

# **Automatic Visual Recognition of Biological Particles**

Marc'Aurelio Ranzato

Pietro Perona

*California Institute of Technology*

Ruggero Frezza

*Universita' degli studi di Padova*

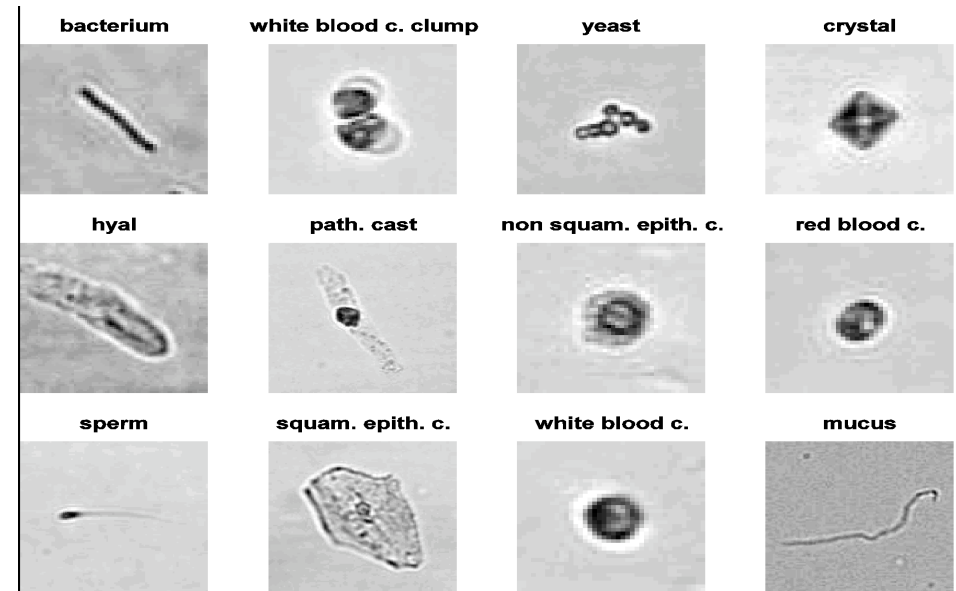
# Agenda

- The goal
- Microscopic analysis: state of the art
- Database
- Detector
- Features
- Classifier
- Experiments
- Conclusions

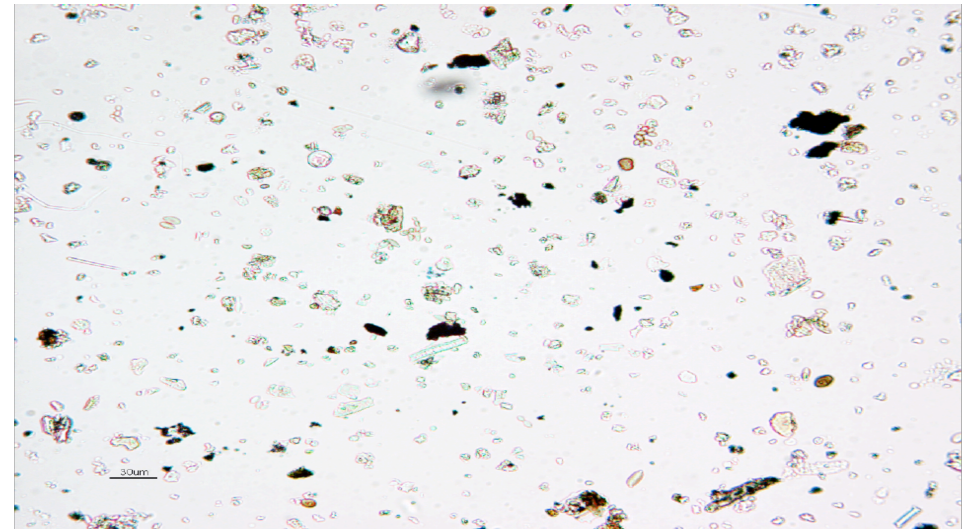


# THE GOAL

➤ RECOGNITION OF PARTICLES IN URINE



➤ RECOGNITION OF AIRBORNE POLLEN



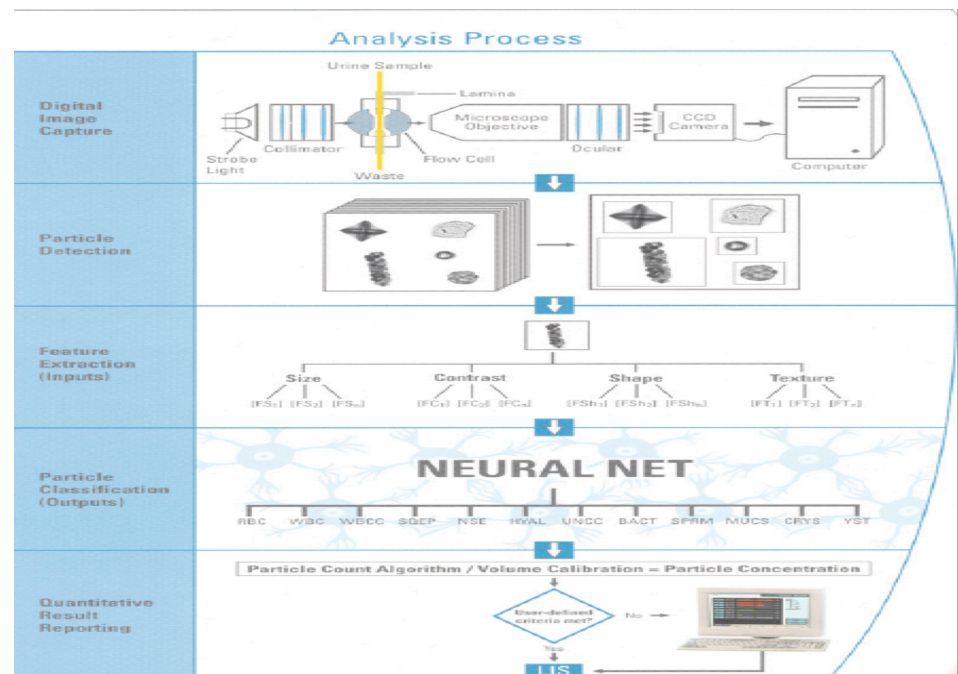
# Microscopic analysis: urine

## *Manual*

- Slow
- Inaccurate
- Expensive

## *Automatic*

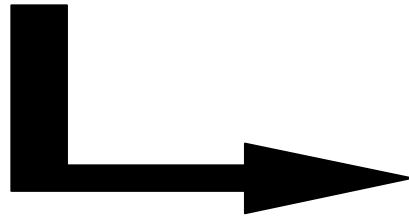
- Performance
- Reliability



# Microscopic analysis: pollen



- Slow
- Expensive
- Inaccurate for low pollen levels
- Not able to cover the territory
- No standardization



# Related work

- *Urine*

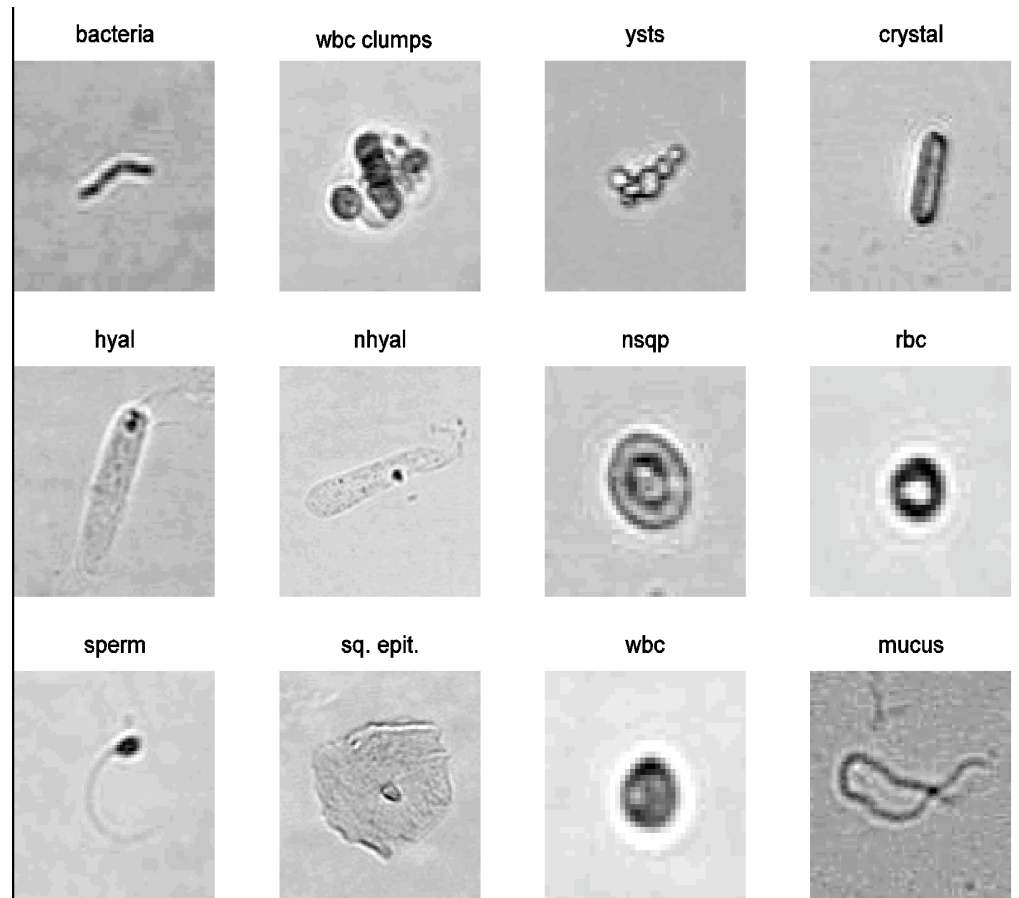
1. IRIS iQ200
  - Automatic visual recognition
2. Dahmen et al. (2000)
  - Classification of red blood cells

- *Pollen*

1. Ronneberger (2000)
  - SEM
  - 3D analysis
2. Duller et al. (1997)
  - Optical microscope images: identification
  - Pollen with double edge

# Database

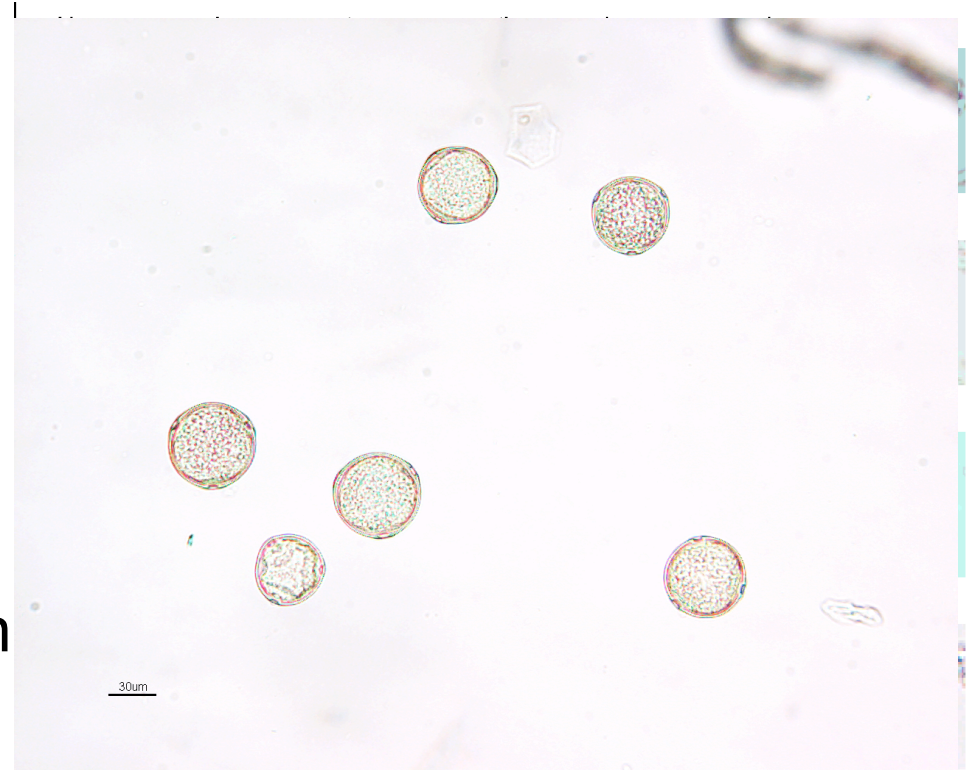
- *URINE*
  - No detection is needed
  - 12 categories
  - 1000 images in 10 categories, 860 and 546 in other two



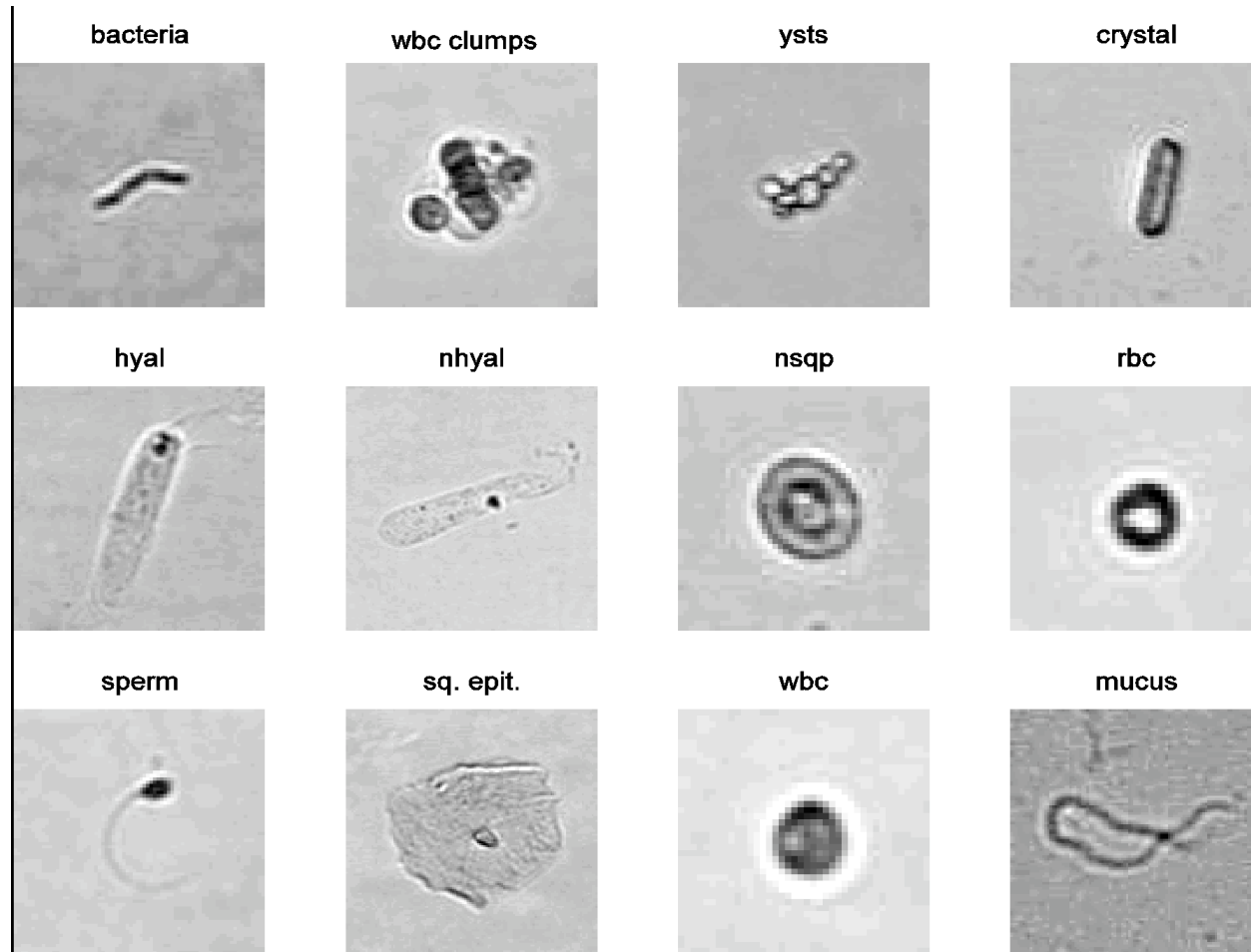


# Database

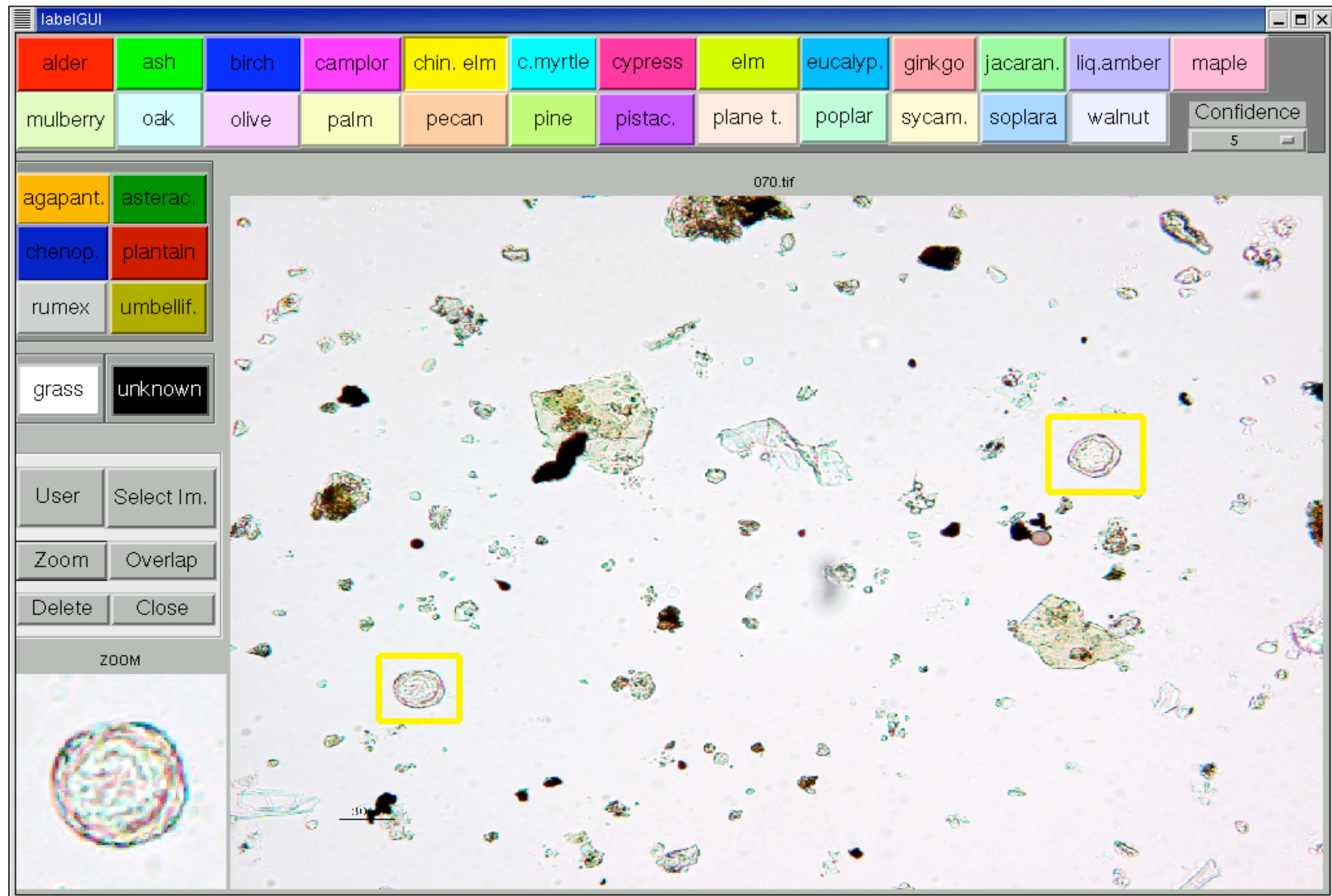
- *POLLEN*
  - 299 sticky tape images
  - 27 categories
  - 778 images with “pure” pollen
  - Two derived data bases of patches
  - 6 “pure” pollen categ. with more than 1000 images
  - 3 air sampled cat. with more than 50 images



# Examples: urine



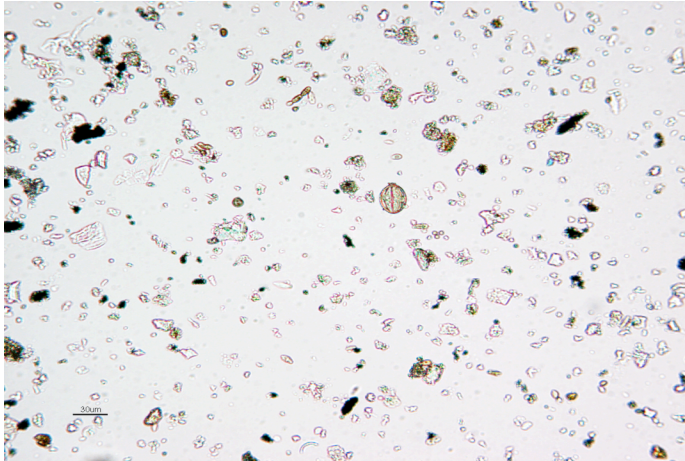
# Labeling of pollen



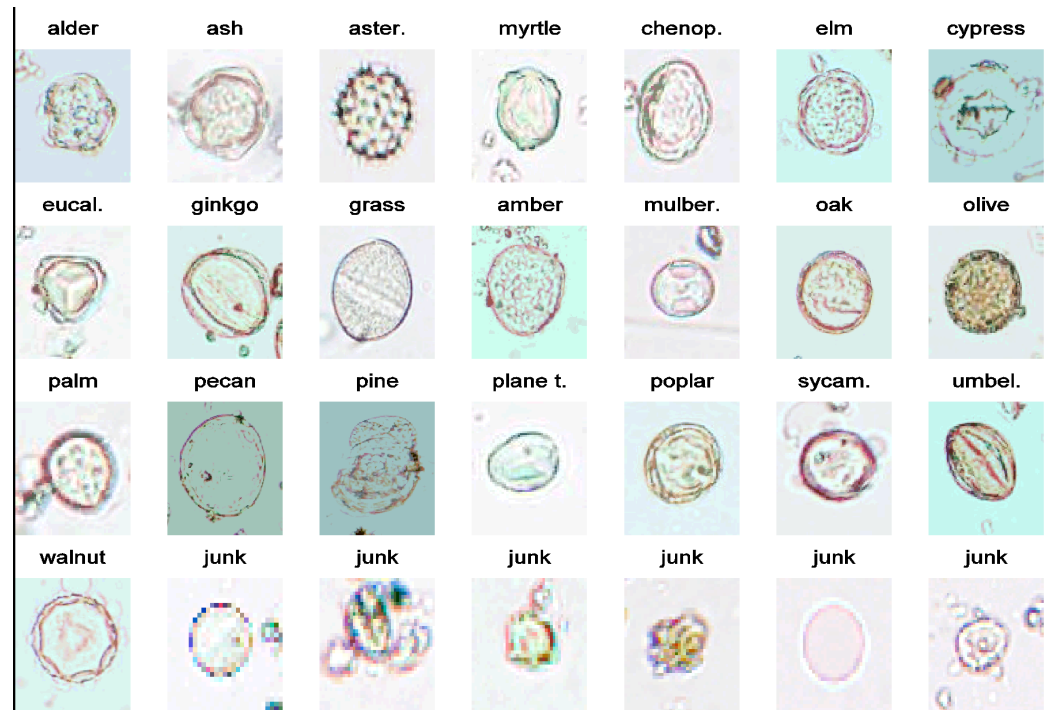
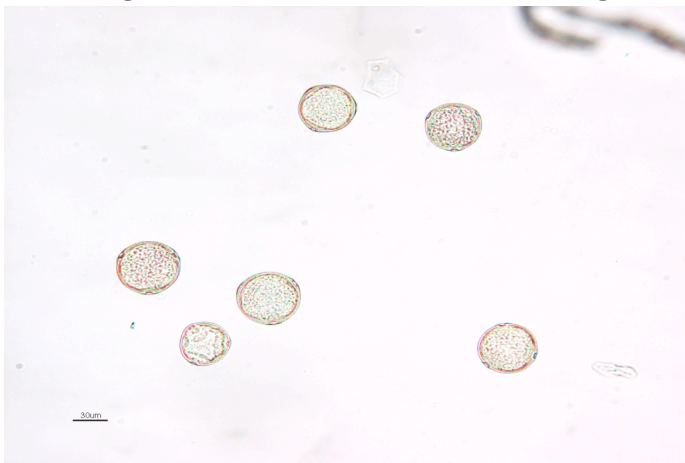


# Examples: pollen

Sticky tape image



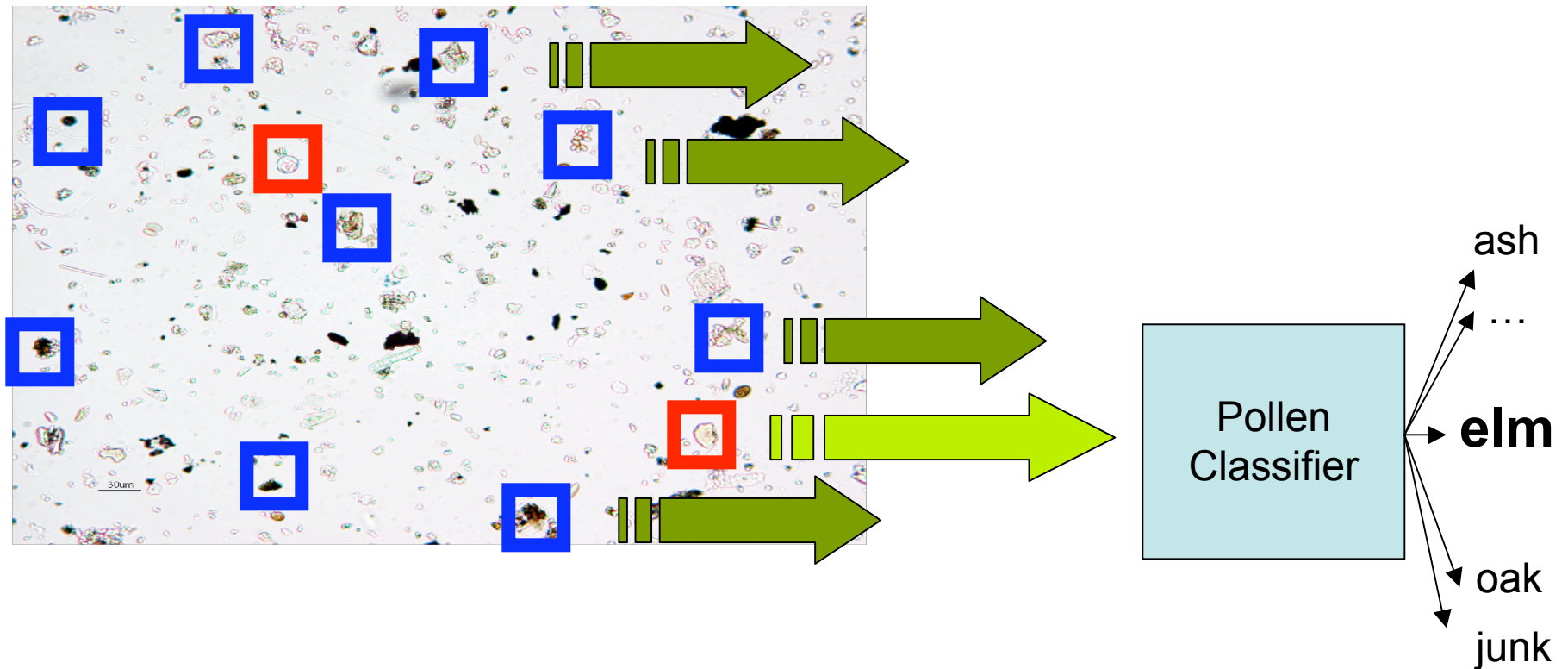
Original “pure” pollen image



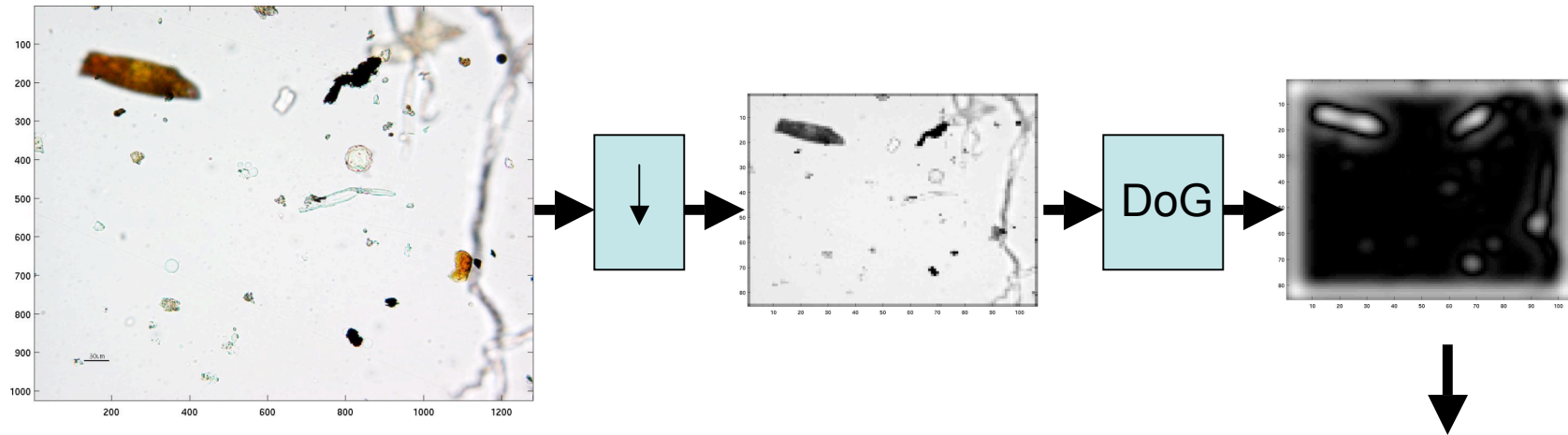
# Recognition System

*DETECTION*

*CLASSIFICATION*

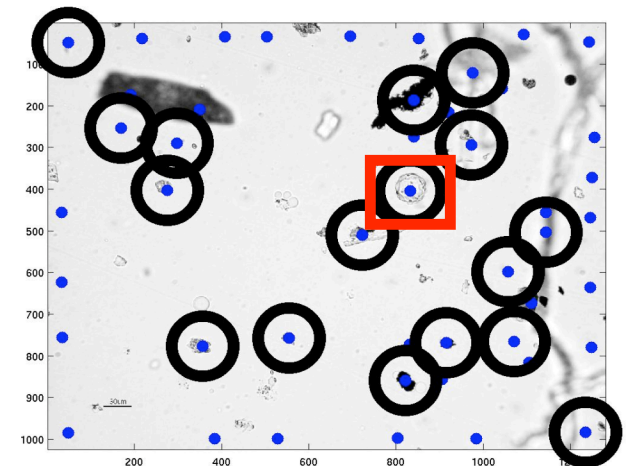


# Detector: based on DoG

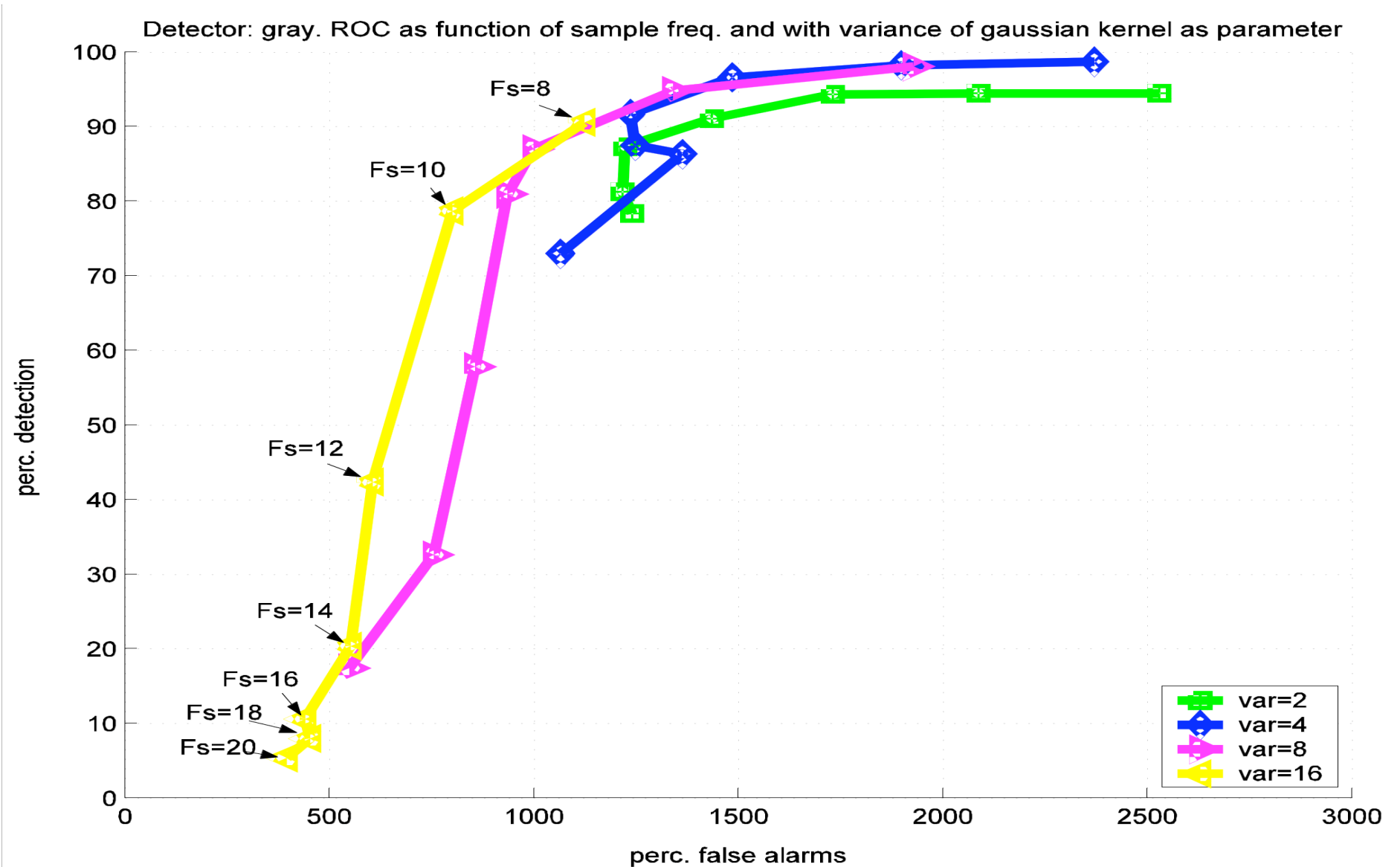


- Patches are squares with side 200 pixels
- To establish detection:

$$\sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2} \leq \alpha \sqrt{\frac{A_1 + A_2}{2}}$$



# DoG performance



# DoG detector

## PROS

- General approach
- Fast: 0.5 sec/img.

## CONS

- Box adaptation
- Performance  
( $F_s, \sigma, l$ ) to be varied

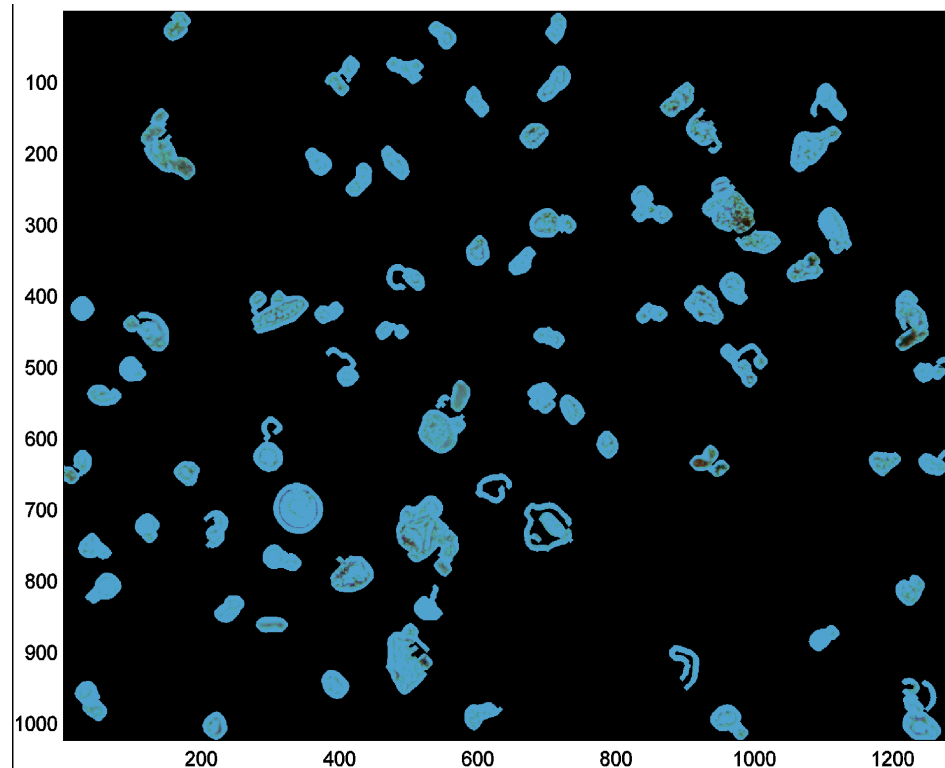
# Detector based on morphological operations

Based on:

- edge detection
- dilation
- erosion
- closing
- opening
- region labeling

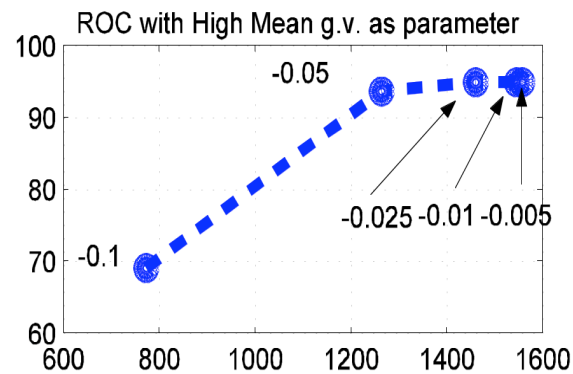
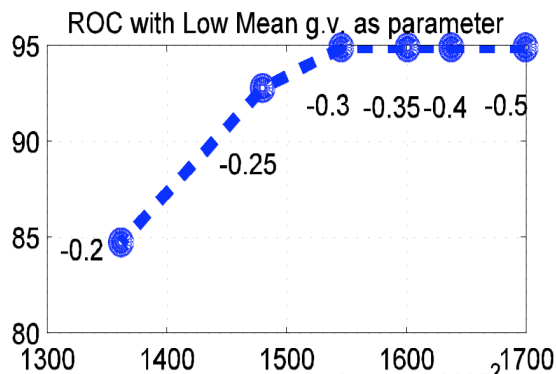
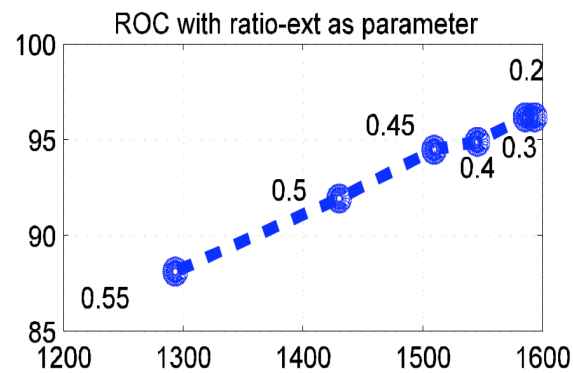
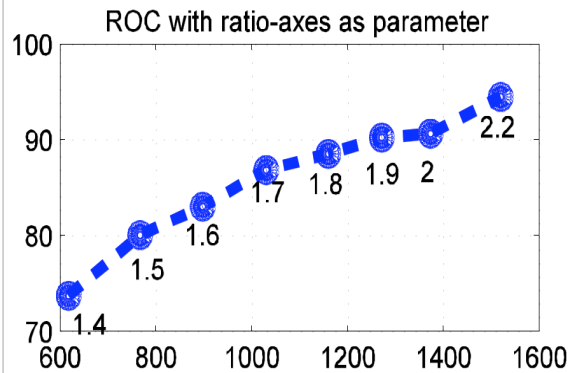
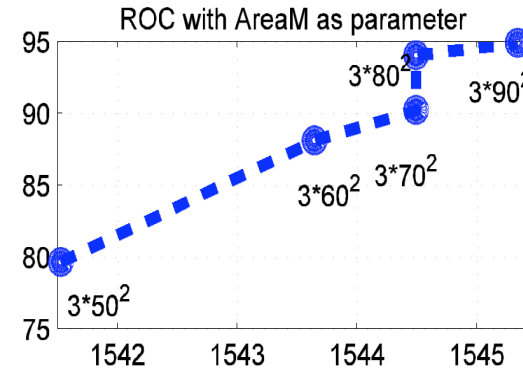
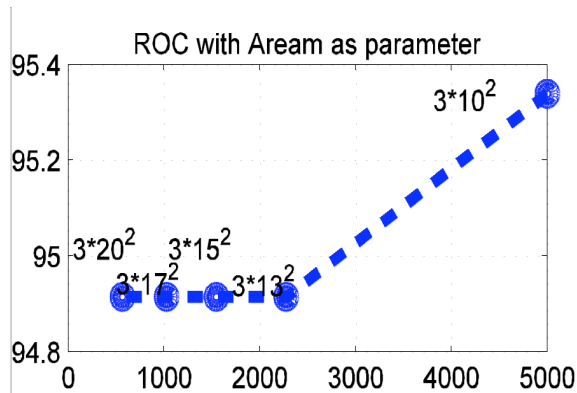
Test on:

- shape
- area
- average brightness





# Morph. detector: performance



DEFAULT VALUES:  $\text{AreaM} = 3 \times 100^2$   $\text{R\_Ext} = 0.4$   
 $\text{Low Mean} = -0.3$   
 $\text{R Axes} = 2.25$   $\text{High Mean} = -0.01$

|                                   |                              |                               |
|-----------------------------------|------------------------------|-------------------------------|
|                                   | <b>71 img.<br/>236 poll.</b> | <b>299 img.<br/>562 poll.</b> |
| <i>perc.<br/>detection</i>        | 93%                          | 87%                           |
| <i>perc.<br/>false<br/>alarms</i> | 956%                         | 1400%                         |

# Morphological detector

## PROS

- Performance
- Good segmentation

## CONS

- High customization (reprogramming?)
- Slowness: 15 sec./img.

# Classification

- Key point is *feature extraction*

“If the features capture the peculiarities of appearance of our particle, then any classifier will perform a competent job”

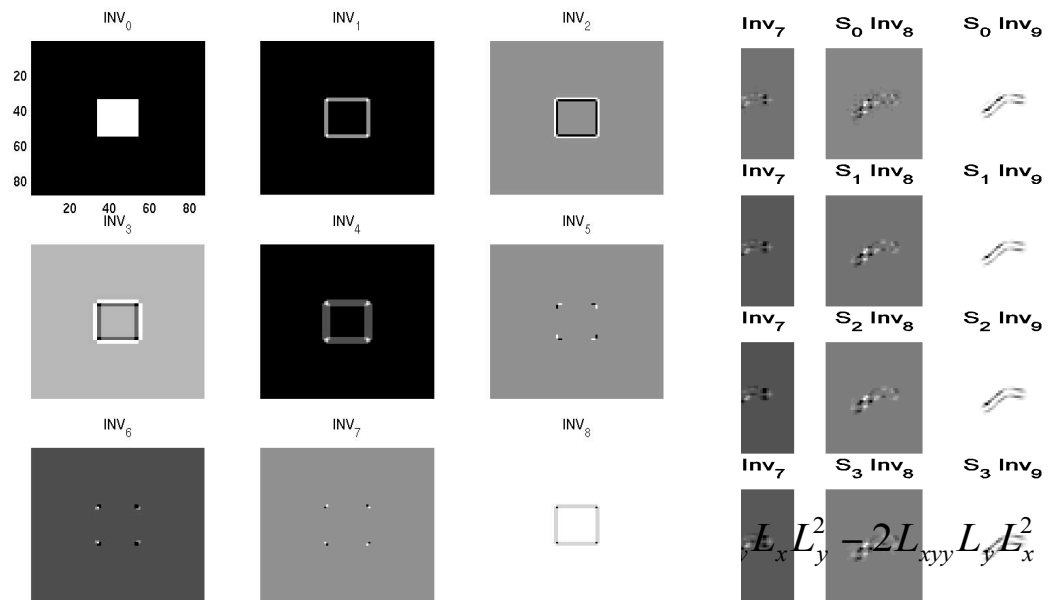
(Duda & Hart, 1973)

- Two kinds of features have been developed  
Based on:
  - local jets (Schmid et al. 1997),
  - image and power spectrum principal components (Torralba et al. 2003)

# Features based on local jets (1)

- **Local jet**: convolution of input image with derivatives of Gaussian kernel
- **Invariant**: linear combination of local jets can be invariant with respect to
  - shift
  - rotation
- **Multi-scale** approach

9 invariants, 4 scales ➡ 36 planes are derived from each image



# Features based on local jets (2)

- Application of nonlinearities

Each plane is divided in:

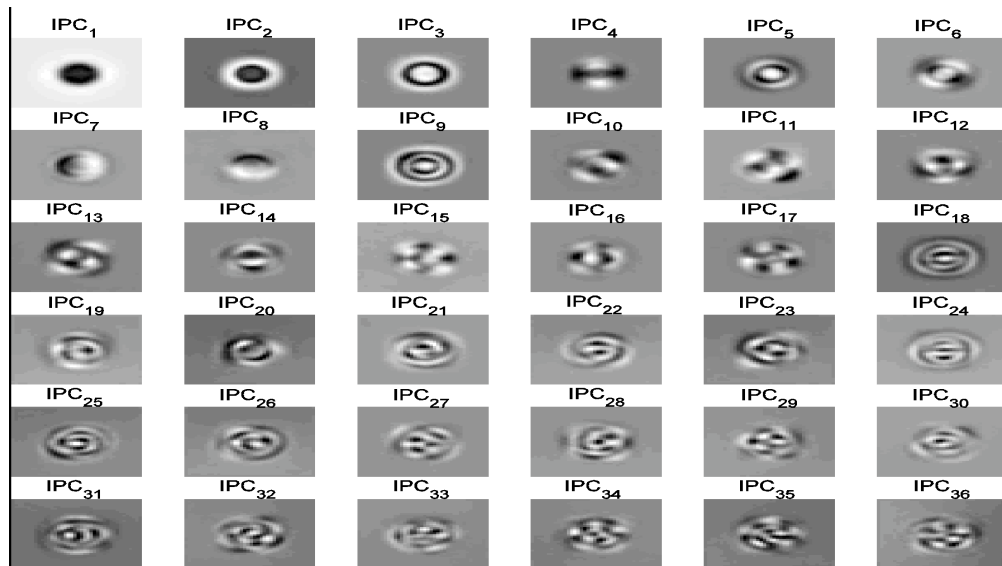
- Positive part
- Negative part
- Absolute value

- Average

- **The feature vector has  $9 \times 4 \times 3 = 108$  components**  
FLD (or PCA) is required

# Features based on image principal components (1a)

- Collect training images rearranged in a column vector
- Let be  $\Sigma$  the estimated covariance matrix
- We call *image principal components* *IPC* the eigenvectors of  $\Sigma$





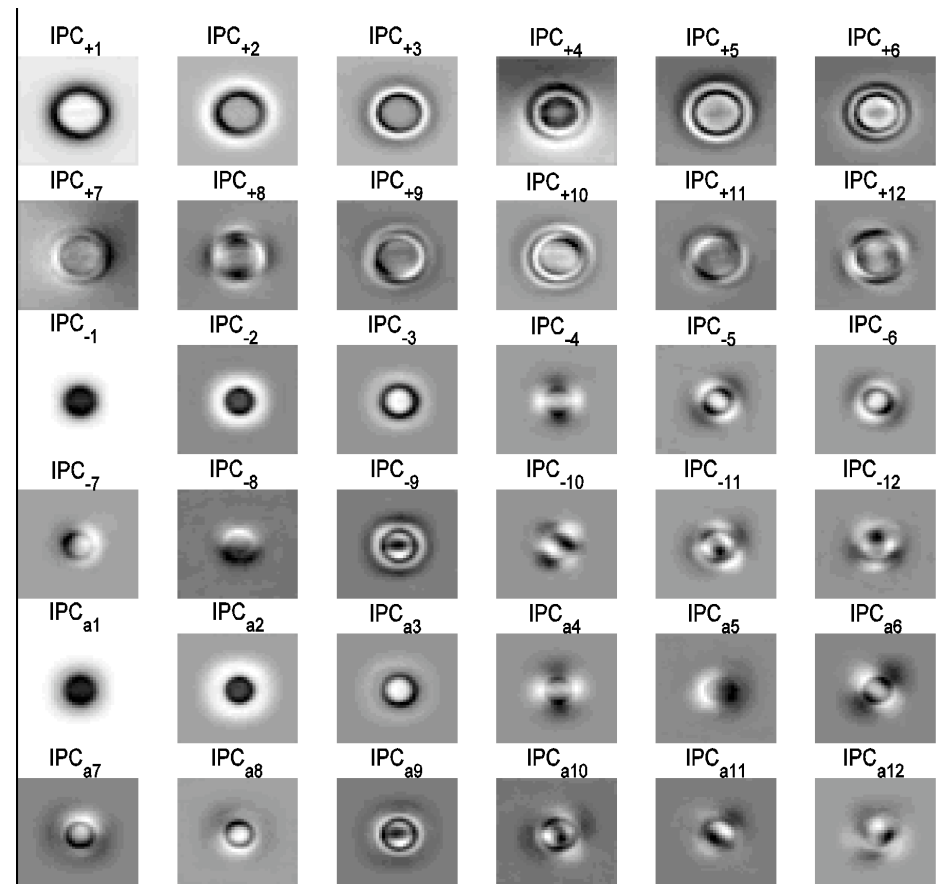
# Features based on image principal components (1b)

- Application of nonlinearities

Image is divided in:

- positive part
- negative part
- absolute value

Then, PCA is applied

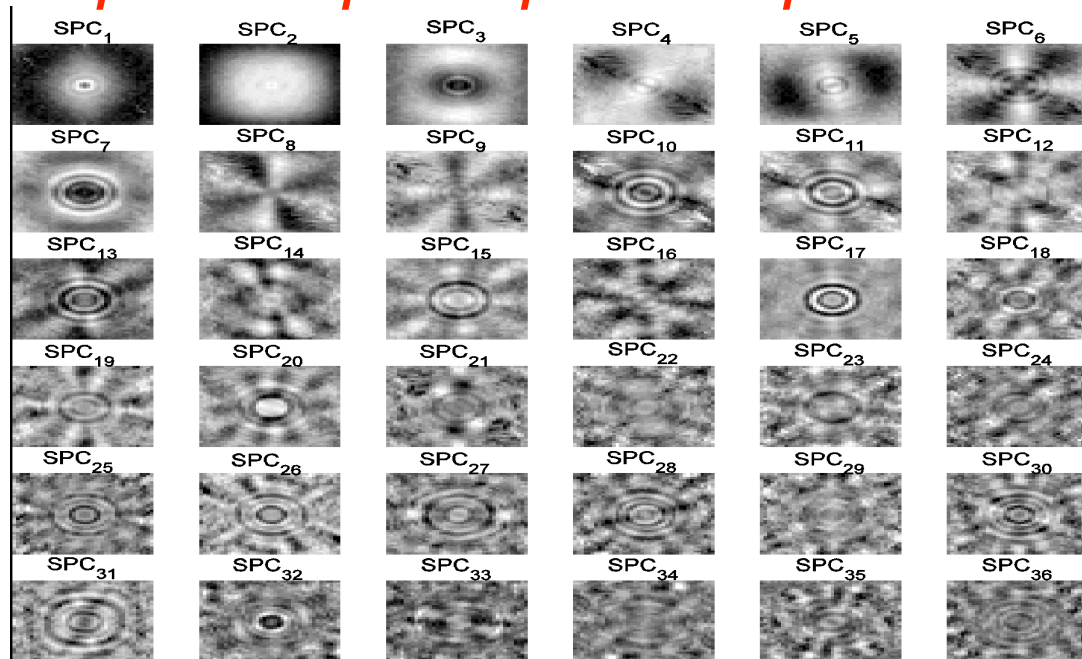


# Features based on power spectrum principal components

Compute:

- DFT  $I(kx, ky)$
- Normalize power spectrum  $R^* = R / std(R)$
- Apply PCA in order to find

*spectral principal components SPC*



# Classification

TRAINING → TEST

- Modeling of training data with MoG
  1. features based on local jets
  2. features based on princ. components
  3. combination of features
- Boosting
- SVM

# Classification: MoG (1)

- Goal:  $P[C_k | x]$ , in order to apply the decision rule

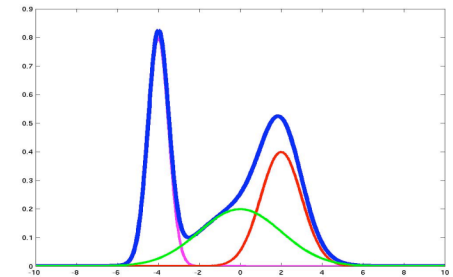
$$x \mapsto r(x) = \arg \max_k P[C_k | x]$$

- Application of Bayes' rule:

$$P[C_k | x] = \frac{f(x | C_k) P[C_k]}{f(x)}$$

- We model the *class-conditional* pdf with a mixture of Gaussians

$$f(x | C_k) = \sum_{j=1}^M f(x | j, C_k) P[j]$$



# Classification: MoG (2)

- Combination of features

- 1) Independence hypothesis

$$f(x_1, x_2 | C_i) = f(x_1 | C_i) f(x_2 | C_i)$$

$$P[C_i | x_1, x_2] \propto f(x_1 | C_i) f(x_2 | C_i)$$

- 2) No hypothesis

- a) two separate trainings

- b) model vector of probability with a MoG,

$$(P[C_1|x, \theta_{pca}], \dots, P[C_1|x, \theta_{lj}], \dots)$$

- c) in test, compute the maximum likelihood

# Classification: AdaBoost

ALGORITHM AdaBoost.M1 customized version

**Input:**

- ◆ sequence of  $N = M \cdot K$  examples  $\langle (x_1, y_1), \dots, (x_N, y_N) \rangle$  with labels  $y_i \in Y = \{1, \dots, K\}$
- ◆ distribution  $D$  over the examples
- ◆ classifier
- ◆ integer  $T$  specifying the number of iterations
- ◆ learning ratio  $\lambda$

**Initialize** the weight vector:  $w_k^1 = D(k) = 1/M$  for  $k = 1, \dots, K$ .

**DO for**  $t = 1, 2, \dots, T$

1. Call classifier, providing it with  $w_k^t$  images for each class; get back the hypothesis  $h_k : X \rightarrow Y$ .
2. Calculate the error overall error rate  $e_k$ .
3. Set  $\beta_k = \frac{e_k}{1 - e_k}$ .
4. Let be  $ER_k$  the error rate for class  $k$  and  $MER = \arg \max_{k \in Y} ER_k$ . Set the new weight vector to be

$$w_k^{t+1} = w_k^t \beta_k^{1 - ER_k / MER}.$$

Let be  $MW = \arg \max_{k \in Y} w_k^{t+1}$  and  $mw = \arg \min_{k \in Y} w_k^{t+1}$ , then rescale the weights between  $\lambda \times mw$  and  $MW$ .

**Output** the hypothesis

$$h_f(x) = \arg \max_{y \in Y} \sum_{k=1}^T \left( \log \frac{1}{\beta_k} \right) \vdash h_k(x_i) = y_i \vdash$$

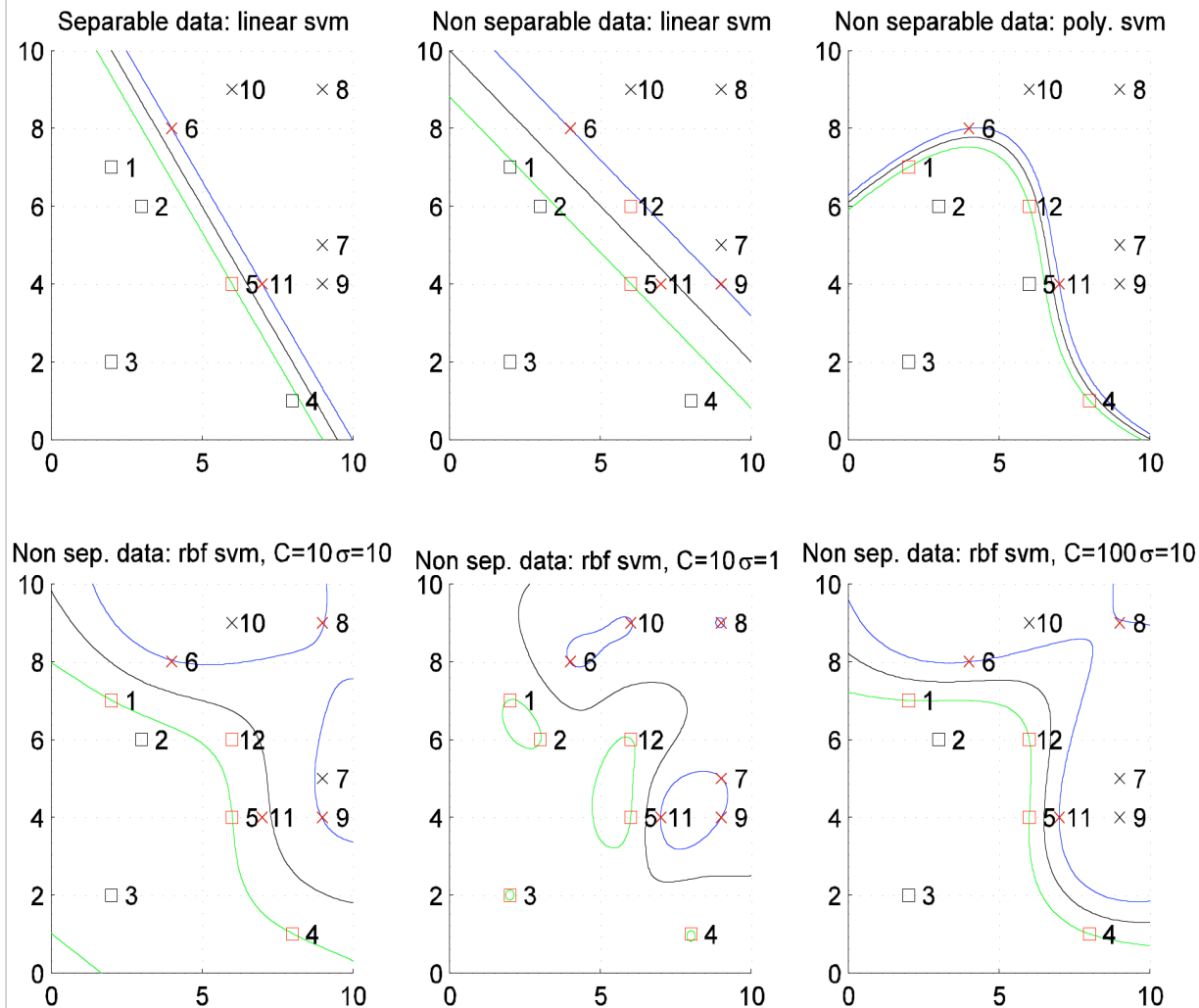
# Classification: AdaBoost

- Algorithm derived from AdaBoost.M1
- At each iteration a hypothesis is made
- The final hypothesis combines the previous ones
- At each round the best classified category has to train on less images
- At each round the worst classified category has to train on the max allowed nr. of images



# Classification: SVM (1)

*“Don’t model the distribution of features but work directly on them!”*



Best performance achieved using RBF kernel

Parameters to tune:

- C, penalty on errors
- $\sigma$ , width of Gaussian kernel
- number of feature space dimensions (FLD)

# Classification: SVM (2)

# How to deal with multi-class classification task?



# Error-correcting output codes technique

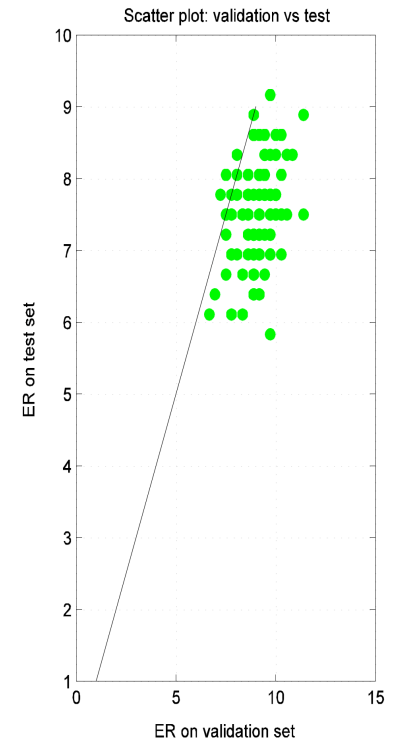
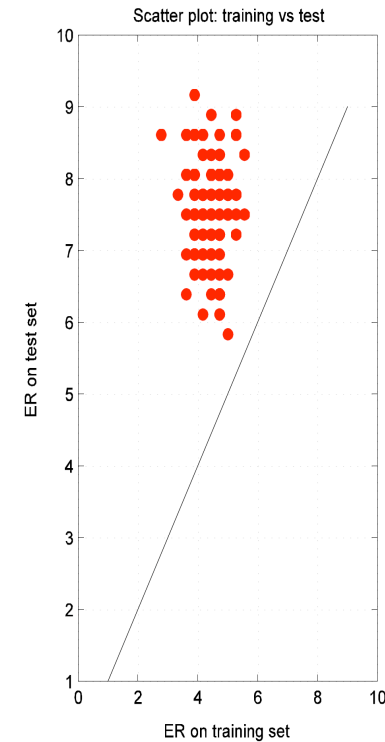
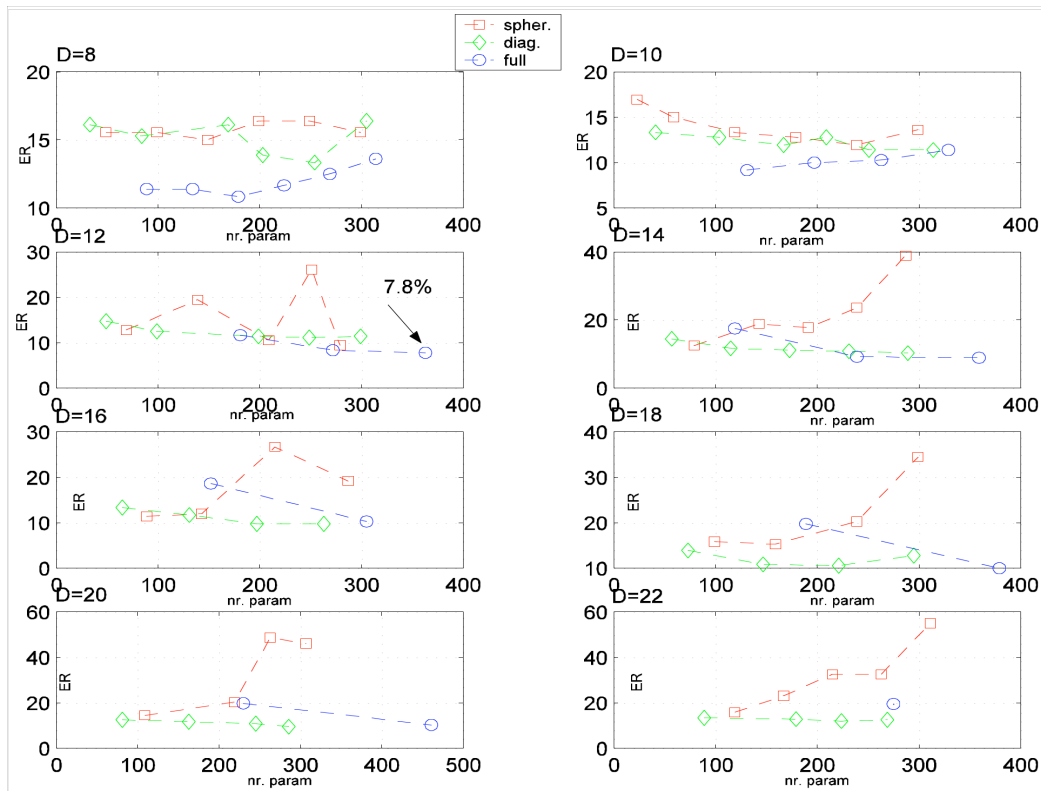
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |   |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |   |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |   |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |   |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |   |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |   |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |   |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |   |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |   |

# Experiments: tuning (1)

- MoG
  - Feature space dimensions
  - Number of components in each mixture
  - Kind of covariance matrix
  - Initial conditions for EM algorithm
- AdaBoost
  - Number of iterations
- SVM
  - Feature space dimensions
  - Kernel
  - Degree polynomial or width RBF kernel
  - Parameter C

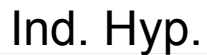
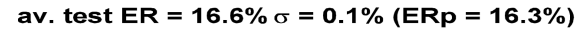
# Experiments: tuning (2)

Example: classifier using MoG and features based on local jets

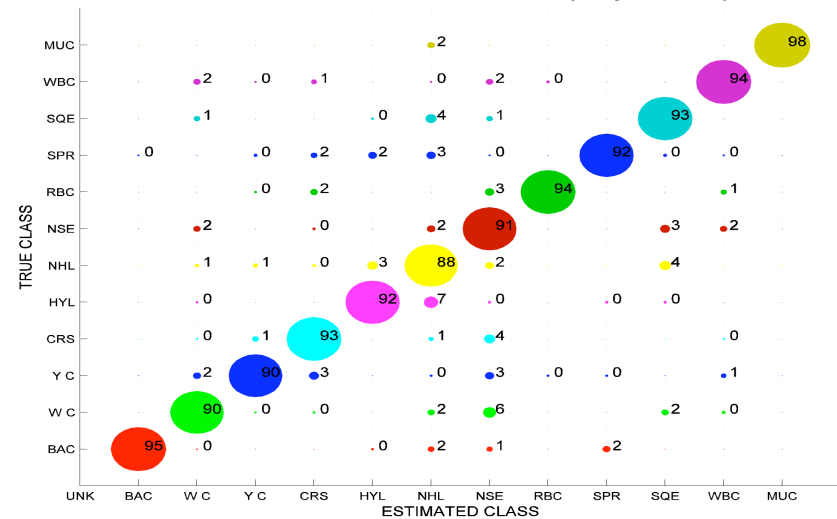
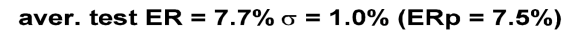


## Loc. jets

# IPC SPC



# Comb. Class.



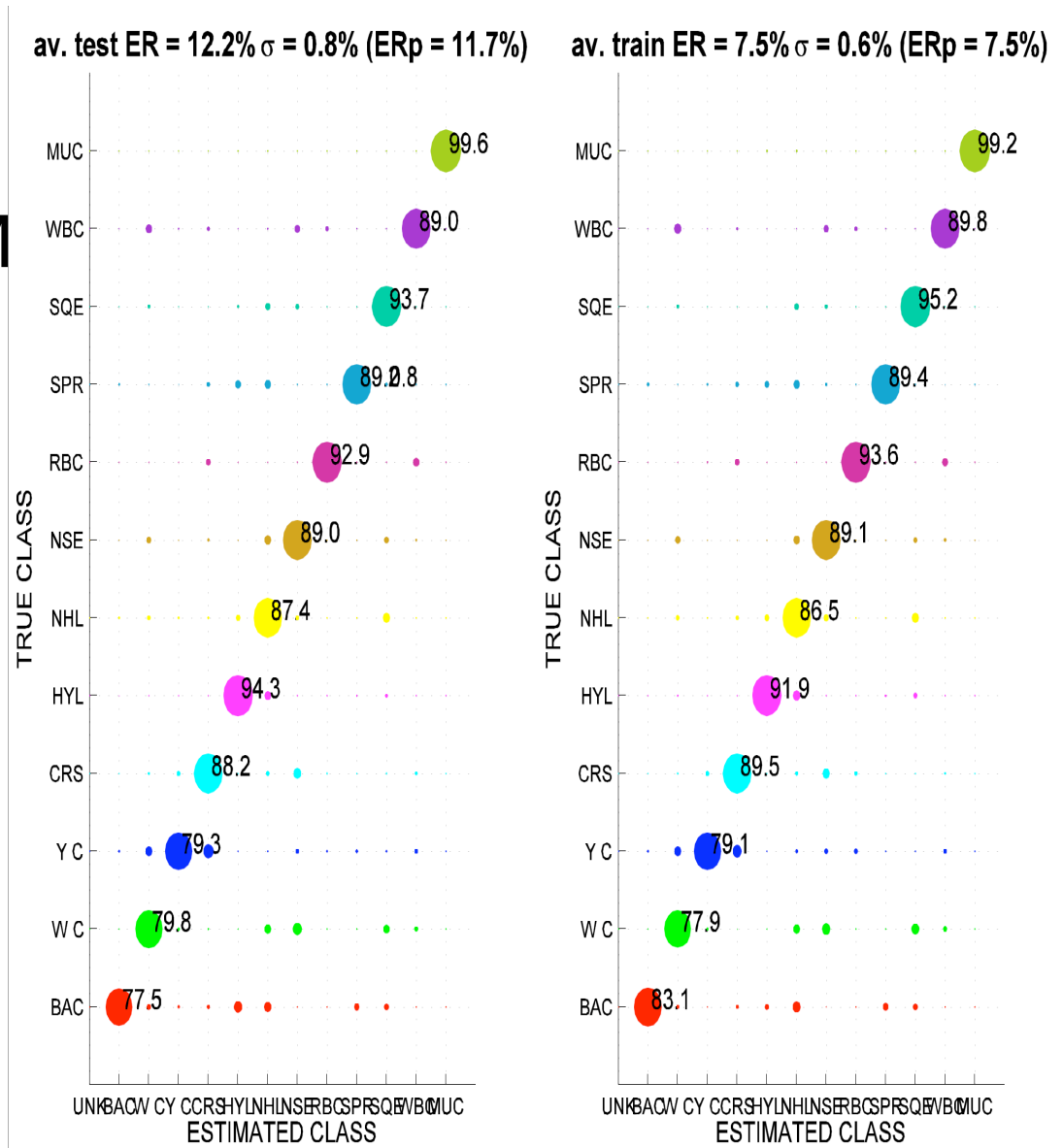
# Experiments: urine (2a)

- Comparison

**MoG – AdaBoost – SVM**

**MoG**

Features based on local jets



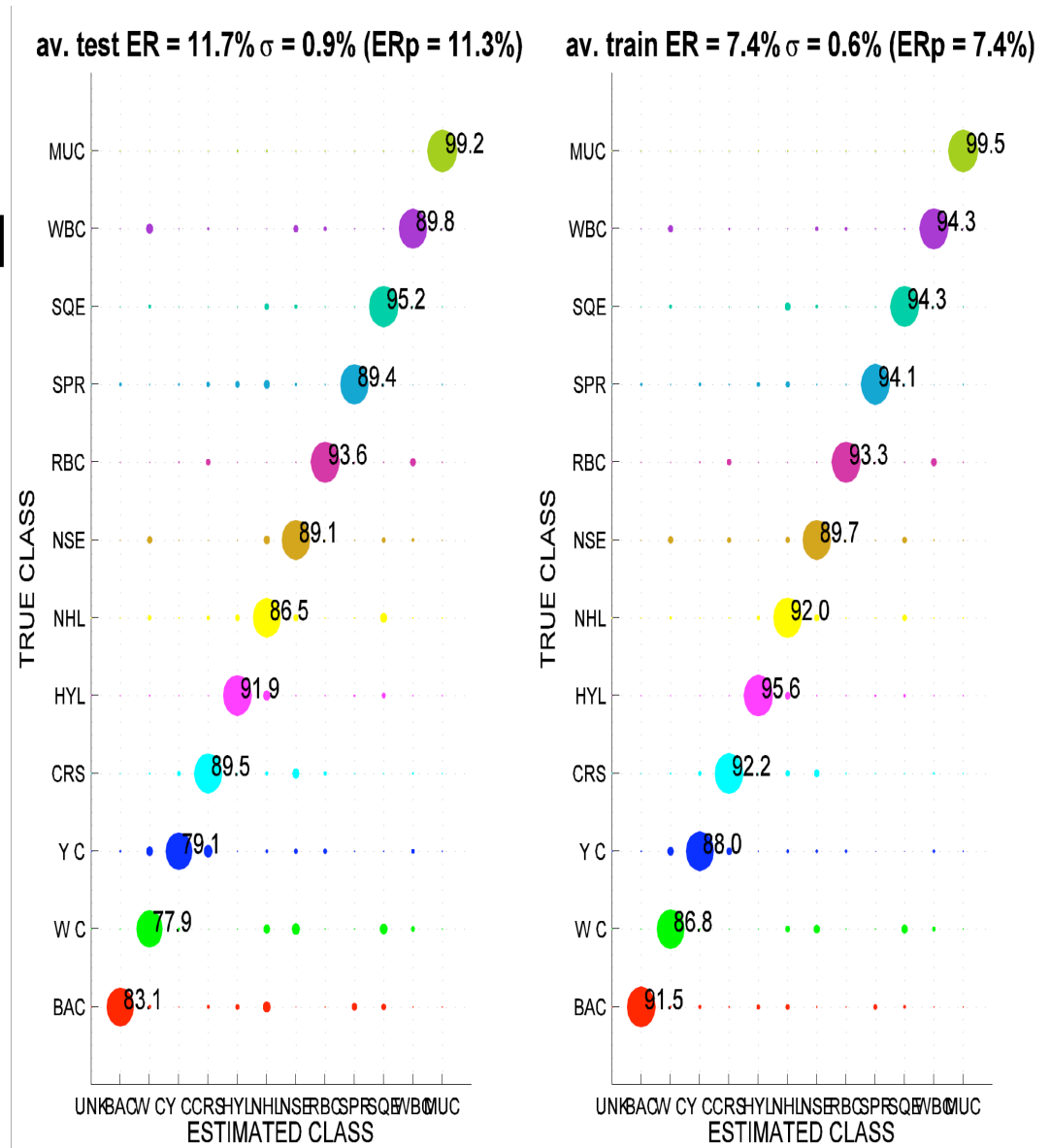
# Experiments: urine (2b)

- Comparison

MoG – **AdaBoost** - SVM

**AdaBoost**

10 generated hypothesis



# Experiments: urine (2c)

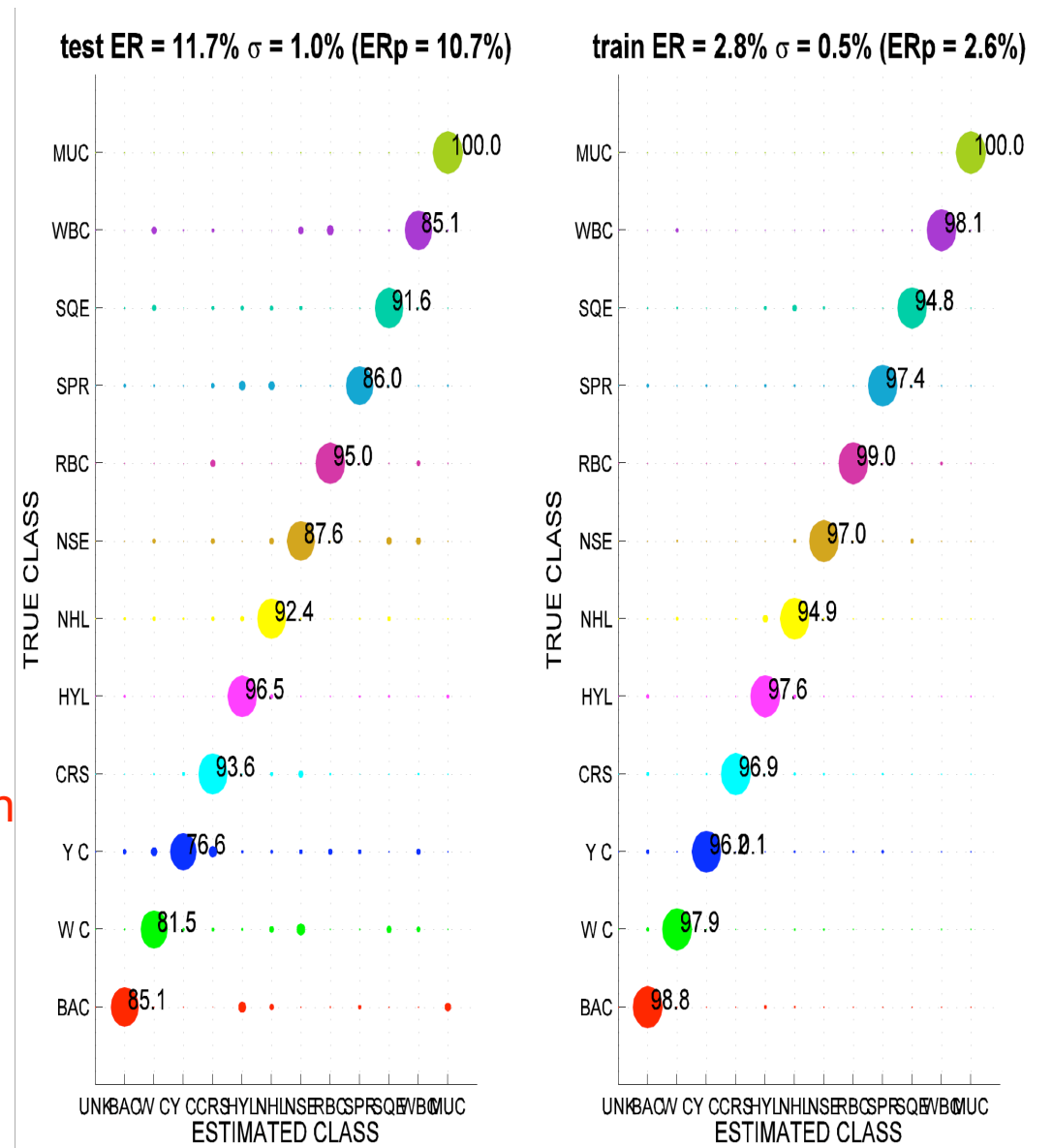
- Comparison  
MoG – AdaBoost - **SVM**

# SVM

$$C=5, \sigma=1, D=20$$

The best performance with a training on 480 images

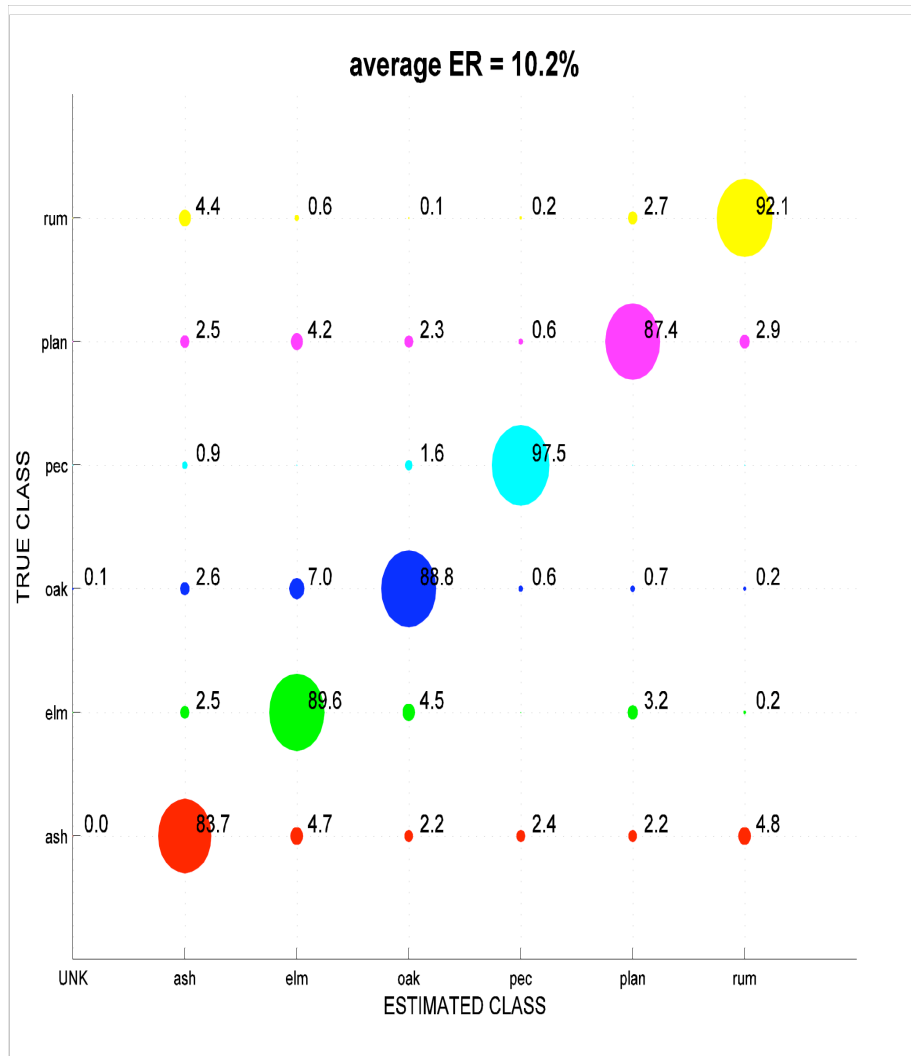
## Overfitting



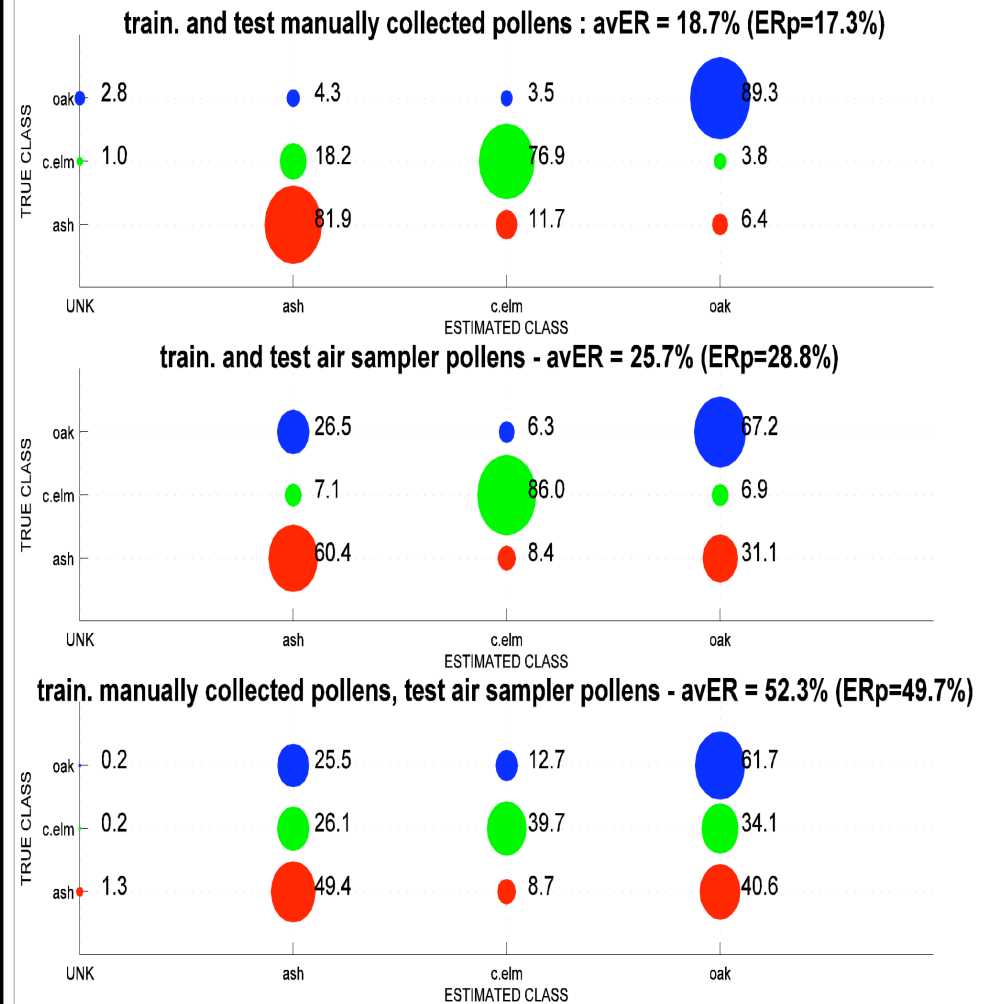


# Experiments: pollen

MoG & local jets:  
6 “pure” pollens categ. with >1000 img.



“pure” vs “Burkard” data base

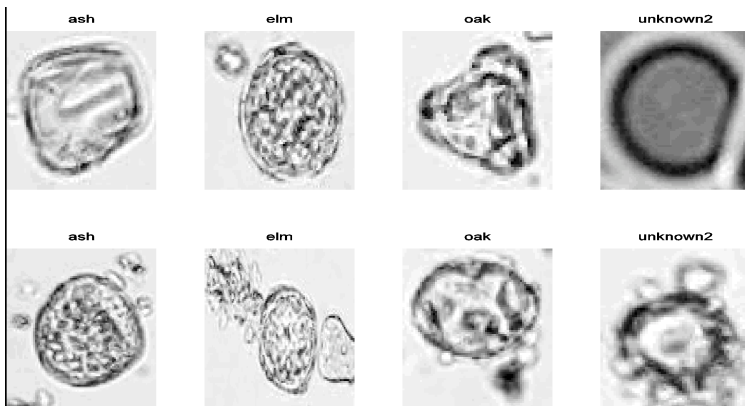


# Whole system

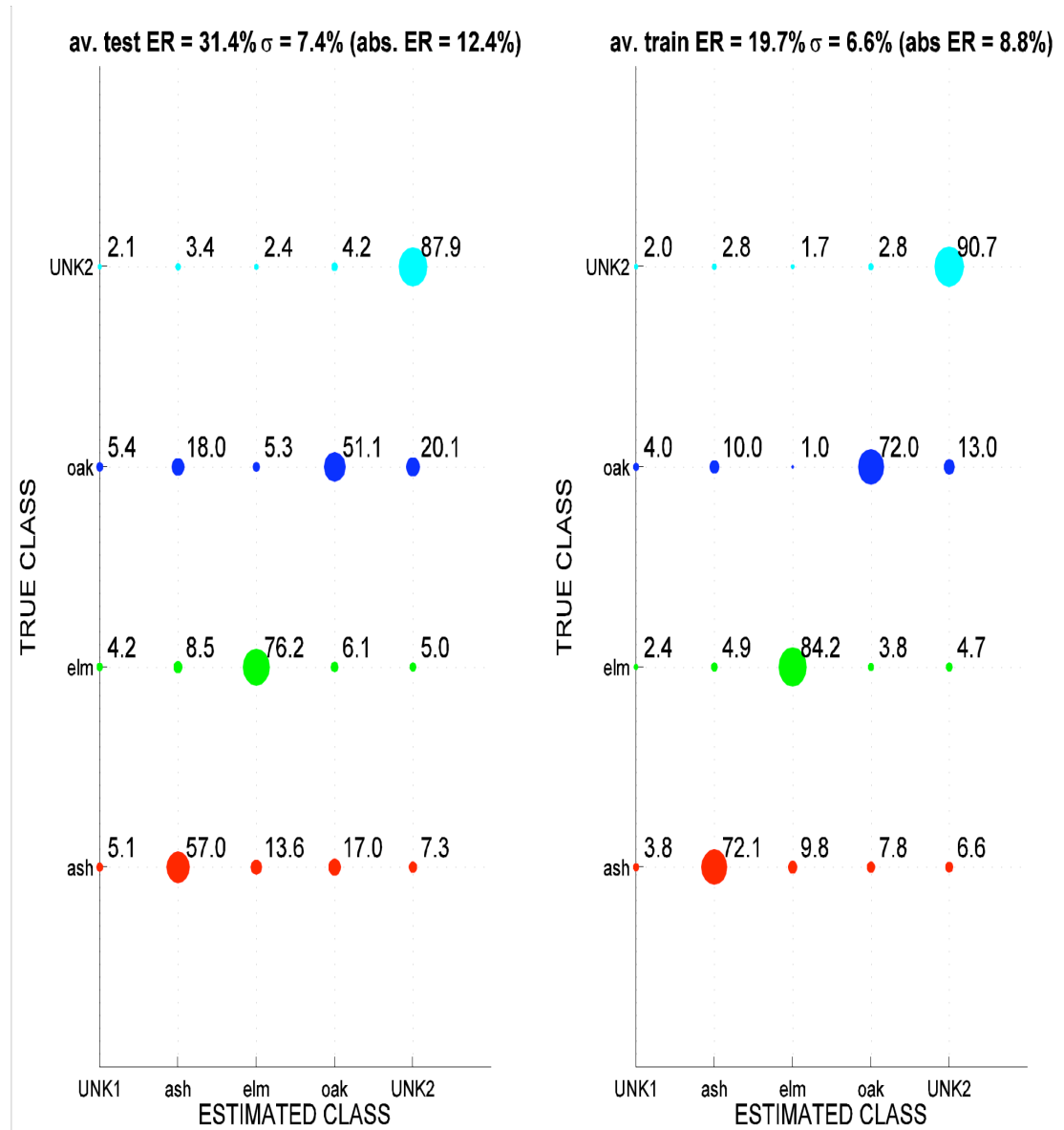
1. Build a “detector” data base
  - a) Folder “unknown2” for false alarms
  - b) 3 categories with more than 50 img.
  - c) Keep the ratio 1/10 between pollen and junk particles:  $72 + 119 + 67 + 258 \cdot 10$  tot. nr. img.
2. Tuning on the four categories by taking 10% of images for testing and making use of a more complex model for class of false alarms
3. 100 experiments taking 10% of images in each class for test

# Whole system: experiment

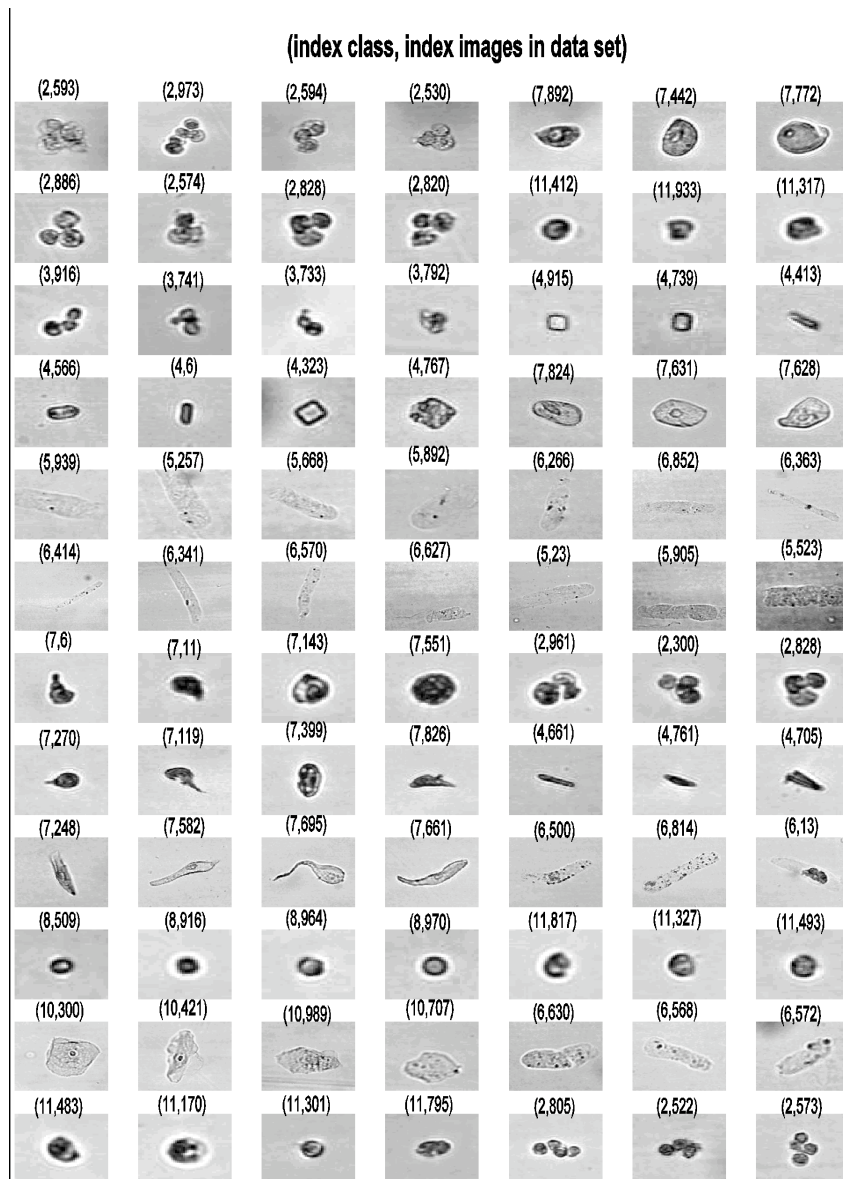
## DB detector



| class | D  | ng | type  | thres |
|-------|----|----|-------|-------|
| Poll. | 16 | 2  | spher | 0.5   |
| junk  | 16 | 6  | full  | 0.5   |



# Errors in classification



CENTRAL COLUMN: not correctly classified images

FIRST THREE COLUMNS: nearest images belonging to the true class

LAST THREE COLUMNS: nearest images belonging to the estimated class

- Misclassified images have usually  $P[C_k | x] < 0.9$
- Some errors could be avoided by taking into account info about shape
- Some errors are due to intrinsic ambiguity of database

# Our contribution

- Improved features suitable to extract information from images with low resolution and poor contrast
- General approach used in classification with low error rate
- No segmentation is required
- First feasible work on pollen recognition
- First research in which detection and classification are combined to detect pollen in optical microscope images

# Future work

- Enrich features with information about shape and color
- Improve detection: towards a fast and general approach

## *POLLEN PROJECT*

1. *New version of software for semi-automatic labeling : build a big DB of “Burkard” pollen*
2. *Define a new mechanical system for pollen collection with an increased spatial resolution*
3. *Real-time & low cost instrument for pollen level count*

# Acknowledgements

- Pietro Perona - California Institute of Technology
- Ruggero Frezza – Univ. degli Studi di Padova
- Computer Vision Laboratory at Caltech

Marco A., Anelia A., Ulrik B., Claudio F., Michael F., Alex H., Fei Fei L.,  
Pierre M., Arvind R., Ueli R., Silvio S., Melissa S., Lihi Z.

***PARTY***

***FRIDAY 9:30 pm***

***CENTRAL RECREATION ROOM  
CATALINA APT.***