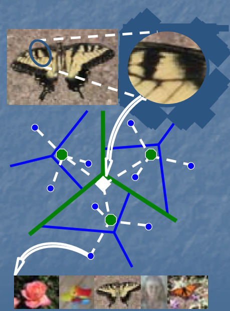
Scalable Recognition with a Vocabulary Tree



by:
David Nistér
Henrik Stewénius

presented by:
William Malpica

CS 395T

Some slides from Nister and Stewenius's CVPR 2006 presentation

Outline

- Abstract
- Strengths
- System Overview
- Animated explanation of the vocabulary tree
- Explanation of the scoring scheme
- Testing Results
- Conclusion

Scalable Recognition with a Vocabulary Tree

- The paper describes a system which can recognize objects from a very large database with great speed and recognition quality.
- The system uses local region descriptors which are hierarchically quantized in a vocabulary tree.

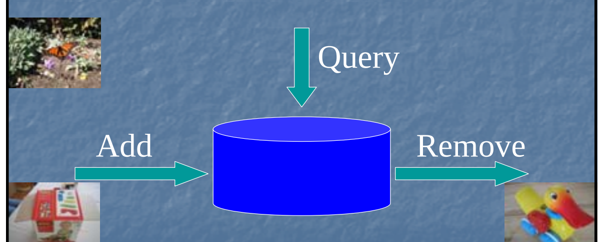
Strengths!

- The vocabulary tree directly defines the quantization.
  - Each high-dimension feature vector is quantized into an integer which corresponds to a path in the vocabulary tree.
  - Results in speed
    - Feature extraction on a 640x480 video frame in 0.2 sec. and database query in 25ms on a 50000 image database.
  - Results in compactness

Strengths!

- Potential for on-the-fly insertion
  - An offline unsupervised training stage is necessary to create the vocabulary, but new images can be added to the database on-the-fly.
  - Images can be added an the same rate as feature extraction.
  - Excellent benefit for large scalable image databases.

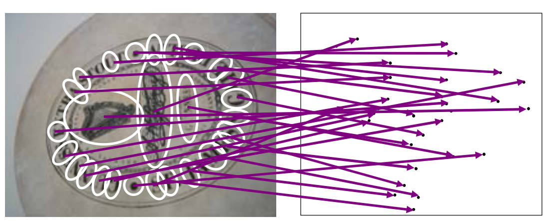
Adding, Querying and Removing Images at full speed

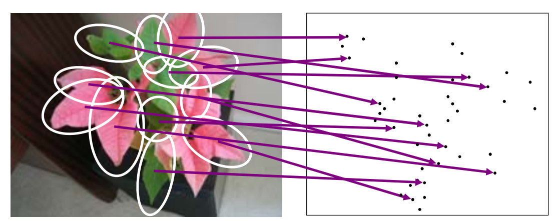


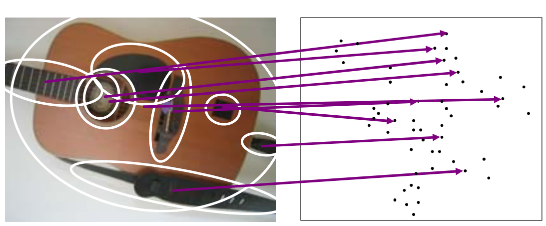
Training and Addition are Separate

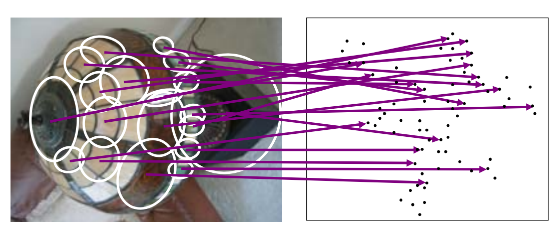
System Overview

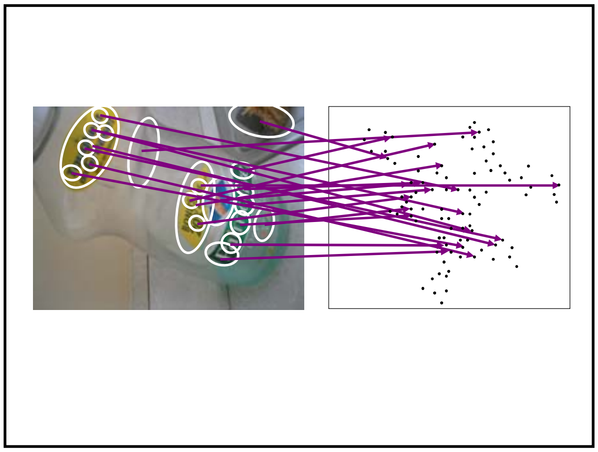
- Maximally Stable Extremal Regions (MSERs) feature extractor.
- SIFT feature descriptor
- Feature space is quantized through k-means clustering and build into a vocabulary tree.
- To retrieve images, a hierarchical scoring scheme is used based on Term Frequency Inverse Document Frequency (TF-IDF).

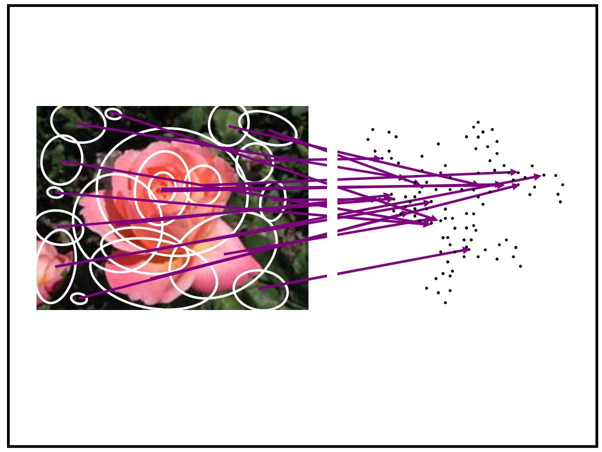


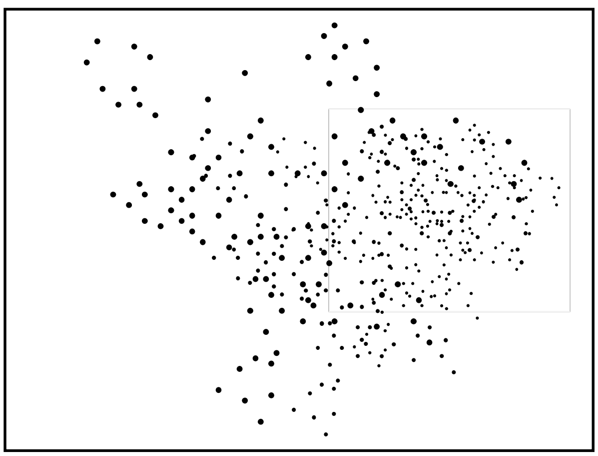


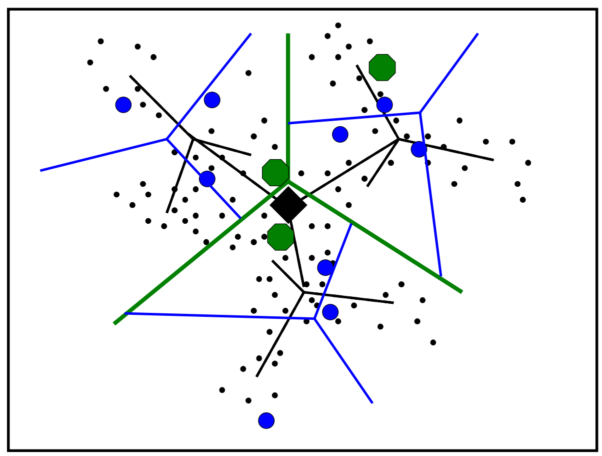


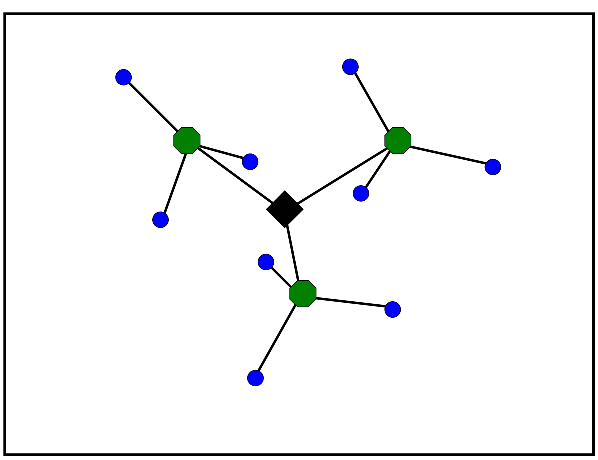


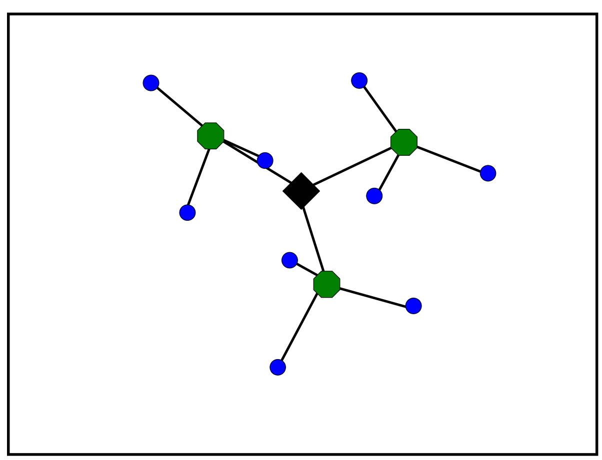


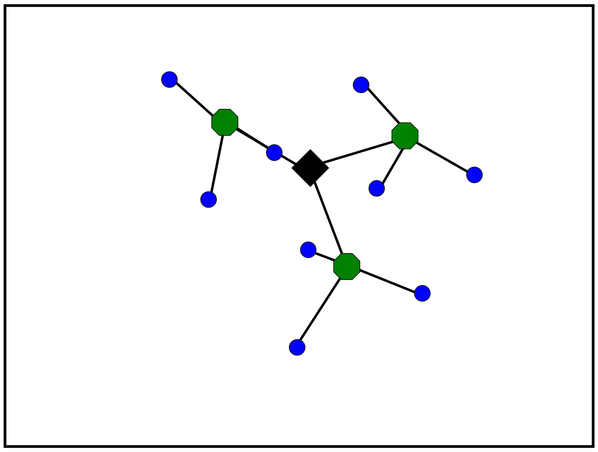
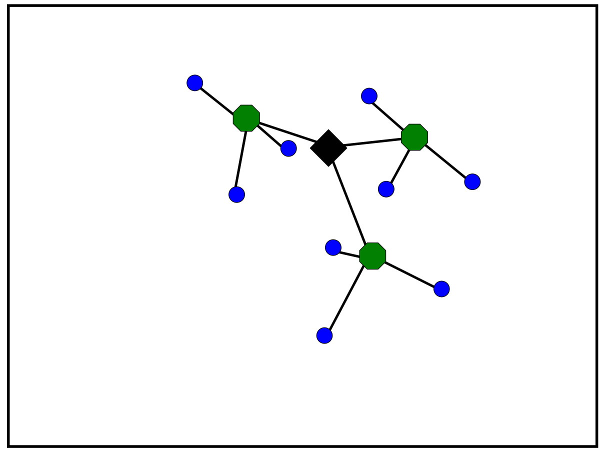
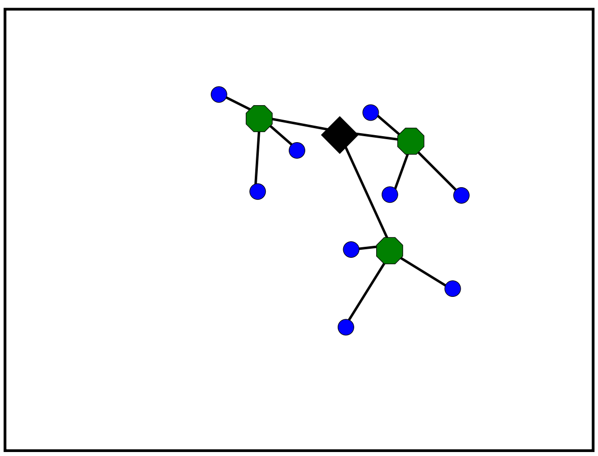
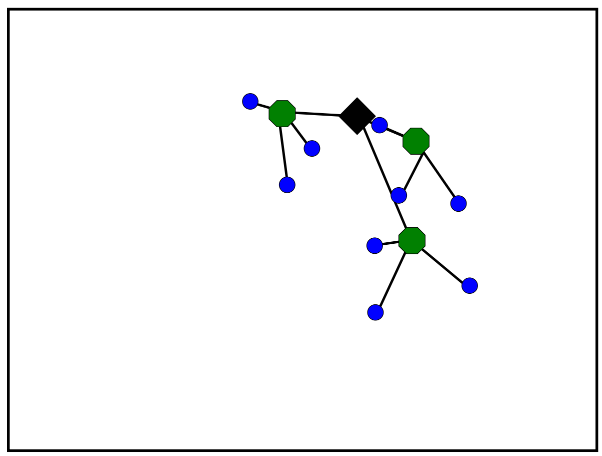
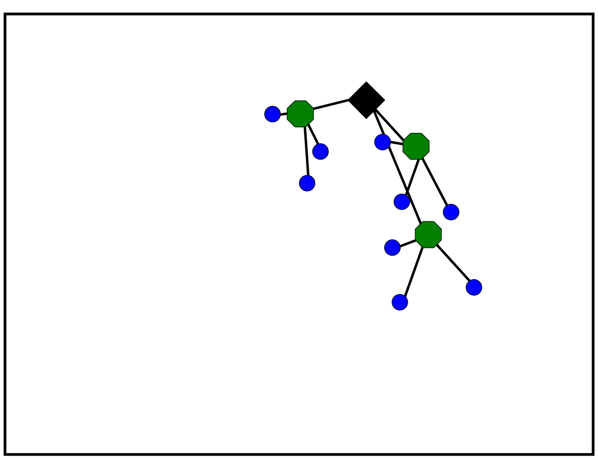
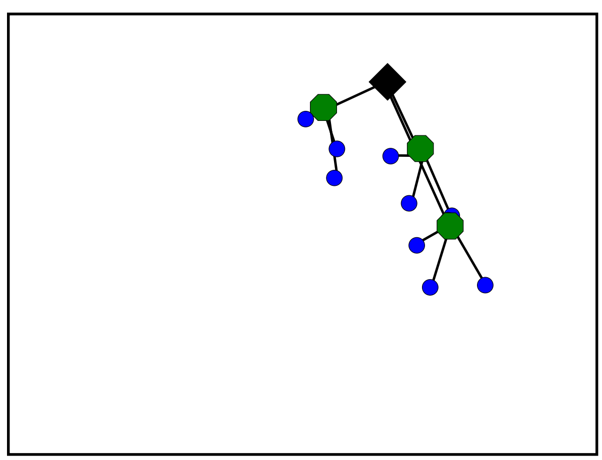


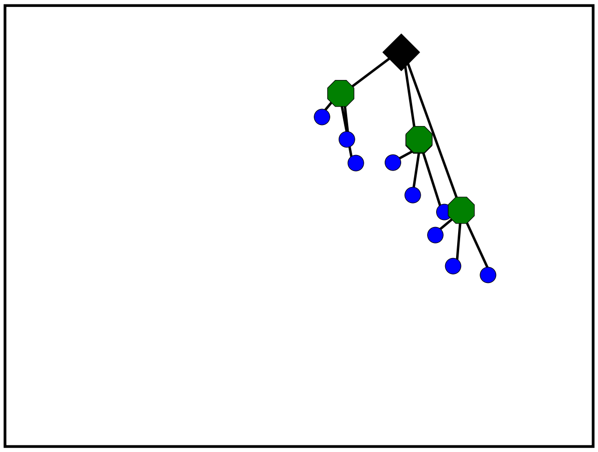


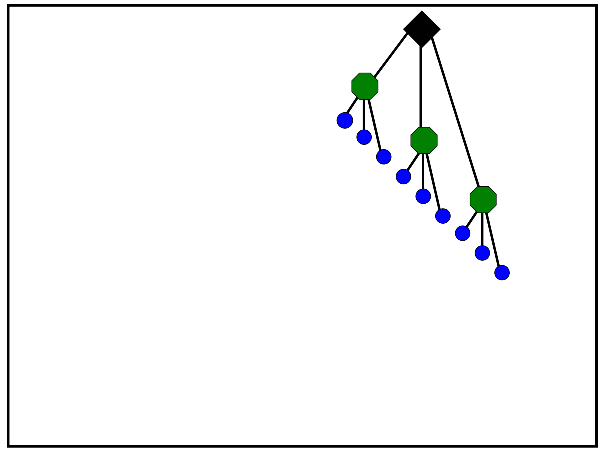


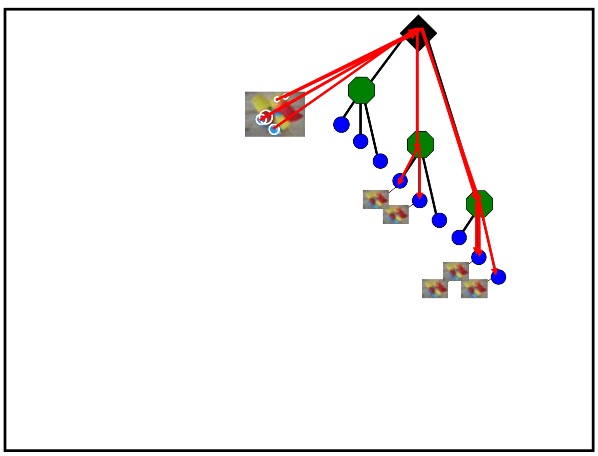


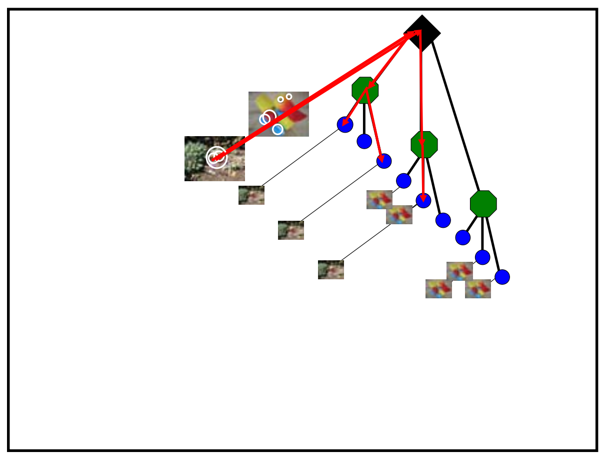


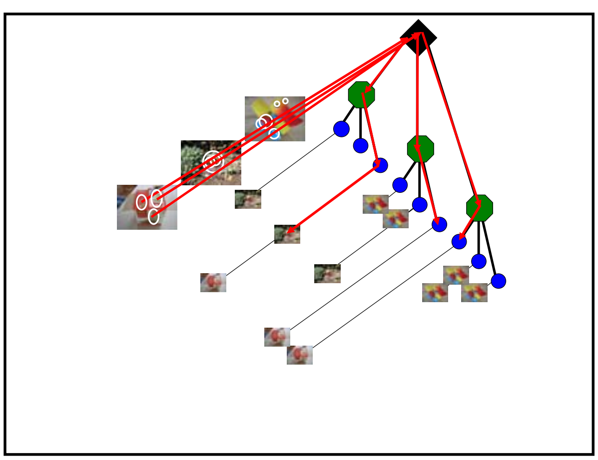







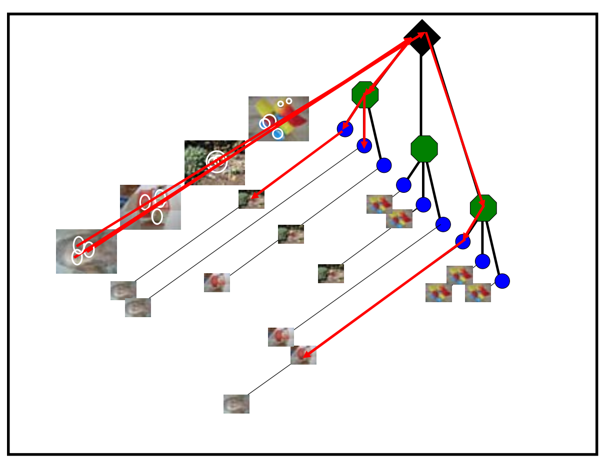


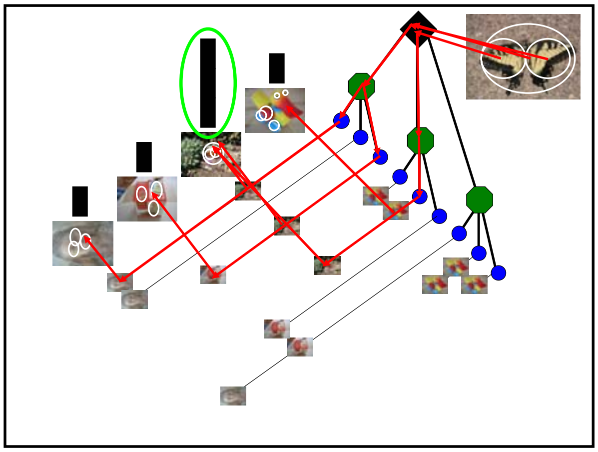












Definition of Scoring

- Weights are assigned to each node (with certain exceptions) $$w_i = \ln \frac{N}{N_i} \quad (1)$$
- Query and database vectors are defined according to their assigned weights $$q_i = n_i w_i \quad (2)$$ $$d_i = m_i w_i \quad (3)$$
- Each database image is given a relevance score based on the normalized differences between the query and database vectors $$s(q,d) = \left\| \frac{q}{\|q\|} - \frac{d}{\|d\|} \right\| \quad (4)$$

Implementation of Scoring

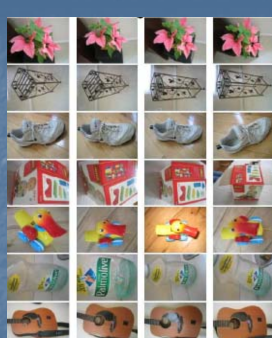
- Every node is associated with an inverted file, although only leaf nodes are explicitly represented. Inner nodes are a concatenation of the leaf nodes.
- Inverted files store the id-numbers of the images in which a particular node occurs, and the term frequency for that image.
- The vectors representing the database images as well as the query images are normalized to unit magnitude.

Normalization

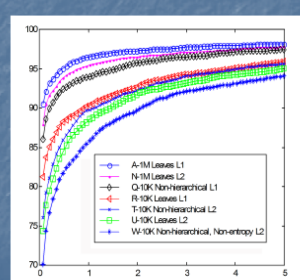
- To compute the normalized difference in Lp-norm:
$$\|\mathbf{q}-\mathbf{d}\|_p^p = \sum_i |q_i - d_i|^p \quad (5)$$
$$\|\mathbf{q}-\mathbf{d}\|_p^p = 2 + \sum_{i|q_i \neq 0, d_i \neq 0} (|q_i - d_i|^p - |q_i|^p - |d_i|^p) \quad (6)$$
- For the case of the L2-norm:
$$\|\mathbf{q}-\mathbf{d}\|_2^2 = 2 - 2 \sum_{i|q_i \neq 0, d_i \neq 0} q_i d_i \quad (7)$$

Testing

- Ground truth database consisted of 6376 images in groups of four.
- The database was queried with every image and was evaluated on how frequently the other three images are found perfectly.



Results for only 1400 images



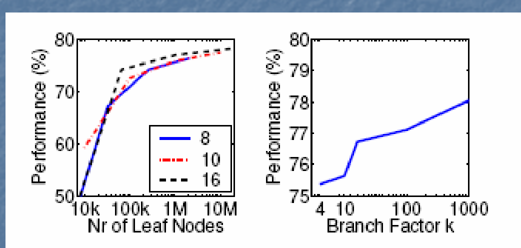
Results for only 1400 images

Me	En	No	S%	Voc-Tree	Le	Eb	Perf
A	**y/y**	**L1**	**0**	**6x10=1M**	**1**	**ir**	**90.6**
B	y/y	L1	0	6x10=1M	1	vr	90.6
C	y/y	L1	0	6x10=1M	2	ir	90.4
D	n/y	L1	0	6x10=1M	2	ir	90.4
E	y/n	L1	0	6x10=1M	2	ir	90.4
F	n/n	L1	0	6x10=1M	2	ir	90.4
G	n/n	L1	0	6x10=1M	1	ir	90.2
H	y/y	L1	m2	6x10=1M	1	ir	90.0
I	y/y	L1	0	6x10=1M	3	ir	89.9
J	y/y	L1	0	6x10=1M	4	ir	89.9
K	y/y	L1	0	6x10=1M	2	vr	89.8

Results for only 1400 images

L	y/y	L1	0	6x10=1M	2	ip	89.0
M	y/y	L1	m5	6x10=1M	1	ir	89.1
N	**y/y**	**L2**	**0**	**6x10=1M**	**1**	**ir**	**87.9**
O	y/y	L2	0	6x10=1M	2	ir	86.6
P	y/y	L1	l10	6x10=1M	2	ir	86.5
Q	**y/y**	**L1**	**0**	**1x10K=10K**	**1**	**-**	**86.0**
R	**y/y**	**L1**	**0**	**4x10=10K**	**2**	**ir**	**81.3**
S	y/y	L1	0	4x10=10K	1	ir	80.9
T	**y/y**	**L2**	**0**	**1x10K=10K**	**1**	**-**	**76.0**
U	**y/y**	**L2**	**0**	**4x10=10K**	**1**	**ir**	**74.4**
V	y/y	L2	0	4x10=10K	2	ir	72.5
W	**n/n**	**L2**	**0**	**1x10K=10K**	**1**	**-**	**70.1**

Results with full 6376 image database



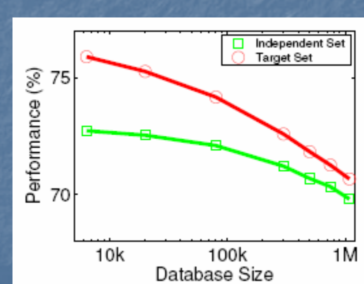
Other Tests – 40000 CD covers

- Method was tested on a database of 40000 CD covers running real-time.

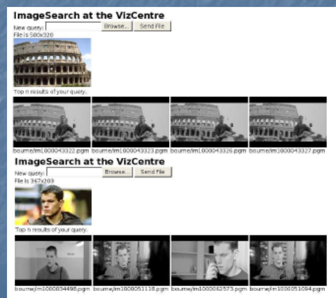
Other Tests – 1 million images

- Method was also tested on a database of 1 million images. The ground truth images were embedded into a database containing all the frames from several movies: The Bourne Identity, The Matrix, Braveheart, Collateral, Resident Evil, Almost Famous and Monsters Inc.
- Queries on a 8GB machine would take about 1 second. Database creation took 2.5 days.

Other Tests – 1 million images



Other Tests – Non movie images queried on 300K frames



Conclusion

- This methodology provides the abililty to make fast searches on extremely large databases.
- Paves the way to someday create an internet-scale content based image search engine.

Questions