

# Keypoint-based Recognition and Object Search

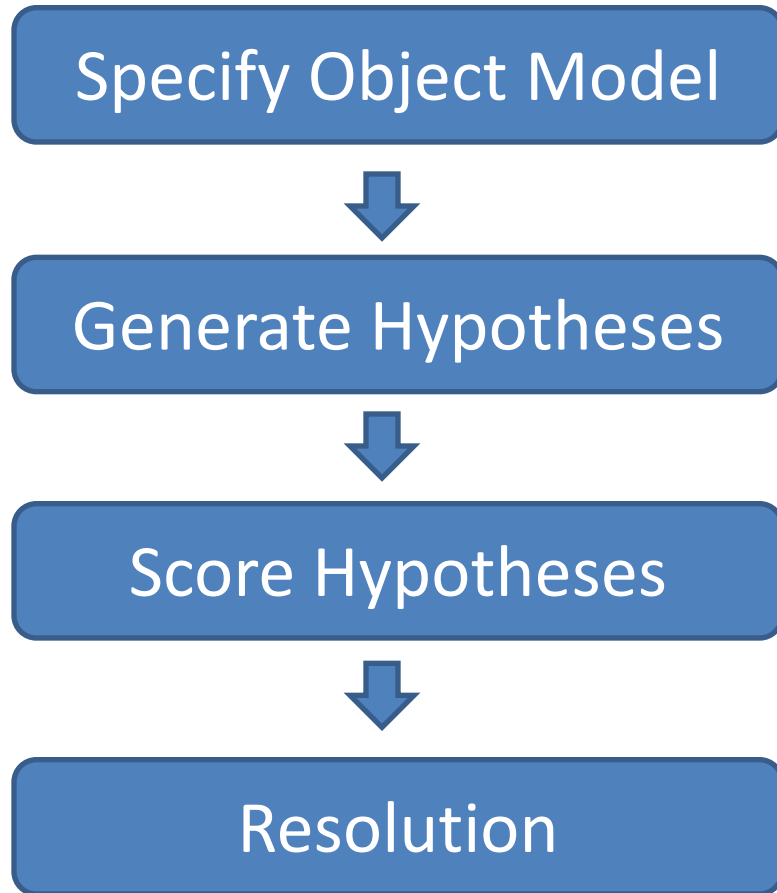
Computer Vision  
CS 543 / ECE 549  
University of Illinois

Derek Hoiem

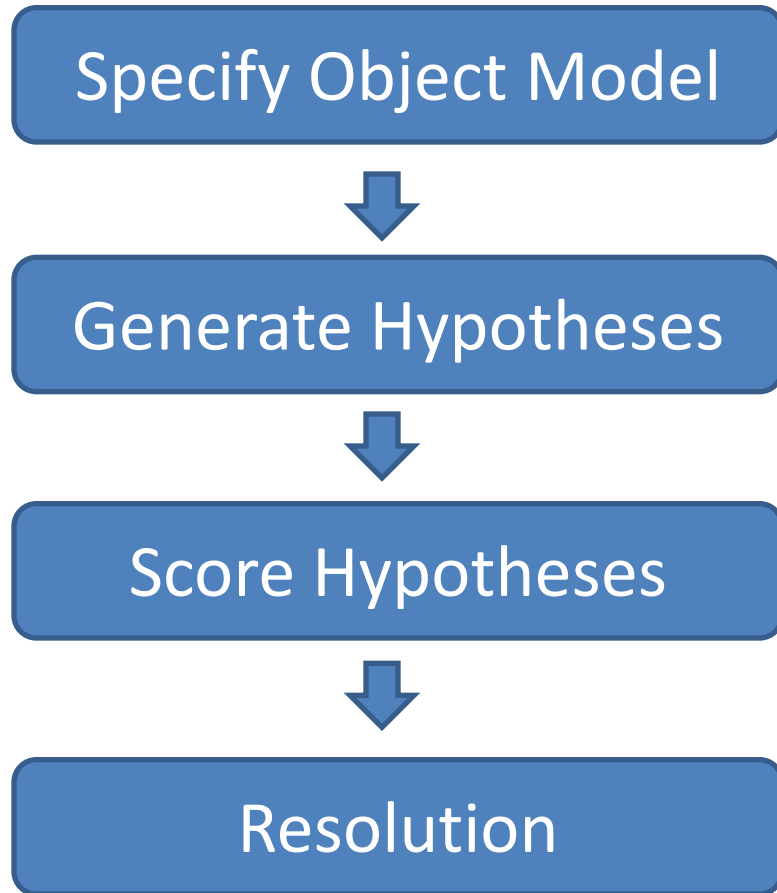
# Notices

- I'm having trouble connecting to the web server, so can't post lecture slides right now
- HW 2 due Thurs
- Guest lecture Thurs by Ali Farhadi

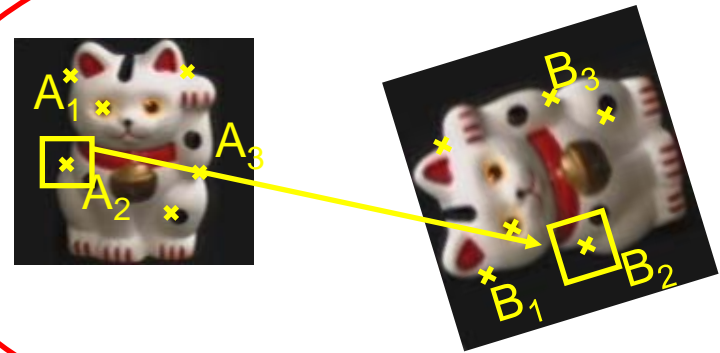
# General Process of Object Recognition



# General Process of Object Recognition

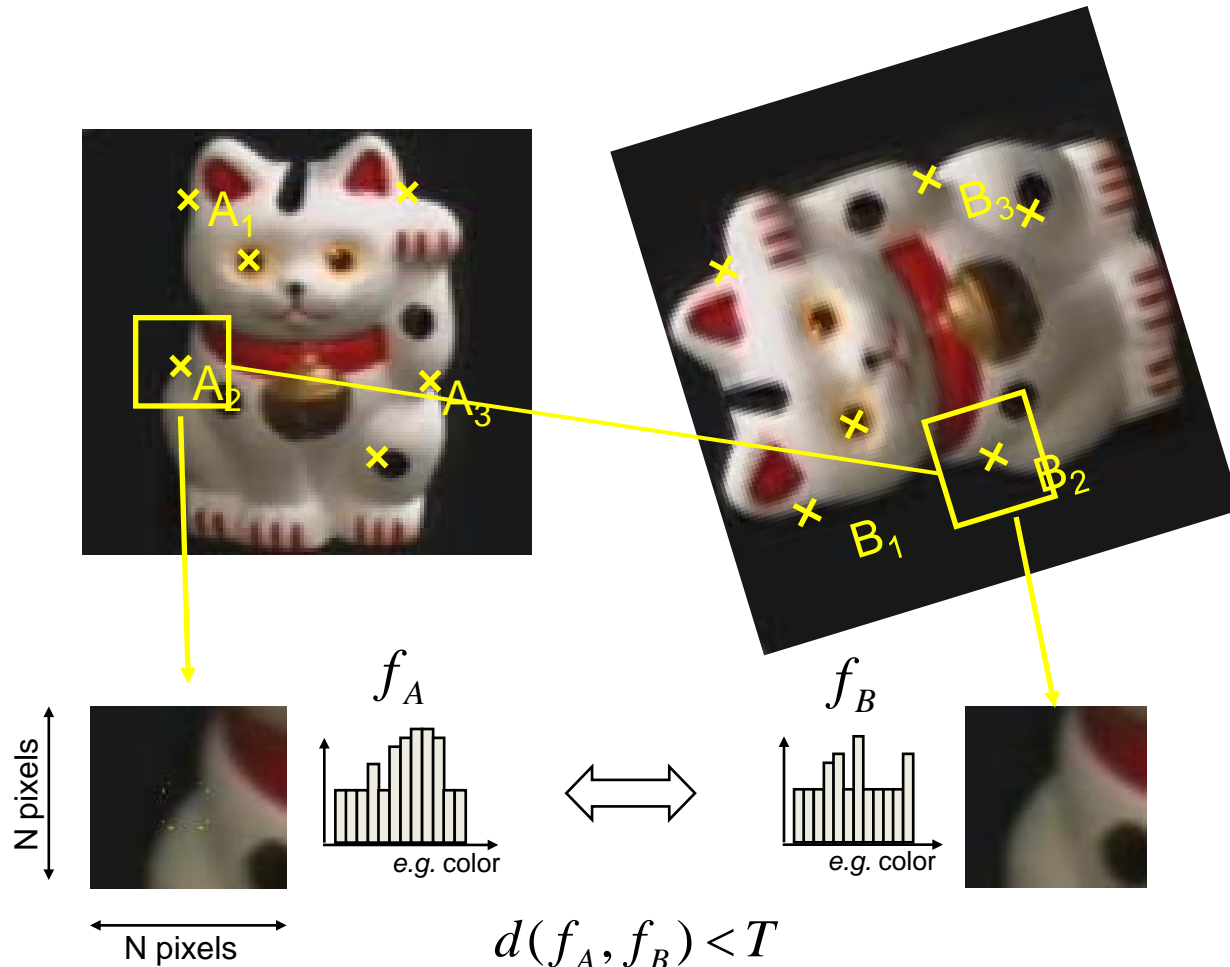


Example: Keypoint-based  
Instance Recognition



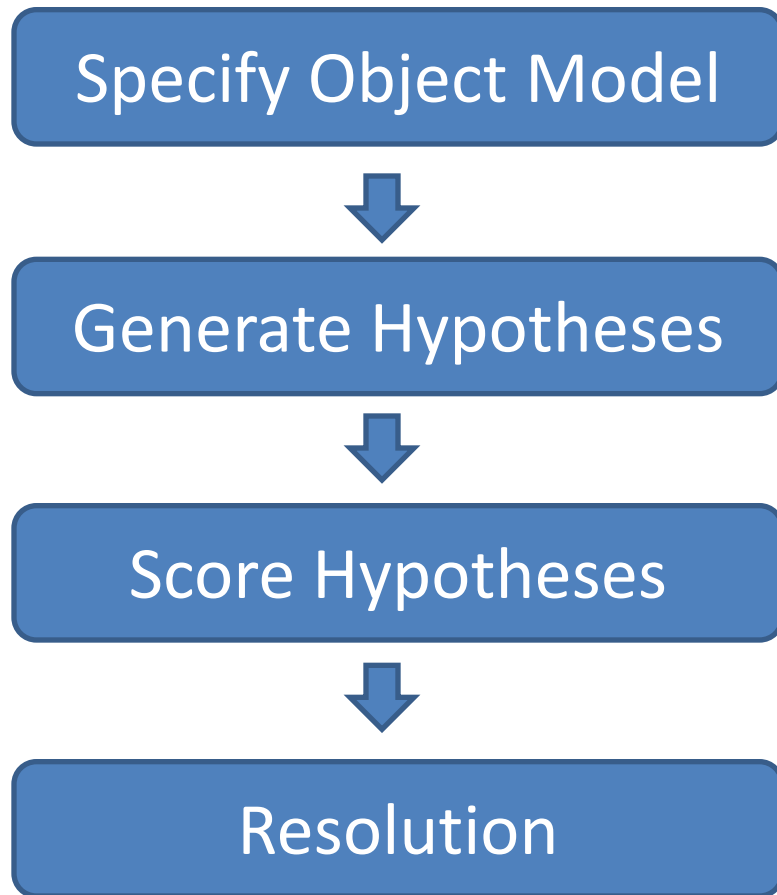
**Last Class**

# Overview of Keypoint Matching

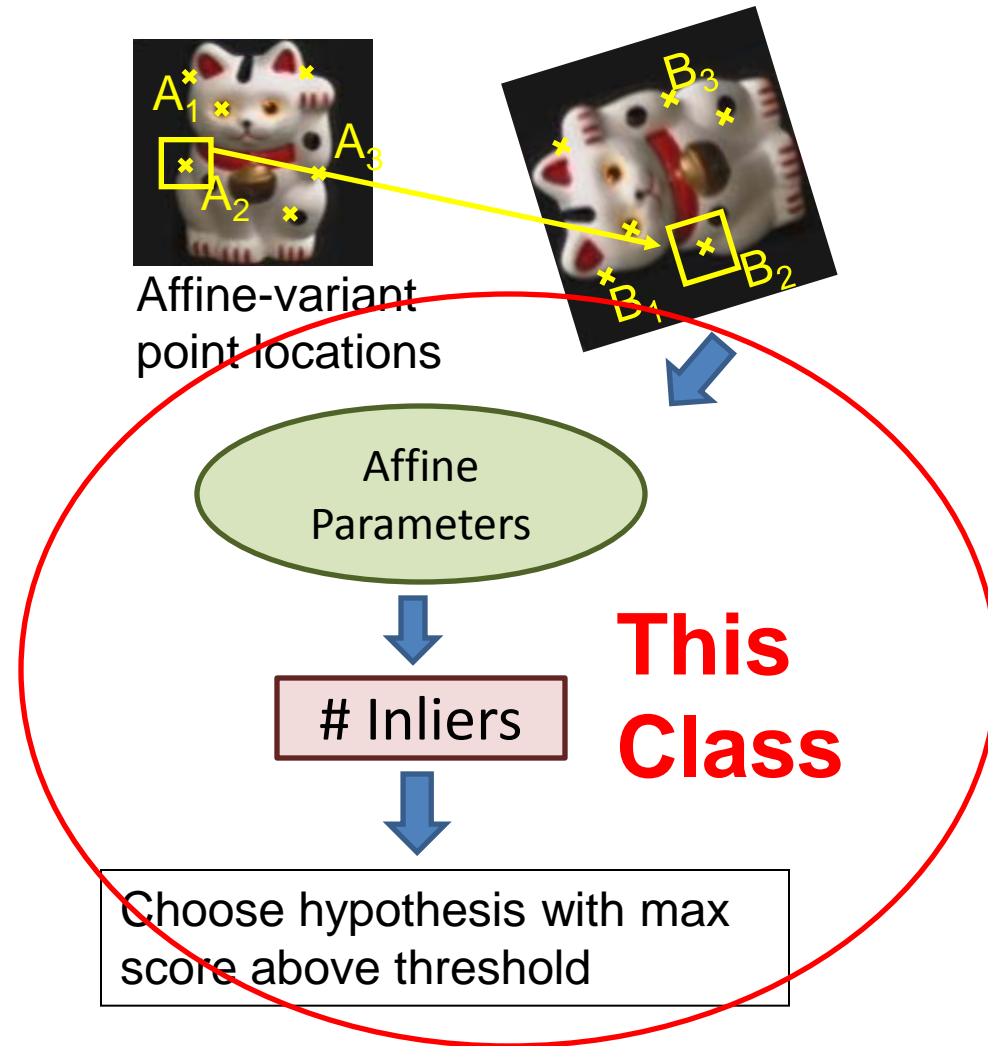
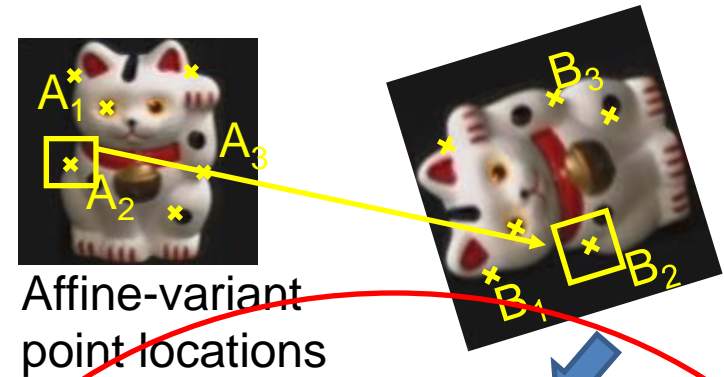


1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

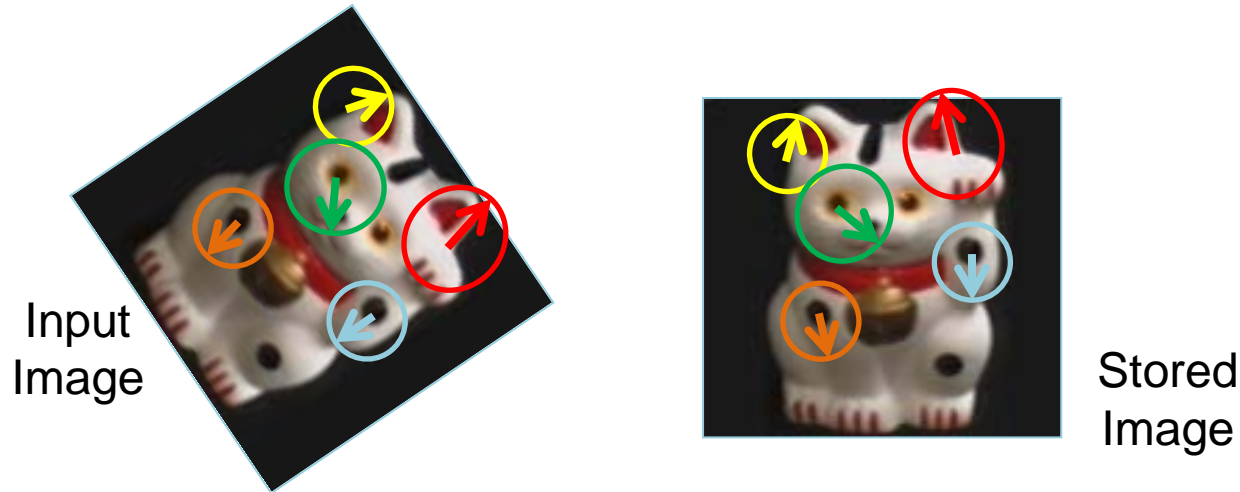
# General Process of Object Recognition



## Example: Keypoint-based Instance Recognition



# Finding the objects (overview)



1. Match interest points from input image to database image
2. Matched points vote for rough position/orientation/scale of object
3. Find triplets of position/orientation/scale that have at least three votes
4. Compute affine registration and matches using iterative least squares with outlier check
5. Report object if there are at least  $T$  matched points

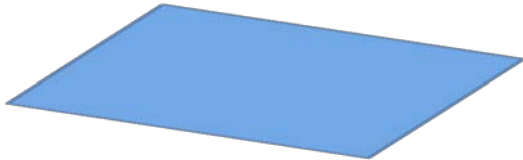
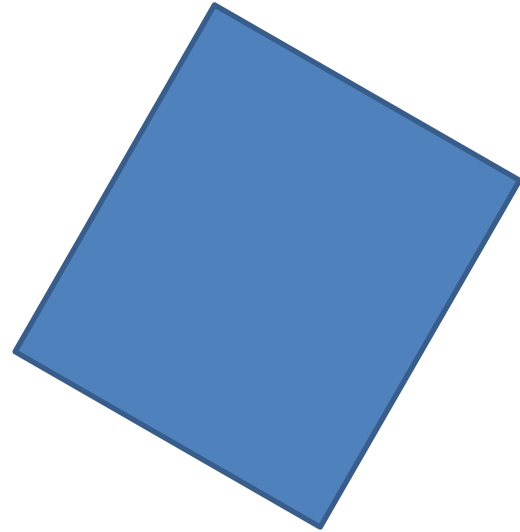
# Matching Keypoints

- Want to match keypoints between:
  1. Query image
  2. Stored image containing the object
- Given descriptor  $x_0$ , find two nearest neighbors  $x_1, x_2$  with distances  $d_1, d_2$
- $x_1$  matches  $x_0$  if  $d_1/d_2 < 0.8$ 
  - This gets rid of 90% false matches, 5% of true matches in Lowe's study



# Affine Object Model

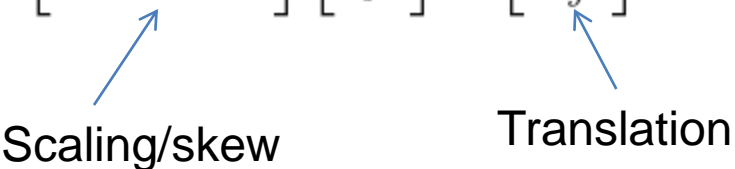
- Accounts for 3D rotation of a surface under orthographic projection



# Affine Object Model

- Accounts for 3D rotation of a surface under orthographic projection

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



Scaling/skew                      Translation

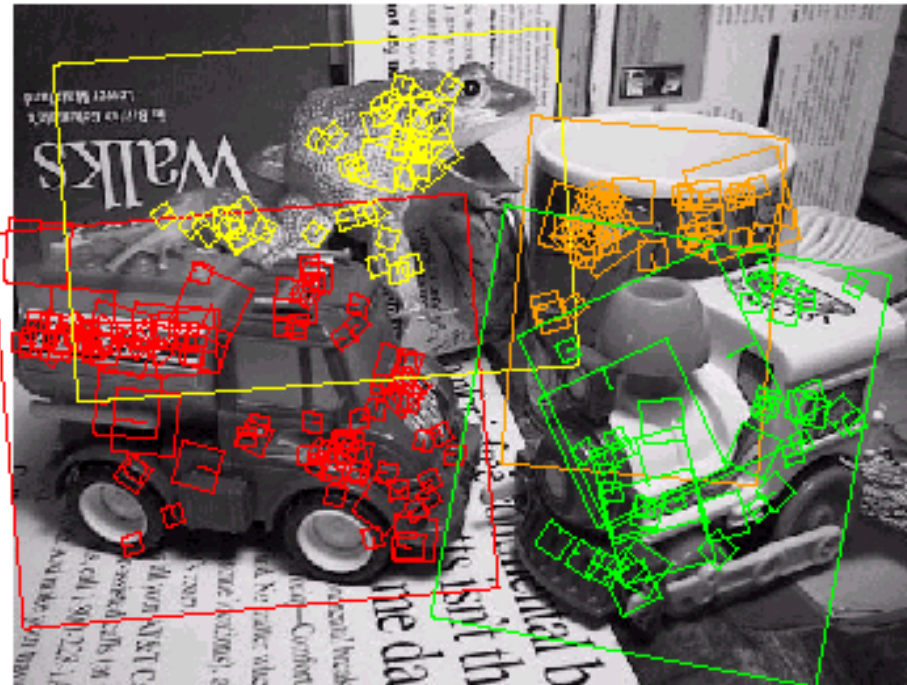
$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \dots & & & \\ & & \dots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix} \quad \mathbf{x} = [\mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{b}$$

What is the minimum number of matched points that we need?

# Finding the objects (in detail)

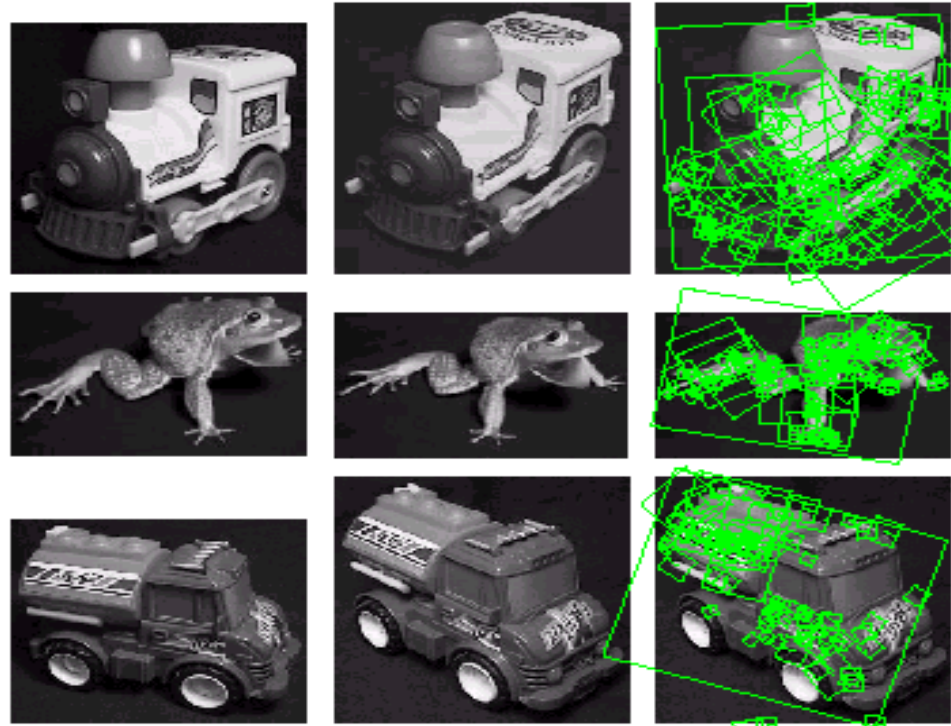
1. Match interest points from input image to database image
2. Get location/scale/orientation using Hough voting
  - In training, each point has known position/scale/orientation wrt whole object
  - Matched points vote for the position, scale, and orientation of the entire object
  - Bins for x, y, scale, orientation
    - Wide bins (0.25 object length in position, 2x scale, 30 degrees orientation)
    - Vote for two closest bin centers in each direction (16 votes total)
3. Geometric verification
  - For each bin with at least 3 keypoints
  - Iterate between least squares fit and checking for inliers and outliers
4. Report object if  $> T$  inliers (T is typically 3, can be computed to match some probabilistic threshold)

# Examples of recognized objects



# View interpolation

- Training
  - Given images of different viewpoints
  - Cluster similar viewpoints using feature matches
  - Link features in adjacent views
- Recognition
  - Feature matches may be spread over several training viewpoints
  - ⇒ Use the known links to “transfer votes” to other viewpoints



[Lowe01]

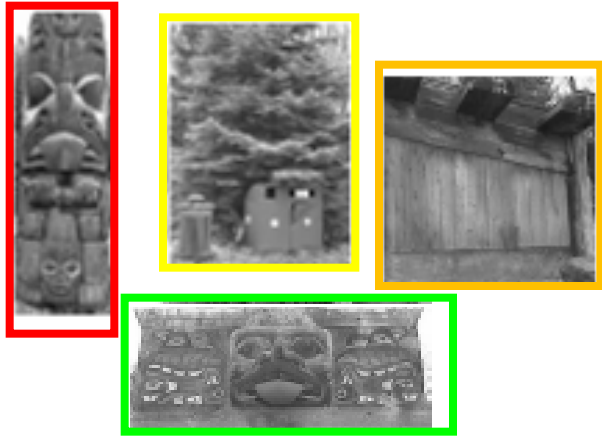
# Applications

- Sony Aibo (Evolution Robotics)
- SIFT usage
  - Recognize docking station
  - Communicate with visual cards
- Other uses
  - Place recognition
  - Loop closure in SLAM

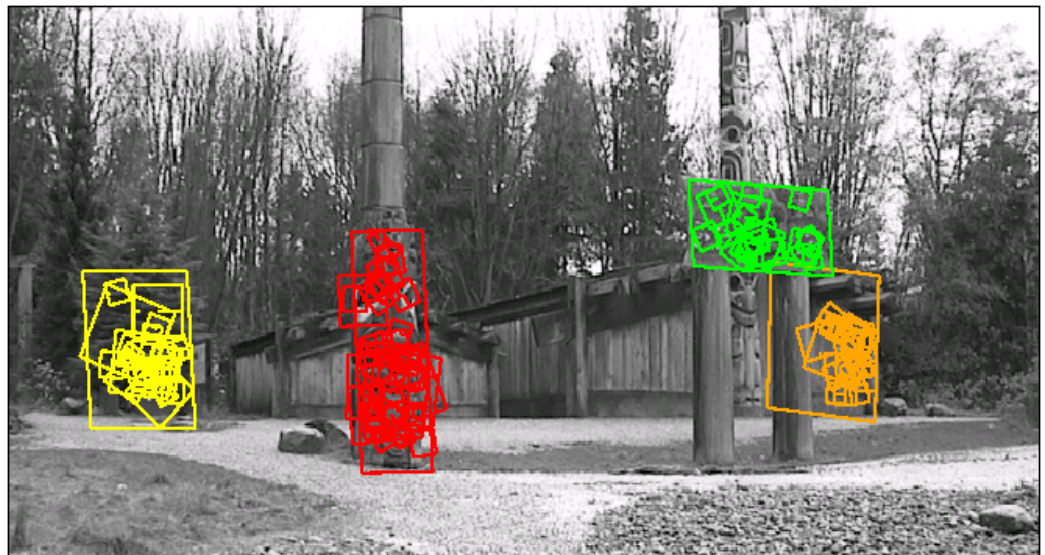




# Location Recognition



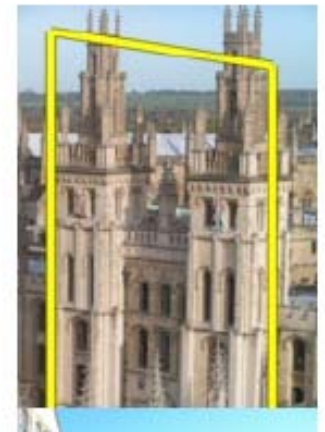
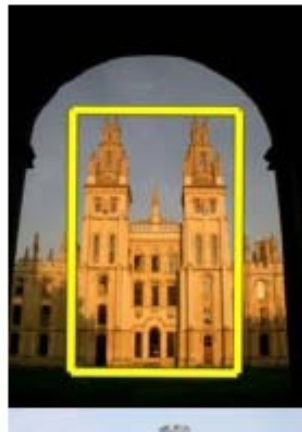
Training



[Lowe04]

Slide credit: David Lowe

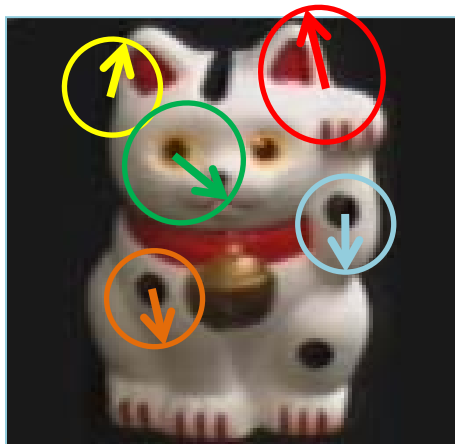
How to quickly find images in a large database that match a given image region?



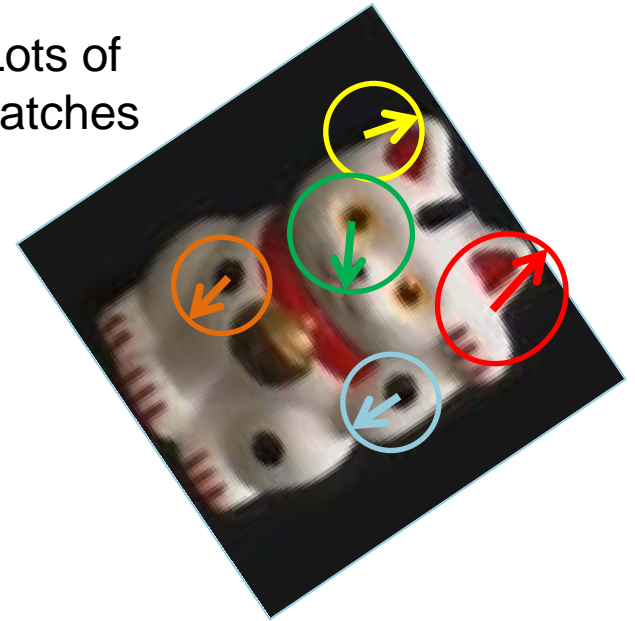


# Simple idea

See how many keypoints are close to keypoints in each other image



Lots of  
Matches



But this will be really, really slow!

Few or No  
Matches



# Fast visual search

“Video Google”, Sivic and Zisserman, ICCV 2003

“Scalable Recognition with a Vocabulary Tree”, Nister and Stewenius, CVPR 2006.

110,000,000  
Images in  
5.8 Seconds



Slide Credit: Nister





Slide Credit: Nister







Slide Credit: Nister

# Key Ideas

- Visual Words
  - Cluster descriptors (e.g., K-means)
- Inverse document file
  - Quick lookup of files given keypoints

tf-idf: Term Frequency – Inverse Document Frequency

The diagram shows the tf-idf formula  $t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$  with four blue arrows pointing to its components from descriptive text labels:

- An arrow from the top-left label "# times word appears in document" points to the  $n_{id}$  numerator.
- An arrow from the bottom-left label "# words in document" points to the  $n_d$  denominator.
- An arrow from the top-right label "# documents" points to the  $N$  numerator of the log term.
- An arrow from the bottom-right label "# documents that contain the word" points to the  $n_i$  denominator of the log term.

# times word appears in document

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

# words in document

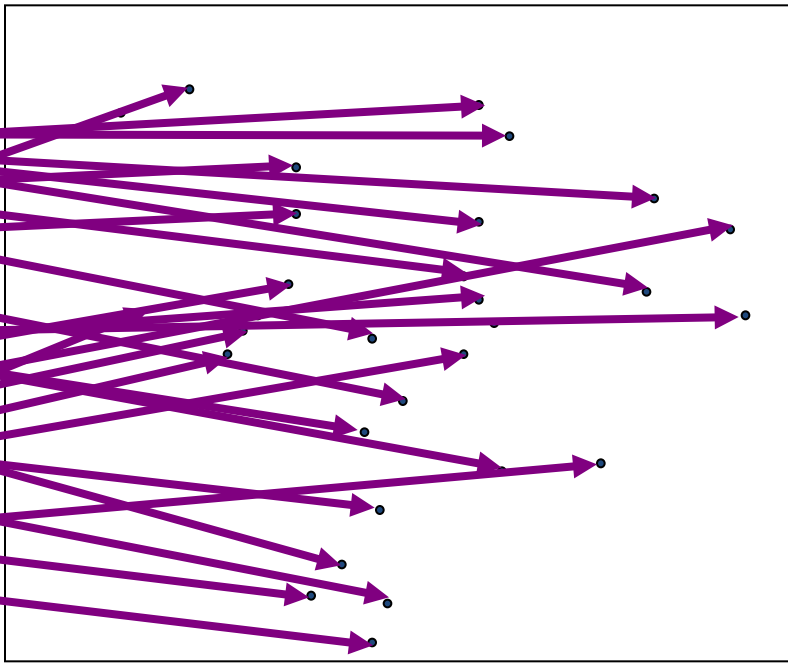
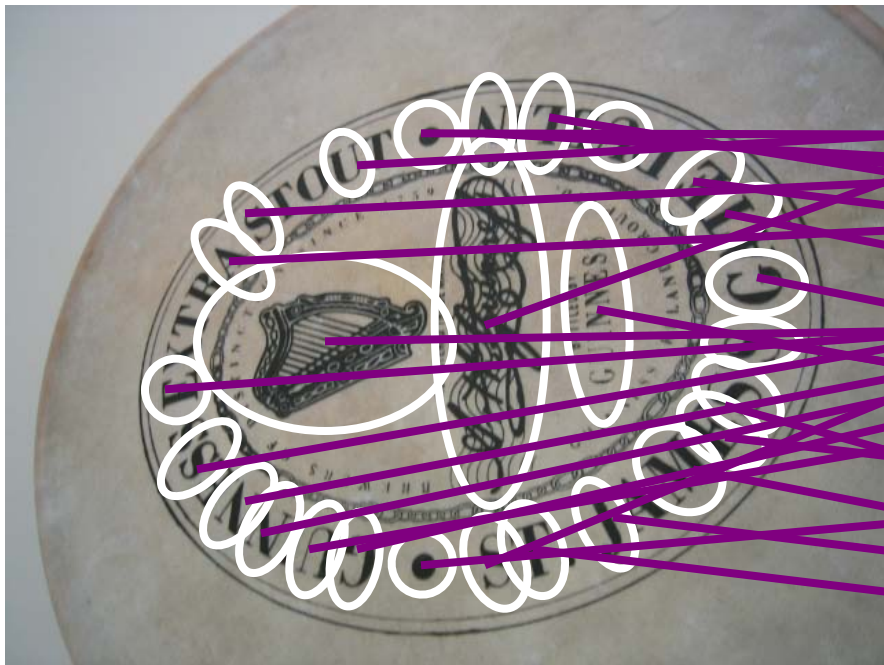
# documents

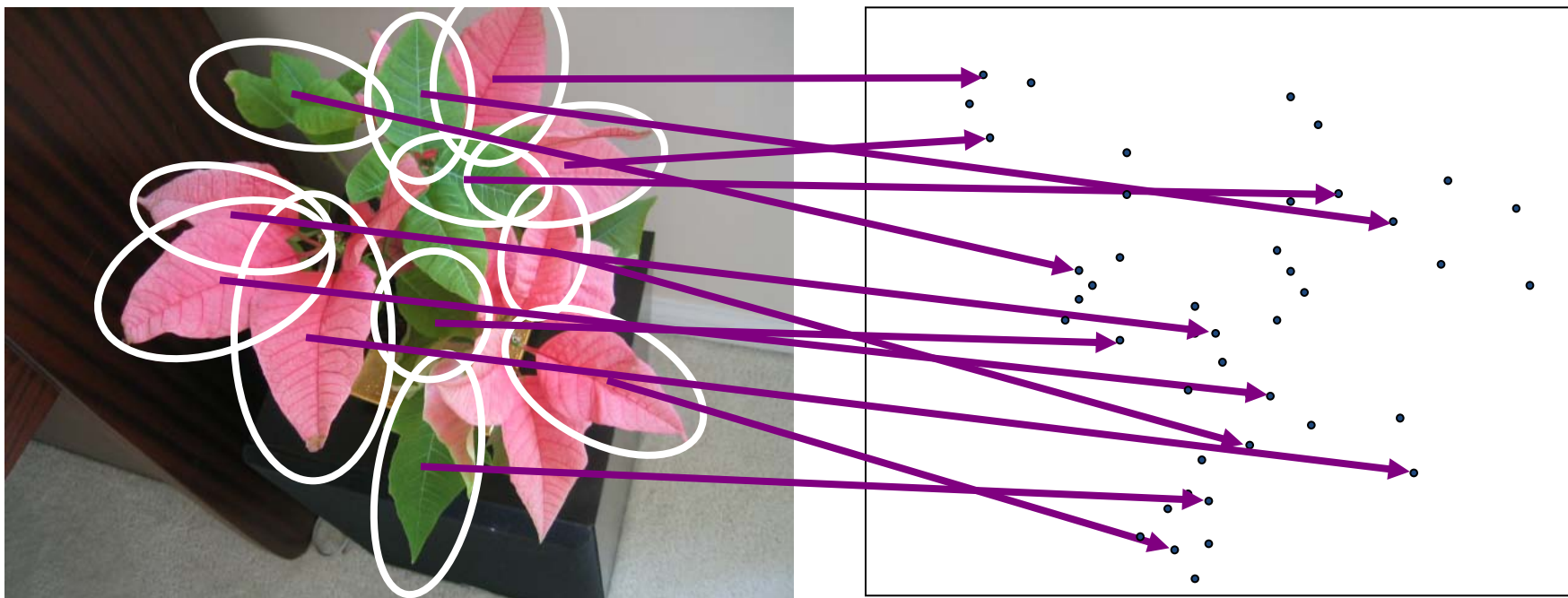
# documents that contain the word

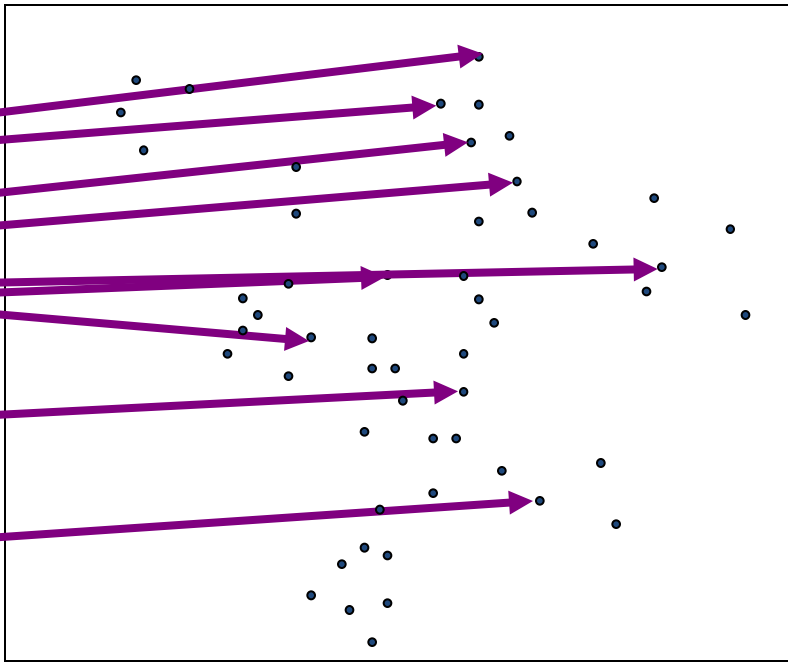
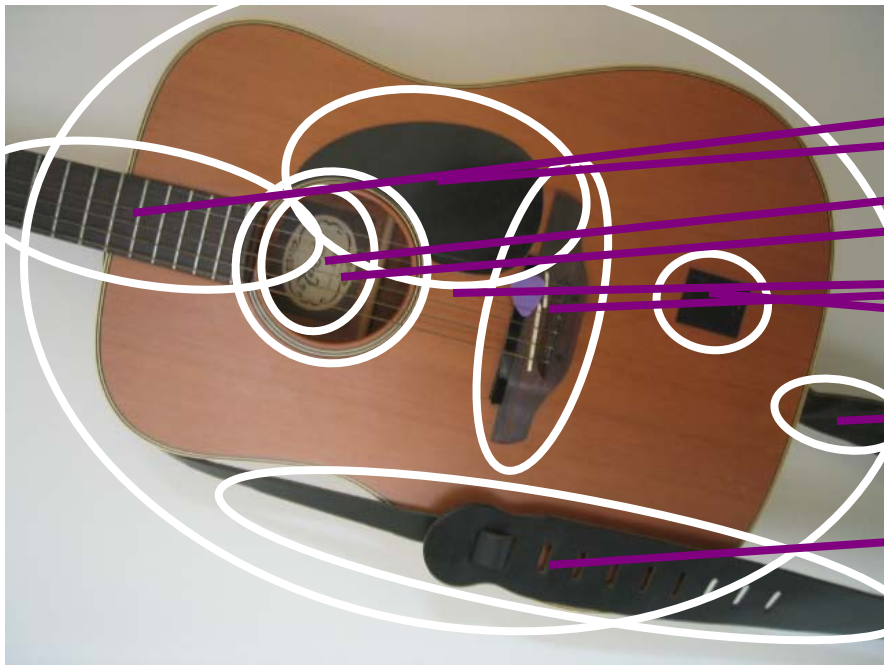
# Recognition with K-tree

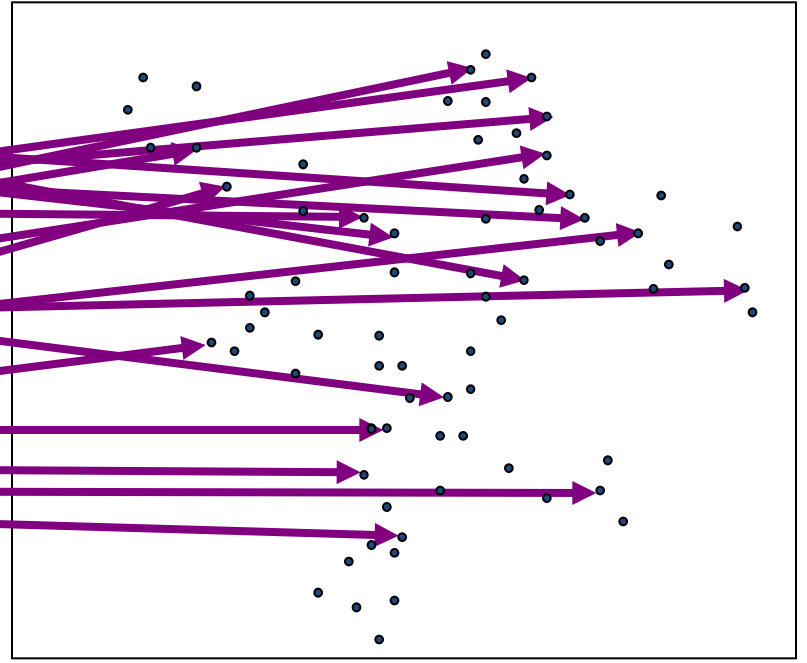
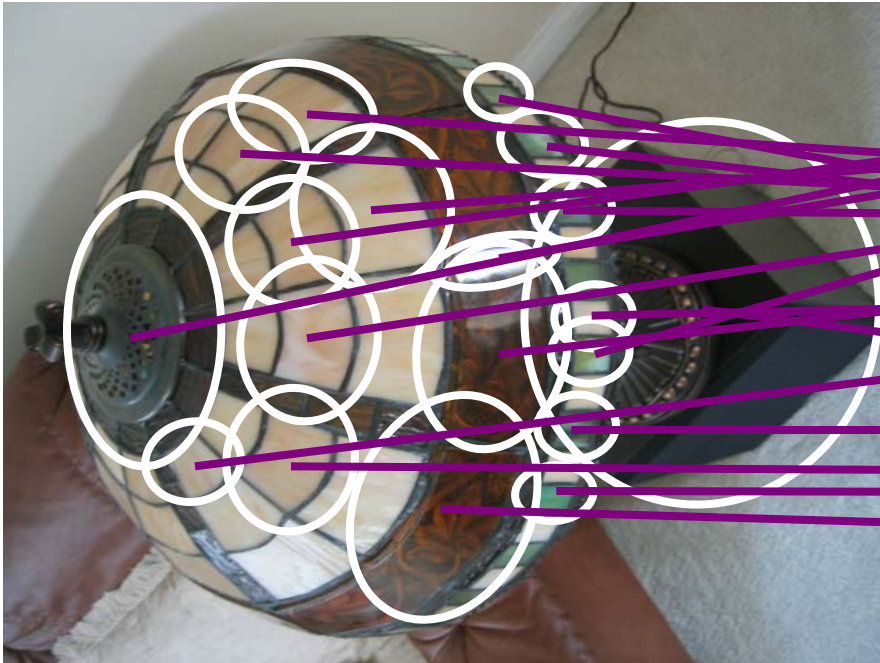
Following slides by David Nister (CVPR 2006)

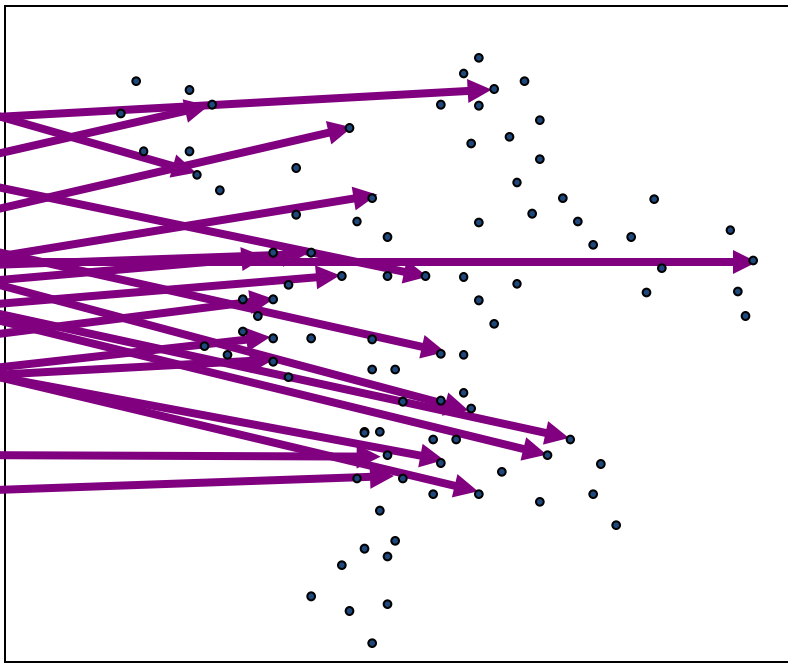


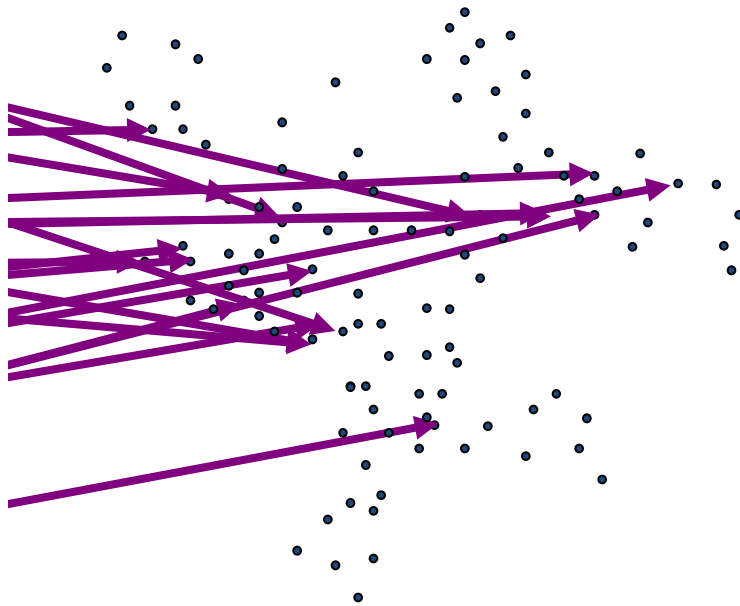


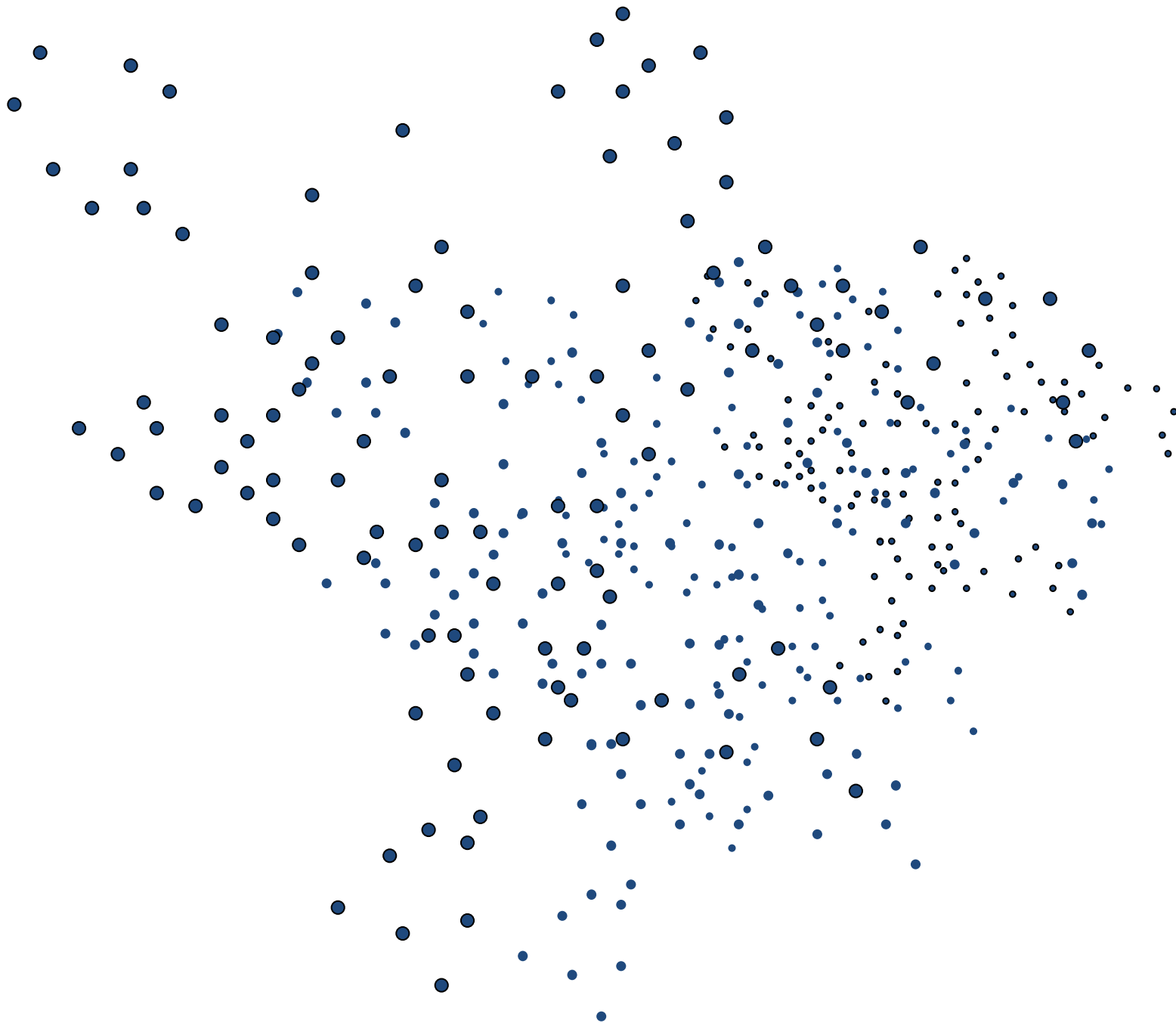


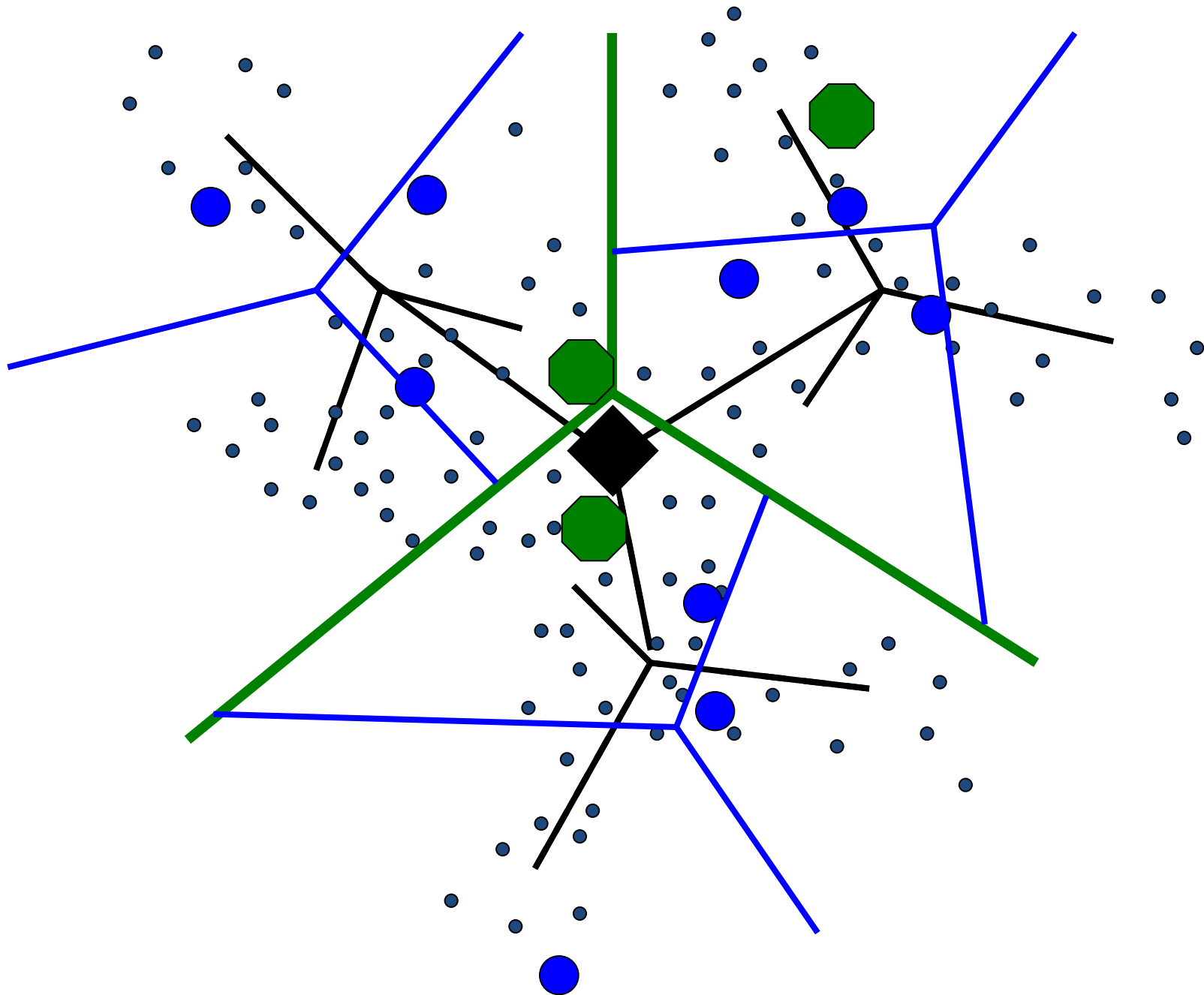




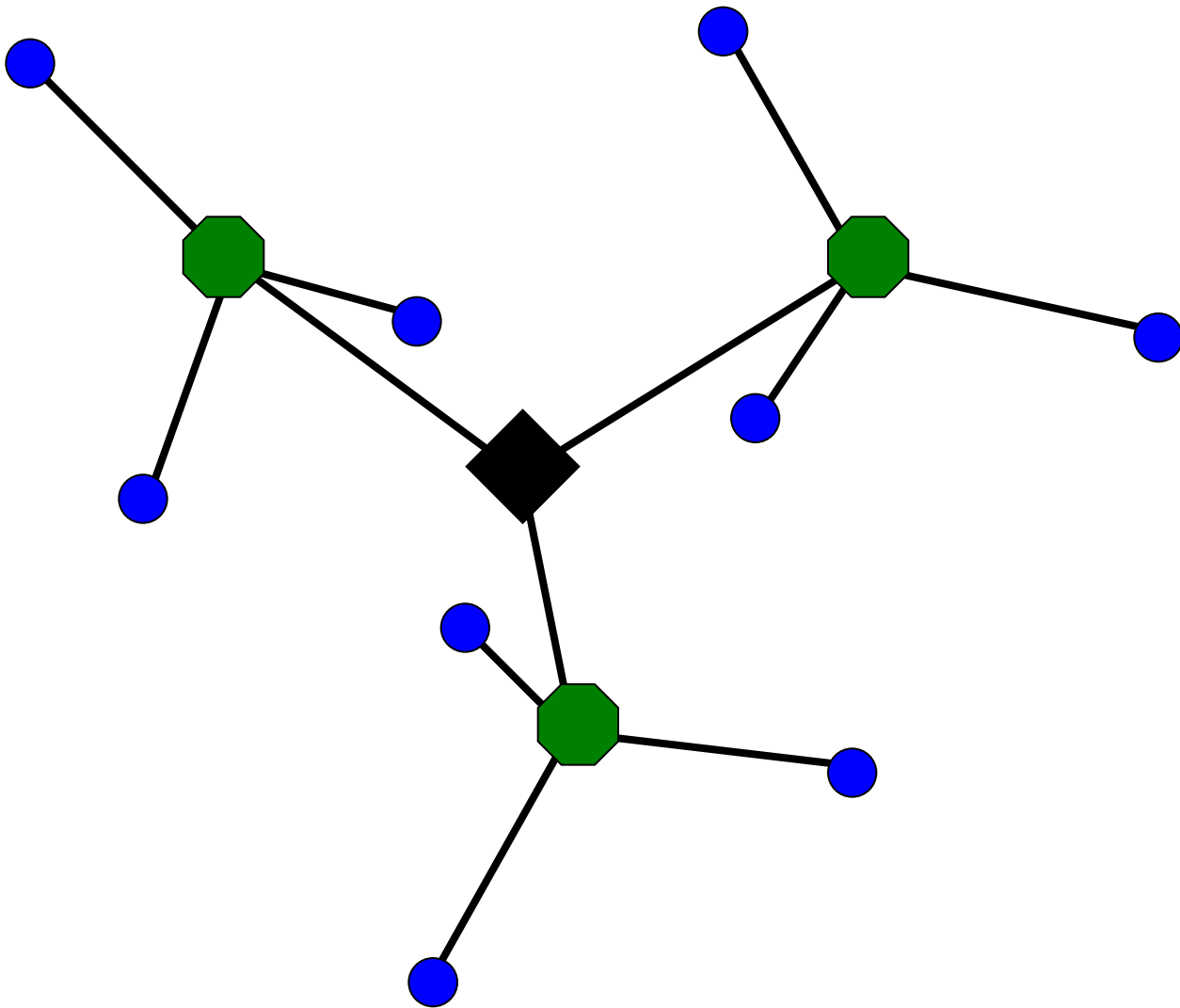


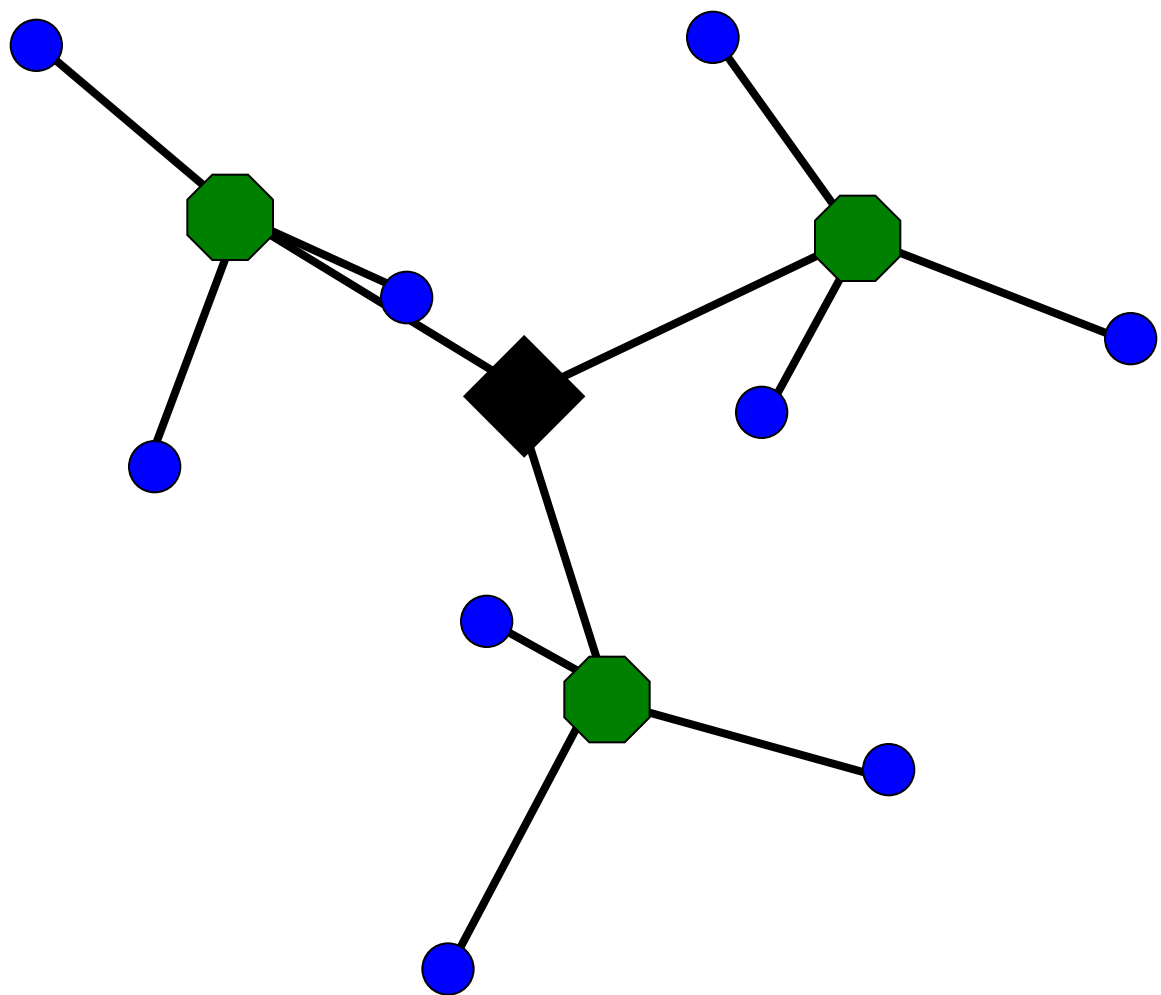


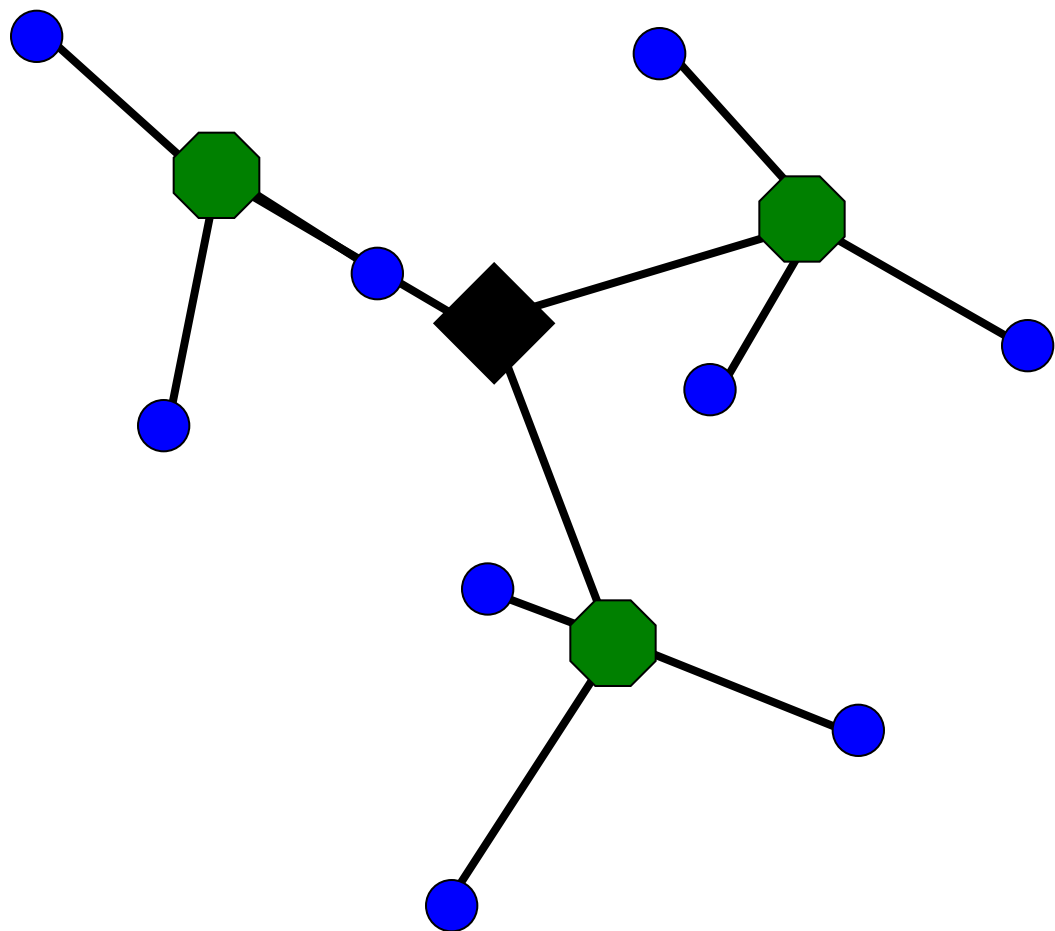


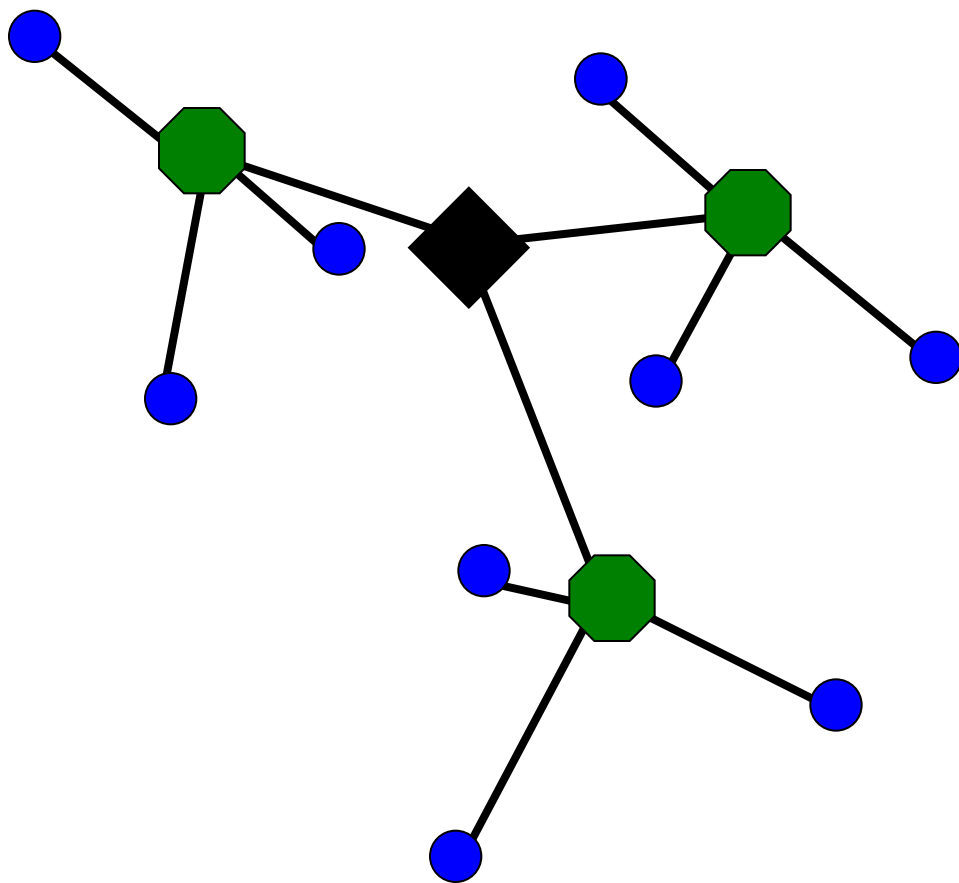


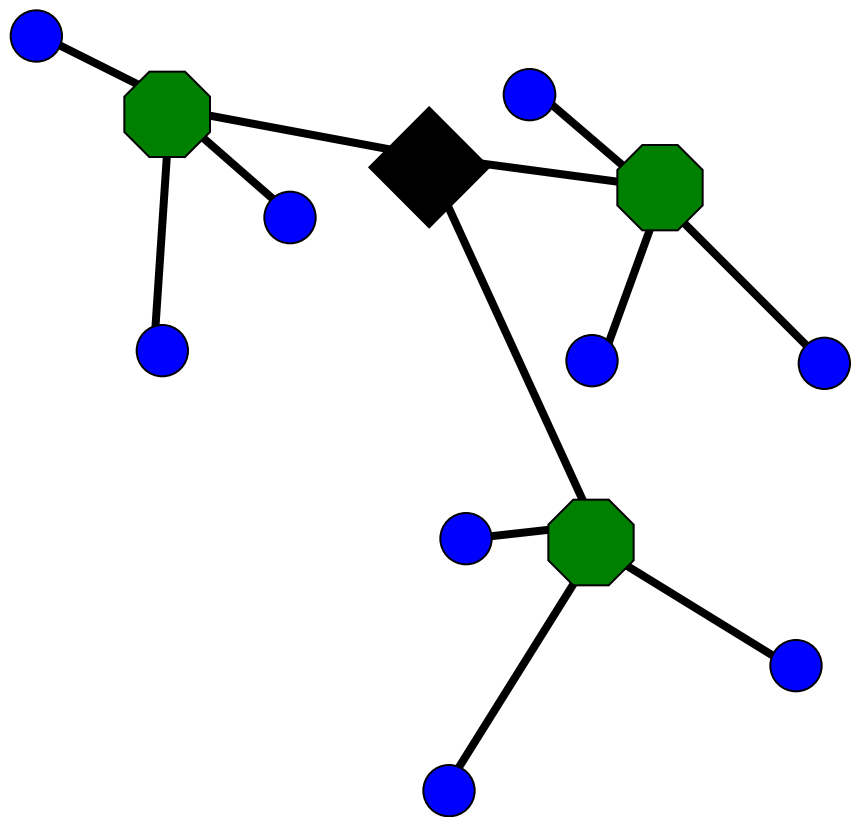


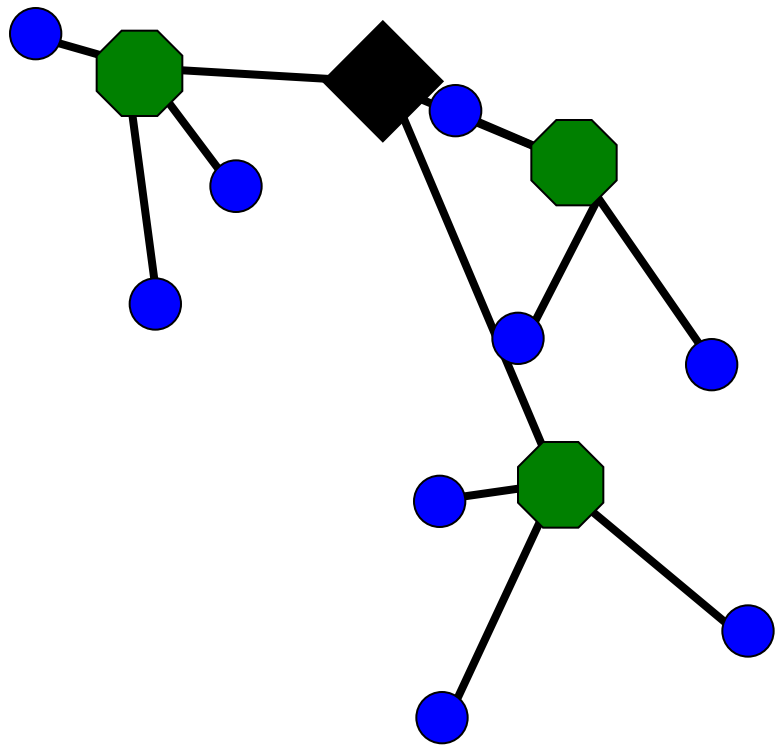


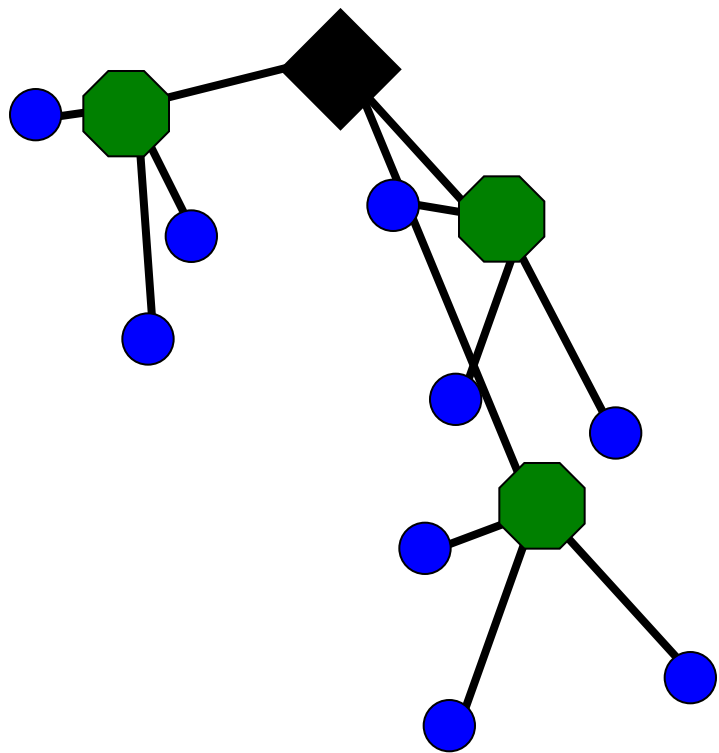


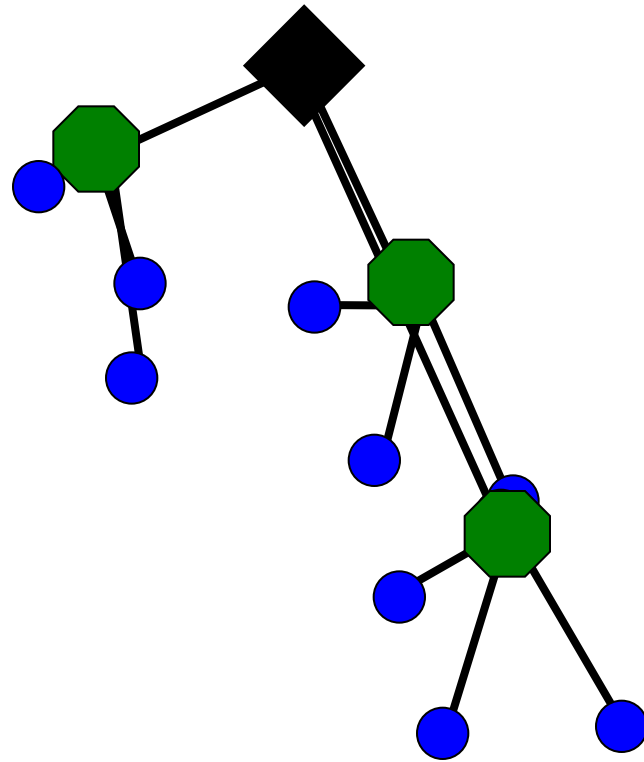




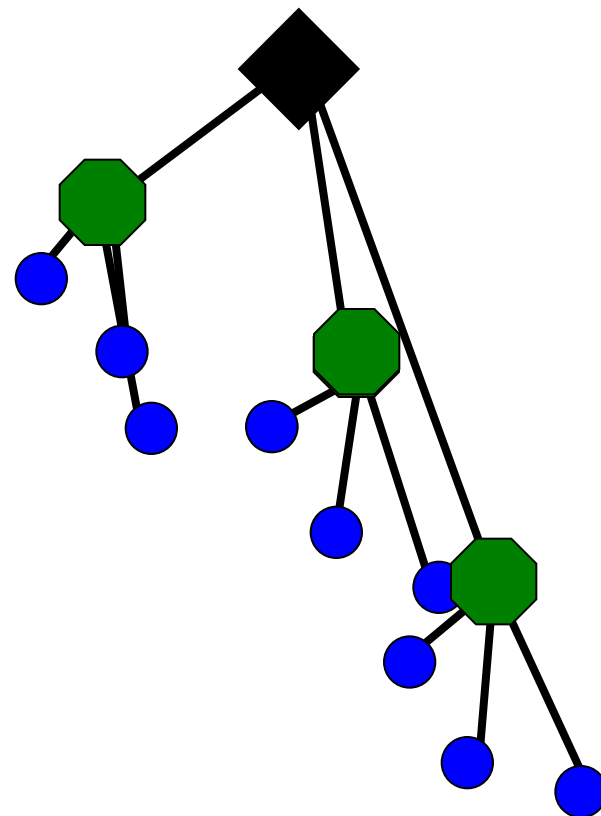


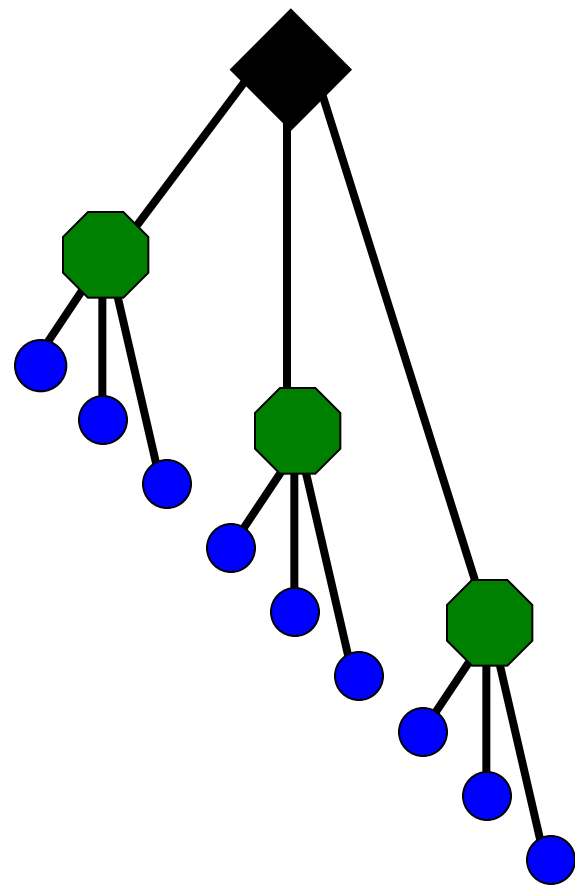


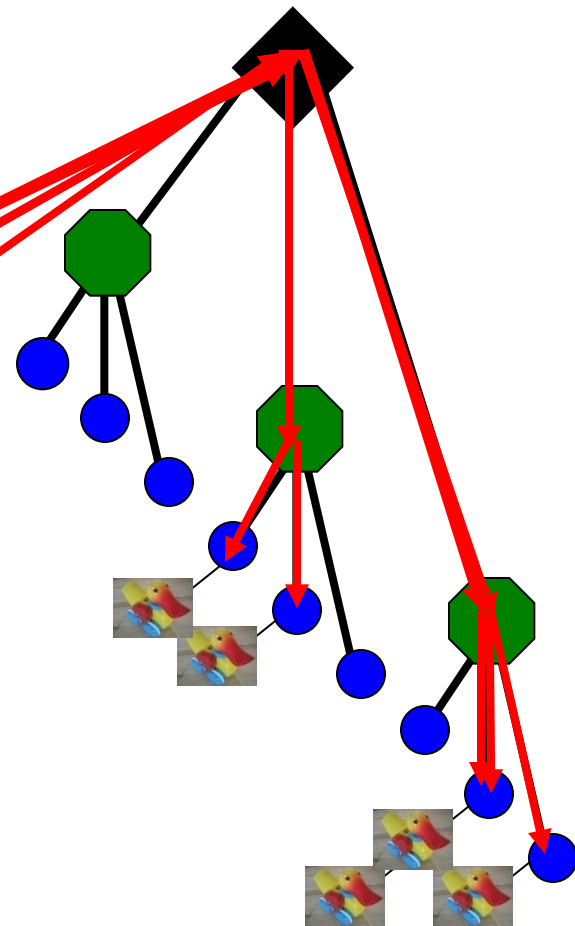
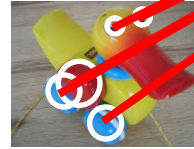


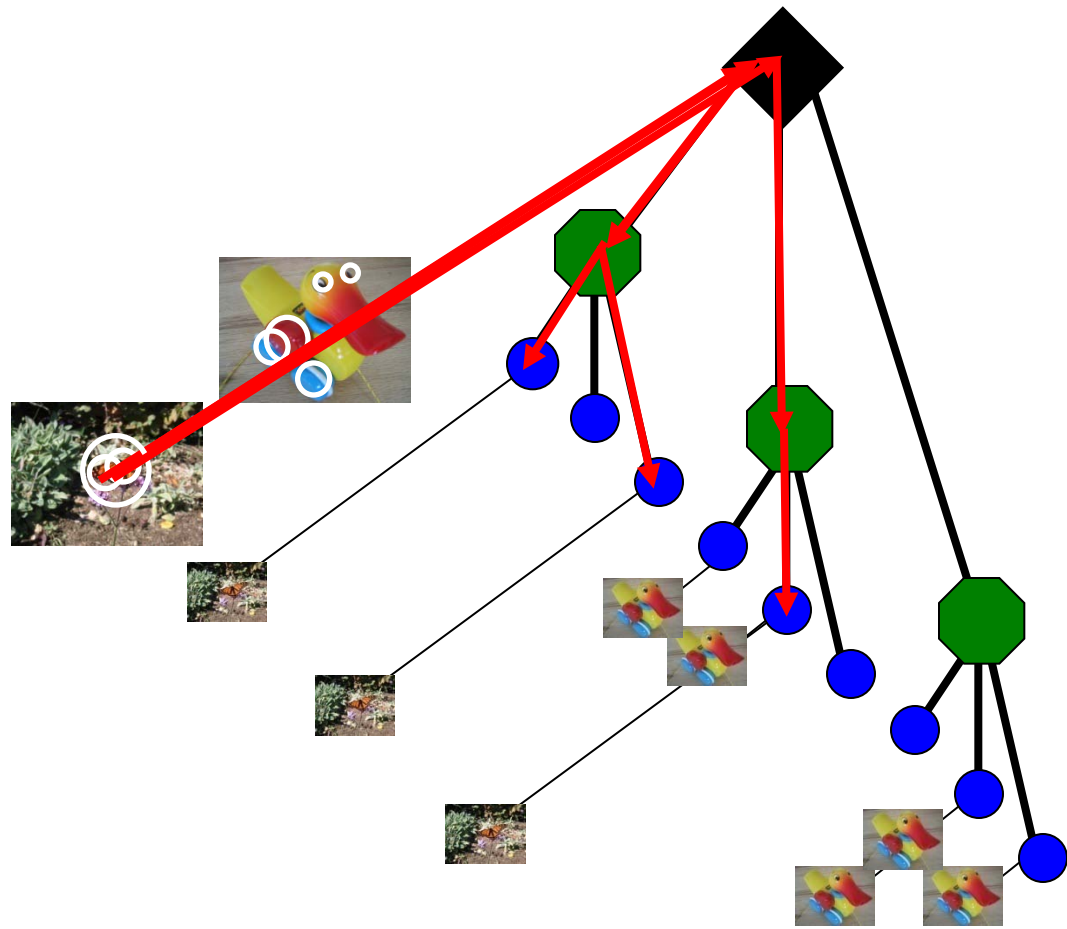


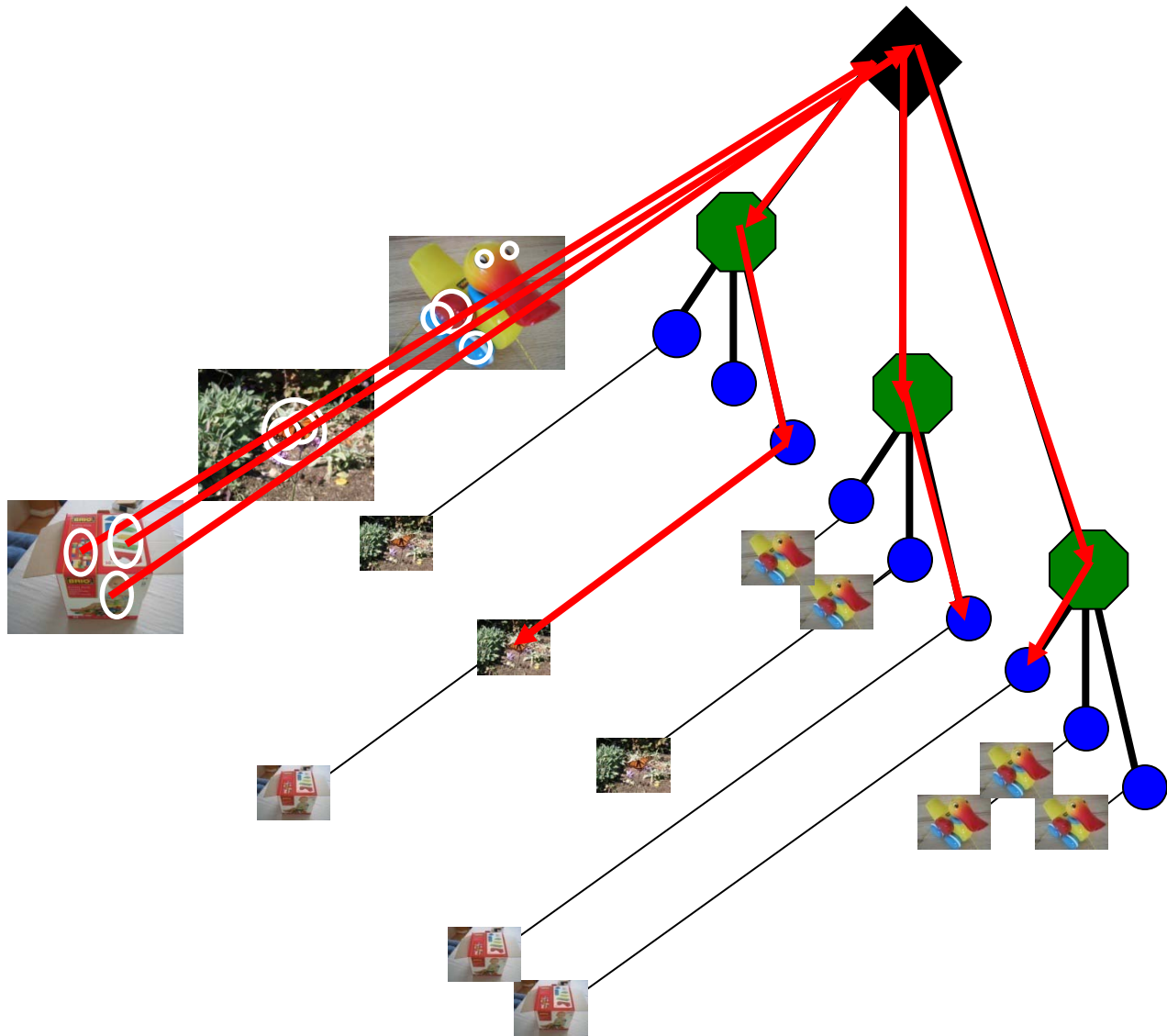


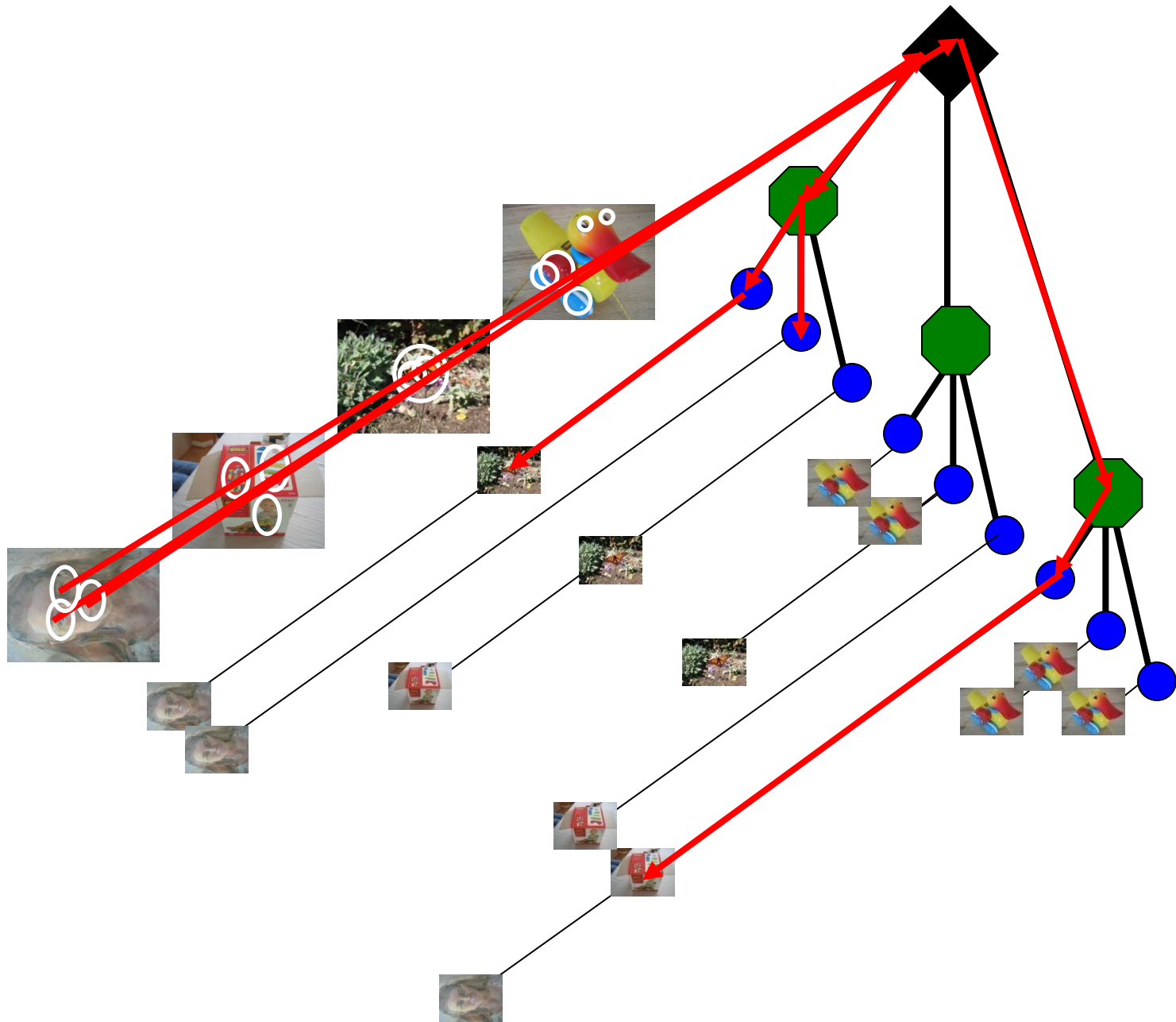


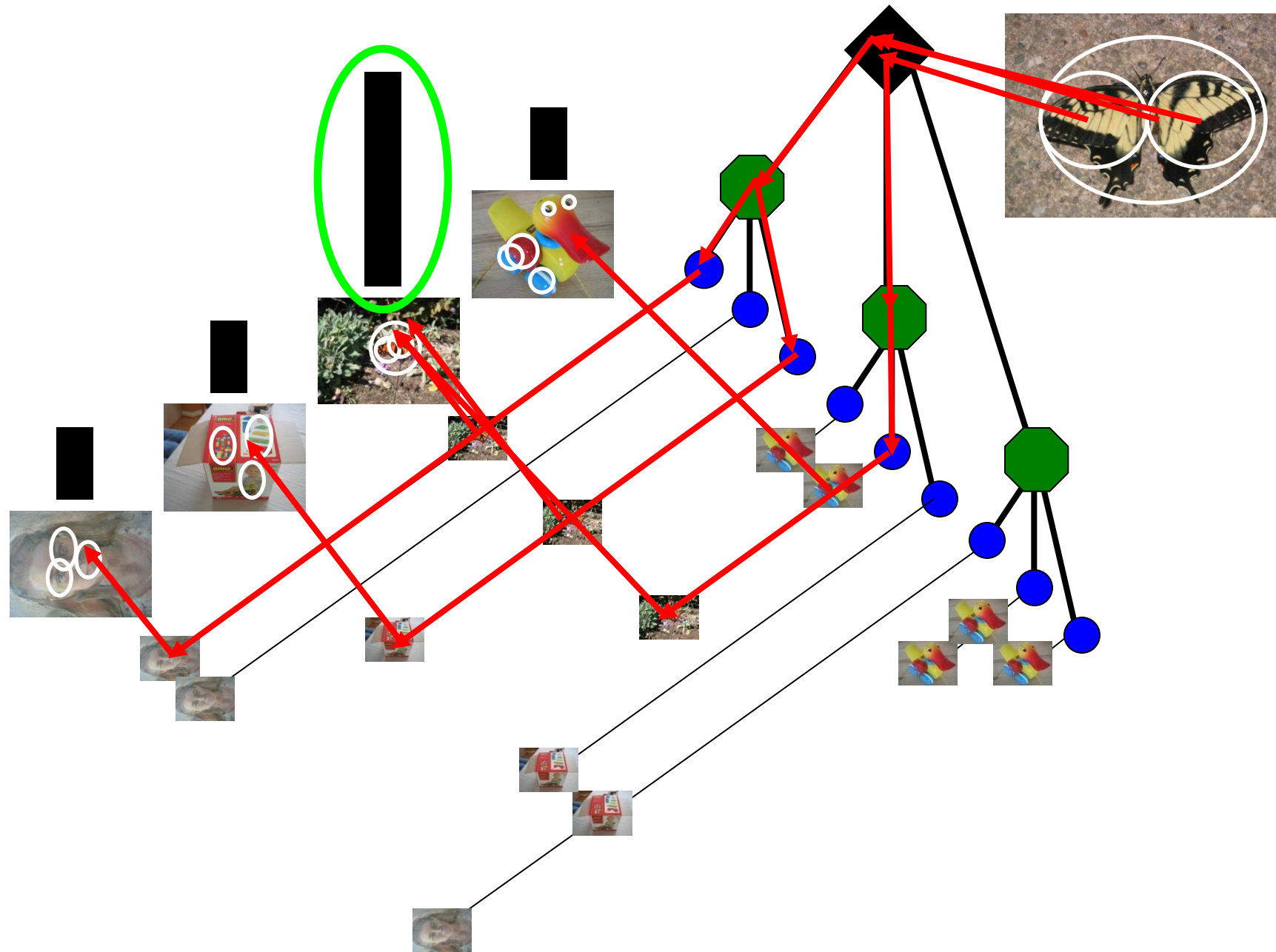




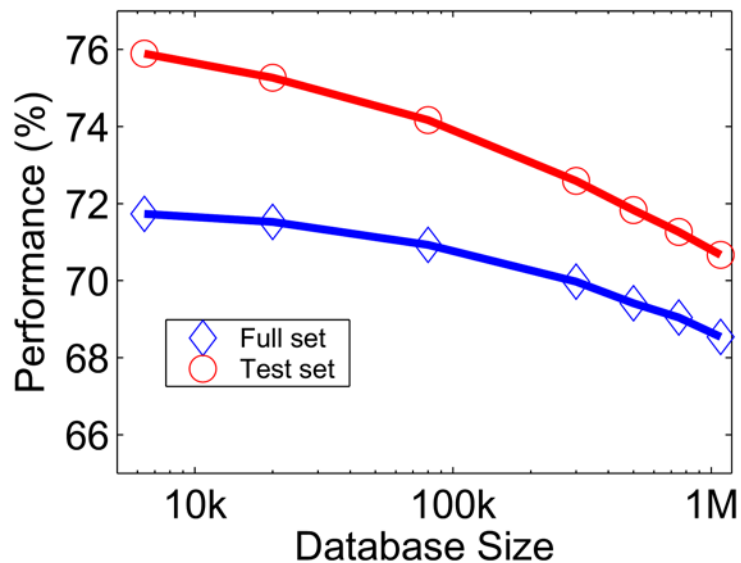








# Performance



## ImageSearch at the VizCentre

New query:

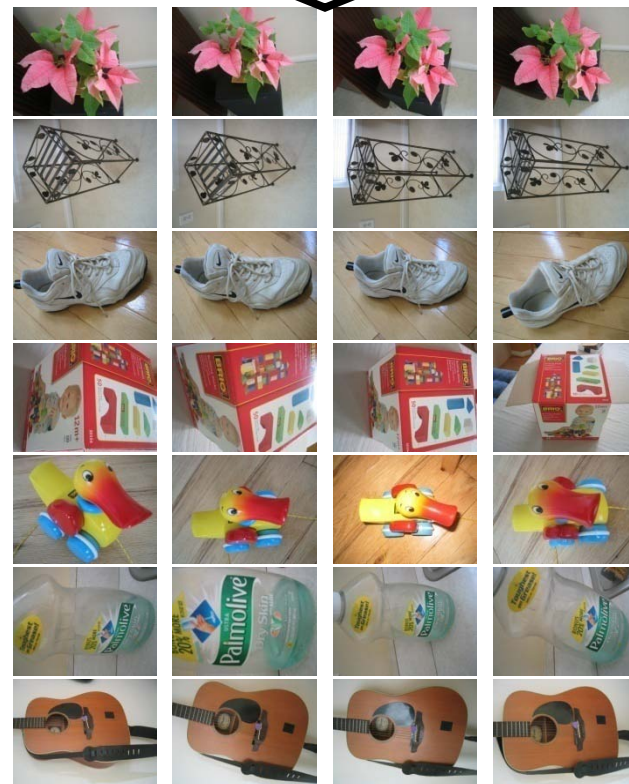
File is 500x320



Top n results of your query.

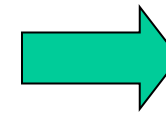


bourne/im1000043322.pgm bourne/im1000043323.pgm bourne/im1000043326.pgm bourne/im1000043327.pgm





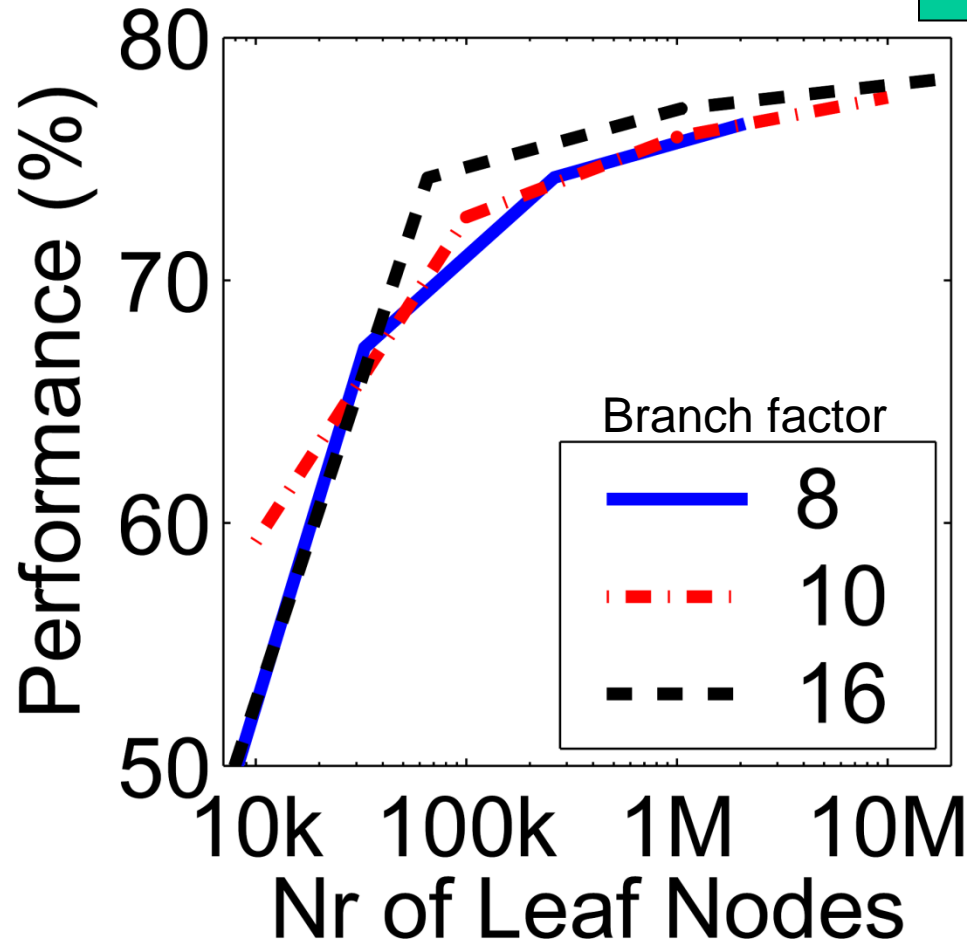
# More words is better



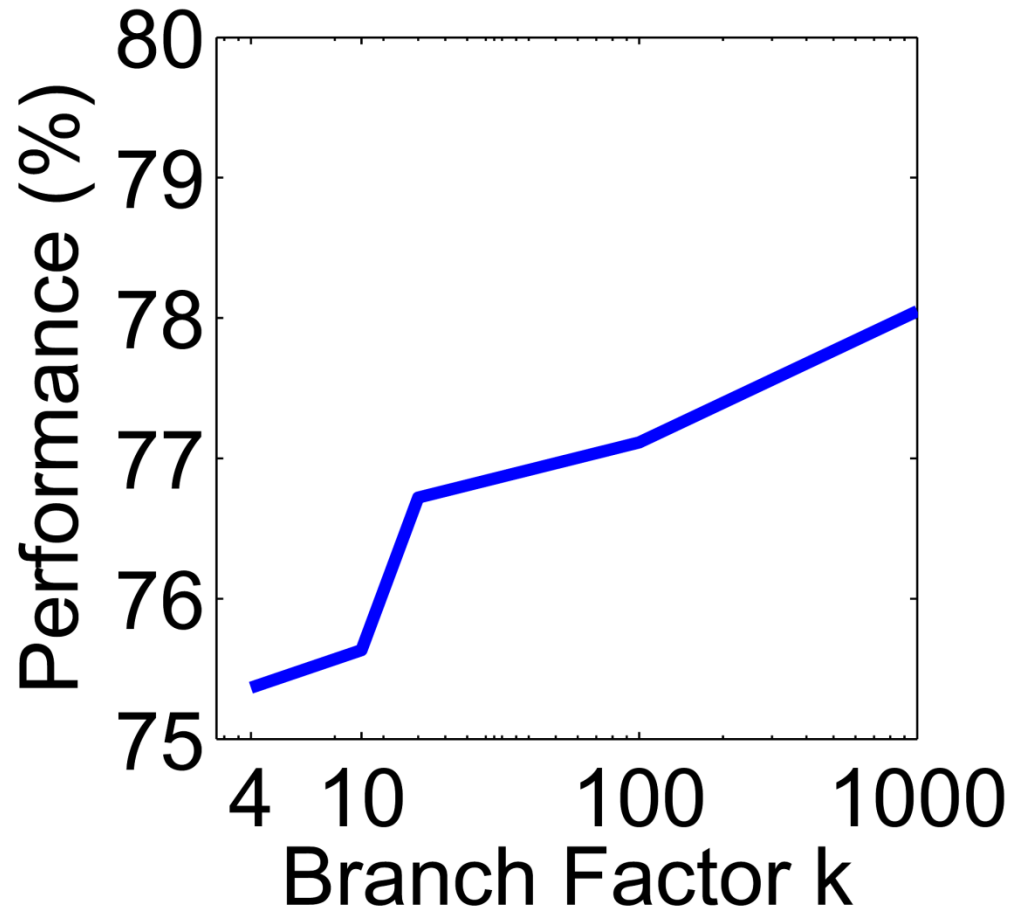
Improves  
Retrieval



Improves  
Speed

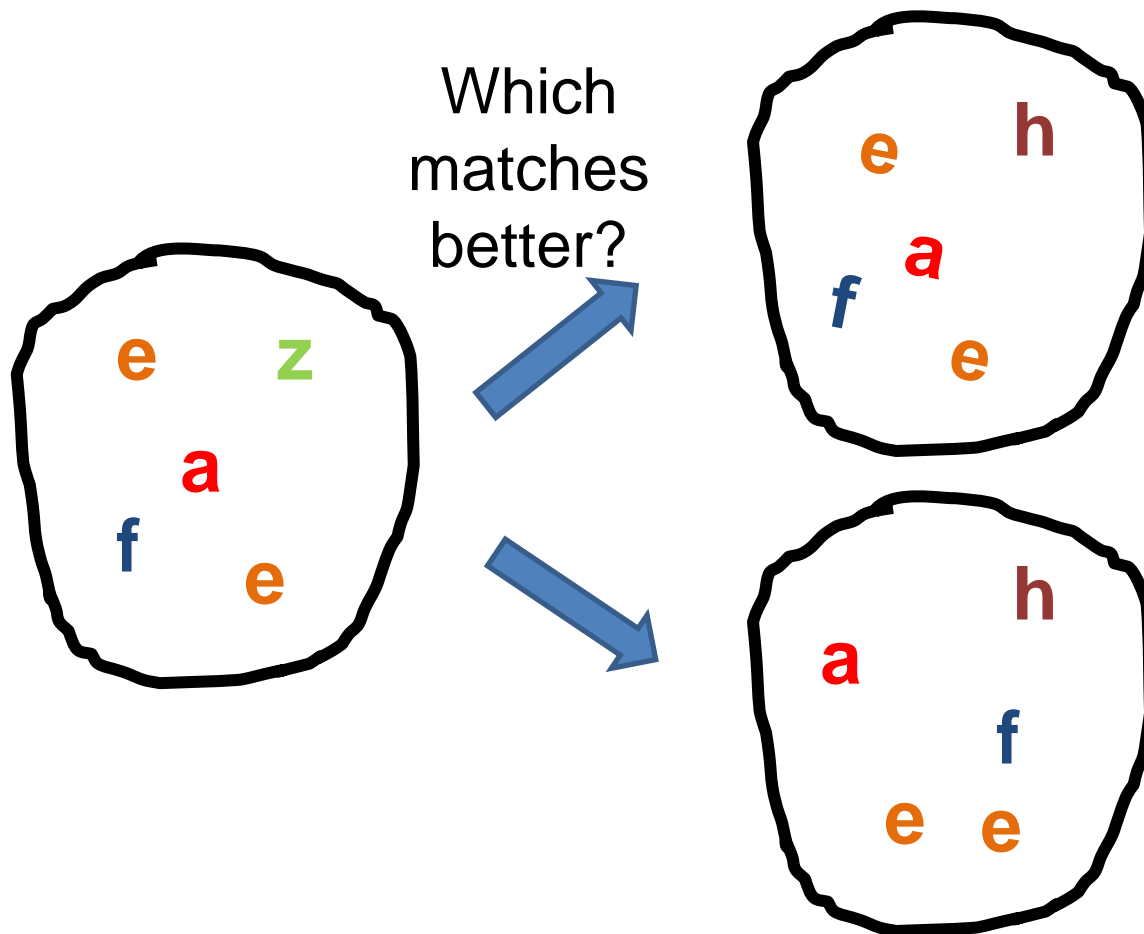


Higher branch factor works better  
(but slower)



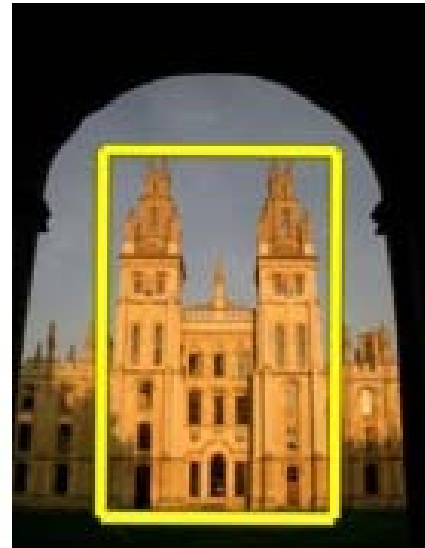
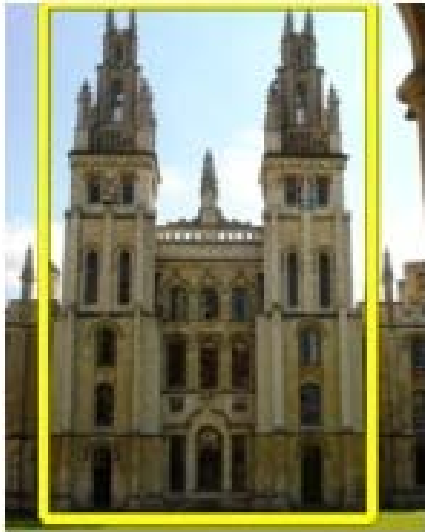
# Can we be more accurate?

So far, we treat each image as containing a “bag of words”, with no spatial information



# Can we be more accurate?

So far, we treat each image as containing a “bag of words”, with no spatial information



Real objects have consistent geometry

# Final key idea: geometric verification

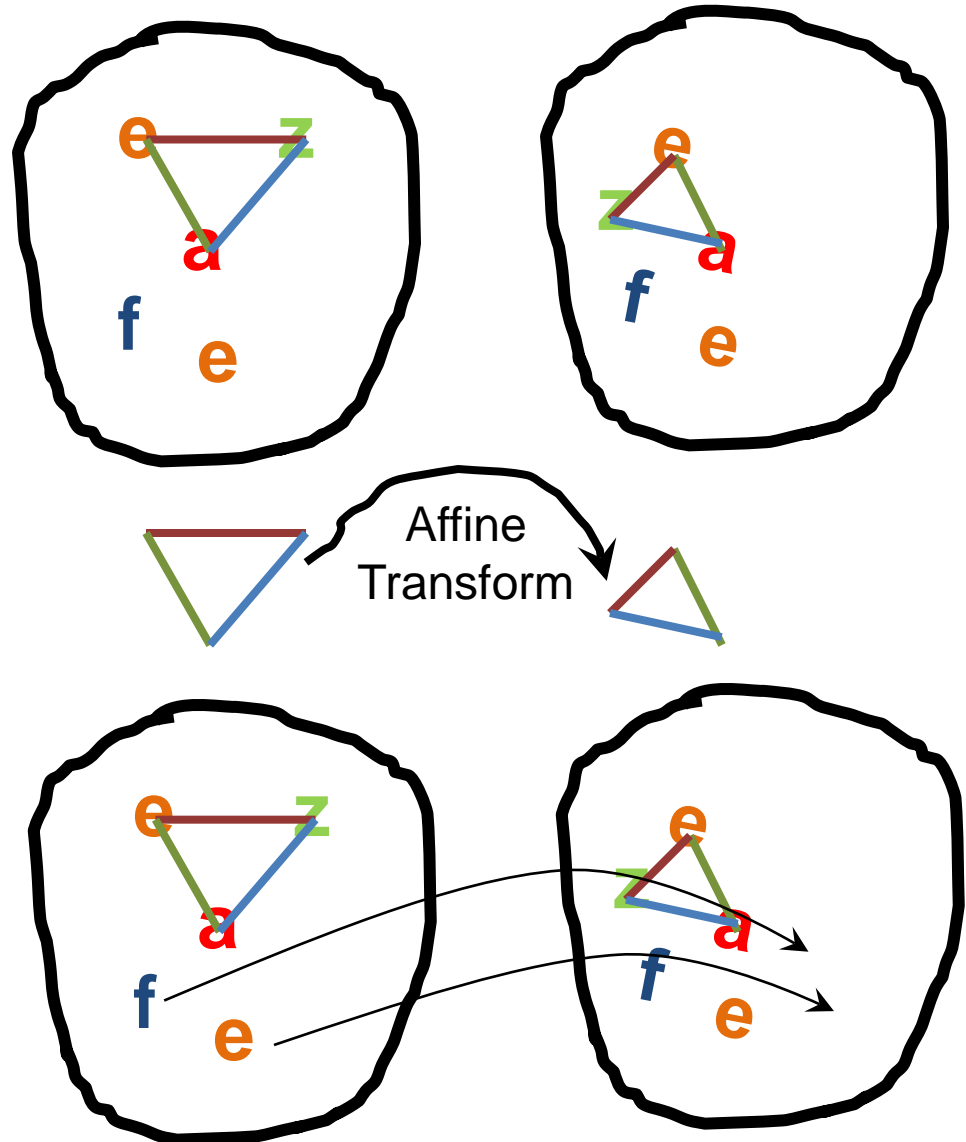
## RANSAC for affine transform

Repeat N times:

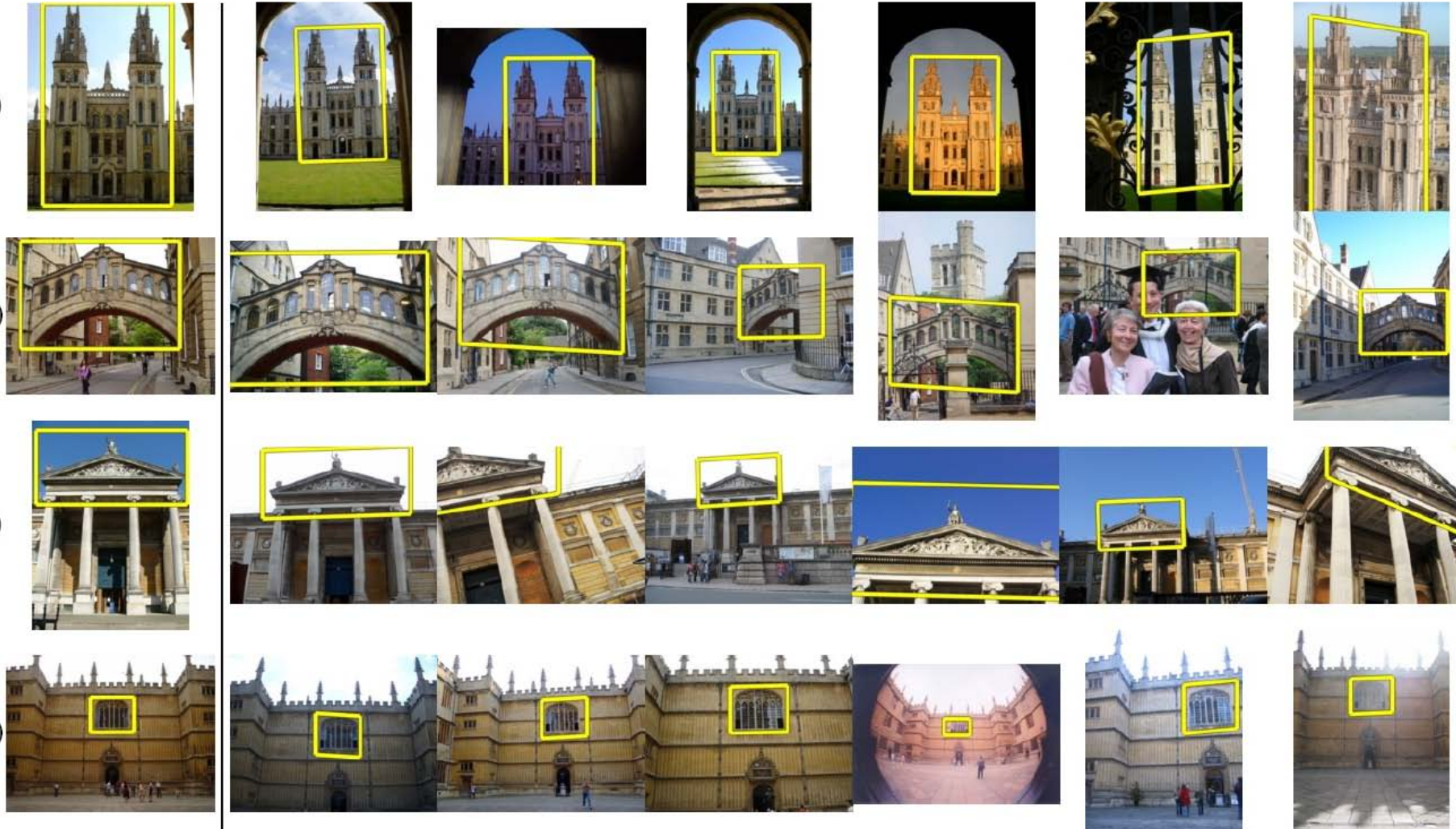
Randomly choose 3  
matching pairs

Estimate  
transformation

Predict remaining  
points and count  
“inliers”



# Application: Large-Scale Retrieval



Query

Results on 5K (demo available for 100K)

K. Grauman, B. Leibe

[Philbin CVPR<sup>07</sup>]

# Example Applications

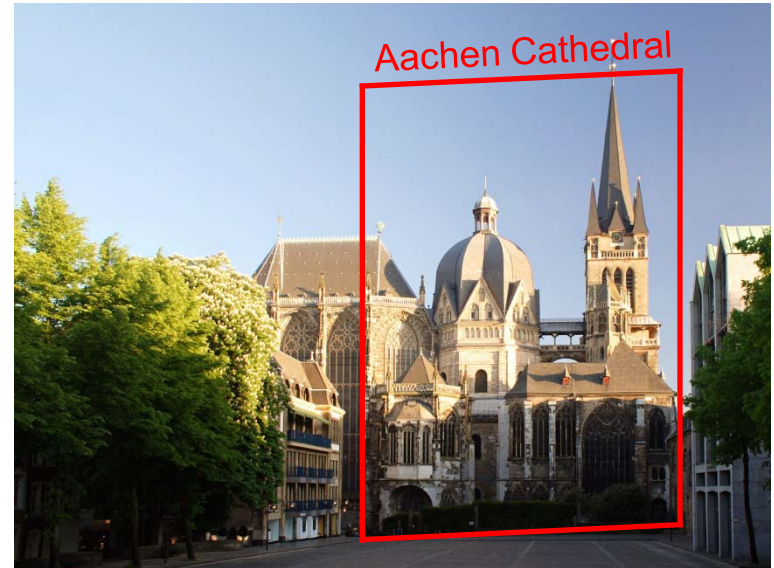


## Mobile tourist guide

Self-localization

Object/building recognition

Photo/video augmentation

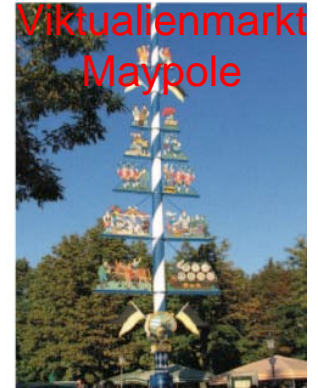




# Application: Image Auto-Annotation



Left: Wikipedia image  
Right: closest match from Flickr





# Video Google System

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

Sivic & Zisserman, ICCV 2003

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



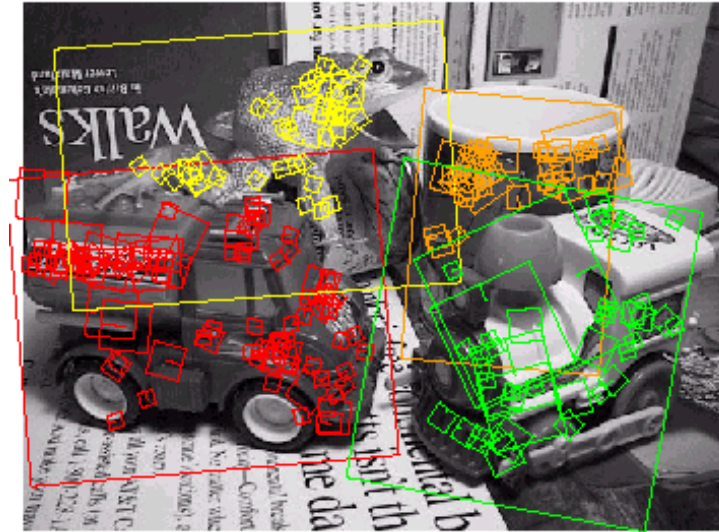
Query region



Retrieved frames

# Things to remember

- Object instance recognition
  - Find keypoints, compute descriptors
  - Match descriptors
  - Vote for / fit affine parameters
  - Return object if  $\# \text{ inliers} > T$



- Keys to efficiency
  - Visual words
    - Used for many applications
  - Inverse document file
    - Used for web-scale search

