Large Scale Mobile Visual Search

Shih-Fu Chang
June 2012
The Explosive Growth of Visual Data

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

- 70,000 TB/year, 100 million hours
- 60 hours of video uploaded every minute
- 4 billion video views per day in 2012
Many research & commercial search engines

VideoGoogle

QBIC™

VisualSEEK

VideoQ

TinEye

SINGINGFISH

3D Model Search Engine

Videosurf

ALIPR™

Automatic Photo Tagging and Visual Image Search

Semantic Video Search Engine

MediaMill

Autonomy

Informedia

Sixth Sense

Snaptell

TalkMiner

Google

IQ Engines

VisionGo

kooaba

INGE

like.com

leavesnap

Mealsnap

Stanford Mobile Visual Search

Google Goggles
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

There are more than 10,000 apps on the App Store, and so far iPhone users have downloaded an incredible 500 million, in every category from games to business.

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
4. Visual search on server database
5. Send results back
Challenges for MVS

1. Take a picture
2. Image feature extraction
3. Send via mobile networks
4. Visual matching with database images
5. Send results back

But need fast response (< 1-2 seconds)

Limited power/memory/speed
Limited bandwidth
Gigantic Database
MVS calls for Distributed Optimization

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

Case Study (MVS, Girod et al, 2011)
Mobile Search System by Hashing

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
- Server 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
Example: keypoint detection

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

Sampling with step $\sigma^4 = 2$

Original image

$\sigma = 2^\frac{1}{4}$

Scale (first octave)

Scale (next octave)

Gaussian

Difference of Gaussian (DOG)

(Slide of K. Grauman)
Describe Appearance of Local Features

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
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
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

011001100100111100…
110110011001100110…
Locality-Sensitive Hashing

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

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

$x$ is an $\varepsilon$-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 ($\leq 50$) and a single table.
- Both time and storage efficient.
Hashing: Active Research Topic

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

- **Unsupervised Hashing**
  - LSH, PCAH, ITQ, KLSH, SH, AGH

- **Semi-Supervised Hashing**
  - SSH, WeaklySH

- **Supervised Hashing**
  - RBM, BRE, MLH, LDAH
Unsupervised hash

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

![Search accuracy](image)

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

**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 1 Million 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

\[ x_i, x_j \text{ are similar } \Rightarrow h(x_i) = h(x_j) \]

\[ x_i, x_j \text{ are dissimilar } \Rightarrow h(x_i) \neq h(x_j) \]

Preferred hashing
Use Code Inner Products to Match Supervised Labels

Liu, Wang, Ji, Jiang, Chang, CVPR2012

\[ H(x_i)H^T(x_j) \rightarrow S_{ij} \]
Learning Supervised Hash

Hashing: \( \mathbf{x}_i \mapsto H_l \)

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

\[
\min_{H_l \in \{1, -1\}^{l \times r}} Q = \left\| \frac{1}{r} H_l H_l^\top - S \right\|_F^2,
\]

- Easy to optimize and extend to kernels
- Sequential learning

Design hash codes to match supervised information
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

(a) Precision @ Hamming radius 2 vs. # bits

Supervised kernel hashing
Spherical Hashing

- 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

---

Heo, Lee, He, Chang, Yoon, CVPR 2012
Spherical Hashing Performance

- 1 Million Images: GIST 384-D features

![Graph showing performance comparison of different hashing methods. The x-axis represents binary code length (number of bits), and the y-axis represents 100-NN mAP. The graph compares various methods including Ours-SHD, Ours-HD, RMMH-L2, GSPICA-RBF, PCA-ITQ, LSH-ZC, LSBC, LSH, and SpecH.]
Point-to-Point Search vs. Point-to-Hyperplane Search
Hashing Principle: Point-to-Hyperplane Angle

\[ D(x, P_w) = \frac{|w^T x|}{||w||} \]

\[ \min D \Rightarrow \min \alpha \]

The ideal neighbors \( \perp w \)
Bilinear Hashing

Bilinear-Hyperplane Hash (BH-Hash)

\[ h^B(z) = \text{sgn}(u^T zz^T v), \text{ i.i.d. } u, v \sim \mathcal{N}(0, I_{d \times d}). \]

- **bilinear** hash bit: +1 for \( \parallel \) points, -1 for \( \perp \) points

query normal \( w \) or database point \( x \)

2 random projection vectors
A Single Bit of Bilinear Hash

\[ h^B(w) = \text{sgn}(u^Tww^Tv) \]
\[ = \text{sgn}(u^Tw)\text{sgn}(v^Tw) \]
\[ = 1 \cdot 1 = 1 \]

\[ h^B(x_1) = \text{sgn}(u^Tx_1)\text{sgn}(v^Tx_1) \]
\[ = 1 \cdot 1 = 1 \]

\[ h^B(x_2) = \text{sgn}(u^Tx_2)\text{sgn}(v^Tx_2) \]
\[ = -1 \cdot 1 = -1 \]
Theoretical Collision Probability

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

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

A concrete measure of difficulty of search in a dataset?
Relative Contrast

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

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

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

• High Relative Contrast \(\rightarrow\) easier search
• If \(C_r \rightarrow 1\), search not meaningful

He, Kumar, Chang, ICML 2012
Estimation of Relative Contrast

- With CLT, and binomial approximation

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

\( \phi \) - standard Gaussian cdf

\( \sigma' \) – 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 $\mathcal{U}[0,1]$

**Graphs:**
- Higher dimensionality $\rightarrow$ bad
- Sparser vectors $\rightarrow$ good
Synthetic Data

- Data sampled randomly from $\mathcal{U}[0,1]$
## Predict Hashing Performance of Real-World Data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dimensionality ($d$)</th>
<th>Sparsity ($s$)</th>
<th>Relative Contrast ($C_r$) for $p = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT</td>
<td>128</td>
<td>0.89</td>
<td>4.78</td>
</tr>
<tr>
<td>Gist</td>
<td>384</td>
<td>1.00</td>
<td>1.83</td>
</tr>
<tr>
<td>Color Hist</td>
<td>1382</td>
<td>0.027</td>
<td>3.19</td>
</tr>
<tr>
<td>Imagenet BoW</td>
<td>10000</td>
<td>0.024</td>
<td>1.90</td>
</tr>
</tbody>
</table>

**Diagram:**

- **28 bits**
  - sift
  - gist
  - color
  - ImageNet

- **16 bits**
  - sift
  - gist
  - color
  - ImageNet
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

Optimal hash bit pool (e.g., 80 bits, PCA Hash or SPICA hash)

1 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1

Random subset

Table 1

Random subset

Table 2

... Random subset

Table 11

Random subset

Table 12

32 bits

Union Results
Data Sets


- **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
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
- Server Search: ~0.4s
- Download/display: 1-2s

video demo

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
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 45°

• Visual Data Organization
  – Six views (images)
  – Also provide panorama (used for visualization in this work)
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]
Solution: Active Query Sensing

- Guide User to a More Successful Search Angle
  - Active Query Sensing [Yu, Ji, Zhang, and Chang, ACM-MM, 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

For each location, we have its most salient view.

The majority of the salient views decides the suggested (second) query.

Salient 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
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

<table>
<thead>
<tr>
<th>First Query</th>
<th>First Result</th>
<th>Ask AQS</th>
<th>Taking Second Query</th>
<th>Second Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
<td><img src="image5" alt="Image" /></td>
</tr>
<tr>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
</tr>
</tbody>
</table>
Performance Improvement

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

• Overall Performance

Reduce error rate from 28% to 12%
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)

• (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)