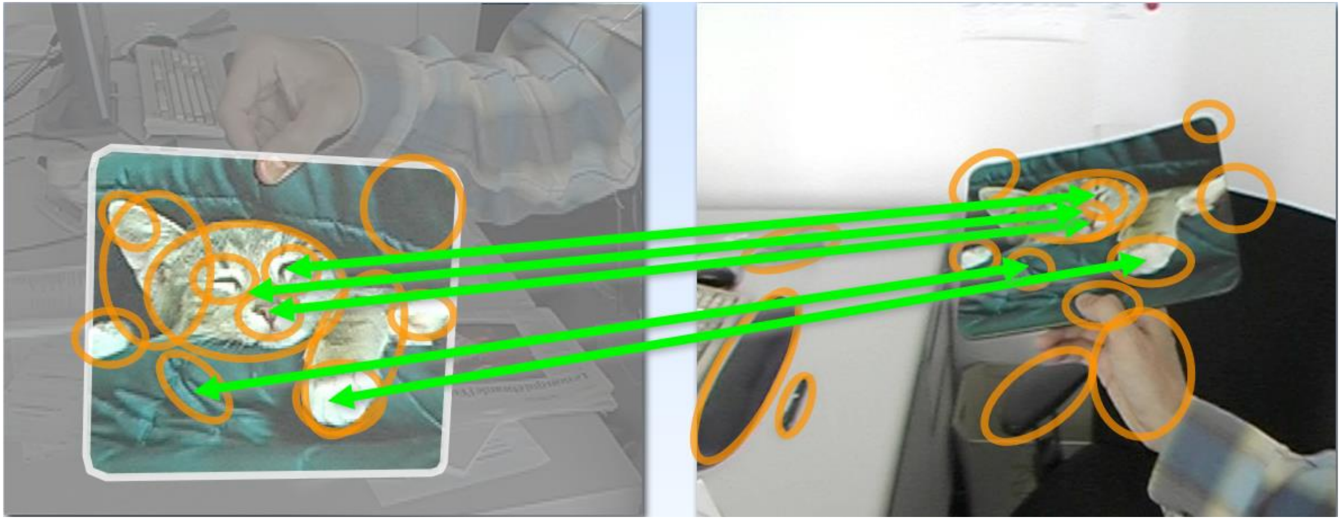
Ferns for traffic sign detection

- Thomas Marschall, Sebastian Lerch, Christoph Ihrke, Sahand Sharifzadeh
- Team Triforce

[Source: trafficsignstore.com]

Training

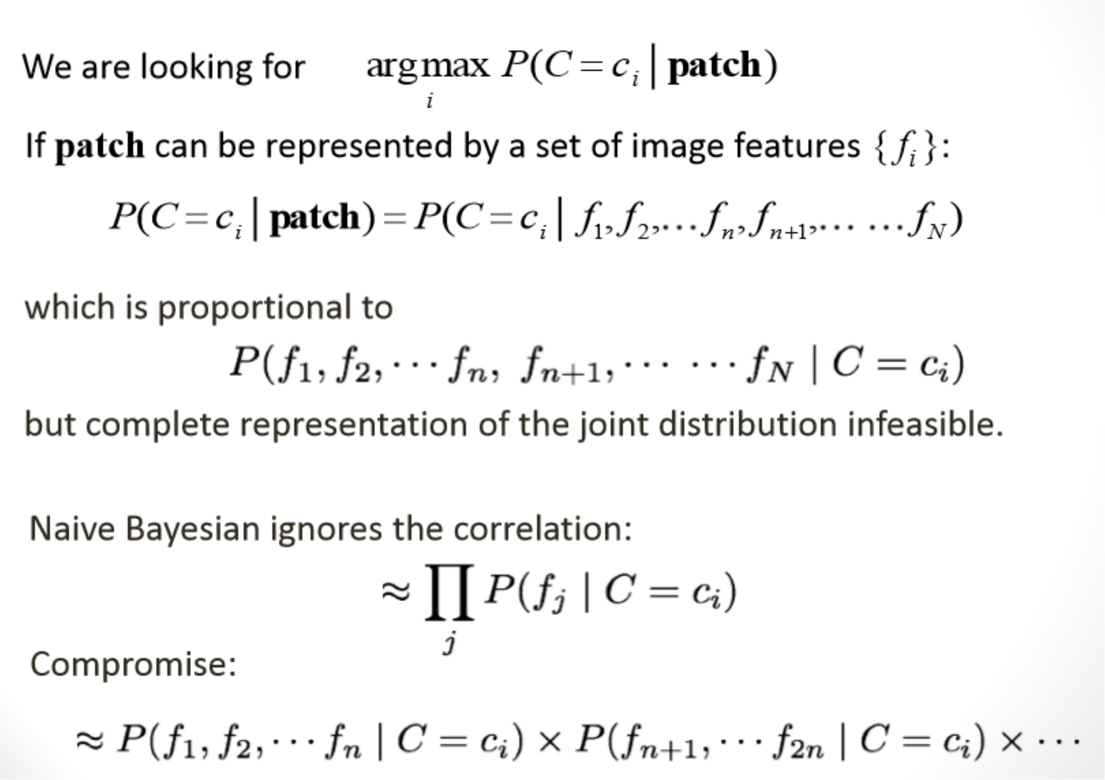


Trained Image

Input Image

[Source: campar.in.tum.de]

Training



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:

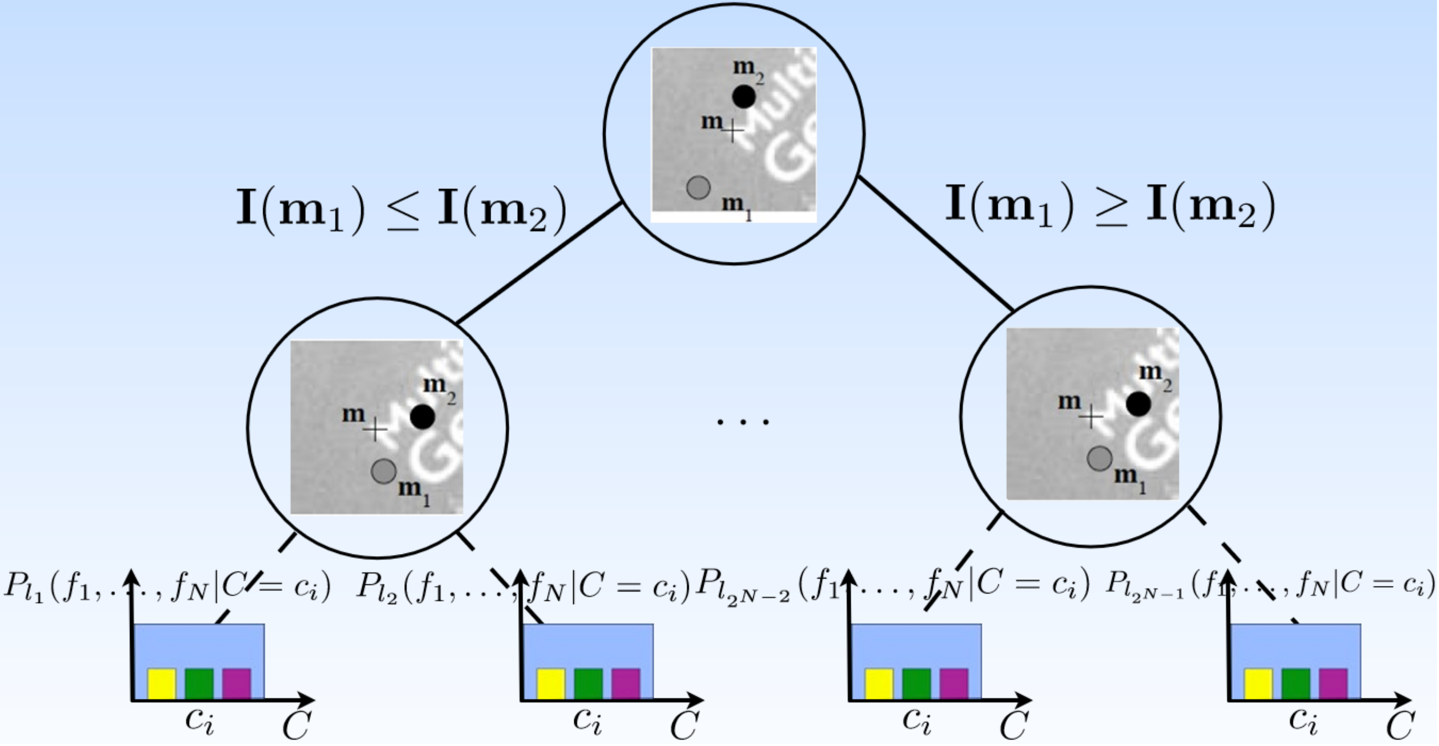
$$\approx \prod_{j} P(f_j \mid C=c_i)$$

Compromise:

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

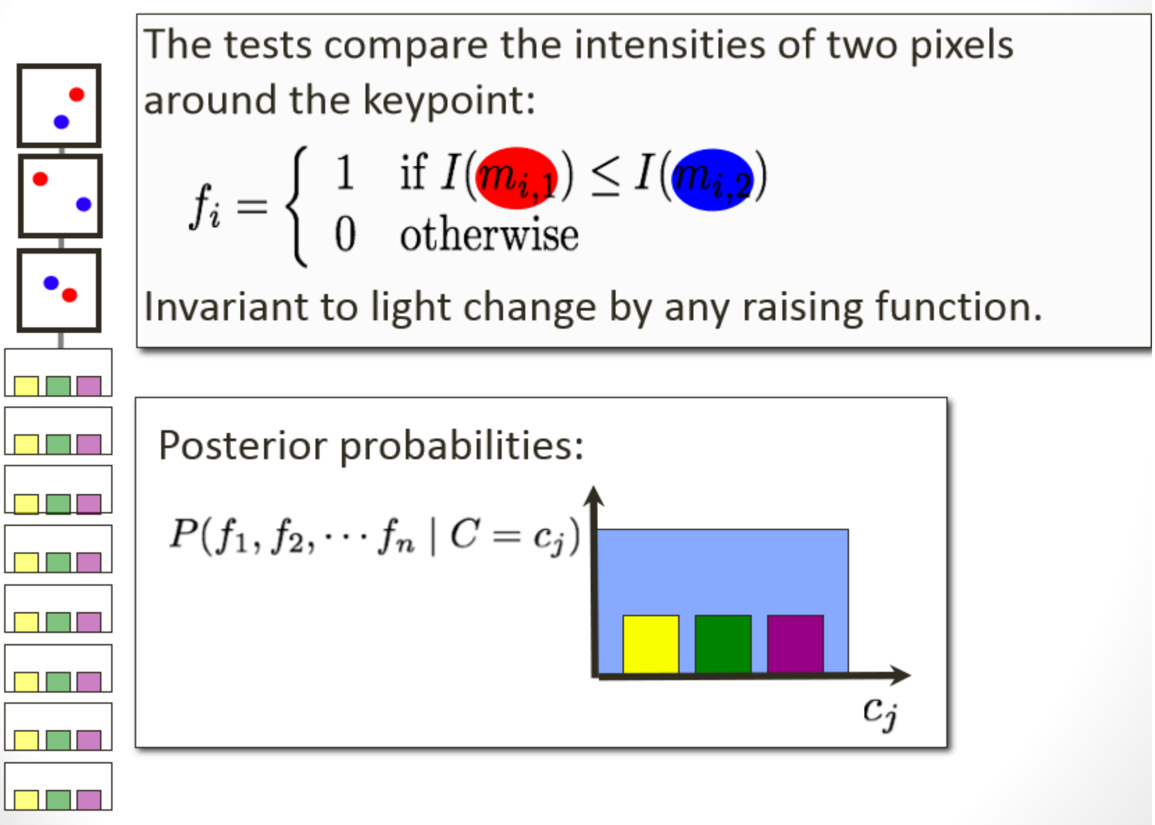
[Source:web.eecs.umich.edu/~silvio/teaching/EECS598_2010]

Training


$\mathbf{I}(\mathbf{m}_1) \leq \mathbf{I}(\mathbf{m}_2)$
$\mathbf{I}(\mathbf{m}_1) \geq \mathbf{I}(\mathbf{m}_2)$
$\mathbf{m}_2$
$\mathbf{m}$
$\mathbf{m}_1$
$\cdots$
$P_{l_1}(f_1, \ldots, f_N | C = c_i)$
$P_{l_2}(f_1, \ldots, f_N | C = c_i)$
$P_{l_{2^N-2}}(f_1, \ldots, f_N | C = c_i)$
$P_{l_{2^N-1}}(f_1, \ldots, f_N | C = c_i)$
c_i
C

[Source: campar.in.tum.de]

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:

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

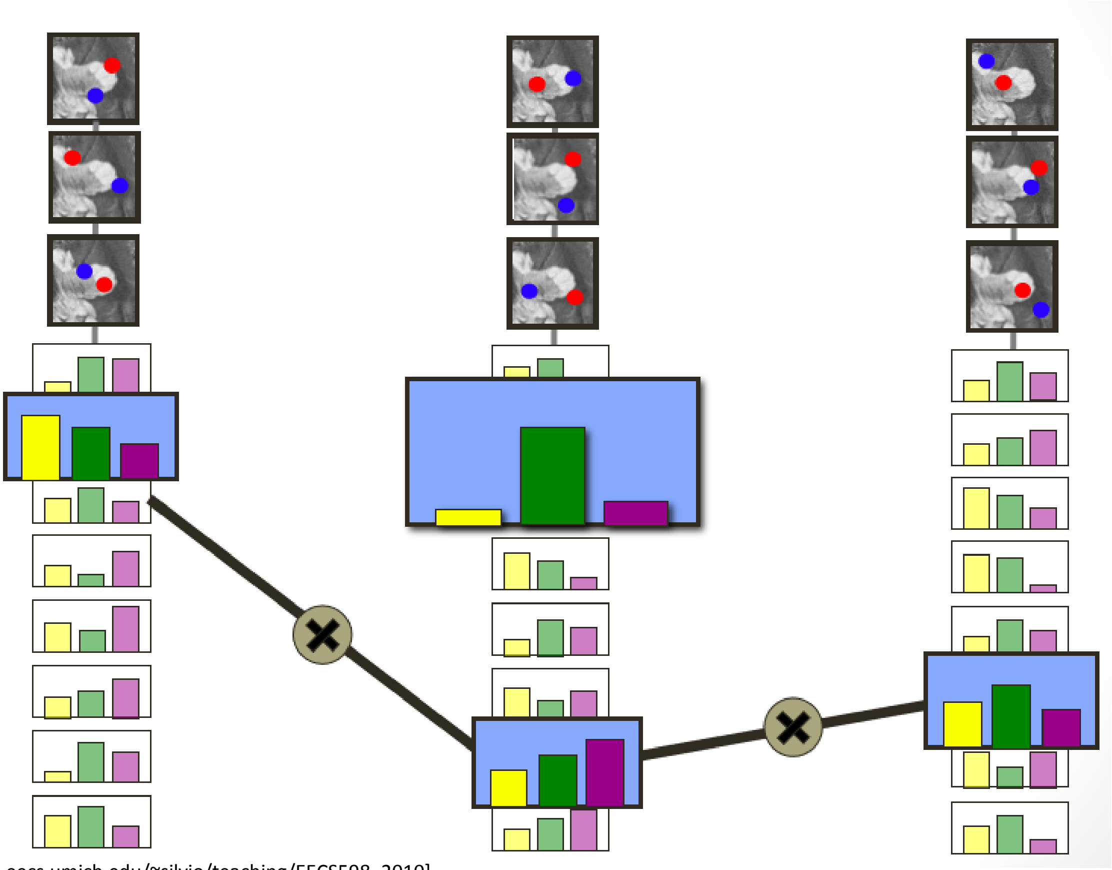
[Source: web.eecs.umich.edu/~silvio/teaching/EECS598_2010]

Implementation: Training

- Find the robust keypoints
  - Find original keypoints
  - Warp image and find keypoints
  - Transform back the warped keypoints
  - Match the original and back warped keypoints

- Train the Ferns
  - Take a patch around each robust keypoint
  - Warp the patch
  - Extract features

Classifier



[Source: web.eecs.umich.edu/~silvio/teaching/EECS598_2010]

Implementation: Classification

- Find keypoints on the test image

- Take patches and extract features

- Calculate probabilities for classes

- Extract highest and apply threshold

Altered Roadmap

Milestone 1: Training (first week)
Keypoint extraction from Training data
Training the Ferns

Actual Milestone 1:
Tried to get comparable code to work (no success there)
Implemented robust keypoint extraction
Trained first few ferns, still buggy probabilities

Altered Roadmap

Milestone 2: Classify (second week)
Keypoint extraction and classification of test data

Actual Milestone 2:

- Finished fern creation
- Added classifier functionality

Altered Roadmap

Milestone 3: Finalizing (last week)
Testing and Tuning
Finding and fixing bugs
Extending

Actual Milestone 3:
Excessive testing
Finding bugs (fixing most?)
Added generic capabilities
Trying to figure out sensible parameters

Demonstration

Questions ?

Thank you
for your attention