Efficient visual search

Gül Varol

IMAGINE team, École des Ponts ParisTech

gul.varol@enpc.fr

https://gulvarol.github.io/

@RecVis, 22.10.2024

With many slides from: J. Sivic, O. Chum, K. Grauman, S. Lazebnik, B. Leibe, D. Lowe, J. Philbin, J. Ponce, D. Nister, C. Schmid, N. Snavely, A. Zisserman, Y. Kalantidis, …

Announcements

- Final project topics out — will go over them at the end of today's lecture

- Assignment 1 due next week

Instance-level recognition

[Week 1] Introduction, local features and matching

[Week 2] Camera geometry, image processing (J. Ponce)

[Today] Efficient visual search

[Next week] Supervised learning and deep learning

Agenda: Efficient Visual Search

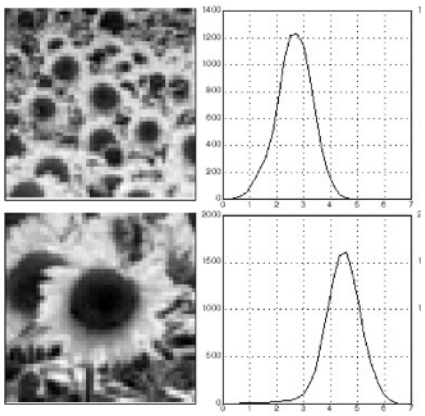
1) Efficient matching of local descriptors
    - Approximate nearest neighbor search with kd-trees
    - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
    - Bag-of-visual-words
    - Query expansion
    - Product quantization
3) Examples from recent works

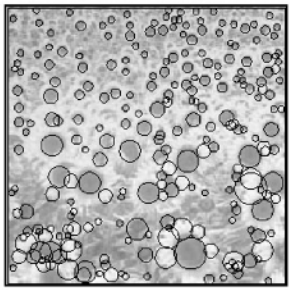
Agenda: Efficient Visual Search

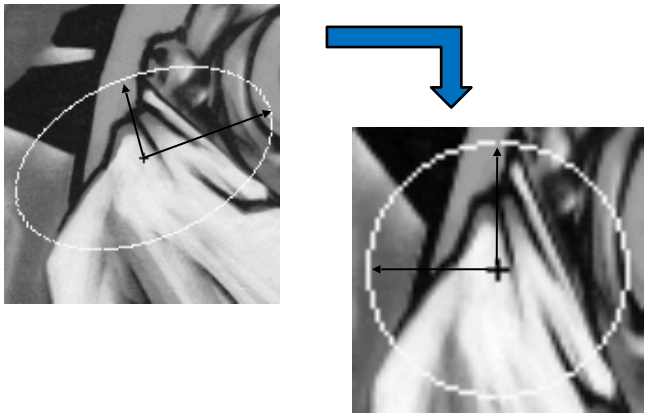
1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

Recap: Local features

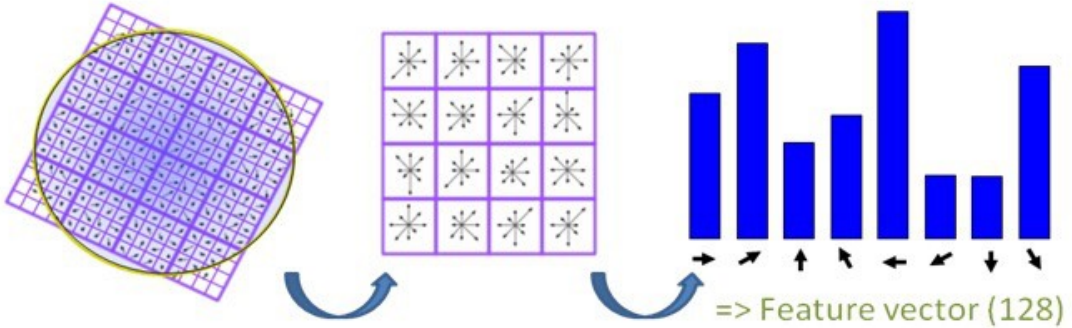
Scale and affine co-variant feature detection





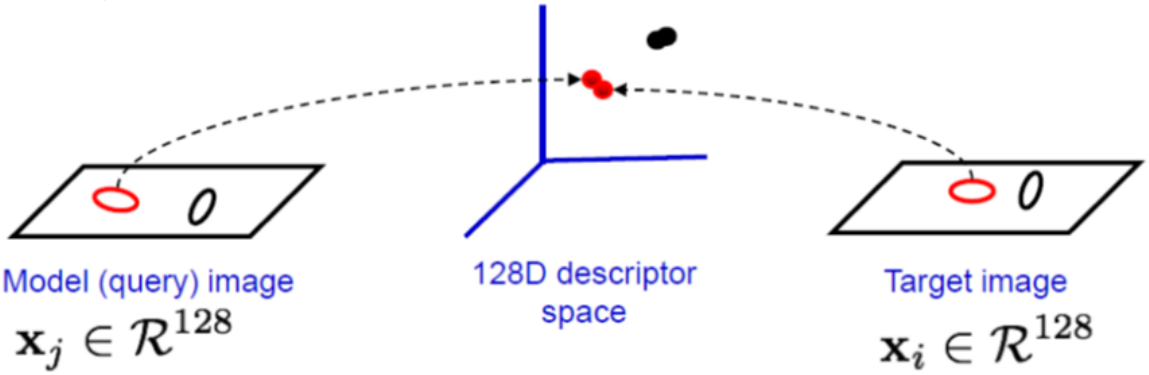


Feature descriptors (SIFT)

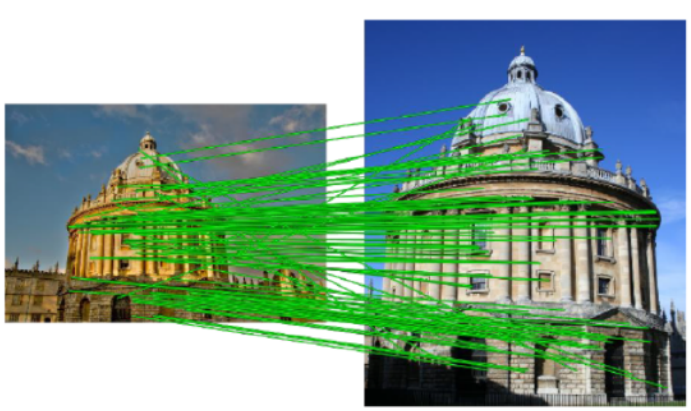


Recap: Matching

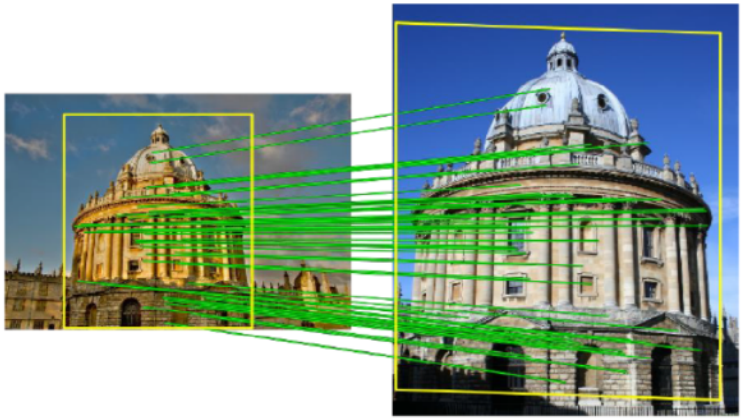
Feature matching



Geometric verification (RANSAC, Hough transform)

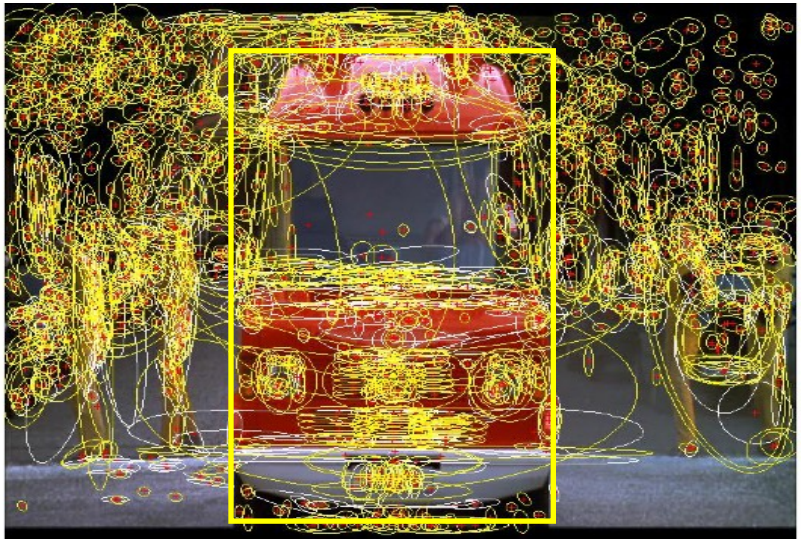


Tentative matches



Matches consistent with an affine transformation

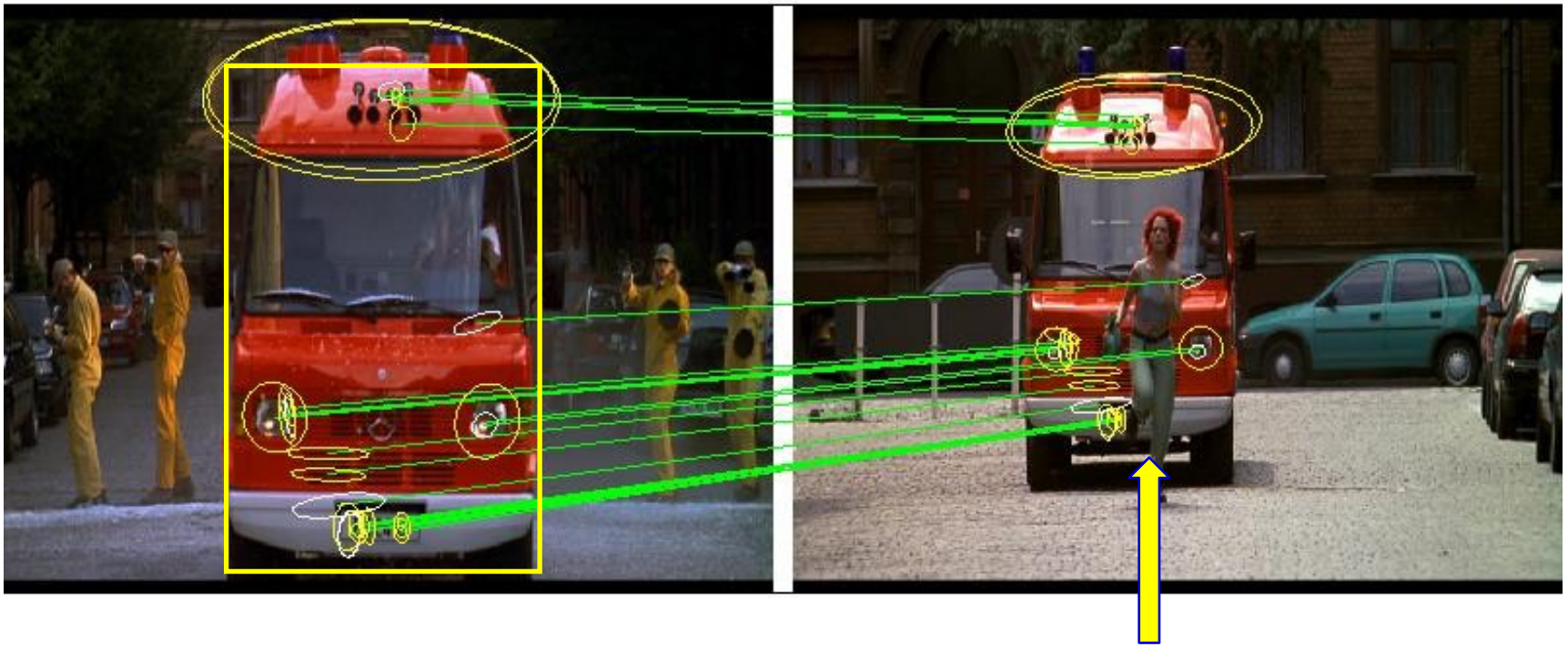
Recap: Matching



1000+ descriptors per image

Recap: Matching

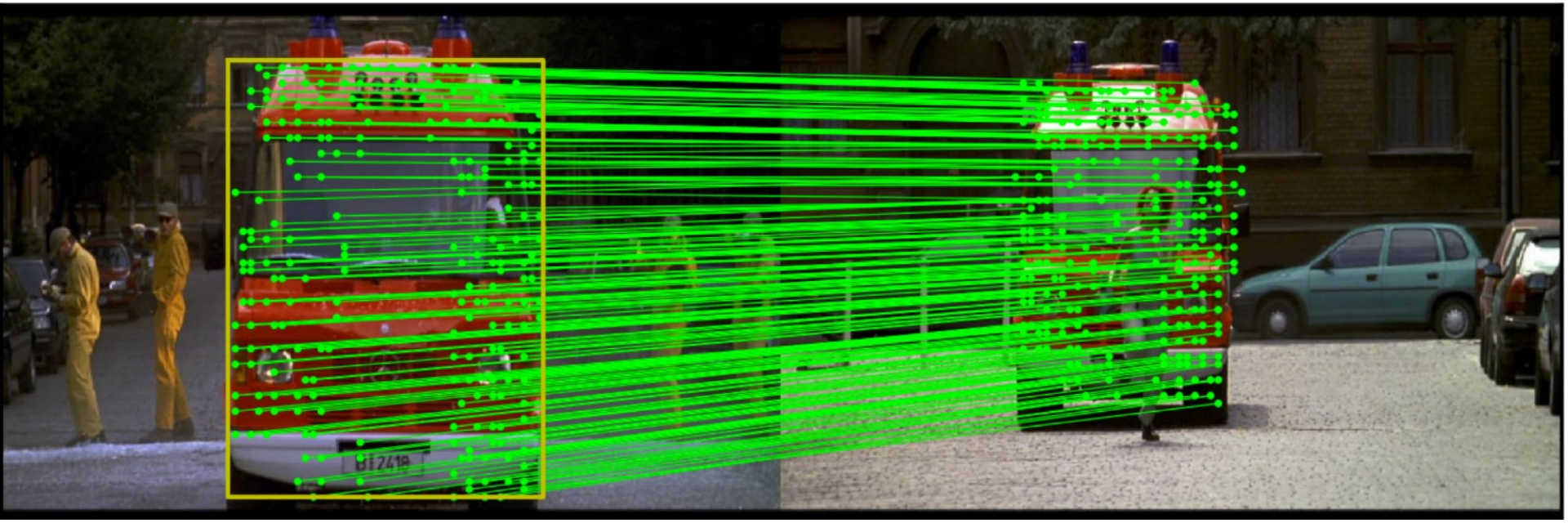
Match regions between frames using SIFT descriptors and spatial consistency



Multiple regions overcome problem of partial occlusion

Matching: Update

Better matches using recent CNN features instead of SIFT



Rocco, Cimpoi, Arandjelovic, Torii, Pajdla, Sivic,
Neighbourhood consensus networks, NIPS 2018

What about multiple images?

So far, we have seen successful matching of a query image to a **single target image** using local features.

How to generalize this strategy to multiple target images with reasonable complexity?

- 10, 10^2, 10^3, …, 10^7, … 10^{10}, … images?
- 10, 100, 1k, …, 10M, … 10B

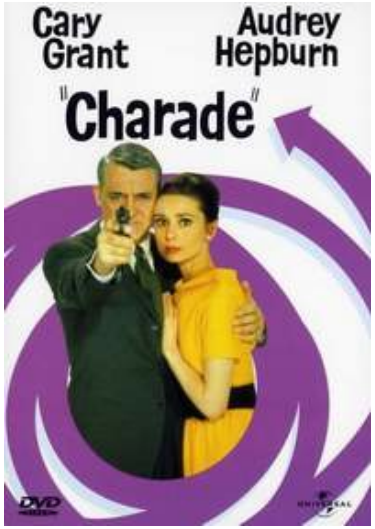
✳ 90B uploads to FB in 2012

✳ 380B and 880B photos taken in 2012, 2014 (~10^{13})

Example: Visual search in an entire feature length movie

Visually defined query

"Find this bag"



"Charade" [Donen, 1963]

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

by Josef Sivic et al. 2003

Example: Visual search in an entire feature length movie

Visually defined query

"Find this bag"

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

by Josef Sivic et al. 2003

Techniques from efficient search are also used for

- Efficient representation of memory in neural networks

  Lample et al., Large memory layers with product keys, arXiv preprint arXiv:1907.05242.

- Reducing memory footprint of neural networks by quantization

  Fan et al., Training with Quantization Noise for Extreme Model Compression, ICLR 2021 (preprint arXiv:2004.07320)

  P. Stock et al., And the Bit Goes Down: Revisiting the Quantization of Neural Networks, ICLR 2020 (preprint arXiv:1907.05686)

  Gong et al., Compressing deep convolutional networks using vector quantization, arXiv:1412.6115, 2014

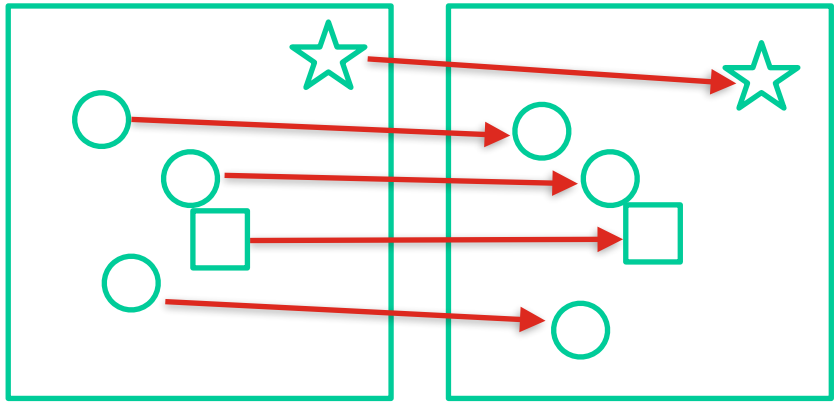
- Efficient search of video language embedding spaces.

  Miech et al., HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips, ICCV 2019.

  Demo: https://www.di.ens.fr/willow/research/howto100m/

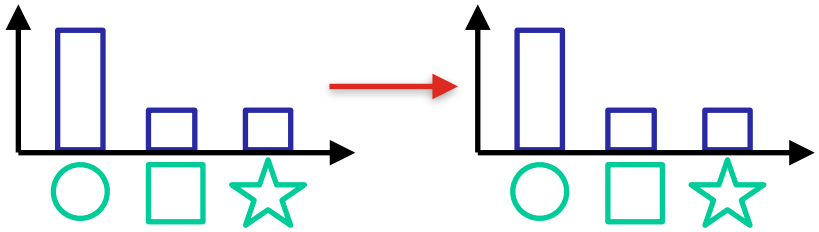
Two strategies

1. Efficient approximate nearest neighbor search on local feature descriptors.

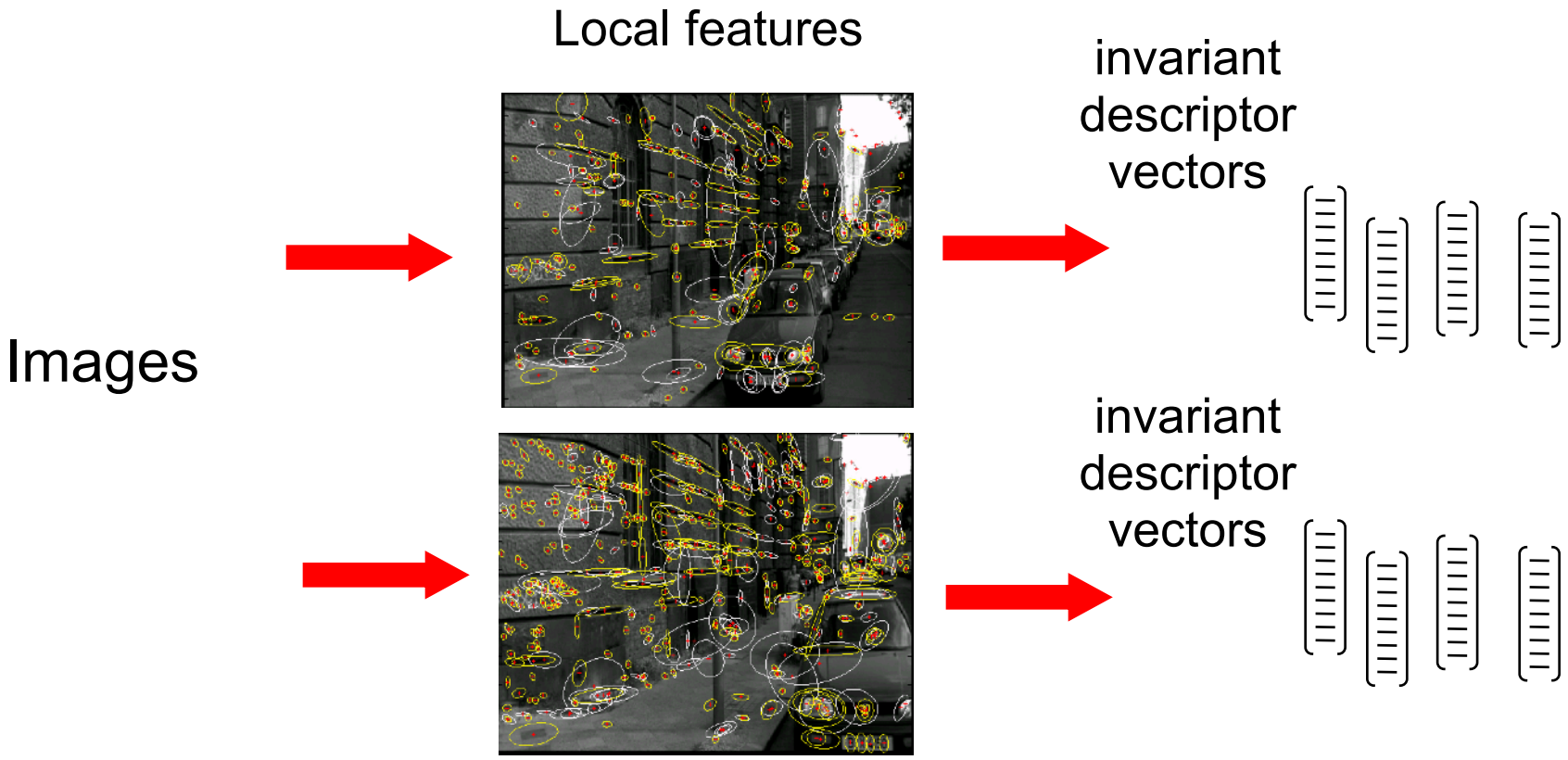


2. Quantize descriptors into a “visual vocabulary” and use efficient techniques from text retrieval.

   (Bag-of-words representation)



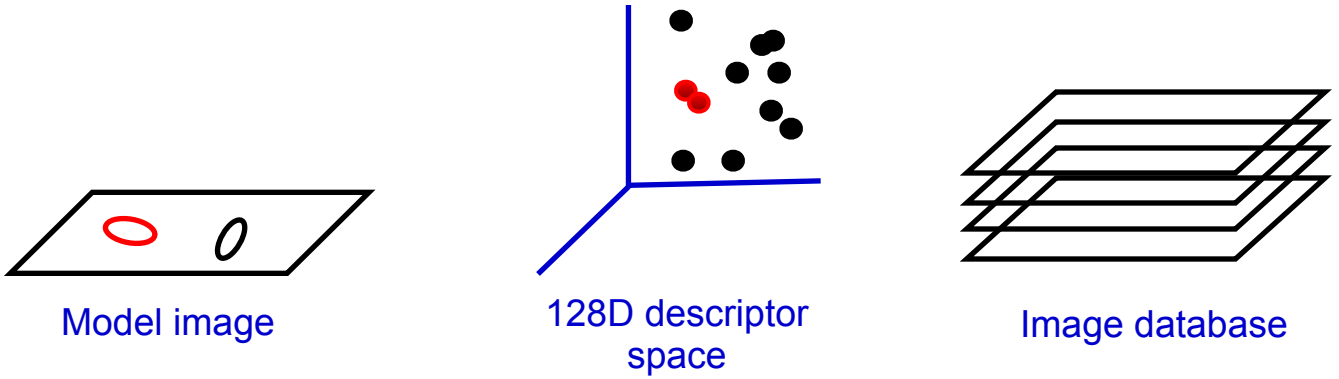
Strategy I: Efficient approximate NN search



1. Compute local features in each image independently (offline)
2. "Label" each feature by a descriptor vector based on its intensity (offline)
3. Finding corresponding features → finding nearest neighbour vectors
4. Rank matched images by number of (tentatively) corresponding regions
5. Verify top ranked images based on spatial consistency

Finding nearest neighbour vectors

Establish correspondences between a query image and images in the database by **nearest neighbour matching** on SIFT vectors



Model image

128D descriptor space

Image database

Solve following problem for all feature vectors, $\mathbf{x}_j \in \mathcal{R}^{128}$, in the query image:

$$\forall j \; NN(j) = \arg\min_i ||\mathbf{x}_i - \mathbf{x}_j||$$

where, $\mathbf{x}_i \in \mathcal{R}^{128}$, are features from **all** the database images.

Quick look at the complexity of the NN-search

N … images

M … regions per image (~1000)

D … dimension of the descriptor (~128)

Exhaustive linear search: O(M NMD)

Example:

- Matching two images (N=1), each having 1000 SIFT descriptors
  Nearest neighbors search: 0.4 s (2 GHz CPU, implementation in C)
- Memory footprint: 1000 * 128 = 128kB / image

# of images	CPU time	Memory req.
N = 1,000	~7min	(~100MB)
N = 10,000	~1h7min	(~ 1GB)
…		
N = 10^7	~115 days	(~ 1TB)
…		
(Some) Images on Facebook:		
N = 10^{10}	~300 years	(~ 1PB)

Nearest-neighbor matching

Solve following problem for all feature vectors, $\mathbf{x}_j$, in the query image:

$$\forall j \; NN(j) = \arg\min_i ||\mathbf{x}_i - \mathbf{x}_j||$$

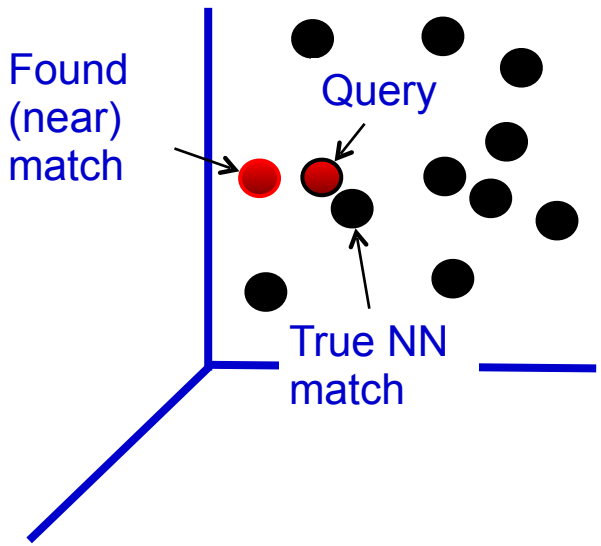
where $\mathbf{x}_i$ are features in database images.

Nearest-neighbour matching is the major computational bottleneck

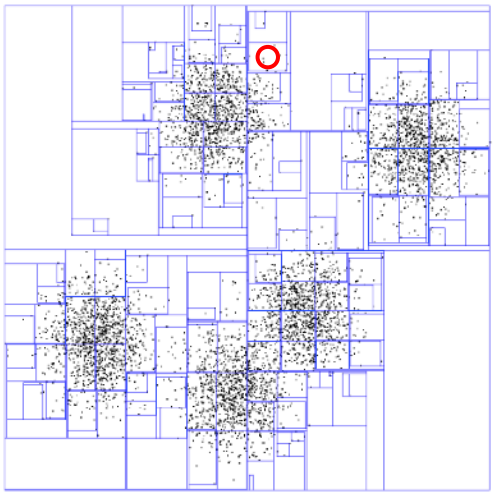
- Linear search performs *dn* operations for *n* features in the database and *d* dimensions
- No exact methods are faster than linear search for d>10
- Approximate methods can be much faster, but at the cost of missing some correct matches

Finding *approximate* nearest neighbour vectors

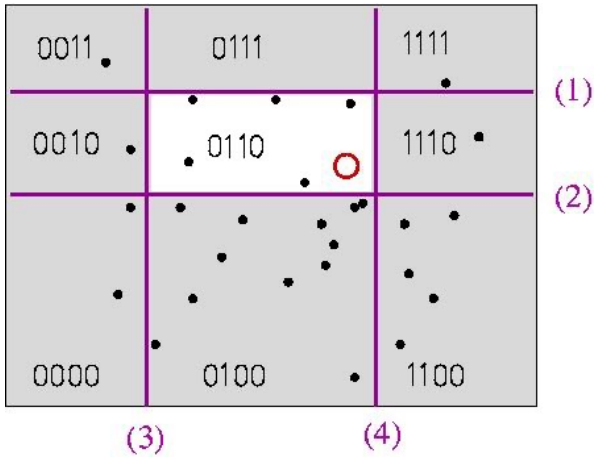
- Approximate method is not guaranteed to find the nearest neighbour.
- Can be much faster, but at the cost of missing some nearest matches



Approximate nearest neighbor search



- **Best-Bin First (BBF)**, a variant of **k-d trees** that uses priority queue to examine most promising branches first [Beis & Lowe, CVPR 1997]
- Extended to **multiple randomized trees** in [Muja & Lowe, 2009]



- **Locality-Sensitive Hashing (LSH)**, a randomized **hashing** technique using hash functions that map similar points to the same bin, with high probability [Indyk & Motwani, 1998]

Can reduce the complexity of the search, e.g. O(log N) for k-d tree.

But at the cost of missing some nearest matches.

Adapted from K. Grauman, B. Leibe

Agenda: Efficient Visual Search

1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

- KD-Trees
- Best Bin First - BBF
- Randomized KD-Trees

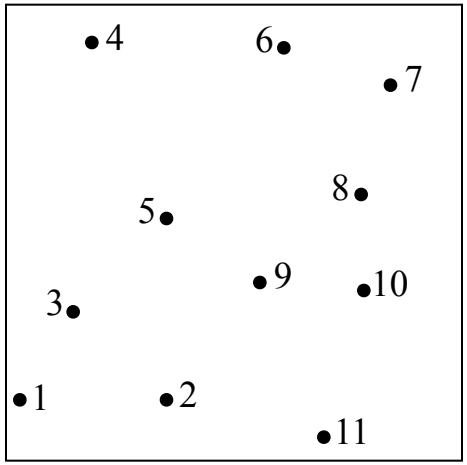
K-d tree

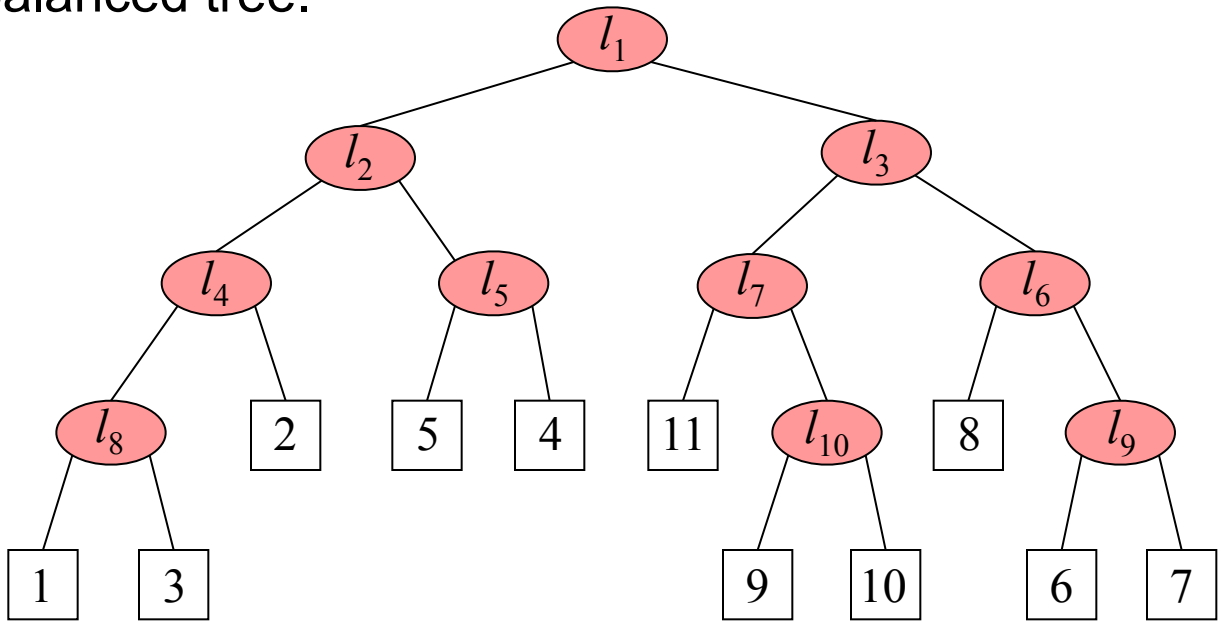
• K-d tree is a binary tree data structure for organizing a set of points in a K-dimensional space.

• Each internal node is associated with an axis aligned hyper-plane splitting its associated points into two sub-trees.

• Dimensions with high variance are chosen first.

• Position of the splitting hyper-plane is chosen as the mean/median of the projected points – balanced tree.

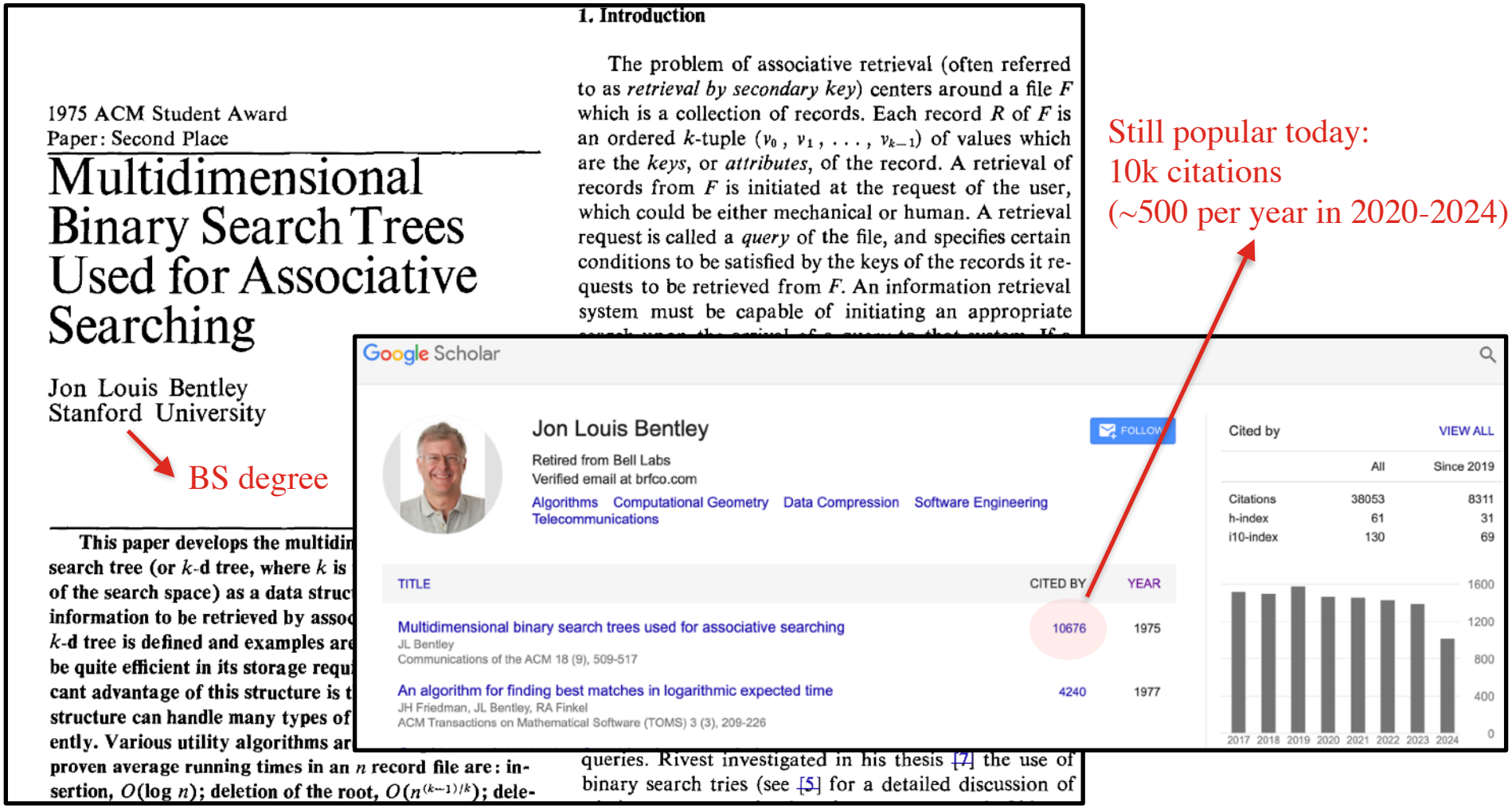




Multidimensional binary search trees used for associative searching, JL Bentley, 1975

Images: Anna Atramentov

K-d tree



1975 ACM Student Award
Paper: Second Place

Multidimensional Binary Search Trees Used for Associative Searching

Jon Louis Bentley
Stanford University

BS degree

1. Introduction

The problem of associative retrieval (often referred to as *retrieval by secondary key*) centers around a file F which is a collection of records. Each record R of F is an ordered k-tuple $(v_0, v_1, \ldots, v_{k-1})$ of values which are the *keys*, or *attributes*, of the record. A retrieval of records from F is initiated at the request of the user, which could be either mechanical or human. A retrieval request is called a *query* of the file, and specifies certain conditions to be satisfied by the keys of the records it requests to be retrieved from F. An information retrieval system must be capable of initiating an appropriate

This paper develops the multidi
search tree (or k-d tree, where k is
of the search space) as a data struc
information to be retrieved by assoc
k-d tree is defined and examples are
be quite efficient in its storage requi
cant advantage of this structure is t
structure can handle many types of
ently. Various utility algorithms ar
proven average running times in an n record file are: in-
sertion, $O(\log n)$; deletion of the root, $O(n^{(k-1)/k})$; dele-

queries. Rivest investigated in his thesis [7] the use of binary search tries (see [5] for a detailed discussion of

Google Scholar

Jon Louis Bentley
Retired from Bell Labs
Verified email at brfco.com
Algorithms Computational Geometry Data Compression Software Engineering Telecommunications

FOLLOW

TITLE	CITED BY	YEAR
Multidimensional binary search trees used for associative searching JL Bentley Communications of the ACM 18 (9), 509-517	10676	1975
An algorithm for finding best matches in logarithmic expected time JH Friedman, JL Bentley, RA Finkel ACM Transactions on Mathematical Software (TOMS) 3 (3), 209-226	4240	1977

Cited by VIEW ALL

	All	Since 2019
Citations	38053	8311
h-index	61	31
i10-index	130	69

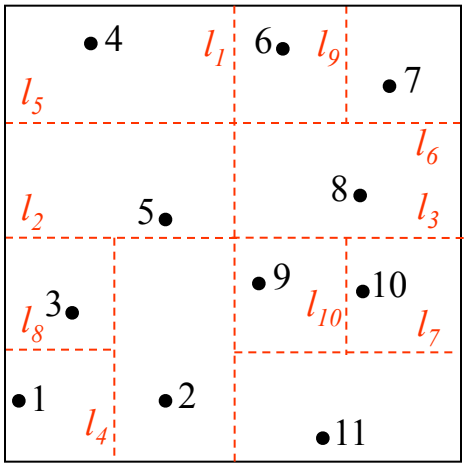
Still popular today:
10k citations
(~500 per year in 2020-2024)

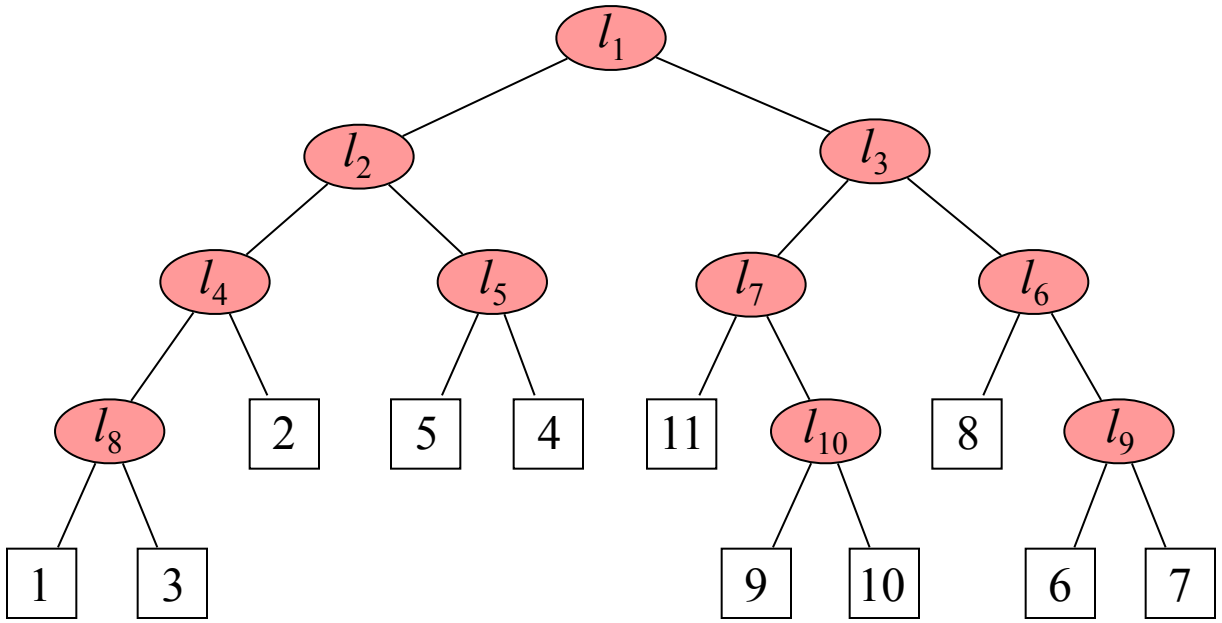
Multidimensional binary search trees used for associative searching, JL Bentley, 1975

Images: Anna Atramentov

K-d tree construction

Simple 2D example



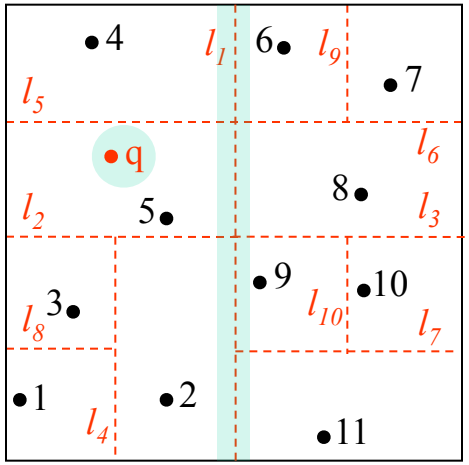


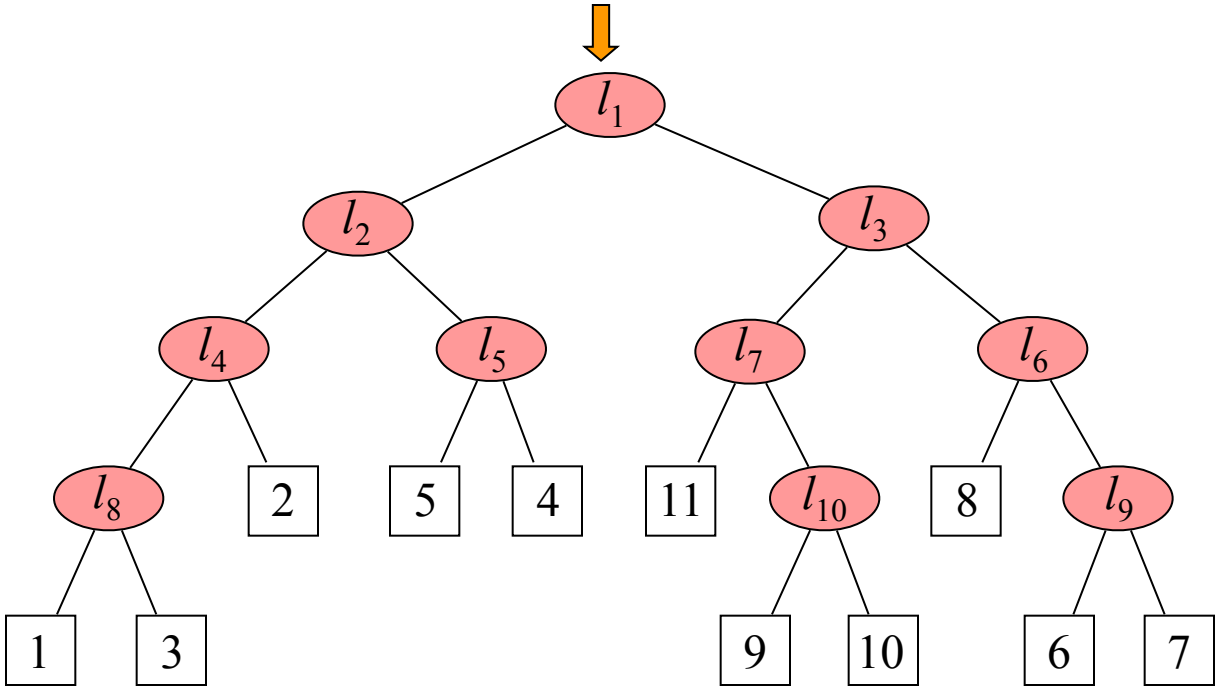
Slide credit: Anna Atramentov

KD-tree: Properties

- Binary tree of depth O(log(n))
- Total nodes: (2n -1) = n-1 nonleaf nodes + n leaves
- Construction time: O(n d log(n))
- Memory requirements:
  - nonleaf node: (dim, threshold)
  - Leaf node: data id
- Need to also store the original data

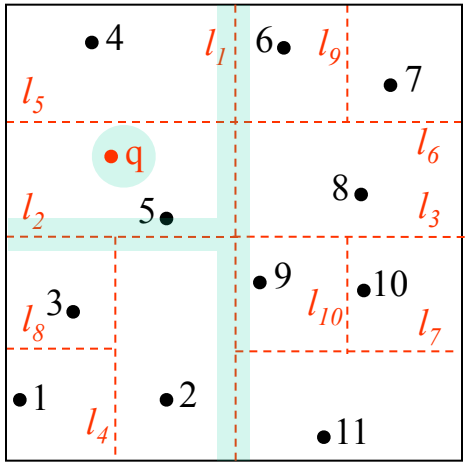
K-d tree query

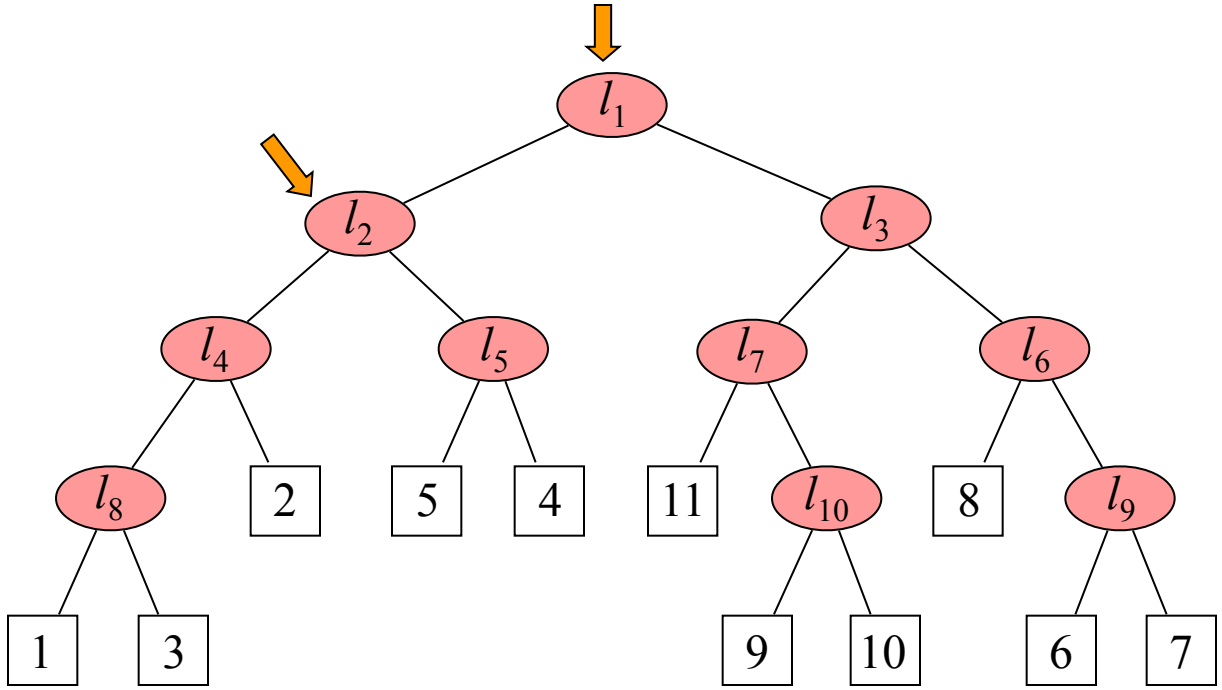




Slide credit: Anna Atramentov

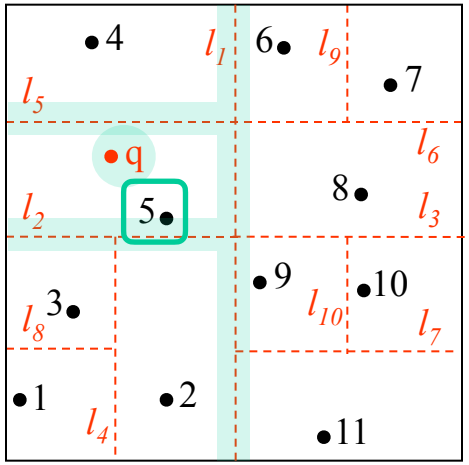
K-d tree query

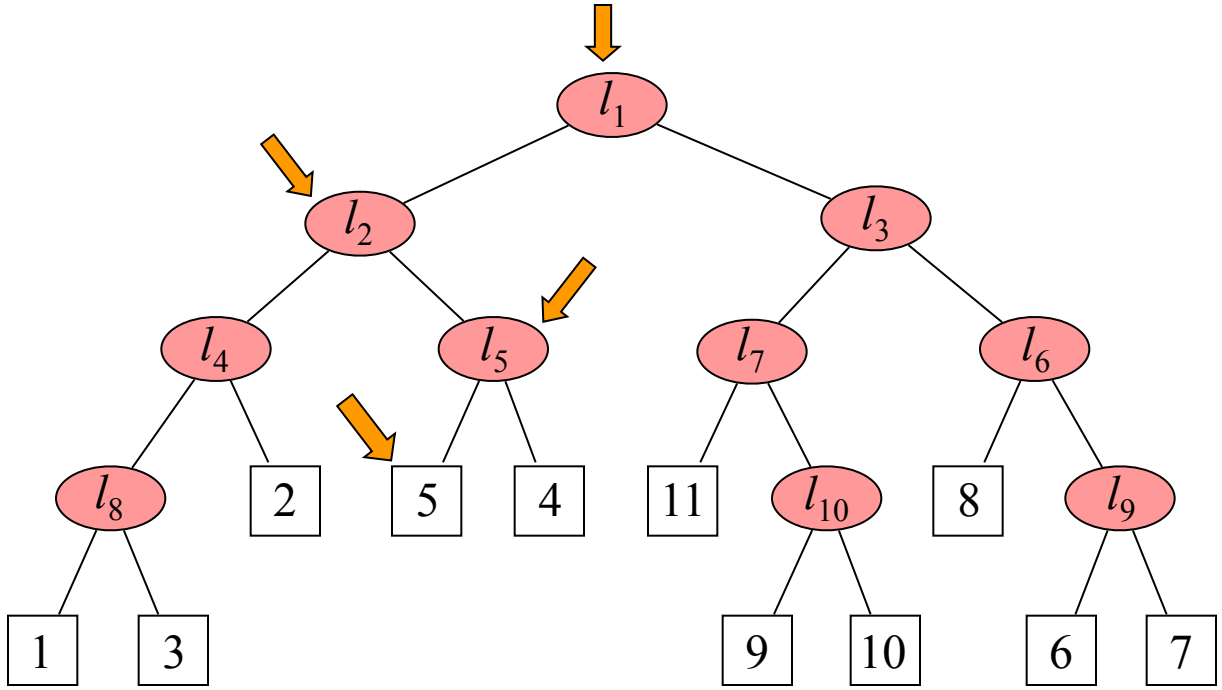




Slide credit: Anna Atramentov

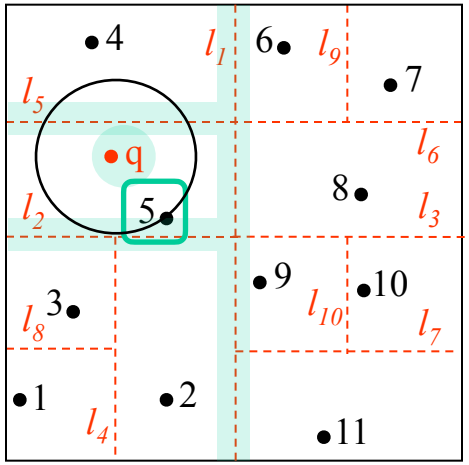
K-d tree query

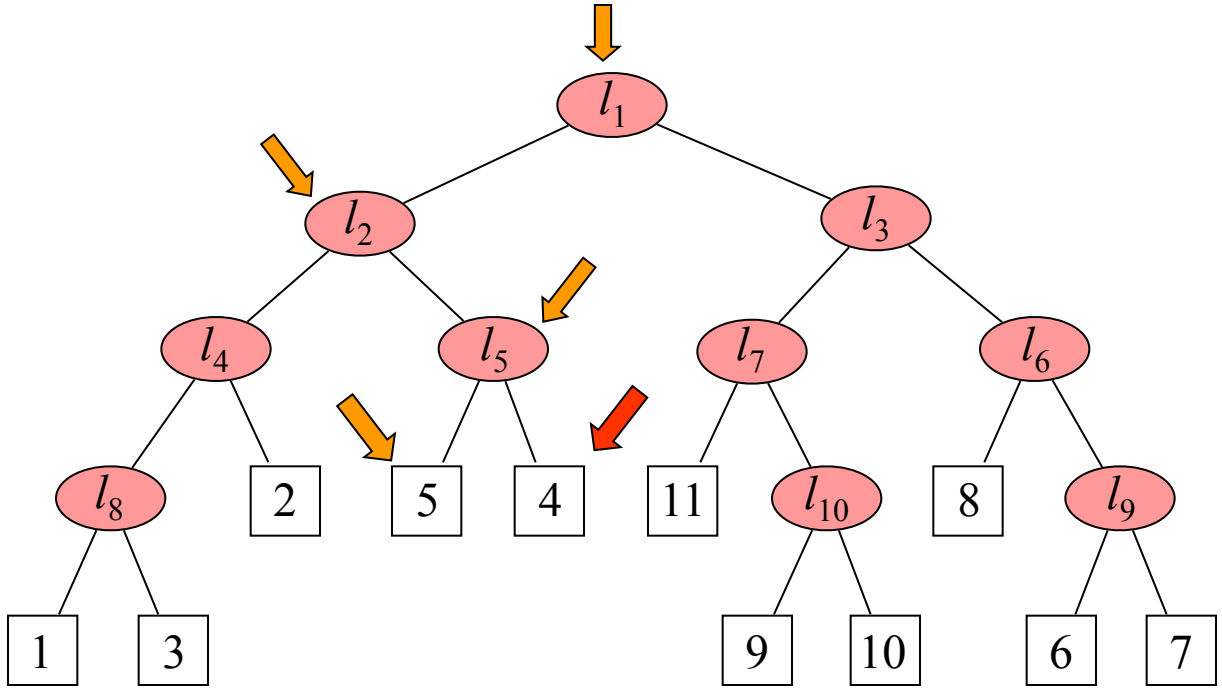




Slide credit: Anna Atramentov

K-d tree query



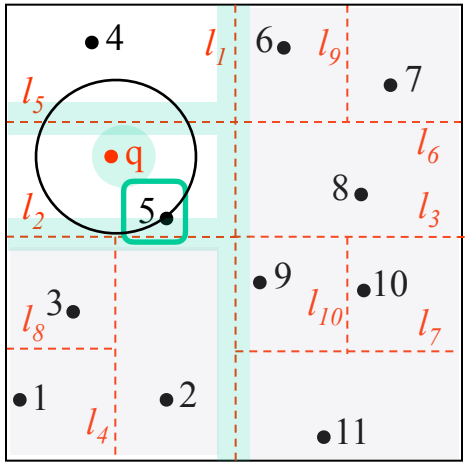


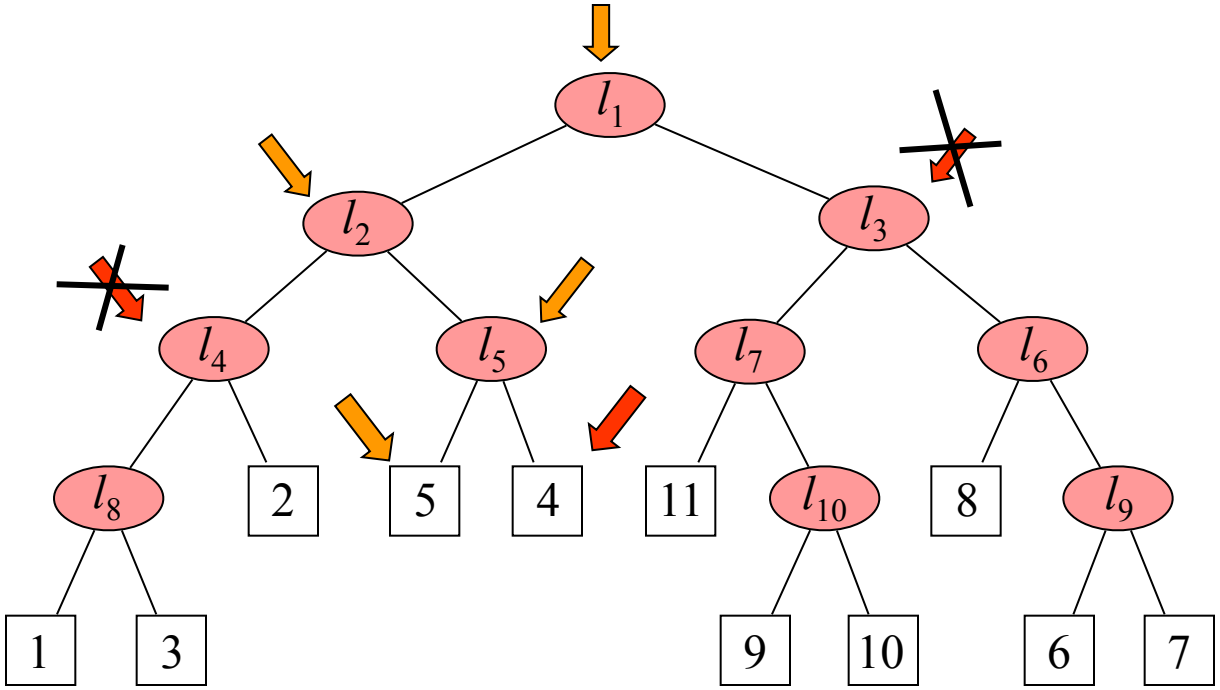
- Circle intersects with the right tree of l_5 so need to backtrack by comparing to 4.

Slide credit: Anna Atramentov

K-d tree query

Found nearest neighbor with 5 inspections for N=11 elements.



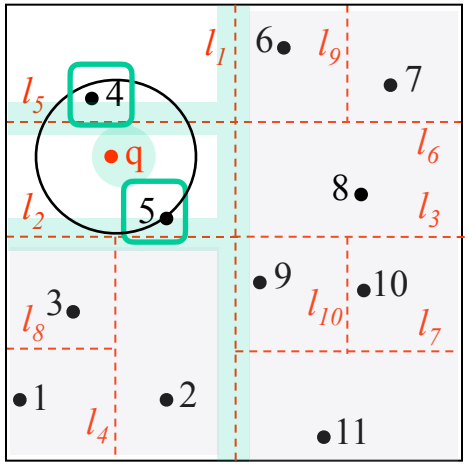


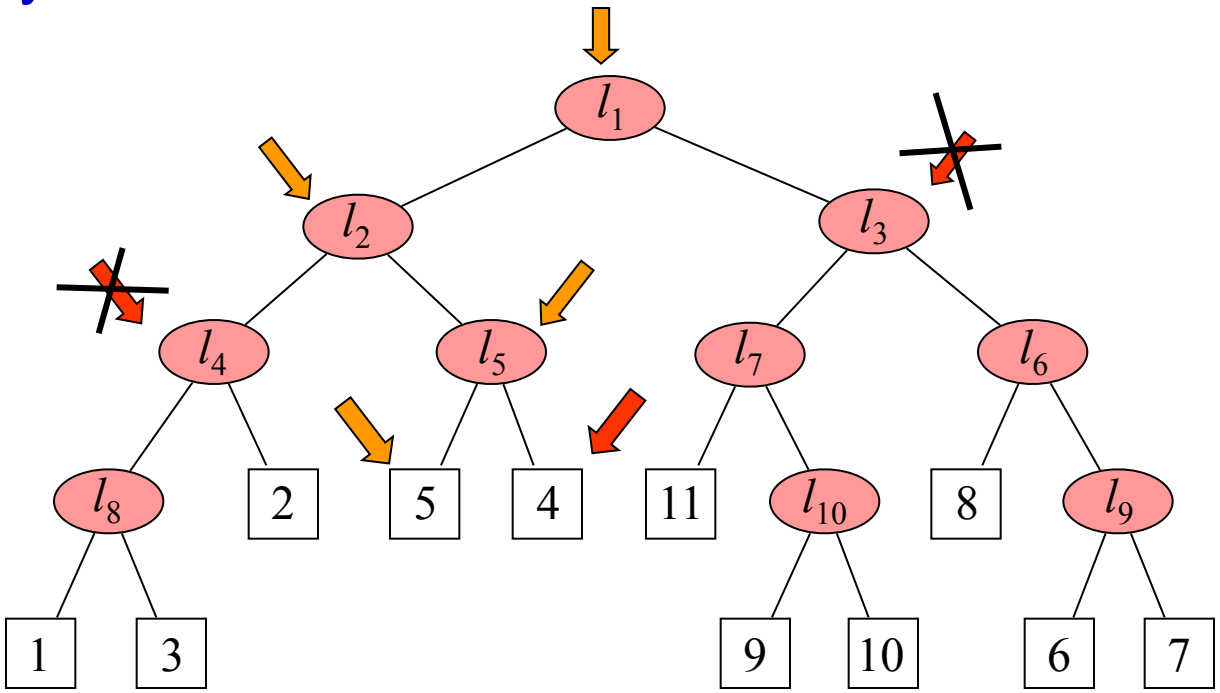
- Circle intersects with the right tree of l_5 so need to backtrack by comparing to 4.
- Circle doesn't intersect with gray subtrees, i.e., no need to inspect them.

Slide credit: Anna Atramentov

K-d tree: Backtracking

Backtracking is necessary as the true nearest neighbor may not lie in the query cell.





- Circle intersects with the right tree of l_5 so need to backtrack by comparing to 4.
- Circle doesn't intersect with gray subtrees, i.e., no need to inspect them.

Slide credit: Anna Atramentov

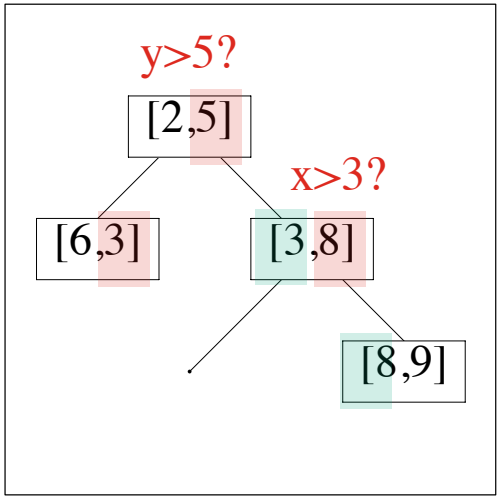


Figure 6.1

A 2d-tree of four elements. The splitting planes are not indicated. The [2,5] node splits along the y = 5 plane and the [3,8] node splits along the x = 3 plane.

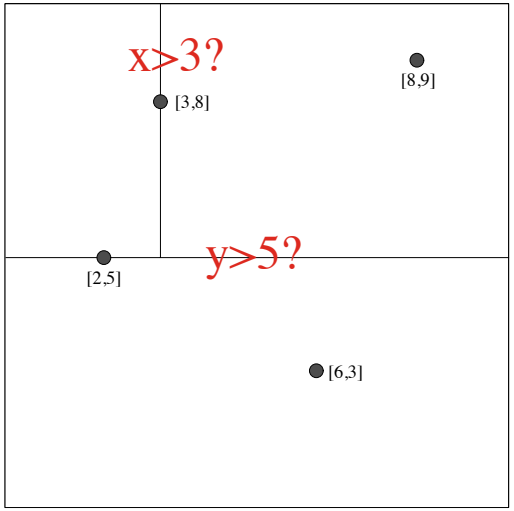


Figure 6.2

How the tree of Figure 6.1 splits up the x,y plane.

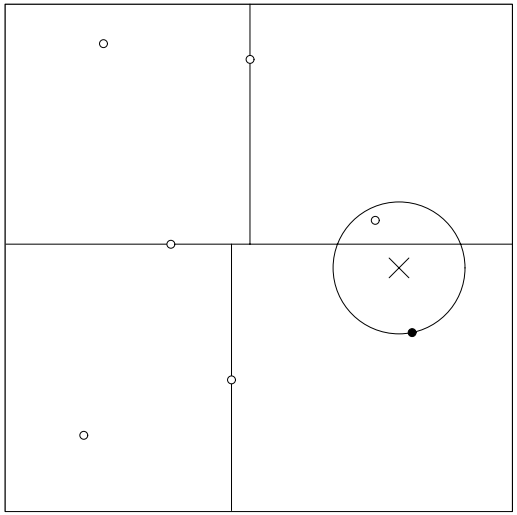


Figure 6.3

The black dot is the dot which owns the leaf node containing the target (the cross). Any nearer neighbour must lie inside this circle.

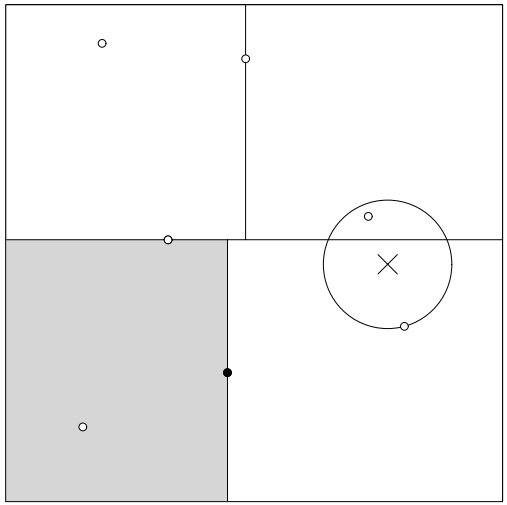


Figure 6.4

The black dot is the parent of the closest found so far. In this case the black dot's other child (shaded grey) need not be searched.

Figure: A. Moore

K-d tree: Backtracking

Backtracking is necessary as the true nearest neighbor may not lie in the query cell.

But in some cases, almost all cells need to be inspected.

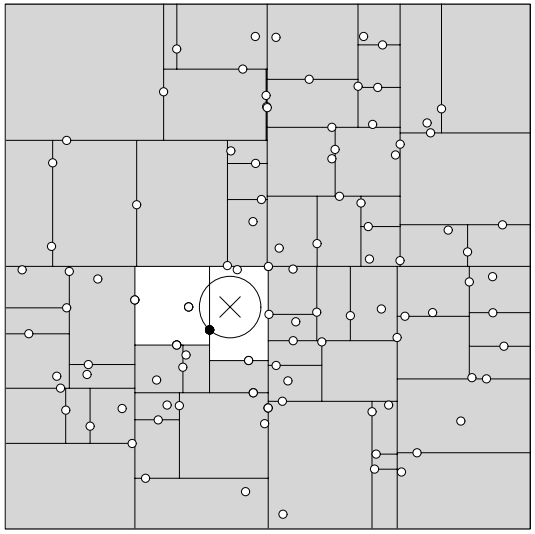


Figure 6.5

Generally during a nearest neighbour search only a few leaf nodes need to be inspected.

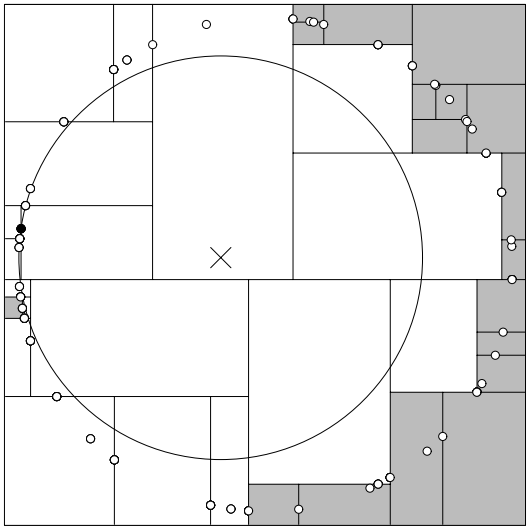


Figure 6.6

A bad distribution which forces almost all nodes to be inspected.

https://www.ri.cmu.edu/pub_files/pub1/moore_andrew_1991_1/moore_andrew_1991_1.pdf

Figure: A. Moore

- KD-Trees
- Best Bin First - BBF
- Randomized KD-Trees

Solution: Approximate nearest neighbor K-d tree

Best Bin First *[Beis & Lowe, CVPR 1997]*

In backtracking, the ***order of examining leaf nodes is according to the tree structure,*** *and does not take into account* ***the position of the query point.***

Simple idea:

- Search in bins in order of increasing distance from the query.

- Implemented using a priority queue
  - At each internal node visited, store (position, distance) in the queue

- Instead of backtracking, pop the closest from the queue and continue from it
  - Limit the number of neighbouring k-d tree bins to explore E_{max}:
  - only approximate NN is found

http://image.ntua.gr/iva/files/ann.pdf

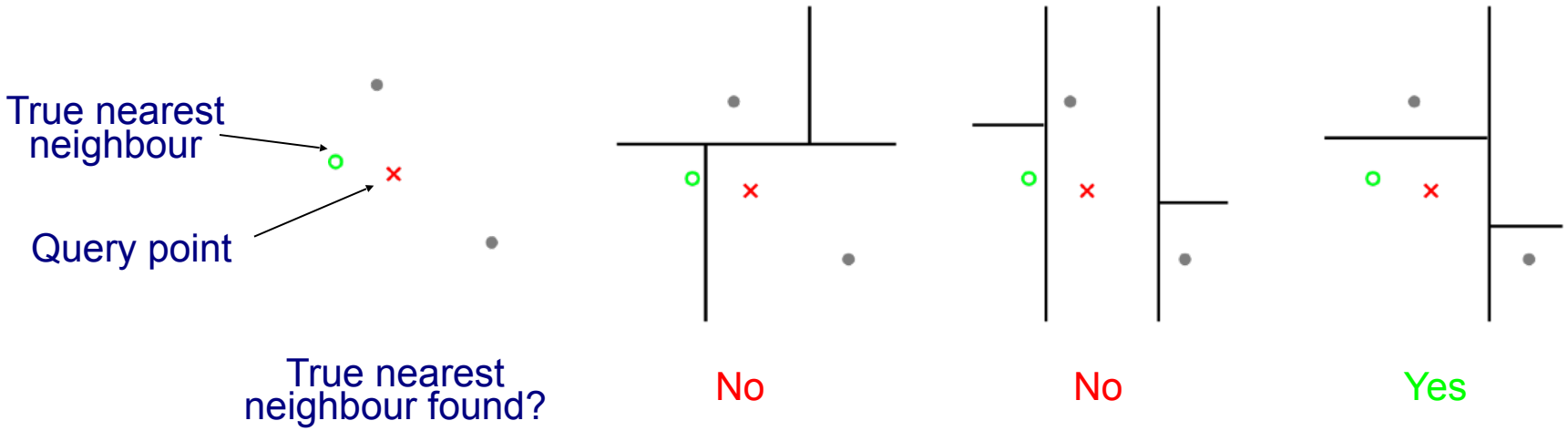
https://www.cs.utexas.edu/~grauman/courses/spring2007/395T/papers/beis97shape.pdf

- KD-Trees
- Best Bin First - BBF
- Randomized KD-Trees

Randomized K-d trees

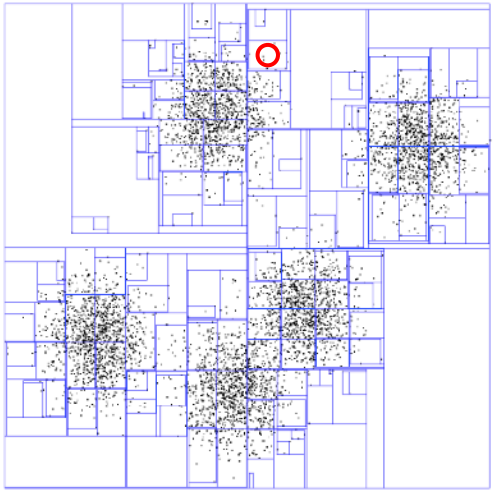
- How to choose the dimension to split and the splitting point?
  - Pick dimension with the highest variance
  - Split at the mean/median

- **Multiple randomized trees** increase the chances of finding nearby points

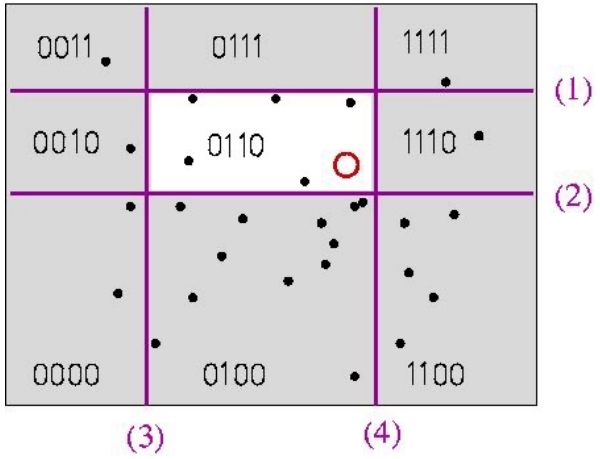


[Silpa-Anan and Hartley 2008, Philbin et al. 2007]

Approximate nearest neighbor search



- **Best-Bin First (BBF)**, a variant of **k-d trees** that uses priority queue to examine most promising branches first [Beis & Lowe, CVPR 1997]
- Extended to multiple **randomized trees** in [Muja & Lowe, 2009]



- **Locality-Sensitive Hashing (LSH)**, a randomized **hashing** technique using hash functions that map similar points to the same bin, with high probability [Indyk & Motwani, 1998]

Adapted from K. Grauman, B. Leibe

Agenda: Efficient Visual Search

1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

Locality Sensitive Hashing (LSH)

Idea: construct hash functions g: $R^d \rightarrow Z^k$ such that

for any points p,q:

If ||p-q|| ≤ r, then Pr[g(p)=g(q)] is "high" or "not-so-small"
If ||p-q|| > cr, then Pr[g(p)=g(q)] is "small"

Example of g: linear projections

$g(p)=<h_1(p),h_2(p),\ldots,h_k(p)>$, where $h_{X,b}(p)= \lfloor (p*X+b)/w \rfloor$

g(p)= binary vector of length k

X_i are randomly sampled from a Gaussian.
b is sampled from uniform distribution [0,w].
w is the width of each quantization bin.
$\lfloor . \rfloor$ is the "floor" operator.

[Datar-Immorlica-Indyk-Mirrokni'04]

Locality Sensitive Hashing (LSH) - Toy example

```
N = 10    # number of data points
d = 5     # data dimensionality
k = 2     # number of buckets at most 2^k
```

```
>>> proj
([[-0.03511321,  0.04873789],
  [ 0.41580021, -0.09016888],
  [ 0.20475851, -2.36594203],
  [ 0.51363479,  0.47604254],
  [-1.39742229,  1.05941025]])
```

```
data    = np.random.randn(N, d)        # [10x5]
proj    = np.random.randn(d, k)        # [5x2]
reduced = np.matmul(data, proj)        # [10x2]
                    [10x5] [5x2]
hashed  = (reduced > 0).astype(int)    # [10x2] binary
```

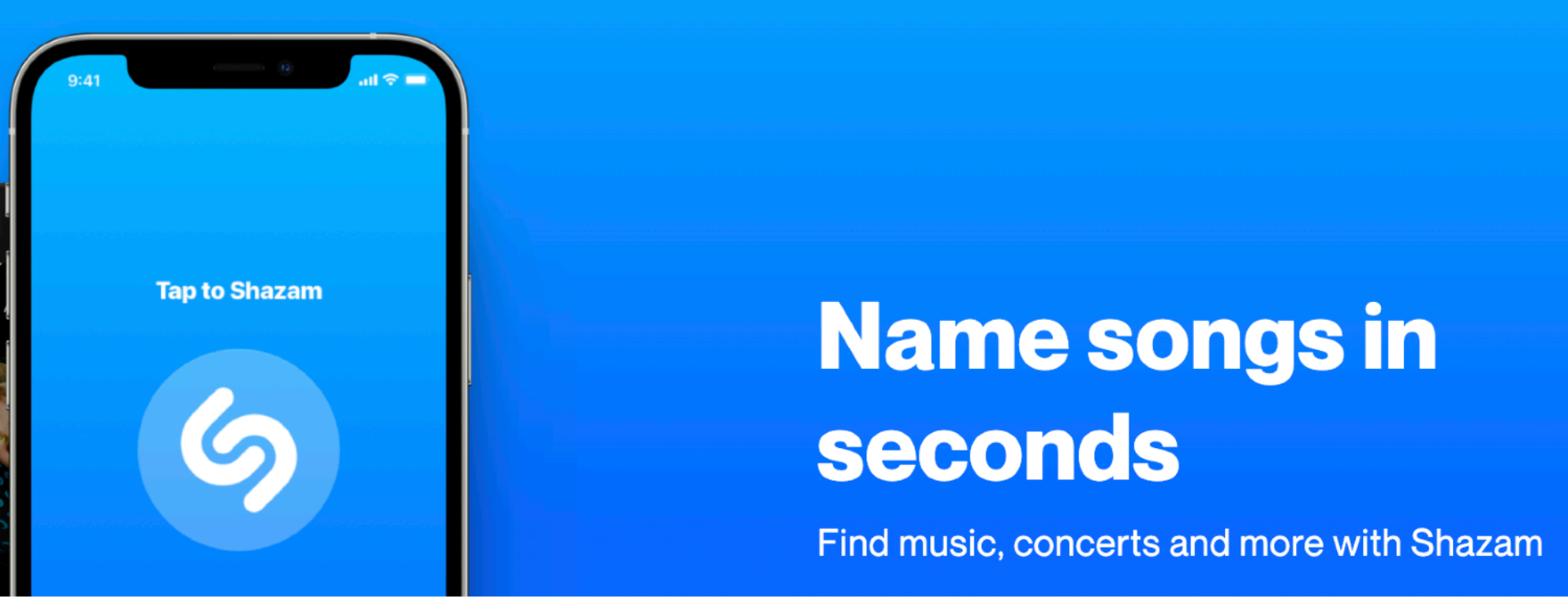
```
>>> data
([[-0.23996565,  0.5514444 , -0.42894206, -1.95205512,  1.59988299],
 [-1.16126365, -0.85375022,  0.55729351,  1.27678351,  2.40379971],
 [-1.21240628, -0.94951025,  0.26493223,  0.44050331, -0.66722836],
 [-0.74552833, -0.25266102,  0.57486544, -0.40378095,  0.43612001],
 [-1.29000346,  1.3974408 ,  2.00283136, -0.7696424 ,  2.08364683],
 [ 0.1193746 ,  0.22652649, -0.21421195,  1.16216535,  0.42436214],
 [ 0.91020221,  0.47788041,  0.43375582, -0.17679428, -0.98689477],
 [ 1.24229921,  0.04827515,  0.86778847, -1.08160739,  1.19754493],
 [ 2.46062593,  0.13836055, -1.43365938, -1.94534837, -1.96558005],
 [ 1.41804401, -0.77976776,  1.01688406,  0.31024282, -0.86179706]])
```

```
>>> reduced
([[-3.08846845,  1.71910468],
 [-2.90342611,  1.85627334],
 [ 0.86066964, -1.09745839],
 [-0.77800978, -1.10383854],
 [-2.27060054, -3.08640633],
 [ 0.05005186,  1.49501923],
 [ 1.54385891, -2.15465776],
 [-2.07488838, -1.24314285],
 [ 1.42512269,  0.49098067],
 [ 1.19784192, -3.03177334]])
```

```
>>> hashed
([[0, 1],
 [0, 1],
 [1, 0],
 [0, 0],
 [0, 0],
 [1, 1],
 [1, 0],
 [0, 0],
 [1, 1],
 [1, 0]])
```

same bucket

https://jaketae.github.io/study/lsh/



… capable of quickly identifying a short segment of music … out of a database of over a million tracks. The algorithm uses a combinatorially hashed time-frequency constellation analysis of the audio …

https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

REFORMER: THE EFFICIENT TRANSFORMER

Nikita Kitaev*
U.C. Berkeley & Google Research
kitaev@cs.berkeley.edu

Łukasz Kaiser*
Google Research

Anselm Levskaya
Google Research
{lukaszkaiser,levskaya}@google.com

ABSTRACT

Large Transformer models routinely achieve state-of-the-art results on a number of tasks but training these models can be prohibitively costly, especially on long sequences. We introduce two techniques to improve the efficiency of Transformers. For one, we replace dot-product attention by one that uses locality-sensitive hashing, changing its complexity from O(L^2) to O($L \log L$), where L is the length of the sequence. Furthermore, we use reversible residual layers instead of the standard residuals, which allows storing activations only once in the training process instead of N times, where N is the number of layers. The resulting model, the Reformer, performs on par with Transformer models while being much more memory-efficient and much faster on long sequences.

LSH: discussion

In theory, query time is O(kL), where k is the number of projections and L is the number of hash tables, i.e., independent of the number of points, N.

In practice, LSH has high memory requirements as large number of projections/ hash tables are needed.

Code and more materials available online:
http://www.mit.edu/~andoni/LSH/

Hashing functions could be also data-dependent (PCA) or learnt from labeled point pairs (close/far).

Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS, 2008.*
R. Salakhutdinov and G. Hinton, "Semantic Hashing," ACM SIGIR, 2007.

See also:
http://cobweb.ecn.purdue.edu/~malcolm/yahoo Slaney2008(LSHTutorialDraft).pdf
http://www.sanjivk.com/EECS6898/ApproxNearestNeighbors_2.pdf

Approximate nearest neighbour search (references)

J. L. Bentley. Multidimensional binary search trees used for associative searching. Comm. ACM, 18(9), 1975.

Freidman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw., 3:209–226, 1977.*

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM, 45:891–923, 1998.*

C. Silpa-Anan and R. Hartley. Optimised KD-trees for fast image descriptor matching. In CVPR, 2008.

M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In VISAPP, 2009.

P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proc. of 30th ACM Symposium on Theory of Computing, 1998*

G. Shakhnarovich, P. Viola, and T. Darrell, “Fast pose estimation with parameter-sensitive hashing,” in *Proc. of the IEEE International Conference on Computer Vision, 2003.*

R. Salakhutdinov and G. Hinton, “Semantic Hashing,” ACM SIGIR, 2007.

Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *NIPS, 2008.*

ANN - search (references continued)

O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. BMVC., 2008.

B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," *Proc. of the IEEE International Conference on Computer Vision, 2009.*

J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," *in CVPR, 2010.*

H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 2011.

A.Gordo and F.Perronnin. Asymmetric distances for binary embeddings. CVPR, 2011.

Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. CVPR, *2011.*

A. Babenko and V. Lempitsky. The inverted multi-index. CVPR, 2012.

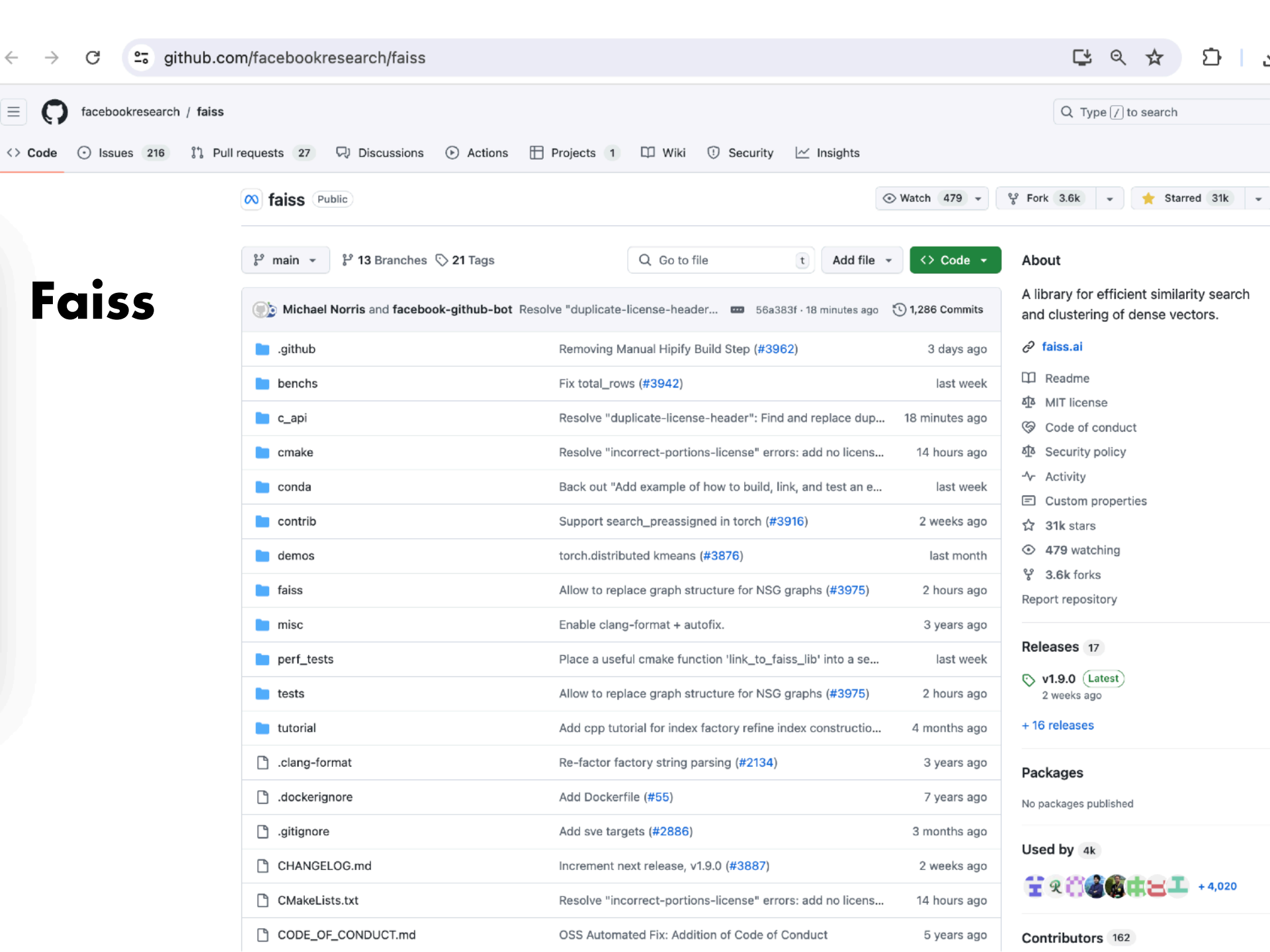
T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. CVPR, 2013.

T. Norouzi and D. Fleet, Cartesian k-means., CVPR, 2013

See tutorial at CVPR'13 by H. Jegou: https://sites.google.com/site/lsvr13

Code: https://github.com/facebookresearch/faiss

Faiss


github.com/facebookresearch/faiss
facebookresearch / faiss
Type / to search
Code
Issues 216
Pull requests 27
Discussions
Actions
Projects 1
Wiki
Security
Insights
faiss Public
Watch 479
Fork 3.6k
Starred 31k
main
13 Branches
21 Tags
Go to file t
Add file
Code
Michael Norris and facebook-github-bot Resolve "duplicate-license-header... 56a383f · 18 minutes ago 1,286 Commits
.github Removing Manual Hipify Build Step (#3962) 3 days ago
benchs Fix total_rows (#3942) last week
c_api Resolve "duplicate-license-header": Find and replace dup... 18 minutes ago
cmake Resolve "incorrect-portions-license" errors: add no licens... 14 hours ago
conda Back out "Add example of how to build, link, and test an e... last week
contrib Support search_preassigned in torch (#3916) 2 weeks ago
demos torch.distributed kmeans (#3876) last month
faiss Allow to replace graph structure for NSG graphs (#3975) 2 hours ago
misc Enable clang-format + autofix. 3 years ago
perf_tests Place a useful cmake function 'link_to_faiss_lib' into a se... last week
tests Allow to replace graph structure for NSG graphs (#3975) 2 hours ago
tutorial Add cpp tutorial for index factory refine index constructio... 4 months ago
.clang-format Re-factor factory string parsing (#2134) 3 years ago
.dockerignore Add Dockerfile (#55) 7 years ago
.gitignore Add sve targets (#2886) 3 months ago
CHANGELOG.md Increment next release, v1.9.0 (#3887) 2 weeks ago
CMakeLists.txt Resolve "incorrect-portions-license" errors: add no licens... 14 hours ago
CODE_OF_CONDUCT.md OSS Automated Fix: Addition of Code of Conduct 5 years ago
About
A library for efficient similarity search and clustering of dense vectors.
faiss.ai
Readme
MIT license
Code of conduct
Security policy
Activity
Custom properties
31k stars
479 watching
3.6k forks
Report repository
Releases 17
v1.9.0 Latest
2 weeks ago
+ 16 releases
Packages
No packages published
Used by 4k
+ 4,020
Contributors 162

Faiss

github.com/facebookresearch/faiss/wiki

Research foundations of Faiss

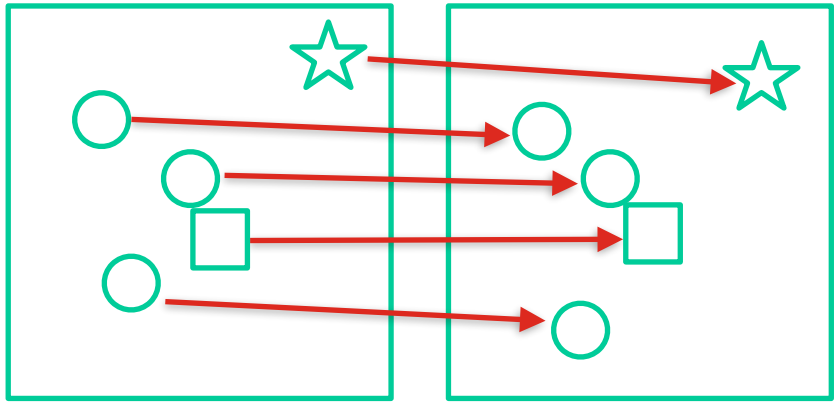
Faiss is based on years of research. Most notably it implements:

- The inverted file from "Video google: A text retrieval approach to object matching in videos.", Sivic & Zisserman, ICCV 2003. This is the key to non-exhaustive search in large datasets. Otherwise all searches would need to scan all elements in the index, which is prohibitive even if the operation to apply for each element is fast
- The product quantization (PQ) method from "Product quantization for nearest neighbor search", Jégou & al., PAMI 2011. This can be seen as a lossy compression technique for high-dimensional vectors, that allows relatively accurate reconstructions and distance computations in the compressed domain.
- The three-level quantization (IVFADC-R aka `IndexIVFPQR`) method from "Searching in one billion vectors: re-rank with source coding", Tavenard & al., ICASSP'11.
- The inverted multi-index from "The inverted multi-index", Babenko & Lempitsky, CVPR 2012. This method greatly improves the speed of inverted indexing for fast/less accurate operating points.
- The optimized PQ from "Optimized product quantization", He & al, CVPR 2013. This method can be seen as a linear transformation of the vector space to make it more amenable for indexing with a product quantizer.
- The pre-filtering of product quantizer distances from "Polysemous codes", Douze & al., ECCV 2016. This technique performs a binary filtering stage before computing PQ distances.
- The GPU implementation and fast k-selection is described in "Billion-scale similarity search with GPUs", Johnson & al, ArXiv 1702.08734, 2017
- The HNSW indexing method from "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs", Malkov & al., ArXiv 1603.09320, 2016
- The in-register vector comparisons from "Quicker ADC : Unlocking the Hidden Potential of Product Quantization with SIMD", André et al, PAMI'19, also used in "Accelerating Large-Scale Inference with Anisotropic Vector Quantization", Guo, Sun et al, ICML'20.
- The binary multi-index hashing method from "Fast Search in Hamming Space with Multi-Index Hashing", Norouzi et al, CVPR'12.
- The graph-based indexing method NSG from "Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph", Cong Fu et al, VLDB 2019.
- The Local search quantization method from "Revisiting additive quantization", Julieta Martinez, et al. ECCV 2016 and "LSQ++: Lower running time and higher recall in multi-codebook quantization", Julieta Martinez, et al. ECCV 2018.
- The residual quantizer implementation from "Improved Residual Vector Quantization for High-dimensional Approximate Nearest Neighbor Search", Shicong Liu et al, AAAI'15.

A general paper about product quantization and related methods: "A Survey of Product Quantization", Yusuke Matsui, Yusuke Uchida, Hervé Jégou, Shin'ichi Satoh, ITE transactions on MTA, 2018.

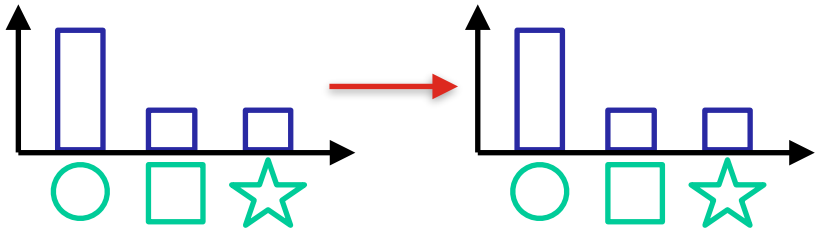
Recap: Two strategies

1. Efficient approximate nearest neighbor search on local feature descriptors.



2. Quantize descriptors into a “visual vocabulary” and use efficient techniques from text retrieval.

   (Bag-of-words representation)



Agenda: Efficient Visual Search

1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

So far …

- Linear exhaustive search can be prohibitively expensive for large image collections

- Answer (so far): approximate NN search methods
  - Randomized KD-trees
  - Locality sensitive hashing

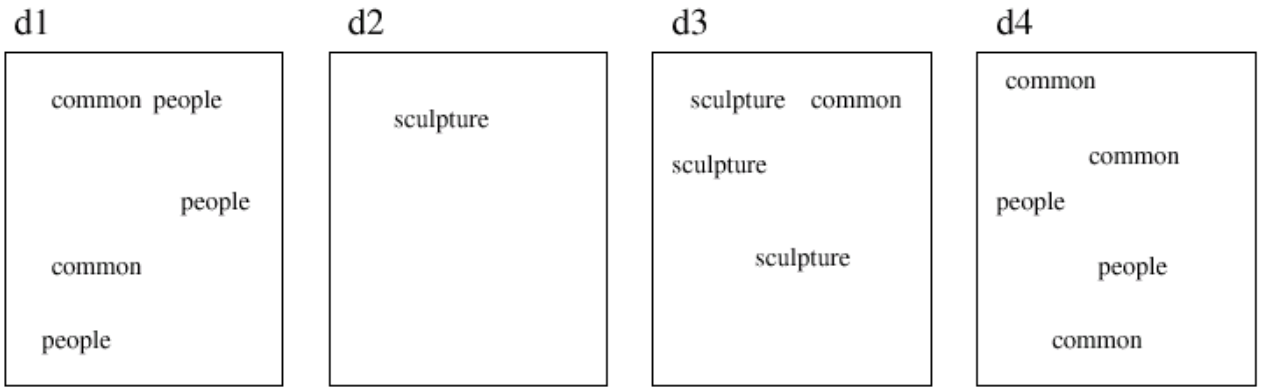
- However, memory footprint can be still high.

  Example: N = 10^7 images, 10^{10} SIFT features with 128B per feature ⟹ 1TB of memory

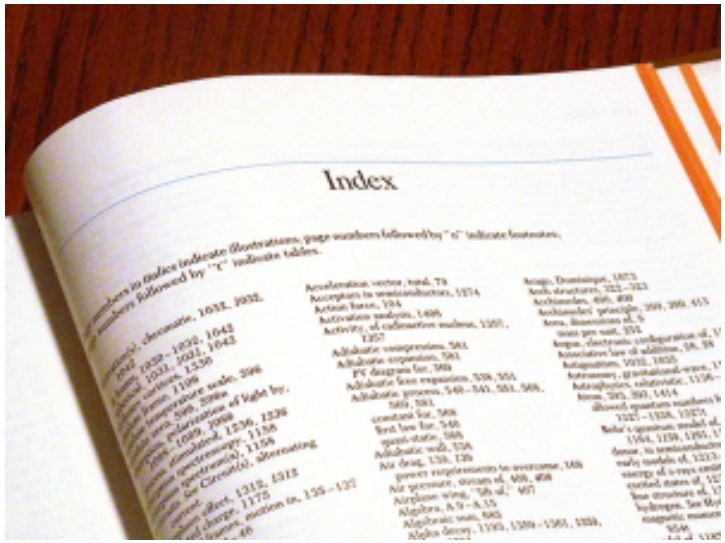
  Look how text-based search engines (Google) index documents – **inverted files**.

Indexing text with inverted files

Document collection:



Inverted file:



Term	**List of hits** (occurrences in documents)
People	[d1:hit hit hit], [d4:hit hit] …
Common	[d1:hit hit], [d3: hit], [d4: hit hit hit] …
Sculpture	[d2:hit], [d3: hit hit hit] …

Need to map feature descriptors to “visual words”.

Singhal, who had overseen the Google search engine for nearly a decade. Ng gave him the same pitch he gave Larry Page, except he focused squarely on the search engine, the jewel in the company's crown. As successful as it had been over the years, becoming the world's primary gateway to the Internet, Google Search answered queries in a simple way: It responded to keywords. If you searched on five words and then scrambled them and searched again, you would probably get the same results each time. But Ng told Singhal that deep learning could improve his search engine in ways that would never be possible without it. By analyzing millions of Google searches, looking for patterns in what people clicked on and what they didn't, a neural network could learn to deliver something much closer to what people were actually looking for. "People could ask real questions, not just type in keywords," Ng said.

Singhal wasn't interested. "People don't want to ask questions. They want to type in keywords," he said. "If I tell them to ask questions, they'll just be confused." Even if he had wanted to move beyond keywords, he was fundamentally opposed to the idea of a system that learned behavior on such a large scale. A neural network was a "black box." When it made a decision, like choosing a

Genius Makers, Cade Metz 2021.

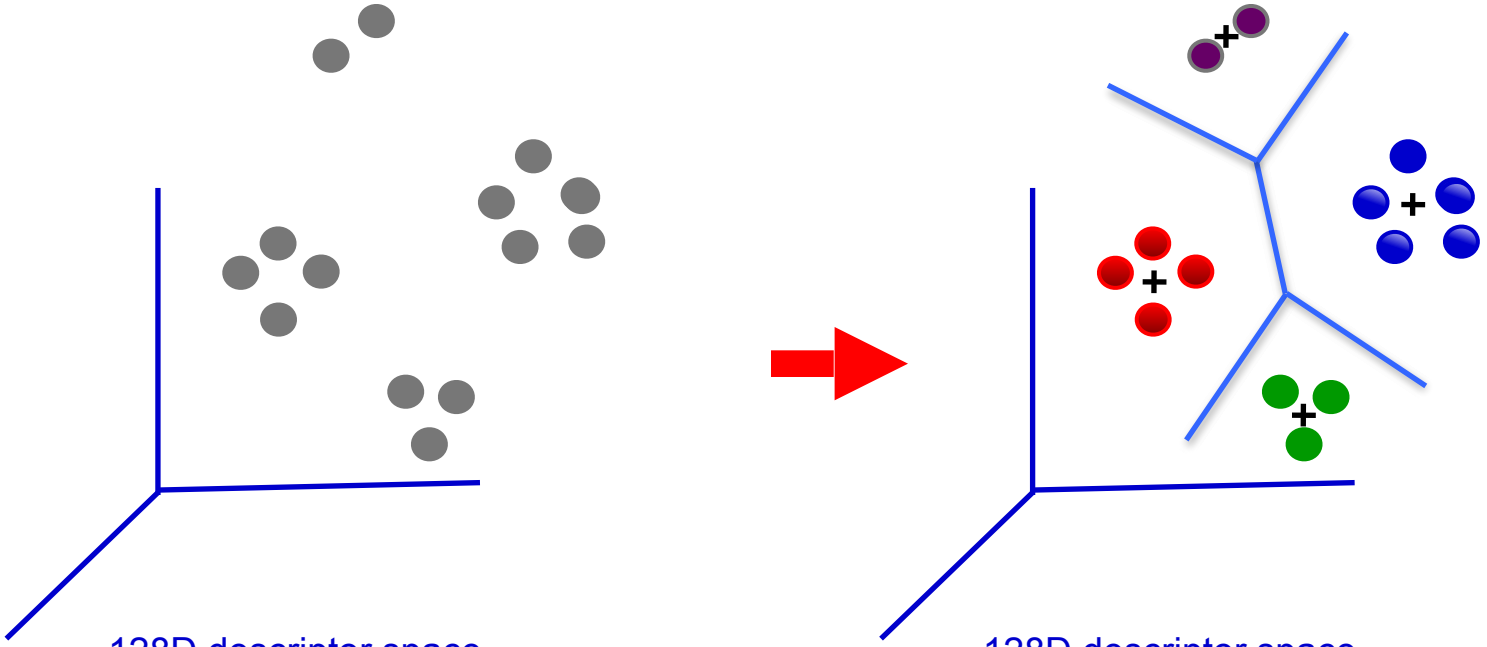
error. After a decade spent running Google Search, Singhal didn't want to lose control over the way his search engine operated. When he and his fellow engineers made changes to their search engine, they knew exactly what they were changing, and they could explain these changes to anyone who asked. That would not be the case with a neural network. Singhal's message was unequivocal. "I don't want to talk to you," he said.

Ng also met with the heads of Google's image search and video search services, and they turned him down, too. He didn't really find a collaborator until he and Jeff Dean walked into the same microkitchen, the very Googley term for the communal spaces spread across its campus where its employees could find snacks, drinks, utensils, microwave ovens, and maybe even a little conversation. Dean was a Google legend.

Build a visual vocabulary

Vector quantize descriptors

- Compute SIFT features from a subset of images
- K-means clustering (need to choose K)

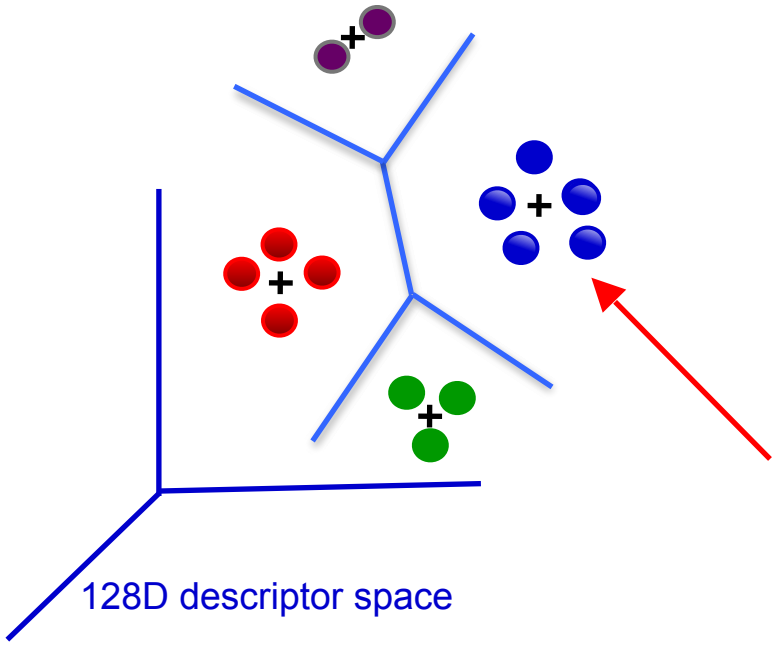


http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html

[Sivic and Zisserman, ICCV 2003]

Visual words

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



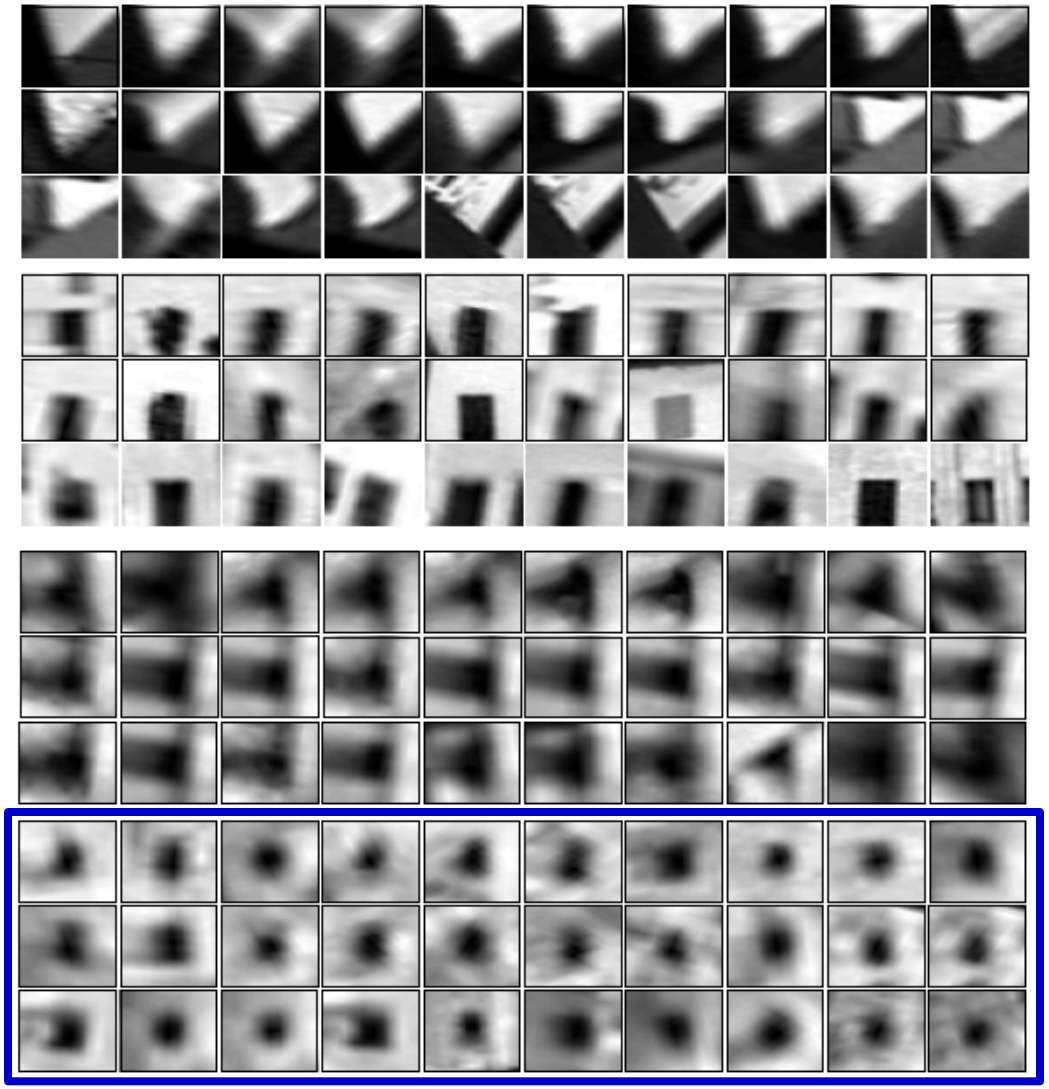
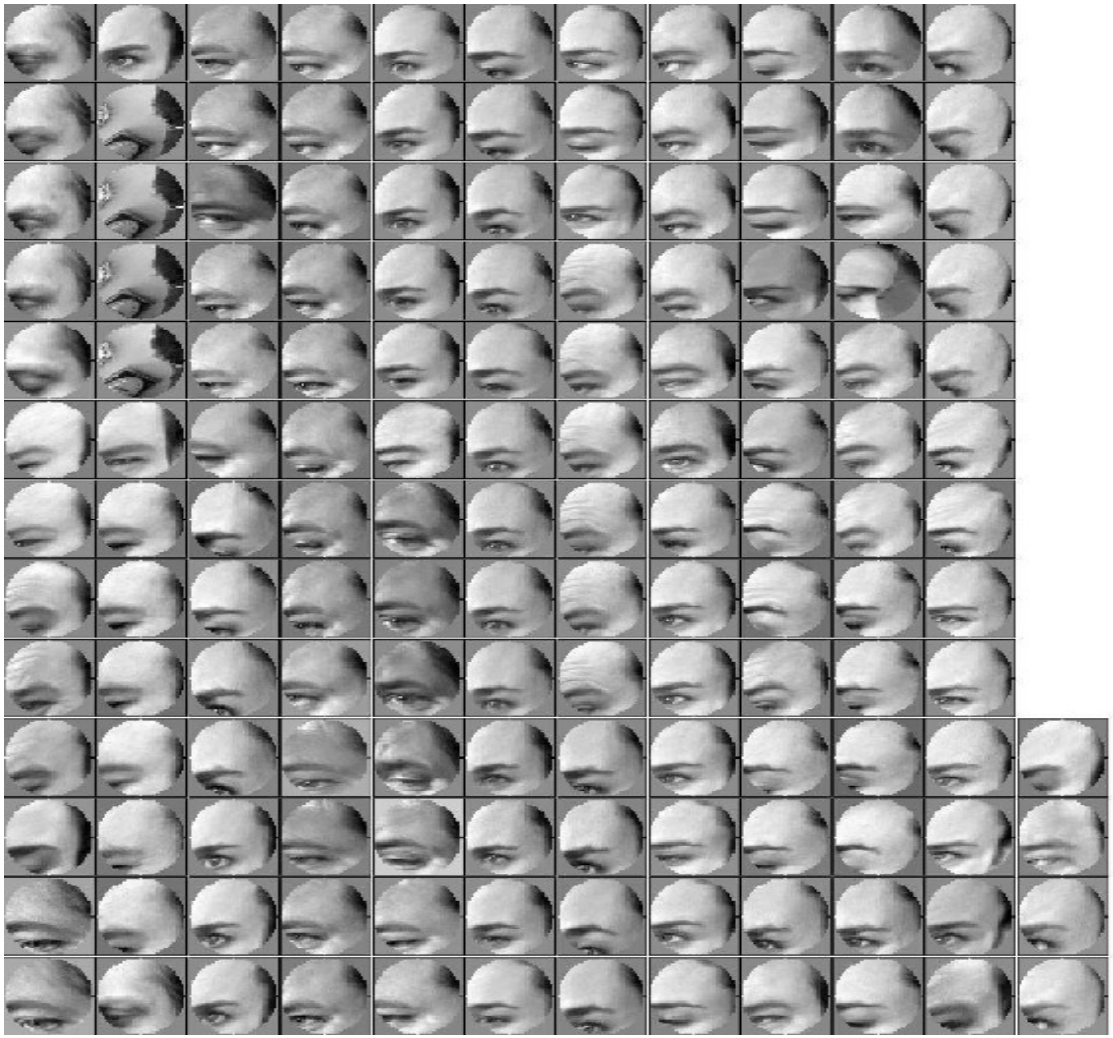


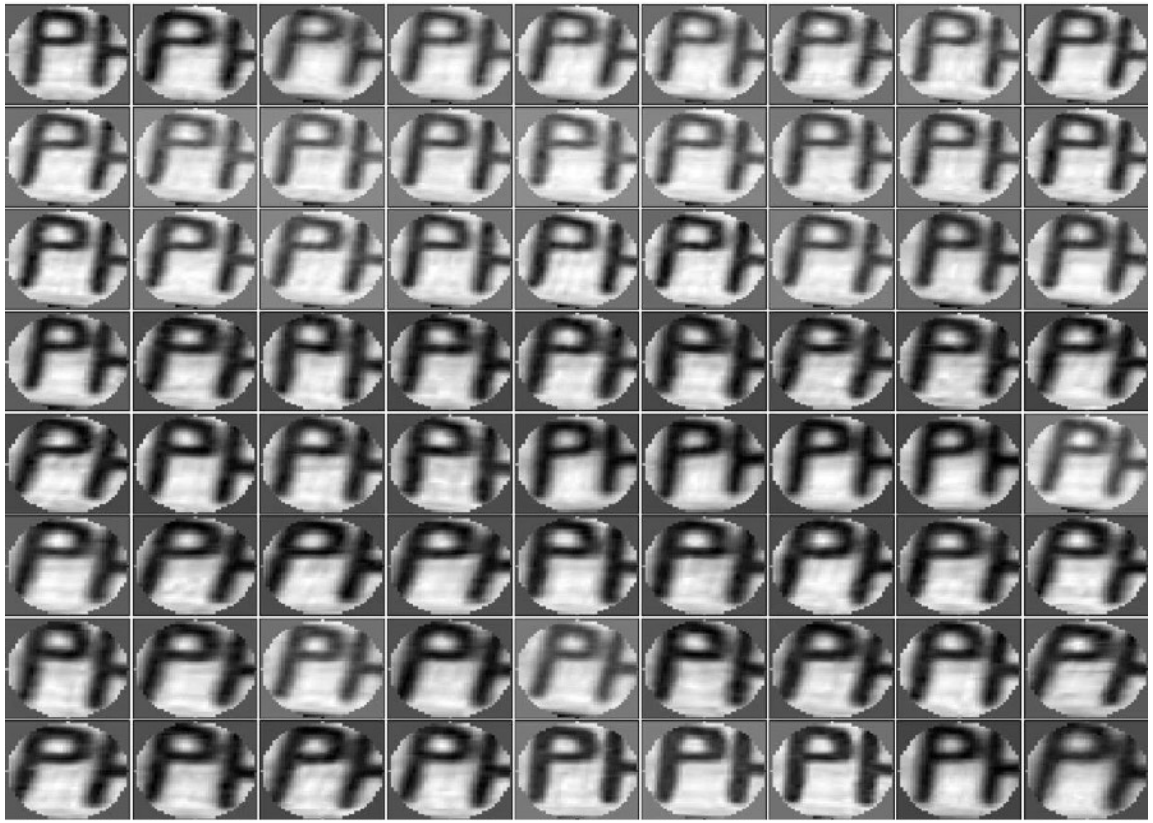
Figure from Sivic & Zisserman, ICCV 2003

Samples of visual words (clusters on SIFT descriptors):



More specific example

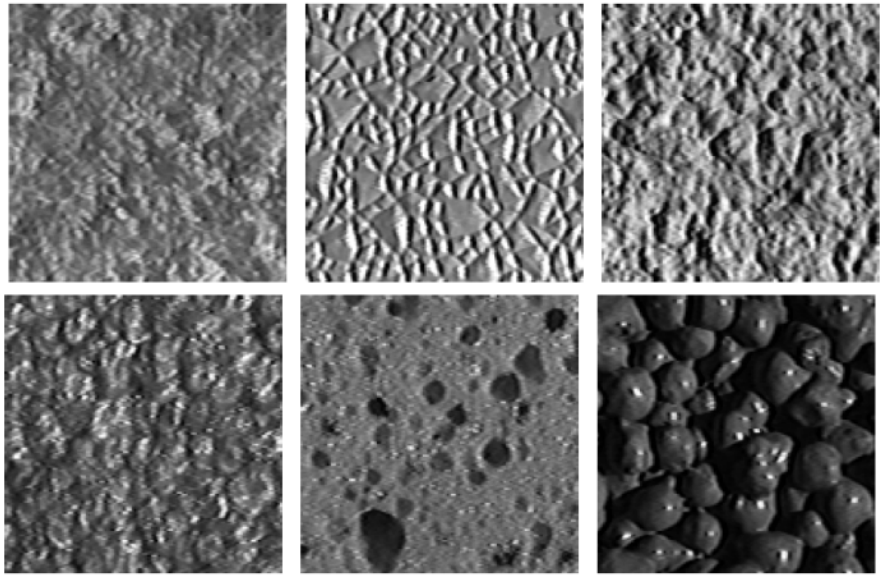
Samples of visual words (clusters on SIFT descriptors):



More specific example

Visual words

- First explored for texture and material representations
- *Texton* = cluster center of filter responses over collection of images
- Describe textures and materials based on distribution of prototypical texture elements.



Leung & Malik 1999; Varma & Zisserman, 2002; Lazebnik, Schmid & Ponce, 2003;

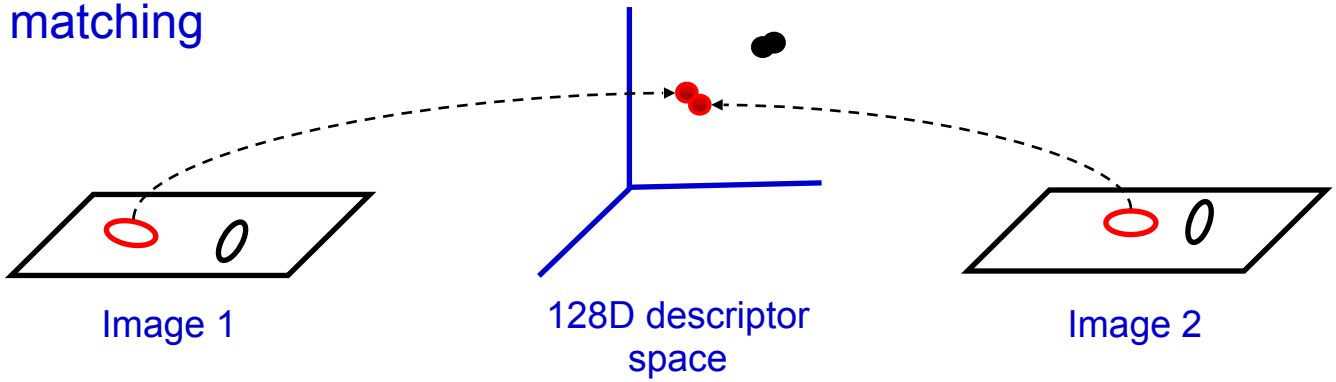
Slide: Grauman&Leibe

Visual words: quantize descriptor space

Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames

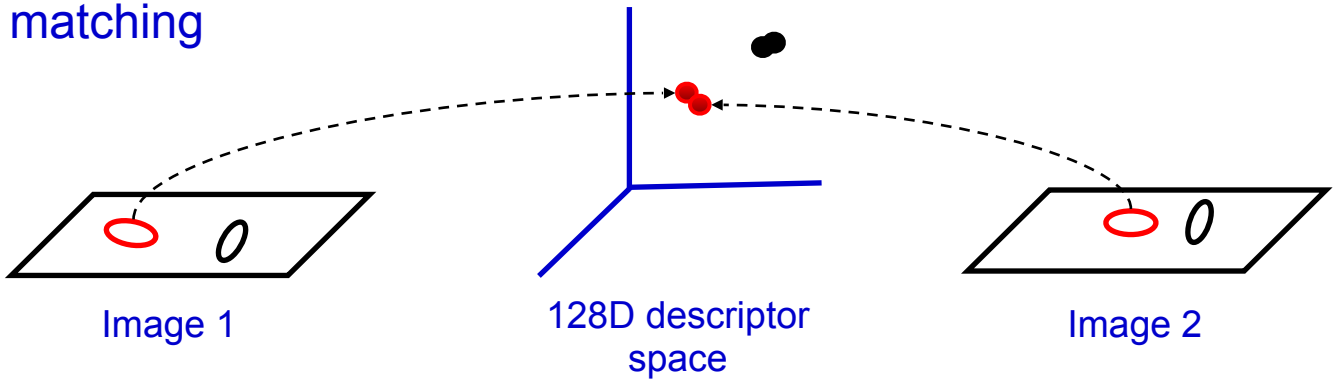


Visual words: quantize descriptor space

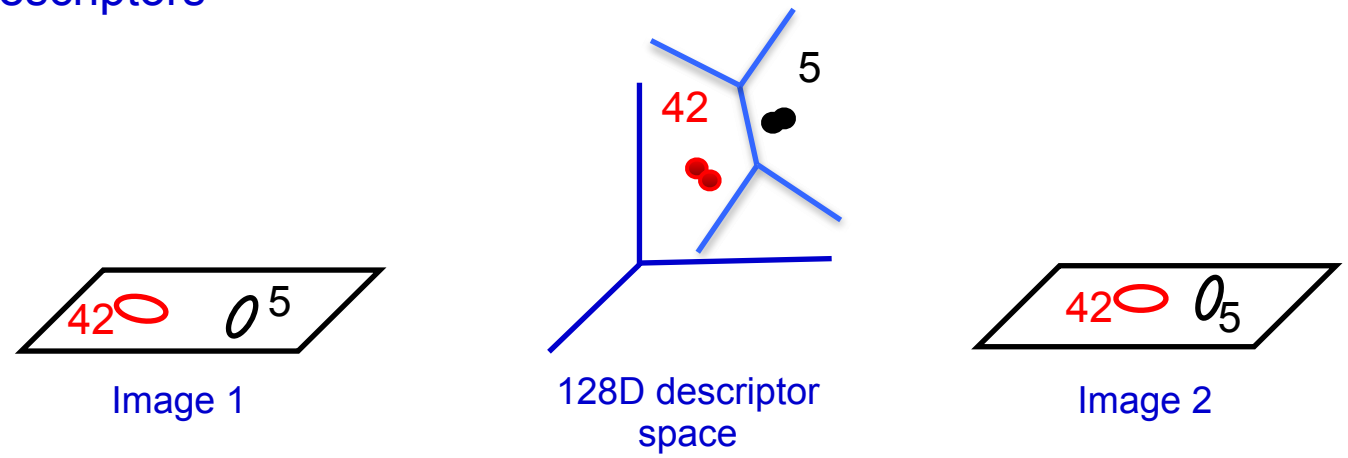
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames



Vector quantize descriptors

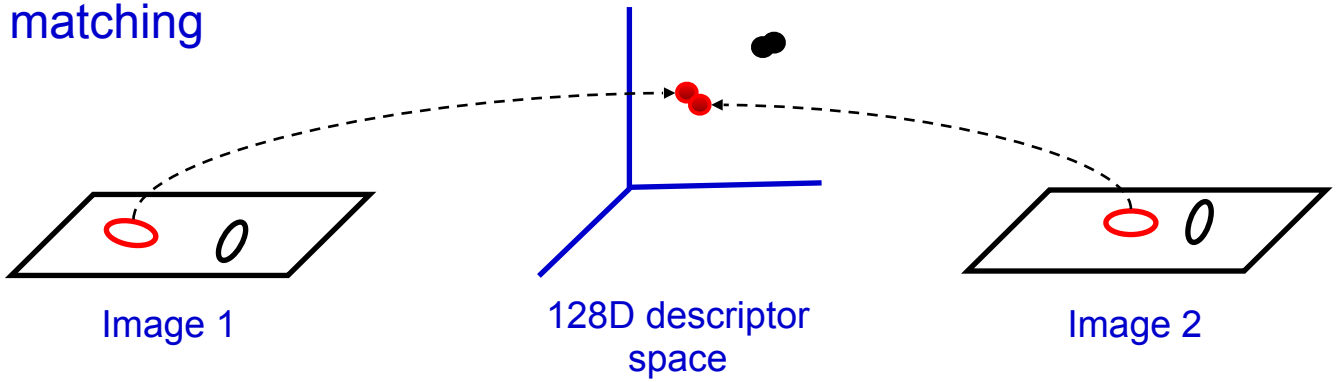


Visual words: quantize descriptor space

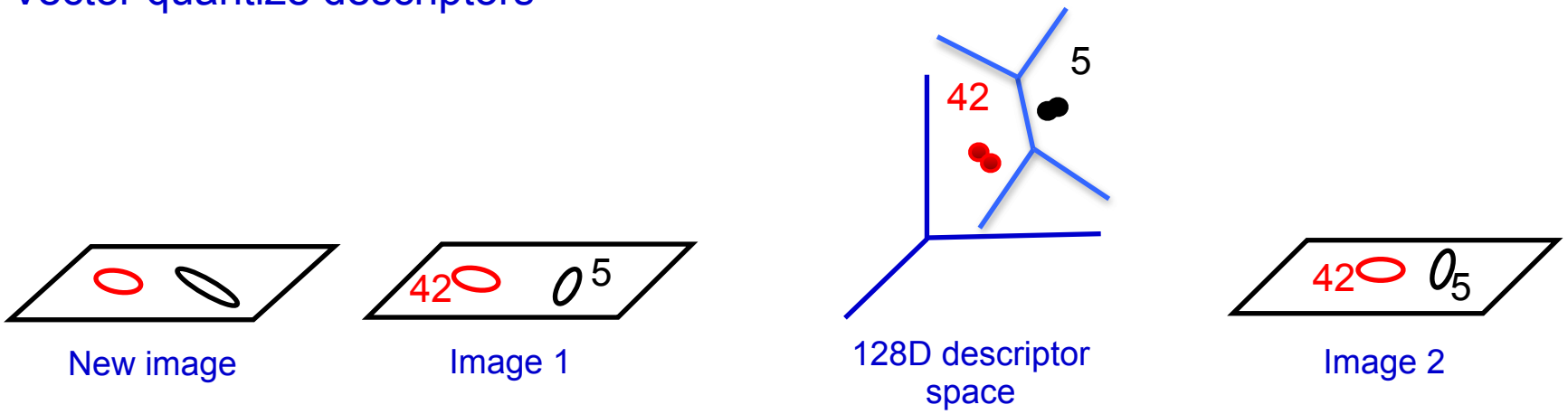
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames



Vector quantize descriptors

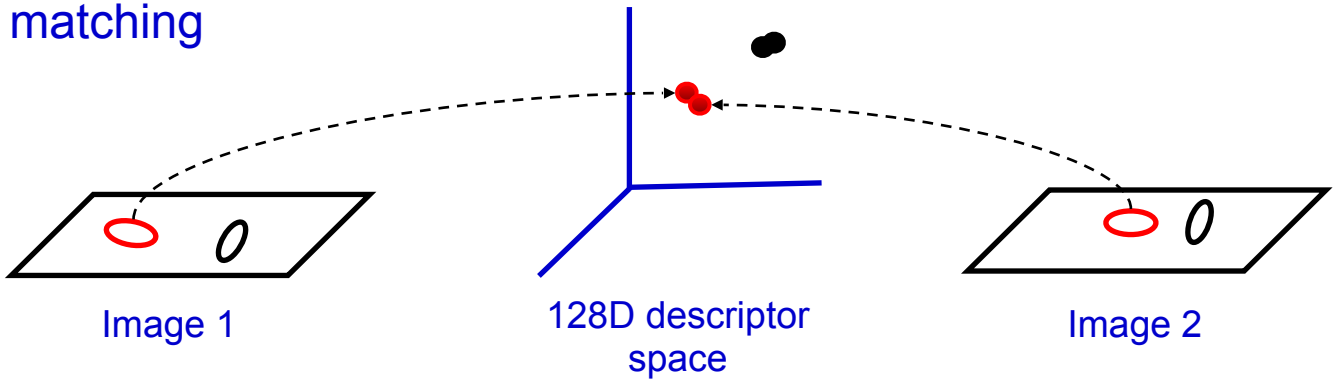


Visual words: quantize descriptor space

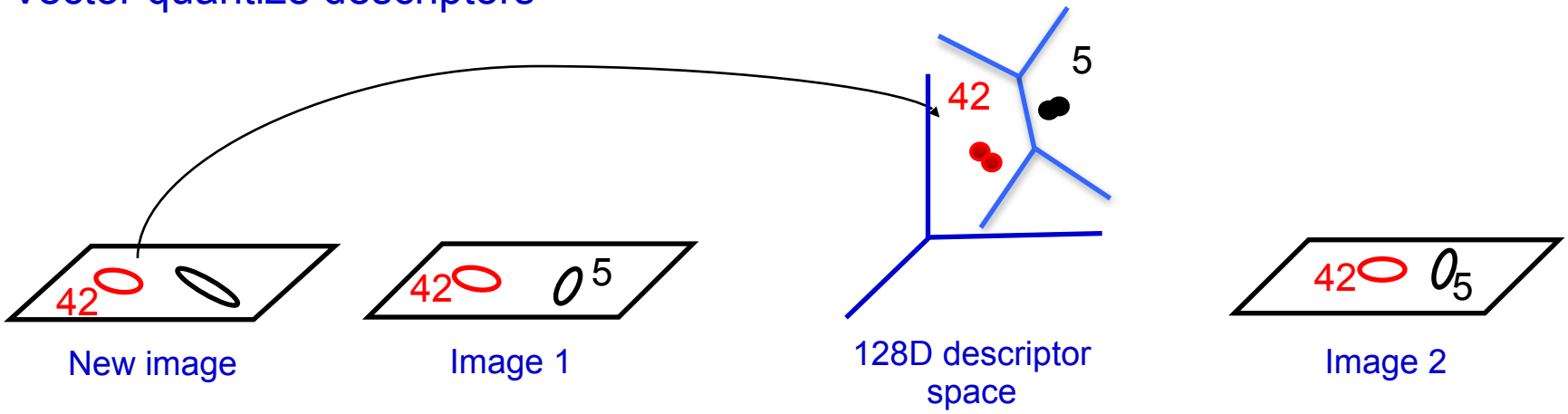
Sivic and Zisserman, ICCV 2003

Nearest neighbour matching

- expensive to do for all frames

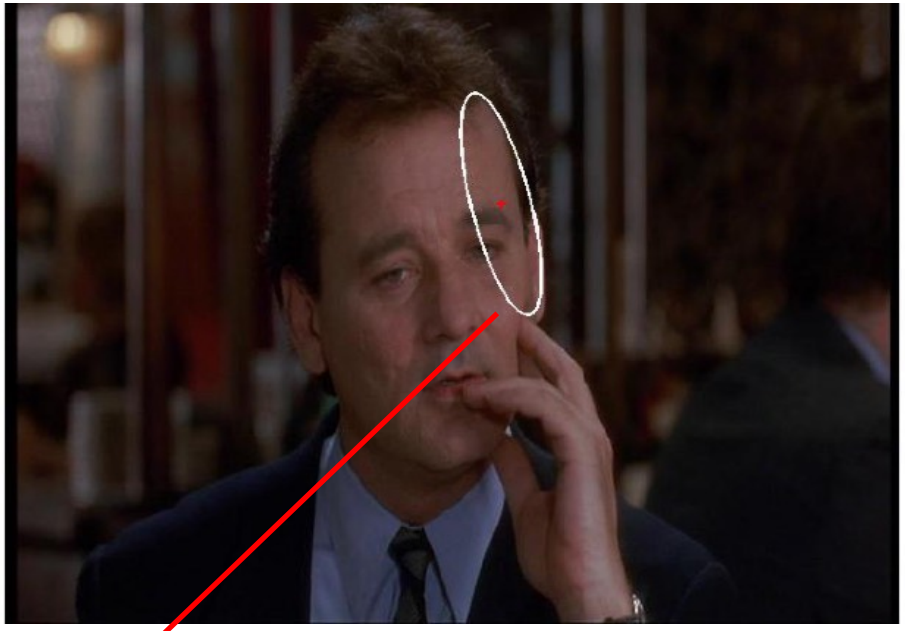


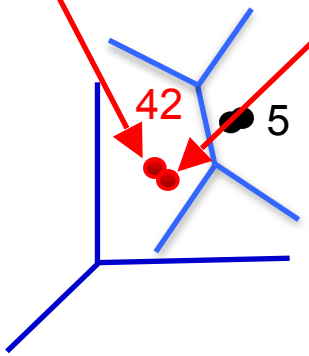
Vector quantize descriptors

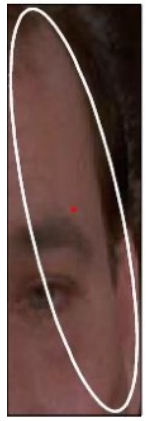


- Each region in the frame gets its label which immediately determines all the matches throughout the entire movie.
  - For example, a region gets label 42 and is matched to all regions with label 42 in the movie.
- The matching is turned into a look up in a table.

Vector quantize the descriptor space (SIFT)





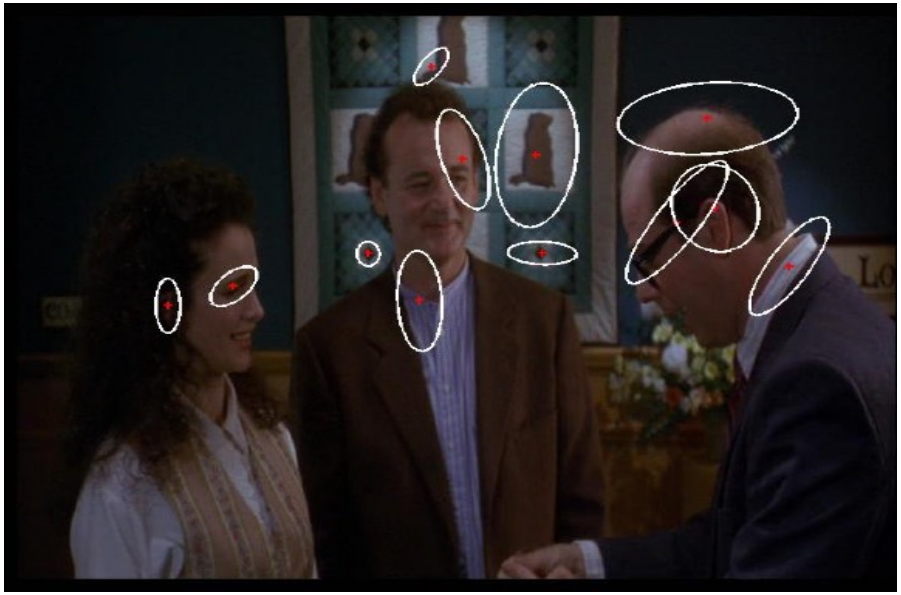


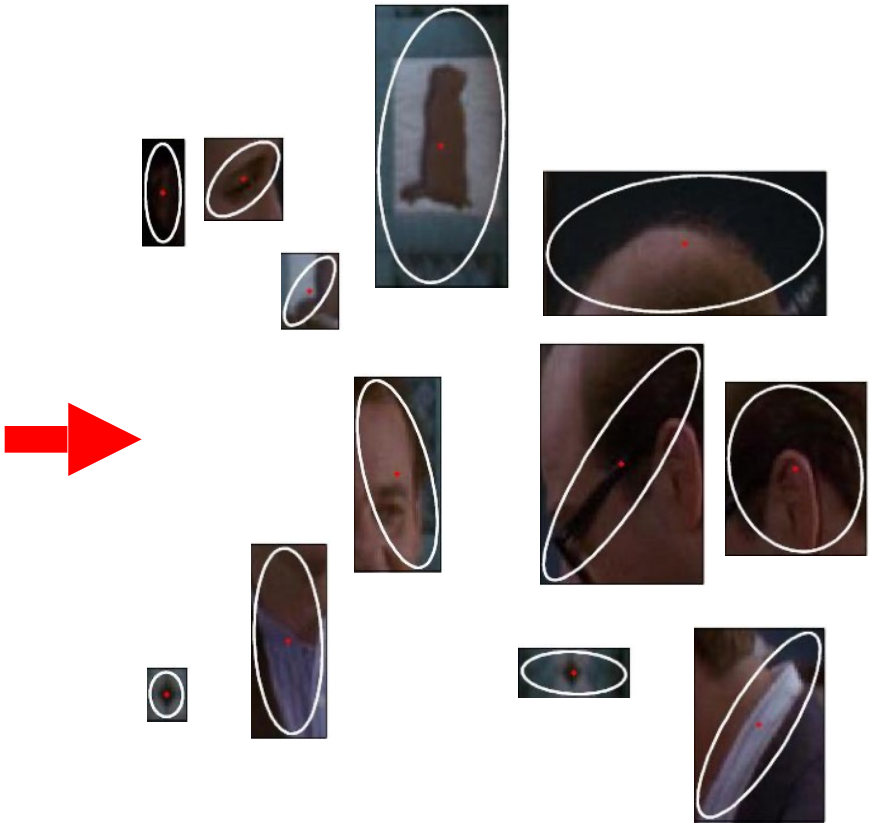
The same visual word

Representation: bag of (visual) words

Visual words are ‘iconic’ image patches or fragments

- represent their frequency of occurrence
- but not their position

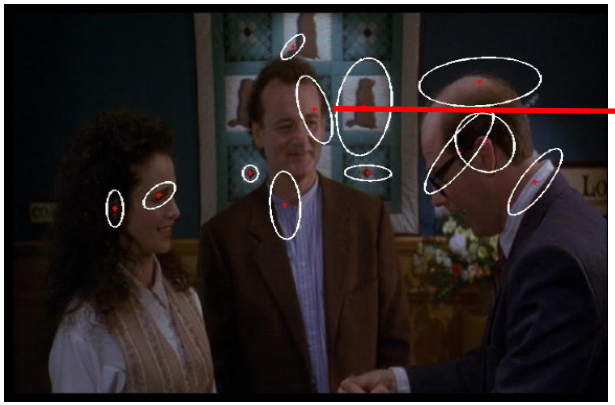




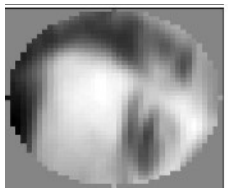
Image

Collection of visual words

Offline: Assign visual words and compute histograms for each image

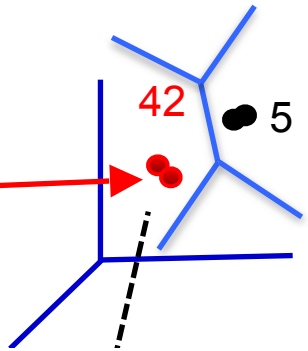


Detect patches

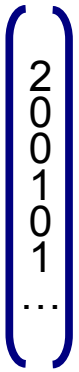


Normalize patch

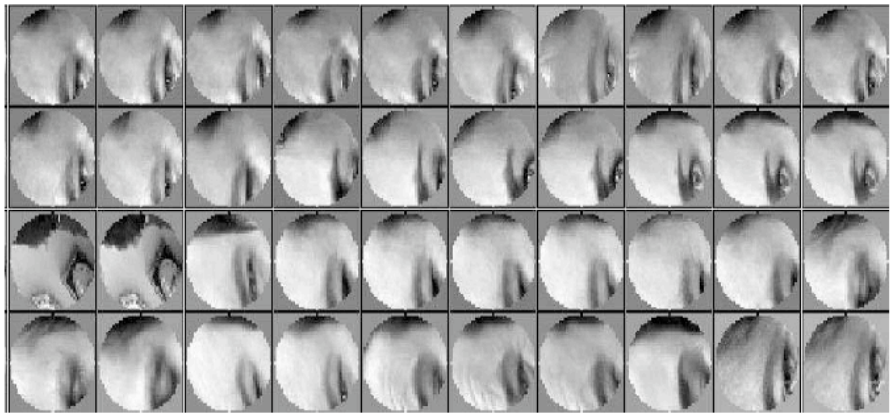
Compute SIFT descriptor



Find nearest cluster center



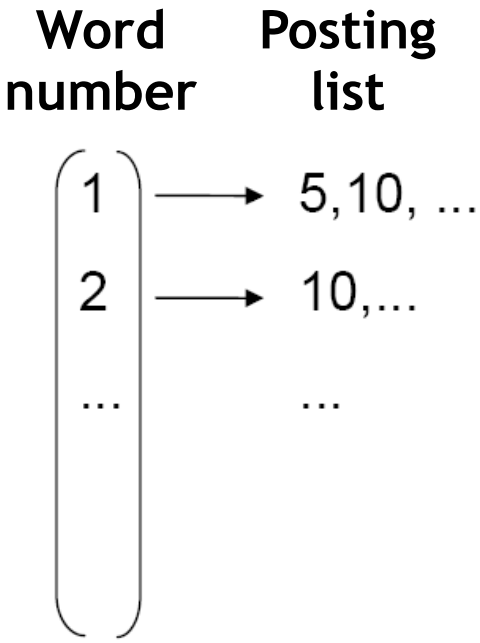
Represent image as a sparse histogram of visual word occurrences



Offline: create an index

frame #5

frame #10

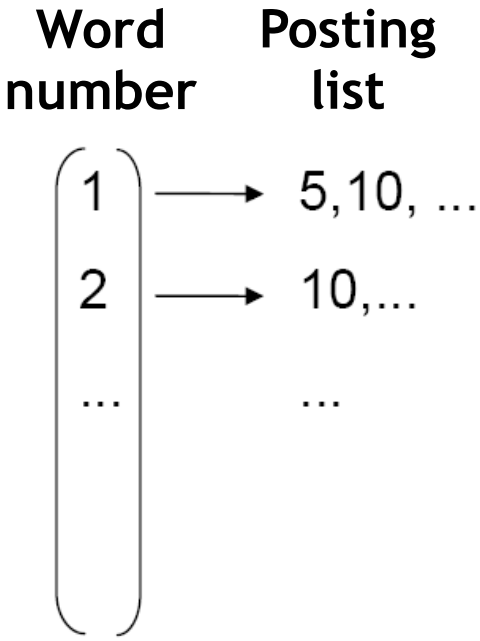


- For fast search, store a “posting list” for the dataset
- This maps visual word occurrences to the images they occur in (i.e. like the “book index”)

At run time

frame #5

frame #10

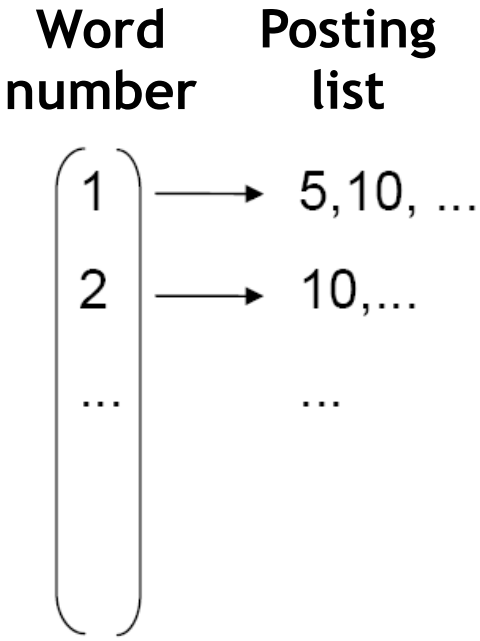


- User specifies a query region
- Generate a short-list of images using visual words in the region
  1. Accumulate all visual words within the query region
  2. Use “book index” to find other frames with these words
  3. Compute similarity for images which share at least one word

At run time

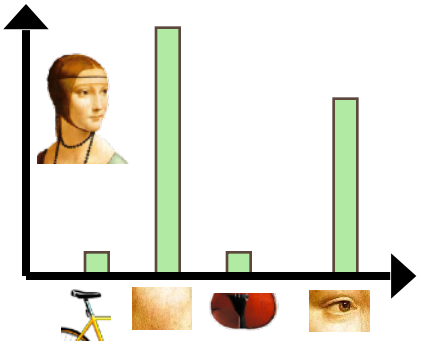
frame #5

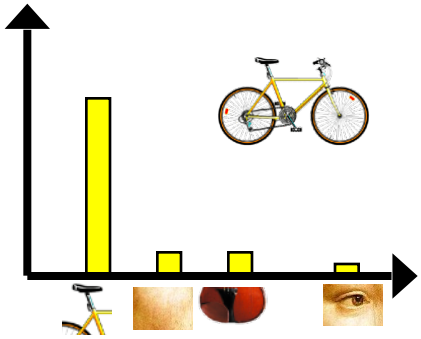
frame #10

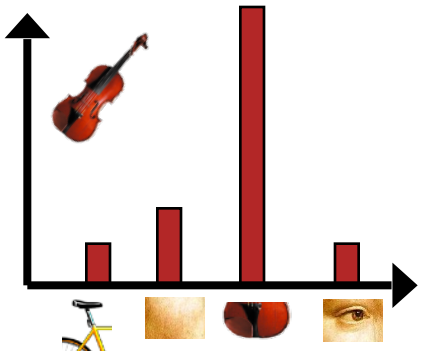


- Score each image by the (weighted) number of common visual words (tentative correspondences)
- Worst case complexity is linear in the number of images N
- In practice, it is linear in the length of the lists (<< N)

Bags of visual words

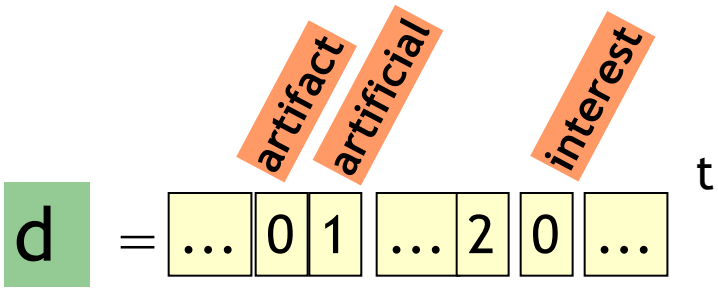






Summarize entire image based on its distribution (histogram) of visual word occurrences.

Analogous to bag of words representation commonly used for text documents.



Hofmann 2001

Slide: Grauman&Leibe, Image: L. Fei-Fei

Another interpretation: the bag-of-visual-words model

For a vocabulary of size K, each image is represented by a K-vector

$$\mathbf{v}_d = (t_1, \ldots, t_i, \ldots, t_K)^\top$$

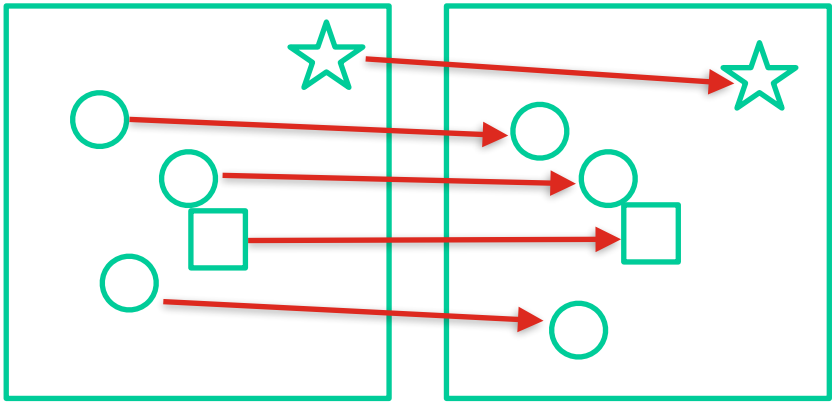
where t_i is the number of occurrences of visual word i.

Images are ranked by the normalized scalar product between the query vector v_q and all vectors in the database v_d:

$$f_d = \frac{\mathbf{v}_q{}^\top \mathbf{v}_d}{\|\mathbf{v}_q\|_2 \; \|\mathbf{v}_d\|_2}$$

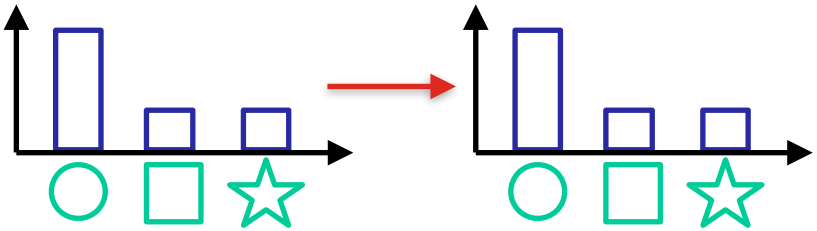
Recap: Two strategies

1. Efficient approximate nearest neighbor search on local feature descriptors.

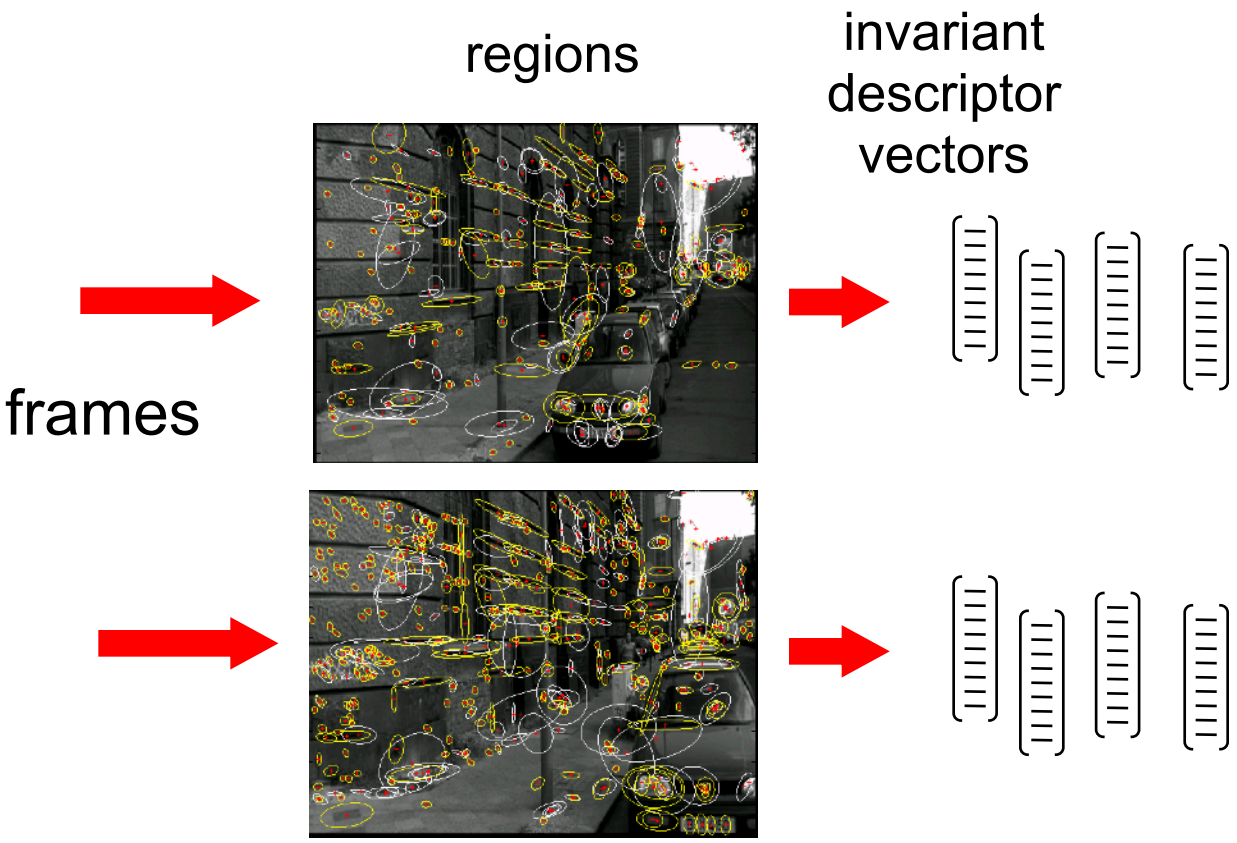


2. Quantize descriptors into a “visual vocabulary” and use efficient techniques from text retrieval.

   (Bag-of-words representation)

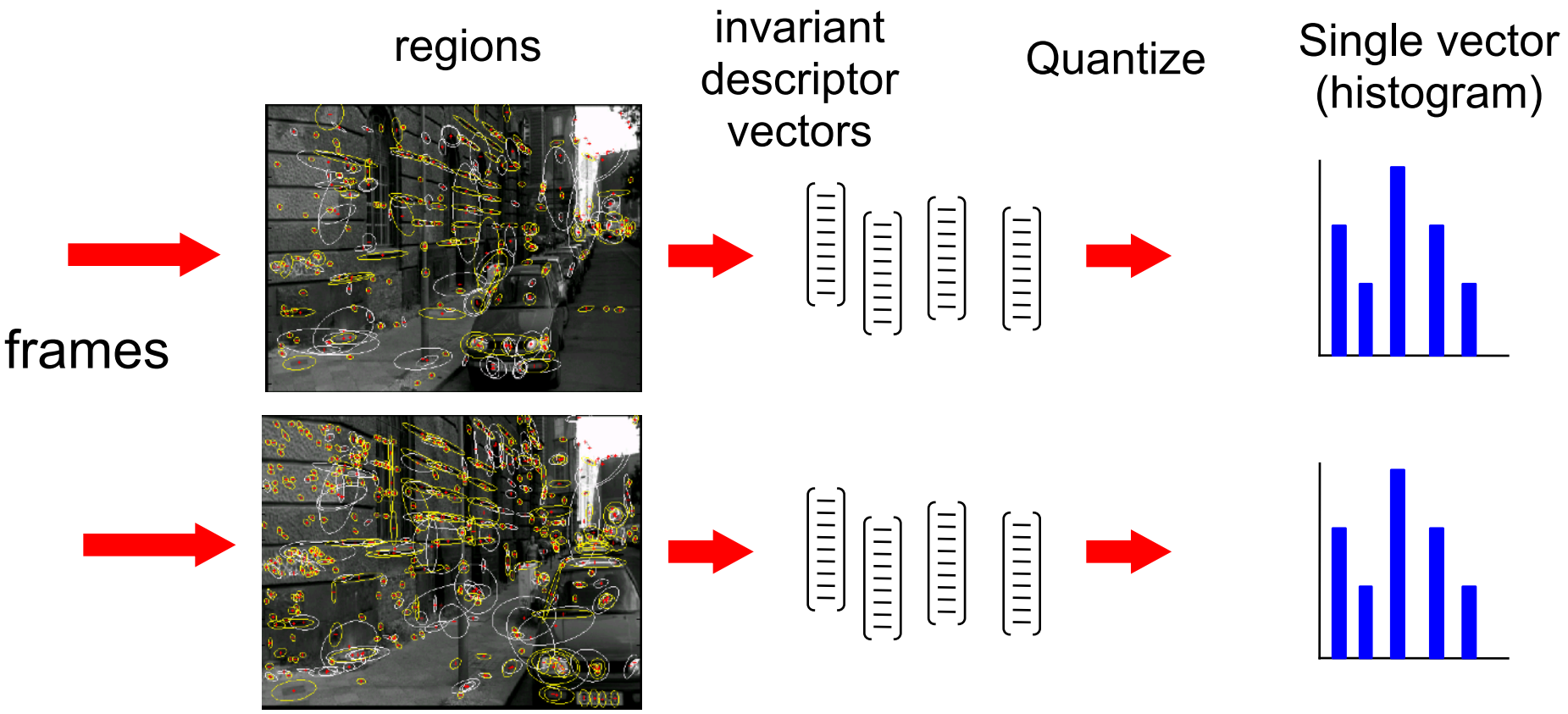


Strategy I: Efficient approximate NN search



1. Compute local features in each image independently (offline)
2. "Label" each feature by a descriptor vector based on its intensity (offline)
3. Finding corresponding features → finding nearest neighbour vectors
4. Rank matched images by number of (tentatively) corresponding regions
5. Verify top ranked images based on spatial consistency

Strategy II: Match histograms of visual words



1. Compute affine covariant regions in each frame independently (offline)
2. "Label" each region by a vector of descriptors based on its intensity (offline)
3. **Build histograms of visual words by descriptor quantization (offline)**
4. **Rank retrieved frames by matching vis. word histograms using inverted files.**
5. Verify retrieved frame based on spatial consistency.

Overview of the retrieval system

query image

[Lowe04, Mikolajczyk07]

Hessian-Affine regions + SIFT descriptors

Set of SIFT descriptors

[Sivic03, Philbin07]

Clustered and quantized to **visual words**

sparse frequency vector

tf-idf weighting

Inverted file

Querying

Ranked short-list of images

Geometric verification

[Lowe04, Philbin07]

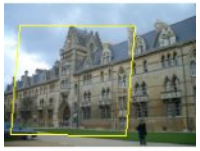
"Re-ranking" (e.g., top 1k)

Results

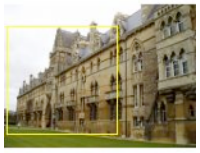
1 

3 

2 

4 

3 

5 

Visual search using local regions (references)

C. Schmid, R. Mohr, Local Greyvalue Invariants for Image Retrieval, PAMI, 1997

J. Sivic, A. Zisserman, Text retrieval approach to object matching in videos, ICCV, 2003

D. Nister, H. Stewenius, Scalable Recognition with a Vocabulary Tree, CVPR, 2006.

J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, CVPR, 2007

O. Chum, J. Philbin, M. Isard, J. Sivic, A. Zisserman, Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval, ICCV, 2007

H. Jegou, M. Douze, C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, ECCV'2008

O. Chum, M. Perdoch, J. Matas: Geometric min-Hashing: Finding a (Thick) Needle in a Haystack, CVPR 2009

H. Jégou, M. Douze and C. Schmid, On the burstiness of visual elements, CVPR, 2009

Visual search using local regions (references)

T. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (WS-LAVD), 2009.

H. Jégou, M. Douze, C. Schmid and P. Pérez, Aggregating local descriptors into a compact image representation, CVPR 2010

A. Mikulík, M. Perdoch, O. Chum, J. Matas, Learning a fine vocabulary, ECCV 2010.

O. Chum, A. Mikulik, M. Perdoch, J. Matas, Total recall II: Query expansion revisited, CVPR 2011

D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors. CVPR, 2011.

R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR,* 2012.

R. Arandjelović, A. Zisserman. DisLocation: Scalable descriptor distinctiveness for location recognition, In Asian Conference on Computer Vision, 2014

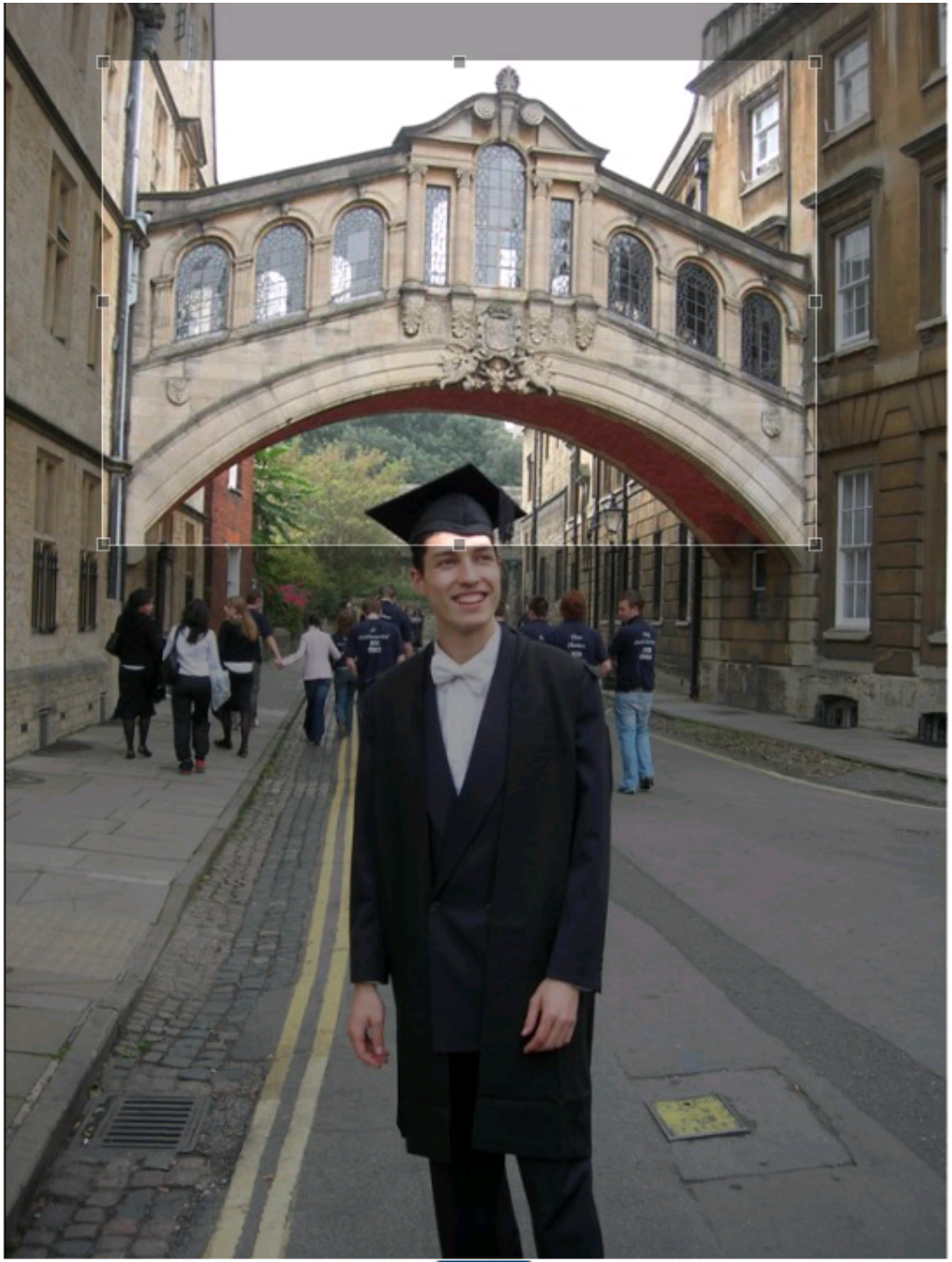
G. Tolias, Y. Avrithis, H. Jégou. Image search with selective match kernels: aggregation across single and multiple. International Journal of Computer Vision, 2016

Efficient visual search for objects and places

Oxford Buildings Search - demo

http://www.robots.ox.ac.uk/~vgg/research/oxbuildings/index.html

Example

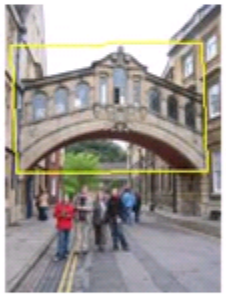


Search results 1 to 20 of 104844

1

ID: oxc1_hertford_000011
Score: 1816.000000
Putative: 2325
Inliers: 1816
Hypothesis: 1.000000 0.000000 0.000015 0.000000 1.000000 0.000031
Detail

2

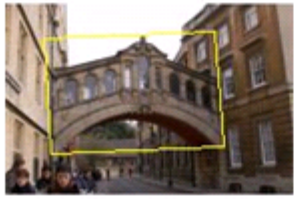


ID: oxc1_all_souls_000075
Score: 352.000000
Putative: 645
Inliers: 352
Hypothesis: 1.162245 0.041211 -70.414459 -0.012913 1.146417 91.276093
Detail

3

ID: oxc1_hertford_000064
Score: 278.000000
Putative: 527
Inliers: 278
Hypothesis: 0.928686 0.026134 169.954620 -0.041703 0.937558 97.962112
Detail

4

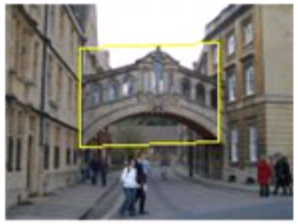


ID: oxc1_oxford_001612
Score: 252.000000
Putative: 451
Inliers: 252
Hypothesis: 1.046026 0.069416 51.576881 -0.044949 1.046938 76.264442
Detail

5

ID: oxc1_hertford_000123
Score: 225.000000
Putative: 446
Inliers: 225
Hypothesis: 1.361741 0.090413 -34.673317 -0.084659 1.301689 -32.281090
Detail

6



ID: oxc1_oxford_001085
Score: 224.000000
Putative: 389
Inliers: 224
Hypothesis: 0.848997 0.000000 195.707611 -0.031077 0.895546 114.583961
Detail

7

ID: oxc1_hertford_000077
Score: 195.000000
Putative: 386
Inliers: 195
Hypothesis: 1.465144 0.069286 -108.473091 -0.097598 1.461877 -30.205191
Detail

Oxford buildings dataset

- Automatically crawled from flickr
- Consists of:

Dataset	Resolution	# images	# features	Descriptor size
i	1024 × 768	5,062	16,334,970	1.9 GB
ii	1024 × 768	99,782	277,770,833	33.1 GB
iii	500 × 333	1,040,801	1,186,469,709	141.4 GB
Total		1,145,645	1,480,575,512	176.4 GB

Ondrej Chum, James Philbin, Josef Sivic, Michael Isard and Andrew Zisserman. *Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval*, ICCV 2007.

Oxford buildings dataset

- Landmarks plus queries used for evaluation

All Soul's

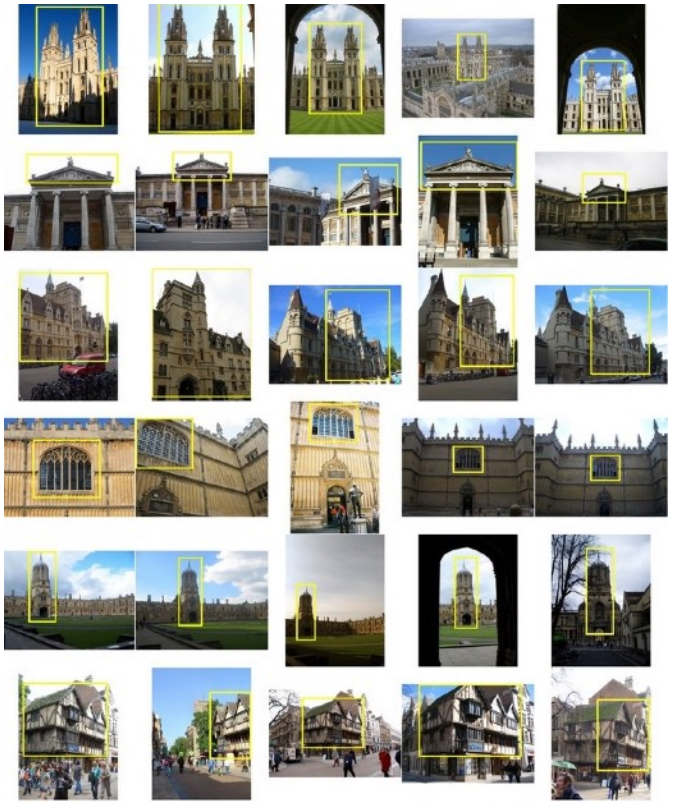
Ashmolean

Balliol

Bodleian

Thom Tower

Cornmarket



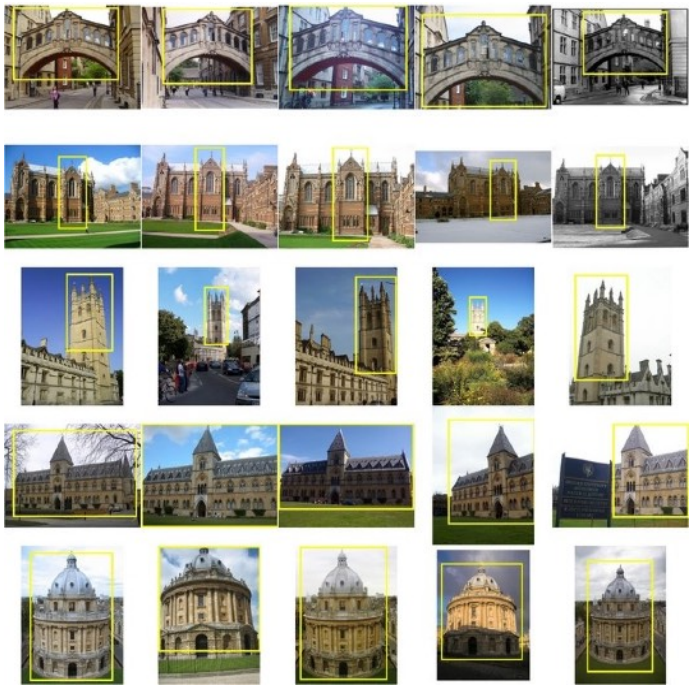
Bridge of Sighs

Keble

Magdalen

University Museum

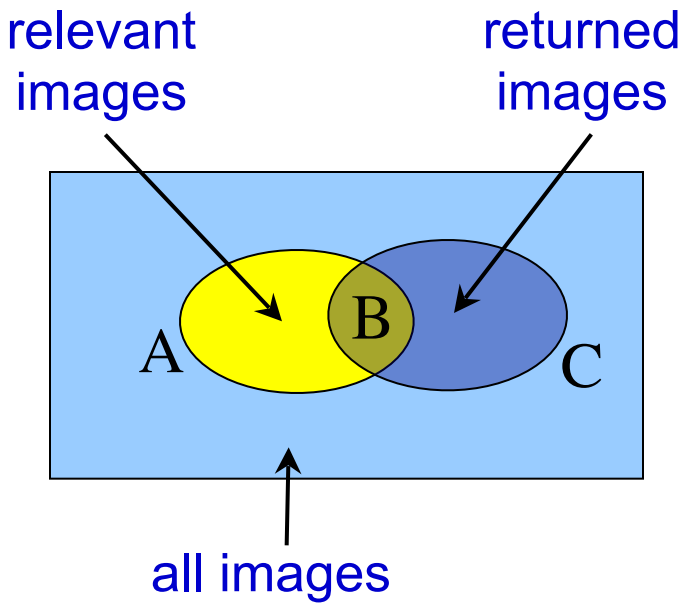
Radcliffe Camera



- Ground truth obtained for 11 landmarks
- Evaluate performance by mean Average Precision

Measuring retrieval performance: Precision-Recall

- Precision: A/B? A/C?
- Recall: B/A? B/C? C/A? C/B?



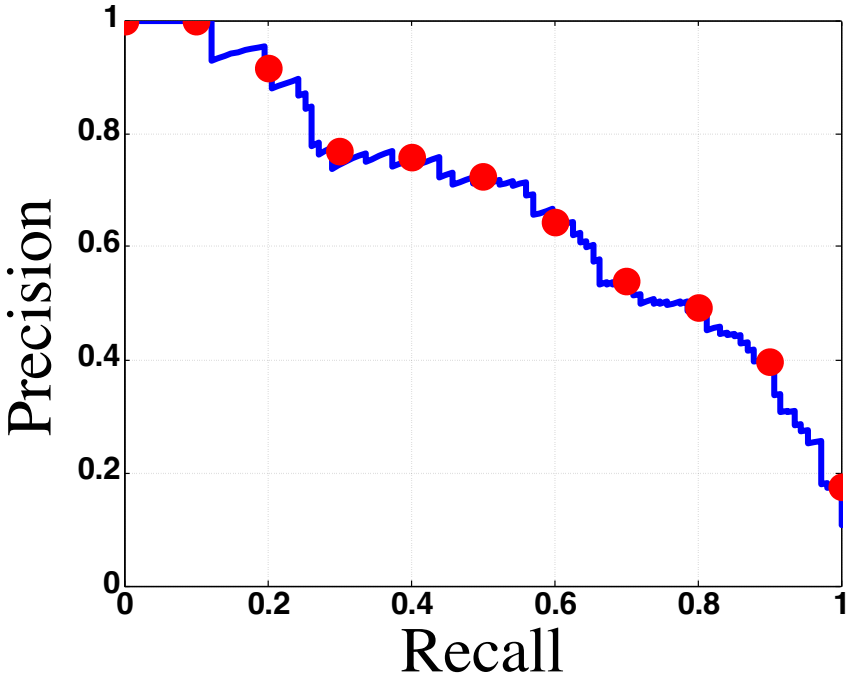
Measuring retrieval performance: Precision-Recall

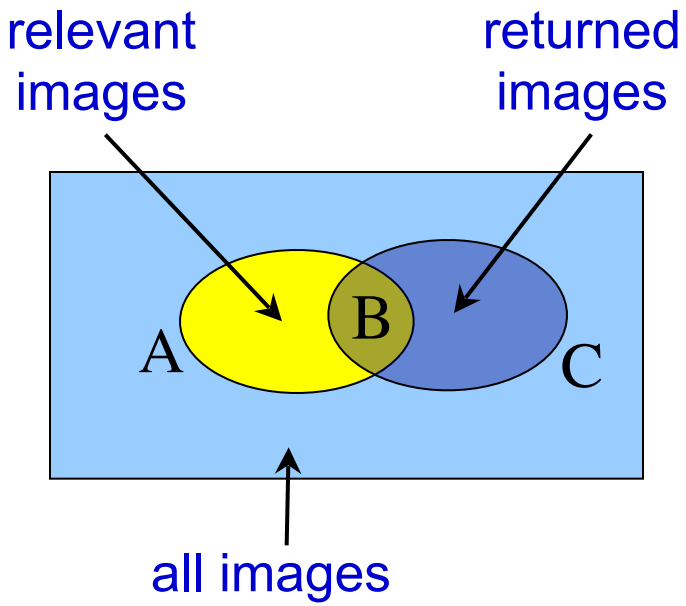
- Precision: % of returned images that are relevant

$$\frac{B}{C}$$

- Recall: % of relevant images that are returned

$$\frac{B}{A}$$





Measuring retrieval performance: Precision-Recall

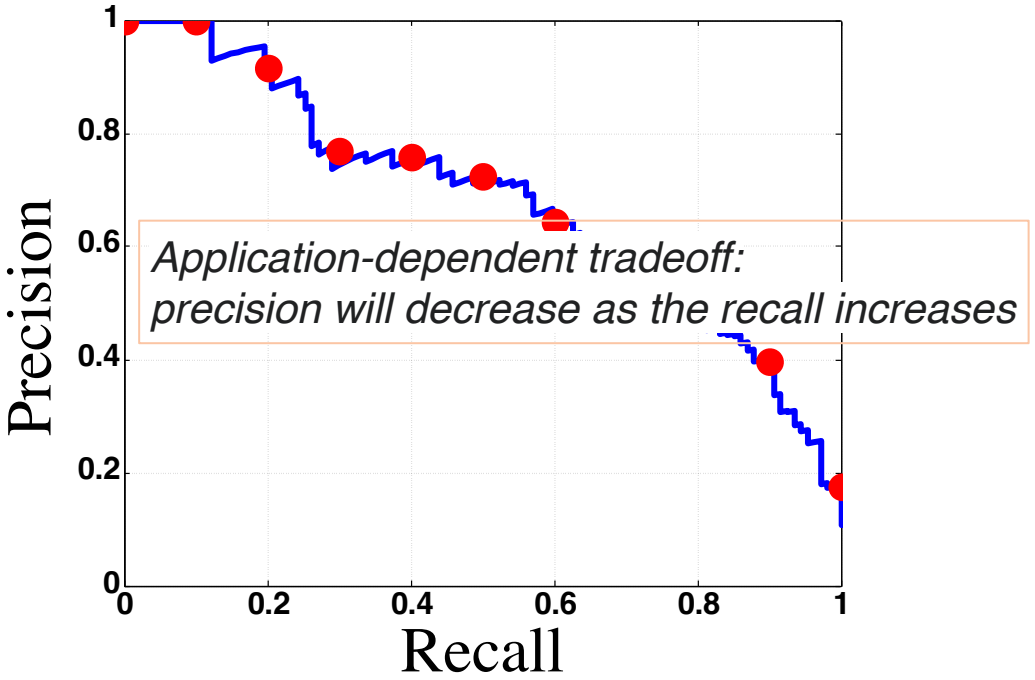
- Precision: % of returned images that are relevant

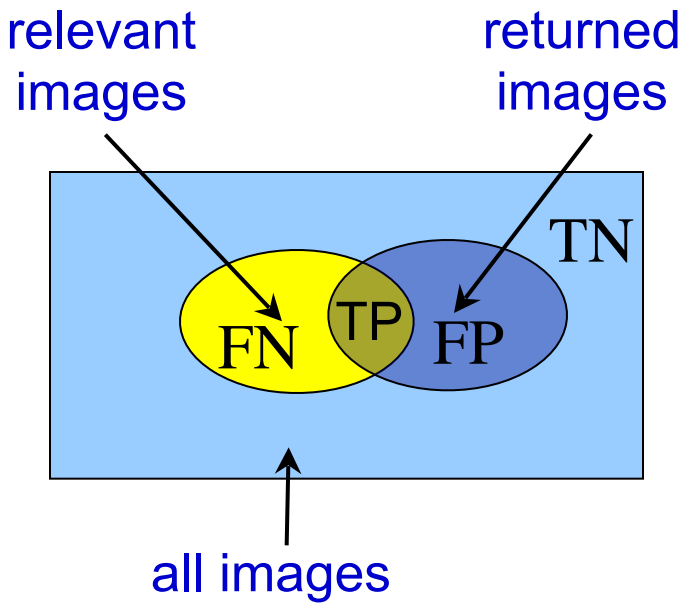
$$\frac{TP}{TP+FP}$$

- Recall: % of relevant images that are returned

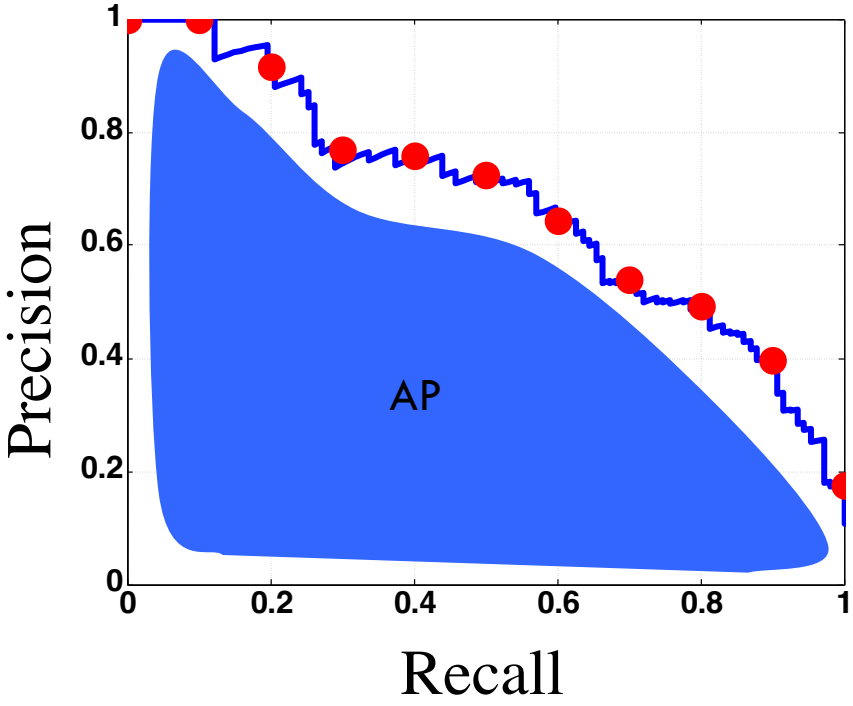
$$\frac{TP}{TP+FN}$$

TP: True Positive
FP: False Positive
FN: False Negative
TN: True Negative



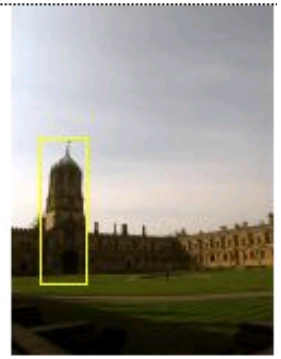


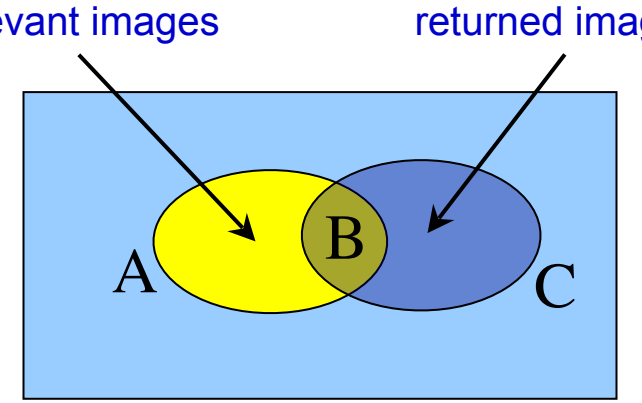
Average Precision

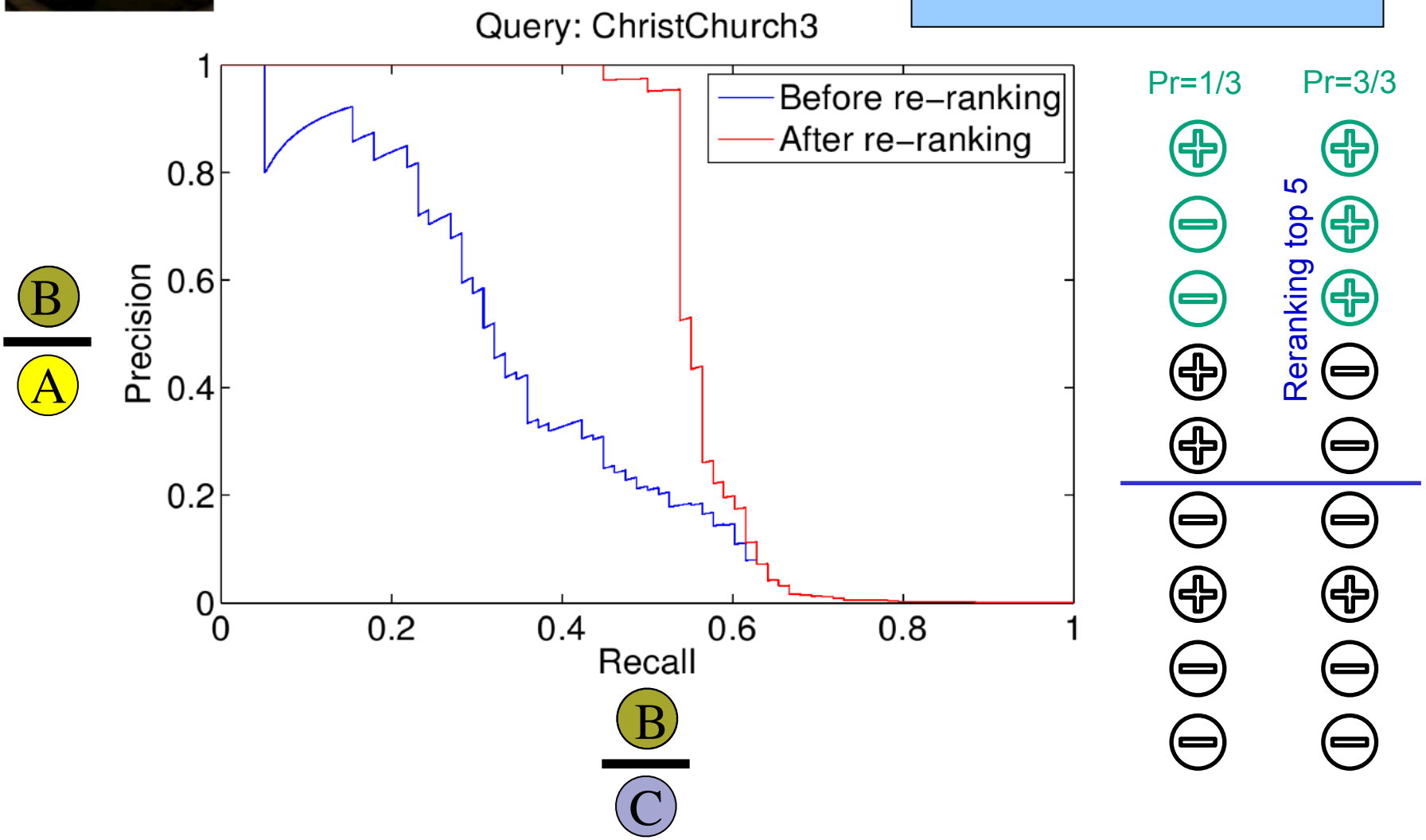


- A good AP score requires both high recall and high precision
- Application-independent

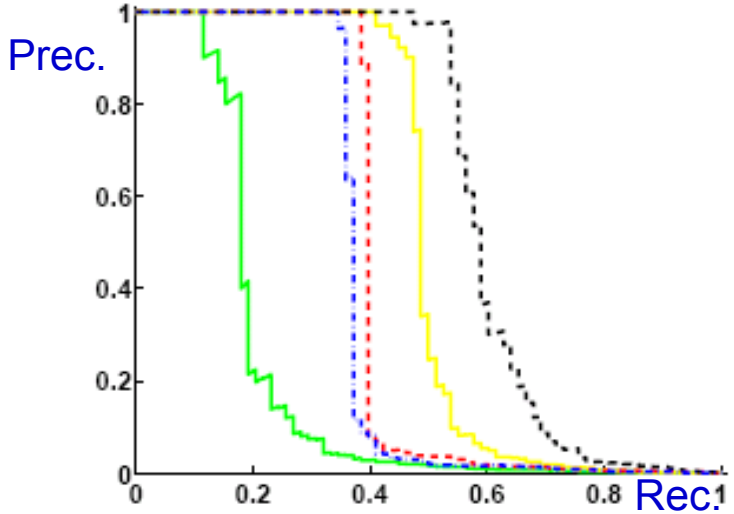
Performance measured by mean Average Precision (mAP) over 55 queries on 100K or 1.1M image datasets




relevant images
returned images
A
B
C


Query: ChristChurch3
Before re-ranking
After re-ranking
Precision
Recall
0
0.2
0.4
0.6
0.8
1
B
A
B
C
Pr=1/3
Pr=3/3
Reranking top 5

Query images



- high precision at low recall (like google)
- variation in performance over query
- none retrieve all instances

Visual search (references)

G. Tolias, Y. Avrithis, H. Jégou. Image search with selective match kernels: aggregation across single and multiple. International Journal of Computer Vision, 2016

G Tolias, R Sicre, H Jégou, Particular object retrieval with integral max-pooling of CNN activations, International Conference on Learning Representations (ICLR) 2016

F Radenović, G Tolias, O Chum, CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples, European Conference on Computer Vision (ECCV) 2016.

F Radenović, A Iscen, G Tolias, Y Avrithis, O Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

Agenda: Efficient Visual Search

1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

Why aren't all objects retrieved?

Obtaining visual words is like a sensor measuring the image

"noise" in the measurement process means that some visual words are missing or incorrect, e.g. due to

- Missed detections
- Changes beyond built in invariance
- Quantization effects

1. Query expansion
2. Better quantization

Consequence: Visual word in query is missing in target image

Query Expansion in text

In text :

- Reissue top-n responses as queries
- Pseudo/blind relevance feedback*
- Danger of topic drift

In vision:

- Reissue spatially verified image regions as queries

- **Explicit feedback:** obtained from assessors of relevance indicating the relevance of a document retrieved for a query.
- **Implicit feedback:** inferred from user behavior (which documents they select for viewing, the time spent viewing a document, or page browsing/scrolling actions.
- ***Pseudo (blind) relevance feedback:** automates the manual part of relevance feedback, assume top-n ranked documents are relevant

https://en.wikipedia.org/wiki/Relevance_feedback

Query Expansion: Text

Original query: Hubble Telescope Achievements

Query expansion: Select top 20 terms from top 20 documents according to tf-idf

Added terms: Telescope, hubble, space, nasa, ultraviolet, shuttle, mirror, telescopes, earth, discovery, orbit, flaw, scientists, launch, stars, universe, mirrors, light, optical, species

Example from: Jimmy Lin, University of Maryland

(Parenthesis: tf-idf)

term frequency - inverse document frequency

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

Measure of **importance of a word to a document** in a corpus, adjusted for the fact that **some words appear more frequently**.

- **Term frequency**, **tf:**
  frequency of a term *t* within a document *d*

$$tf\,(t,d) = \frac{Count\ of\ a\ term\ in\ document}{Total\ number\ of\ terms\ in\ document}$$

- **Inverse document frequency (idf)**:
  how much information the word provides,
  i.e., how common or rare it is across *all* documents *D*.

$$idf(t, D) = log(\frac{N}{|d : d \in D \text{ and } t \in d|}) = log\ \frac{Total\ number\ of\ documents\ in\ the\ corpus}{Number\ of\ documents\ where\ the\ term\ t\ appears}$$

log(37/df)

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Shakespeare's N=37 plays

A high weight in tf–idf => high term frequency (in the given document) and low document frequency of the term in the whole collection of documents.

https://en.wikipedia.org/wiki/Tf-idf

https://web.stanford.edu/~jurafsky/slp3/14.pdf

(Parenthesis: tf-idf)

term frequency - inverse document frequency

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

Measure of **importance of a word to a document** in a corpus, adjusted for the fact that **some words appear more frequently**.

- **Term frequency**, **tf:**
  frequency of a term *t* within a document *d*

$$tf\,(t,d) = \frac{\textit{Count of a term in document}}{\textit{Total number of terms in document}}$$

- **Inverse document frequency (idf)**:
  how much information the word provides,
  i.e., how common or rare it is across *all* documents *D*.

$$idf(t, D) = log(\frac{N}{|d : d \in D \text{ and } t \in d|}) = log\,\frac{\textit{Total number of documents in the corpus}}{\textit{Number of documents where the term t appears}}$$

log(37/df)

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Shakespeare's N=37 plays

"Romeo", "Falstaff", and "salad" appears in very few plays, so seeing these words, one could get a good idea as to which play it might be. In contrast, "good" and "sweet" appears in every play and are completely uninformative as to which play it is.

https://en.wikipedia.org/wiki/Tf-idf

https://web.stanford.edu/~jurafsky/slp3/14.pdf

Query Expansion: Text

Original query: Hubble Telescope Achievements

Query expansion: Select top 20 terms from top 20 documents according to **tf-idf**

Added terms: Telescope, hubble, space, nasa, ultraviolet, shuttle, mirror, telescopes, earth, discovery, orbit, flaw, scientists, launch, stars, universe, mirrors, light, optical, species

Example from: Jimmy Lin, University of Maryland

Automatic query expansion

Visual word representations of two images of the same object may differ (due to e.g. detection/quantization noise) resulting in missed returns

Initial returns may be used to add new relevant visual words to the query

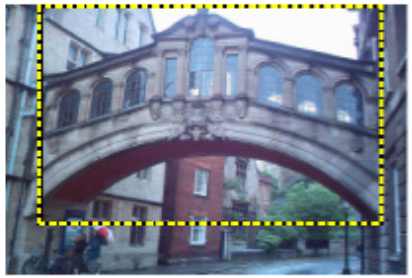
Strong spatial model prevents ‘drift’ by discarding false positives

[Chum, Philbin, Sivic, Isard, Zisserman, ICCV’07;

Chum, Mikulik, Perdoch, Matas, CVPR’11]

Visual query expansion - overview

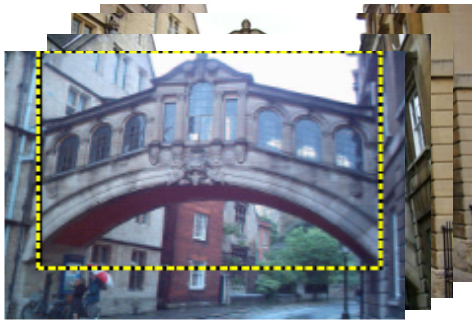
1. Original query



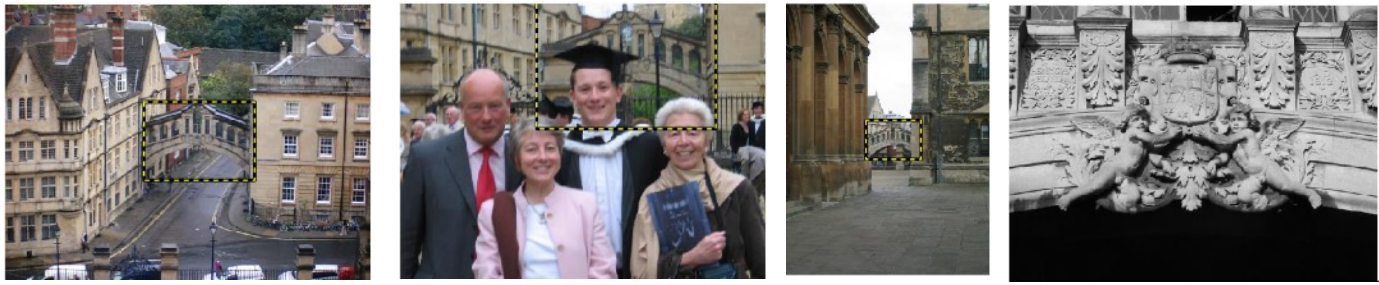
2. Initial retrieval set

3. Spatial verification (can prevent drift as opposed to text)

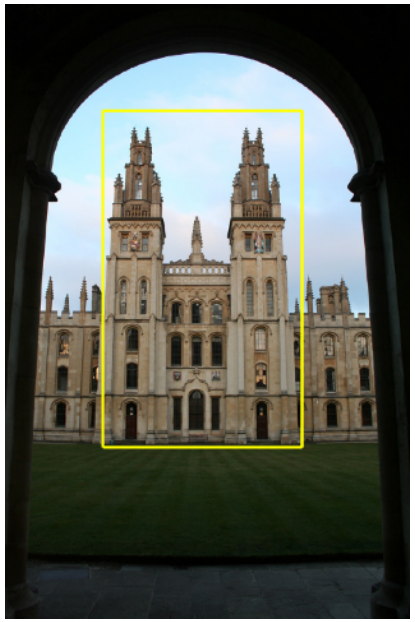
4. New enhanced query



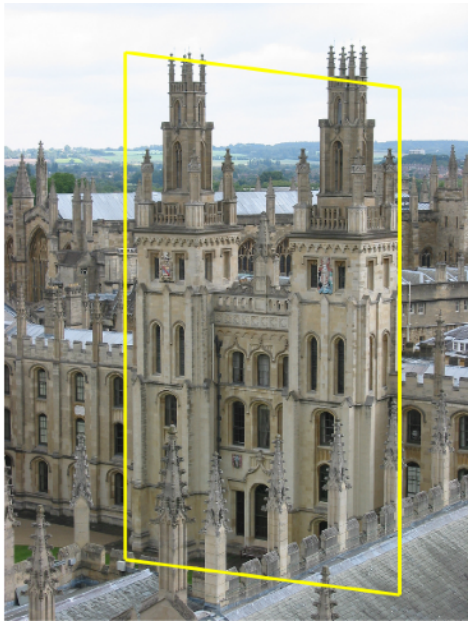
5. Additional retrieved images



Query Expansion



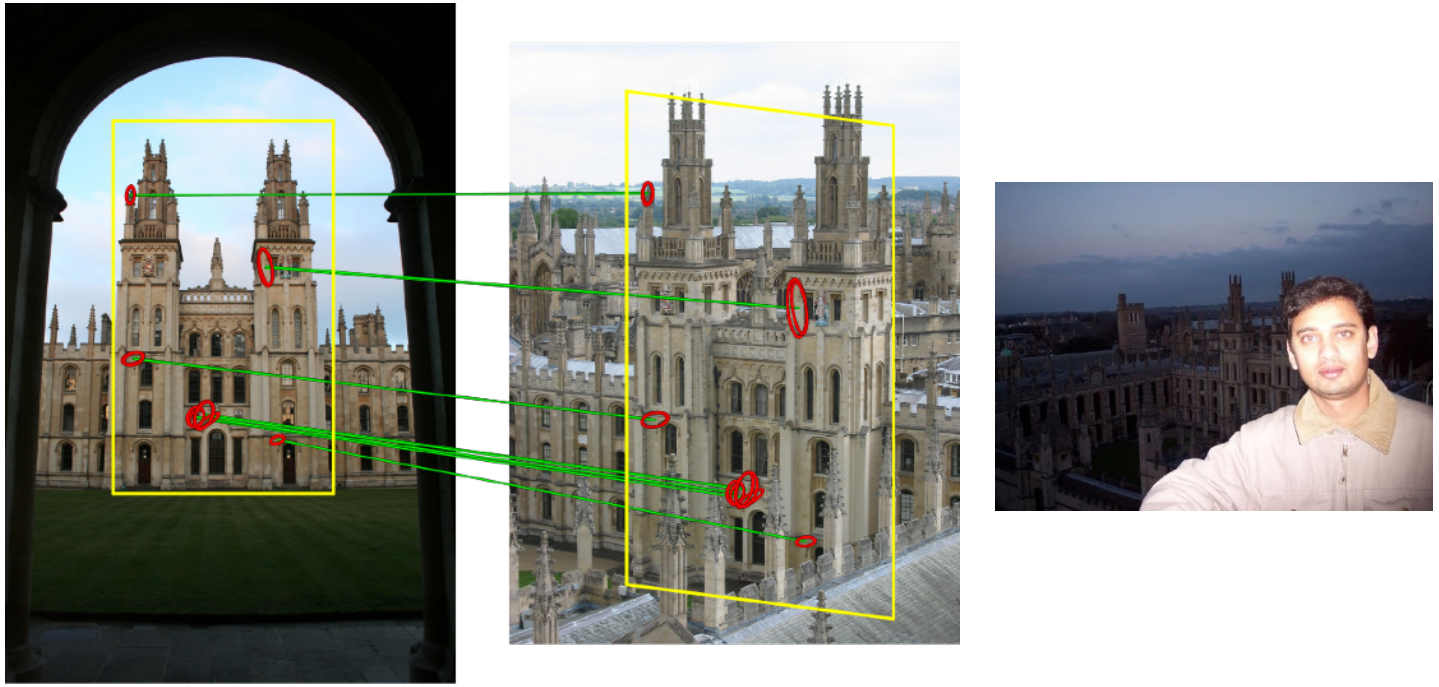
Query Image



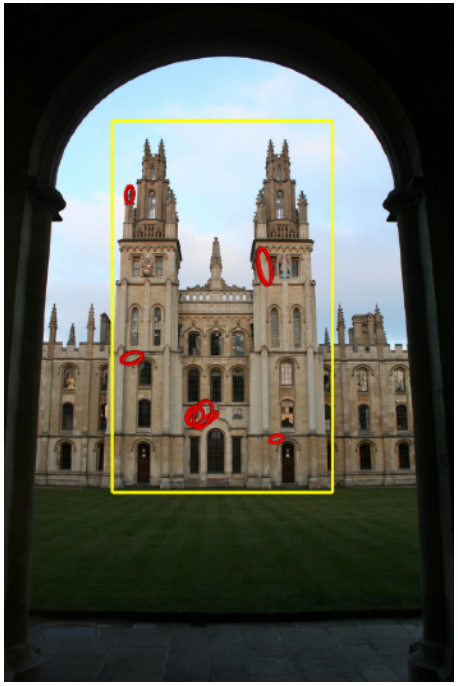
Originally retrieved image

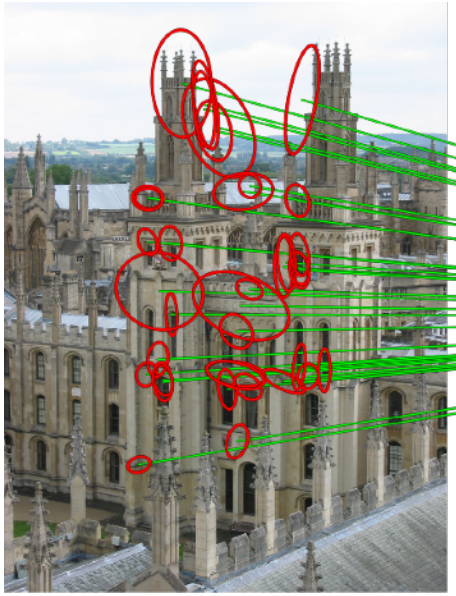
Originally not retrieved

Query Expansion

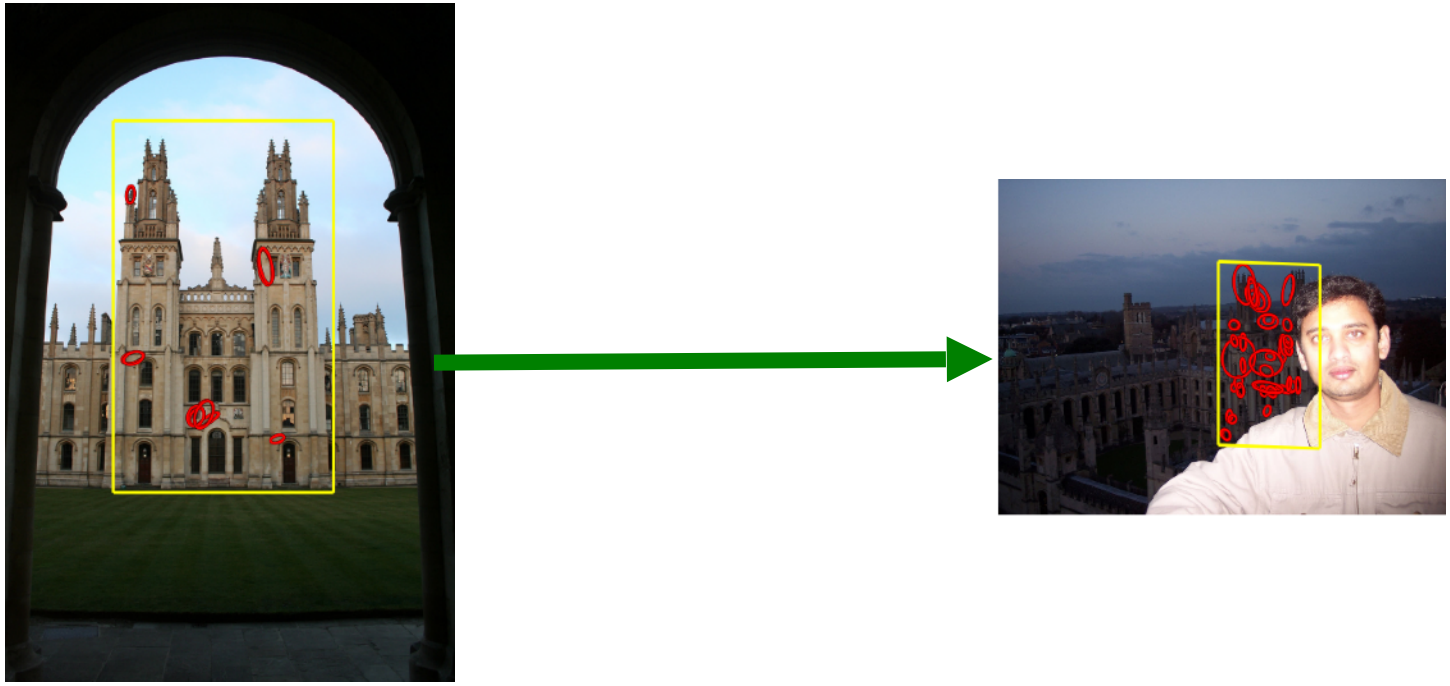


Query Expansion



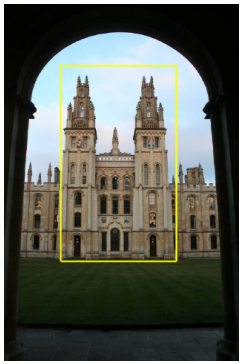


Query Expansion

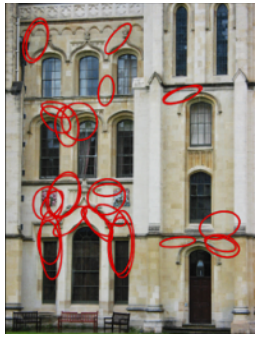
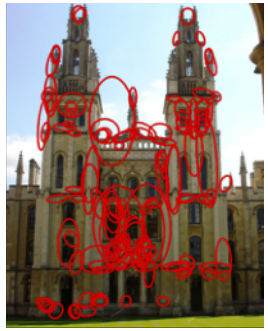
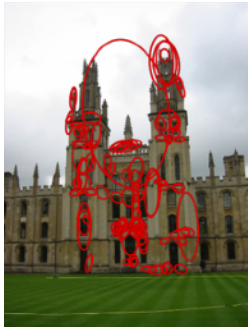


Query Expansion

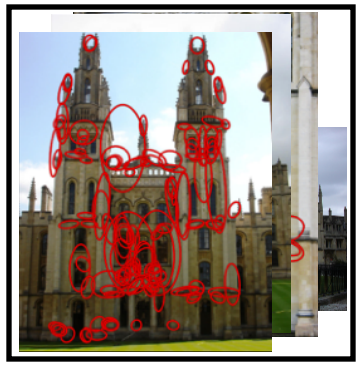
Query Image



Spatially verified retrievals with matching regions overlaid


...



New expanded query

New expanded query is formed as

- the **average** of visual word vectors of spatially verified returns
- only inliers are considered
- regions are back-projected to the original query image

Query Expansion

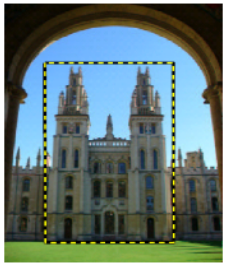
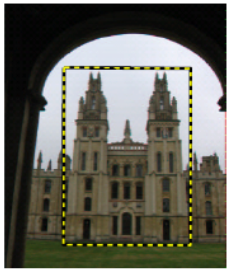
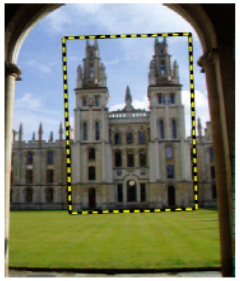
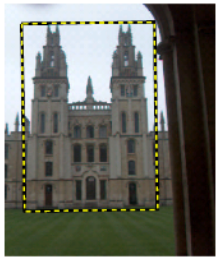
Query image

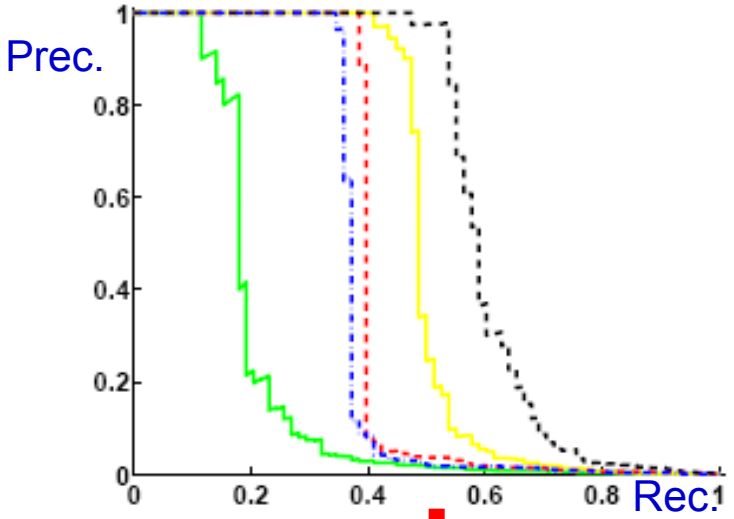
Originally retrieved

Retrieved only after expansion

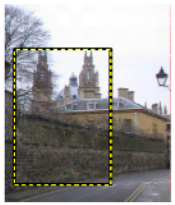
Query image

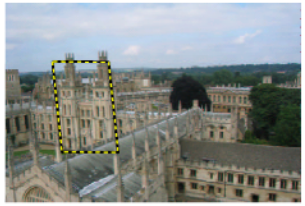
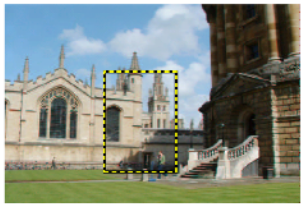
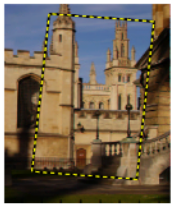
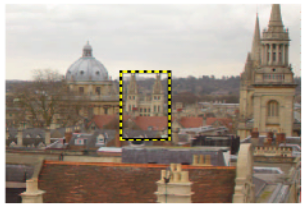
Original results (good)

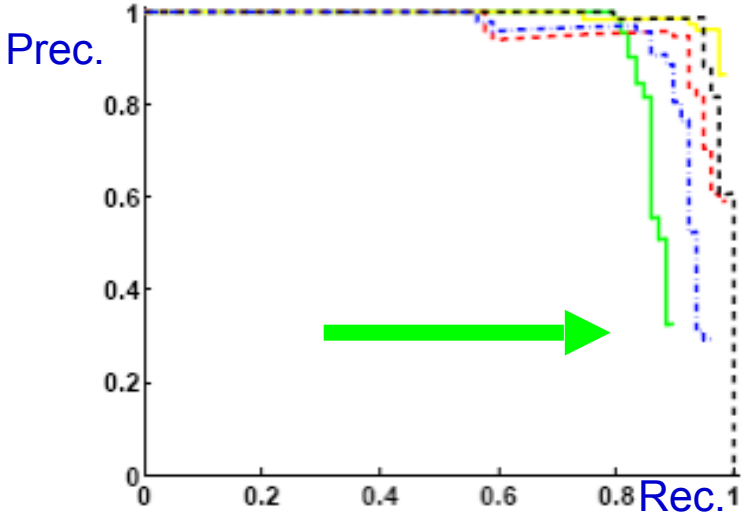







Expanded results (better)



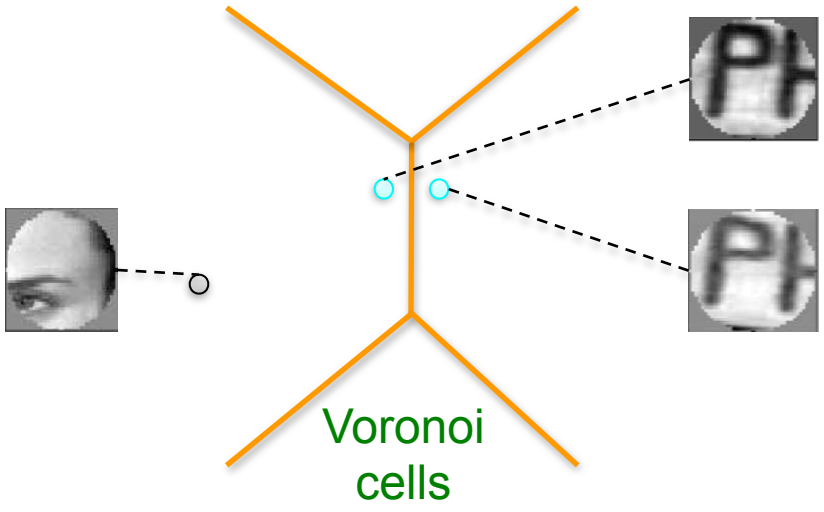







Quantization errors

Typically, quantization has a significant impact on the final performance of the system **[Sivic03,Nister06,Philbin07]**

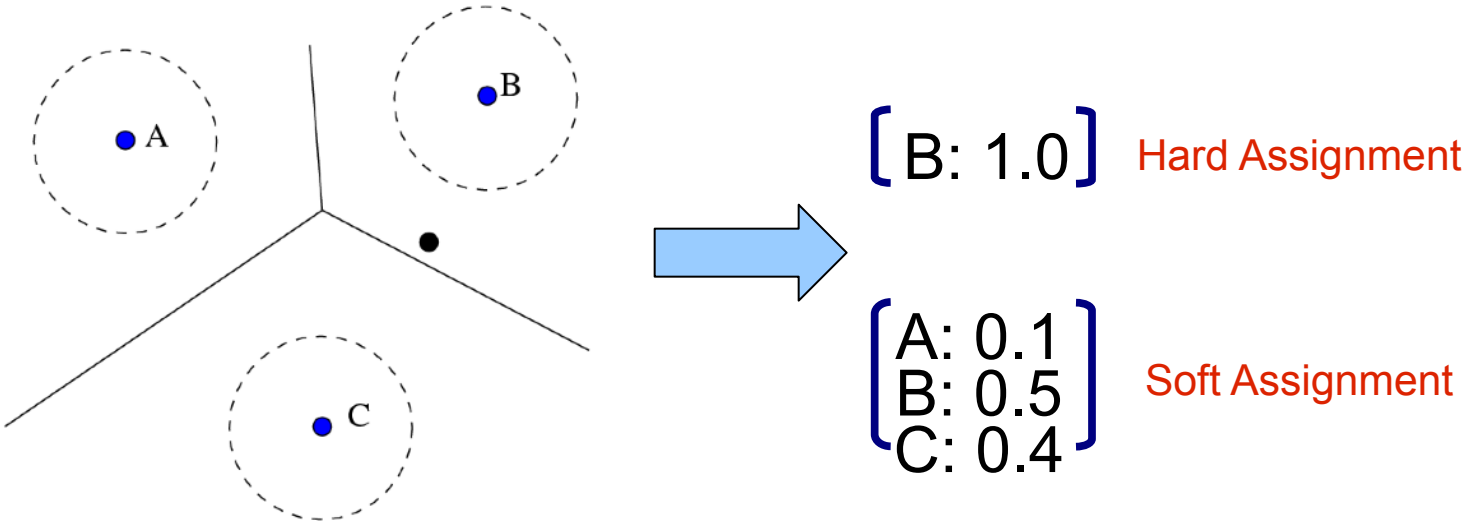
Quantization errors split features that should be grouped together and confuse features that should be separated



Overcoming quantization errors

- Soft-assign each descriptor to multiple cluster centers

[Philbin et al. 2008, Van Gemert et al. 2008]



Learning a vocabulary to overcome quantization errors

[Mikulik et al. ECCV 2010, Philbin et al. ECCV 2010]

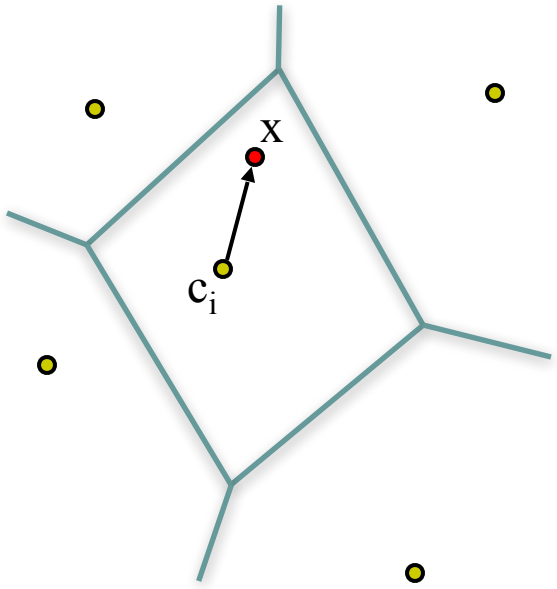
Beyond bag-of-visual-words

VLAD – Vector of locally aggregated descriptors

[Jegou et al. 2010]

Measure (and quantize) the difference vectors from the cluster center.

K-Means



Beyond bag-of-visual-words

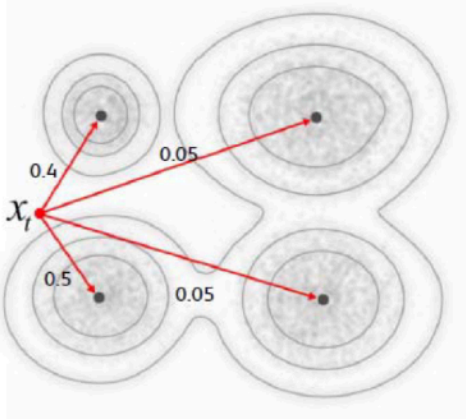
FV – Fisher Vector

[Perronin et al. 2010]

FV formulas:

$$\mathcal{G}^X_{\mu,i} = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^{T} \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right)$$

$$\mathcal{G}^X_{\sigma,i} = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right]$$



Gaussian Mixture Model (GMM)

Soft-assignment to each Gaussian

Concatenation of per-Gaussian gradient vectors

Agenda: Efficient Visual Search

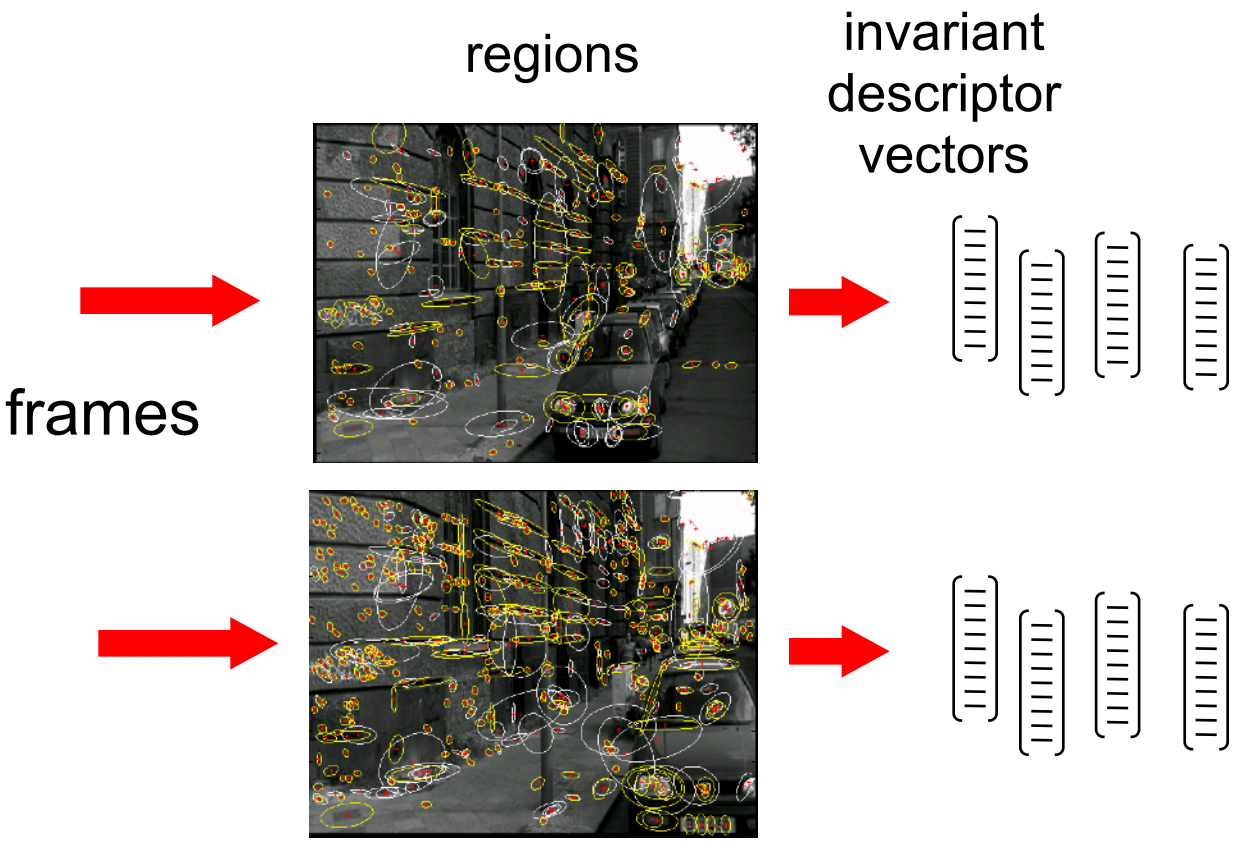
1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

Towards very large-scale image search

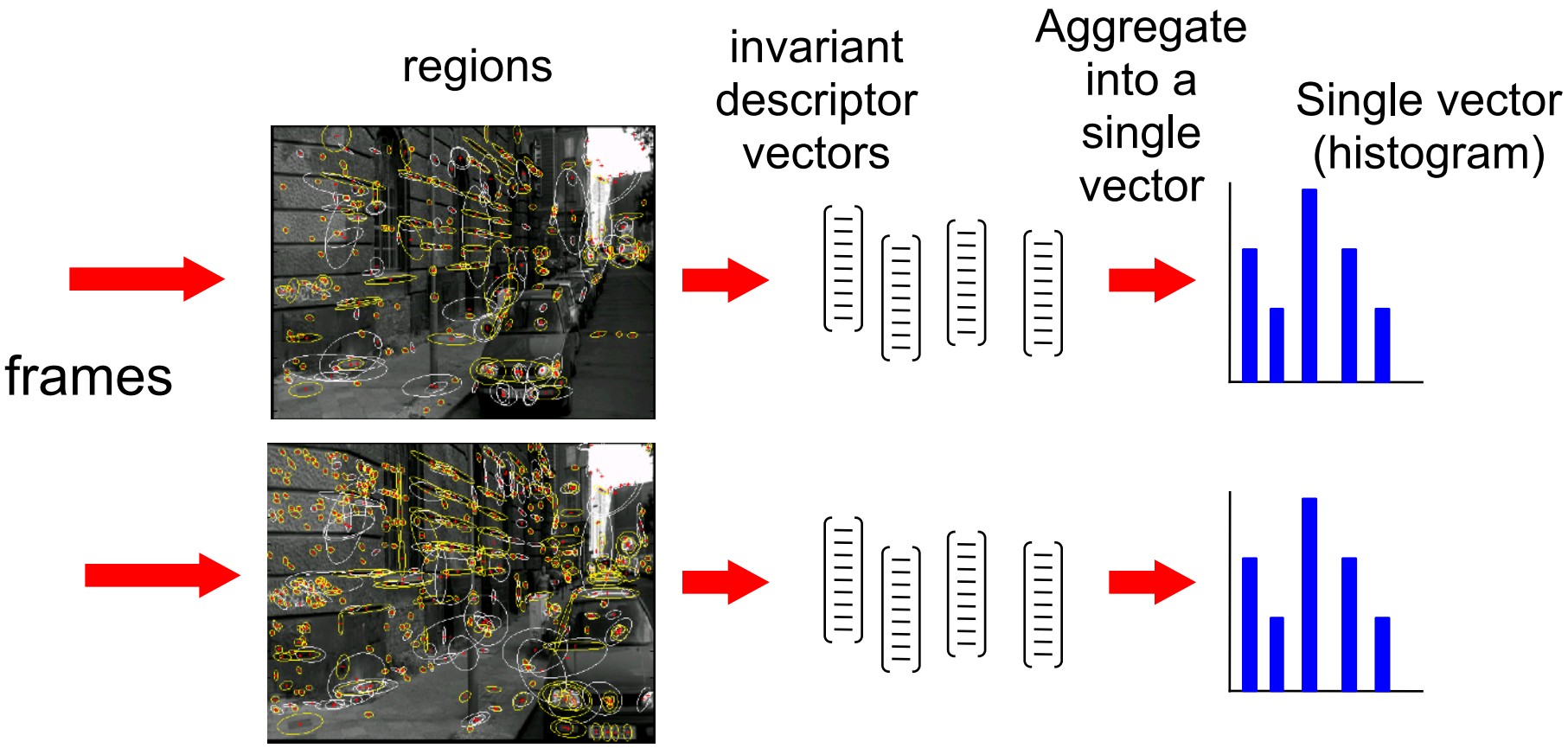
- BOF+inverted file can handle up to ~10 millions images
  - with a limited number of descriptors per image → RAM: 40GB
  - search: 2 seconds
- Web-scale = billions of images
  - with 100 M per machine → search: 20 seconds, RAM: 400 GB
  - not tractable
- Solution: represent each image by one **compressed** vector

Slide credit: C. Schmid

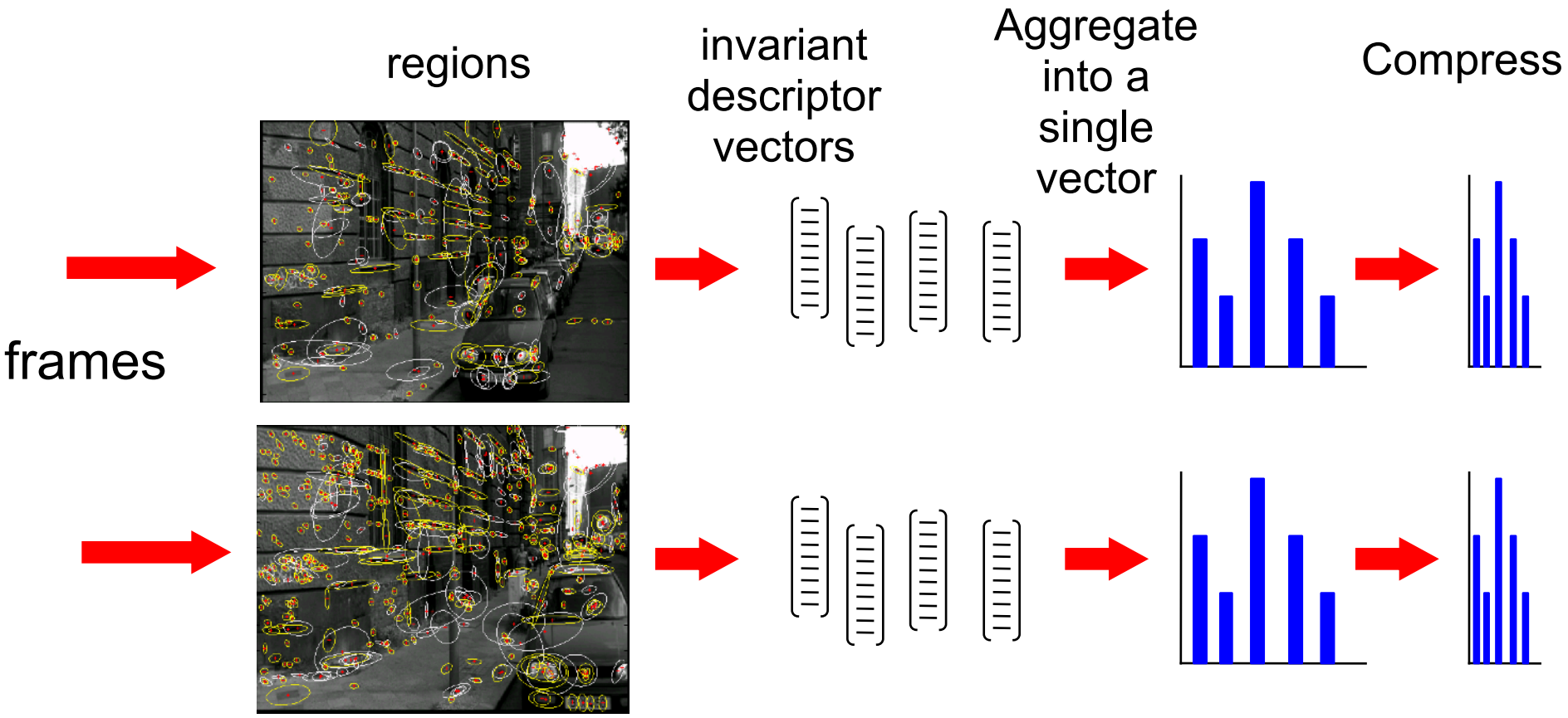
Strategy I: Efficient approximate NN search



Strategy II: Match histograms of visual words



Strategy II+: Match compressed vectors

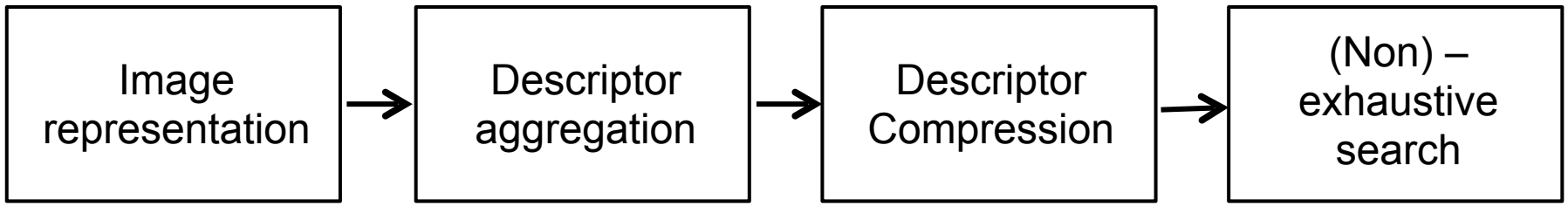


Slide credit: C. Schmid

Compact image representation

- Aim: improving the tradeoff between
  - search speed
  - memory usage
  - search quality

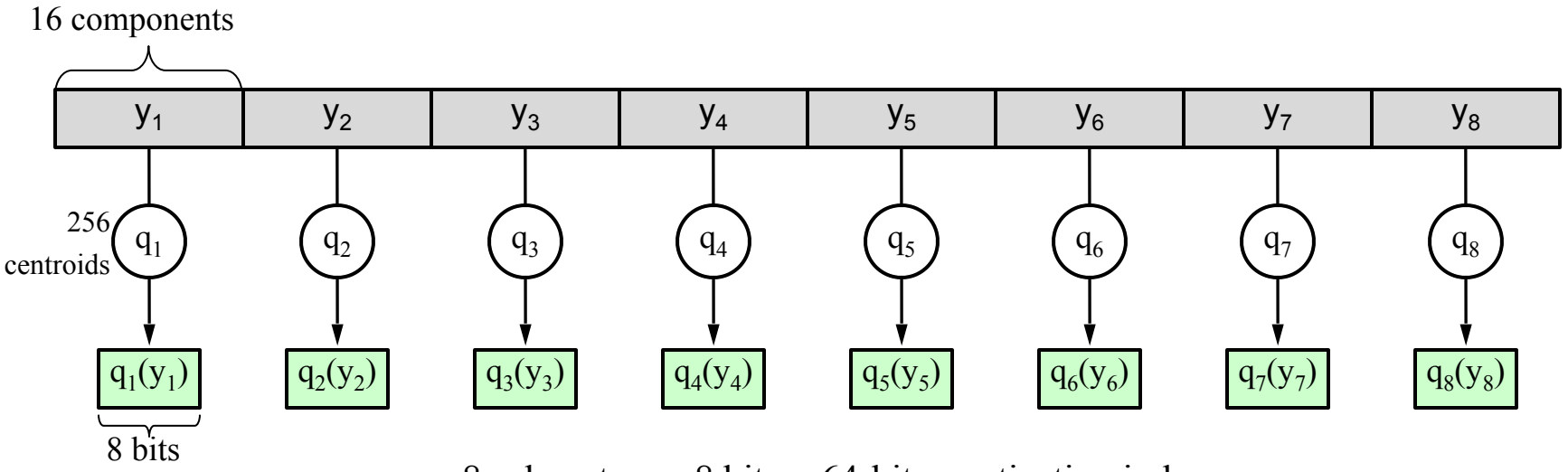
- Approach: joint optimization of three stages
  - descriptor aggregation
  - dimension reduction
  - indexing algorithm



Slide credit: C. Schmid

Product quantization for nearest neighbor search

- Vector split into *m* subvectors: $y \rightarrow \left[y_1 | \ldots | y_m\right]$
- Subvectors are quantized separately by quantizers $q(y) = \left[q_1(y_1) | \ldots | q_m(y_m)\right]$ where each q_i is learned by *k*-means with a limited number of centroids
- Example: y = 128-dim vector split in 8 subvectors of dimension 16
  - each subvector is quantized with 256 centroids -> 8 bit
  - very large codebook 256^8 ~ 1.8x10^19



⇒ 8 subvectors x 8 bits = 64-bit quantization index

[Jegou, Douze, Schmid, PAMI'11]

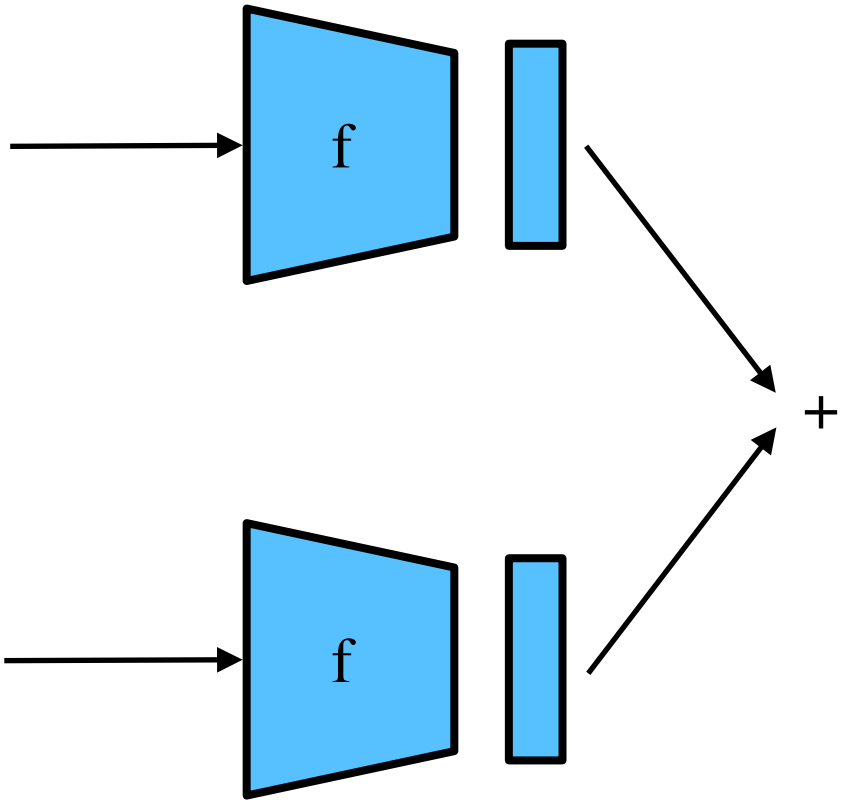
Slide credit: C. Schmid

Agenda: Efficient Visual Search

1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

“Learned” representations

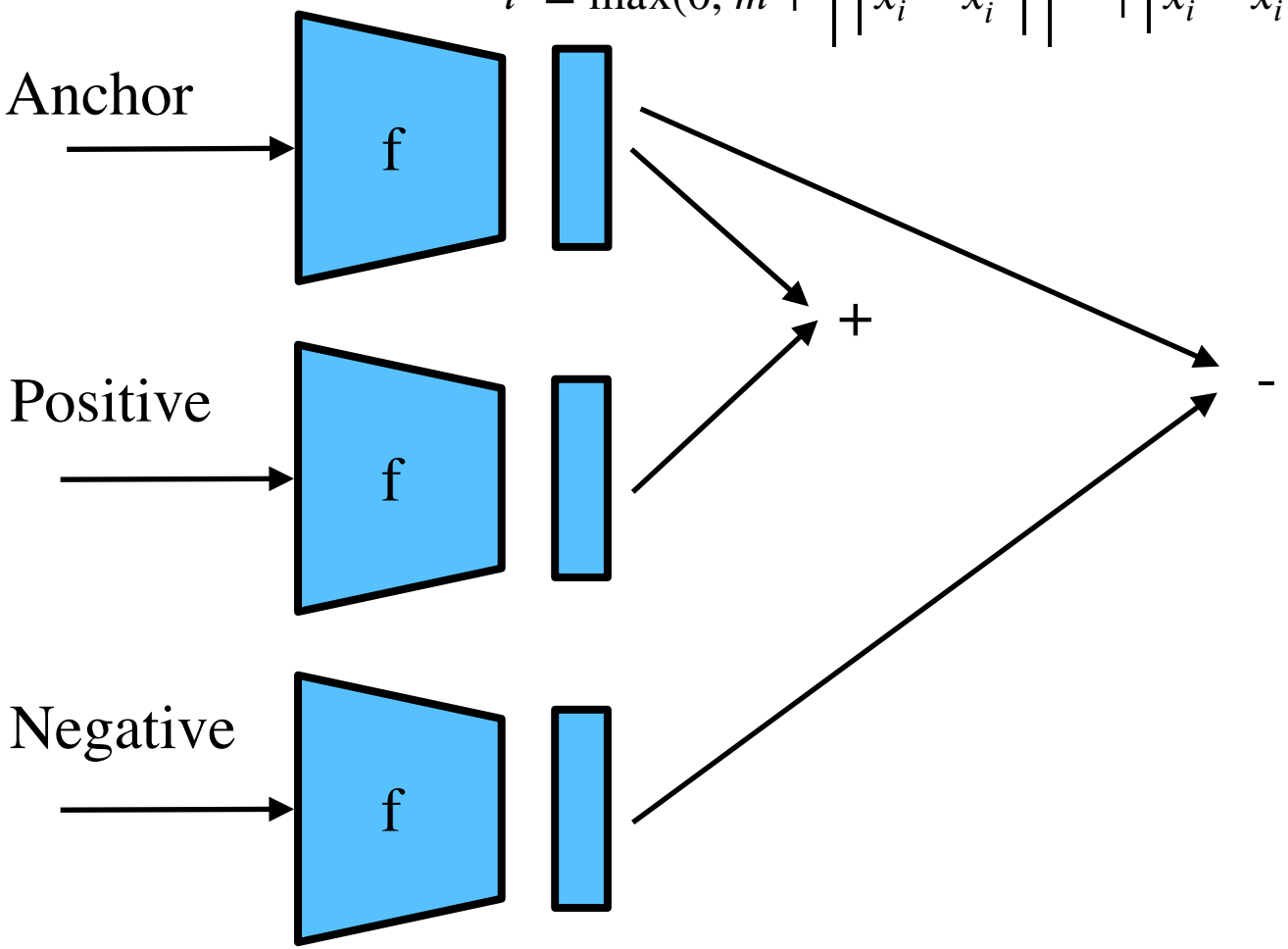
More details on upcoming lectures



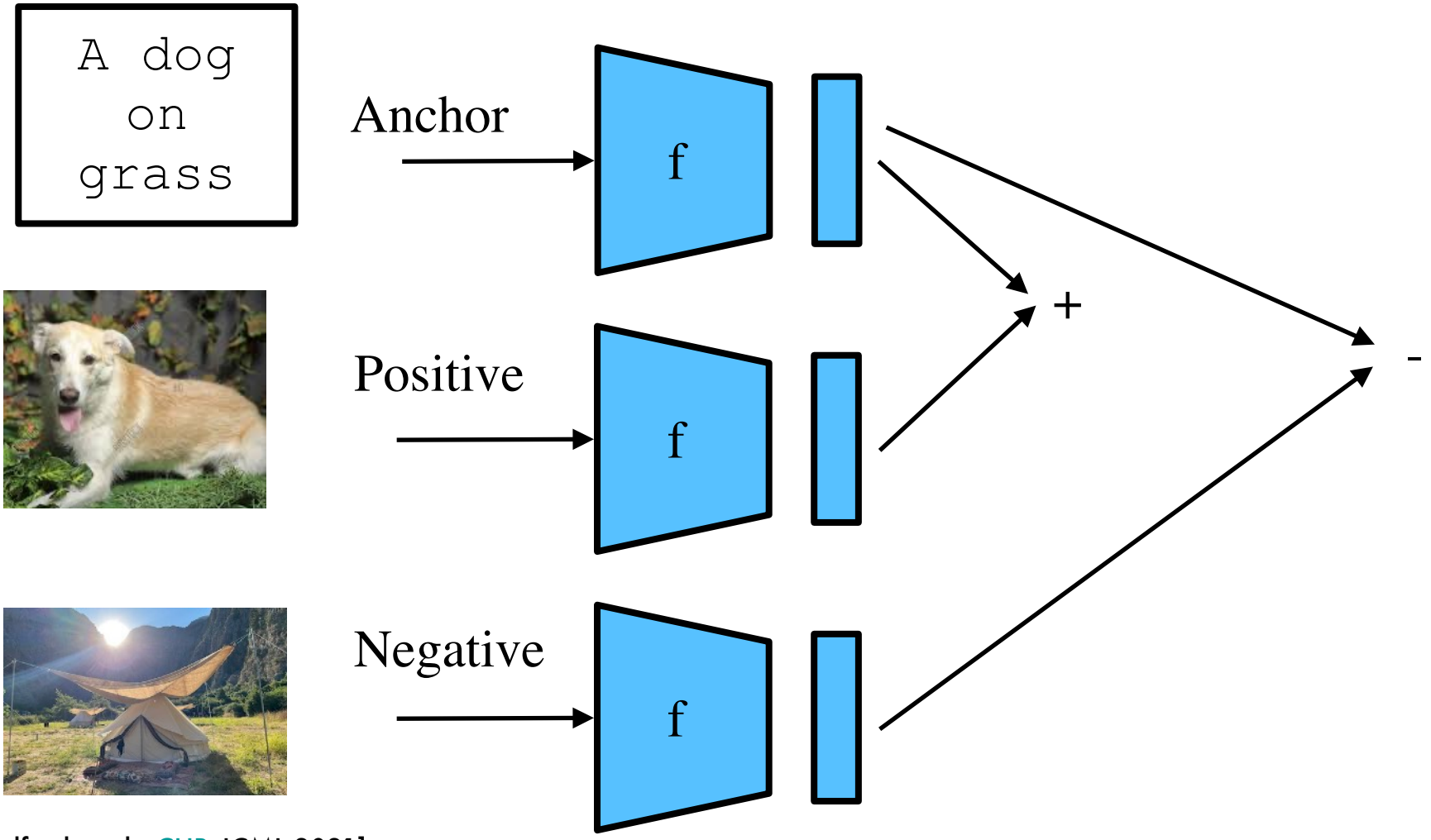
Learning to match: Image-to-image retrieval

Max-margin loss:
Distance to positive should be less than distance to negative more than a margin m.

$$l = \max(0, m + \left|\left|x_i - x_i^+\right|\right| - \left|\left|x_i - x_i^-\right|\right|)$$

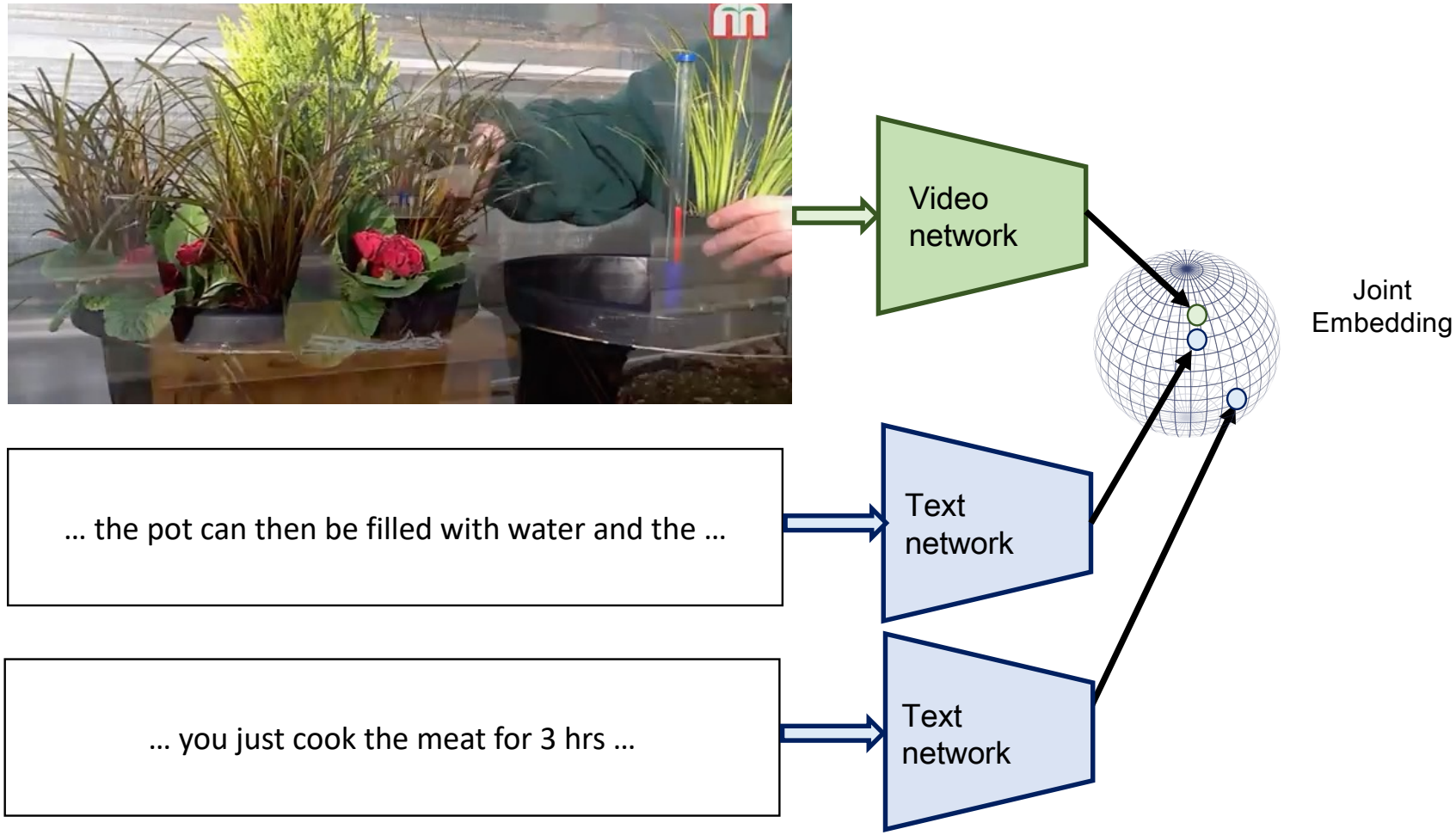


Learning to match: Text-to-image retrieval



[Radford et al., CLIP, ICML 2021]

Learning to match: Text-to-video retrieval



See also [Miech et al., 1706.06905, 1804.02516], and e.g. [Frome et al., NIPS 2013] [Gong et al., IJCV 2014]

“Metric Learning”

- Sometimes, you want to learn a distance. $d(x, y) = \|f(x) - f(y)\|$
- Distance (e.g. cosine) between intermediate features trained for another task often gives surprisingly good baseline for generic image comparison/distance.
- To train the function *f*, you typically define positive/negative relationship. (x_i, x_i^+, x_i^-)
- **Pairwise ranking loss**

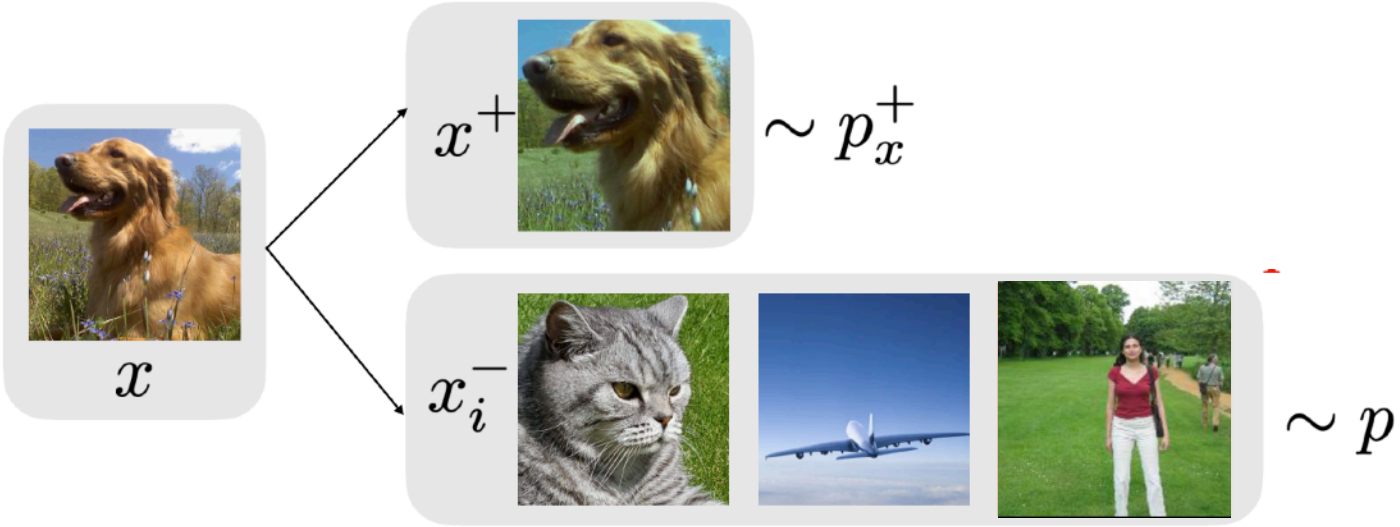
$$l = \begin{cases} \left|\left| x_i - x_i^+ \right|\right| & if\ positive\ pair \\ \max(0, m - \left|\left| x_i - x_i^- \right|\right|) & if\ negative\ pair \end{cases}$$

- **Triplet ranking loss**

$$l = \max(0, m + \left|\left| x_i - x_i^+ \right|\right| - |\left| x_i - x_i^- \right||)$$

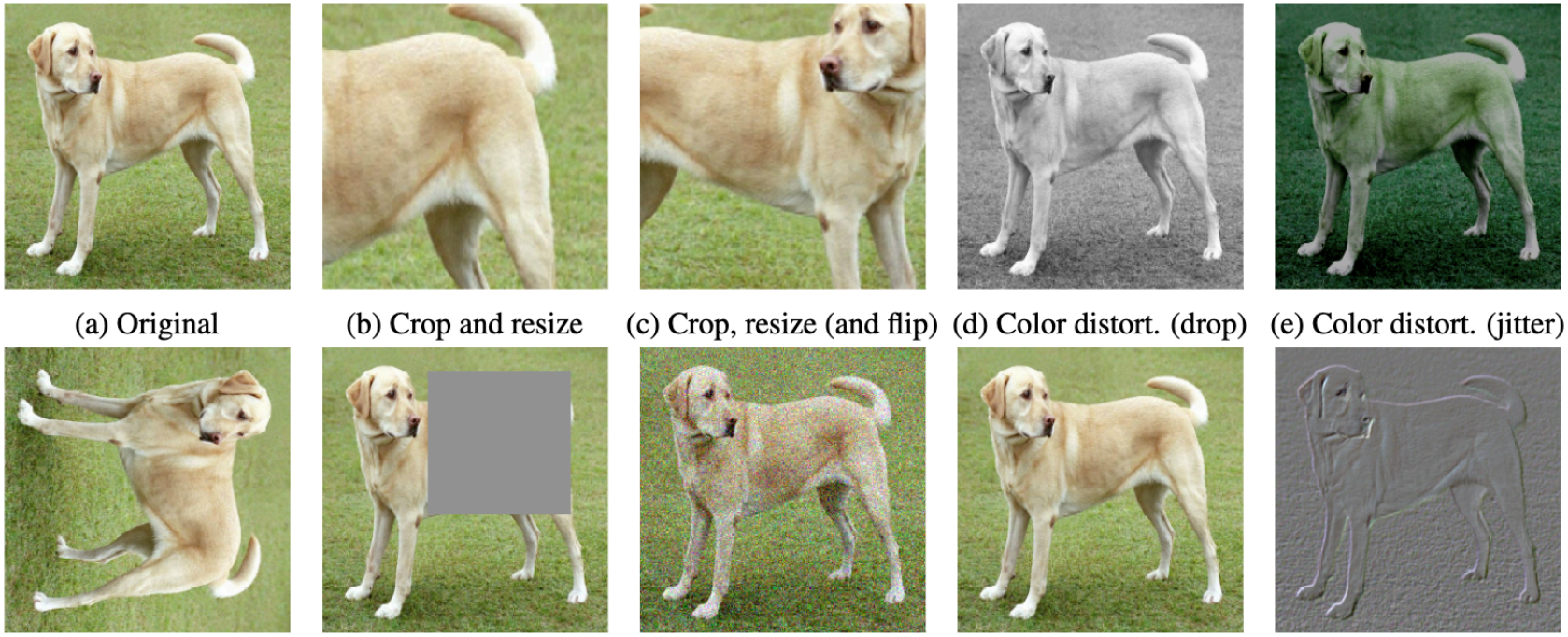
- Lots of terminology: Ranking loss, (max) margin loss, **contrastive** loss, triplet loss…
- Bonus: Noise Contrastive Estimation (NCE) and its variants are very popular today.

Metric Learning in Self-Supervised Setting



Why “self”-supervised?

Idea: make augmented versions of the image map to similar encodings



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

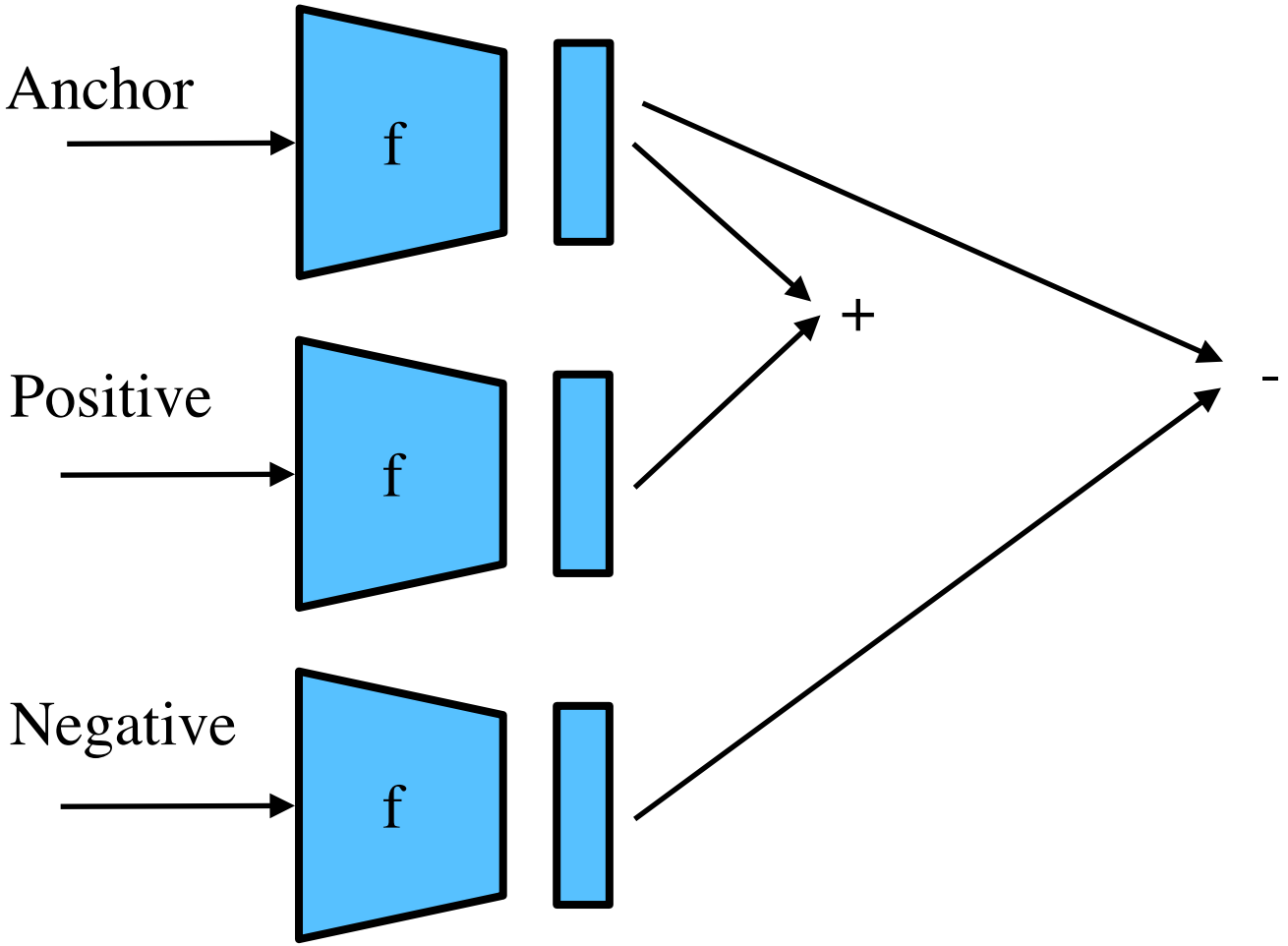
(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

A Simple Framework for Contrastive Learning of Visual Representations, Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton, ICML 2020.

Idea: make augmented versions of the image map to similar encodings

(a) Original

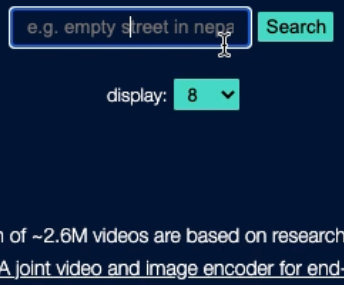
(f) Rotate $\{90°, 180°, 270°\}$

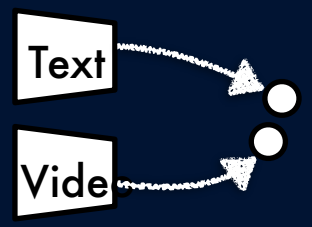


Example: Text Queries for Video Search:

[Bain, Nagrani, Varol, Zisserman, "Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval", ICCV 2021]

https://www.robots.ox.ac.uk/~vgg/research/frozen-in-time/





Need captioned videos for training

WebVid-2M Dataset

2.5M video-text pairs scraped from the web

"Runners feet in a sneakers close up. realistic three dimensional animation."

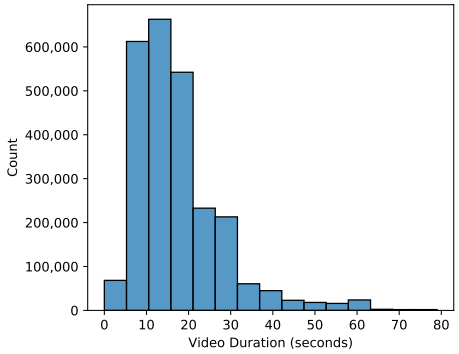
"Female cop talking on walkietalkie, responding emergency call, crime prevention"

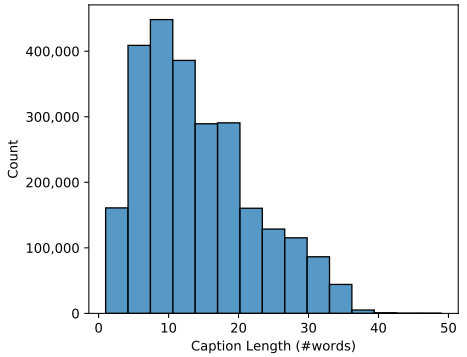
"Billiards, concentrated young woman playing in club"

"Lonely beautiful woman sitting on the tent looking outside. wind on the hair and camping on the beach near the colors of water and shore. freedom and alternative tiny house for traveler lady drinking"

"Kherson, ukraine - 20 may 2016: open, free, rock music festival crowd partying at a rock concert. hands up, people, fans cheering clapping applauding in kherson, ukraine - 20 may 2016. band performing"

"Cabeza de toro, punta cana/ dominican republic - feb 20, 2020: 4k drone flight over coral reef with manta"





What about speech in videos?

$\mathcal{P}$ Positive candidates

.60	it's quite a simple technique for
.53	beginners to learn and basically all I
.63	do is squeeze out three little circles
.49	then with the back of a teaspoon
.47	simply press the teaspoon into the

$\mathcal{P}$ Positive candidates

.50	main body of the laptop cover the
.63	duct tape with aluminum cover all
.61	remaining gaps edges with aluminum
.56	tape use the leftover poster board to
.50	create the keyboard keys I made my

$\mathcal{P}$ Positive candidates

.67	spinach what's the name
.57	keep it simple you just want to add
.58	fresh herbs maybe some oregano
.59	you can add cilantro basil they give
.50	it a couple more copies and when you

Self-supervised?

- HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips, Miech, Zhukov, Alayrac, Tapaswi, Laptev, Sivic. ICCV 2019.
- End-to-End Learning of Visual Representations from Uncurated Instructional Videos, Miech, Alayrac, Smaira, Laptev, Sivic, Zisserman. CVPR 2020.

Example: Text-to-video retrieval in sign language videos

Query Sentence: We're growing tomatoes almost all year round now.

Top 5 Retrieved Videos

(0.81)	(0.78)	(0.78)	(0.76)	(0.75)
				
(Unused) **We're growing tomatoes almost all year round now.**	To get the best performance from the **fruit** and the **plant**, we need to give the **plant** everything that it needs.	Because you need extra heat to **grow tomatoes** out of **season**, it makes them more expensive.	July to October were always the **months** to eat **tomatoes** but now vast heated greenhouses mean we can grow them between February and November.	So, why did you choose **tomatoes** as the main medium for this gene?

[Raude*, Prajwal*, Momeni*, Bull, Albanie, Zisserman, Varol, **arXiv 2024**]

Words in the lexical field of the query video subtitle are in blue.

Example: Video-to-text retrieval

Query Video → **Top 5 Retrieved Sentences** **(Similarity)**

(Unused) Subtitle:
Like getting dressed up on a Friday night?

1. Like getting dressed up on a Friday night? (0.73)
2. But the homecoming celebration won't begin until much later this evening. (0.69)
3. This is the festivities of the night. (0.67)
4. Friday nights in... ..with just the BBC Two of us. (0.67)
5. It really does look furious tonight. (0.66)

[Raude*, Prajwal*, Momeni*, Bull, Albanie, Zisserman, Varol, **arXiv 2024**]

Words in the lexical field of the query video subtitle are in blue.

Example: Sign Spotting

Given a **dictionary** video of an **isolated** sign of interest,
and a **continuous** sign language video **sequence**,
determine **where** the sign appears in a continuous sequence.

Dictionary input
Sign: "strong"

Continuous input

Localised sign
(coloured and slowed down x 2)

[Momeni*, Varol*, Albanie*, Afouras, Zisserman, *Watch, read and lookup: learning to spot signs from multiple supervisors*, **ACCV 2020**]

Can you spot the sign “***accept***”?

Dictionary input

Continuous input

Localised sign
(coloured and slowed down x 2)

Can you spot the sign “***happy***”?

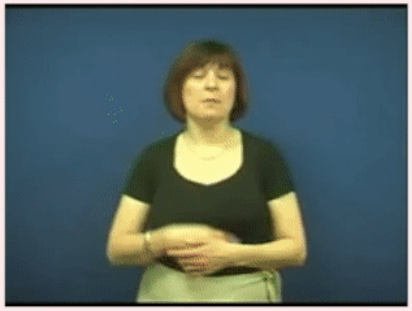
Dictionary input

Continuous input

Model detection prediction
(coloured and slowed down x 2)

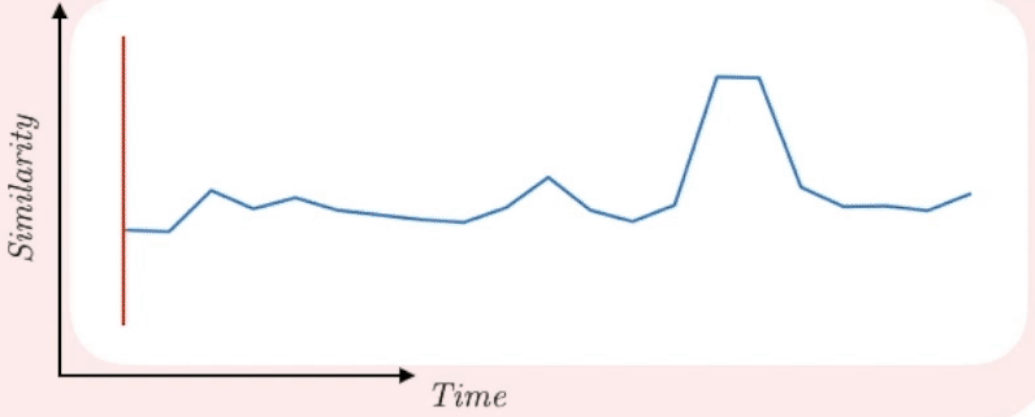
Example: Sign Spotting

Isolated sign "Apple"
from dictionary



Continuous signing

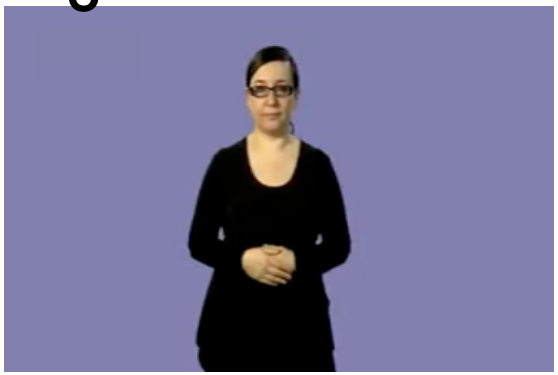
"We have some reasonable-sized apple trees in the garden."



[Momeni*, Varol*, Albanie*, Afouras, Zisserman, *Watch, read and lookup: learning to spot signs from multiple supervisors*, **ACCV 2020**]

Example: Sign Spotting
Indexing for content-based search

Sign: **Perfect**

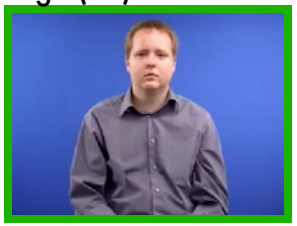


[Momeni*, Varol*, Albanie*, Afouras, Zisserman, *Watch, read and lookup: learning to spot signs from multiple supervisors*, **ACCV 2020**]

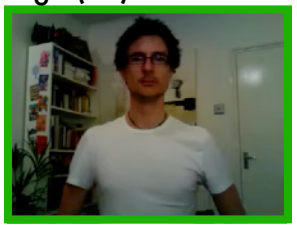
Example: Sign Spotting
Sign variant identification

Subtitle: "So it's important that this **business** is profitable to stay

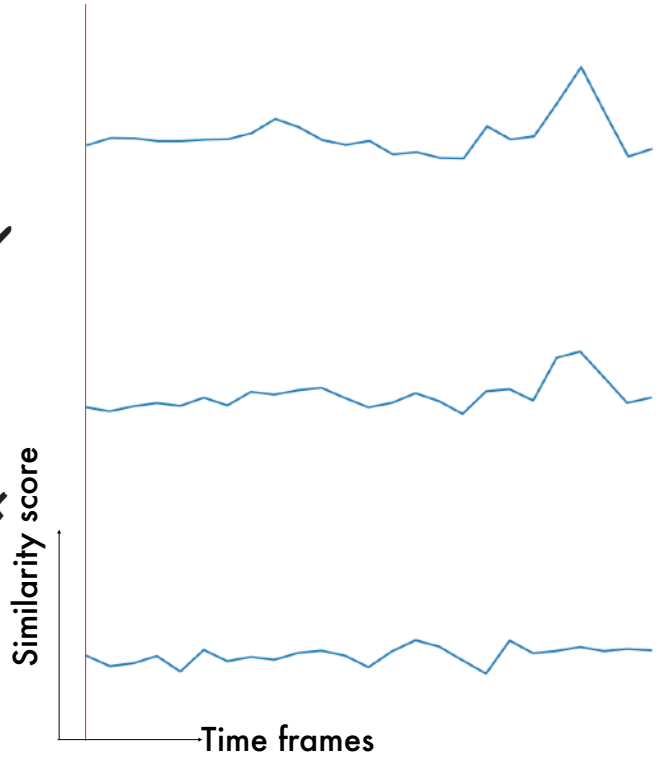
Sign (v1): **Business** ✔



Sign (v1): **Business** ✔



Sign (v2): **Business** ✘



[Momeni*, Varol*, Albanie*, Afouras, Zisserman, *Watch, read and lookup: learning to spot signs from multiple supervisors*, **ACCV 2020**]

Example: Sign Spotting

Identifying similar signs between two sign languages

British Sign Language (BSL)

Prepare

American Sign Language (ASL)

Vacation

"Faux amis": same/similar sign, different meaning

[Momeni*, Varol*, Albanie*, Afouras, Zisserman, *Watch, read and lookup: learning to spot signs from multiple supervisors*, **ACCV 2020**]

Example: Sign Spotting

Identifying similar signs between two sign languages

British Sign Language (BSL)

Stand

American Sign Language (ASL)

Stand

same/similar sign, same meaning

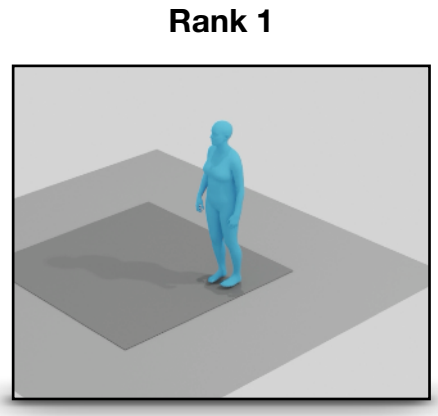
[Momeni*, Varol*, Albanie*, Afouras, Zisserman, *Watch, read and lookup: learning to spot signs from multiple supervisors*, **ACCV 2020**]

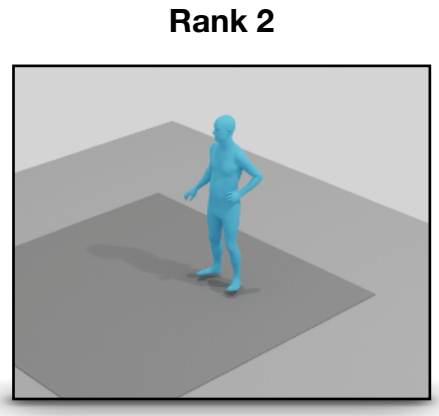
Example: Text-to-motion retrieval

Input: Text query

Someone is walking in a circle clockwise

Output: Ranking





Gallery of motions

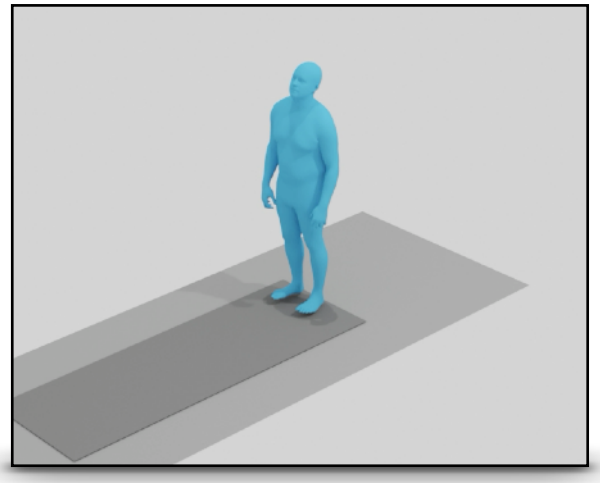
Example: Text-to-motion retrieval

A person begins to walk forward up the stairs

Qualitative results

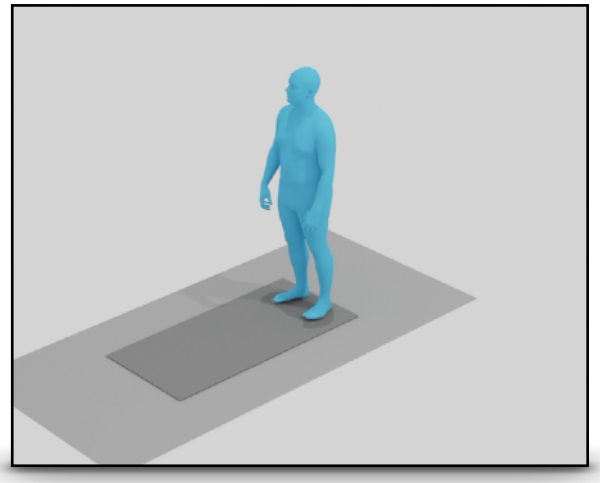
Searching on the whole test set of HumanML3D

R1 / S=0.91



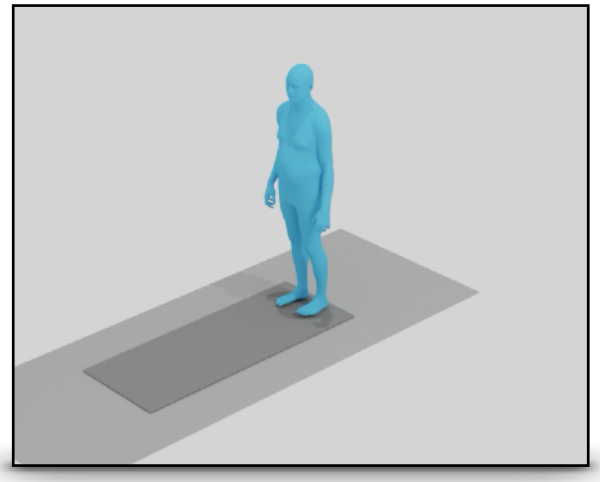
The person is stepping on something

R2 / S=0.90



A person walks up four steps with their hands by their sides and their lean forward slightly as they go up the stairs and once they've stopped going up the stairs, they straighten up again

R3 / S=0.89



A figure appears to climb stairs

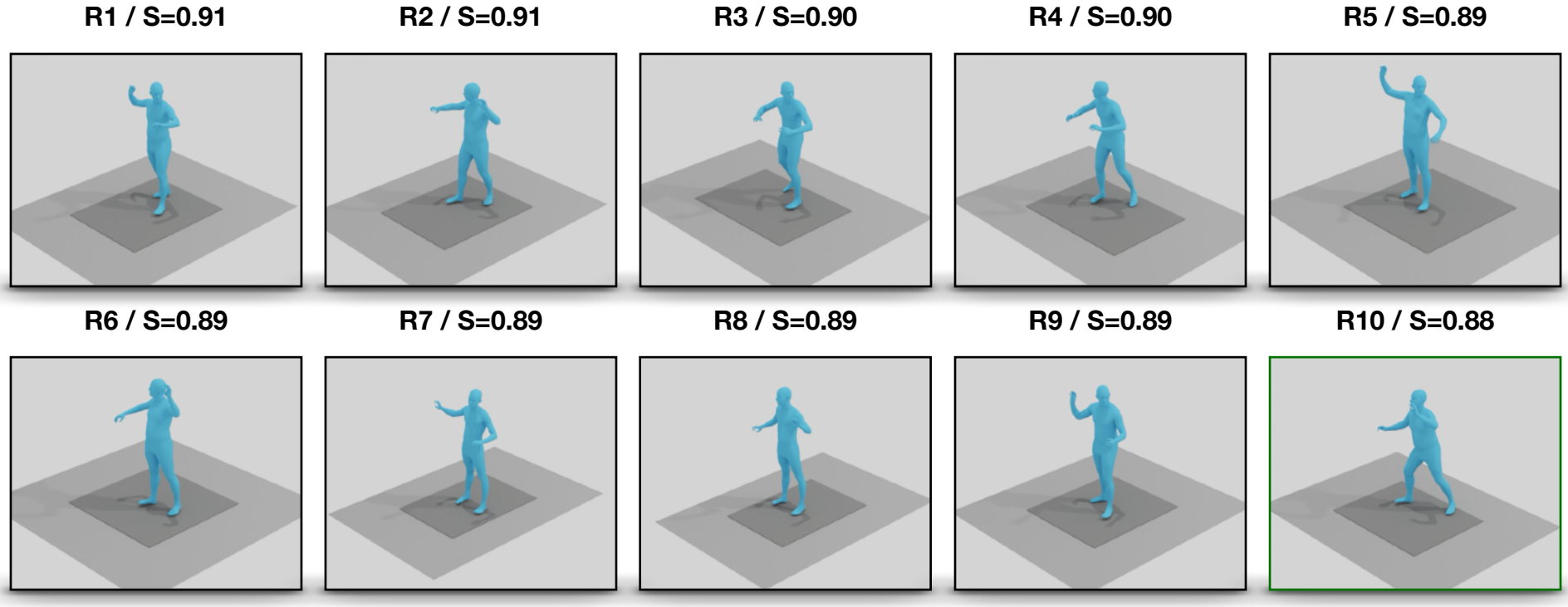
Note: Ground-truth text descriptions are displayed for illustration purposes, but are unused at test time.

Example: Text-to-motion retrieval

A person winds up his arm and then pitches a ball

Qualitative results

Searching on the whole test set of HumanML3D

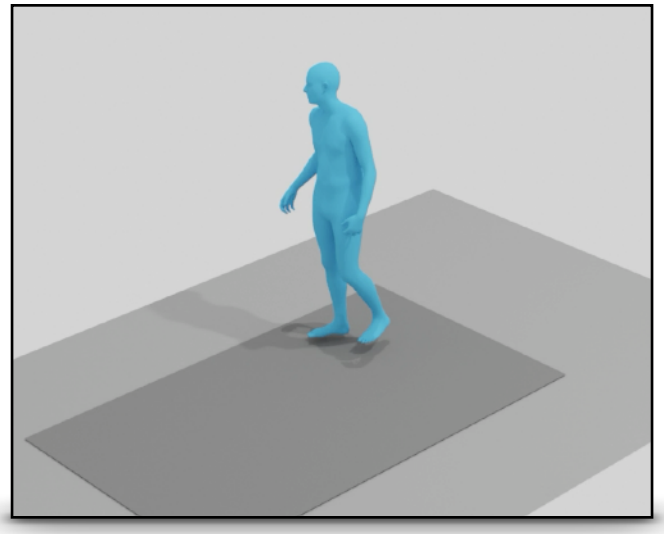


Example: Text-to-motion retrieval

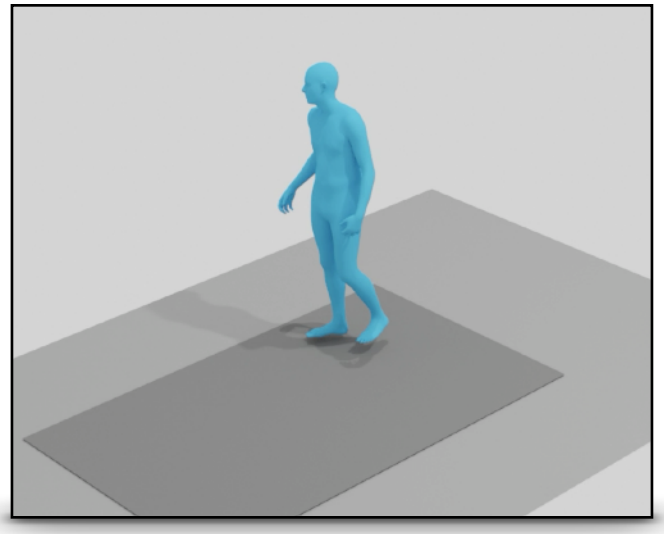
Use case: zero-shot moment retrieval

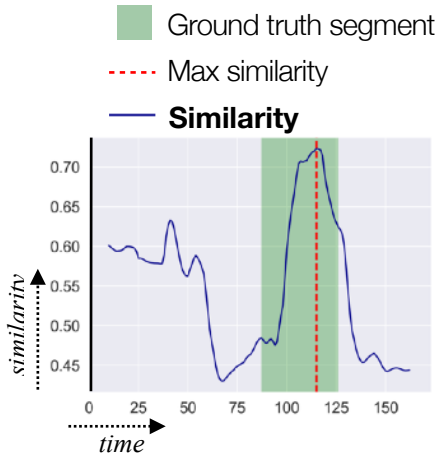
Back flip

TMR retrieval results



Ground truth segment in BABEL



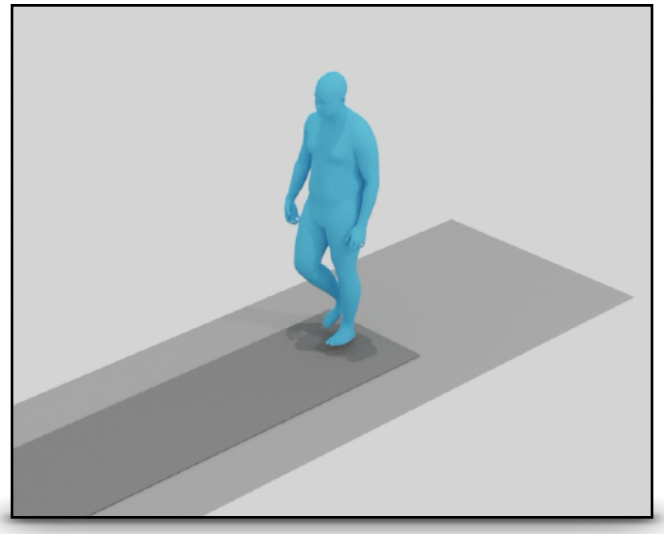


Example: Text-to-motion retrieval

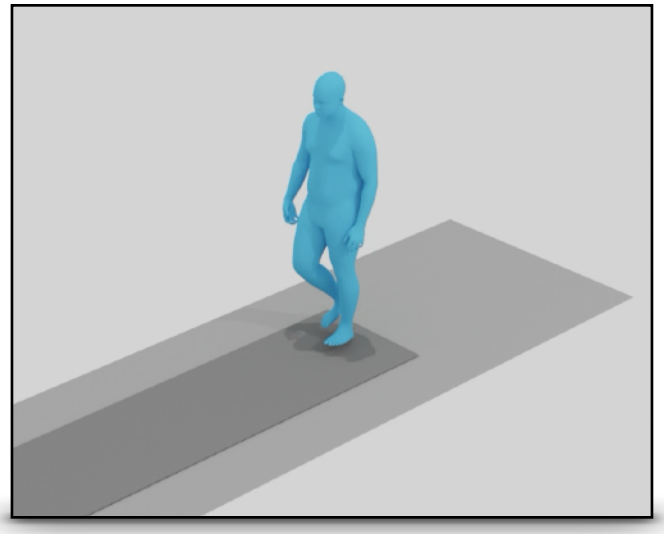
Use case: zero-shot moment retrieval

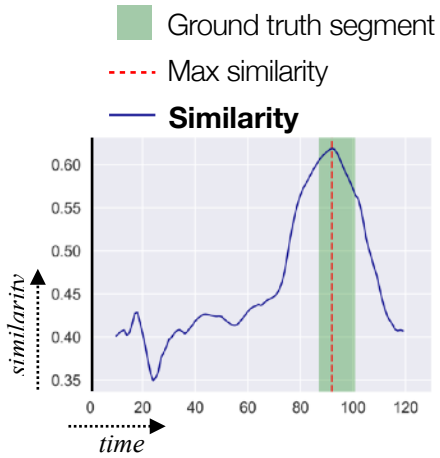
Pivots body 180 degrees

TMR retrieval results



Ground truth segment in BABEL





Agenda: Efficient Visual Search

1) Efficient matching of local descriptors
   - Approximate nearest neighbor search with kd-trees
   - Locality-sensitive hashing
2) Aggregate local descriptors into a single vector
   - Bag-of-visual-words
   - Query expansion
   - Product quantization
3) Examples from recent works

4) Final Project Topics