Context-driven Probabilistic Object Classification

Object Recognition

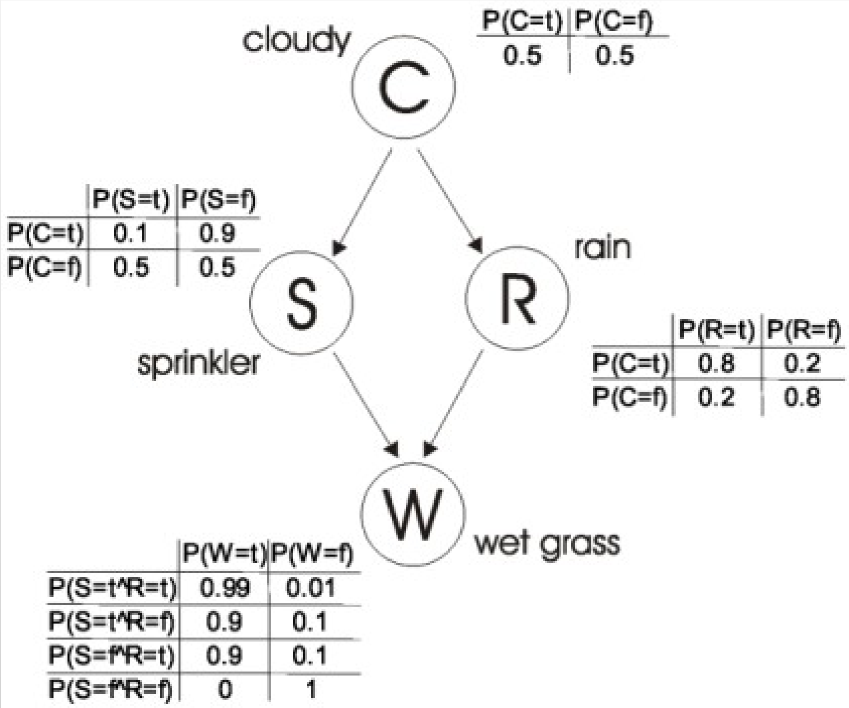
Joseph Cooper

Outline

- General idea
  - Bayesian reasoning
  - Non-uniform photography practices
- Algorithm
- Implementation
  - Details for exploration
- Results

Bayesian Reasoning

- Allows computation with probabilistic relationships between variables
- Information flows in both directions
- Learning the relationships can be quite difficult but it is generally easier than learning and storing the full joint probability table



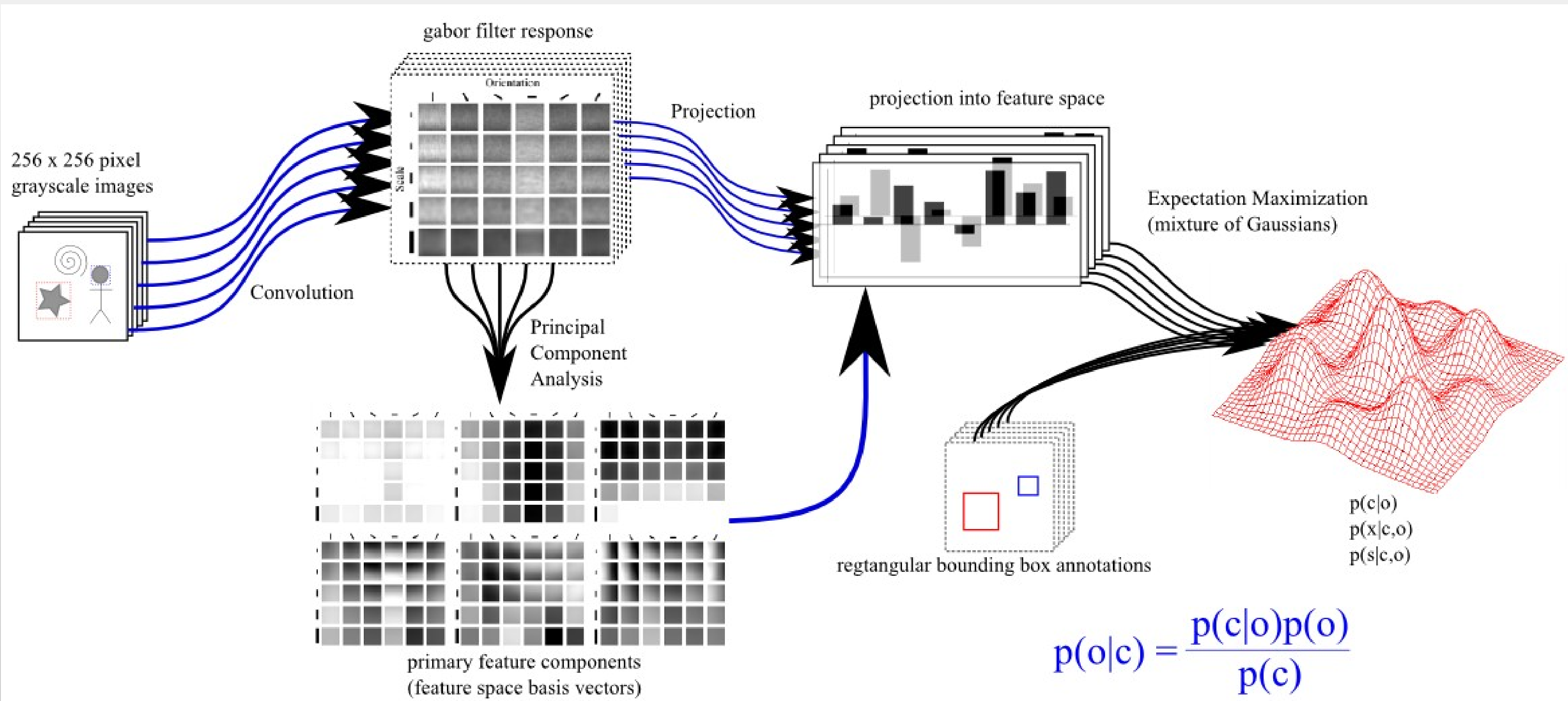
p(c|r,s) = p(c|r,w,s) = 0.444444
p(c|r) = 0.8
p(c|r,w) = 0.793713

Image from http://www.ra.cs.uni-tuebingen.de/software/JCell/tutorial/ch03s03.html

Photographs are not taken in a uniform distribution

- Distributions of foreground and background are related
- Foreground objects are related
- Location of an object in a picture is related to its scale

Algorithm Overview



Capture the ‘gist’ of the image

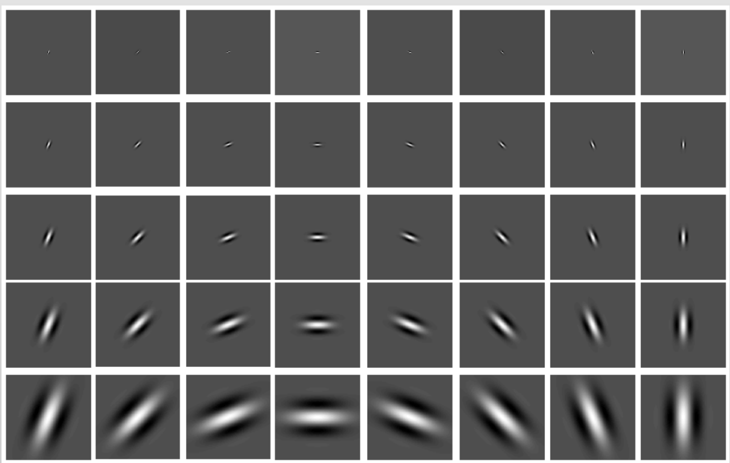
Database

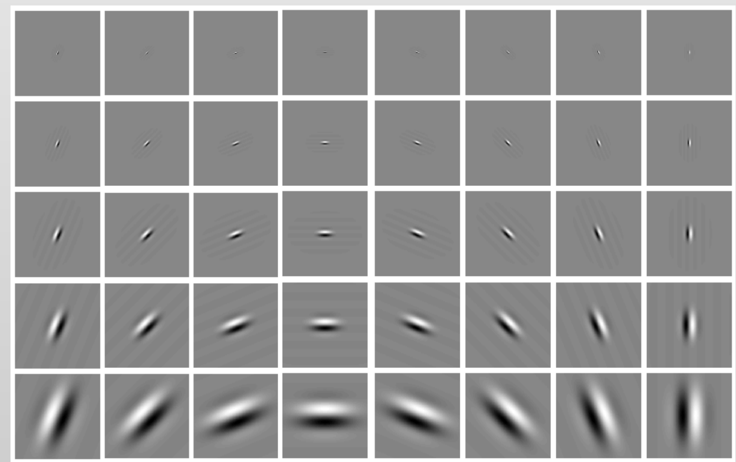
- LabelMe set (transformed to grayscale)
- 2688 fully labeled images
- Test Classes:
  - Person, Boat
  - Tree, Building, Car

Gabor filters

$$g_{\lambda,\theta,\phi,\sigma,\gamma}(x,y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda} + \phi\right)$$

$$x' = x\cos(\theta) + y\sin(\theta)$$
$$y' = -x\sin(\theta) + y\cos(\theta)$$

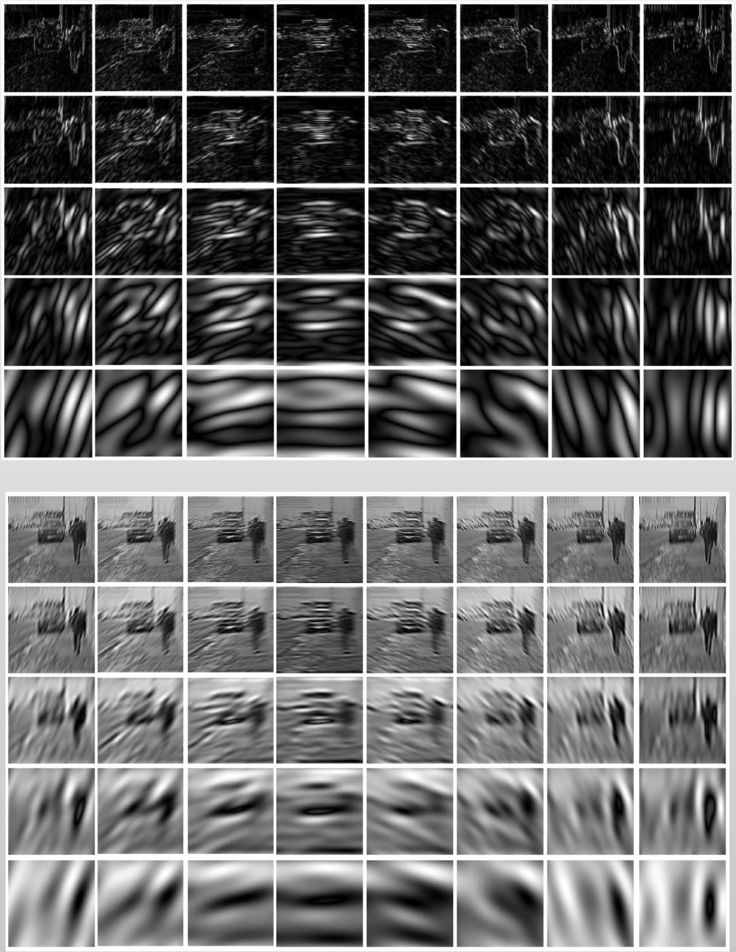




- Multiple scales, orientations, and phases
- Applied in frequency domain

Principal Components Analysis

- Subtract mean filter response across images (for each set of parameters)
- Find k principal eigenvectors

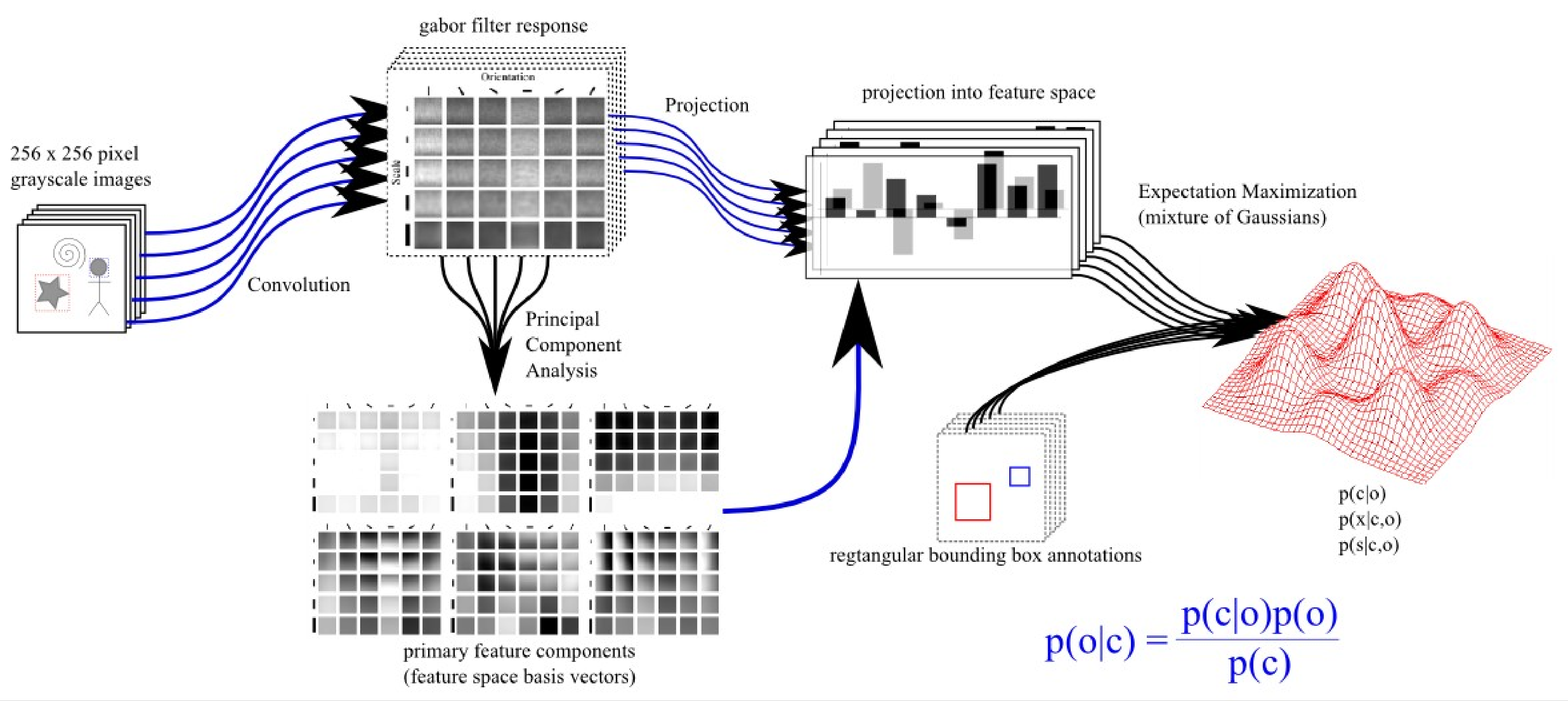


Example set of filter responses

Calculate PDF using EM

- Project filter responses into k-dimensional component space
- Separate class/non-class vectors
- Use EM to find most likely mixture of M Gaussians for
  - p(context|class)
  - p(context|!class)
- Use EM to find most likely locations and scales

Algorithm Overview



Capture the ‘gist’ of the image

Testing

- Apply filter bank
  - To all images
  - To a subset of images
- Project into component space
  - Each filter
  - All filters
- Calculate probability of containing each object

  $$p(object|context) = p(context|object)*p(object)/p(context)$$

- If probability>threshold, calculate probable locations and scales

Research Questions

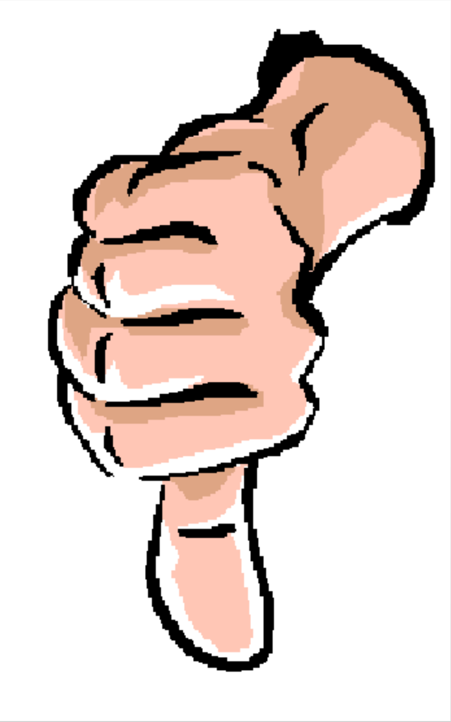
- Which Gabor filters (s,Θ,Φ)?
- How many components (k)?
- How many Gaussians (m)?
- How can you avoid a fixed-size requirement?
- How do you find enough memory?
- Can you make it iterative so that you do not need all images up front?

Experimental results

- Varied scales
- Varied orientations
- Varied phase

```
Out of memory. Type HELP MEMORY for your options.
```

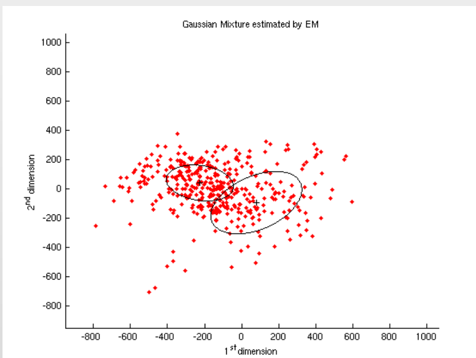
- Third try's a charm

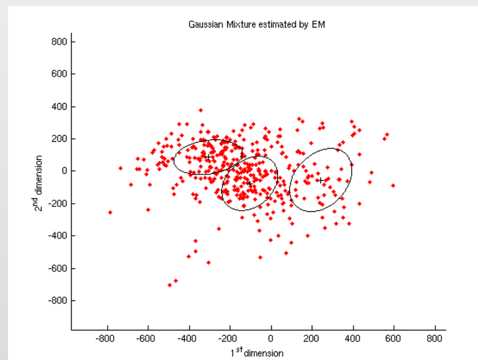


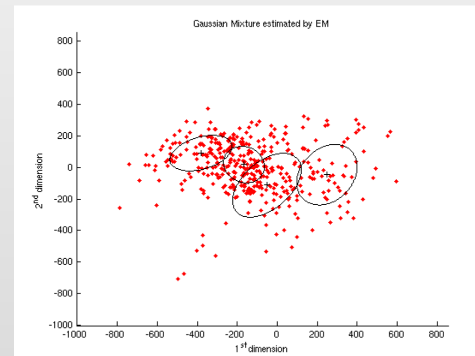
Experimental results

- Varied number of gaussians

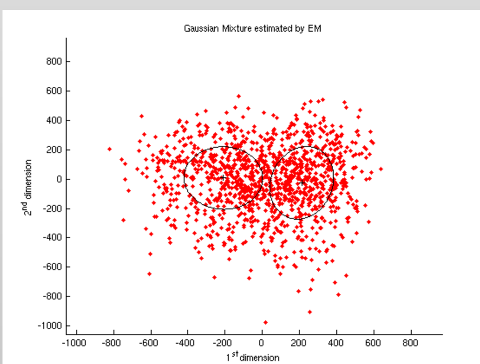
In

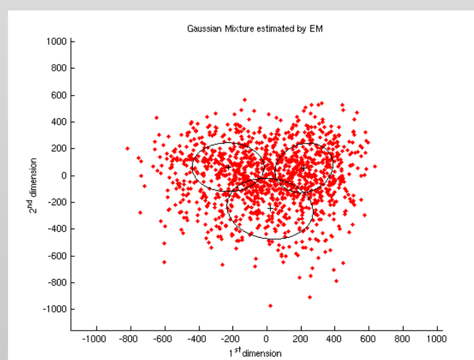


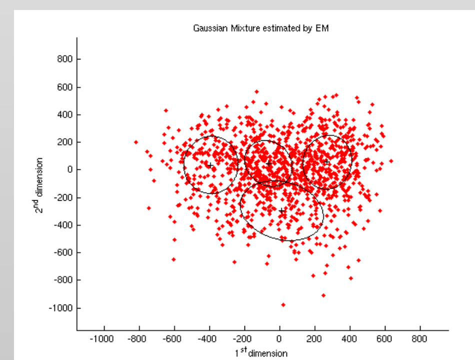




Out

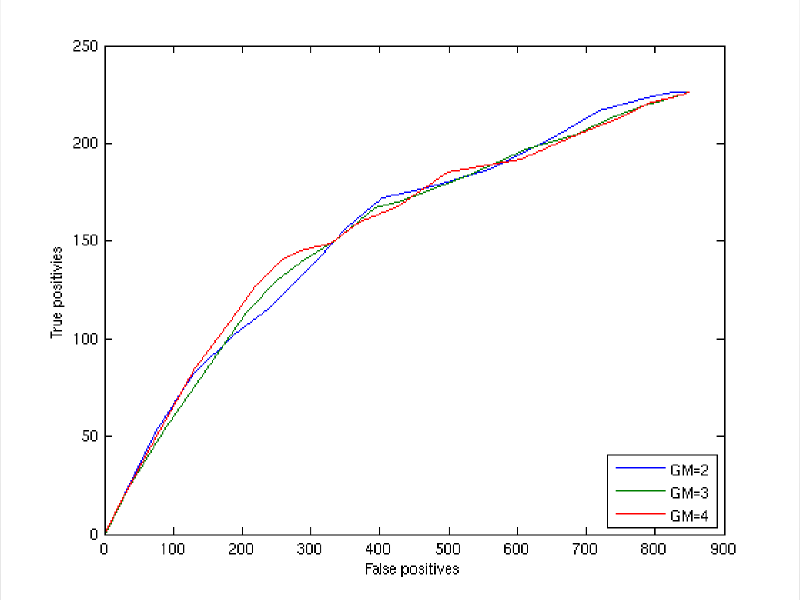




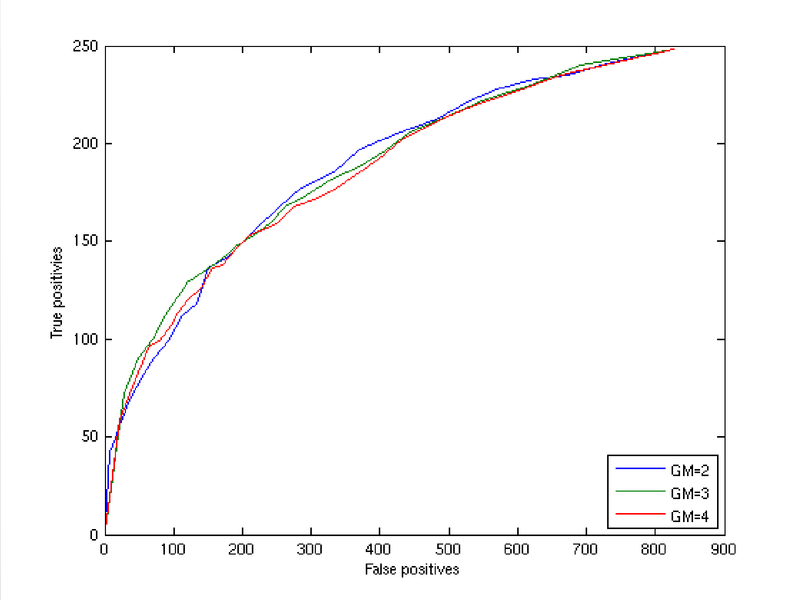


Experimental Results

➢ Different numbers of components



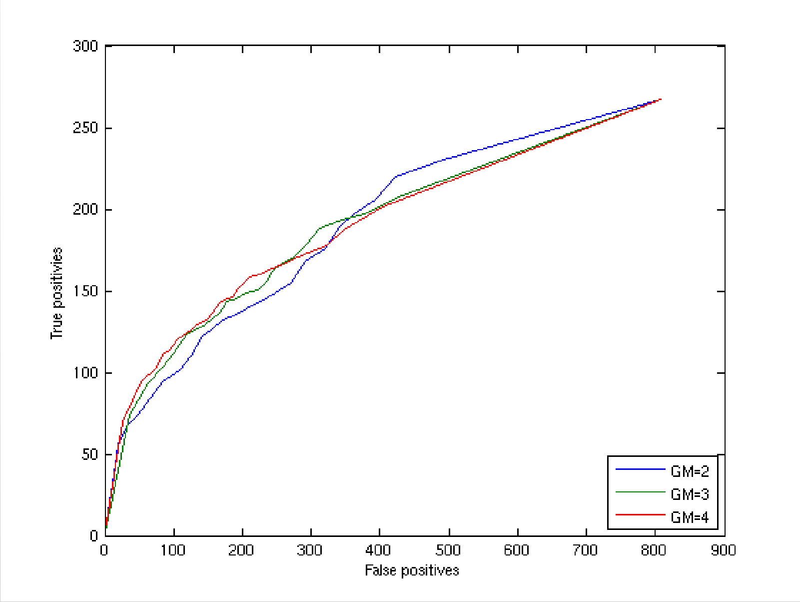
Cars: 2 components



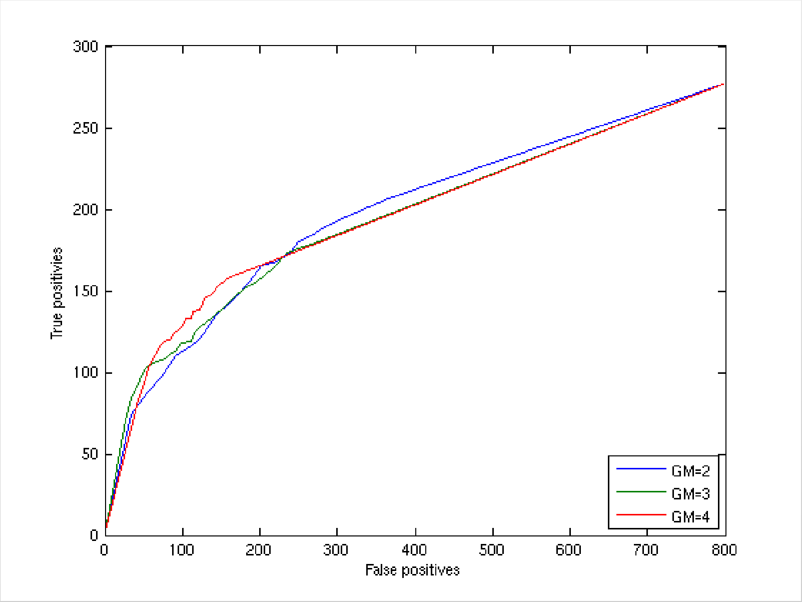
Cars: 10 components

Experimental Results

- Different numbers of components



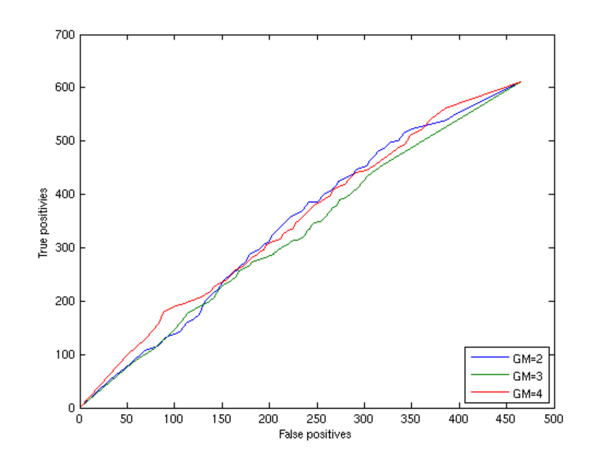
Cars: 20 components



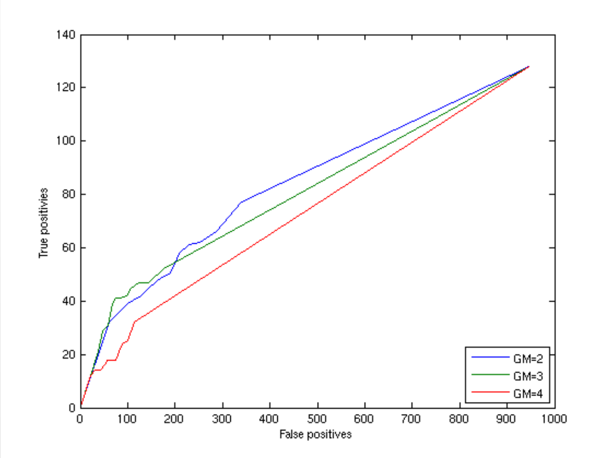
Cars: 30 components

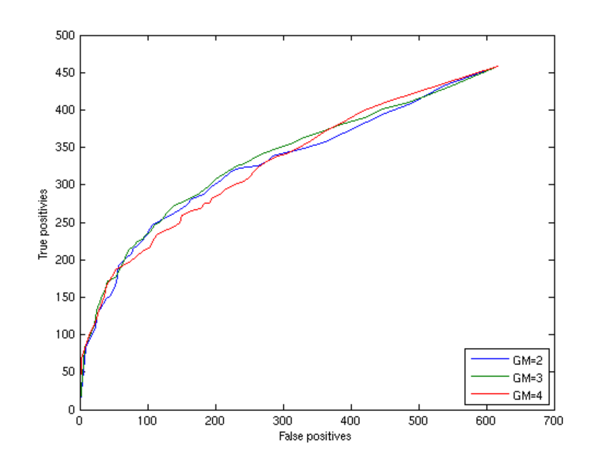
More results

Tree: 30

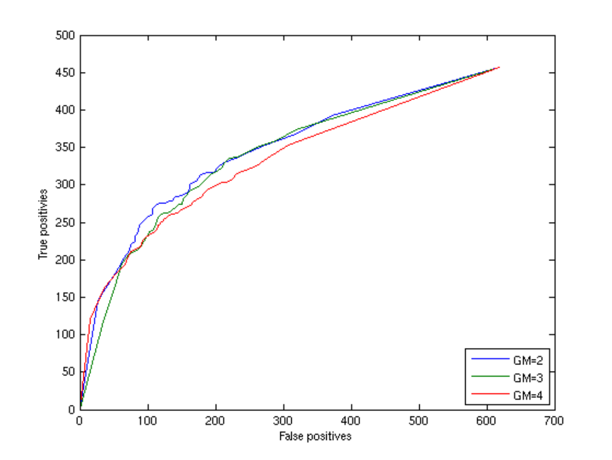


Person: 30





Building: 15



Building: 30

Conclusion

- Needs work
  - Iterative method
  - Lower memory requirements
  - Discover new components as needed
- Higher components may have more discriminating features
- Additional Gaussians do not seem to add much

Future Work

- Variable image size
  - Shifting window
  - Combine with other feature detectors
- Learn additional probabilistic relationships
- Iterative change
- Try tweaking filters again