FERNs : Matching as Classification

Presented by Chen Feng

Outline

- Background
- General Schema
- First Try – K-Means + Nearest Neighbor Search
- Second Try – Randomized Trees
- Final Method – FERNs
- Modification for Mobile Platform

Outline

- Background
- General Schema
- First Try – K-Means + Nearest Neighbor Search
- Second Try – Randomized Trees
- Final Method – FERNs
- Modification for Mobile Platform

Background

- Problem to solve
  - Object Tracking and Pose Estimation
- Goals to Achieve
  - Real-time performance
  - Accuracy
- Previous Methods
  - Detector + Descriptor
  - Point matching too expensive!
- Another way to look at the problem ?

Matching as Classification?

- Object are represented by

$$\text{a set } \mathbf{F_O} \text{ of } N \text{ prominent feature points}$$

- Given an input image patch $\mathbf{p} \in \mathbf{P}$, where P is a space of all image patches of a given size

- Label p with

$$Y(\mathbf{p}) \in \mathbf{C} = \{-1, 1, 2, \ldots, N\}$$

- Since Y can not be directly observed, we need to construct a classifier

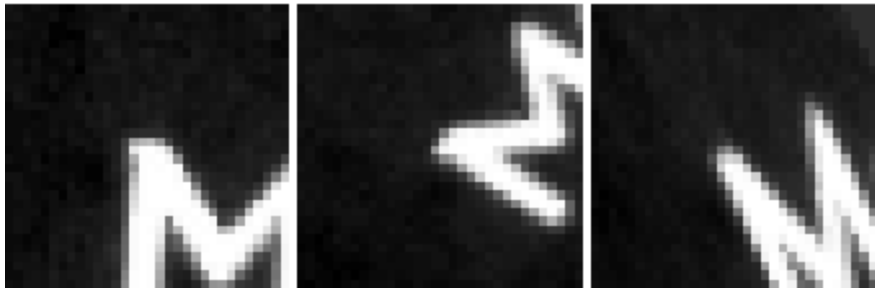
$$\hat{Y} : \mathbf{P} \rightarrow \mathbf{C}$$

Outline

- Background
- General Schema
- First Try – K-Means + Nearest Neighbor Search
- Second Try – Randomized Trees
- Final Method – FERNs
- Modification for Mobile Platform

General Schema (I)

- First step : Feature point extraction + Create ViewSets
- Feature point extraction
  - Not our focus, use Harris corner detector, or whatever…
- Create ViewSets
  - the set of all its possible appearances for a feature point
  - Random sampling (affine transform) under local planarity assumption



General Schema (II)

- Second Step : Classification
- Several trys until they get to FERNs

Outline

- Background
- General Schema
- First Try – K-Means + Nearest Neighbor Search
- Second Try – Randomized Trees
- Final Method – FERNs
- Modification for Mobile Platform

First Try

- Keywords: PCA, K-means, NN search
- 200 feature points extracted, each point represented by a viewset made of 100 samples, sample patch size 32*32
- Keep first 20 dimensions from PCA, each viewset classified with 20-means
- Time : 200 milliseconds on a 2GHz PC
- Pretty decent result, but not fast enough…

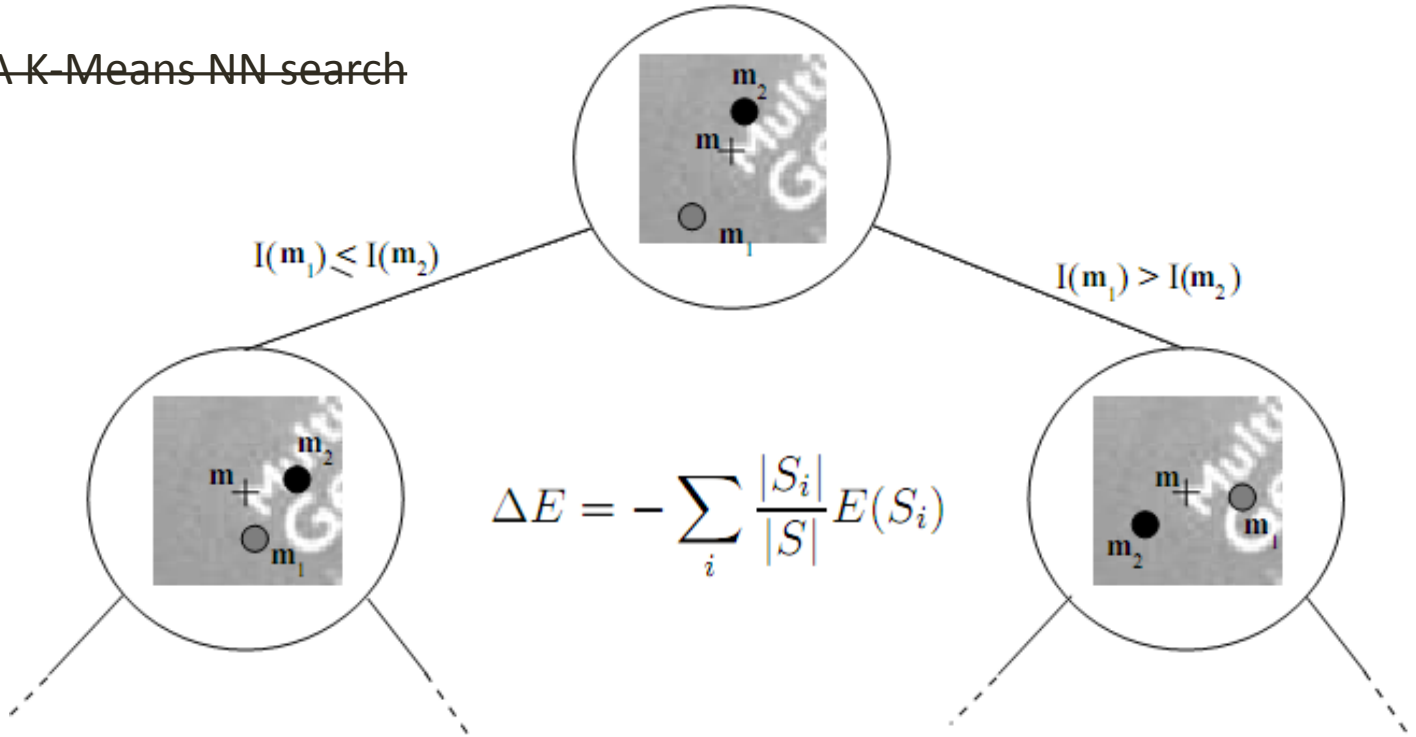
First Try : Experiment

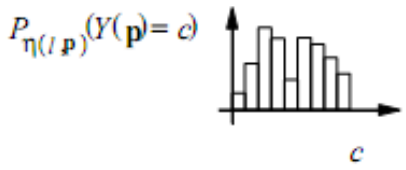
Outline

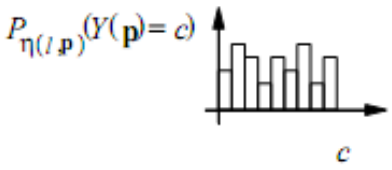
- Background
- General Schema
- First Try – K-Means + Nearest Neighbor Search
- Second Try – Randomized Trees
- Final Method – FERNs
- Modification for Mobile Platform

Second Try

~~PCA K-Means NN search~~







Second Try

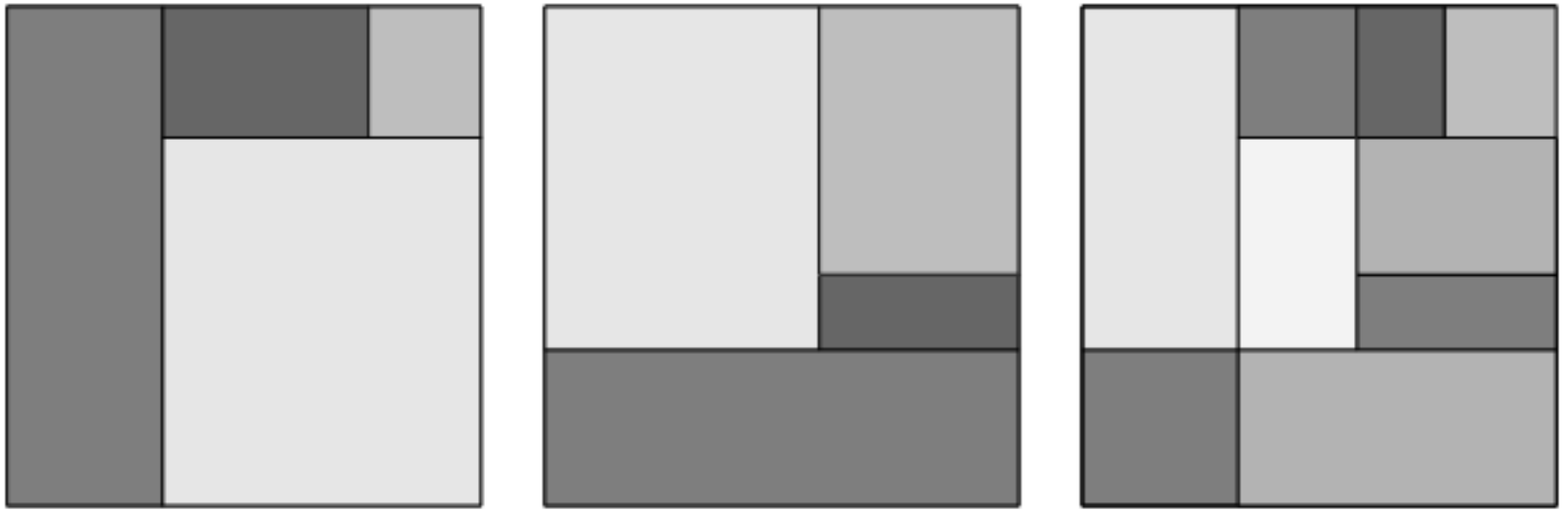
$$\hat{Y}(\mathbf{p}) = \underset{c}{\operatorname{argmax}}\ p_c(\mathbf{p}) = \underset{c}{\operatorname{argmax}}\ \frac{1}{L} \sum_{l=1\ldots L} P_{\eta(l,\mathbf{p})}(Y(\mathbf{p}) = c)$$

the tree leaves store posterior probabilities $P_{\eta(l,\mathbf{p})}(Y(\mathbf{p}) = c)$

c is a label in $\mathbf{C}$

$\eta(l, \mathbf{p})$ is the leaf of tree T_l reached by the patch $\mathbf{p}$

Second Try



- Heuristic

  when the tests are chosen randomly, the power of the approach derives not from the tree structure itself but from the fact that combining groups of binary tests allows improved classification rates.

- Lead us to …

Outline

- Background
- General Schema
- First Try – K-Means + Nearest Neighbor Search
- Second Try – Randomized Trees
- Final Method – FERNs
- Modification for Mobile Platform

Final Method

We are looking for $\arg\max_i P(C = c_i \mid \mathbf{patch})$

If **patch** can be represented by a set of image features $\{ f_i \}$:

$$P(C = c_i \mid \mathbf{patch}) = P(C = c_i \mid f_1, f_2, \ldots f_n, f_{n+1}, \ldots \ldots f_N)$$

which is proportional to

$$P(f_1, f_2, \cdots f_n,\ f_{n+1}, \cdots \cdots f_N \mid C = c_i)$$

but complete representation of the joint distribution infeasible.

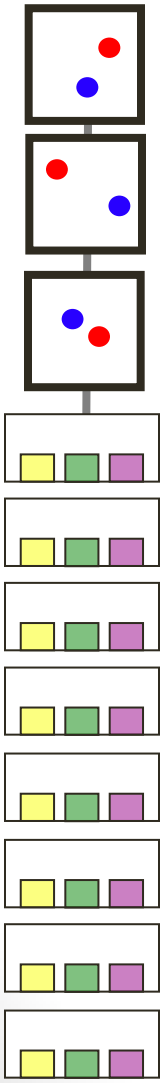
Naive Bayesian ignores the correlation:

$$\approx \prod_j P(f_j \mid C = c_i)$$

Compromise:

$$\approx P(f_1, f_2, \cdots f_n \mid C = c_i) \times P(f_{n+1}, \cdots f_{2n} \mid C = c_i) \times \cdots$$

Training



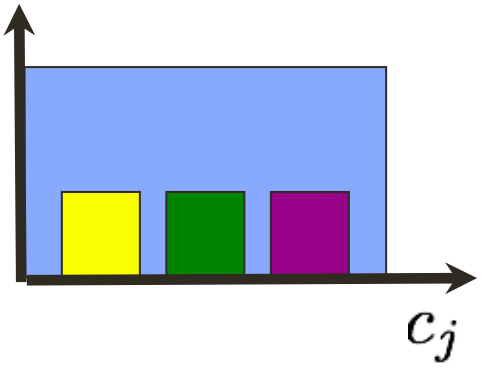
The tests compare the intensities of two pixels around the keypoint:

$$f_i = \begin{cases} 1 & \text{if } I(m_{i,1}) \leq I(m_{i,2}) \\ 0 & \text{otherwise} \end{cases}$$

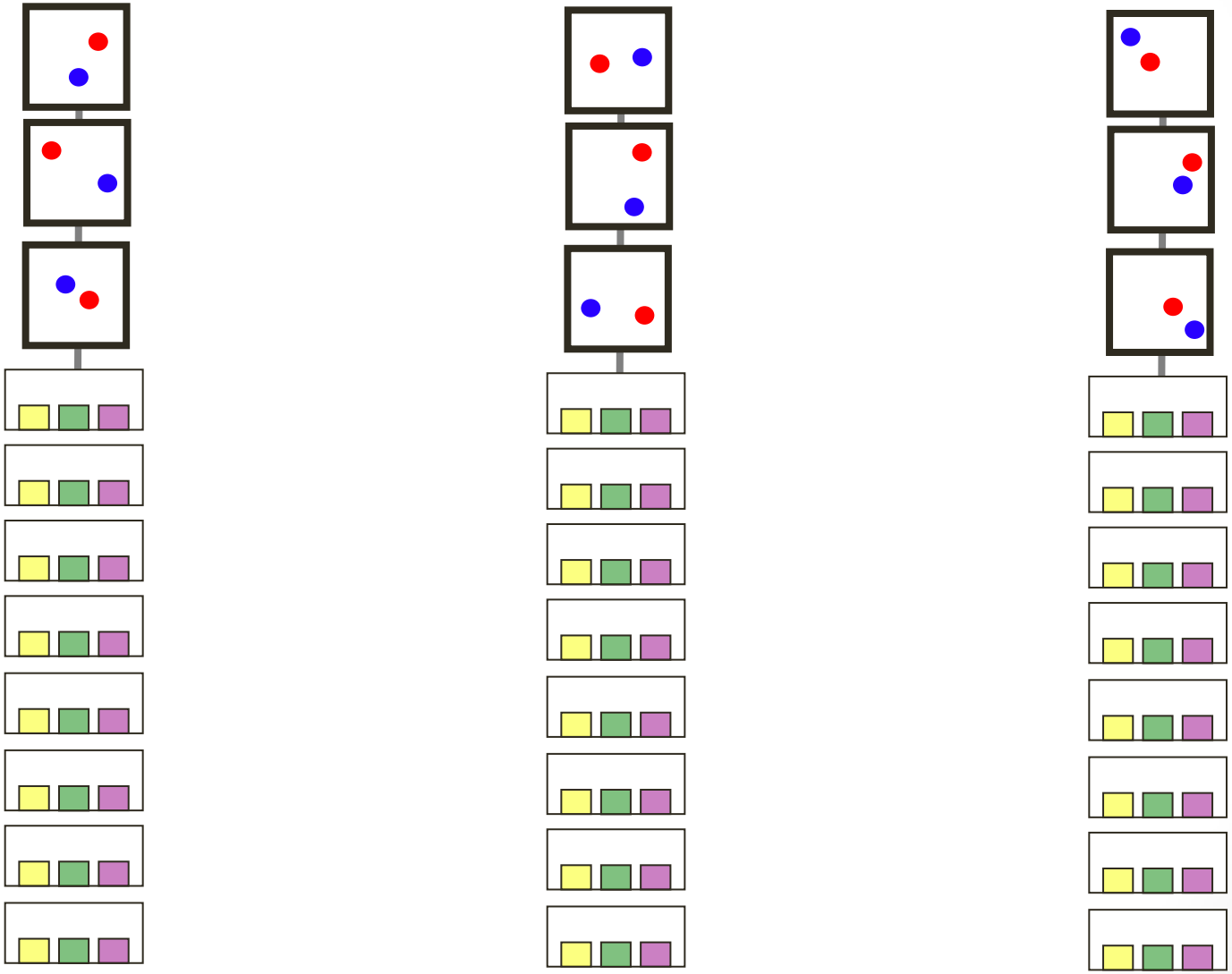
Invariant to light change by any raising function.

Posterior probabilities:

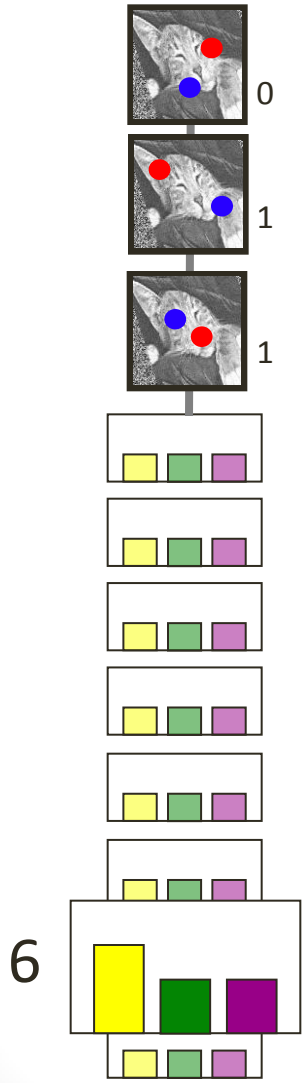
$$P(f_1, f_2, \cdots f_n \mid C = c_j)$$

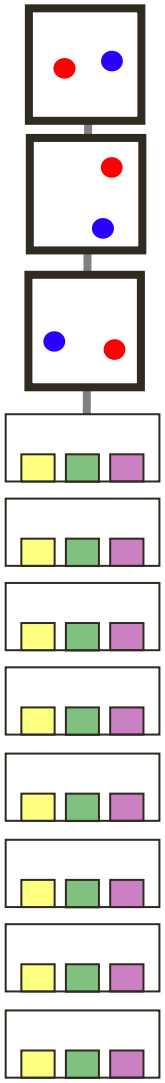


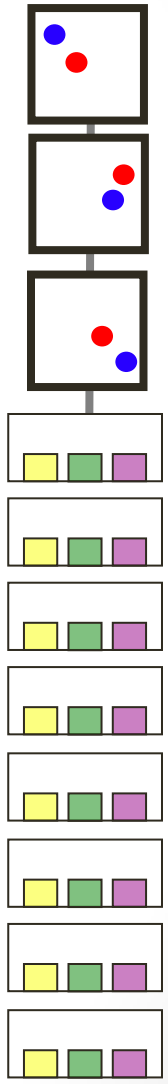
Training



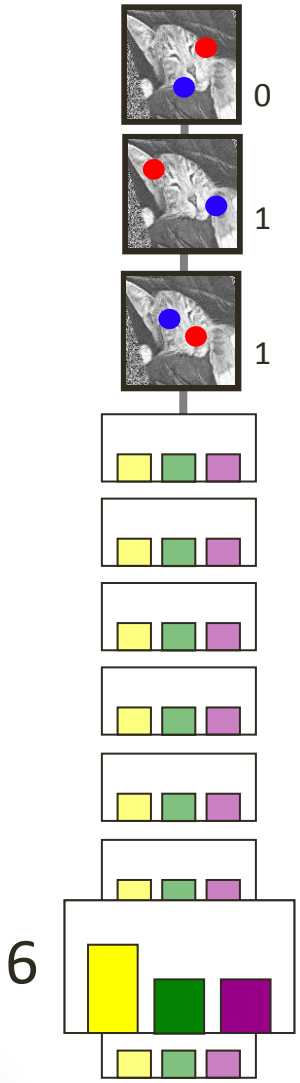
Training

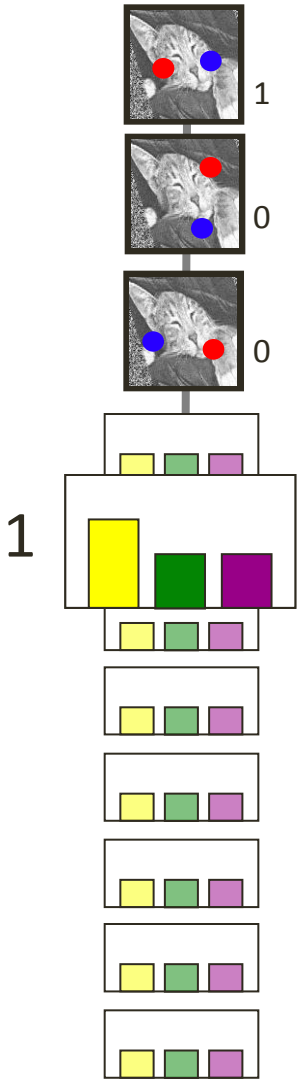


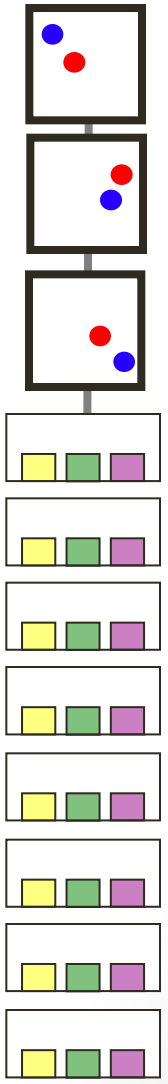




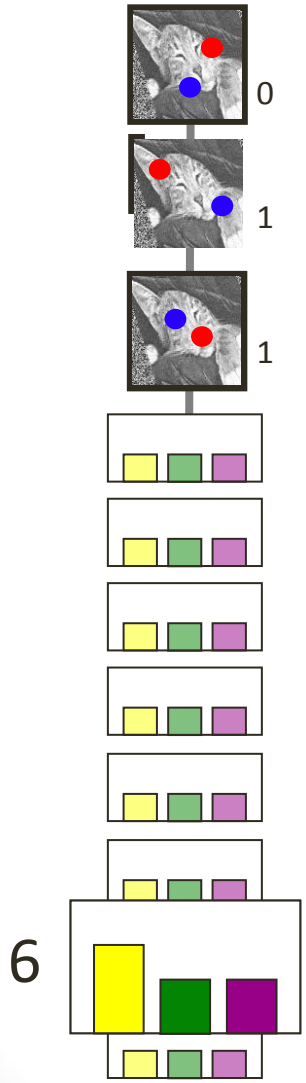
Training

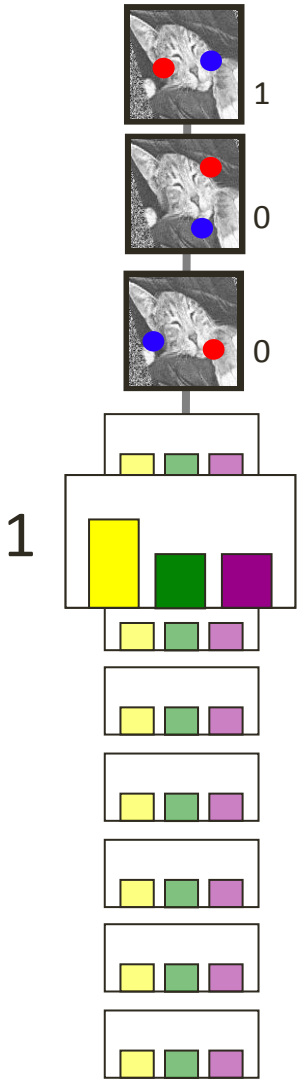


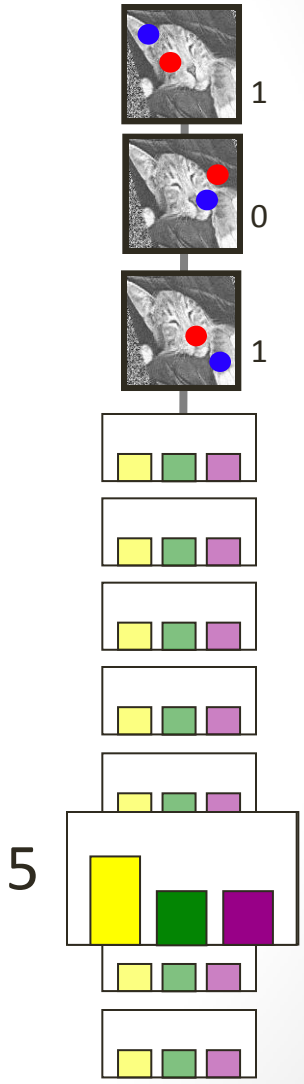




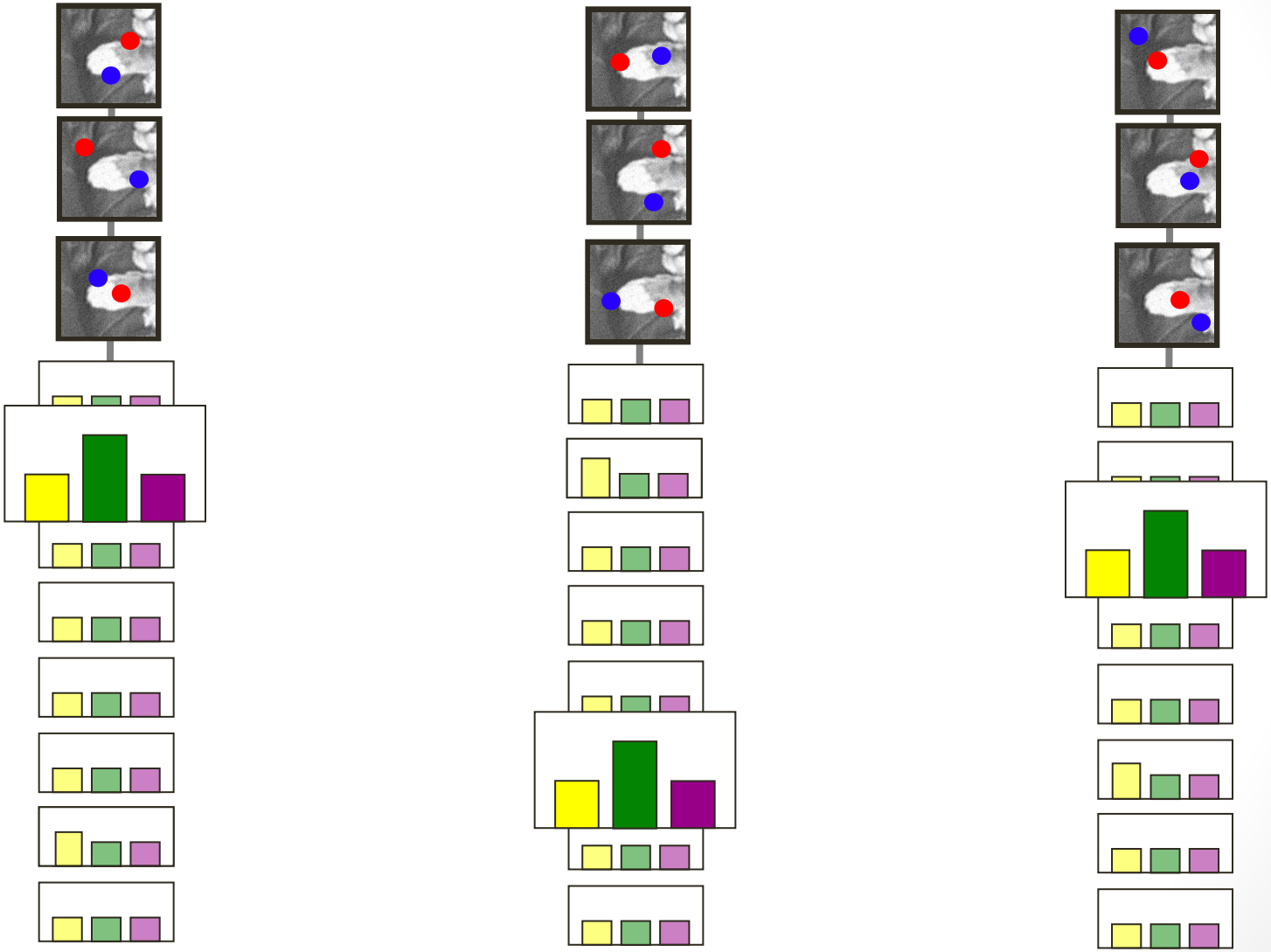
Training



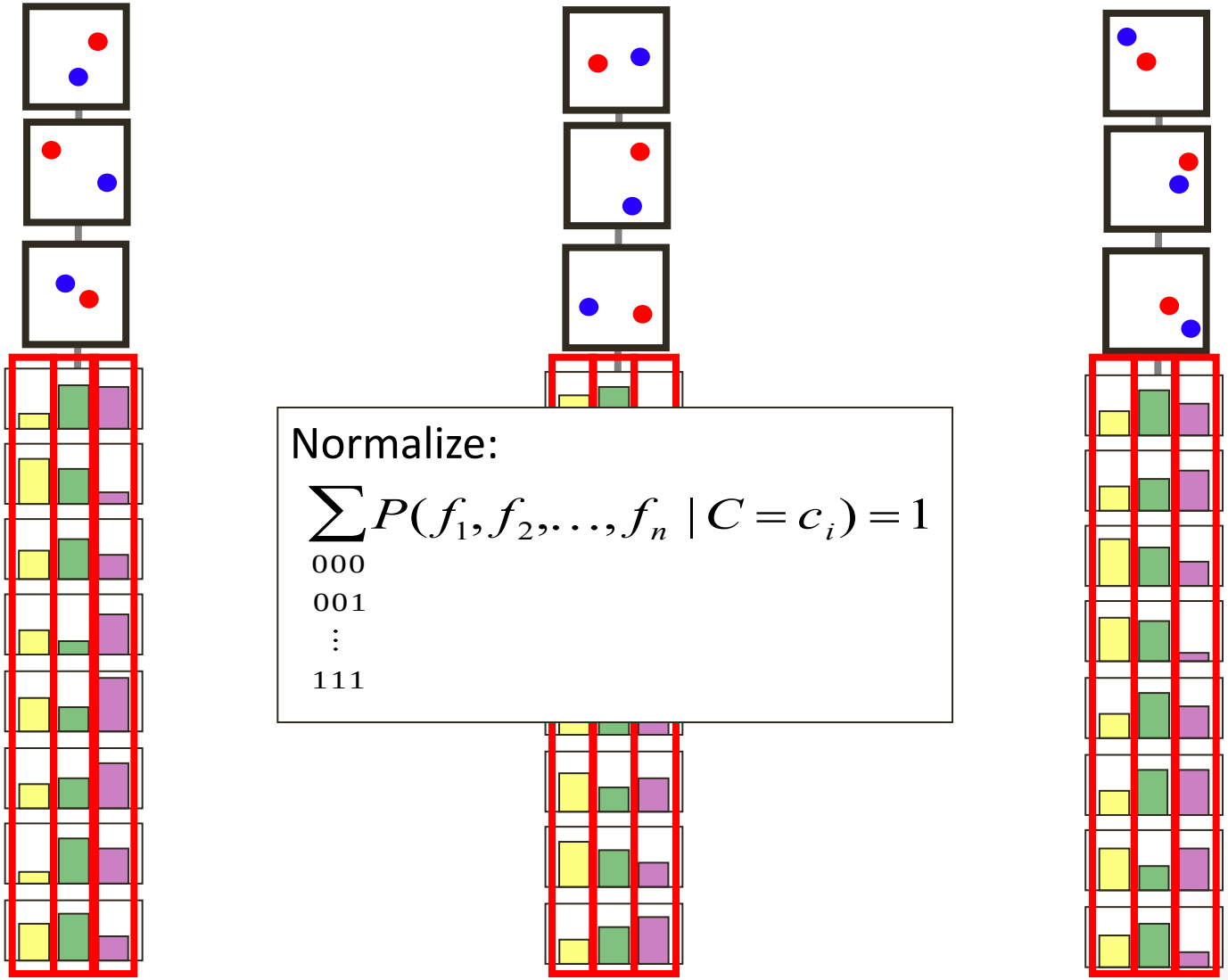




Training



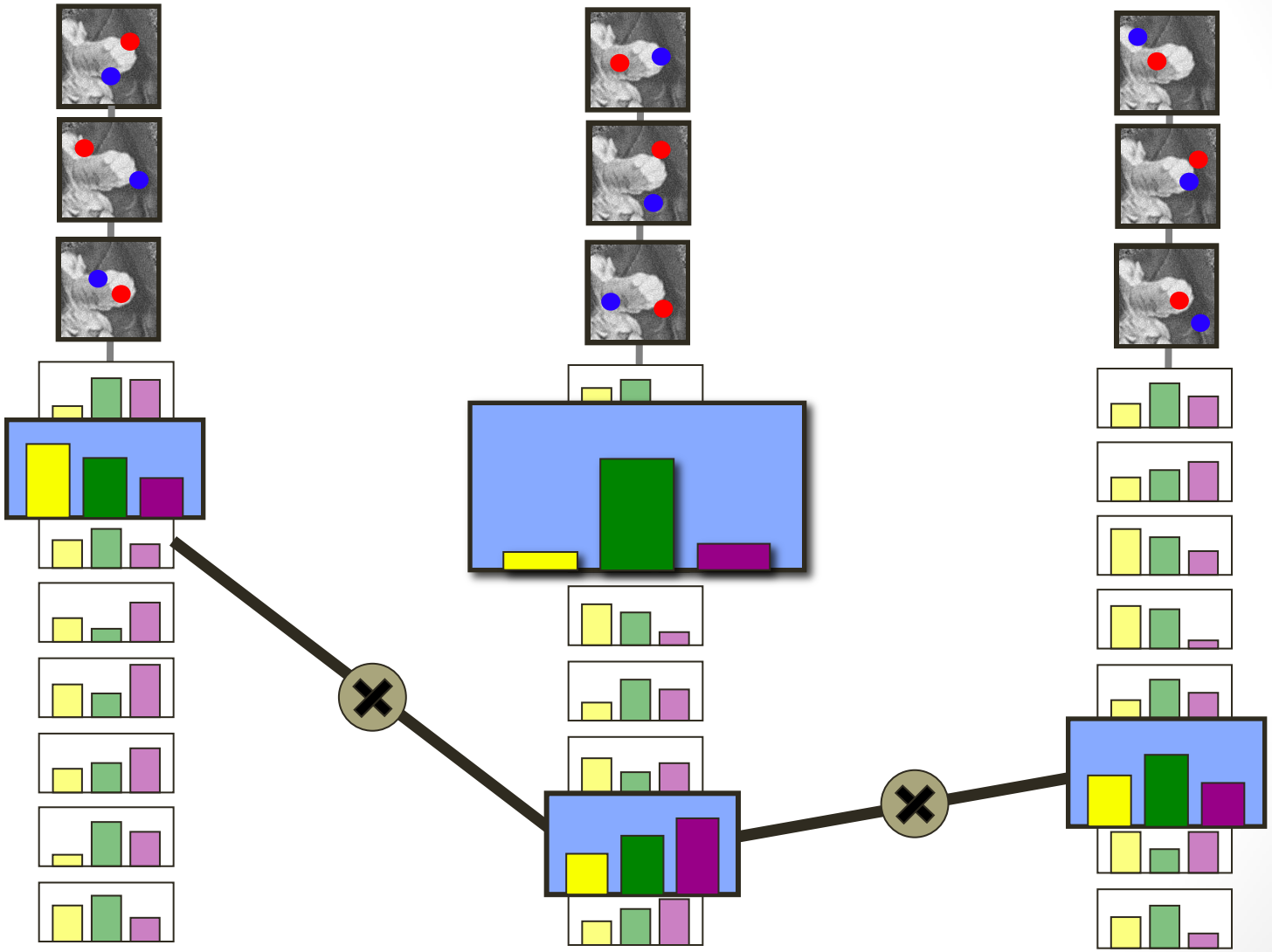
Training Results



Normalize:

$$\sum_{000}^{111} P(f_1, f_2, \ldots, f_n \mid C = c_i) = 1$$

Recognition



Keypoint Recognition in Ten Lines of Code

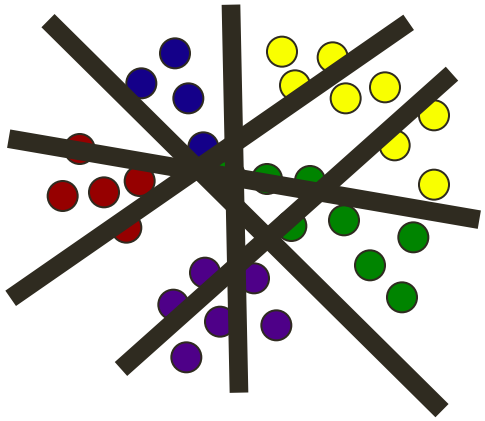
```
 1: for(int i = 0; i < H; i++) P[i ] = 0.;
 2: for(int k = 0; k < M; k++) {
 3:   int index = 0, * d = D + k * 2 * S;
 4:   for(int j = 0; j < S; j++) {
 5:     index <<= 1;
 6:     if (*(K + d[0]) < *(K + d[1]))
 7:       index++;
 8:     d += 2;
      }
 9:   p = PF + k * shift2 + index * shift1;
10:   for(int i = 0; i < H; i++) P[i] += p[i];
     }
```

Very simple to implement;
No need for orientation nor perspective correction;
(Almost) no parameters to tune;
Very fast.

We Can Use Random Tests

For a small number of classes

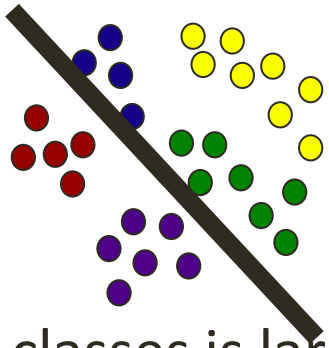
- we can try several tests, and
- retain the best one according to some criterion.



We Can Use Random Tests

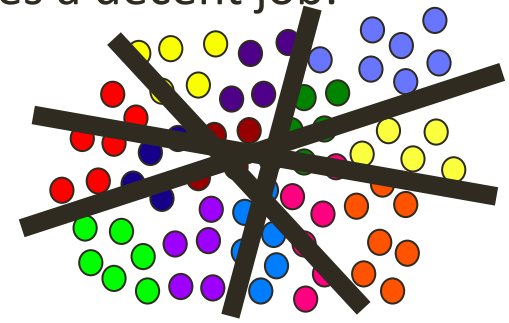
For a small number of classes

- we can try several tests, and
- retain the best one according to some criterion.

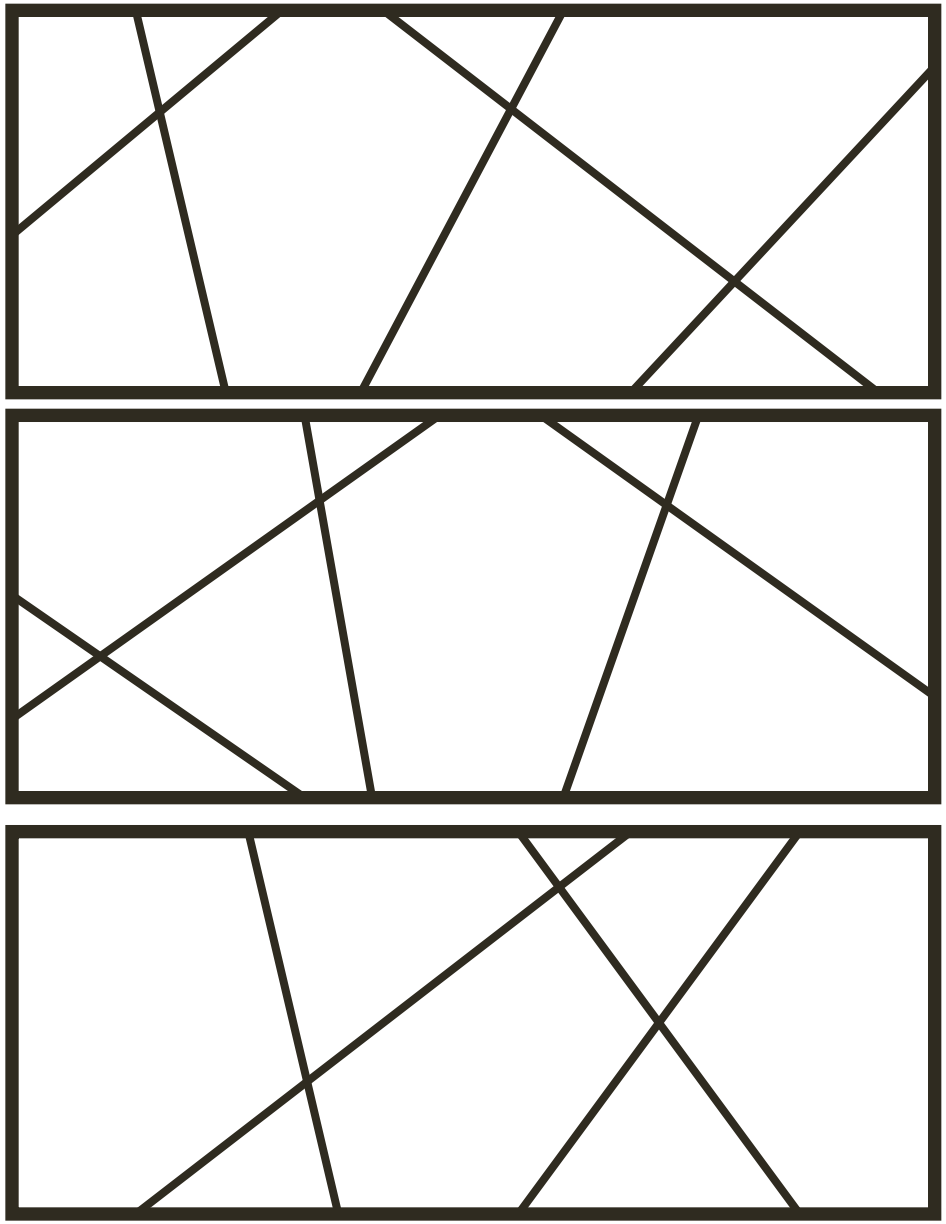


When the number of classes is large

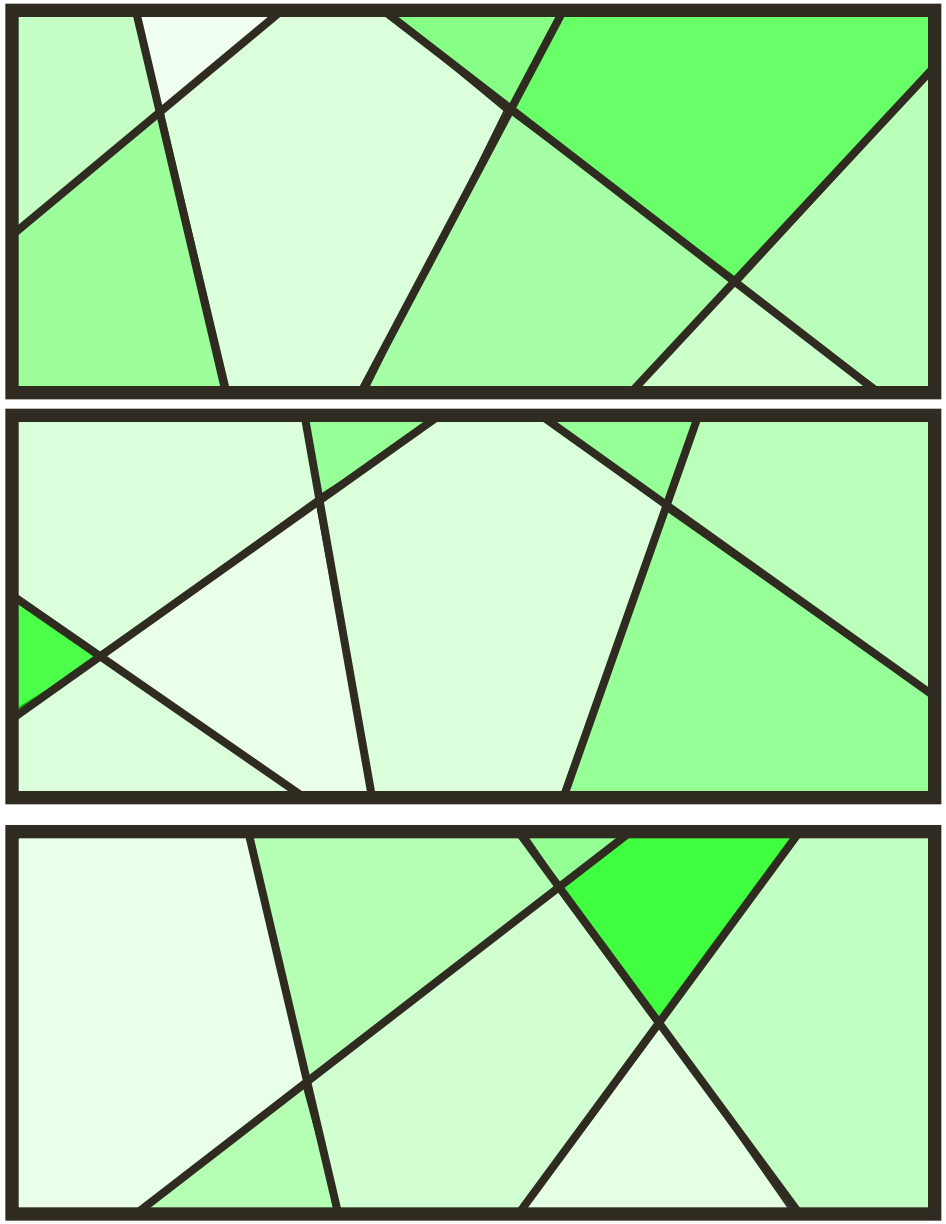
- any test does a decent job:



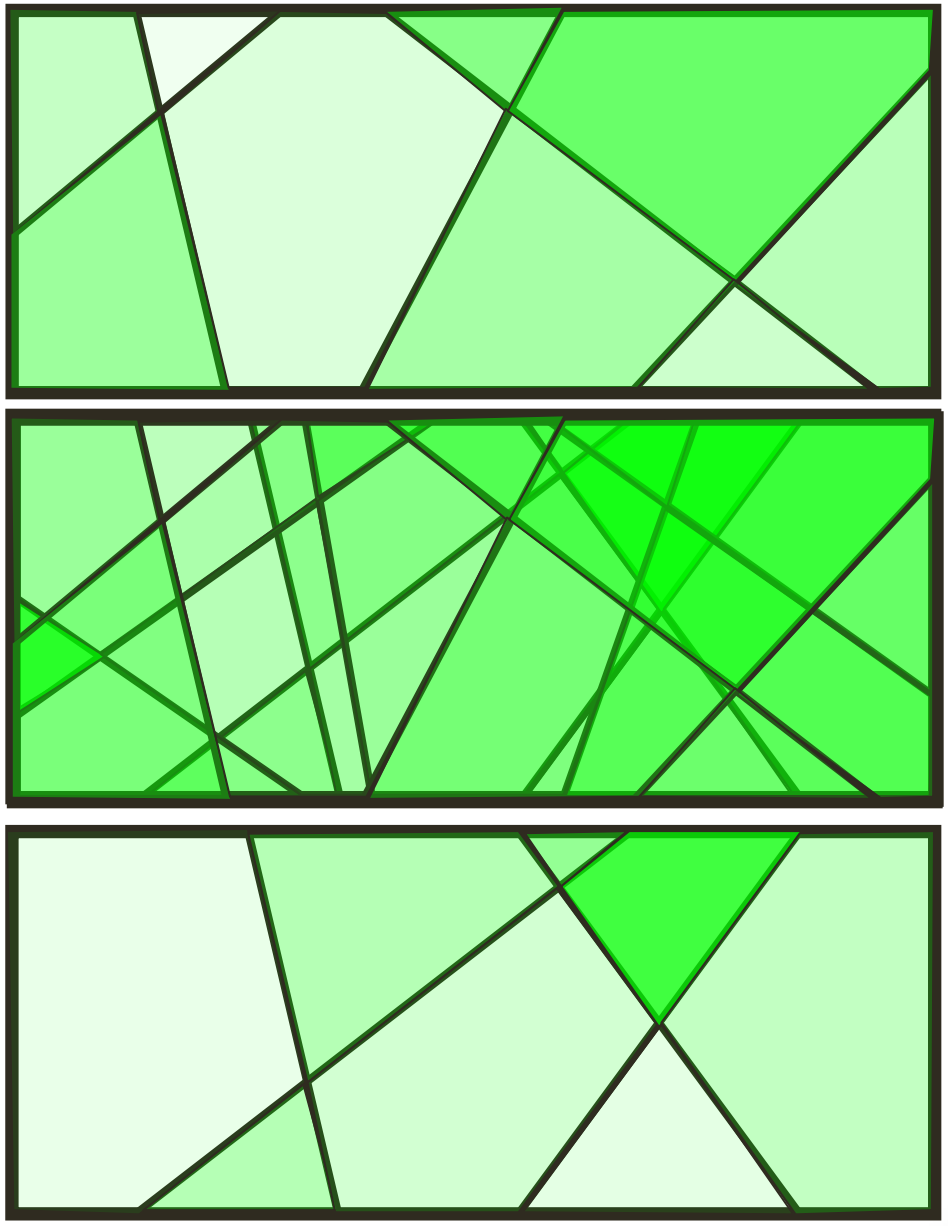
Another Graphical Interpretation



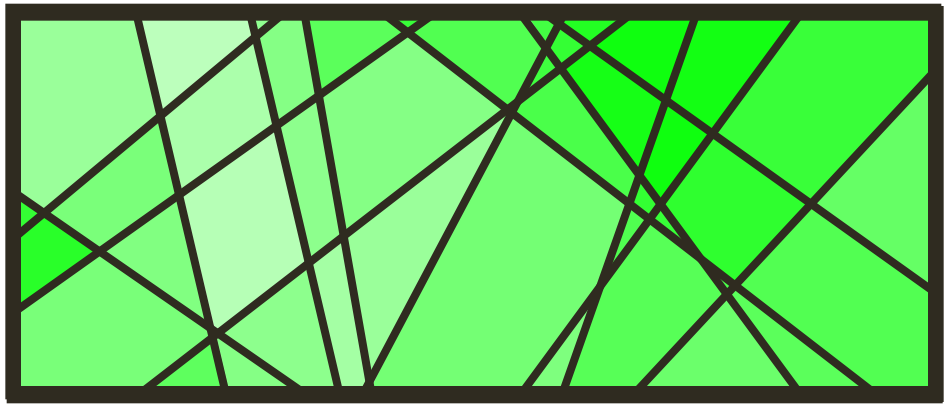
Another Graphical Interpretation



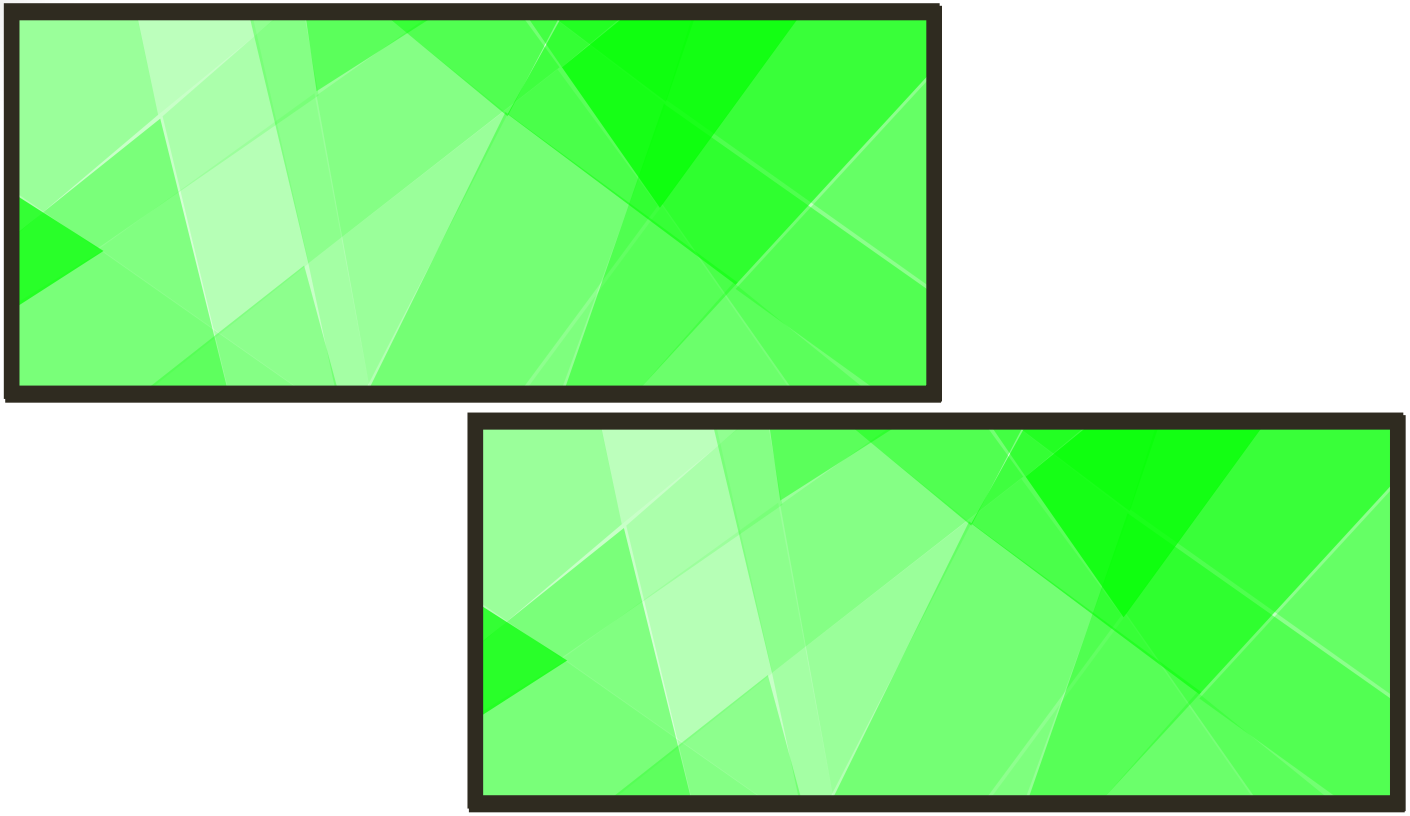
Another Graphical Interpretation



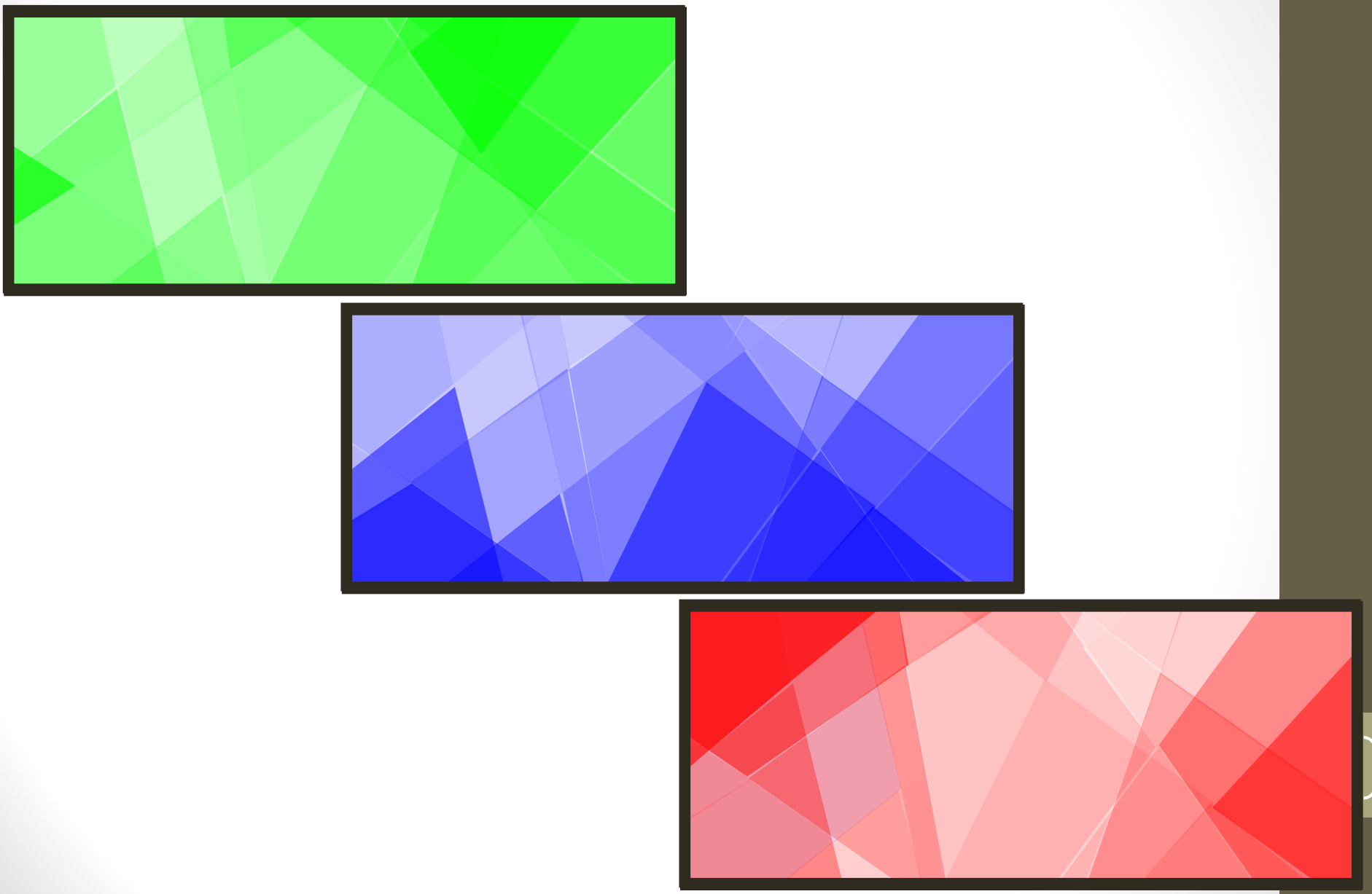
Another Graphical Interpretation



Another Graphical Interpretation



Another Graphical Interpretation



Outline

- Background
- General Schema
- First Try – K-Means + Nearest Neighbor Search
- Second Try – Randomized Trees
- Final Method – FERNs
- Modification for Mobile Platform

END

Thank You!

Any Questions?