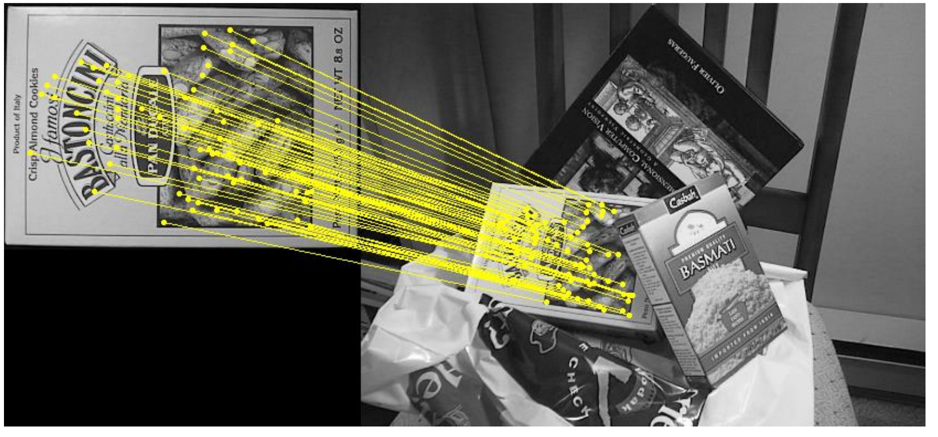
Keypoint Recognition Using Randomized Trees

Lecturer: Sang Hwa Lee

Introduction

- In many vision applications, the runtime performance of the object detection and pose estimation is very important

- We already have several techniques having good object detection performance: SIFT, MSER, Hessian Affine, etc..



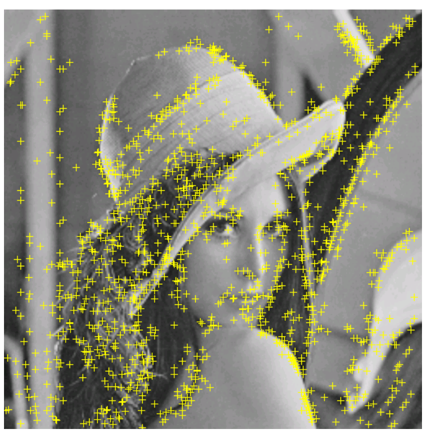
- However, none of them has real-time object detection performance

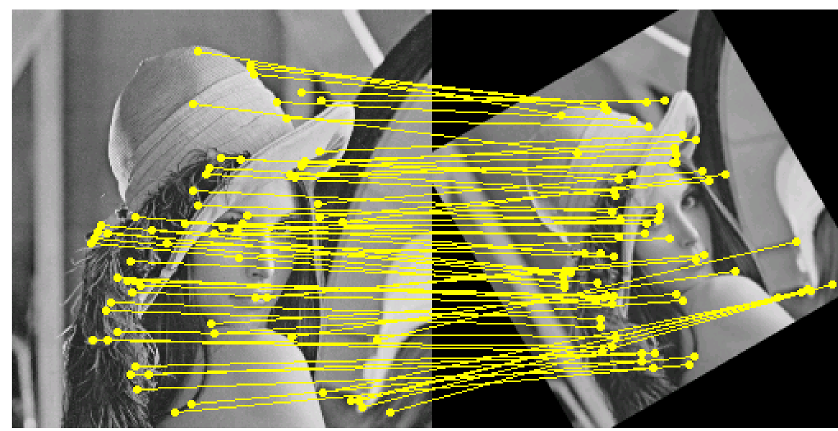
Introduction

- Generally, we have plenty of time for training detector of known object
- Basic idea:
  - Shift much of the computational burden to the training phase
  - Formulate the matching problem as a classification problem using classification trees
  - If we design classification tree well, the detection performance can be increased for the scene having large perspective and scale variations

Introduction

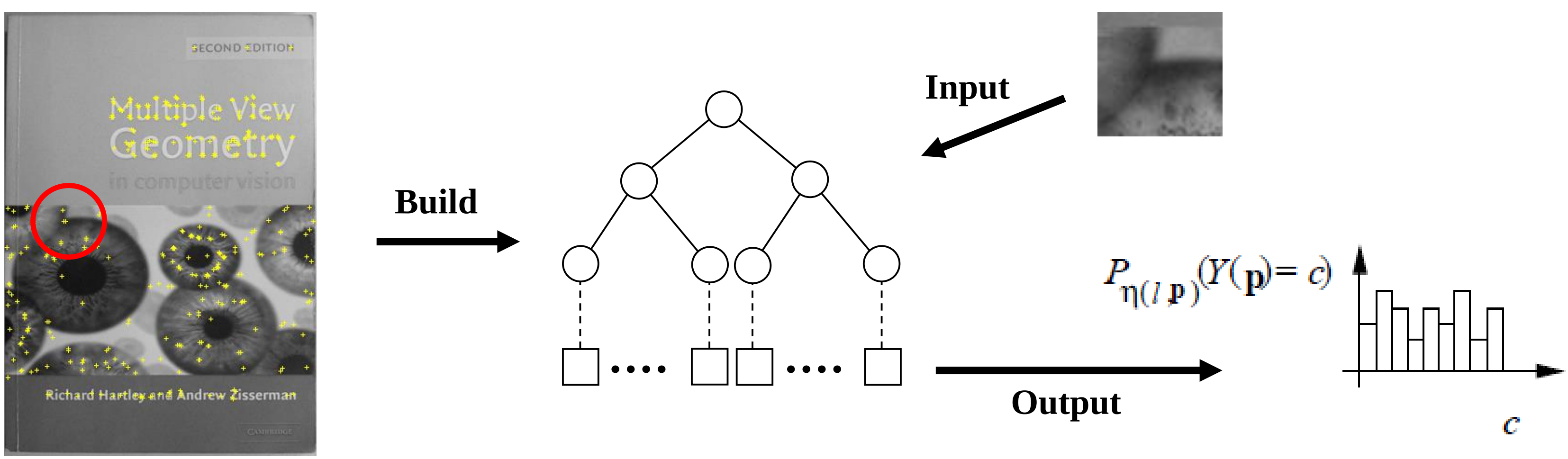
- General approach of the object detection (SIFT):
  - Detect keypoint locations
  - Generate keypoint descriptors
    - 128 dimensional feature vector
  - Match keypoint descriptors of input and model scene
    - Nearest neighbor search
  - Model verification





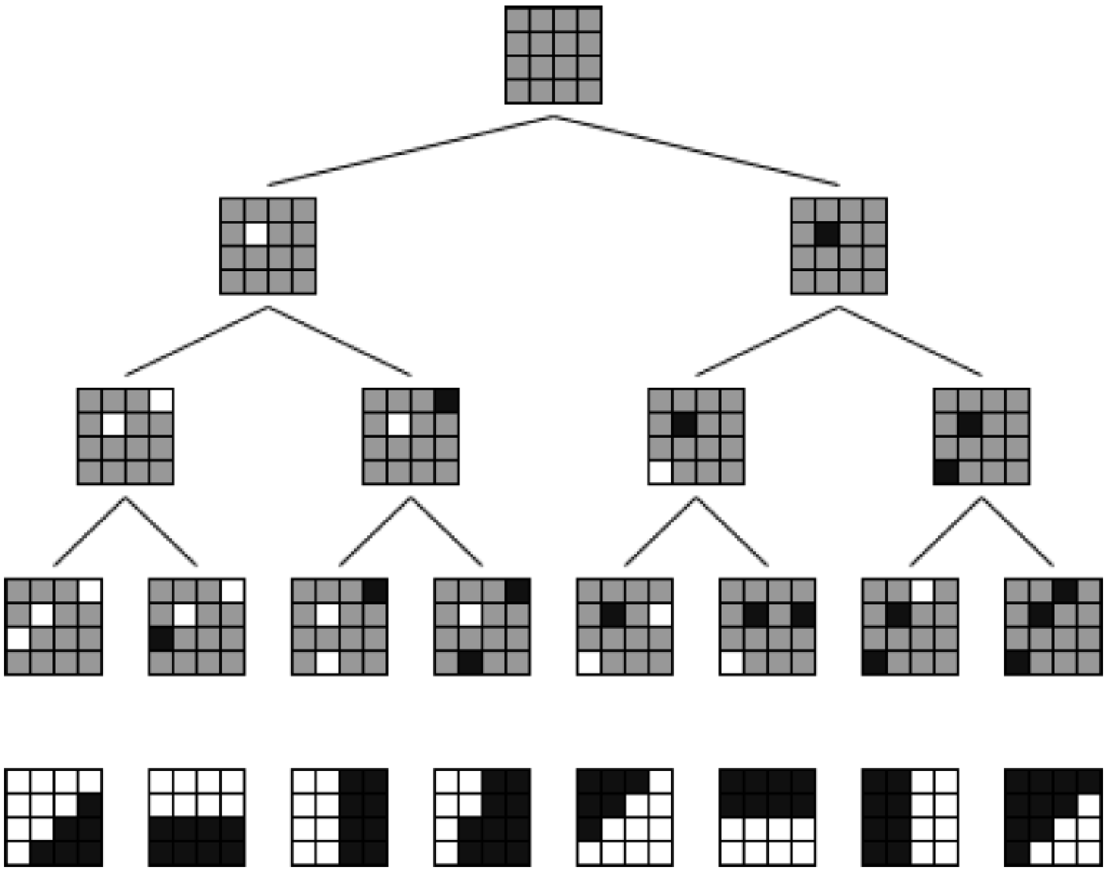
Point Matching as a Classification Problem

- Instead of using the feature vector and NN search, we turn the point matching to the classification problem
- For classification problem, we build the classification tree using a training image
- And for the input image patches from keypoint locations, we get class probabilities using the classification tree
  - Class: the keypoint index of the training image



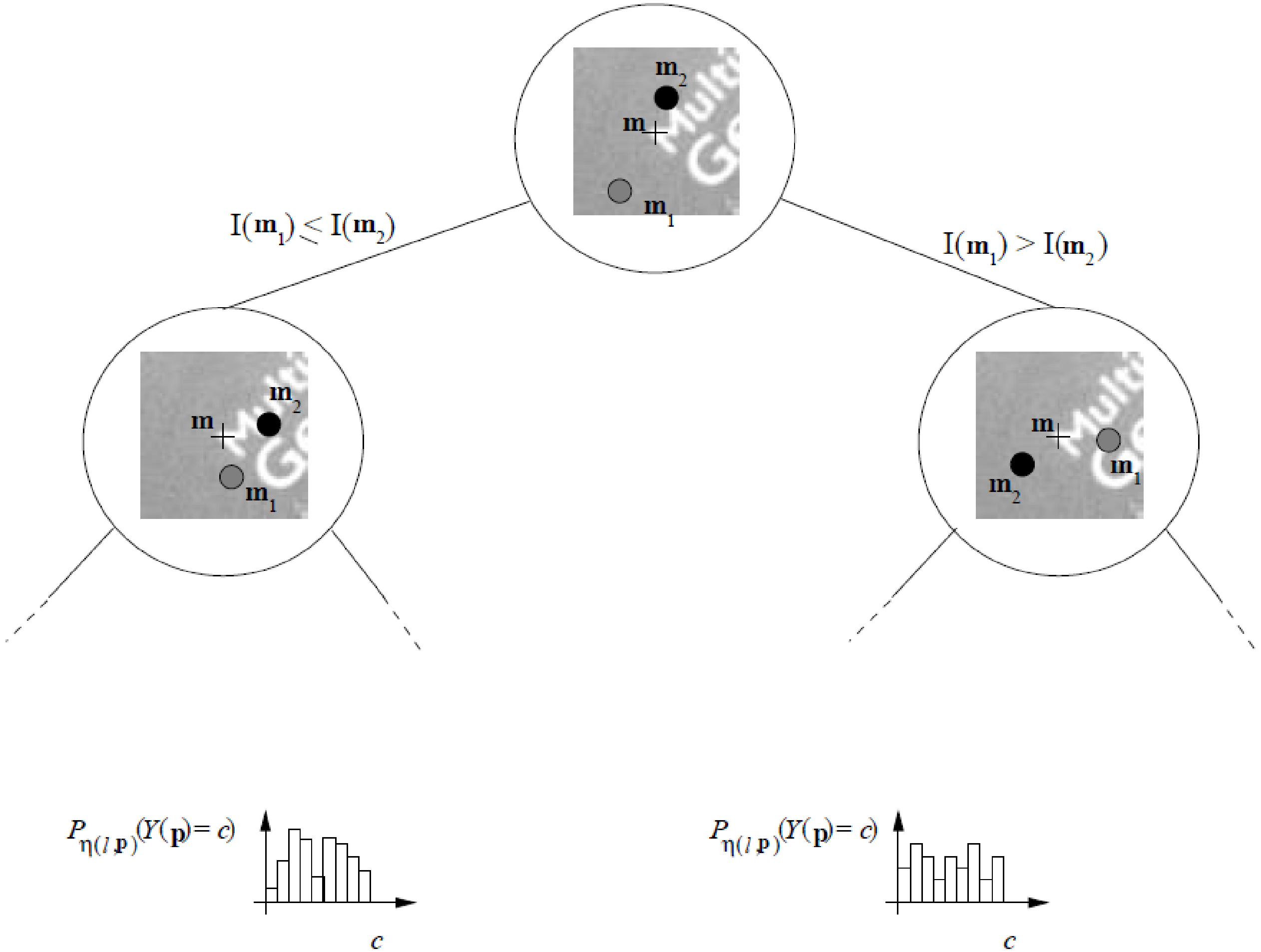
Classification Tree?

- An example (1996, Y. Amit and D. Geman)



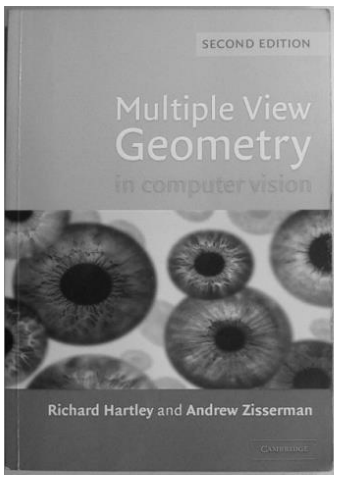
Classification Tree?

- Keypoint Classification Tree



Building the View Sets

- Keypoint selection



Model Image

Random Affine Transformation

• • • • • •

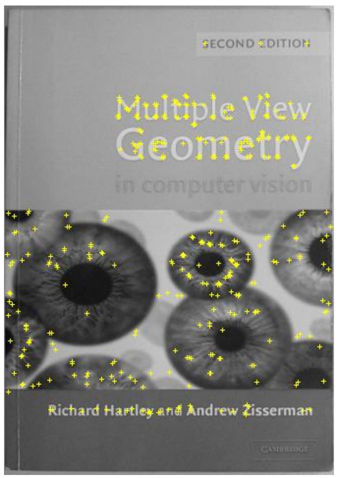
Affine parameters:

Scale = [0.6, 1.5]

Theta, Phi = [-PI, PI]

Building the View Sets

- Keypoint selection



Keypoint Selection

Agglomerate clustered keypoints

• • • • • •

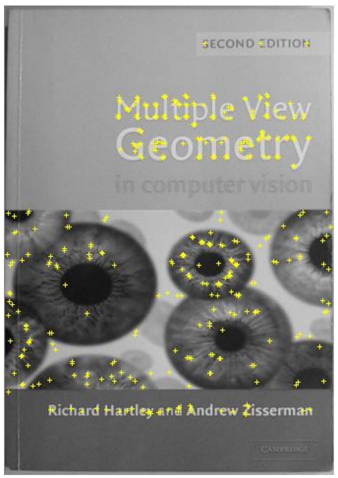
Model Image

Use 3 Octaves:

1, 1/2, 1/4 scale of model image

Building the View Sets

- Building the view sets



Model Image

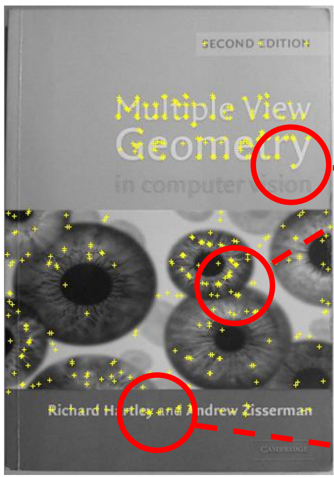
Random Affine Transformation

• • • • • •

Building the View Sets

- Building the view sets

The "View Sets"



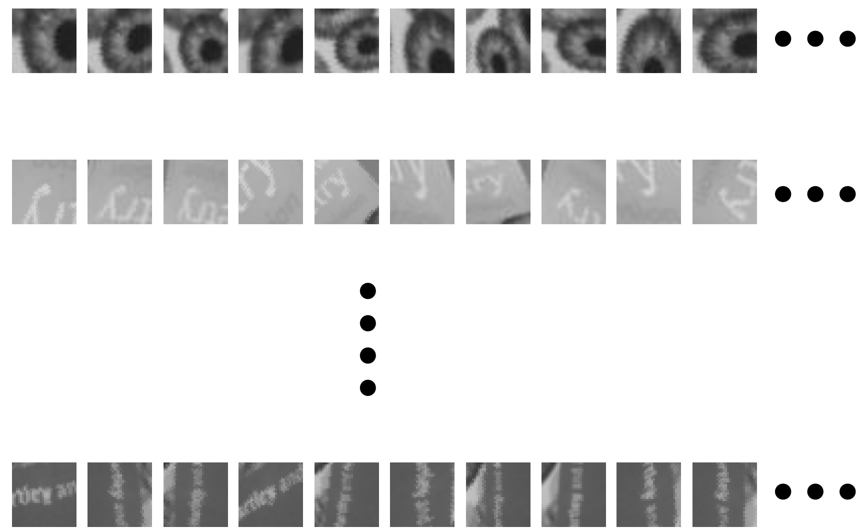
Model Image

Random Affine Transformation

Keypoint 1:

Keypoint 2:

Keypoint N:



Orientation normalization

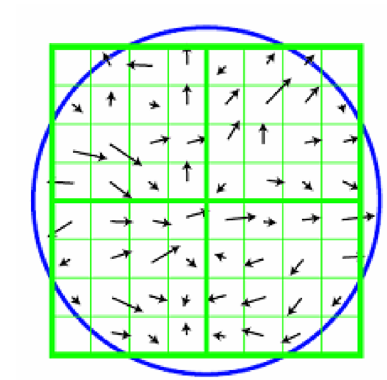
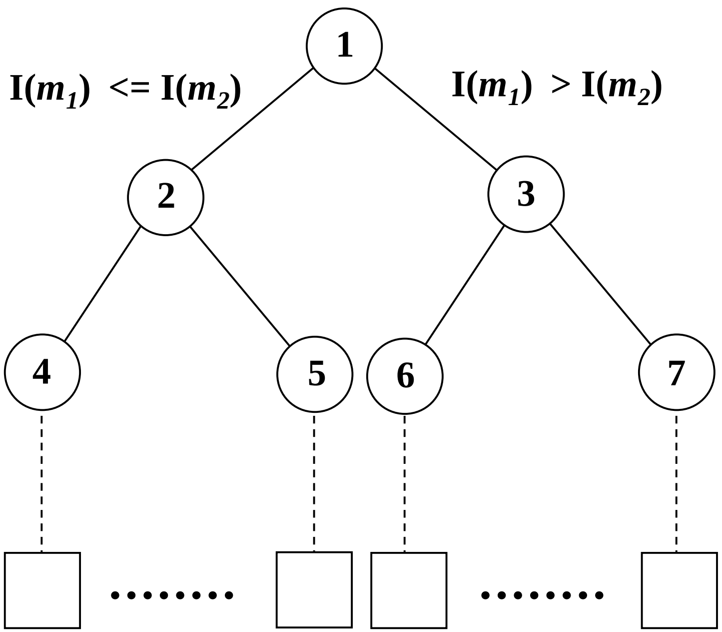


Image gradients

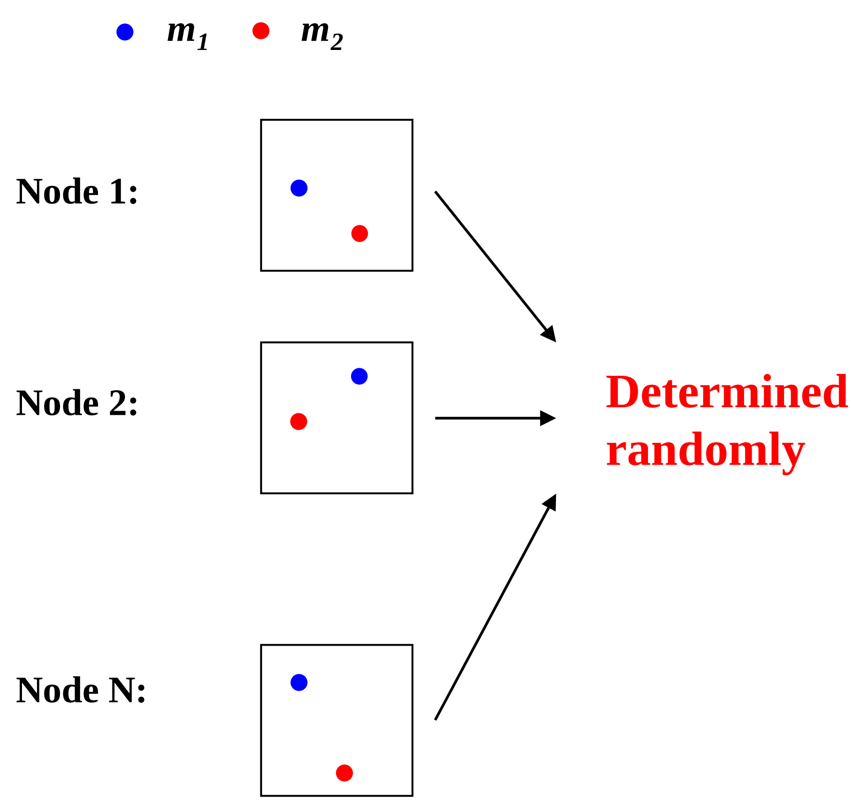
Training the Classification Tree

- Build a tree using random tests
  - The most simple approach



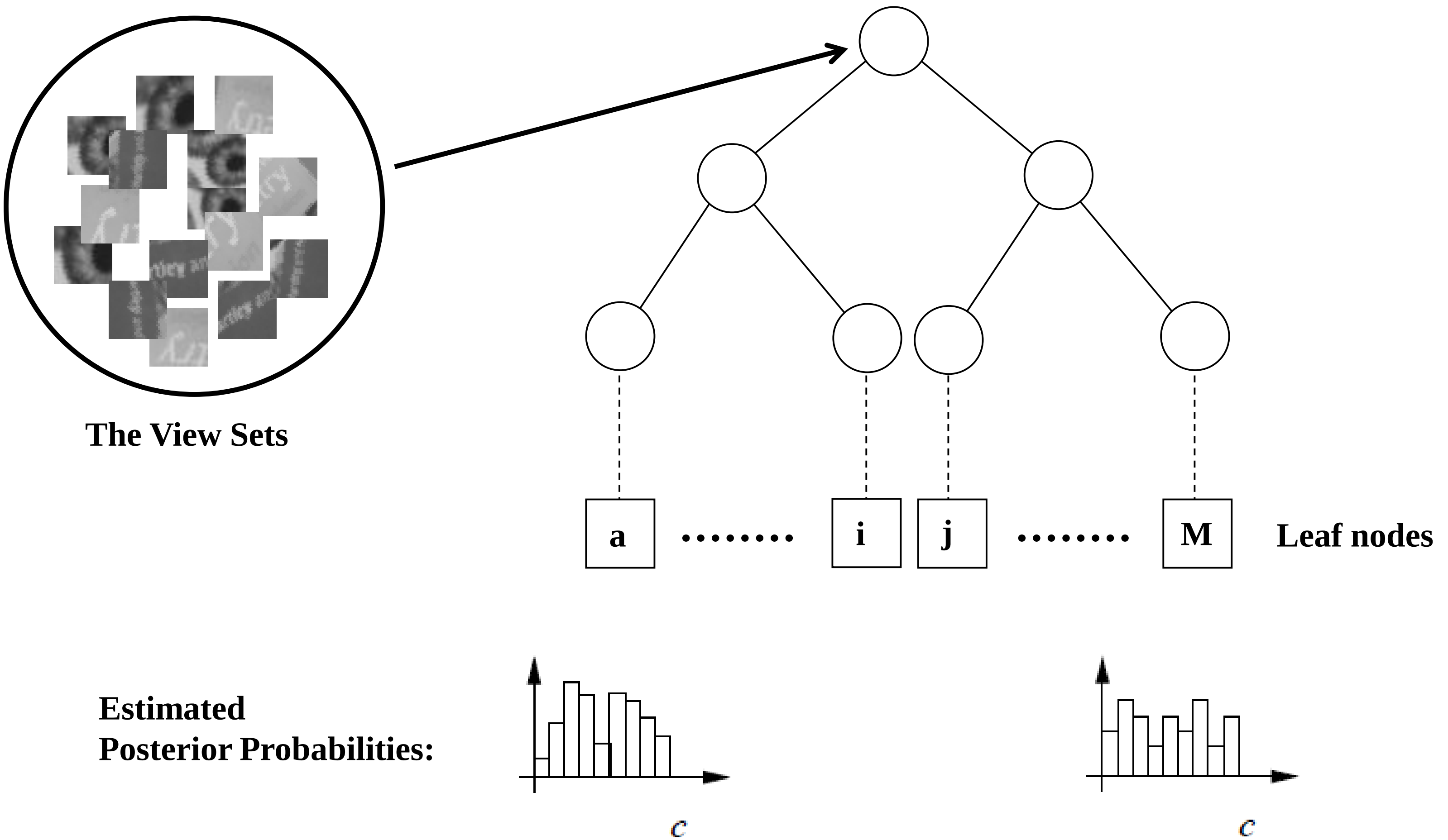
I(m_1) <= I(m_2) : goto the left child node

I(m_1) > I(m_2) : goto the right child node



Training the Classification Tree

- Estimate posterior probability at the leaf nodes

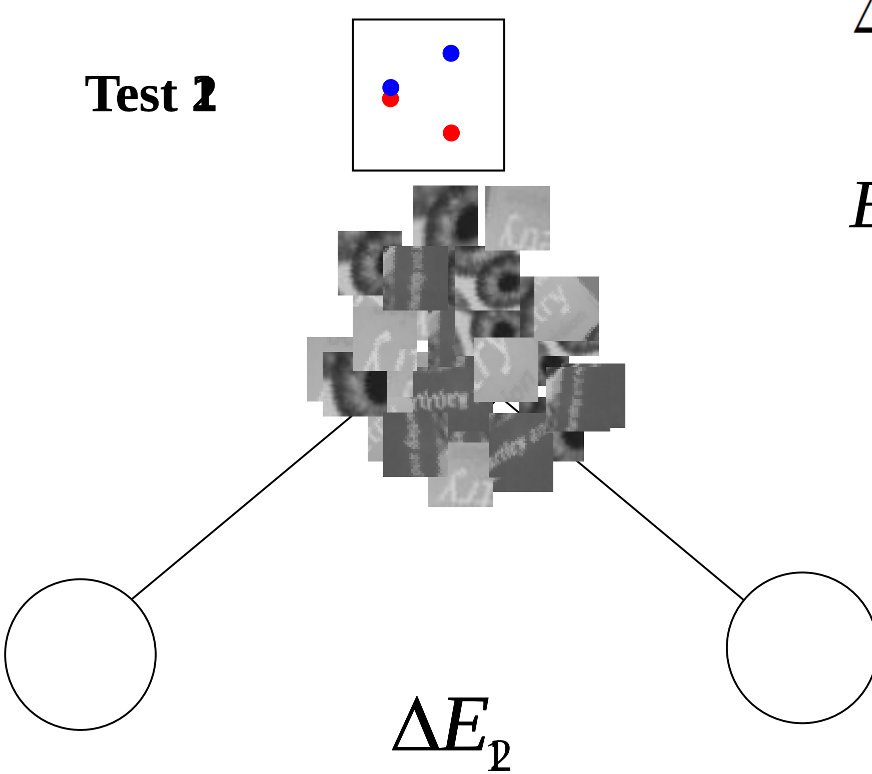


Tree Building Using Entropy Optimization

- The trees are constructed in the top-down manner
- The tests are chosen by a greedy algorithm to best separate the given examples

$$\Delta E = -\sum_i \frac{|S_i|}{|S|} E(S_i)$$

$$E(s) = -\sum_{j=1}^{N} p_j \log_2(p_j)$$

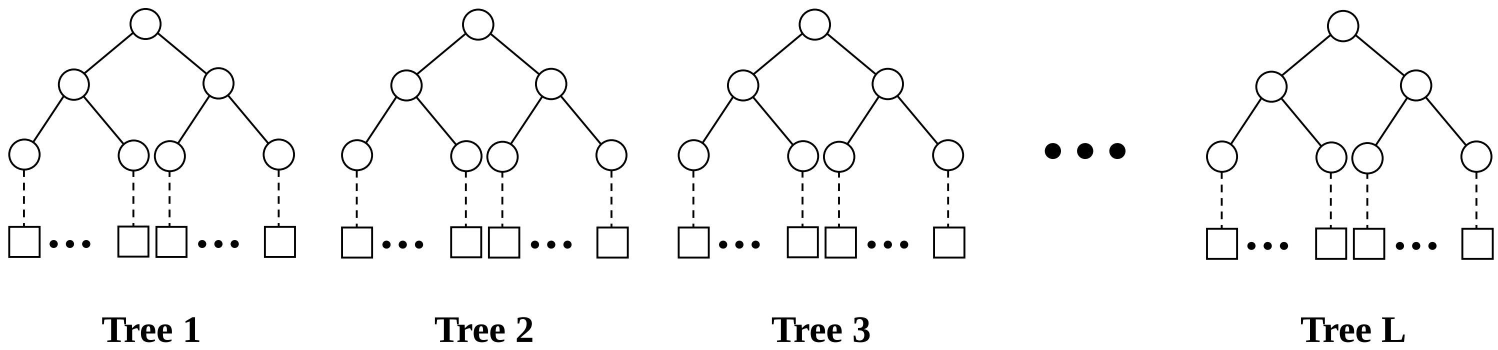


$$\Delta E_2 > \Delta E_1$$

: We select Test 2

Keypoint Recognition

- Build L classification trees using same methods
  - Each tree divide the patch space in different manner



- Use MAP estimation of the average of the posterior probabilities

$$\tilde{Y}(\mathbf{p}) = \arg\max_c p_c(\mathbf{p}) = \arg\max_c \frac{1}{L} \sum_{l=1\ldots L} P_{\eta(l,\mathbf{p})}(Y(\mathbf{p}) = c)$$

  - Input: a image patch **p**
  - Output: the class index *c*

Keypoint Recognition

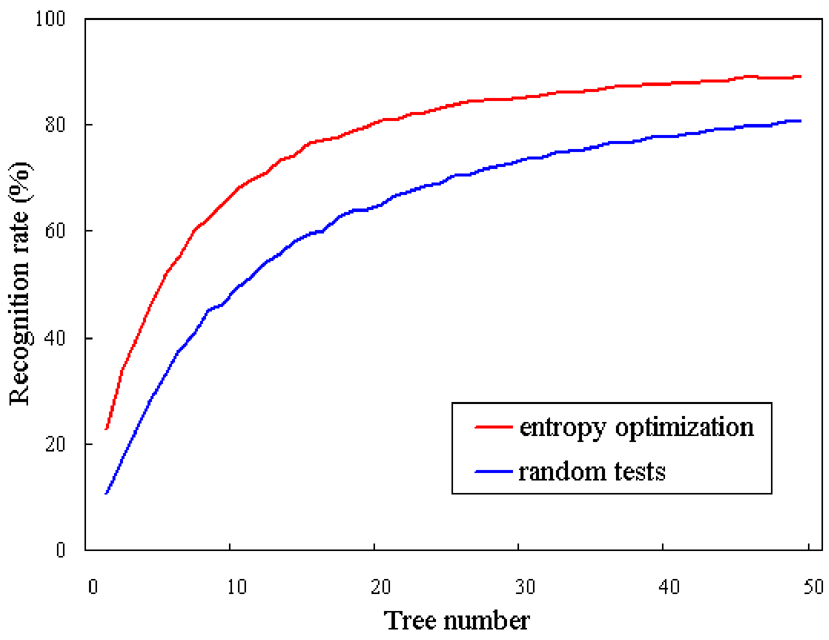
- Estimate a threshold during training to determine background or misclassified keypoints

$$P(Y(\mathbf{p}) = c | \tilde{Y}(\mathbf{p}) = c, p_c(\mathbf{p}) > T_c) > s$$

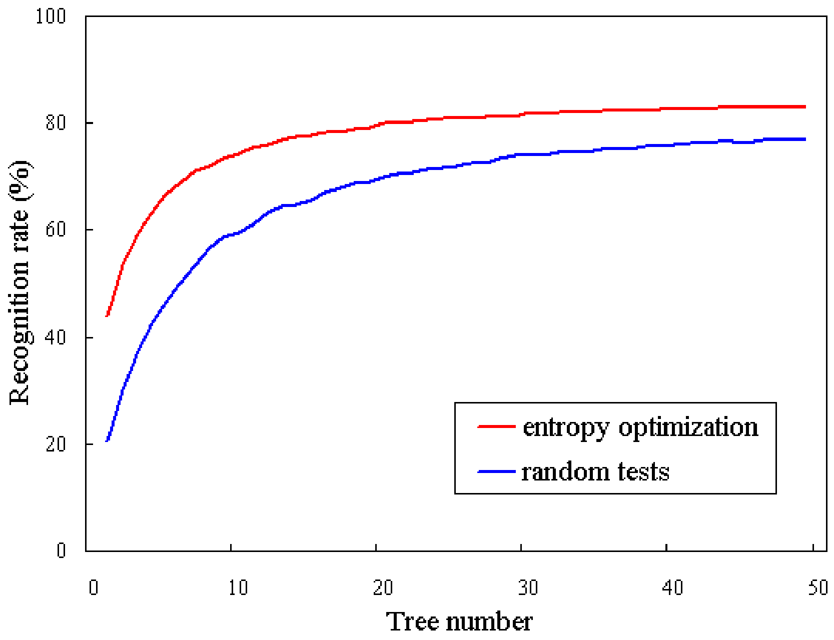
- Usage
  - $c' = \mathrm{argmax}_c \, p_c(\mathbf{p})$
  - If $p_{c'}(\mathbf{p}) > T_{c'}$, $\mathbf{p}$ is inlier
  - Otherwise, $\mathbf{p}$ is outlier

Experimental Results

- Comparing two tree building methods with or without orientation normalization
- Parameters: 200 keypoints, tree depth = 12, patch size = 32, 1000 view sets (200 x 1000 patches) for training and test trees
- Recognition rate R = # of correctly recognized patches / # of total test patches
- Tree building time: few seconds for random tests, tens of minutes for entropy opt.



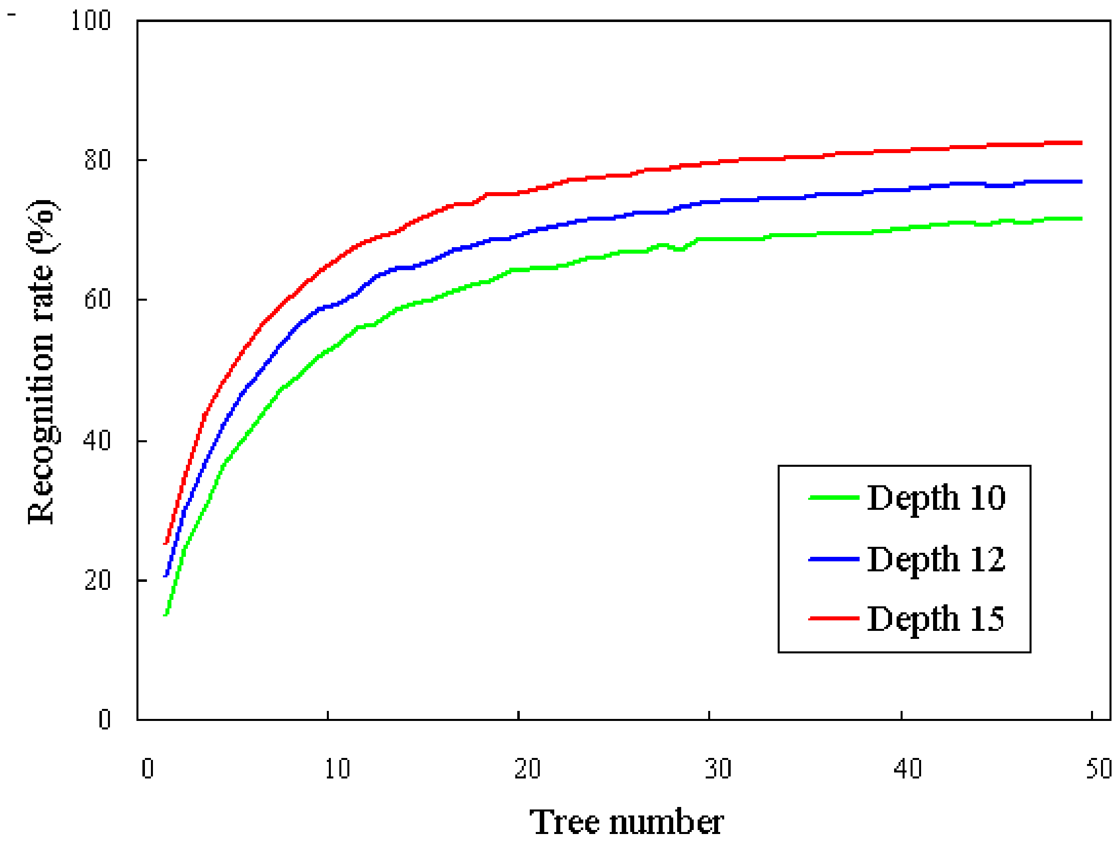
Without orientation normalization



With orientation normalization

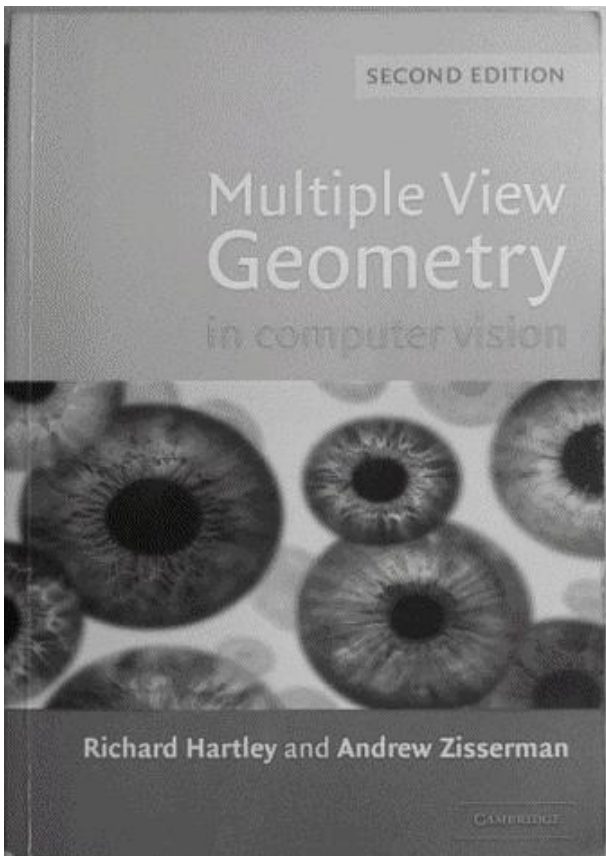
Experimental Results

- Testing the effect of the tree depth
- Parameters: 200 keypoints, 1000 view sets (200 x 1000 patches) for training and test trees



Experimental Results

- Real time object detection and pose estimation test
- Parameters: 400 keypoint, tree depth = 12, tree number = 15, patch size = 32
- Get nearest neighbor by classification tree
- Affine fitting using RANSAC, followed by homography estimation



About 3 min for learning, under 100 ms for detection for 3.4 Ghz Pentium machine

References

- [1] V. Lepetit and P. Fua. Keypoint Recognition using Randomized Trees. *IEEE Trans. Pattern Anal. Machine Intell.*, 28(9):1465–1479, 2006.

- [2] V. Lepetit and P. Fua. Towards Recognizing Feature Points using Classification Trees. EPFL Technical Report, 2004.

- [3] V. Lepetit, J. Pilet, and P. Fua. Point Matching as a Classification Problem. CVPR, 2004.