Announcements

- Coordinating with other presenters
- Presentation length: ~20 minutes
- HW1 questions?

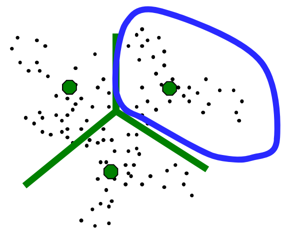
- Today:
  – Wrap-up on instance recognition
  – Large-scale visual search
  – Paper discussion

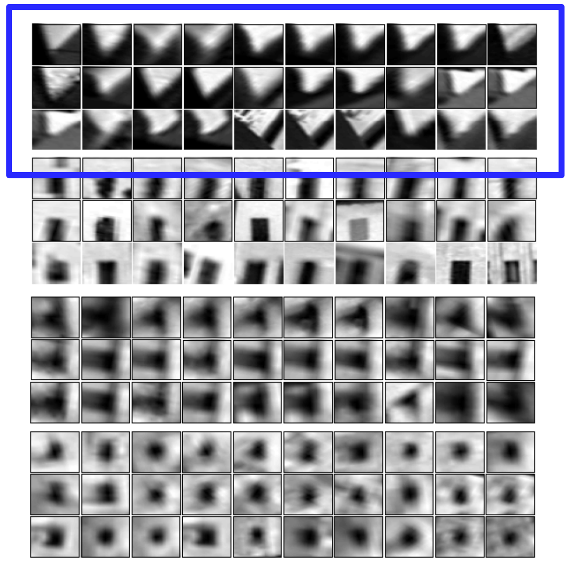
Wrap-up from last time: instance recognition

- Visual words
  - quantization, index, bags of words
- Spatial verification
  - affine; RANSAC, Hough
- Other text retrieval tools
  - tf-idf, query expansion
- Example applications

Visual words

- Example: each group of patches belongs to the same visual word





Kristen Grauman

Figure from Sivic & Zisserman, ICCV 2003

Inverted file index and bags of words similarity

New query image

Word #	Image #
1	3
2	
…	
7	1, 2
8	3
9	
10	
…	
91	2
⋮	⋮

…

1. Extract words in query
2. Inverted file index to find relevant frames
3. Compare word counts

Kristen Grauman

Visual words/bags of words

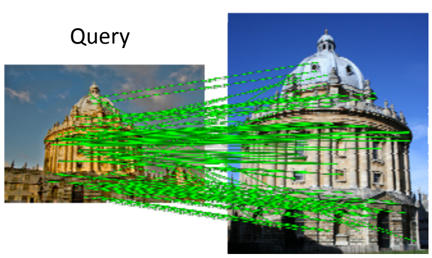
+ flexible to geometry / deformations / viewpoint
+ compact summary of image content
+ provides vector representation for sets
+ very good results in practice

- background and foreground mixed when bag covers whole image
- optimal vocabulary formation remains unclear
- basic model ignores geometry – must verify afterwards, or encode via features

Kristen Grauman

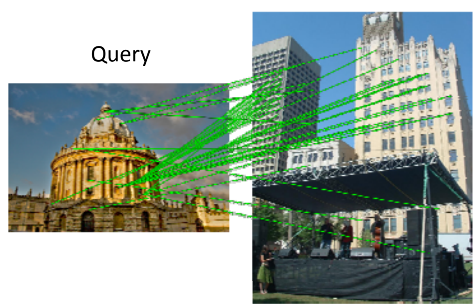
Spatial Verification

Query



DB image with high BoW similarity

Query



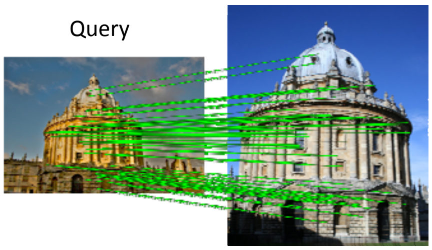
DB image with high BoW similarity

Both image pairs have many visual words in common.

Slide credit: Ondrej Chum

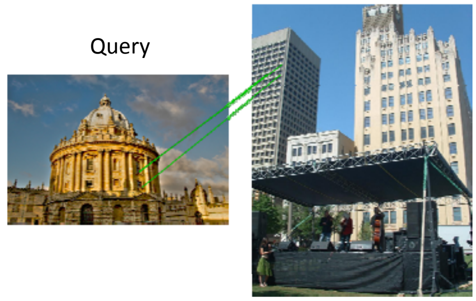
Spatial Verification

Query



DB image with high BoW similarity

Query



DB image with high BoW similarity

Only some of the matches are mutually consistent

Slide credit: Ondrej Chum

Spatial Verification: two basic strategies

- RANSAC
- Generalized Hough Transform

Kristen Grauman

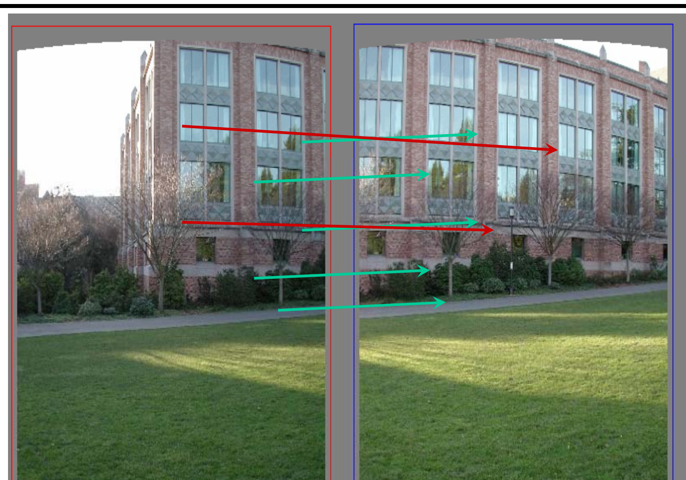
RANSAC: General form

- RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate
2. Compute model from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute estimate of model on all of the inliers

- Keep the model with the largest number of inliers

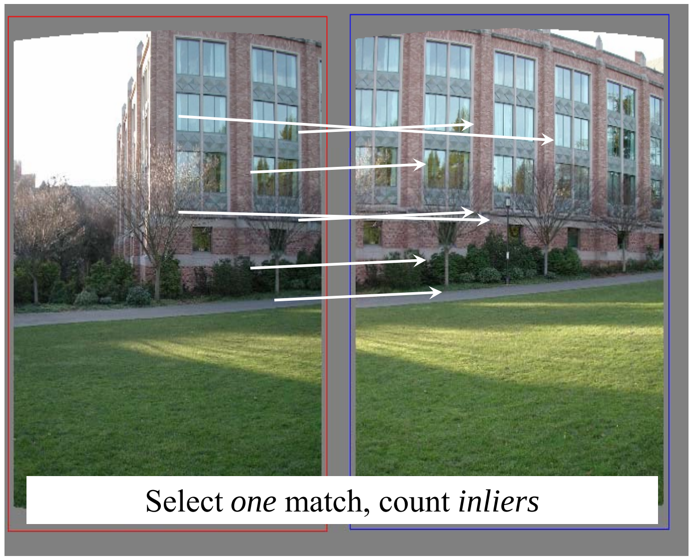
RANSAC example: Translation



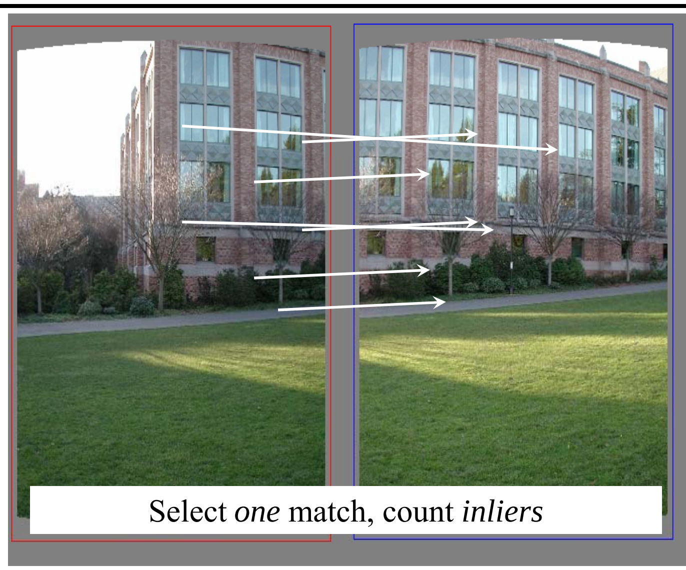
Putative matches

Source: Rick Szeliski

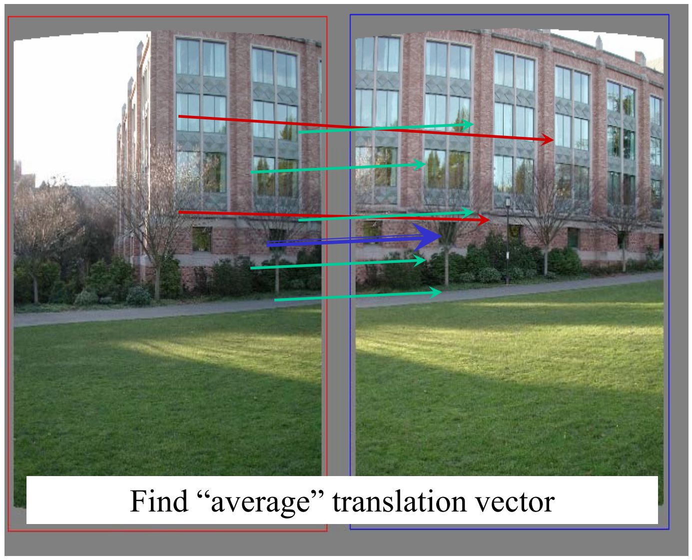
RANSAC example: Translation



RANSAC example: Translation



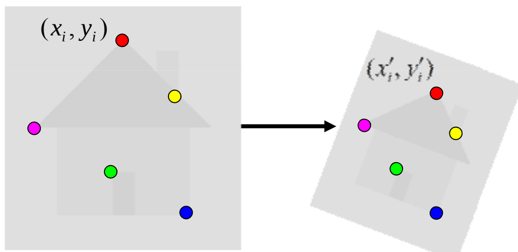
RANSAC example: Translation



RANSAC verification

For matching specific scenes/objects, common to use an **affine transformation** for spatial verification

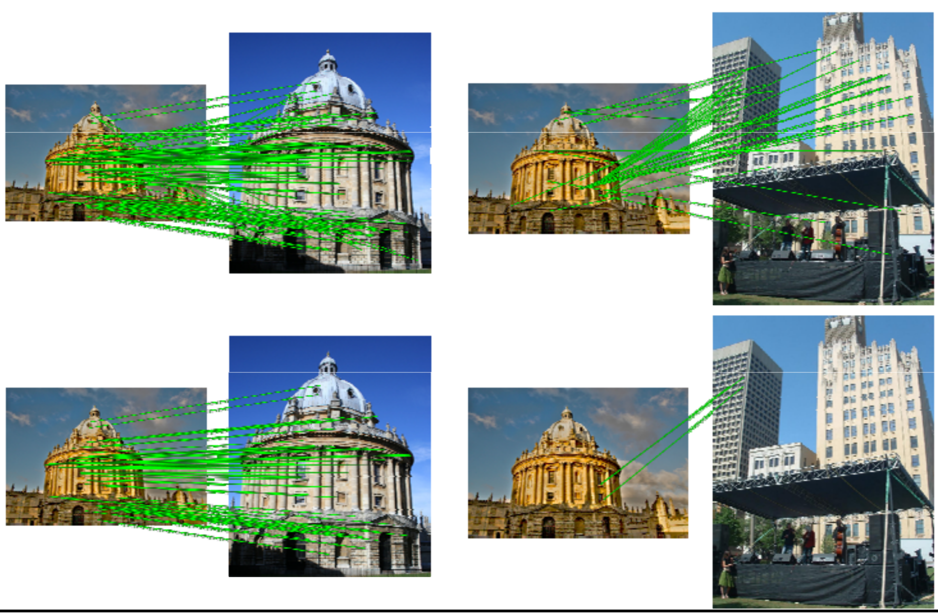
Fitting an affine transformation



Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras.

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

RANSAC verification



Spatial Verification: two basic strategies

- RANSAC
  - Typically sort by BoW similarity as initial filter
  - Verify by checking support (inliers) for possible affine transformations
    - e.g., "success" if find an affine transformation with > N inlier correspondences
- Generalized Hough Transform
  - Let each matched feature cast a vote on location, scale, orientation of the model object
  - Verify parameters with enough votes

Kristen Grauman

Spatial Verification: two basic strategies

- RANSAC
  - Typically sort by BoW similarity as initial filter
  - Verify by checking support (inliers) for possible affine transformations
    - e.g., "success" if find an affine transformation with > N inlier correspondences
- Generalized Hough Transform
  - Let each matched feature cast a vote on location, scale, orientation of the model object
  - Verify parameters with enough votes

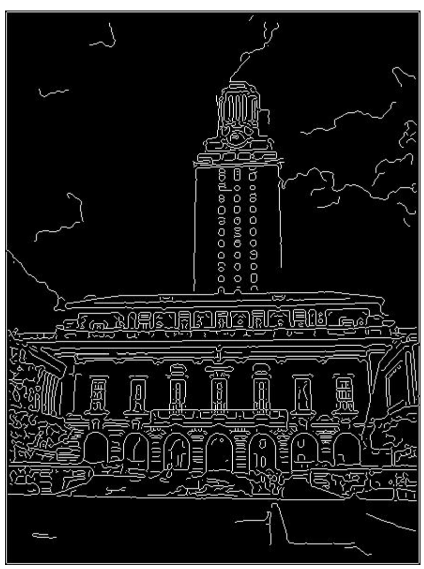
Kristen Grauman

Voting

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features *vote for all models that are compatible with it*.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, *but* typically their votes should be inconsistent with the majority of "good" features.

Kristen Grauman

Difficulty of line fitting



Kristen Grauman

Hough Transform for line fitting

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?

- **Hough Transform** is a voting technique that can be used to answer all of these questions.

  Main idea:
  1. Record vote for each possible line on which each edge point lies.
  2. Look for lines that get many votes.

Kristen Grauman

Finding lines in an image: Hough space

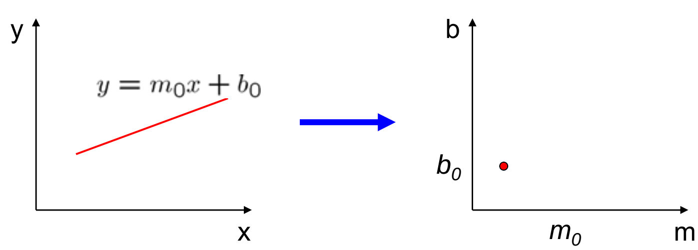


image space | Hough (parameter) space

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

Slide credit: Steve Seitz

Finding lines in an image: Hough space

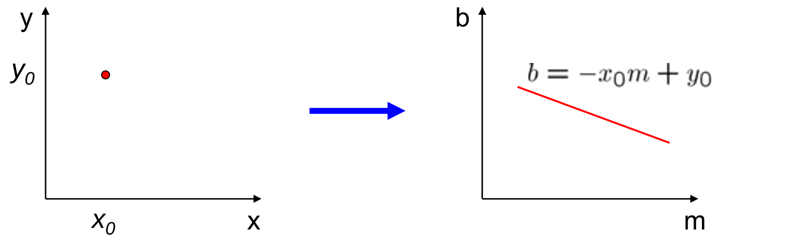


image space

Hough (parameter) space

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - this is a line in Hough space

Slide credit: Steve Seitz

Finding lines in an image: Hough space

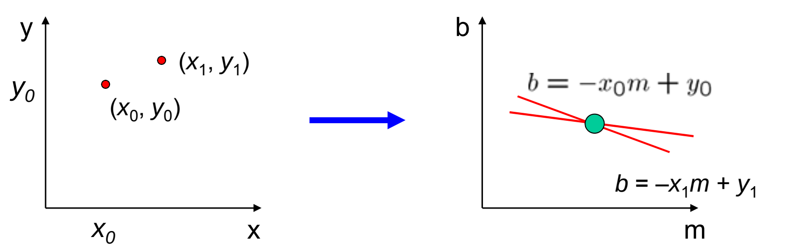


image space

Hough (parameter) space

What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1)?

- It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

Finding lines in an image: Hough algorithm

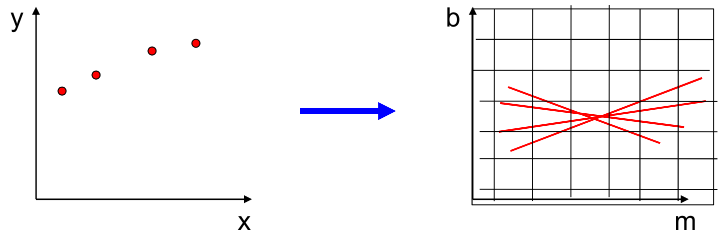


image space

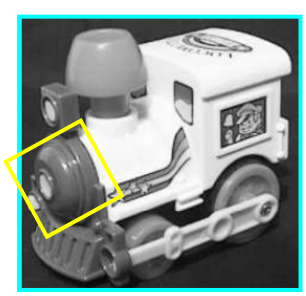
Hough (parameter) space

How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?

- Let each edge point in image space *vote* for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Voting: Generalized Hough Transform

- If we use scale, rotation, and translation invariant local features, then **each feature match** gives an alignment hypothesis (for scale, translation, and orientation of model in image).



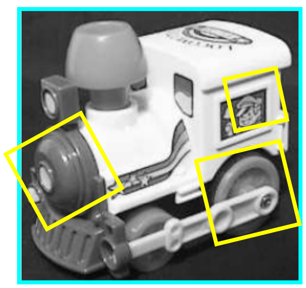
Model

Novel image

Slide credit: Lana Lazebnik

Voting: Generalized Hough Transform

- A hypothesis generated by a single match may be unreliable,
- So let each match **vote** for a hypothesis in Hough space



Model

Novel image

Gen Hough Transform details (Lowe's system)

- **Training phase:** For each model feature, record 2D location, scale, and orientation of model (relative to normalized feature frame)
- **Test phase:** Let each match btwn a test SIFT feature and a model feature vote in a 4D Hough space
  - Use broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times image size for location
  - Vote for two closest bins in each dimension
- Find all bins with at least three votes and perform geometric verification
  - Estimate least squares *affine* transformation
  - Search for additional features that agree with the alignment

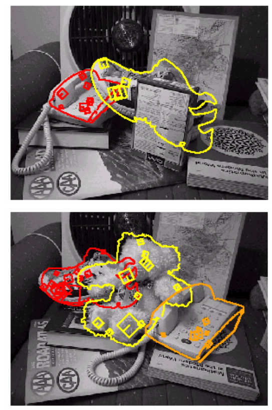
David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Slide credit: Lana Lazebnik

Example result

Background subtract
for model boundaries

Objects recognized,



Recognition in
spite of occlusion

[Lowe]

Difficulties of voting

- Noise/clutter can lead to as many votes as true target
- Bin size for the accumulator array must be chosen carefully

- In practice, good idea to make broad bins and spread votes to nearby bins, since verification stage can prune bad vote peaks.

Gen Hough vs RANSAC

GHT

- Single correspondence -> vote for all consistent parameters
- Represents uncertainty in the model parameter space
- Linear complexity in number of correspondences and number of voting cells; beyond 4D vote space impractical
- Can handle high outlier ratio

RANSAC

- Minimal subset of correspondences to estimate model -> count inliers
- Represents uncertainty in image space
- Must search all data points to check for inliers each iteration
- Scales better to high-d parameter spaces

Kristen Grauman

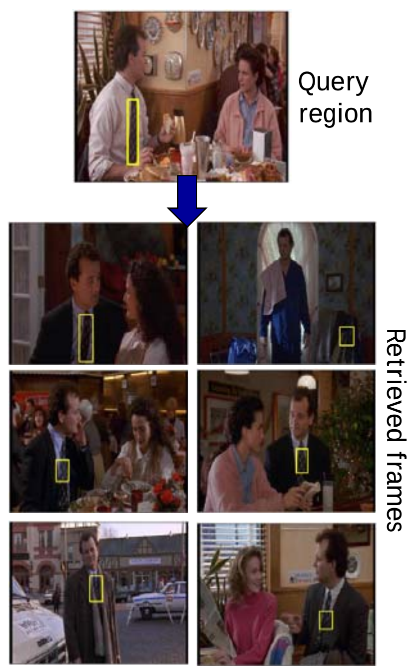
Visual Object Recognition Tutorial

Video Google System

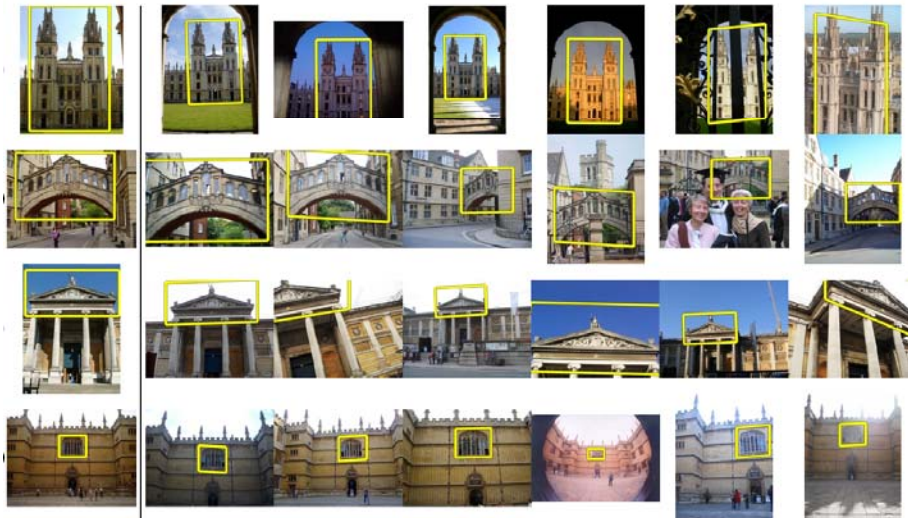
1. **Collect all words within query region**
2. **Inverted file index to find relevant frames**
3. **Compare word counts**
4. **Spatial verification**

Sivic & Zisserman, ICCV 2003

- **Demo online at :**
  http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html



Object retrieval with large vocabularies and fast spatial matching, Philbin et al., CVPR 2007



Query   Results from 5k Flickr images (demo available for 100k set)

[Philbin CVPR'07]

World-scale mining of objects and events from community photo collections, Quack et al., CIVR 2008

Auto-annotate by connecting to content on Wikipedia!

Visual Object Recognition Tutorial

Example Applications

Mobile tourist guide
- **Self-localization**
- **Object/building recognition**
- **Photo/video augmentation**

B. Leibe

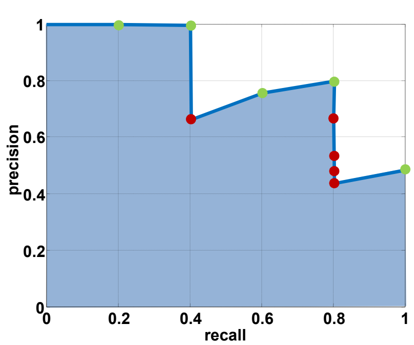
[Quack, Leibe, Van Gool, CIVR'08]

Scoring retrieval quality

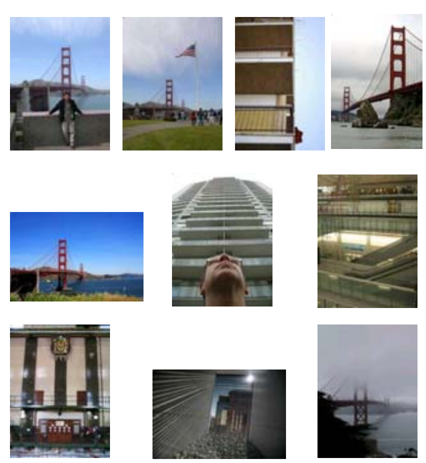
Query

Database size: 10 images
Relevant (total): 5 images

precision = #relevant / #returned
recall = #relevant / #total relevant

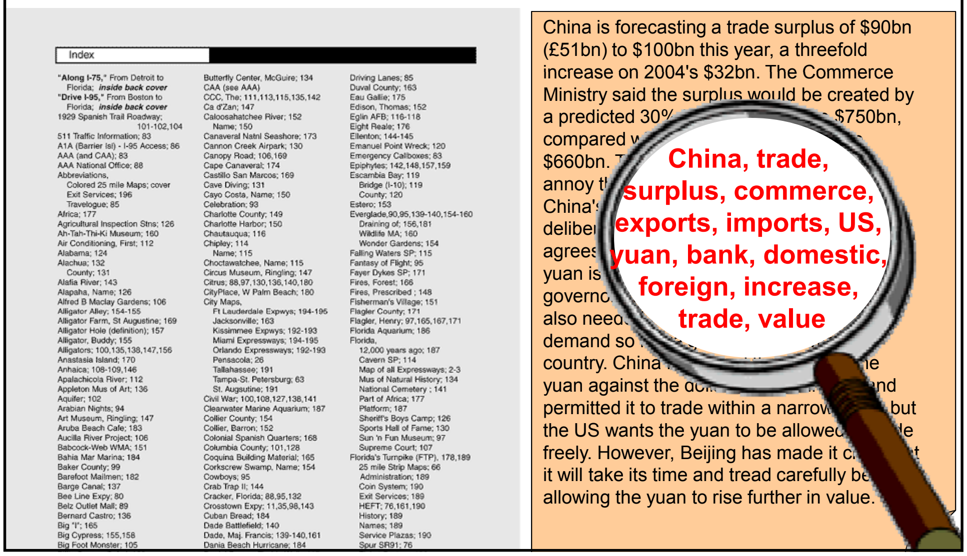


Results (ordered):



Slide credit: Ondrej Chum

What else can we borrow from text retrieval?



tf-idf weighting

- **T**erm **f**requency – **i**nverse **d**ocument **f**requency
- Describe frame by frequency of each word within it, downweight words that appear often in the database
- (Standard weighting for text retrieval)

Number of occurrences of word i in document d

Number of words in document d

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

Total number of documents in database

Number of documents word i occurs in, in whole database

Kristen Grauman

Query expansion

Query: ***golf green***

Results:

- How can the grass on the ***greens*** at a ***golf*** course be so perfect?
- For example, a skilled ***golf***er expects to reach the ***green*** on a par-four hole in **…**
- Manufactures and sells synthetic ***golf*** putting ***green****s* and mats.

Irrelevant result can cause a `topic drift':

- Volkswagen ***Golf***, 1999, ***Green***, 2000cc, petrol, manual, , hatchback, 94000miles, 2.0 GTi, 2 Registered Keepers, HPI Checked, Air-Conditioning, Front and Rear Parking Sensors, ABS, Alarm, Alloy

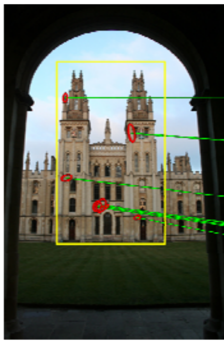
Slide credit: Ondrej Chum

Query Expansion

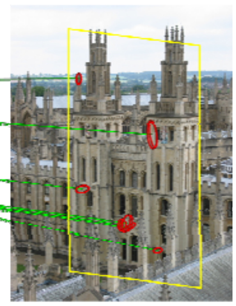
Chum, Philbin, Sivic, Isard, Zisserman: Total Recall..., ICCV 2007

Slide credit: Ondrej Chum

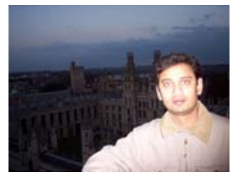
Query Expansion Step by Step



Query Image



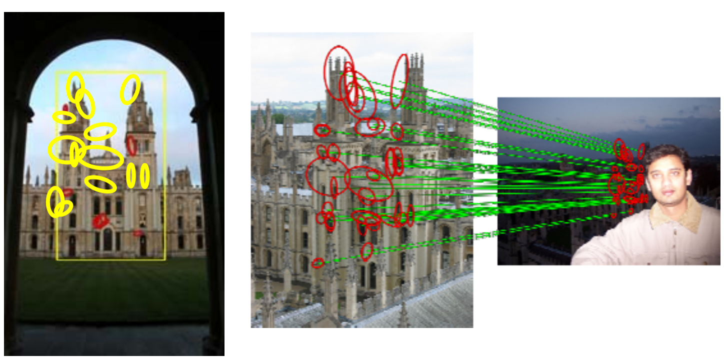
Retrieved image



Originally not retrieved

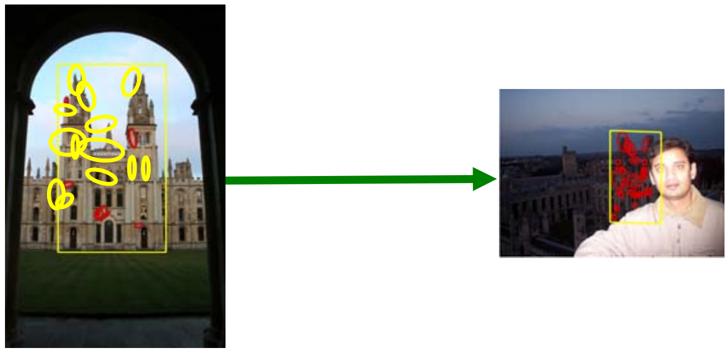
Slide credit: Ondrej Chum

Query Expansion Step by Step



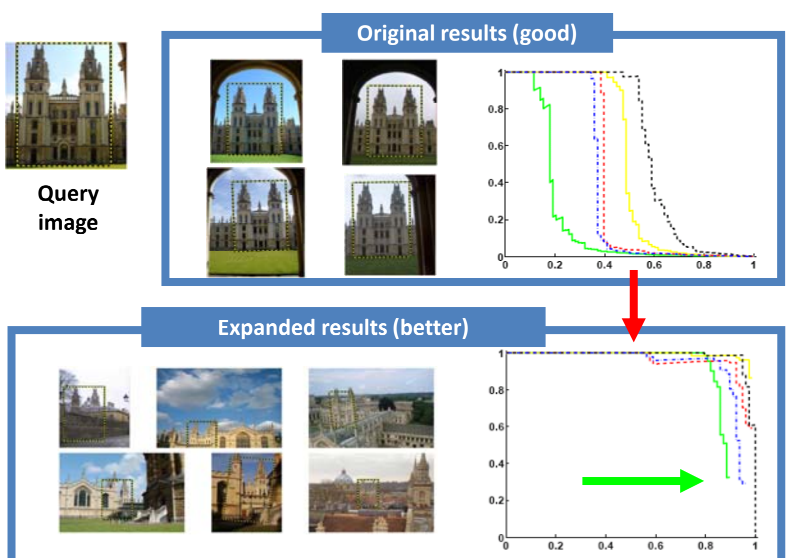
Slide credit: Ondrej Chum

Query Expansion Step by Step



Slide credit: Ondrej Chum

Query Expansion Results



Slide credit: Ondrej Chum

Visual Object Recognition Tutorial

Web Demo: Movie Poster Recognition

50’000 movie posters indexed

Query-by-image from mobile phone available in Switzer-land

http://www.kooaba.com/en/products_engine.html#

Recognition via feature matching+spatial verification

Pros:

- Effective when we are able to find reliable features within clutter
- Great results for matching specific instances

Cons:

- Scaling with number of models
- Spatial verification as post-processing – not seamless, expensive for large-scale problems
- Not suited for category recognition.

Kristen Grauman

Summary: instance recognition

- **Matching local invariant features**
  - Useful not only to provide matches for multi-view geometry, but also to find objects and scenes.
- **Bag of words** representation: quantize feature space to make discrete set of visual words
  - Summarize image by distribution of words
  - Index individual words
- **Inverted index**: pre-compute index to enable faster search at query time
- **Recognition of instances via alignment:** matching local features followed by spatial verification
  - Robust fitting : RANSAC, GHT

Kristen Grauman

Large-scale visual search

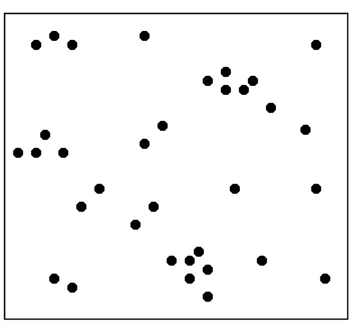
- How to efficiently find similar images/features?
  - Inverted file indexing schemes
  - Low-dimensional descriptors: can use standard efficient data structures for nearest neighbor search
  - High-dimensional descriptors: approximate nearest neighbor search methods more practical
- How to inject supervision into the search?
- How to summarize large collections?

Indexing local features: KD-trees

- Binary tree data structure to store set of points from a *k*-dimensional space.
- Partition points into axis-aligned boxes:
  - Divide the points in half by a hyperplane perpendicular to one of the axes.
  - Recursively construct KD trees for the two sets

[Friedman et al. 1977]

KD-Tree: Construction

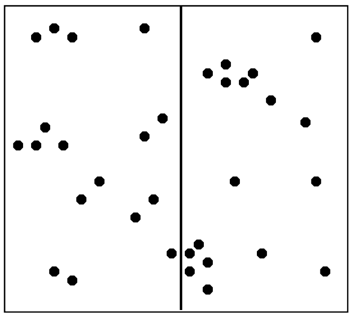


Pt	X	Y
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
…	…	…

We start with a list of *k*-dimensional points.

Slide credit Brigham Anderson, Auton Lab

KD-Tree: Construction



X>.5

NO

Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

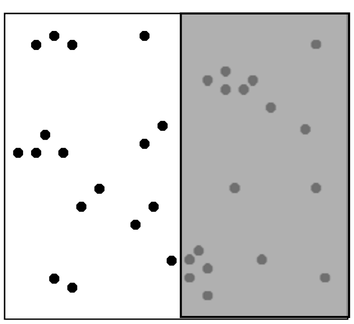
YES

Pt	X	Y
2	1.00	4.31
…	…	…

We can split the points into 2 groups by choosing a dimension X and value V and separating the points into X > V and X <= V.

Slide credit Brigham Anderson, Auton Lab

KD-Tree: Construction



X>.5

NO

Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

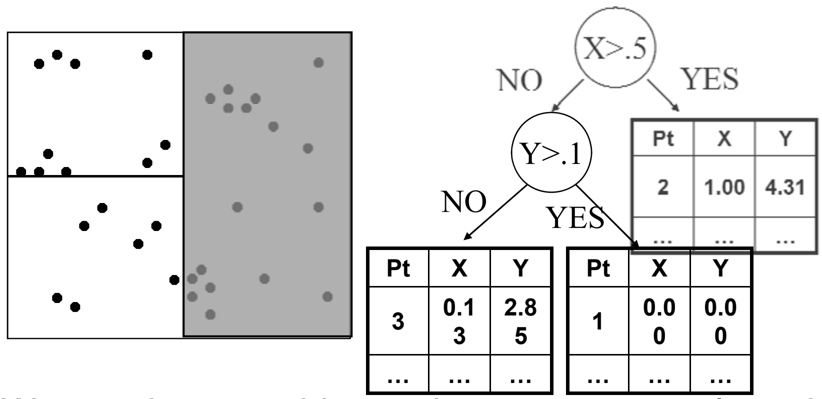
YES

Pt	X	Y
2	1.00	4.31
…	…	…

We can then consider each group separately and possibly split again (along same/different dimension).

Slide credit Brigham Anderson, Auton Lab

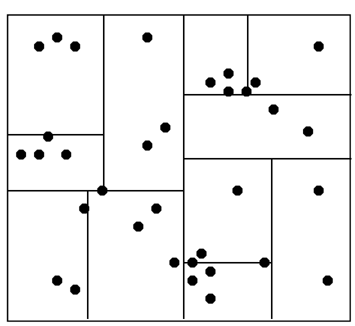
KD-Tree: Construction

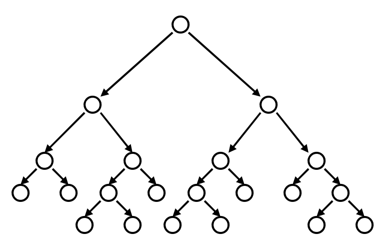


We can then consider each group separately and possibly split again (along same/different dimension).

Slide credit Brigham Anderson, Auton Lab

KD-Tree: Construction

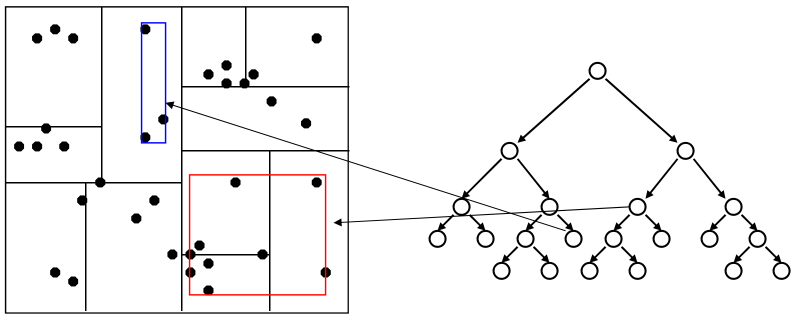




- Keep splitting the points in each set to create a tree structure.
- Each node with no children (leaf node) contains a list of points.

Slide credit Brigham Anderson, Auton Lab

KD-Tree: Construction



Keep track of the (tight) bounds of the points at or below each node.

Slide credit Brigham Anderson, Auton Lab

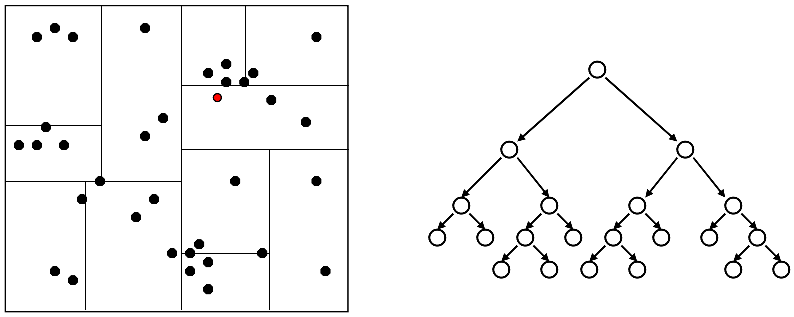
KD-Tree: Construction

Heuristics to make splitting decisions:

- Which dimension do we split along?
  - Widest – axis with highest variance
- Which value do we split at?
  - Median of value of that split dimension for the points.
- When do we stop?
  - When there are fewer then *m* points left OR the box has hit some minimum width.

Slide credit Brigham Anderson, Auton Lab

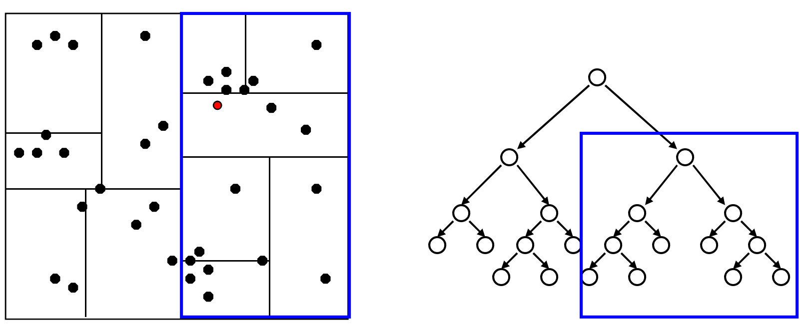
Nearest Neighbor with KD Trees



We traverse the tree looking for the nearest neighbor of the query point.

Slide credit Brigham Anderson, Auton Lab

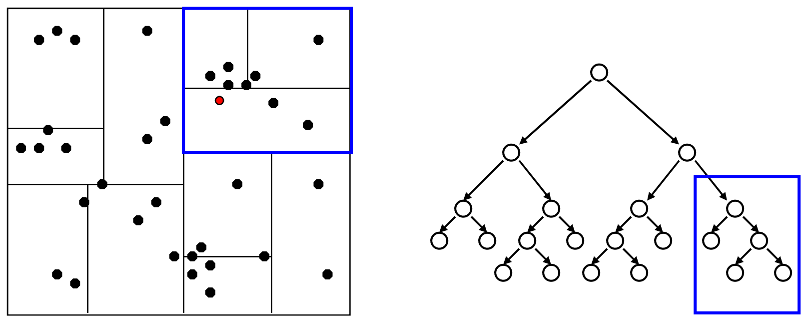
Nearest Neighbor with KD Trees



Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

Slide credit Brigham Anderson, Auton Lab

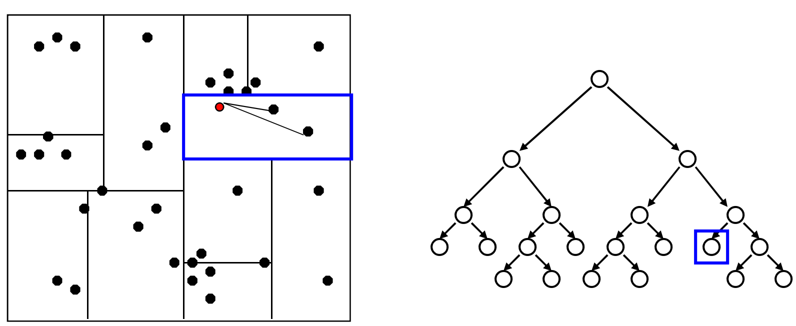
Nearest Neighbor with KD Trees



Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

Slide credit Brigham Anderson, Auton Lab

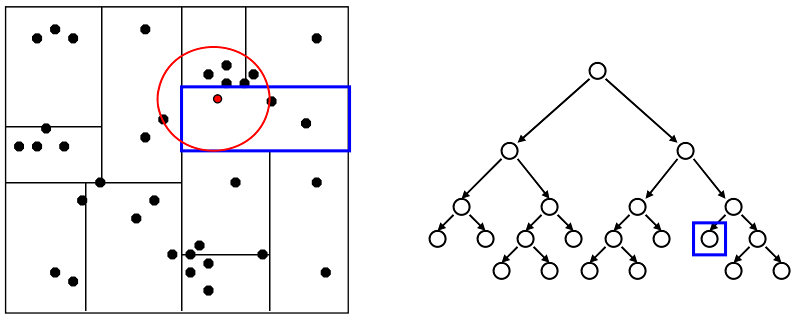
Nearest Neighbor with KD Trees



When we reach a leaf node: compute the distance to each point in the node.

Slide credit Brigham Anderson, Auton Lab

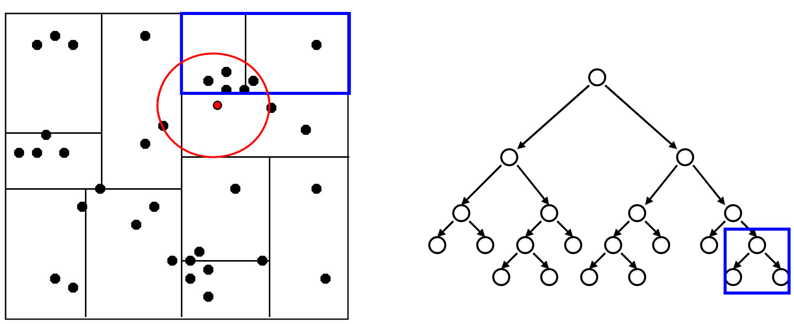
Nearest Neighbor with KD Trees



When we reach a leaf node: compute the distance to each point in the node.

Slide credit Brigham Anderson, Auton Lab

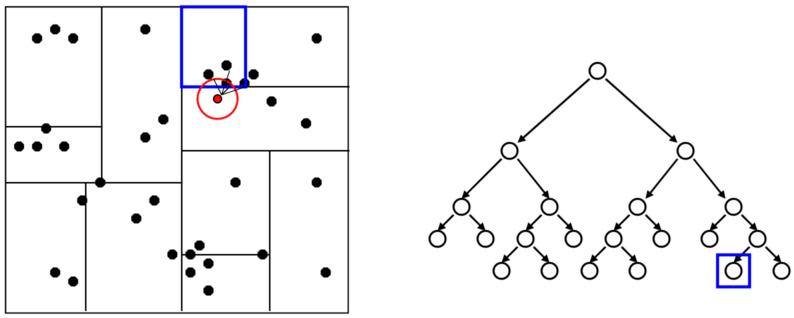
Nearest Neighbor with KD Trees



Then we can backtrack and try the other branch at each node visited.

Slide credit Brigham Anderson, Auton Lab

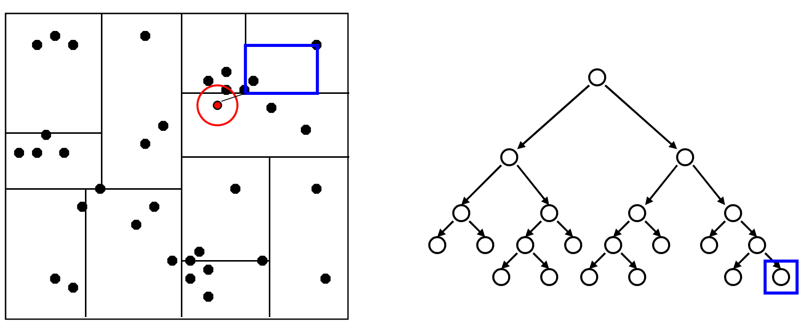
Nearest Neighbor with KD Trees



Each time a new closest node is found, we can update the distance bounds.

Slide credit Brigham Anderson, Auton Lab

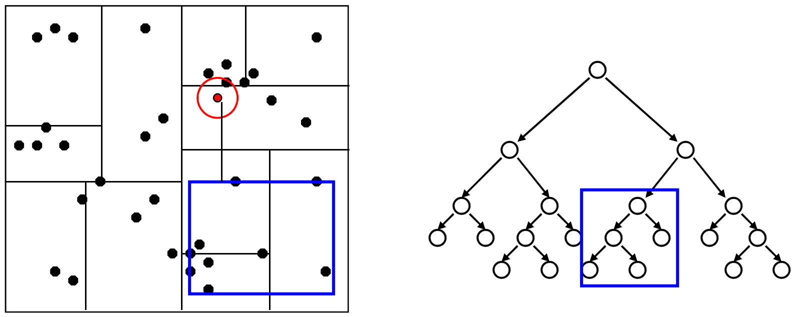
Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Slide credit Brigham Anderson, Auton Lab

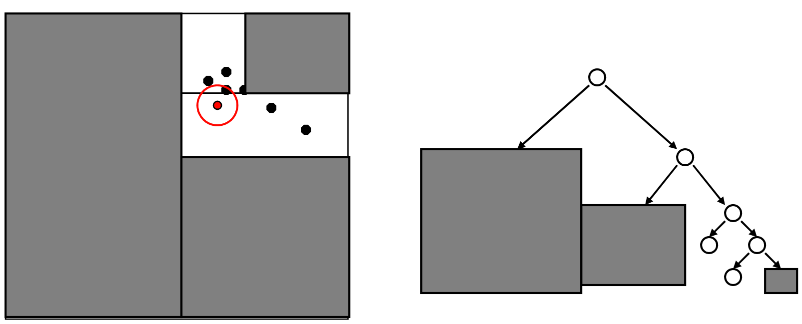
Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Slide credit Brigham Anderson, Auton Lab

Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Slide credit Brigham Anderson, Auton Lab

KD-trees: Variants

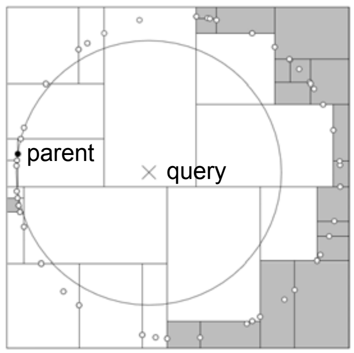
- Approximate search with *priority queue* [Arya & Mount, Beis & Lowe]
- Create *multiple randomized* trees and search all at query time with priority queue
  - e.g. choose split dimension randomly from first D dims of greatest variance [Silpa-Anan & Hartley 2008]
- Stop search early when fixed number of leaf nodes examined (approx result)
- PCA on data first, to align axes with directions of highest variance

KD-trees: Complexity

- Constructing tree with *n* points:
  - O(*n log n) time and O(dn) storage*
- Inserting a new point
  - O(log *n*) time
- Querying for neighbors:
  - $O(n^{1-1/k})$ time

KD-tree limitations

- Poor search time performance with high-dimensional data
- Sensitive to data distribution, bin shapes

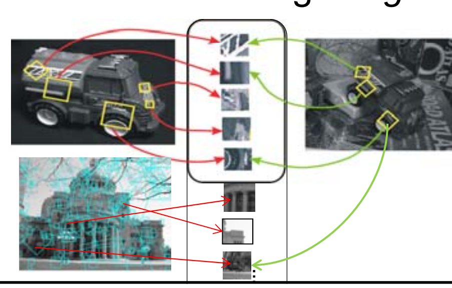


Example: a “bad” distribution that forces almost all nodes to be inspected.

[Andrew Moore, PhD thesis]

KD-tree limitations

- Poor search time performance with high-dimensional data
- Sensitive to data distribution, bin shapes
- Storage requirements
- Purely vector space matching: not exploiting sparsity of features among *images*…



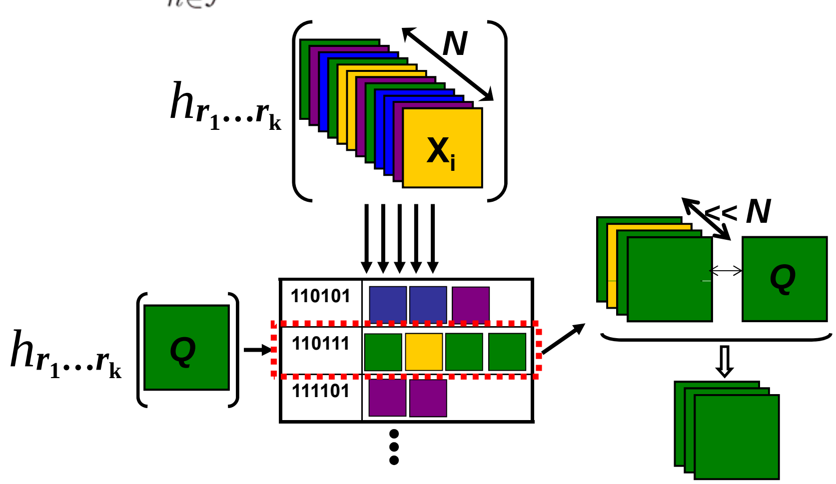
Large-scale visual search

- How to efficiently find similar images/features?
  - Inverted file indexing schemes
  - Low-dimensional descriptors: can use standard efficient data structures for nearest neighbor search
  - High-dimensional descriptors: approximate nearest neighbor search methods more practical
- How to inject supervision into the search?
- How to summarize large collections?

Locality Sensitive Hashing (LSH)

[Indyk and Motwani ‘98, Gionis et al.’99, Charikar ‘02, Andoni et al. ‘04]

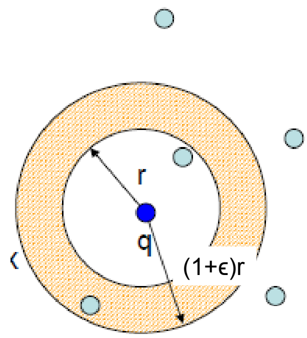
$$\Pr_{h \in \mathcal{F}} [h(x) = h(y)] = sim(x, y)$$



Locality Sensitive Hashing (LSH)

[Indyk and Motwani '98, Gionis et al.'99, Charikar '02, Andoni et al. '04]

- Formally, ensures "approximate" nearest neighbor search
  - With high probability, return a neighbor within radius $(1+\epsilon)r$, if there is one.
  - Guarantee to search only $O\left(N^{\frac{1}{1+\epsilon}}\right)$ of the database
- LSH functions originally for Hamming metric, Lp norms, inner product.

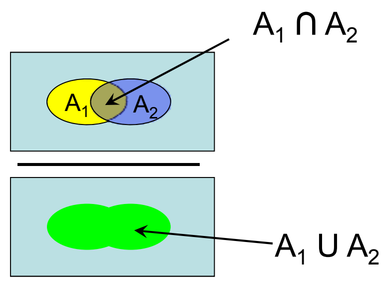


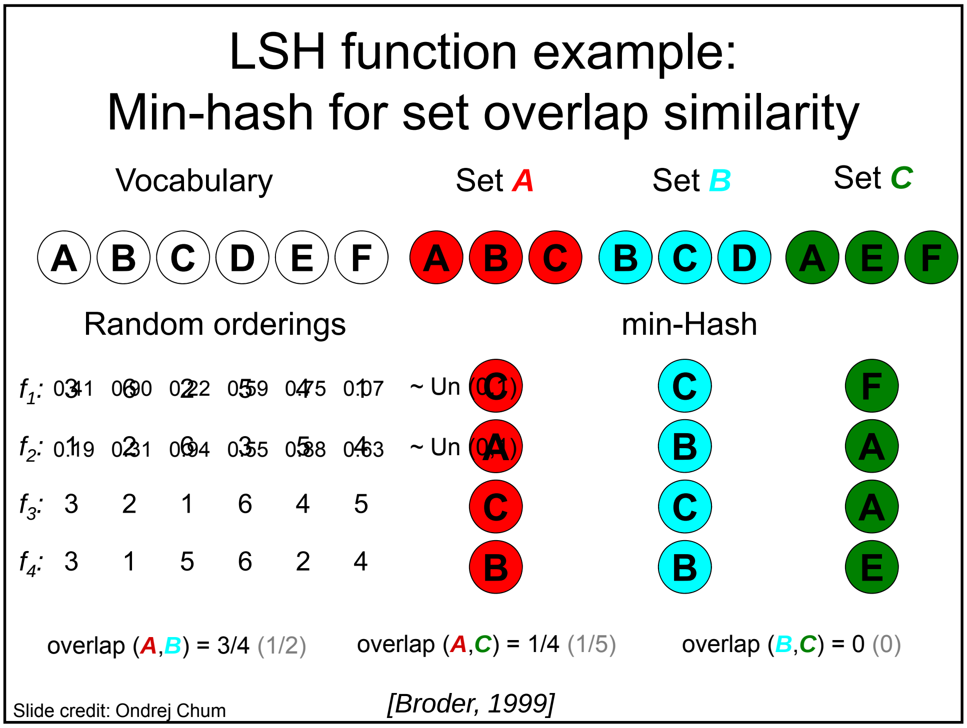
LSH function example: Min-hash for set overlap similarity

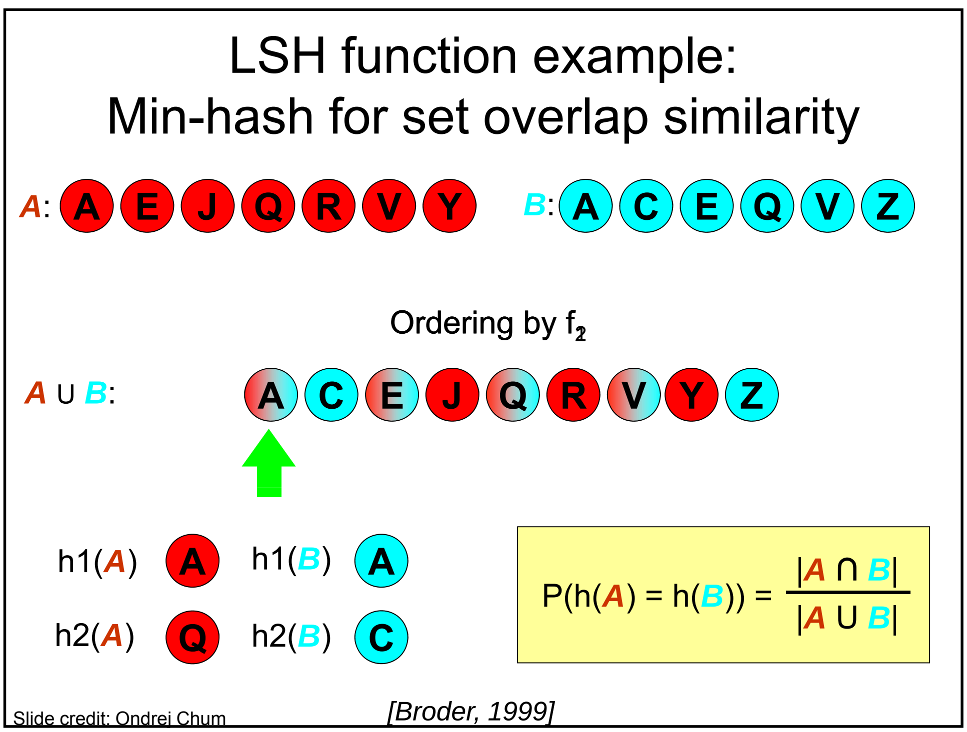
[Broder, 1999]

$$\Pr_{h \in \mathcal{F}}[h(x) = h(y)] = sim(x, y)$$

$$\text{sim}(\mathcal{A}_1, \mathcal{A}_2) = \frac{|\mathcal{A}_1 \cap \mathcal{A}_2|}{|\mathcal{A}_1 \cup \mathcal{A}_2|} \in \langle 0, 1 \rangle$$



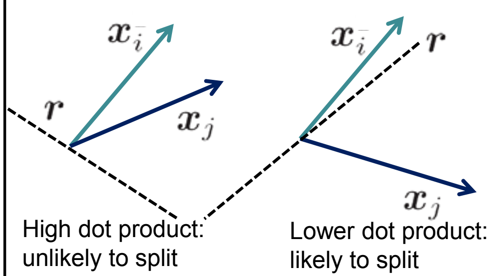

LSH function example:
Min-hash for set overlap similarity
Vocabulary
Set A
Set B
Set C
A B C D E F
A B C
B C D
A E F
Random orderings
min-Hash
f1: 3 6 2 5 4 1
f1: 0.41 0.90 0.22 0.59 0.75 0.07 ~ Un (0,1)
C C F
f2: 1 2 6 3 5 4
f2: 0.19 0.31 0.94 0.55 0.88 0.63 ~ Un (0,1)
A B A
f3: 3 2 1 6 4 5
C C A
f4: 3 1 5 6 2 4
B B E
overlap (A,B) = 3/4 (1/2)
overlap (A,C) = 1/4 (1/5)
overlap (B,C) = 0 (0)
Slide credit: Ondrej Chum
[Broder, 1999]


LSH function example:
Min-hash for set overlap similarity
A: A E J Q R V Y
B: A C E Q V Z
Ordering by f1
Ordering by f2
A U B:
A C E J Q R V Y Z
h1(A) A
h1(B) A
h2(A) Q
h2(B) C
P(h(A) = h(B)) = |A ∩ B| / |A U B|
Slide credit: Ondrej Chum
[Broder, 1999]

LSH function example: inner product similarity

The probability that a *random hyperplane* separates two unit vectors depends on the angle between them:

$$\Pr[\text{sign}(\boldsymbol{x}_i^T \boldsymbol{r}) = \text{sign}(\boldsymbol{x}_j^T \boldsymbol{r})] = 1 - \frac{1}{\pi}\cos^{-1}(\boldsymbol{x}_i^T \boldsymbol{x}_j)$$



Corresponding hash function:

$$h_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T \boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

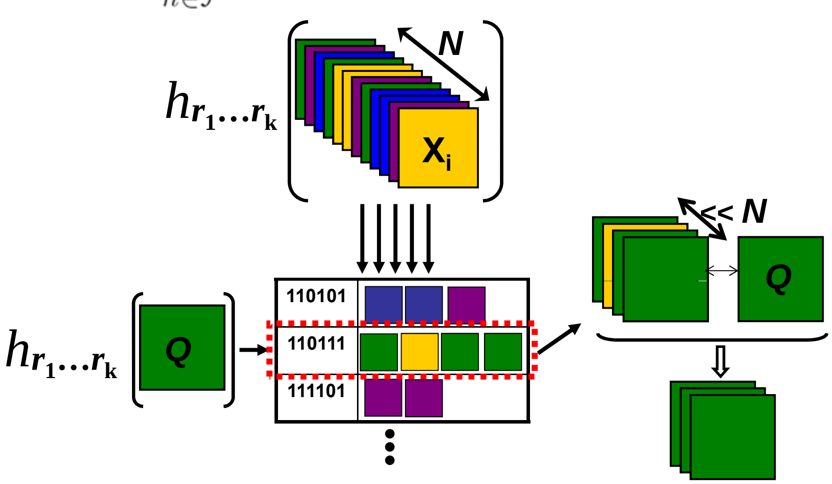
for $\vec{r}_i \sim N(\mu = 0, \sigma^2 = 1)$

[Goemans and Williamson 1995, Charikar 2004]

Locality Sensitive Hashing (LSH)

[Indyk and Motwani '98, Gionis et al.'99, Charikar '02, Andoni et al. '04]

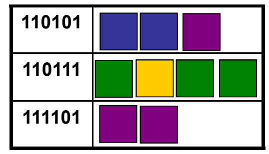
$$\Pr_{h \in \mathcal{F}}[h(x) = h(y)] = sim(x, y)$$



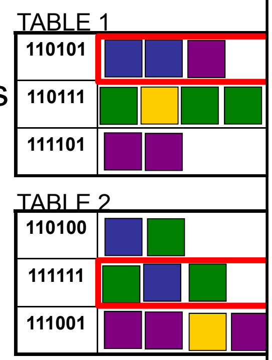
Multiple hash functions and tables

- Generate k such hash functions, concatenate outputs into hash key:

$$P\left(h_{1,\dots,k}(x) = h_{1,\dots,k}(y)\right) =$$



- To increase recall, search multiple independently generated hash tables
  - Search/rank the union of collisions in each table, or
  - Require that two examples in at least T of the tables to consider them similar.



Kernelized LSH (KLSH)

[Kulis & Grauman, ICCV 2009]

Given: an *arbitrary* kernel function:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

Goal: compute hash function

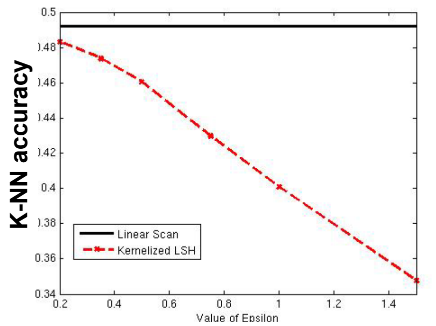
$$h_{\boldsymbol{r}}(\phi(\boldsymbol{x})) = \text{sign}(\boldsymbol{r}^T \phi(\boldsymbol{x}))$$

for some Gaussian random vector $\boldsymbol{r}$.

Key issue: random vectors are also in the implicit feature space, to which we may only have access via the kernel.

Our result: derive appropriate hash functions of the form:

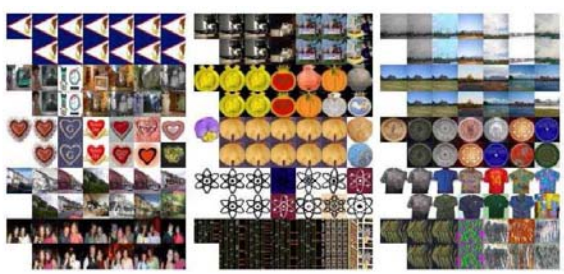
$$h(\phi(\boldsymbol{x})) = \text{sign}(\textstyle\sum_i \boldsymbol{w}(i)\kappa(\boldsymbol{x}, \boldsymbol{x}_i))$$

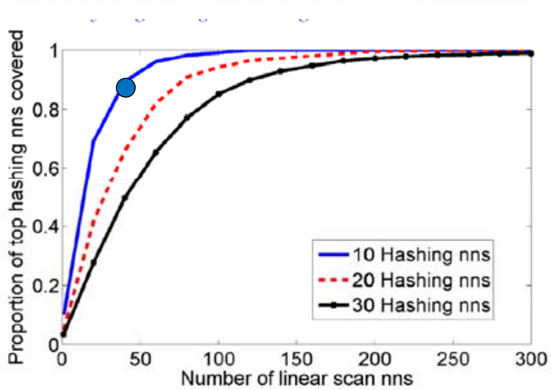


slower search ⟷ faster search

KLSH results with the **Flickr scenes** dataset

- Acceptance of arbitrary kernels makes hashing with popular **X^2 kernel** feasible.
- ϵ parameter allows control on speed-accuracy tradeoff.





KLSH results with the **80 Million** Tiny Image dataset

- KLSH searches **less than 1%** of the database to find a query's approximate near neighbors.
- How accurate are the retrievals? For an average query, 90% of KLSH's top 10 retrievals will be within the top 50 linear scan neighbors

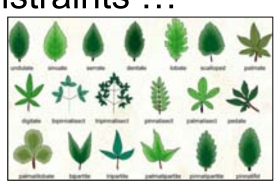
Large-scale visual search

- How to efficiently find similar images/features?
  - Inverted file indexing schemes
  - Low-dimensional descriptors: can use standard efficient data structures for nearest neighbor search
  - High-dimensional descriptors: approximate nearest neighbor search methods more practical
- How to inject supervision into the search?
- How to summarize large collections?

Learning how to compare images

Choosing a generic distance function ignores task-specific constraints …

User feedback



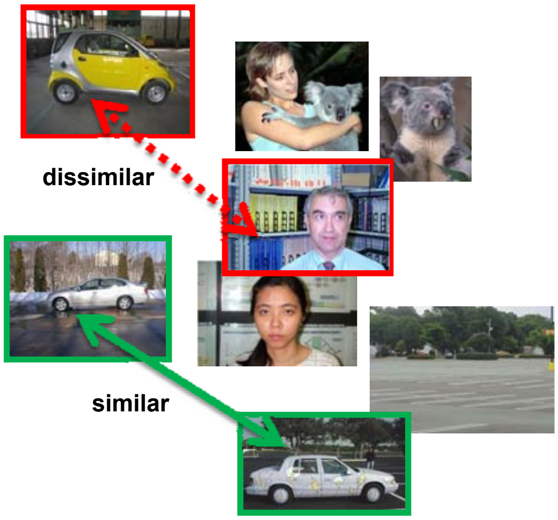
Problem-specific knowledge

Detected video shots, tracked objects

Partially labeled image databases

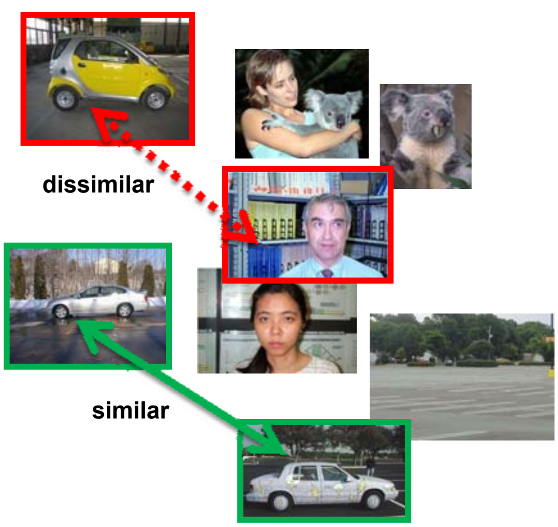
Known correspondences

Learning how to compare images



- Exploit (dis)similarity constraints to construct more useful distance *and* hash functions

Learning how to compare images

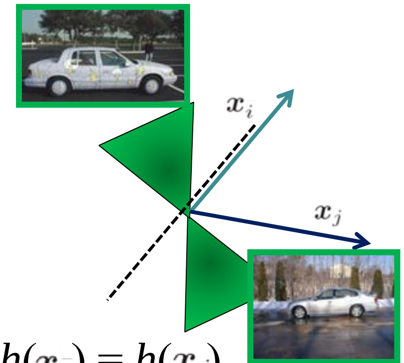


- Exploit (dis)similarity constraints to construct more useful distance *and* hash functions

[Weinberger et al. 2004, Hertz et al. 2004, Frome et al. 2007, Varma & Ray 2007, Kumar et al. 2007,…]

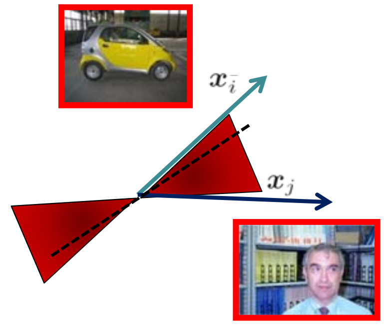
Semi-supervised hash functions

[Jain, Kulis, & Grauman, CVPR 2008]



$h(x_i) = h(x_j)$

Less likely to split pairs like those with similarity constraint



$h(x_i) \neq h(x_j)$

More likely to split pairs like those with dissimilarity constraint

Semi-supervised hash functions: Learned Mahalanobis metrics

- Distance parameterized by p.d. $d \times d$ matrix A:

$$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A(\boldsymbol{x}_i - \boldsymbol{x}_j)$$

- Similarity measure (kernel) is associated generalized inner product

$$s_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j$$

- An efficient method to learn the parameters: information theoretic metric learning [Davis et al. 2007]

$$\begin{array}{lll}
\min_A & D_{\ell d}(A, A_0) & \\
\text{s.t.} & (\mathbf{x}_i - \mathbf{x}_j)^T A(\mathbf{x}_i - \mathbf{x}_j) \leq u & \text{if } (i,j) \in \mathcal{S} \text{ [similarity constraints]} \\
 & (\mathbf{x}_i - \mathbf{x}_j)^T A(\mathbf{x}_i - \mathbf{x}_j) \geq \ell & \text{if } (i,j) \in \mathcal{D} \text{ [dissimilarity constraints]}
\end{array}$$

Semi-supervised hash functions: Learned Mahalanobis metrics

- Given learned metric with $A = G^T G$
- We generate parameterized hash functions for $s_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j$:

$$h_{\boldsymbol{r},A}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T G\boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

This satisfies the locality-sensitivity condition:

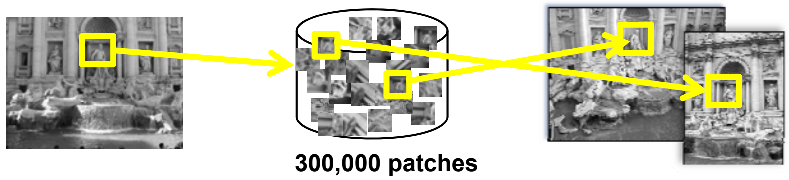
$$\Pr\left[h_{\boldsymbol{r},A}(\boldsymbol{x}_i) = h_{\boldsymbol{r},A}(\boldsymbol{x}_j)\right] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\boldsymbol{x}_i^T A \boldsymbol{x}_j}{\sqrt{|G\boldsymbol{x}_i||G\boldsymbol{x}_j|}}\right)$$

Semi-supervised hash functions: Learned Mahalanobis metrics

- Image data often high-dimensional, or want to apply metric learner in kernel space.
- High-d inputs are sparse, but $A = G^T G$ may be dense $\longrightarrow$ how to compute $r^T G x$?
- We derive an *implicit* update rule that simultaneously updates metric and hash function parameters:

$$r^T G x = \underbrace{r^T x} + \sum_{i=1}^{c} \gamma_i^r \underbrace{x_i^T x}_{\text{Compare input to constrained examples}}$$

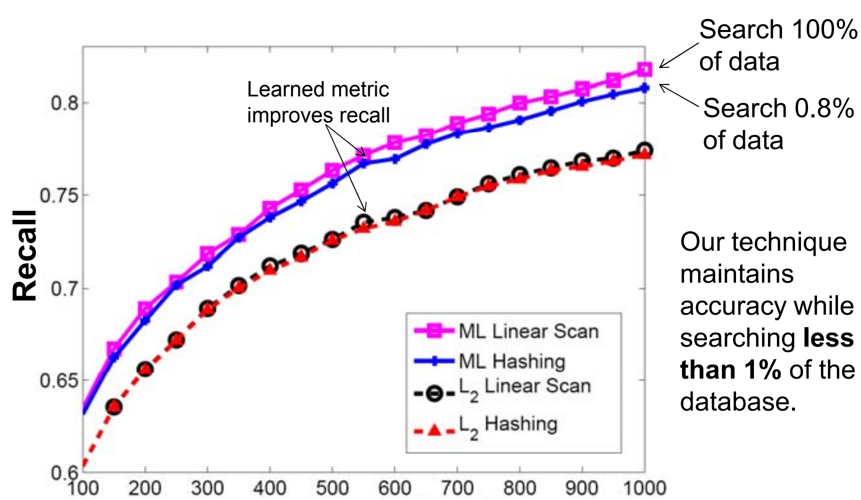
Results: Photo Tourism dataset



[Snavely, Seitz, Szeliski, 2006]

- Goal: match patches associated with same 3d object point
- More accurate matches → better reconstruction

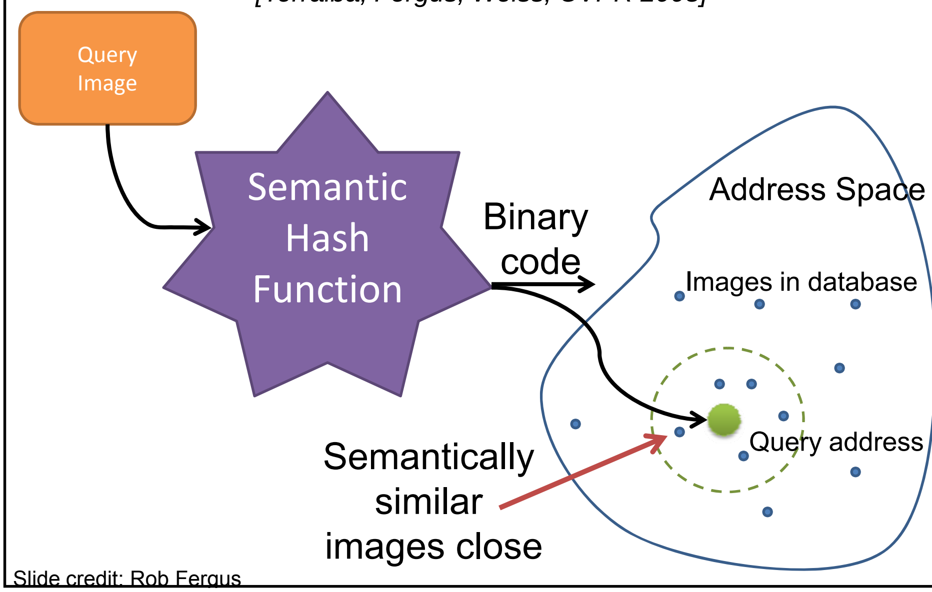
Results: Photo Tourism dataset



Our technique maintains accuracy while searching **less than 1%** of the database.

Semantic Hashing

[Torralba, Fergus, Weiss, CVPR 2008]



Slide credit: Rob Fergus

Semantic Hashing

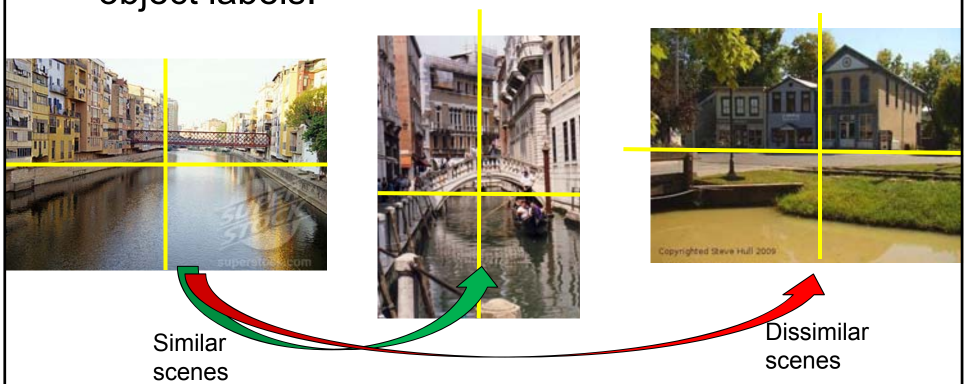
[Torralba, Fergus, Weiss, CVPR 2008]

- Each image code is a memory address
- Find neighbors by exploring Hamming ball around query address
- Lookup time depends on radius of ball & length of code
- Explored with different semantic distances and learning algorithms (e.g., boosting, RBMs)
  - Learned functions outperform raw LSH functions

Slide credit: Rob Fergus

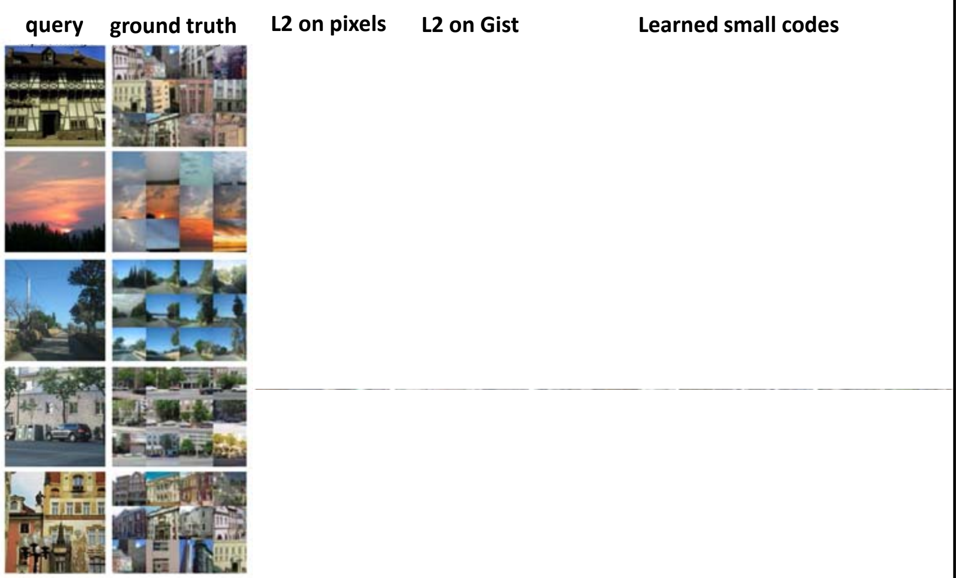
Semantic Hashing

- **Idea**: Define *target semantic distance* to be the spatial pyramid distance, but where tokens are object labels.



- Train with similarity constraints from labeled examples.

Results: example LabelMe retrievals



- Neighbors under different distance metrics, 22K image db

Large-scale visual search

- How to efficiently find similar images/features?
  - Inverted file indexing schemes
  - Low-dimensional descriptors: can use standard efficient data structures for nearest neighbor search
  - High-dimensional descriptors: approximate nearest neighbor search methods more practical
- How to inject supervision into the search?
- How to summarize large collections?

Mining for common visual patterns

In addition to visual search, want to be able to **summarize, mine, and rank** the large collection as a whole.

- What is common?
- What is unusual?
- What co-occurs?
- Which exemplars are most representative?

Mining for common visual patterns

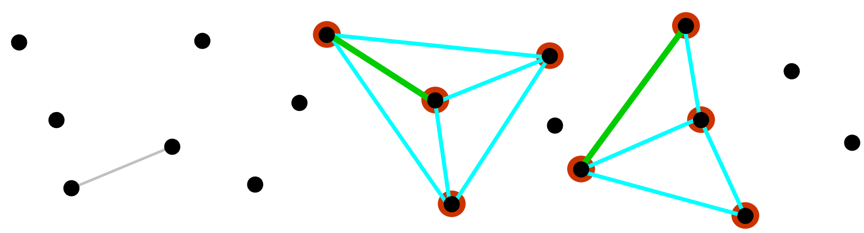
In addition to visual search, want to be able to **summarize, mine, and rank** the large collection as a whole.

We'll look briefly at a few recent examples:

- **Connected component clustering** via hashing [Geometric Min-hash, Chum et al. 2009]
- **Visual Rank** to choose "image authorities" [Jing and Baluja, 2008]
- **Frequent item-set mining** with spatial patterns [Quack et al., 2007]

Connected component clustering with hashing

1. Detect seed pairs via hash collisions
2. Hash to related images
3. Compute connected components of the graph



Contrast with frequently used quadratic-time clustering algorithms

Slide credit: Ondrej Chum

Geometric Min-hash

[Chum, Perdoch, Matas, CVPR 2009]

- Main idea: build spatial relationships into the hash key construction:
  - Select first hash output according to min hash ("central word")
  - Then append subsequent hash outputs from within its neighborhood

Figure from Ondrej Chum

Results: Geometric Min-hash clustering

[Chum, Perdoch, Matas, CVPR 2009]

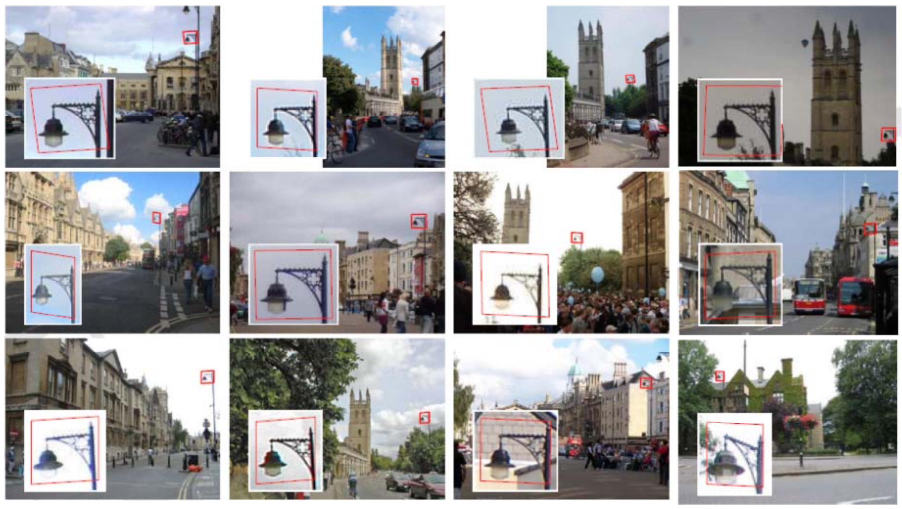
100 000 Images downloaded from FLICKR

Includes 11 Oxford Landmarks with manually labeled ground truth

Slide credit: Ondrej Chum

Results: Geometric Min-hash clustering

[Chum, Perdoch, Matas, CVPR 2009]



Slide credit: Ondrej Chum

Discovering small objects

Results: Geometric Min-hash clustering

[Chum, Perdoch, Matas, CVPR 2009]

Slide credit: Ondrej Chum

Discovering small objects

Mining for common visual patterns

In addition to visual search, want to be able to **summarize, mine, and rank** the large collection as a whole.

We'll look briefly at a few recent examples:

- **Connected component clustering** via hashing [Geometric Min-hash, Chum et al. 2009]
- **Visual Rank** to choose "image authorities" [Jing and Baluja, 2008]
- **Frequent item-set mining** with spatial patterns [Quack et al., 2007]

Visual Rank: motivation

Product search **Mixed-type search**

- **Goal:** select small set of "best" images to display among millions of candidates

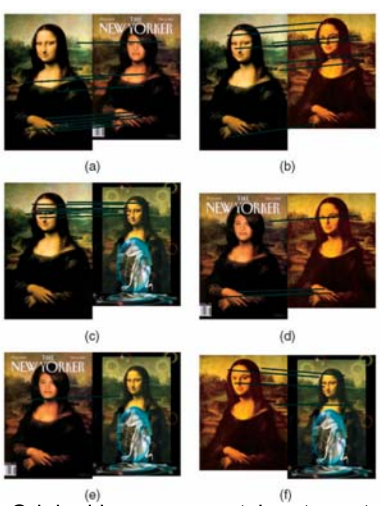
Visual Rank

[Jing and Baluja, PAMI 2008]

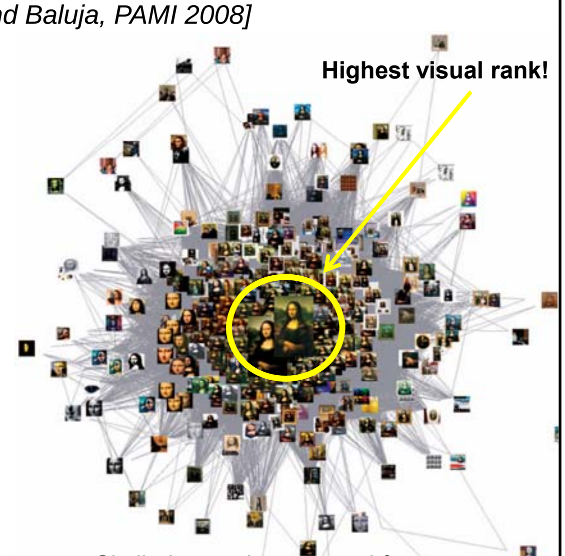
- Compute relative "authority" of an image based on random walk principle.
  - Application of PageRank to visual data
- **Main ideas**:
  - Graph weights = number of matched local features between two images
  - Exploit text search to narrow scope of each graph
  - Use LSH to make similarity computations efficient

Results: Visual Rank

[Jing and Baluja, PAMI 2008]



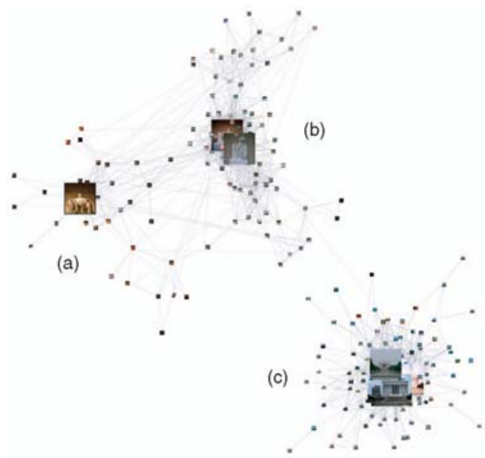
Original has more matches to rest



Similarity graph generated from top 1,000 text search results of "Mona-Lisa"

Results: Visual Rank

[Jing and Baluja, PAMI 2008]



Similarity graph generated from top 1,000 text search results of "Lincoln Memorial".
Note the diversity of the high-ranked images.

Mining for common visual patterns

In addition to visual search, want to be able to **summarize, mine, and rank** the large collection as a whole.

We'll look briefly at a few recent examples:

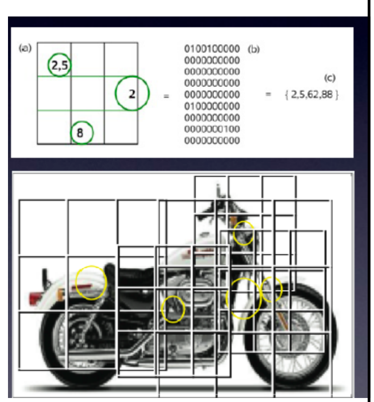
- **Connected component clustering** via hashing [Geometric Min-hash, Chum et al. 2009]
- **Visual Rank** to choose "image authorities" [Jing and Baluja, 2008]
- **Frequent item-set mining** with spatial patterns [Quack et al., 2007]

Frequent item-sets

Frequent item-set mining for spatial visual patterns

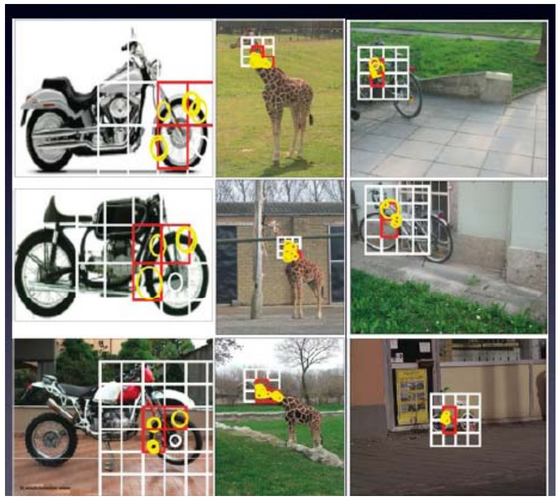
[Quack, Ferrari, Leibe, Van Gool, CIVR 2006, ICCV 2007]

- What configurations of local features frequently occur in large collection?
- **Main idea**: Identify *item-sets* (visual word layouts) that often occur in *transactions* (images)
- Efficient algorithms from data mining (e.g., Apriori algorithm, Agrawal 1993)



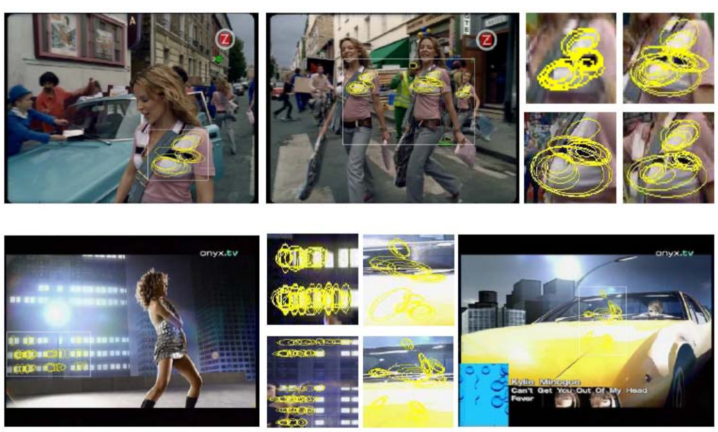
Frequent item-set mining for spatial visual patterns

[Quack, Ferrari, Leibe, Van Gool, CIVR 2006, ICCV 2007]



Frequent item-set mining for spatial visual patterns

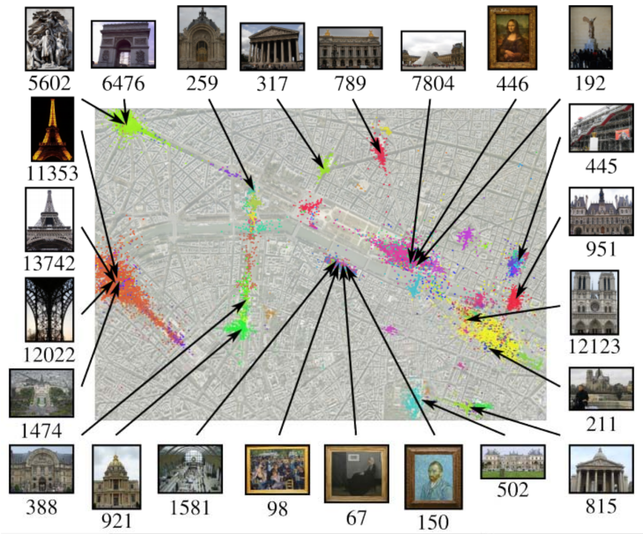
[Quack, Ferrari, Leibe, Van Gool, CIVR 2006, ICCV 2007]



Two example itemset clusters

Discovering favorite views

Discovering Favorite Views of Popular Places with Iconoid Shift. T. Weyand and B. Leibe. ICCV 2011.



Conclusions

- Key considerations in visual search design: similarity, representation, scalable search procedures, integrating learning
- Tradeoffs in large-scale data: complexity but also data richness
- Allowing visual queries and automatic organization can transform how *both* visual and non-visual data are accessed.