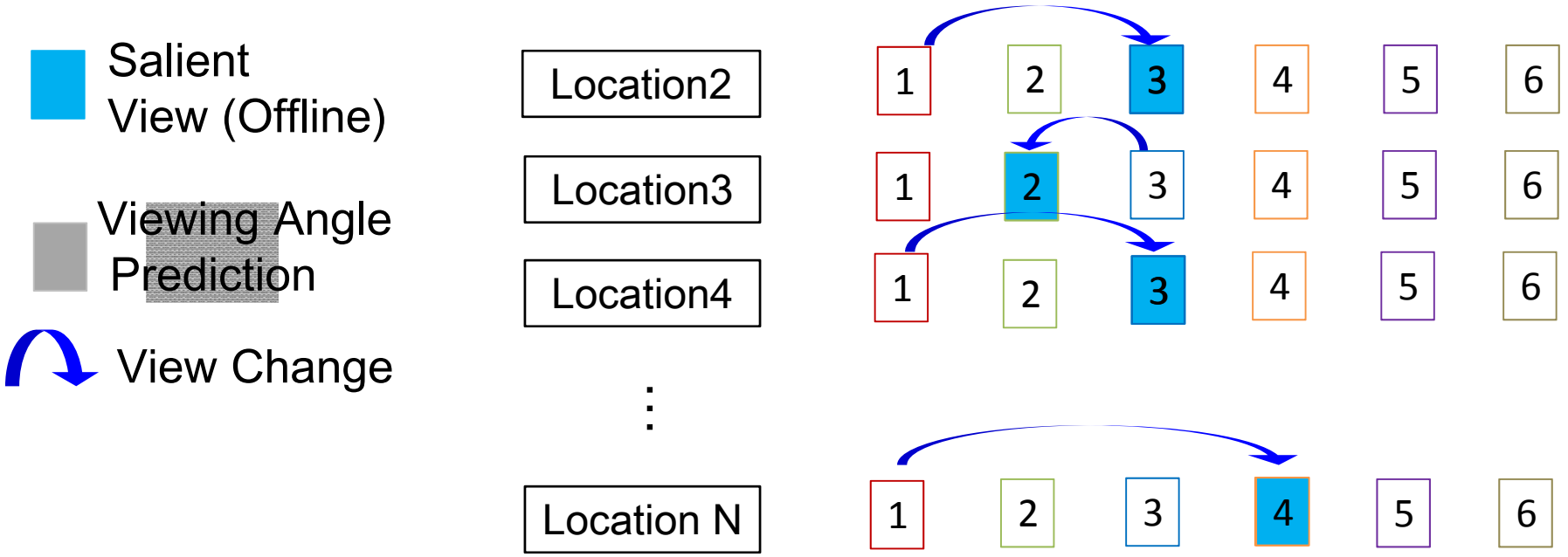
Offline Training: Train view prediction classifiers offline

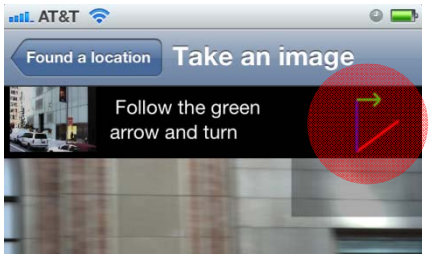
Online Prediction: View alignment based on the image matching

Our solution is to combine them both

Active Query Sensing (case 2) unknown query view

- Step II: Majority voting in terms of view change

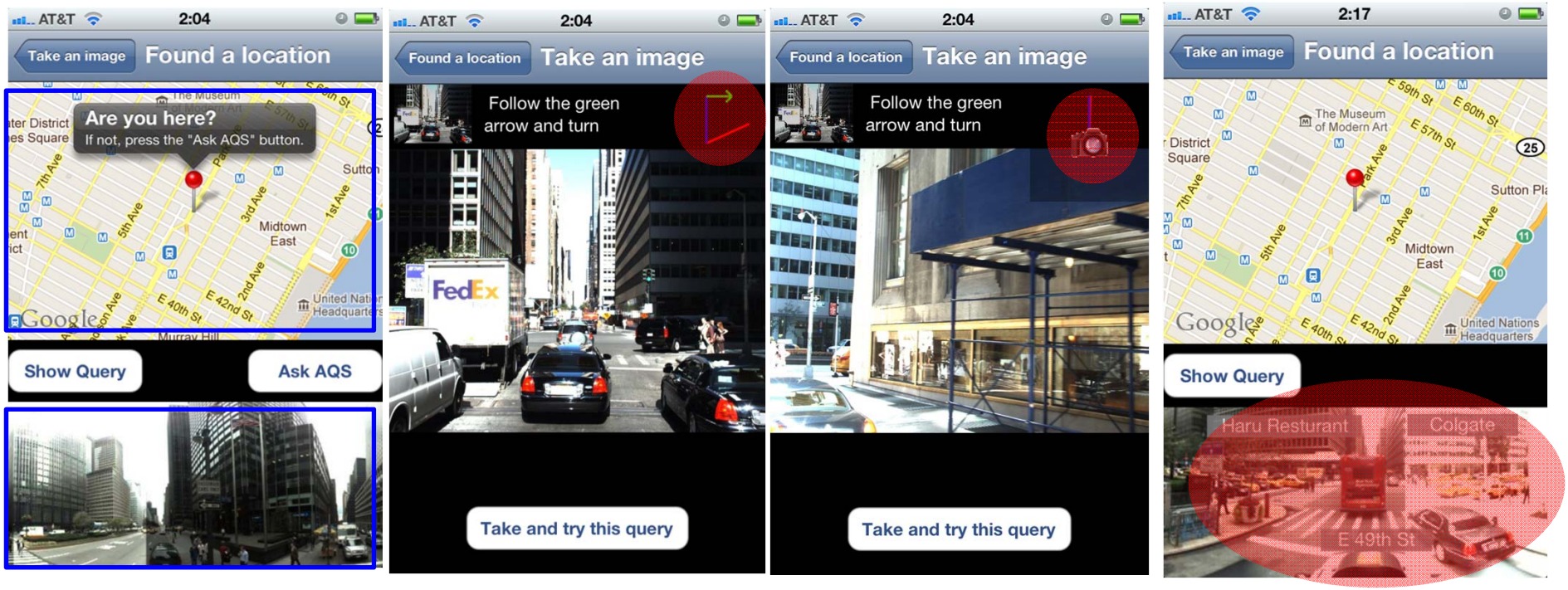




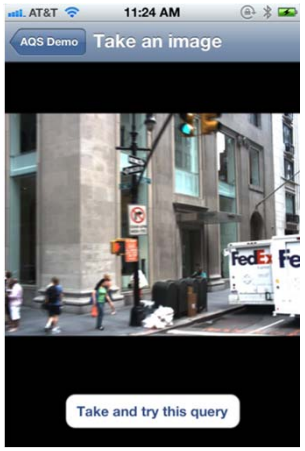
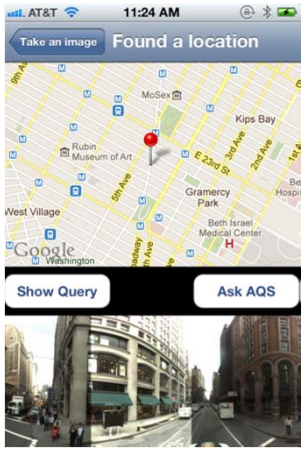
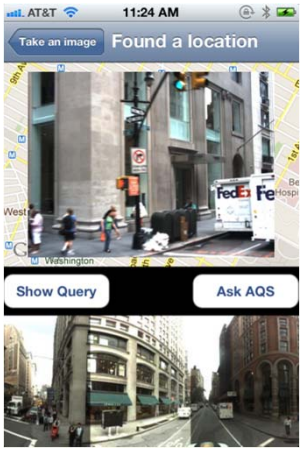
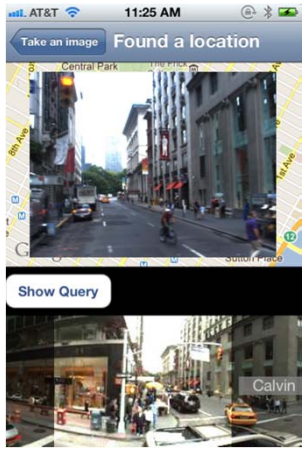
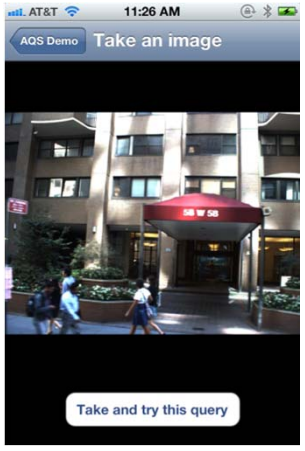
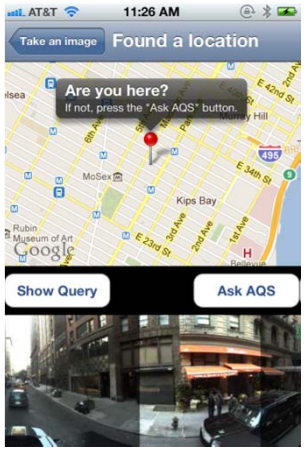
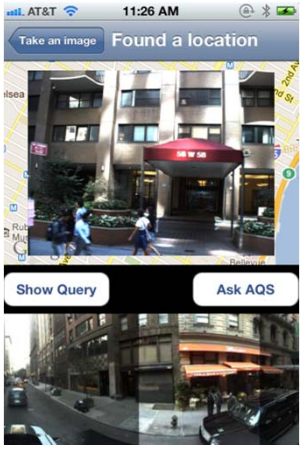
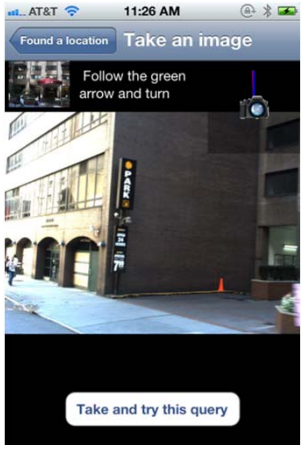
Turn 90 degrees to the right

User Interface

- Help user determine whether the first query is correct
  - Panorama
  - Geographical map context
- Guide the user to take the second query
  - Compass, camera icon
- Show point of interest



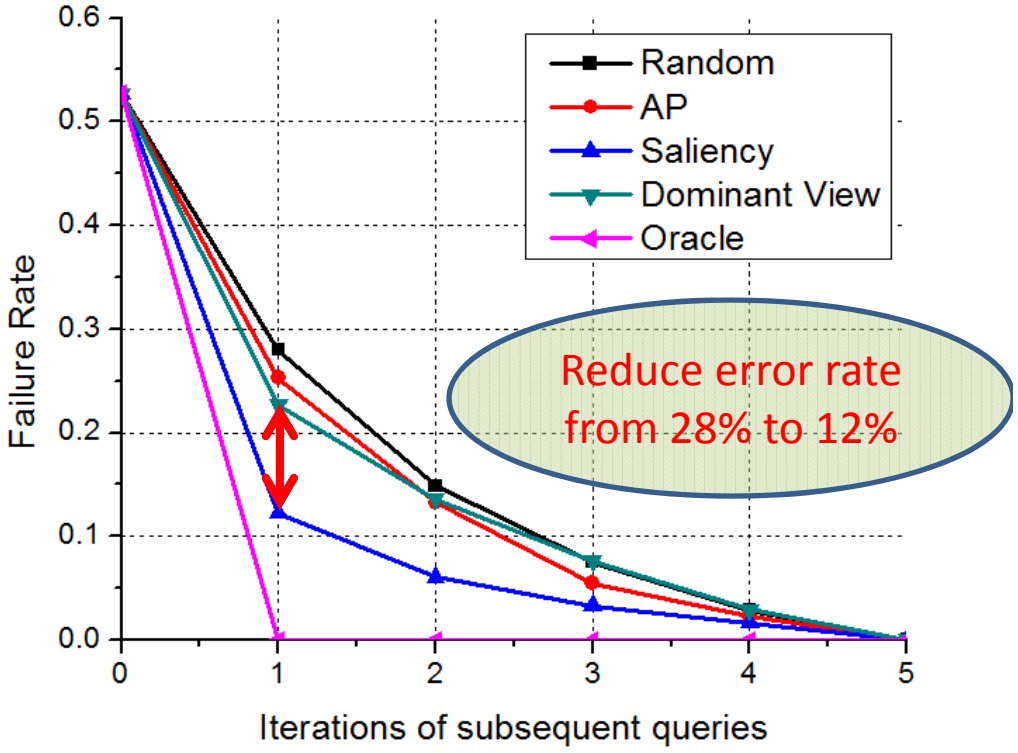
AQS Examples

First Query	First Result	Ask AQS	Taking Second Query		Second Result
					
					

Performance Improvement

- The AQS system helps user select the best angle for searching location
- It reduces failure rate by more than half

- Overall Performance



Failure rates over successive query iterations.

Conclusions

- Bags of Hash Bits (BoHB) for fast mobile product search
  - Simultaneously address power, bandwidth, and large database issues
- Promising research in hashing
- Active Query Sensing for interactive search
  - New paradigm for interactive mobile visual search
  - Guide user in the loop

References

- (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
- (SPICA Hash)
  J. He, R. Radhakrishnan, S.-F. Chang, C. Bauer, Compact Hashing with Joint Optimization of Search Accuracy and Time, CVPR 2011 (oral).
- (Supervised Kernel Hash)
  W. Liu, J. Wang, R. Ji, Y. Jiang, and S.-F. Chang, Supervised Hashing with Kernels, CVPR 2012 (oral)
- (Spherical Hashing)
  Jae-Pil Heo, YoungWoon Lee, Junfeng He, Shih-Fu Chang, Sung-eui Yoon. Spherical Hashing. CVPR 2012.
- (Hyperplane Hashing)
  Wei Liu, Jun Wang, Yadong Mu, Sanjiv Kumar, Shih-Fu Chang. Compact Hyperplane Hashing with Bilinear Functions. ICML 2012
- (Active Mobile Location Search)
  F. X. Yu, R. Ji, T. Zhang, S.-F. Chang. Active Query Sensing for Mobile Location Search, ACM Multimedia 2011.