

*Automatic visual recognition of
biological particles*

Marc'Aurelio Ranzato

26th August 2004

Contents

1	Introduction	1
2	Microscopic analysis of biological particles	3
2.1	Measuring airborne pollen level in air	3
2.2	Urinalysis	6
2.3	Conclusion	7
3	Database	8
4	Overview on related work	14
4.1	IRIS urine analysis system	18
4.2	Fourier Mellin Transform	21
4.3	Detection of small objects	25
4.4	Using SEM and 3D data for pollen recognition	26
4.5	Pollen identification with Paradise network	27
4.6	Conclusion	28
5	Detector	29
5.1	Outline	30
5.1.1	The difference of Gaussians image	33
5.1.2	Harris algorithm	33
5.1.3	Bounding box determination	35
5.1.4	Overlap criterion	36
5.2	Examples	37
5.3	Experiments	42
5.3.1	Results on whole database	45
5.4	Analysis of results	47
6	Feature extraction	52
6.1	Local Jets	52
6.1.1	Average and application of non-linearities	56

6.1.2	Decreasing the dimensionality	61
6.2	Image and Power Spectrum Principal Components	62
6.3	Interpretation	67
7	Classification	68
7.1	Mixture of Gaussians	68
7.1.1	Training	69
7.1.2	Test	70
7.2	Combination of features	70
7.2.1	Independence hypothesis	71
7.2.2	Without hypothesis	71
7.3	Boosting	72
7.4	Support Vector Machines	76
7.5	Experiments	79
7.5.1	Tuning	80
7.5.2	Final experiments	91
8	Analysis of errors	109
8.1	Detector	109
8.2	Classifier	110
9	Whole system	113
9.1	Classification of 8 pollen categories with false alarms	114
9.2	Classification during specific seasons	120
9.3	Conclusion	124
10	Conclusion	125
	Bibliography	128

Chapter 1

Introduction

In this report, we propose an automatic visual recognition system for biological particles. In particular, we focus on recognition of particles that are found in microscopic urinalysis, and we also deal with the recognition of airborne pollen grains.

Our approach is general, segmentation free, able to achieve a very good performance on many different categories of particles. For these reasons, the system is suitable to be used for recognition of other kinds of biological particles.

In chapter 2, we describe the goal of microscopic analysis in aerobiology and in urinalysis and the usually applied techniques. We conclude this part with an assessment on the need to automate these analyses that will justify this research.

Chapter 3 introduces our databases giving a complete list of available images with a brief description about their characteristics.

The following chapter provides an overview on related work which is a reference point to evaluate the performance of our system. Moreover, we introduce the main stages of a recognition system: detection and classification.

Chapter 5 tackles the problem of particle detection in a given image. We will discuss a detector that is based on a filtering approach. We will describe how it works and what its performance and limitations are.

Chapter 6 reviews the features developed for the classifier. These features are derived from previous work of Schmid et al. [10] and Torralba et al. [31]. Thanks to some kind of elaboration, we are able to get from each image a new kind of very informative feature.

In chapter 7, we discuss many classifiers: Bayesian classifiers, a classifier inspired by AdaBoost algorithm and a last one using support vector machines. This chapter is completed by an experimental part which shows the experi-

ments done to tune the system and to compute the global performance.

The following chapter provides an insight about errors done by detector and classifier during the previous experiments.

In chapter 9, we conclude with an extensive analysis of the performance of a complete system, namely detector followed by classifier, for pollen recognition.

The last chapter summarizes the main points of this research. We discuss our contribution in automatic visual recognition of particles and the tasks that will challenge future works.

Chapter 2

Microscopic analysis of biological particles

Microscopic analysis is used in many fields of technology and physics, for example in aerobiology, geology, biomedical analysis, industrial product inspection.

Basically, the input image has a resolution of about 1024x1024 pixels while objects of interest have resolution of about 50x50 pixels. In all these applications detection and classification are difficult, because of the poor resolution and maybe strong variability of objects of interest, and because the background can also be very noisy and highly variable. In this work, we focus on recognition of biological particles and especially on recognition of airborne pollen and on recognition of particles that can be found in urine. Since our approach is general and our performance is very good on many categories of different kinds of corpuscles, we can claim that this research and its results are naturally applicable to other many kinds of biological particles found in microscopic analysis. In this research, we develop a recognition system that is not a specific case study, but it is segmentation free and it can be easily updated in order to deal with new classes.

Focusing in the next sections on airborne pollen recognition and urinalysis, we will explain why and how these analyses are performed, what the problems of manual analysis are, and at the end, we will assess the need to automate this process.

2.1 Measuring airborne pollen level in air

Estimates from a skin test survey suggest that allergies affect as many as 40 to 50 million people in the United States of America. Allergic diseases

affect more than 20% of the U.S. population and are the sixth leading cause of chronic disease. In this country, the estimated overall costs for allergic rhinitis in 1996 totaled 6 billion dollars and two years later, the increased absenteeism and reduced productivity due to allergies cost to companies more than 250 million dollars [1]. Moreover, from 1982 to 1996, the prevalence of asthma increased to 97% among women and 22% among men. These statistics are an example that perfectly corresponds to the situation and trend in all other countries in the world.

An allergy is an abnormal reaction to an ordinarily harmless substance called *allergen*, [1] [30]. When an allergen is absorbed into the body of an allergic person, the person's immune system views the allergen as an invader and a chain of abnormal reactions begins. The effects of this response are runny nose, watery eyes, itching and sneezing. People with these symptoms are unable to work and even to sleep. The most common allergens are: pollen particles, molds, dust mites, animal dander, foods, medications, insect stings. Pollen grains are clinically the most important outdoor allergens and the most allergenic species are: *American elm*, *paper birch*, *red alder*, *white oak*, *white ash*, *olive*, *mulberry*, *pecan*, *black walnut*, *sycamore*, *grass*, *chenopod*, *rumex*, *plantago*. Note that asthma and allergies can affect anyone, regardless to age, gender, race.

In order to make forecasts and to aid in the diagnosis, treatment and management of allergic diseases, many stations with air sampling equipment are spread in the territory to collect airborne pollen particles.

The most popular device is the *volumetric spore trap*, see Figure 2.1, which draws air into the sampler at a given rate through a little opening. The particles in the air land on an adhesive coated microscope slide attached to a slowly rotating wheel. After a period of sampling, for example one day but most usually one week, the pollen grains caught on the sticky slides are counted under the microscope by simply cutting the long slide into several pieces. The analysis at the microscope is performed by trained observers (aerobiologists) who count and classify pollen grains. This method is slow, expensive, and inaccurate. First of all: the response time is inadequate for many applications. In a typical installation pollen particles are collected during the week on sticky tape. The tape is analyzed only once a week. The results of such analysis are therefore sometimes available one week after the fact, rendering them useless for preparations of medical response in hospitals.

Second, the analysis of one weekly tape takes up to 8 hours of work by a skilled professional, thus, the yearly cost of measuring pollen contents in the air at one location could approach \$30,000, too expensive for many institutions and too expensive to allow fine spatial sampling of air pollen contents.



Figure 2.1: Burkard volumetric spore trap is used to collect airborne pollen grains and fungi.

For example, the National Allergy Bureau, section of the American Academy of Allergy, Asthma and Immunology's Aeroallergen Network, is the institution responsible for reporting current pollen count to the media and it has *only* 75 counting stations throughout the US and its members are all *volunteers*; if you check the website of AAAAI [1] the pollen counts are never updated to the last week!

Most importantly: relying on humans produces inaccurate measurements. This for two reasons: first, the process is tedious, it is well documented that the attention of a human operator tends to flag after 30 minutes on a demanding repetitive job. Second, in order to accomplish the task at all, human operators sample coarsely the collected tapes. Measurements are thus accurate for high pollen counts, but inaccurate for low pollen counts, an even more inaccurate as to estimating the concentration of pollen grains over time. It is entirely possible to miss the presence of a given pollen in the air if the critical time when it was released is not sampled by the operator. Moreover, it is difficult to provide accurate pollen levels for areas not near to a counting station and so the actual counts are useless for most of the physicians.

To summarize, there are many reasons to justify the recent strong interest and the need to automate the count and identification of airborne pollen particles. The manual collection and analysis is not adequate because it is

too slow, too expensive, not precise and not able to cover all the territory.

2.2 Urinalysis

Urinalysis can reveal diseases that have gone unnoticed because they do not produce striking signs or symptoms; examples are diabetes mellitus, various forms of glomerulonephritis and chronic urinary tract infections [2]. There are three kinds of urinalysis: macroscopic, chemical and microscopic. The first is direct to visual observation to assess the color and cloudiness. The second one, uses a dipstick to determine pH, specific gravity, content of proteins, glucose, etc. and it is based on the color change of the strip and on its comparison to a color chart. The third analysis, the one of our interest, requires a light microscope and is the most complex. A sample of well-mixed urine is centrifugated in a test tube and then, a drop of sediment is poured onto a glass slide for the examination. Using low magnification, a well-trained expert identifies crystals, casts, squamous cells and other “large” objects. After another adjustment of the microscope to get a higher magnification, the expert can count the number of smaller objects like bacteria and red blood cells.

Particles in urine can be classified into 12 categories, they are: red blood cells, white blood cells, bacteria, hyaline casts, pathological casts, crystals, squamous epithelial cells, non squamous epithelial cells, yeasts, white blood cell clumps, sperm and mucus. The microscopic analysis is very useful and generally required because it is non-invasive and it gives several indications about disease progress and therapeutic efficacy, [6].

However, this kind of analysis if manually done is intrinsically not precise, time consuming and expensive. There is no standardization in the process of taking a volume of fluid, there is no reliability of the result because the experts may have a different training and experience and last but not least, this work is annoying because repetitive and difficult. The difficulty relies on the strong similarity among some categories of particles and in the variability existing also among corpuscles belonging to the same family. Moreover, this process is slow and expensive for hospitals.

For these reasons, an automatic recognition system is required in several situations above all when many specimens have to be analyzed. Some systems for automatic recognition of particles in urinalysis are currently sold. An interesting machine using computer vision approach is made by IRIS, a company in the field of biomedical analysis. This company also provided us the urine database.

Our interest in urinalysis is justified because the manual analysis is not efficient in terms of precision, cost and time and because the automatic recognition can be still improved. In the next chapter, we will describe IRIS system and its very good performance but we will highlight also its main flaws. Other systems using different techniques, such as analysis of particle refraction when these particles are hit by a laser beam, have also drawbacks because of their suboptimal performance and the difficulty to verify analysis outcomes.

2.3 Conclusion

Both aerobiology and urinalysis need automation.

First, manual analysis is slow. For instance, the pollen level in the air is usually given in the following week and so, physicians have available this information when by this time it is too late. Second, manual analysis requires very skilled experts with a high cost for the community and the institutions. Third, because this kind of work is repetitive and difficult, results may be not accurate. Experts' results depend on their experience, their subjective judgment, their way to set up the system. There is a need of standardization in the measurements that can be achieved only with an automatic system.

On the other hand, today there is no completely automatic system to detect and identify pollen particles and the machines for urinalysis have a lot of margin of improvement.

Besides, visual recognition of particles in microscopic images and more generally, recognition of small object categories in images with poor resolution and contrast is a field of research which has never been studied deeply but nonetheless, has many challenging problems to tackle and can find several applications.

Chapter 3

Database

This chapter describes the databases we used in our work.

A company in biomedical analysis, IRIS, provided us the database of cells in urine; these particles are: red blood cells, white blood cells, bacteria, hyaline casts, pathological casts, crystals, squamous epithelial cells, non squamous epithelial cells, yeasts, white blood cell clumps, sperm and mucus. These are twelve categories, samples of this database can be found in Figure 3.1. The resolution is variable and goes from 36x36 up to 250x250 pixels with $0.68\mu m/pix$.

Table 3.1 shows the number of images in each class of urine database.

CLASS	NR. OF SAMPLES
1. bacteria	1000
2. w. blood c. clumps	1000
3. yeast	1000
4. crystal	1000
5. hyal casts	1000
6. path. casts	860
7. non squam. epith. c.	999
8. red blood c.	1000
9. sperm	1000
10. squam. epith. c.	1000
11. white blood c.	1000
12. mucus	546

Table 3.1: Number of images in each category of urine database

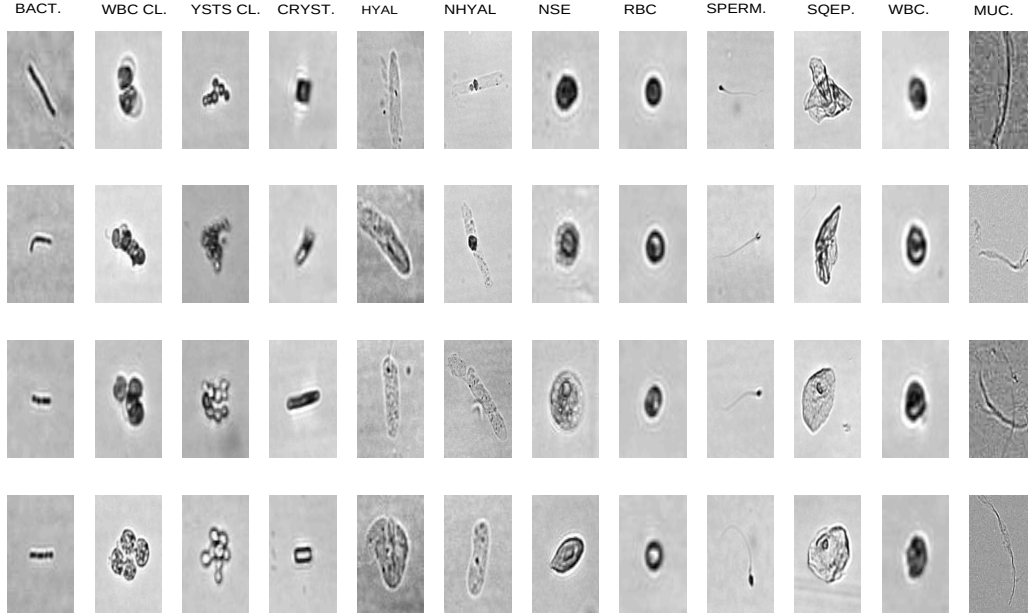


Figure 3.1: Urine database: samples belonging to the twelve categories.

The other database is a collection of pictures of airborne pollen particles taken with the volumetric spore trap (see par. 2.1). Some examples of these pictures are given in Figure 3.2 and they show the extreme variability in the background. These images have a resolution of 1024×1280 pixels with $0.5 \mu m / pix$.

In order to build this database, we started developing a software for enabling human operators to manually collect a large training set, consisting of about 1000 samples of grains of pollen for each of 32 species which are found in Pasadena's area. For this purpose we built a Matlab graphical user interface (see Figure 3.3) with which we have already collected up a number of samples summarized in Table 3.3. Thanks to this software, we have a reference list which stores for each image the position and the genus of each pollen identified by some expert. Given this information, we derived a database of patches centered on pollen particles. Figure 3.4 shows some samples belonging to these pollen classes. In chapters 5 and 7, we will see that there is sometimes a high variability in shape and texture also for samples belonging to the same class.

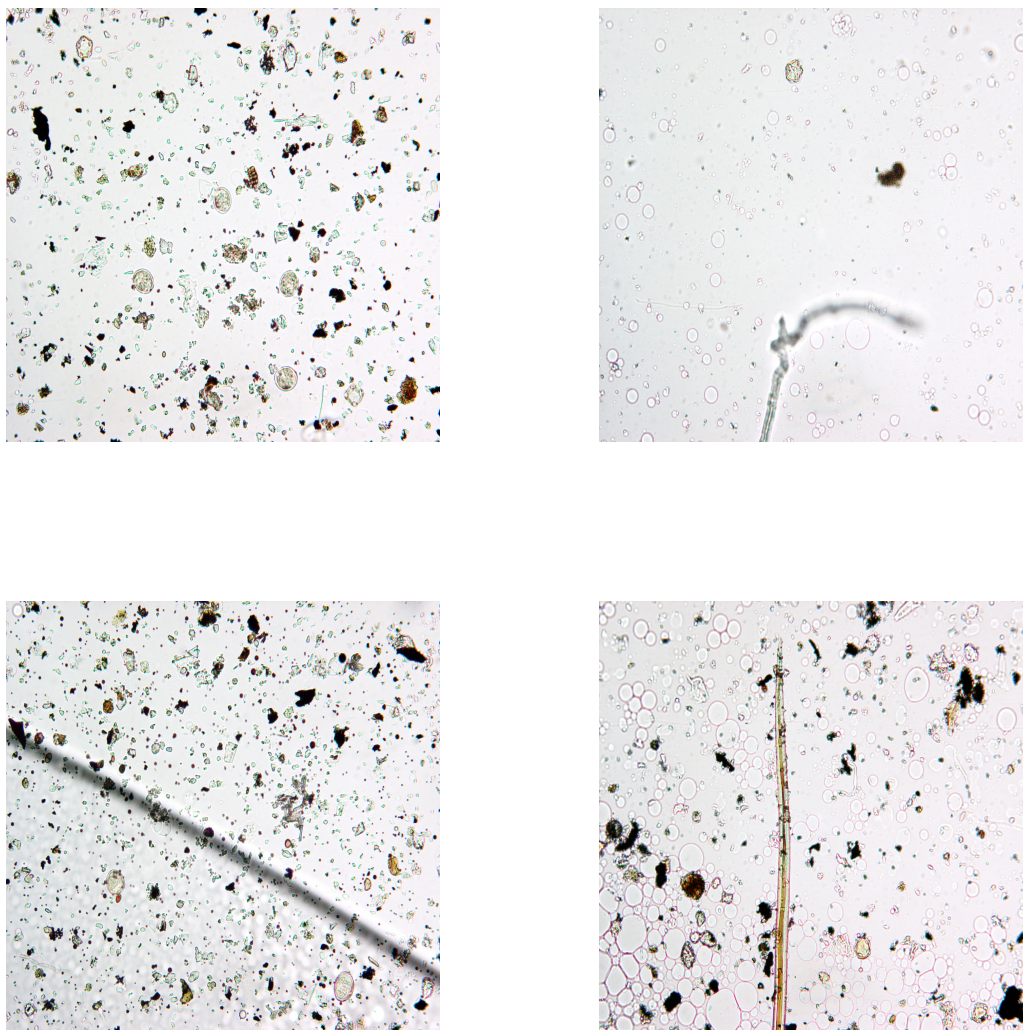


Figure 3.2: Example of images in pollen database. There can be images with a high density of particles (upper left corner), with a low density of particles (upper right corner), with big objects inside (lower left corner) and with bubbles of glue (lower right corner).

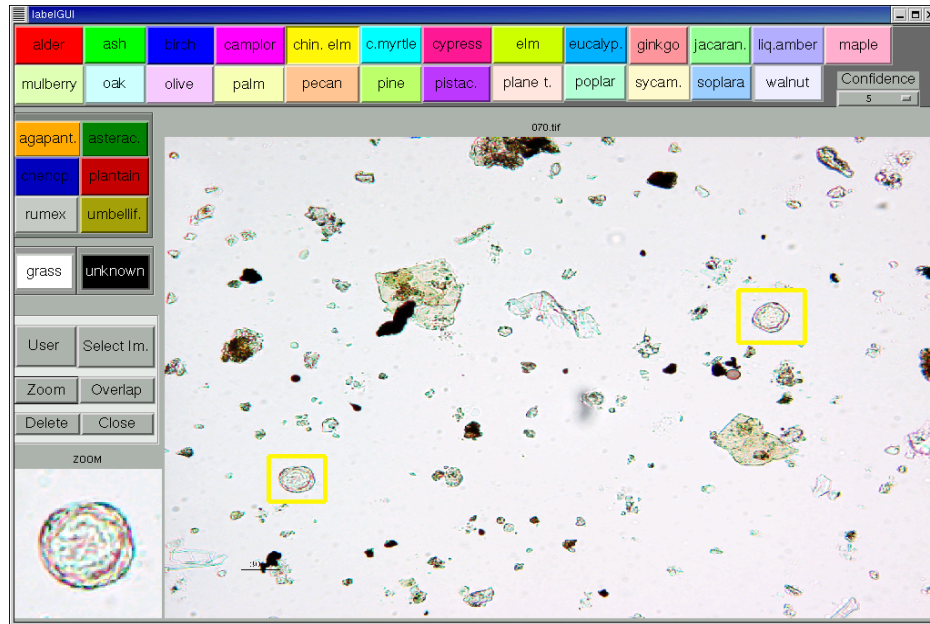


Figure 3.3: Screen shot of software for labeling pollen grains. The expert has to identify pollen particles in the image and to select around them boxes. The color of the box is related to the pollen genus. The user can remove and zoom boxes, view the previous work and change the confidence of his classification. The output is a list with position and genus of each identified pollen.

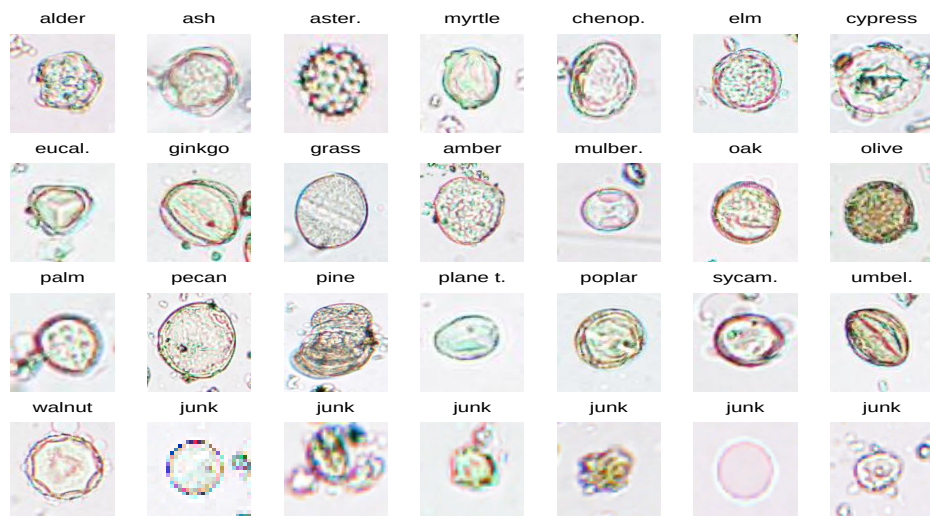


Figure 3.4: Examples of pollen grains collected in Caltech campus. In the last row, there are other particles that are similar to pollen grains and that are found in the acquired images. The average resolution is about 50x50 pixels.

folder	images
*010403	195
022203	19
022303	10
022403	15
030310	22
030311	80
*032102	115
*042304_0-12	100
*042802	180
091903	30
120102	44
120202	8
*122902	158
*ChinElm	216
TrialPollen	71
<i>TOTAL</i>	1429

Table 3.2: Pollen database: list of folders and images in each folder. The (*) denotes the new available data in addition to the old database (March 04) (see [26]).

genus	number of samples
alder	416
ash	329
asterac.	6
birch	4
camplor	9
chenopod	32
chin. elm	522
c. myrtle	3
cypress	46
eucal.	30
ginkgo	3
jacaranda	282
liq. amber	44
mulberry	111
oak	413
olive	326
palm	39
pecan	4
pine	705
plane t.	10
poplar	7
sycamore	33
walnut	36
unknown	263
pistach.	0
plantain	0
elm	1
rumex	1
umbellif.	4
grass	19
<i>TOTAL</i>	<i>3686</i>

Table 3.3: List of pollen particles found in the images of tab.3.2. In bold characters, we highlight the species with more than 100 samples.

Chapter 4

Overview on related work

The aim is to build a system for automatic recognition of particle categories. Furthermore, we are interested in a general approach enabling us to work with several kinds of corpuscles which are found in microscopic analysis. In this way, it is easy to add new classes to the already considered set and, no new customized reprogramming is required.

Generally, input images of a such system have resolution of nearly 1024x1024 pixels, while the objects to which we are interested on have square bounding box with side between 30 and 200 pixels. The average side is around 60 pixels, then, the system has to handle objects with low resolution. Its output is the number of detected particles in each category.

The automated analysis proceeds in two parts: *detection* and *classification*, an outline is shown in Figure 4.1.

In the detection stage the system will select those parts of the image which are likely to contain a particle of interest, for example a pollen. This process is akin to visual attention in humans: most of the computation is devoted to the most promising areas of the image.

The second stage, that is classification, consists of taking image portions which contain a visual event associable to a particle of interest, and attributing such event either to one out of a number of known species or, to an “unknown object” grab-bag category.

In urinalysis detection is not a problem, because in the pictures of stream (see next section for a detailed description) all particles have to be analyzed. Once that overlap among cells is avoided, all objects in the image will be classified. Moreover because of the typical low concentration of particles in the fluid, particles are quite well distributed on the original image. Detection for urinalysis is not a great task. For this reason, our urine database is a collection of patches with particles already well centered; whereas in pollen

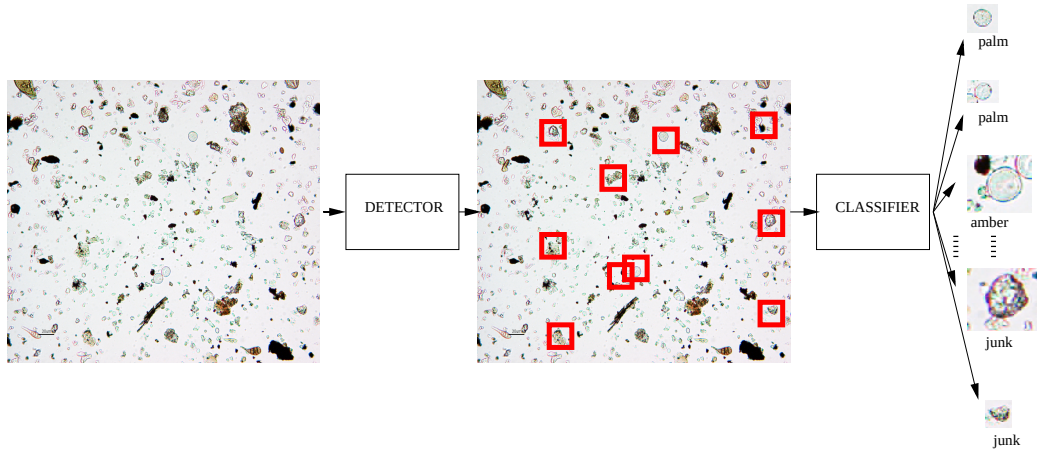


Figure 4.1: Outline of recognition system. Detector finds interesting points in the input image and gives back to the classifier a certain number of patches. For each patch, the classifier establishes the class to which the object in foreground belongs.

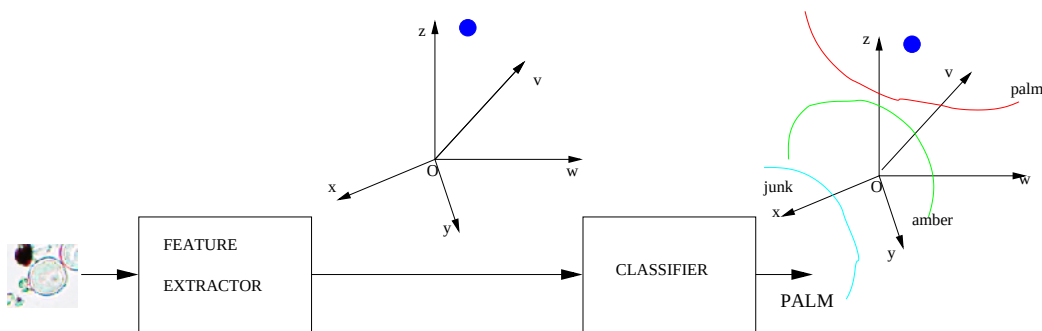


Figure 4.2: Outline of classifier: the system extracts from the input patch a set of features. In the training period, the system learns how to divide the feature space. In the test phase, the system assigns a class to the input image looking at the region where the test feature falls.

recognition, detection is still crucial, because the input images of the system are pictures of sticky tape.

A given section of tape has accumulated airborne particles for many hours. Such an image contains hundreds of particles like: pollen grains, dust, insect parts, etc. The detection process should quickly flag the location of those particles that have at least a small chance of being pollen particles. At this stage the number of false rejects, i.e. pollen particles that are not selected, must be extremely low because some species can have low counts and we have to take into account also classifier mistakes.

Classification of both urine particle patches and pollen patches has to establish if the particle is of interest, in which case the class will be also identified, otherwise, it has to be discarded.

This is a visual *pattern recognition* problem, whose solution is conceptually simple. Given a signal s belonging to a class k , $k = 1, \dots, K$, the main stages of classification are:

1. a number of features are extracted from the original signal to form the *feature vector* $x \in \mathbb{R}^D$, that is the image transformed into a *pattern*.
2. a decision function must be determined:

$$\begin{aligned} r : \mathbb{R}^D &\rightarrow 1, \dots, K \\ x &\mapsto r(x) \end{aligned} \tag{4.0}$$

In many cases, a discriminant function $g(x, k)$ is used:

$$r : x \mapsto \max_{k \in 1, \dots, K} g(x, k)$$

This can be regarded as a *trained pattern classifier*, an outline of a classification system is shown in Figure 4.2.

It is well-known, that in the pattern recognition the crucial step is designing informative features: if our features do capture the peculiarities of the appearance of our objects, then, almost any classifier will perform a competent job (Duda and Hart, [14]).

The task of classification even if conceptually simple, is practically a hard challenge. The classifier has to handle images with very low resolution and often poor contrast.

In addition, pollen identification is difficult because [30]:

- there is a large number of look-alike categories
- each genus of plant has many species (strictly speaking, we are classifying among genera)
- high winds may carry unusual kinds of pollen
- pollen shape and texture can be corrupted by previous impacts
- pollen shape depends on its orientation on slide; most frequently are seen in the equatorial position, in which they all appear as ovoid
- “apparently trees do not read literature and insist upon producing pollen grains with the same number of sides as some of other pollen kinds” [30]

Urine particle classification is difficult because:

- some cells are semi-transparent, i.e. casts and mucus
- almost all categories have high variability in shape, texture and size; for example, there is no defined shape and size for clumps, crystals can be square, rectangular or hexagonal, white cells can have a nucleus or a uniform texture
- there is a smooth transition between particles belonging to different categories and even human experts do not agree in the classification of some casts or blood cells; for example, a “big” red blood cell is completely not distinguishable from a “small” white blood cell.

To summarize, the system for automatic visual recognition of particles is composed by a detector and a classifier. The detector is simply achieved in urinalysis, because all objects in foreground are objects of interest while in pollen recognition the detector has to deal with a lot of non-pollen particles. The detector must have an extremely low missed-detection rate. The classification of patches passed by the detector, is similar in both cases. In both situations in fact, the classifier handles images with poor resolution and contrast, and there is a strong variability *between* classes, but sometimes just a little variability *within* some categories.

In the next sections, we will present approaches developed by other research groups in the field of visual recognition of particles or, more generally, visual recognition of small objects in images with low contrast and resolution.

We will start describing IRIS urine analysis system that is somewhat our reference point for the urine project, then, we will see another approach due to Dahmen et al. [12]; for this last method, we had enough details to make a program based on that research and so we can make a direct comparison between this technique and our method. Finally, we will focus on the detection problem and we will present the work of Burl et al. [9] about the identification of volcanoes in Venus surface, and we conclude introducing a technique for pollen recognition [27] and another technique for its identification, [15].

4.1 IRIS urine analysis system

Thanks to the courtesy of IRIS, we will refer to [6] and to many discussions with scientists from IRIS to describe their system. However, we have available only few information about the machine they are selling and we will be able to show only an outline of their technique.

The *iQ200 Automated Urinalysis System* is a bench-top analyzer able to perform chemical and microscopic analysis. The former one is focused on the determination of color, gravity and composition of urine. The latter one is used to analyze the sediment, this is useful in the diagnosis of renal and urinary tract diseases. Each kind of analysis is performed by two different but connected modules. The goal of the module for particle recognition is to overcome the limitations of manual microscopy: lack of standardization and precision, risk of biohazard, slow processing and high costs because it requires highly trained technologists. This analyzer is able to classify and quantify 12 particle categories (the same ones of our urine database) based on a visual recognition system.

In Figure 4.3, you can see an outline of this module. They take from a given specimen $2\mu L$ of centrifugated urine, they make a very thin flow through a channel and then, they take 24 pictures per second with a CCD digital camera coupled with a microscope. In this way, they get around 500 images with resolution 1024×1280 and $0.68 \mu m/pix$. Given a frame, they are able to detect patches centered on particles by morphological operations or simply by the application of a threshold. Note that they mechanically avoid the overlap among particles by regulating the thickness of the flow. Then, they extract from each patch features that take into account: the image size, the contrast of particle in foreground with respect to the background mean, the shape of the cell and its texture. Given these features from each patch, they train a neural network to make a decision in test. They claim that on our database their system has a very good performance with an error

rate¹ of nearly 5%. In the every day use of their system with the data of their customers, they state the classifier has an error rate of about 10%. A specimen is processed in nearly 1 minute and their system allows a continuous load with a maximum of 55 samples.

They claim that the hard tasks for this system are:

- to handle low contrast particles because they perform masking operations to determine some features like shape and size; for this reason, the most not correctly classified classes are Mucus and Casts
- to handle unknown particles

This last point requires a special comment. In urine, there are corpuscles of unknown origin which have never been studied and which are not of interest in the actual analysis. All these particles should be collected in an artifact class which collects also images out of focus or not well segmented. However, they state that this class fills all the feature space and makes classification quite difficult. For this reason, they have chosen the “abstention” rule: when the outcome of the system has a low confidence they discard the image and put it on the artifact class. In this way, they show to the user images with high confidence and reliability for the twelve categories of interest with possibility to check mistakes (the patches are stored in the memory of two computers positioned under the bench). There is still the problem that the artifact class captures also a lot of particles belonging to other categories. In our database, we have not available patches belonging to the artifact class.

We observe that a segmentation-free approach could solve the problem of misclassification among low contrast categories and boost the performance of the system. To take into account features based on shape, interest points can be extracted from a given patch. Our method is based on the first principle and achieves a very low error rate in these categories, is much simpler, is more easily updated with new classes even if it has a slightly higher global error rate.

¹The error rate is the ratio between the number of not correctly classified test images and the total number of tested images; its value allows to assess the performance of a classifier.

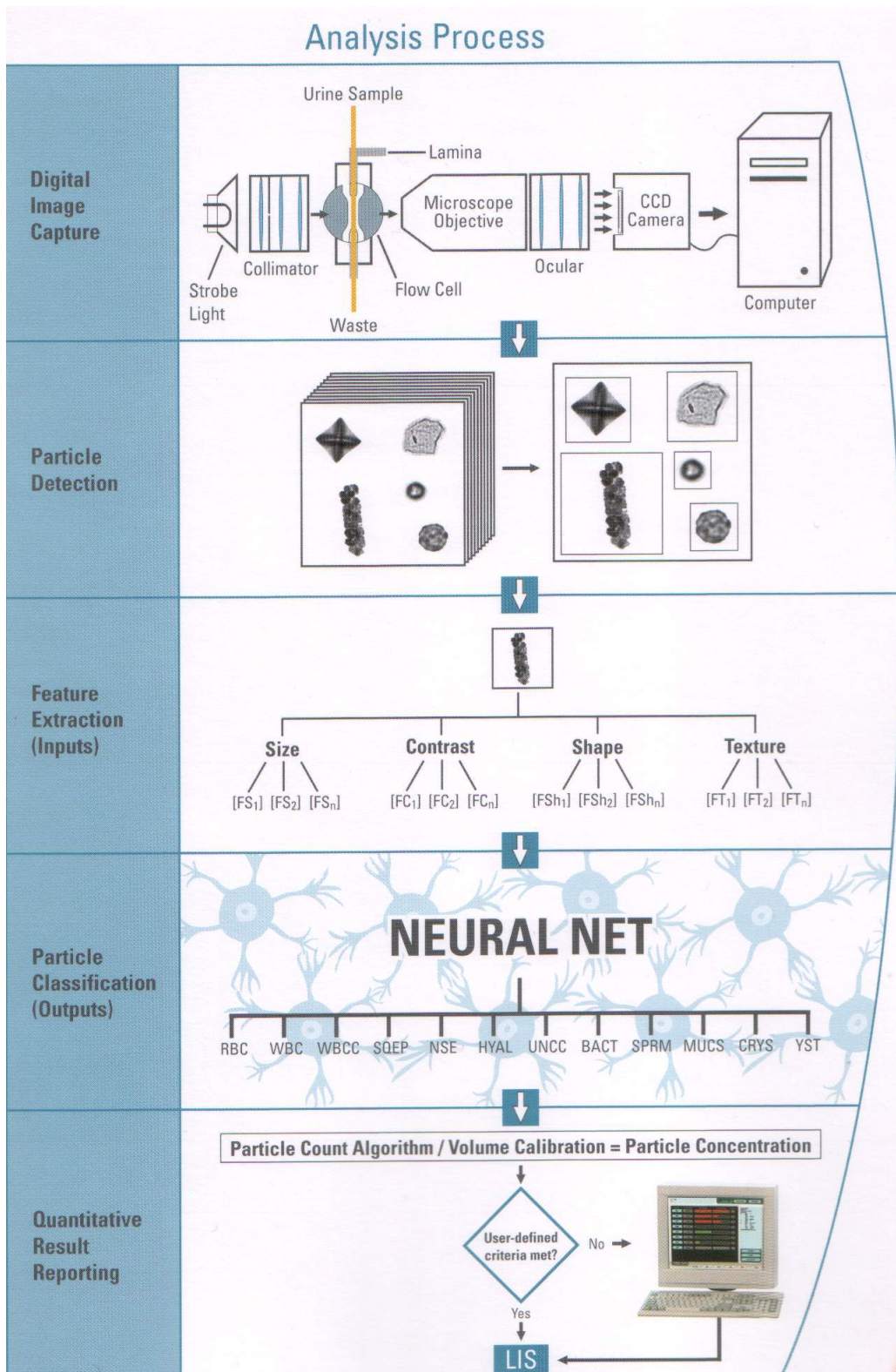


Figure 4.3: Outline of IRIS system for visual recognition of particles in urine (courtesy of IRIS), see par. 4.1.

4.2 Fourier Mellin Transform

The approach formulated by Dahmen et al. [12] is general and segmentation free like the system we are describing in this thesis. Their idea to extract features that are invariant with respect to shift, rotation and scale is powerful because it allows to obtain the same measurement from cells that are found in different positions, orientations and scales.

They aim to classify three different kinds of red blood cells (namely, *stomatocyte*, *echinocyte*, *discocyte*) in a database of grayscale images with resolution 128x128 pixels. Given an image, they extract Fourier Mellin based features that are invariant with respect to 2D rotation, scale and translation. Then, they model the distribution of the observed training data using Gaussian mixture densities. Using these models in a Bayesian framework they are able to perform the test and to achieve an error rate of about 15%.

Let be $f[x, y] \in \mathbb{R}^{N \times N}$ a 2D discrete image, its discrete Fourier Transform is defined as

$$F(u, v) = \frac{1}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} f[j, k] e^{-\frac{2\pi i (uj+vk)}{N}} \quad (4.1)$$

with $i = \sqrt{-1}$ and $u, v = 0, 1, \dots, N-1$. It can be easily shown that the amplitude spectrum A of $F(u, v)$ is invariant with respect to translation, inverse invariant with respect to scaling and variant with respect to rotation. By transforming the rectangular coordinates (u, v) of $A(u, v)$ to polar coordinates (r, θ) and by using logarithmic scale for the radial axis, image scales and rotations become shifts in the log-polar representation $\hat{A}$ of A . If you compute the amplitude spectrum B of the Fourier Transform of $\hat{A}$, you can extract features which are invariant with respect to rotation, scale and translation. Moreover, $\hat{A}$ is real valued, thus B is symmetric and you can consider the values that fall in window of size for example 25x12 pixels (the origin will fall in the middle of a longer side of this window). They reduce the dimensionality of the space to d using a linear discriminant analysis (we suppose they apply and we applied in our implementation Fisher's discriminants, [7] [25]).

For example, in our implementation of their system using our urine database we sampled the images to a common resolution of 90x90 pixels². So, reshaping the image in a column vector we decrease the feature space dimension from 8100 to 300 thanks to the symmetry of Fourier Mellin transform and to the windowing operation, then we achieve a further reduction projecting

²Note that when an image has a resolution smaller than 90x90 pixels we add a frame with values equal to the estimated background mean.

the data along directions given by Fisher linear discriminants and thus, we work in a final 11 dimensional feature space.

In Figure 4.4, we show the Fourier Mellin Transform of a bacterium image. Note that in order to change the coordinates from rectangular to log-polar you have to apply the inverse mapping and interpolate the values. If (x, y) and (r, θ) are the coordinates in the cartesian system and in the polar one respectively, then

$$\begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = \arctan y/x \end{cases} \begin{cases} x = r \cos \theta \\ y = r \sin \theta \end{cases} \quad (4.0)$$

Given a grid of (r, θ) values, the correspondences in (x, y) are computed and the values in these points are taken (generally with interpolation).

With reference to Figure 4.4, the plot on the upper left corner is obtained automatically by Matlab command *cart2pol*, instead the plot on the upper right corner is a mesh of the data computed by inverse mapping. You can also notice the symmetry of the Fourier Mellin Transform in the lower right corner; the box superimposed shows the window of features that will be extracted.

Once a Gaussian mixture model is estimated for each class, to classify an observation $x \in \mathbb{R}^d$ the Bayesian decision rule is applied

$$x \mapsto r(x) = \operatorname{argmax}_k p(k)p(x | k) \quad (4.0)$$

where $p(k)$ is the *a priori* probability of class k , $p(x | k)$ is the *class conditional* probability for the observation x given class k and $r(x)$ is the classifier decision.

The results of this technique on our database are quite good with an error rate of about 21% for the classification of the 12 categories of urine database, see Figure 4.5.

This performance is consistent with the one achieved by the authors on their database, error rate of 15% on three classes. Using the same data, our technique is able to halve this error rate (see par. 7.5.2).

This method is particularly interesting because it is able to achieve a good performance with quite simple features and it does not require any segmentation. On the other hand, the performance is still not acceptable for any real application and not comparable with commercial systems.

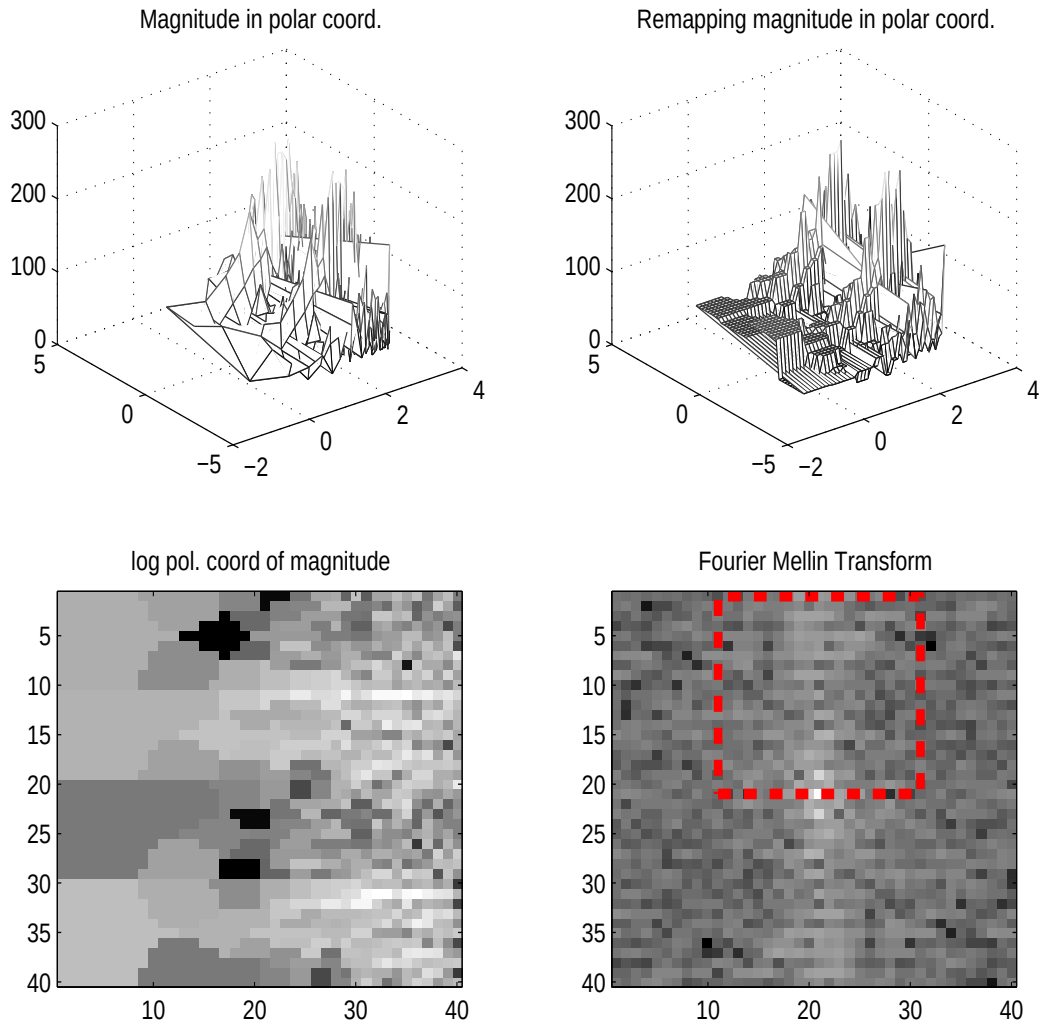
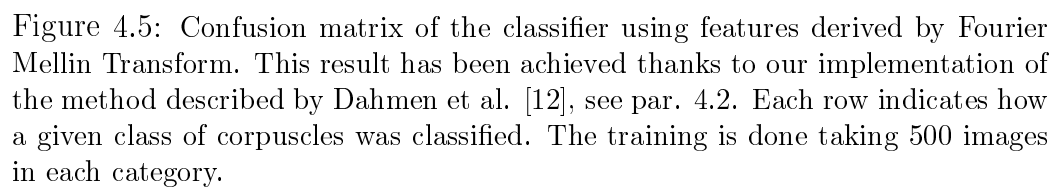


Figure 4.4: Operations involved in the computation of Fourier Mellin Transform: example taken from a bacterium image. Upper left corner: mesh of “cart2pol” command to Fourier Transform of the image; upper right corner: mesh obtained by inverse mapping; lower left corner: magnitude of Fourier Transform in log-polar coordinates; lower right corner: Fourier Mellin Transform.



4.3 Detection of small objects

In this section we will refer to the research of Burl et al. [9] about the identification of volcanoes in Venus surface. A typical image, of their database is shown in Figure 4.6. Their database had many thousands of 1024x1024 images derived from the successful Magellan mission to Venus of NASA-JPL (1989). Their task was to identify in this large database the volcanoes by an automatic vision system. We are referring to this work because the detection of these small objects in such a big image is similar to the detection of pollen grains in a picture of sticky tape piece.

As we did to build the pollen database, through a graphical user interface they labeled examples in a certain number of images. Applying a suitable matched filter and looking for the maximum values in the cross-correlation image they find interest points. Given an interest point, they consider a patch of $k \times k$ pixels centered on it and they use principal component analysis (or discrete Karhunen-Loeve transform [17]) to reduce the space dimension. Finally, the classification is achieved using the quadratic classifier, that is they model the two classes (volcano and not-volcano) with gaussian densities and then, in test they apply Bayes' rule

$$p(\omega_i | y) = \frac{p(y | \omega_i)p(\omega_i)}{p(y | \omega_1)p(\omega_1)p(y | \omega_2)p(\omega_2)} \quad (4.0)$$

where y is the observed feature vector and ω_i is the i -th class.

The main common aspects between this research and our research are:

- method to build the database (we are referring to the pollen database)
- the philosophy of the detector for which we allow a high number of false alarms to get an high detection rate
- the classifier philosophy because there is a modeling of training data and then a decision based on Bayes' rule

On the other hand, the main differences are:

- their classifier works in a binary case (only two classes)
- their detector handles images with a lot of gaussian noise, instead our (pollen database) images have only much more variability in the background
- our categories can have less variability in size but generally, have more variability in shape and texture

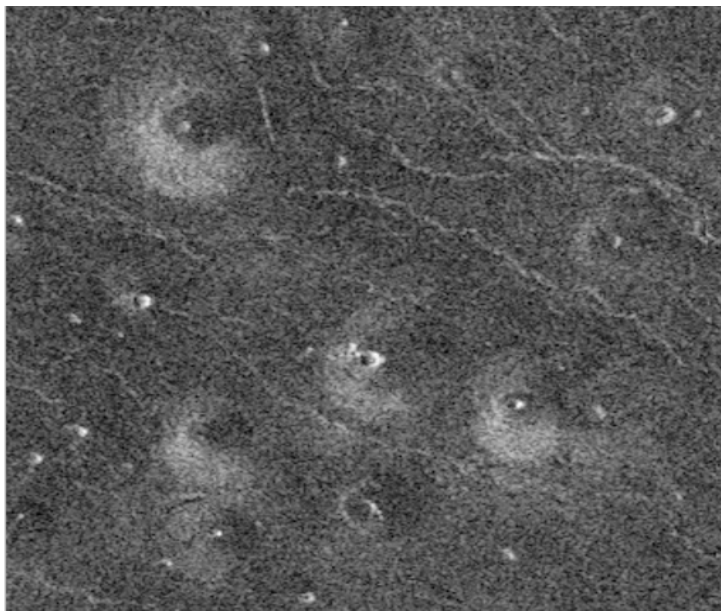


Figure 4.6: This image shows a $30\text{Km} \times 30\text{Km}$ region on Venus containing a number of small volcanoes.

4.4 Using SEM and 3D data for pollen recognition

We describe here a research developed mainly by Ronneberger [27] at Freiburg University in collaboration with the weather service of Germany and Swiss. In their work, they highlight that by using fluorescence microscopy instead of the common translucent one it is easy to discriminate between biological and non-biological material. Moreover, they observe that an expert has to analyze a pollen grain on different planes to be sure of his classification. For these reasons, they take with the highest available quality 3D data of pollen grains by a laser scanning microscope.

Given a database in which each pollen is represented by a stack of images (one image for each scanned plane), they compute some gray-scale invariants to avoid object-specific programming. These invariants are kernels (i.e. multiplication of two gray values belonging to pixel at a certain distance) evaluated and summed over all angles or over all the shifts. Finally, they use support vector machines to classify.

Their database has a collection of 26 species of “pure” pollen particles for a total of 385 corpuscles. They assessed the performance with the technique “leave-one-out”, that is, they performed 385 tests on each pollen taking

for training the rest of images in the database. They achieved an error rate of 8%.

The main weak points of this work are:

- it is not feasible, a laser scanning microscope is very expensive and not suitable for a real-time and low-cost instrument
- the method is cumbersome because it requires a 3D analysis
- their database is very small and so their performance is not reliable
- “pure” pollen grains are much less variable in appearance and size than pollen particles captured by air sampler machine (see chapter 8 for an experimental prove).

4.5 Pollen identification with Paradise network

In [15], the authors describe two methods of pollen identification in images taken by optical microscope.

This work is one of the first attempts to automate the process of pollen identification using optical microscope. Indeed, several researches were done using scanning electron microscope (SEM) and good results were achieved, [32] [20] [24] [21]. But, SEM is expensive, slow and absolutely not suitable for a real-time application.

Their first method is a model based approach and it assumes that pollen grains have a “double edge” in the exine. Thus, a “snake” (a proper spline) is used to detect the presence of this edge. Since not all pollen grains show the double edge property and since the achieved performance is not so good we do not go in further details.

The second approach uses the so-called “Paradise Network”. Several small templates which are able to identify important features in the object, are generated and then linked together to form a certain number of patterns. This scomposition is repeated in test and the system looks for the best matching between the pollen and non-pollen patterns. The network is able to recognize nearly 80% of pollen as pollen and misclassify 4% of the debris as pollen.

This work is interesting, because they use images acquired with the common equipment, namely volumetric spore trap and the optical microscope. On the other hand, their performance is still not acceptable for any practical

application and is referred to a system that receives as input already segmented images of pollen or junk particles. Moreover, they limit their study only to pollen particles with a double edge in the exine.

4.6 Conclusion

In the previous chapter we explained the need for an automatic system for particle recognition as opposed to the manual analysis done by highly trained experts.

In this chapter, an overview was given on researches focused on automation of microscopic analysis.

In the urinalysis case, even if there is already a good system for automatic particle recognition, it can be still improved using segmentation free approach and trying to build a more flexible system able to extend the classification to new categories.

Pollen recognition is still manually done. The attempts to use SEM give good but useless results because no cheap, portable and real-time instrument can adopt this technology. On the other hand, no research was able to find good performance in detection and classification using optical microscope images of sticky tape pictures.

All these considerations justify our interest and research in automatic visual recognition of biological particles.

Chapter 5

Detector

Detection is the first stage of a recognition system. Given an input image with a lot of particles, it aims to find key points, which have at least a little probability to be a corpuscle of interest. Once that one of these points is detected, a patch is centered on it and then, it is passed to the classifier in the next stage. It is extremely important to detect almost all the objects of interest when they are rarely found in images and their number is low, as it typically happens in pollen detection.

In our research, we need to detect points only in pollen database. This database was labelled by human experts and a reference list has been built with information about genus and position of pollen particles sampled from air and now in images.

Thus, we evaluate performance by measuring how well our automatic detector agrees with the set of reference labels. A *detection* occurs if the algorithm indicates the presence of an object at a location where a pollen exists according to the reference list. Similarly, a *false alarm* occurs if the algorithm indicates the presence of an object at a location where no pollen exists according to the reference list [9].

The overall performance can be evaluated computing:

$$\begin{aligned}\text{perc. of detection} &= \frac{\text{nr. of detected interest particles}}{\text{nr. of particles of interest}} * 100 & (5.0) \\ \text{perc. of false alarms} &= \frac{\text{nr. of detected particles which are not of interest}}{\text{nr. of particles of interest}} * 100 & (5.0)\end{aligned}$$

More precisely, we consider a box centered on the detected interest point. We say that there is a matching between this box and the expert's one, if

$$\sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2} \leq \alpha \sqrt{\frac{A_1 + A_2}{2}} \quad (5.0)$$

where x_{ci}, y_{ci} are the coordinates of the box centroid, A_i is the area of the box and α is a parameter to be chosen. In our experiment, we chose α equal to $1/\sqrt{2}$. This value and parameter are the same ones we used in the software we developed to label pollen grains. In that situation, eq. (5) is used to establish when two expert's boxes overlap and if this condition is satisfied then, the software removes the old box because there is too much overlap and we interpret the old box as a mistake. Analogously for the detector, if the same condition holds, we say that there is a matching between the automatically detected box and the expert's one.

Typically, a detection system has some parameter to tune. When one parameter is varied then, also the false alarm and detection percentages change. Thus, by varying at each time a threshold, we can compute a sequence of percentages. The resulting curve is known as receiver operating characteristic (ROC) curve.

In the following sections, we will describe the detector we developed and we will assess its performance thanks to the percentages of detection and false alarms.

5.1 Outline

This detector is based on a filtering approach [22] [23] [18]. An outline of this system is shown in Figure 5.1 and Figure 5.2.

The input image is converted in gray level values. It is blurred and sampled at two different frequencies, F_1 and F_2 . In order to avoid artifacts near the image border, we normalize the background brightness to zero¹. We apply to both sampled images a filtering stage: for each such image we compute a difference of Gaussians [22] or the image suggested by Harris algorithm [18]. A set of interest points is extracted looking at the extrema of the two previously found images. Then, we discard points that are too close to the image border and for the remaining ones we find a proper bounding box. After removing those patches that are too small, too much overlapped with other patches and patches whose average brightness is too high, we have a set of patches that might contain a particle of interest and then, these patches will be given to the classifier.

¹The estimate of the background brightness is iterative and it is based on computation of the histogram peak of all the image pixel values.

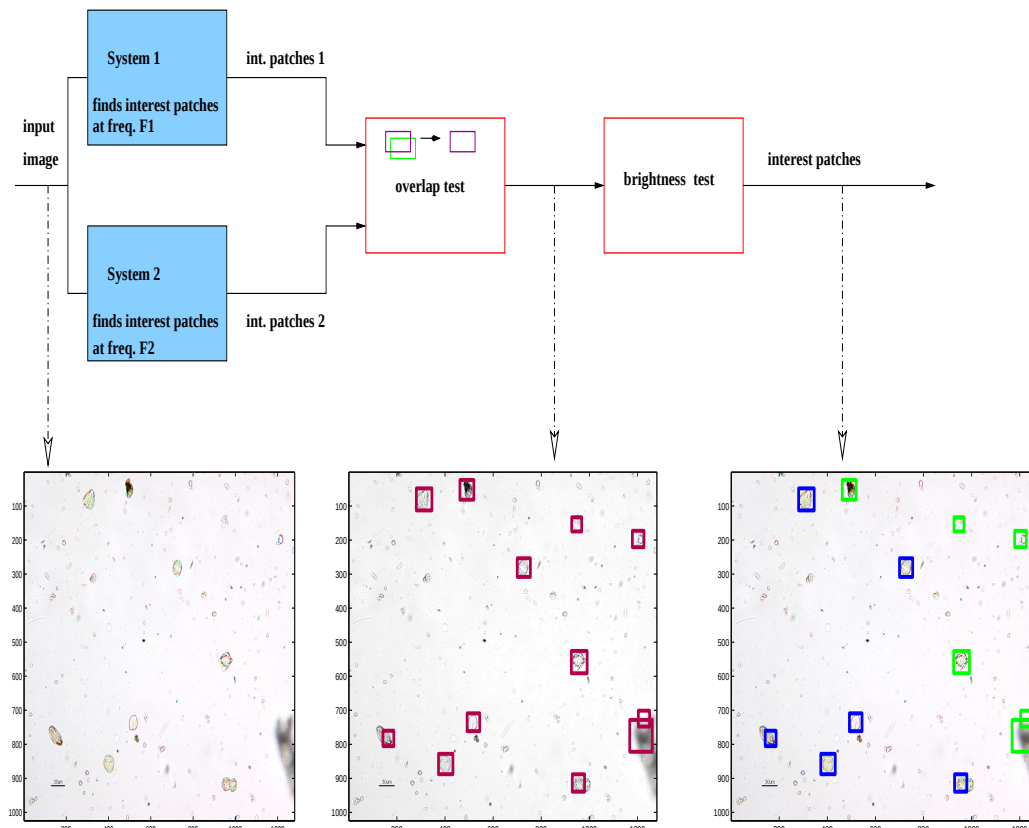


Figure 5.1: Outline of particle detector. Patches of interest are found by a system using two different frequencies (the light blue boxes that are described in Figure 5.2). Patches coming from these two systems are retained only if they pass a test on overlap and a test on average brightness (boxes with red border). For example, if two patches are too much overlapped we keep only one patch. If the average brightness of pixels falling inside the box is too high, we discard this patch because pollen particles have always some texture. The output is a set of patches that will be given to the classifier. In the example, the blue boxes are pollen grains, the green ones are false alarms.

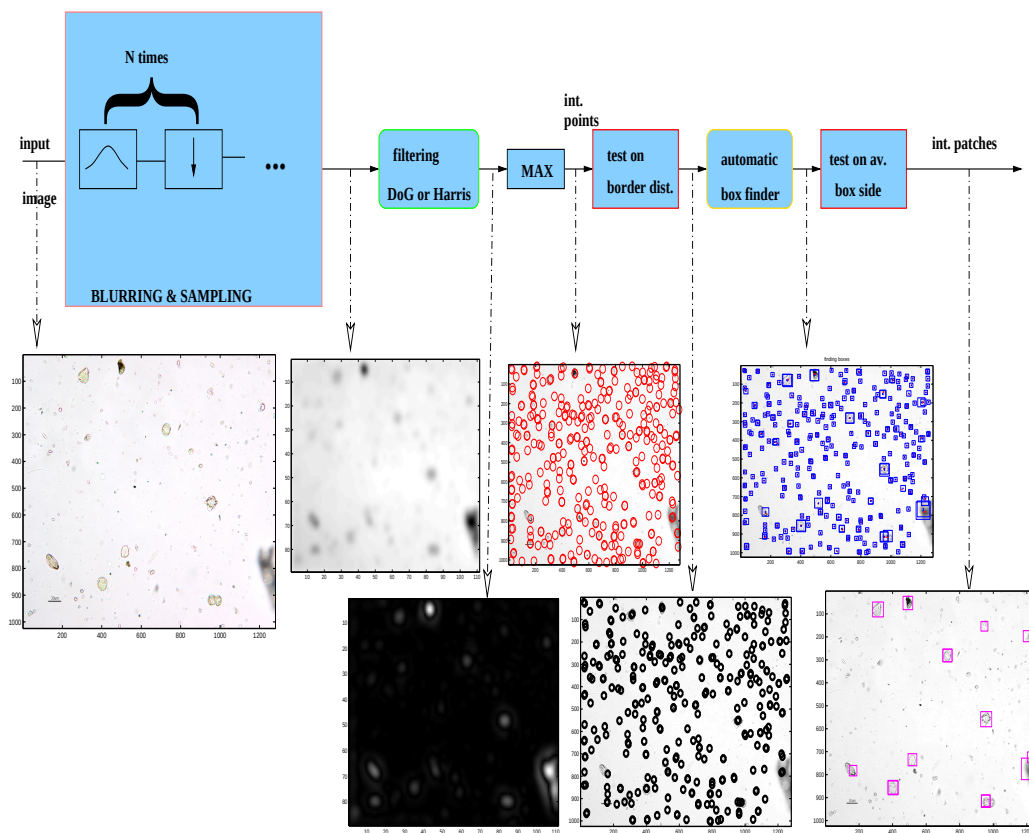


Figure 5.2: Zoom of the light blue boxes shown in Figure 5.1. The input image (after conversion to gray level values) is blurred and sampled at a certain frequency. Then, we look for the maxima of the difference of Gaussians (DoG) image or the maxima of the image computed according to Harris algorithm. These maxima are our initial set of interest points. Then, we discard points too close to the image border. After that, we automatically compute a bounding box around each interest point. Finally, we discard the patches whose average side is too small. The output is a set of patches that might contain pollen grains.

In the following subsections, we will explain in some details the key points of the system.

5.1.1 The difference of Gaussians image

Given a properly blurred and sampled image, we blur the image twice with a Gaussian kernel. The difference of Gaussians (DoG) image is the difference between the outputs of the two filters.

In Figure 5.3, we show the plots of Gaussian kernel and difference of Gaussian (DoG) kernel and their DFT in the 1-D case.

The goal of the sampling stage is to make the smallest pollen appear a little blob of few pixels. This automatically removes dust particles smaller than pollen grains and reduces the amount of computation. With a proper choice of the variance of the Gaussian kernel σ , the output image will have a peak in correspondence of round shaped objects with a certain size. Figure 5.4 shows a synthetic example in which two filters with DoG kernels having different σ are applied to an image. When σ is low, the output image has a narrow peak in correspondence of the little blob of the input image. When σ is high, the peak is in correspondence of the largest blob. The variance of the Gaussian kernel allows to select the size of the interest objects.

In order to detect how good a point of extremum is in the output image, we fit around it a paraboloid. The vertex of this paraboloid gives the exact position of the interest point and its curvature is a measure of its quality. If this value is too low then, it is likely that this extremum is generated by a strip shaped object or by an object that is smaller than the one we are looking for. If this value is high then the object in that position is round and it has a size that fits the size of our target.

5.1.2 Harris algorithm

Interest points can also be found considering the extrema points of another image.

Let be I the image and $I_x = \partial I / \partial x$ then, we define:

$$A = G(0, \sigma) \otimes I_x^2$$

$$B = G(0, \sigma) \otimes I_x I_y$$

$$C = G(0, \sigma) \otimes I_y^2$$

where $G(0, \sigma)$ is a zero-mean Gaussian kernel of variance σ^2 . The image we compute in order to find interest points is:

$$1/2 \mid (A + C - \sqrt{(A - C)^2 + 4B^2}) \mid$$

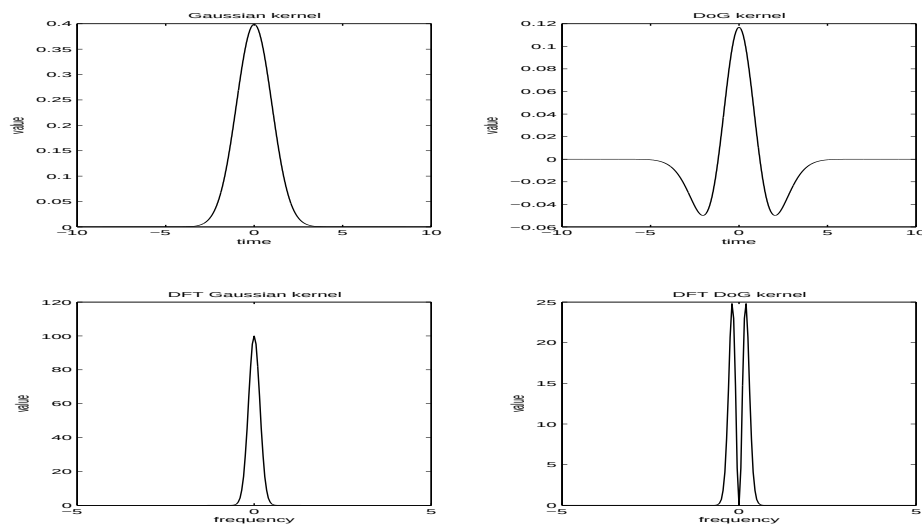


Figure 5.3: First row: basic kernels, namely Gaussian kernel and the equivalent kernel of the whole system. Second row: DFT of previous kernels.

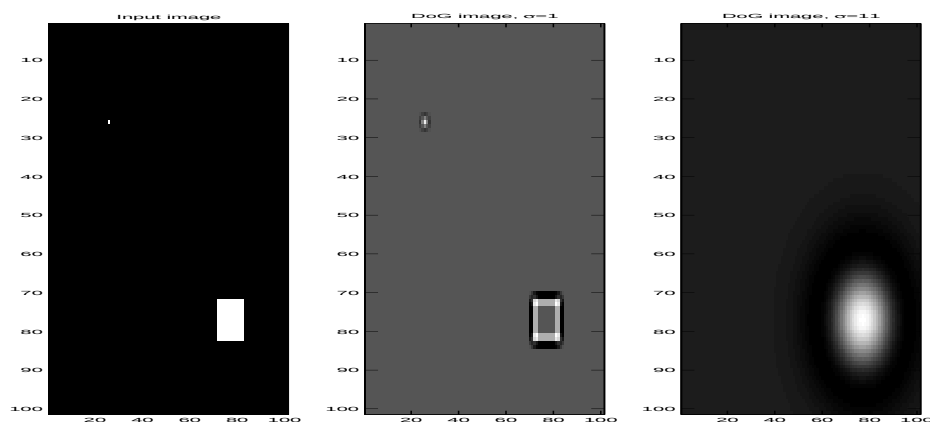


Figure 5.4: This example shows how the choice of scale σ is related with the dimension of objects of interest. The first image has a little blob of 1 pixel in the upper left corner and a larger one of 10 pixel diameter in the lower right corner. The second image shows a good peak in the position of the smaller blob, σ was chosen equal to 1. The third image shows a good peak in the position of the larger blob, σ was chosen equal to $8\sqrt{2}$.

that is the smallest eigenvalue of the matrix

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix}$$

Interest points are the extrema of this image because in these points there is the maximum variation of curvature along x and y directions.

5.1.3 Bounding box determination

The algorithm to determine a bounding box given an interest point is very straightforward. Its pseudocode is given in Figure 5.5 and in Figure 5.6 we show an example of application of this algorithm.

ALGORITHM for bounding box determination

Input: interest point inside particle

DO:

1. consider 150 pixel values starting from your interest point in direction: N, S, E, W, NW, NE, SE, SW; you will have 8 vectors $V_1 \dots V_8$
2. group the pixel values in each vector in set of three and compute their mean; you will have 8 vectors $v_1 \dots v_8$, each with 50 items
3. for each vector v_i , estimate the background brightness in that direction looking at the peak of the histogram of values contained in v_i
4. the border of the object is found when the mean squared error of 4 consecutive values in v_i with respect to the background brightness is less than a certain threshold; you will get 8 points around your interest point
5. fit a circle taking three points and try many combinations (i.e. points in N-NE-E, N-SW-E, etc.)
6. the optimal bounding circle is the one with minimum error e ; for each estimated circle, we compute the distances of the points not involved in the estimation (that are 5) to the circumference and we define e as the sum of the three smallest distances
7. we take the center of the circle as centroid and its diameter as side of a square bounding box.

Output: bounding box

Figure 5.5: Pseudocode of algorithm for determination of bounding box.

We consider 150 pixel values in each direction because pine pollen (the biggest one) can be bounded by a square box of 300 pixels. Then, we compute the mean of groups of three pixel values in order to make less computations

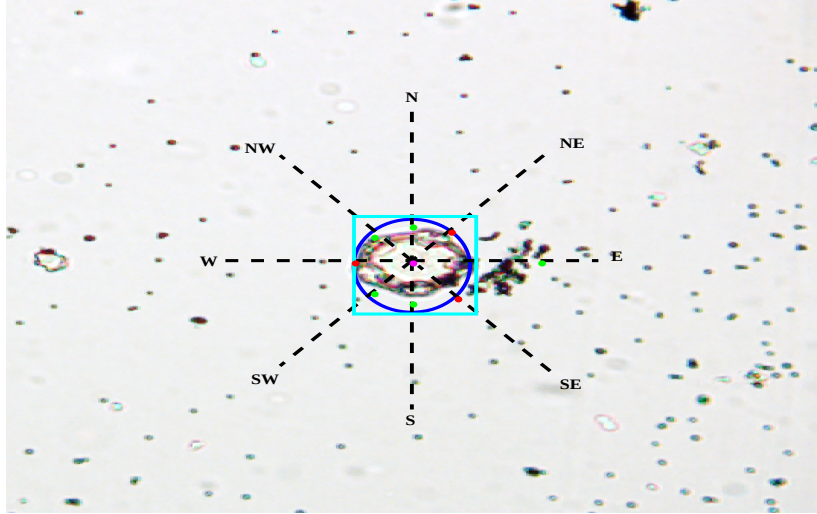


Figure 5.6: Example of bounding box detrermiantion. The point in magenta is the interest point. The green and red points are the estimated points around the particle border. The red points give the lowest error when fitting the circle. The resulting box is the cyan square.

and to reduce the noise. We consider 4 values of mean for the mean squared error (point 4) because we assume that outside the pollen the signal is stable around the background value and there are at least 12 pixels of separation among particles. Error e takes into account only three directions because we allow to have two wrong points around the border of the object (think about grains that are grouped together or see the dust near the pollen of Figure 5.6). On the other hand, we can be confident that the circle with minimum e describes quite well 6 points out of 8 staying in the object border.

5.1.4 Overlap criterion

When images are very noisy in the background or pollen grains are grouped together, there can be more than one bounding box for particle or there can be a lot of bounding boxes that are very close and overlapped. Then, we have to choose which bounding box to retain and which one to remove.

Our criterion focuses on two main cases. If box A is overlapped to box B² and no other box is overlapped with A and B then, we keep the box with minimum error e as defined in the previous subsection. If box A is overlapped to box B and both boxes A and B are overlapped to box C, then, we keep the box with the smallest value of e . Finally, if box A is overlapped to box

²According to the formula in eq.(5).

B, and only box B is overlapped to box C then, we keep box B if it has the smallest error e otherwise we remove only box B.

5.2 Examples

In order to have a better understanding of the computations performed by this system and to see its major limitations, we will take into account two images in our database. The first one is an example of very good detection (Figure 5.7, Figure 5.8, Figure 5.9), the latter one of poor detection (Figure 5.10, Figure 5.11, Figure 5.12).

Main causes of errors are:

- some pollen grains are very low contrast and they don't give any interest point
- pollen grains are sometimes grouped together and we can usually get only an interest point per group, the algorithm for bounding box determination can fail in this situation
- bounding box determination and overlap test can fail in very noisy images.

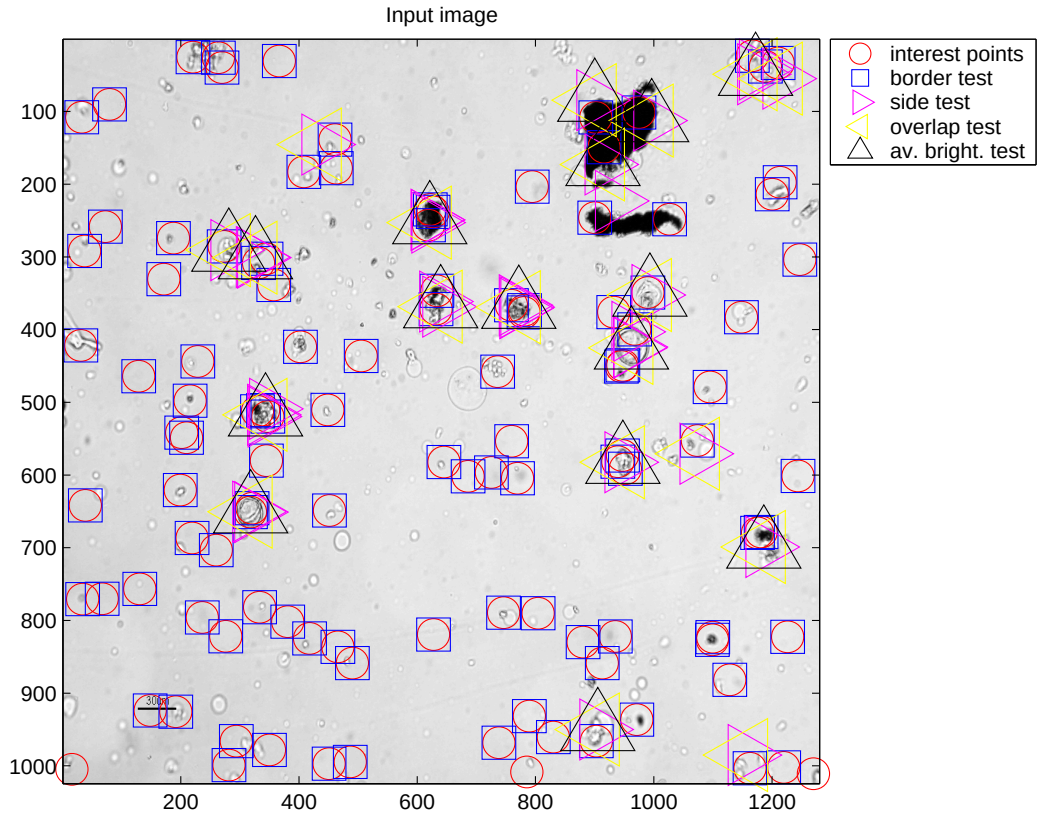


Figure 5.7: Example of good detection. The round red markers show the position of interest points. The square blue markers show the points that passed the test on the distance to the border. The magenta triangular markers are positioned in the centroids of those patches that have passed the test on the size length. The yellow triangular markers are located in the centroids of those patches that have passed the test on overlap. Finally, the black triangular markers show the position of the centroids of patches that passed the test on average brightness.

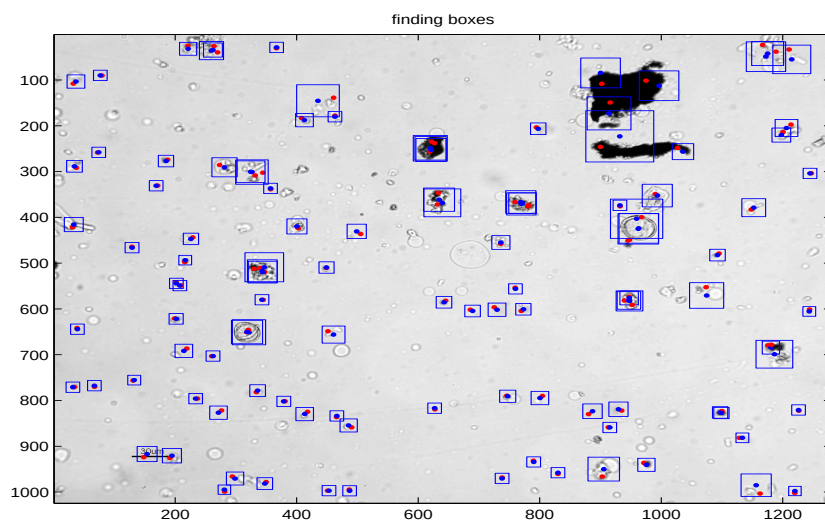


Figure 5.8: Example of good detection. The red points are the interest points that passed the test on border distance. The blue square boxes are the automatically computed bounding boxes. The blue point is the centroid of these bounding boxes.

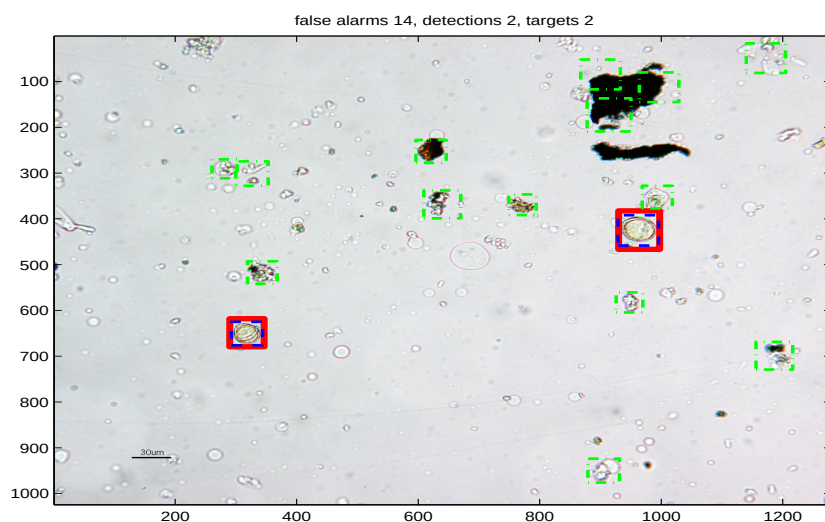


Figure 5.9: Example of good detection. The red boxes are dragged by the expert. The blue boxes show the patches computed by the system that match the expert's ones. The green boxes are patches computed by the system that are false positives. In this image, all pollen grains are detected and other 14 particles that are not pollen grains are detected.

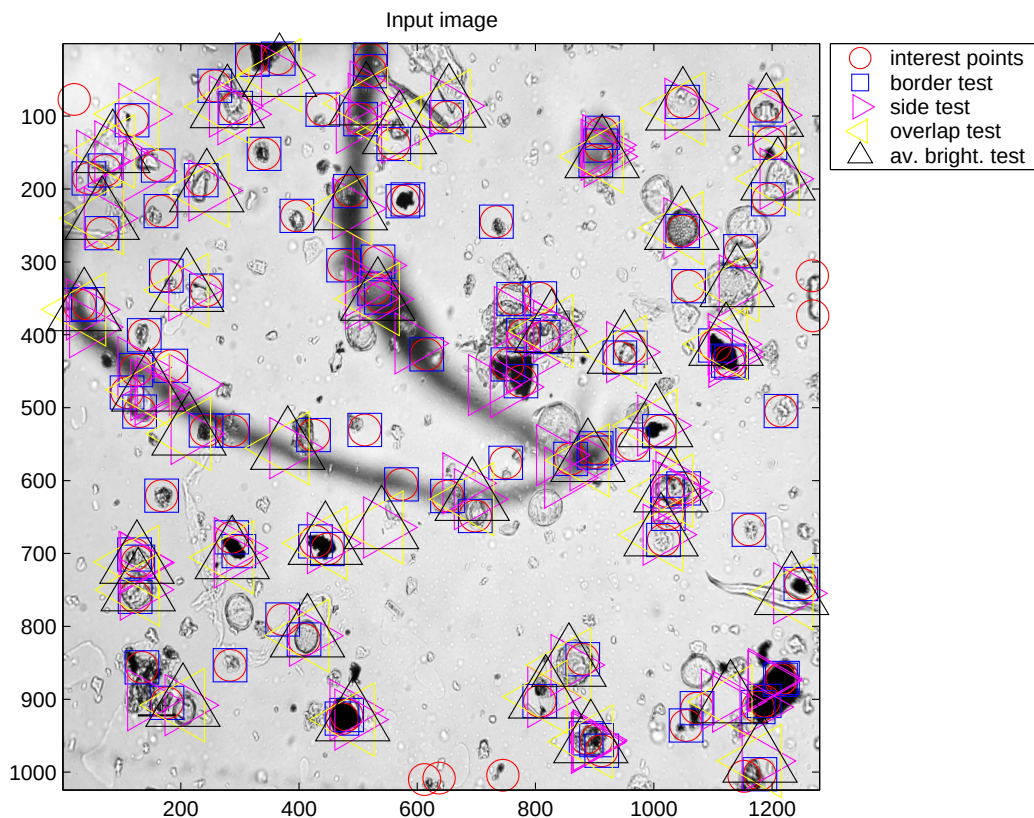


Figure 5.10: Example of bad detection. The round red markers show the position of interest points. The square blue markers show the points that passed the test on the distance to the border. The magenta triangular markers are positioned in the centroids of those patches that have passed the test on the size length. The yellow triangular markers are located in the centroids of those patches that have passed the test on overlap. Finally, the black triangular markers show the position of the centroids of patches that passed the test on average brightness.

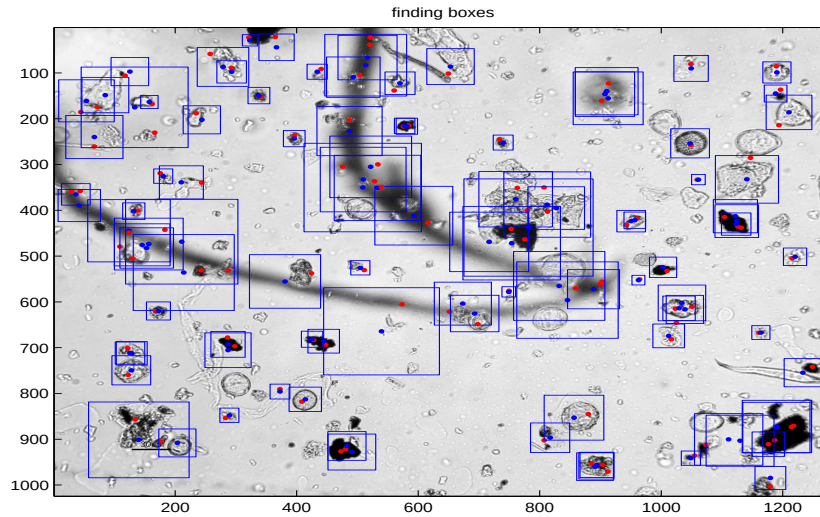


Figure 5.11: Example of bad detection. The red points are the interest points that passed the test on border distance. The blue square boxes are the automatically computed bounding boxes. The blue point is the centroid of these bounding boxes.

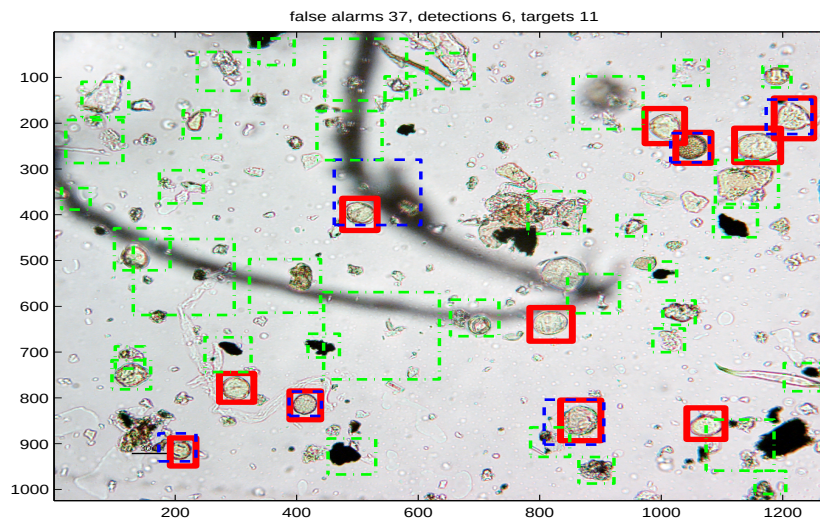


Figure 5.12: Example of bad detection. The red boxes are dragged by the expert. The blue boxes show the patches computed by the system that match the expert's ones. The green boxes are patches computed by the system that are false positives. In this image, there are 6 detections, 5 missed detections and 37 false alarms.

5.3 Experiments

We studied the performance of this detector using DoG images and images computed with Harris algorithm. For each method, we changed the values of the sample frequency and the variance of the Gaussian kernel. With the best achieved performance, we run many experiments varying the parameters MIN_K and MAX_ERR in the function that computes the interest points (see “*fit_round_paraboloid.m*”). Finally, we tried many values for the threshold used in the brightness test. Results are shown in table 5.1 and following ones. Experiments are done using the 100 images in the folder “042304_0 – 12” which contain 500 labeled pollen grains.

Sample freq.		DoG		Harris	
F_1	F_2	%det	%false al.	%det	%false al.
$4\sqrt{2}$	8	83.8	575	86.4	396
$4\sqrt{2}$	$8\sqrt{2}$	90.6	569	84.2	383
$4\sqrt{2}$	16	87.4	557	84.6	347
8	$8\sqrt{2}$	86.8	487	72.0	336
8	16	84.0	492	70.0	309
$8\sqrt{2}$	16	82.0	443	57.8	251

Table 5.1: Performance with σ equal to 1 for many values of the sample frequencies. We have kept MAX_ERR equal to 0.2, MIN_K equal to 0.1 and the threshold of brightness BR equal to 0.97.

Sample freq.		DoG		Harris	
F_1	F_2	%det	%false al.	%det	%false al.
$4\sqrt{2}$	8	90.0	583	88.0	513
$4\sqrt{2}$	$8\sqrt{2}$	89.2	579	88.4	494
$4\sqrt{2}$	16	87.0	559	86.6	464
8	$8\sqrt{2}$	90.0	519	80.8	437
8	16	87.8	500	79.6	412
$8\sqrt{2}$	16	84.8	456	71.2	339

Table 5.2: Performance with σ equal to $\sqrt{2}$ for many values of the sample frequencies. We have kept MAX_ERR equal to 0.2, MIN_K equal to 0.1 and the threshold of brightness BR equal to 0.97. The chosen values for the DoG detector are $F_1 = 8$, $F_2 = 8\sqrt{2}$, $\sigma = \sqrt{2}$.

Sample freq.		DoG		Harris	
F_1	F_2	%det	%false al.	%det	%false al.
$4\sqrt{2}$	8	91.0	571	90.0	538
$4\sqrt{2}$	$8\sqrt{2}$	90.2	568	88.2	523
$4\sqrt{2}$	16	87.0	540	85.8	487
8	$8\sqrt{2}$	88.4	509	85.8	457
8	16	87.4	490	85.0	425
$8\sqrt{2}$	16	81.0	432	73.8	348

Table 5.3: Performance with σ equal to 2 for many values of the sample frequencies. We have kept MAX_ERR equal to 0.2, MIN_K equal to 0.1 and the threshold of brightness BR equal to 0.97. The chosen parameters for the Harris detector are $F_1 = 4\sqrt{2}$, $F_2 = 8$, $\sigma = 2$.

Sample freq.		DoG		Harris	
F_1	F_2	%det	%false al.	%det	%false al.
$4\sqrt{2}$	8	62.4	437	76.0	298
$4\sqrt{2}$	$8\sqrt{2}$	75.2	436	73.4	287
$4\sqrt{2}$	16	58.2	394	71.2	272
8	$8\sqrt{2}$	74.0	391	57.0	201
8	16	74.0	396	57.2	189
$8\sqrt{2}$	16	71.0	291	35.2	131

Table 5.4: Performance with σ equal to $2\sqrt{2}$ for many values of the sample frequencies. We have kept MAX_ERR equal to 0.2, MIN_K equal to 0.1 and the threshold of brightness BR equal to 0.97.

Parameters		DoG		Harris	
<i>MIN_K</i>	<i>MAX_ERR</i>	%det	%false al.	%det	%false al.
0.05	0.1	79.0	373	78.2	376
	0.2	90.0	520	90.4	546
	0.3	92.4	566	90.8	575
	0.4	93.0	581	91.2	581
0.10	0.1	78.2	372	73.2	361
	0.2	90.0	519	90.0	538
	0.3	92.2	565	90.6	567
	0.4	92.8	580	91.0	573
0.20	0.1	64.2	334	25.0	191
	0.2	80.2	470	52.4	339
	0.3	83.6	517	59.8	375
	0.4	84.6	535	60.8	380
0.30	0.1	05.4	111	00.4	03.2
	0.2	11.4	166	03.4	23.0
	0.3	17.2	194	07.4	32.8
	0.4	19.0	210	08.4	35.8

Table 5.5: Performance with σ and F_i equal to the values written in bold characters in the previous tables, for many values of *MAX_ERR* and *MIN_K*. We have kept the threshold of brightness BR equal to 0.97. Based on these results, we have chosen *MIN_K* = 0.1 and *MAX_ERR* = 0.2.

Brightness thres.	DoG		Harris	
BR	%det	%false al.	%det	%false al.
0.95	87.6	436	85.8	458
0.96	89.4	481	88.4	500
0.97	90.0	519	90.0	538
0.98	90.2	551	90.2	579
0.99	90.6	583	90.8	609
1.00	90.8	609	90.8	631

Table 5.6: Performance with σ and F_i equal to the values written in bold characters in the previous tables, *MIN_K* = 0.1, *MAX_ERR* = 0.2 for many values of the threshold of brightness BR. Based on these results, we have chosen *BR* = 0.97.

5.3.1 Results on whole database

With the best parameters found in the previous section (see also tab.5.7), we run a final experiment on all the images of pollen database. The result is shown in tab.5.8. We are able to detect more than 90% of pollen particles with a percentage of false alarms around 1000% using the DoG image.

Note that the system is quite fast because it takes about 8 sec. to evaluate an image 1024x1280 pixels on machine working with a 2GHz processor.

Parameter	DoG	Harris
F_1	8	$4\sqrt{2}$
F_2	$8\sqrt{2}$	8
σ	$\sqrt{2}$	2
MAX_{ERR}	0.2	0.2
MIN_K	0.1	0.1
<i>Bright. thres.</i>	0.97	0.97

Table 5.7: Parameters of detector when using DoG image and Harris algorithm.

folder	images	nr. grains	DoG		Harris	
			%det	%false al.	%det	%false al.
*010403	195	959	98.0	515	95.6	650
022203	19	35	88.6	3030	94.3	3150
022303	10	11	100	3140	100	3245
022403	15	28	96.4	1610	96.4	1700
030310	22	56	92.9	1560	91.1	1730
030311	80	102	98.0	1820	93.1	2010
*032102	115	426	89.9	742	89.7	835
*042304_0-12	100	500	90.0	519	90.0	538
*042802	180	520	86.3	1236	88.1	1230
091903	30	66	93.9	1090	90.9	1180
120102	44	56	98.2	1560	98.2	1730
120202	8	12	91.7	2390	91.7	2720
*122902	158	273	98.2	2470	96.0	2620
*ChinElm	216	424	97.2	1000	95.1	1150
TrialPollen	71	235	96.6	950	93.6	1000
<i>TOTAL</i>	<i>1263</i>	<i>3703</i>	<i>93.9</i>	<i>995</i>	<i>91.8</i>	<i>1240</i>

Table 5.8: Detection performance on each folder of the current database. In the last row, the final result is given.

5.4 Analysis of results

We have already observed at the end of section 5.2 that the main causes of errors are due to dust particles that are attached or very close to pollen grains or to pollen particles that are grouped in a single blob. In these situations, we are not always able to have an interest point for each pollen grain and the algorithm to find a proper bounding box can fail.

Since our final goal is to classify the detected patches (see chapter 9), we have built a database with the detected patches (using DoG images) and we have sorted them according to the expert's reference list. Then, we have checked the patches belonging to the most numerous classes in order to discard the *bad* samples. Bad samples are produced by a mistake of our bounding box algorithm in the situations above mentioned. This procedure will give us a more reliable percentage of detection and it will allow us to run experiments of classification. In this way, we do not risk to consider as pollen particles some false alarm samples both in training and test stage.

According to this analysis, the new final results are shown in tab.5.9. Examples of good and bad patches for some of the most numerous classes are shown in Figure 5.13 and following.

genus	expert's	detected	% det.	detected good	% det.
alder	416	402	96.6	355	85.3
ash	329	323	98.8	303	92.1
chin. elm	522	506	96.9	466	89.3
jacaranda	282	259	91.8	223	79.1
mulberry	111	88	79.3	74	66.7
oak	413	384	93.0	329	79.7
olive	326	288	88.3	220	67.5
pine	705	694	98.4	606	86.0
<i>TOTAL</i>	<i>3104</i>	<i>2944</i>	<i>94.9</i>	<i>2576</i>	<i>83.0</i>

Table 5.9: Performance of detector using DoG images when considering the most numerous classes. The number of detected patches is given and also the percentage of detection with reference to the number of expert's patches for each class. The last two columns refer to the patches that can be reasonably classified.

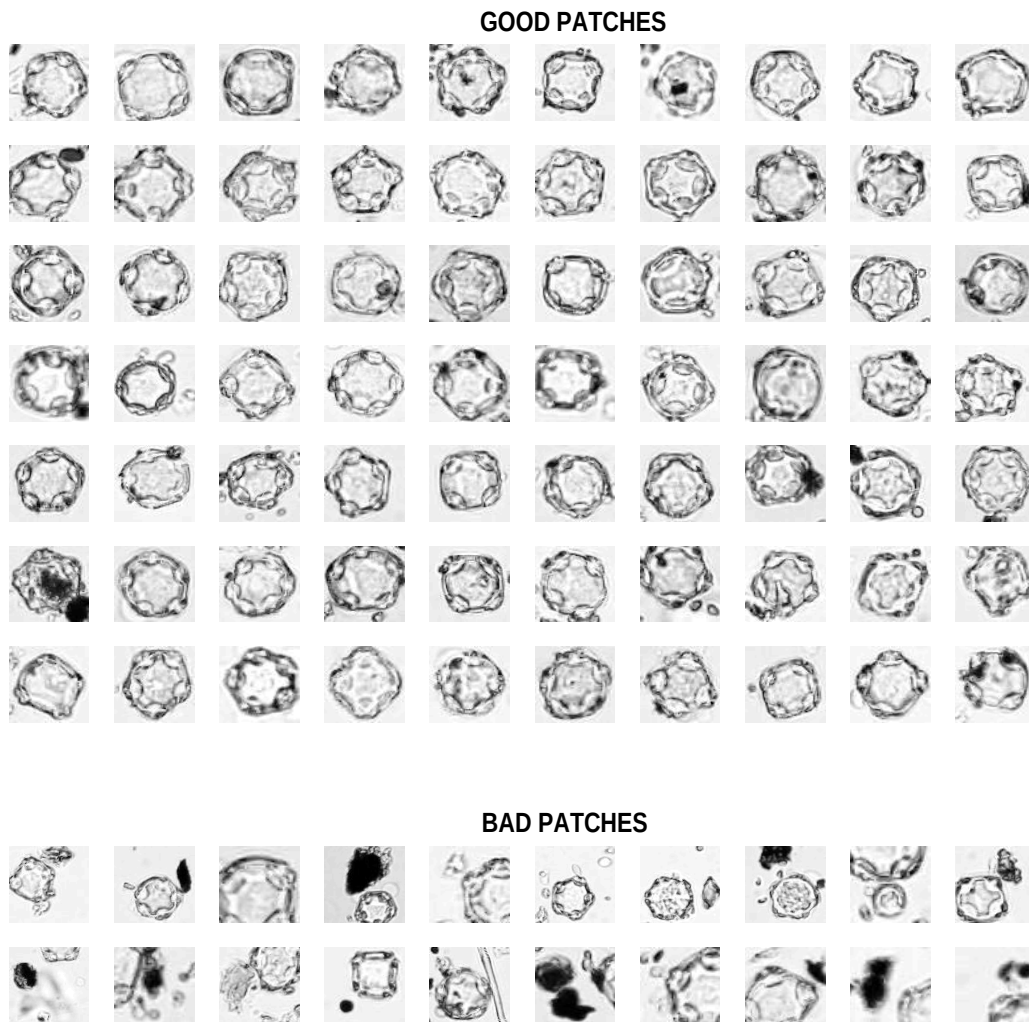


Figure 5.13: Example of 90 patches of alder pollen extracted by detector. The first 7 rows show examples of good segmentation. The last 2 rows show samples that have been removed because of a bad segmentation. Problems usually arise when there is junk attached to the pollen or when pollen grains are grouped in a single blob. Note that images have been rescaled to a common resolution in order to fit the grid.

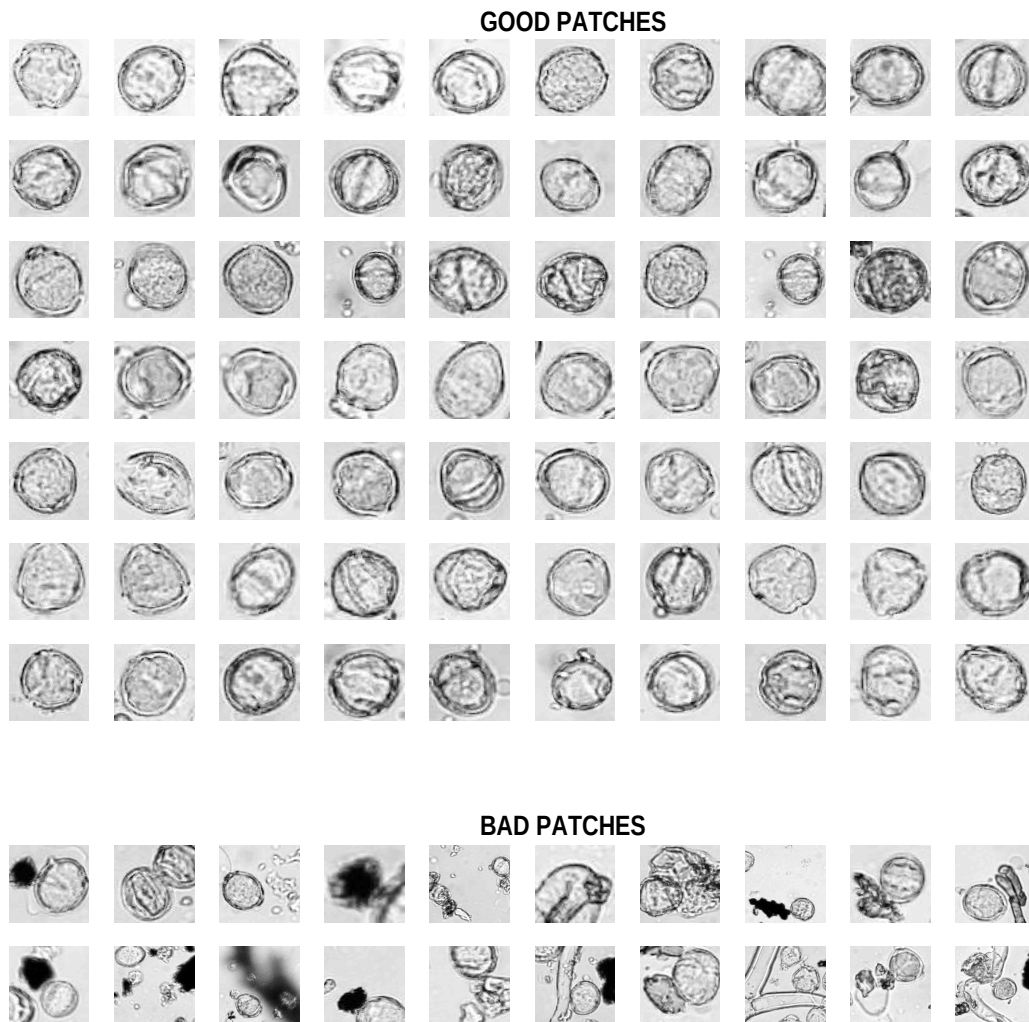


Figure 5.14: Example of 90 patches of jacaranda pollen extracted by detector. The first 7 rows show examples of good segmentation. The last 2 rows show samples that have been removed because of a bad segmentation. Problems usually arise when there is junk attached to the pollen or when pollen grains are grouped in a single blob. Note that images have been rescaled to a common resolution in order to fit the grid.

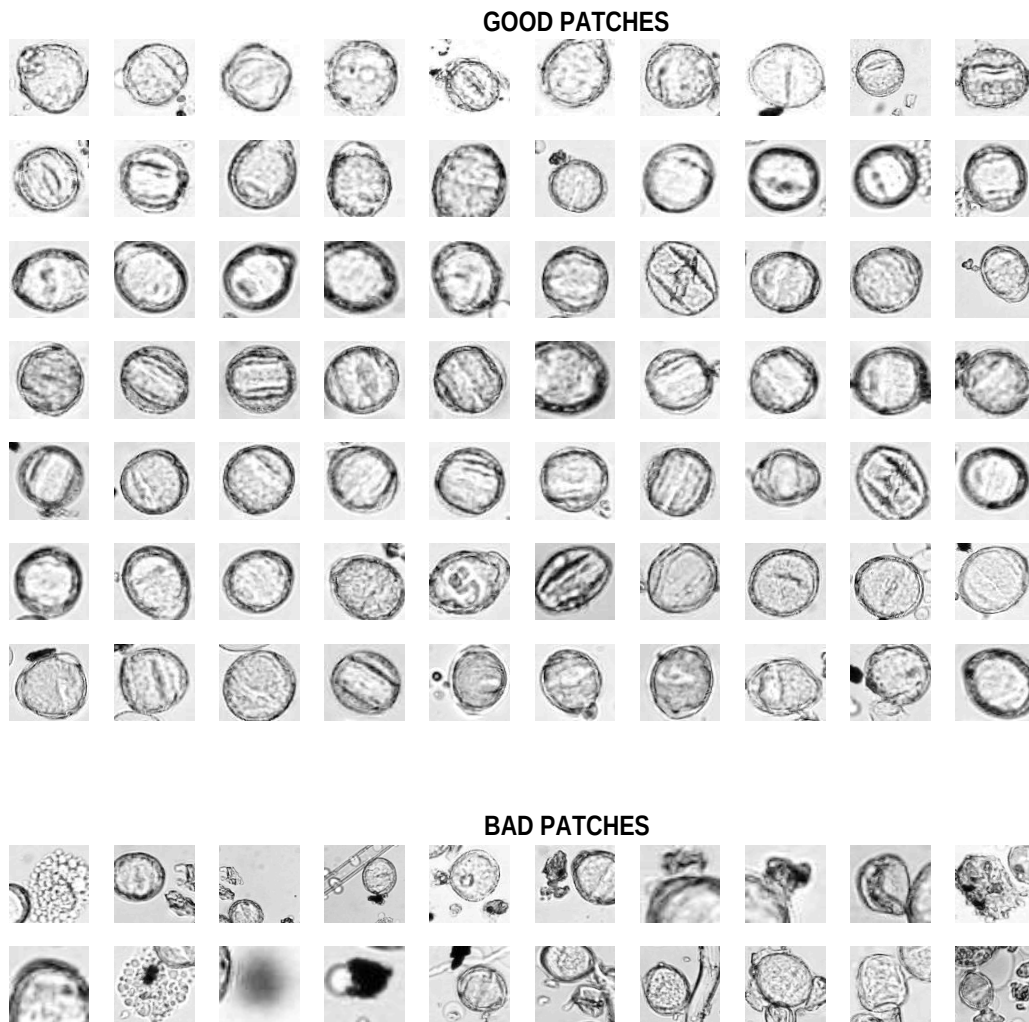


Figure 5.15: Example of 90 patches of oak pollen extracted by detector. The first 7 rows show examples of good segmentation. The last 2 rows show samples that have been removed because of a bad segmentation. Problems usually arise when there is junk attached to the pollen or when pollen grains are grouped in a single blob. Note that images have been rescaled to a common resolution in order to fit the grid.

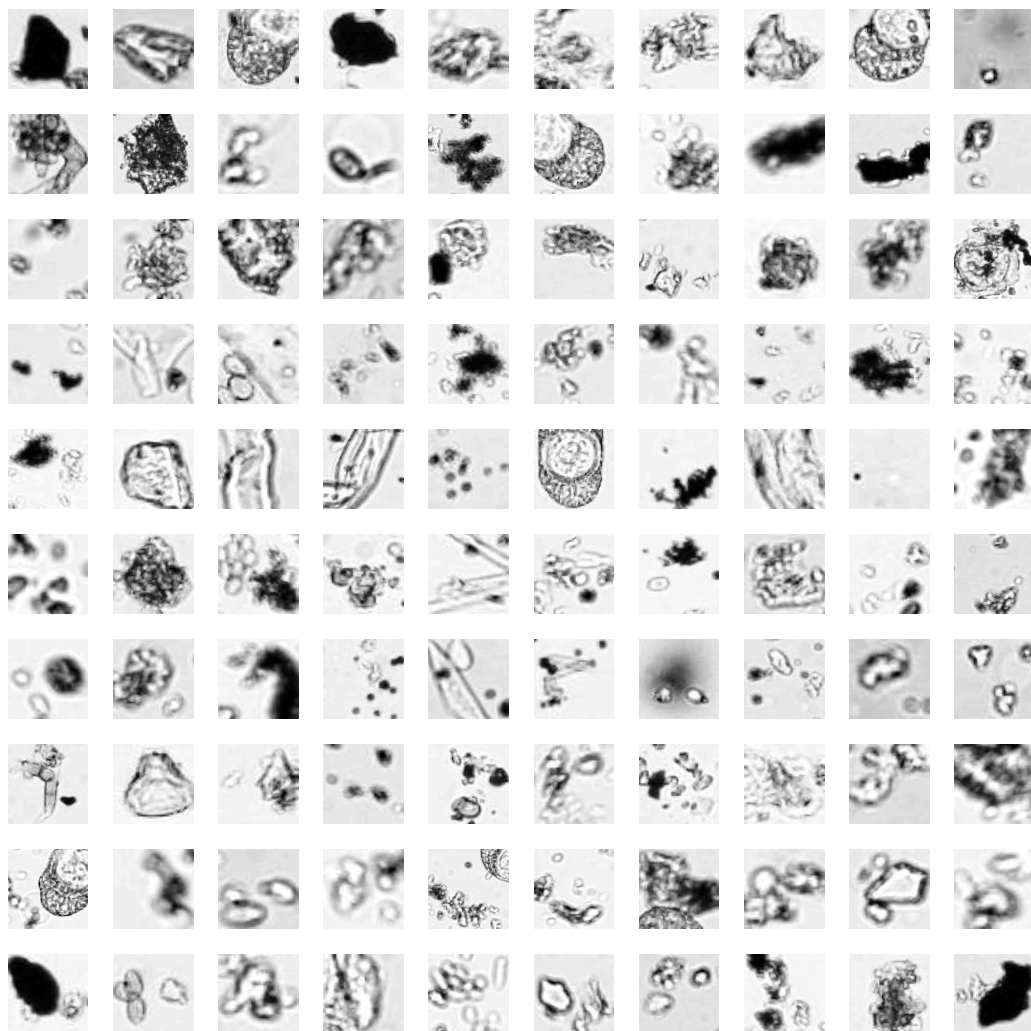


Figure 5.16: Example of 100 patches extracted by detector and belonging to the false alarm class.

Chapter 6

Feature extraction

In order to recognize an object in an image, it must be extracted some kind of measurements from it. These values should express the characteristics and all the information contained in the image. Moreover, we have to look for the smallest set of values with these good properties in order to speed up the recognition and to achieve more reliability. This combination of input data is called feature and it is based on some understanding of the particular problem being tackled or by some automatic procedures, [7]. It is the heart of the classification system. If the features extracted are not representative of input images, then, no classifier will be able to do a good job.

In this chapter, we describe two different kinds of feature. The first one, is based on the research of Schmid et al. [10]. The latter one is derived from a work of Torralba et al. [31], in which they aim to discriminate between scenes containing man-made objects and scenes with natural objects. In the next chapter, we will show the very good results of classifiers using these two sets of features.

6.1 Local Jets

In [10], the problem of matching an image in a large data set was addressed. In this method, from each interest point found in a image (using Harris' detector) a set of local gray-value invariants were computed. Our features are based on these invariants, originally studied by Koenderink et al. [19].

We want to extract from each image pixel a set of values, which are invariants with respect to shift and rotation. For example, we aim to acquire the same measurements from two images with the same rotated and shifted bacterium. Furthermore, we want to get the same values when the same object is at different distance from the camera, property referred as *scale*

invariance. Even though we are not interested in specific object recognition, the scale, shift and rotation invariance is important in our classification task. With this kind of features, we are able to cluster together the measurements of images belonging to a certain class because we acquire the same measurements, no matter the position and the orientation of the cell and these values are robust against little variations in size.

Let be $I(x, y)$ an image with spatial coordinates x and y . We define *local jet* of order N the convolution of image I with the derivatives with respect to N times

$(\overbrace{x, \dots, y, \dots}^N)$ of a Gaussian kernel with standard deviation σ . We indicate these local jets with $L_{i_1 \dots i_n}$, $i_k \in \{x, y\}$. The subscripts refer to the kind of Gaussian kernel derivation. Note that the convolution of I with a derivative of Gaussian kernel is equal to the convolution of Gaussian kernel with the same derivative of I .

In order to obtain invariance with respect to shifts and rotations, linear (or more precisely a differential) combinations of local jets are computed. Our set of invariants is limited to 9, that is to a combination of third order derivatives. These invariants are¹:

$$\begin{aligned}
I_0 &= I \\
I_1 &= L_x^2 + L_y^2 \\
I_2 &= 2L_x L_{xy} L_y + L_x^2 L_{xx} + L_y^2 L_{yy} \\
I_3 &= L_{xx} + L_{yy} \\
I_4 &= L_{xx}^2 + L_{yy}^2 + 2L_{xy}^2 \\
I_5 &= L_{xxx} L_y^3 - L_{yyy} L_x^3 + 4L_{xyy} L_x^2 L_y - 4L_{xxy} L_x L_y^2 \\
I_6 &= L_{xxy} L_y^3 - L_{xyy} L_x^3 + L_{xxy} L_x^2 L_y - L_{xyy} L_x L_y^2 \\
I_7 &= L_{xyy} L_y^3 - L_{xxy} L_x^3 + L_{xxx} L_y L_x^2 - L_{yyy} L_x L_y^2 + 2L_{xxy} L_x L_y^2 - 2L_{xyy} L_x^2 L_y \\
I_8 &= L_{xxx} L_x^3 + L_{yyy} L_y^3 + 3L_{xxy} L_x^2 L_y + 3L_{xyy} L_x L_y^2 \quad (6.-7)
\end{aligned}$$

In order to deal with scale changes, these 9 invariants are computed at different scales. At each stage a blurred image (with a Gaussian kernel) is considered as starting point to derive the invariant images. In our experiments, we considered 4 scales that is, we compute invariants using the original image, its blurred version, the blurred version of this last version and so on for another stage. Totally, given an image we build a stack with

¹The first invariant is the average luminance, the second one is the square of the gradient magnitude, the fourth one is the Laplacian.

$9 \times 4 = 36$ images. Figure 6.1 shows an example of this images of invariants using synthetic data: impulse (11x11 pixels) in first and second row, square and rhombus (100x100 pixels) in third, fourth and fifth, sixth row respectively; the gray level values are (approximately) proportional to the invariants values. Figure 6.2 shows all the 36 invariants for a bacterium image.

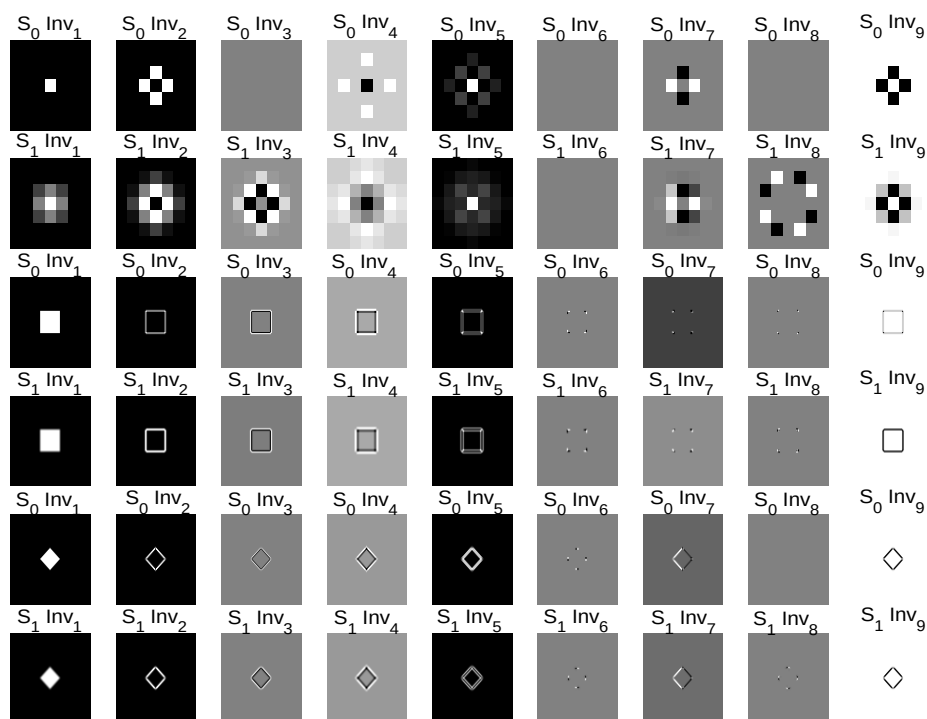
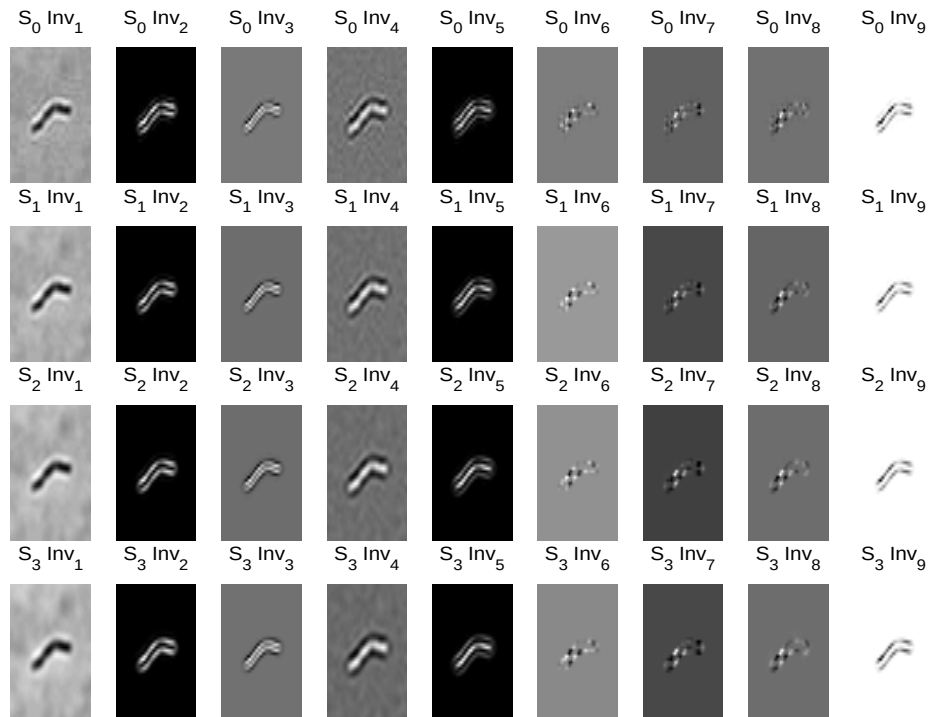


Figure 6.1: Invariants at scale 0 and 1 for synthetic data: impulse (11x11 pixel image), square and rhombus (100x100 pixel image). Note that some asymmetries are due to the approximation used to represent a value in gray-level and the rotation of the square in the rhombus is not perfect due to the poor resolution used.

Figure 6.2: Example of invariants at scale $[0, \dots, 3]$ for an image of bacterium.

6.1.1 Average and application of non-linearities

Till now, we have followed the same approach of [10] to extract from pixels feature vectors. In order to get a single feature vector from an image, we make an average taking features from each image pixel.

Just to have an idea of how much these features are able to separate our particles, consider Figure 6.3. Here, we have taken only 100 images from each of two pollen categories and we have extracted features with only two components (without non-linearity application). However, the separation between the data is already quite good.

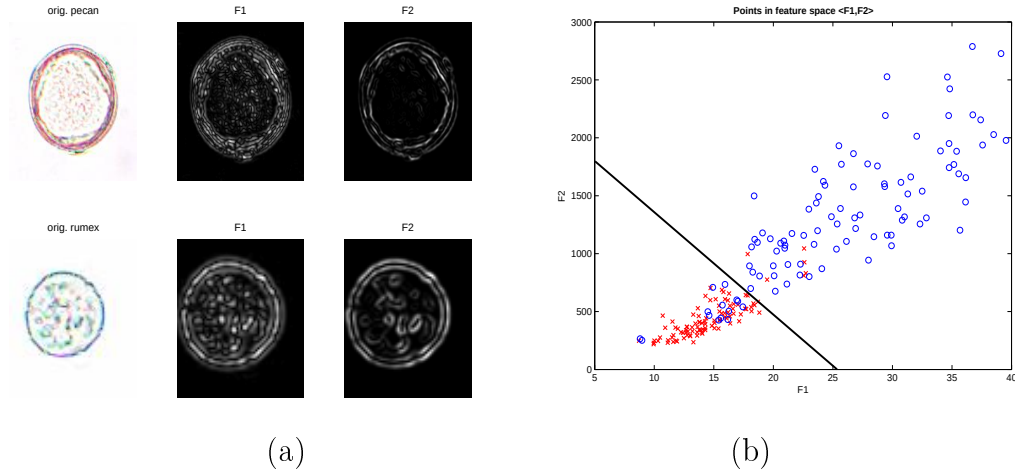


Figure 6.3: (a) Left column: two kinds of pollen grains, pecan and rumex. Central and right columns: processing on previous images in order to get from each image a feature vector with two components. (b) Example of classification: points in feature space for pecan and rumex pollen, cross and circle respectively. A linear classifier is able to reach an error rate of about 15%.

Furthermore, we have experimentally found that the performance is improved if some non-linearities are applied to the pixel features before performing the average. In this way, we aim to reduce the noise and to emphasize the signal. This can be interpreted as a way to do a “soft” masking avoiding any complication deriving from morphological operations. Every time a non-linearity is applied to each pixel feature then, we compute the above mentioned average to get the final image feature.

The first non-linearity we have studied is shown in Figure 6.4. When invariants take values in a huge range then, piece-wise linear functions are applied otherwise piece-wise quadratic functions. For a given dimension, the mean and the standard deviation of the background is estimated from a 10

pixel wide frame around the border of the image. The mean is subtracted to all invariants in that dimension and then, values between the mean and a proper fraction of the standard deviation are reset to zero while the other points are transformed. More precisely, only invarinats derived from the blurred image are transformed with a quadratic function.

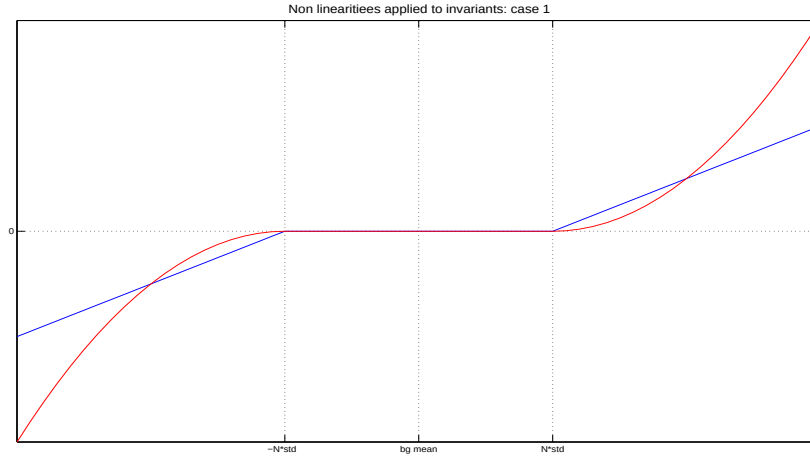


Figure 6.4: First non-linearity applied to invariants before the average: piece-wise linear and piece-wise quadratic transformation depending on the range of input invarinats.

Now, we describe a second option. The range of variation of the signal in each dimension is divided into three parts. Let be m the mean of the background, σ its standard deviation then, we can define:

$$A = m - N\sigma$$

$$B = m + N\sigma$$

where N is parameter to be chosen. Then, each component of each pixel feature produces three values that are:

$$f_l(x) = \begin{cases} x & \text{if } x < A \\ \frac{1}{2}(\cos(\frac{x-(A+B)/2}{(B+C)/2-(A+B)/2}\pi) + 1)x & \text{if } A \leq x \leq m \\ 0 & \text{if } x > m \end{cases}$$

$$f_h(x) = \begin{cases} x & \text{if } x > B \\ \frac{1}{2}(-\cos(\frac{x-(B+C)/2}{(C+D)/2-(B+C)/2}\pi) + 1)x & \text{if } m \leq x \leq B \\ 0 & \text{if } x < m \end{cases}$$

$$f_m(x) = x - f_l(x) - f_h(x)$$

Figure 6.5 shows the mapping functions.

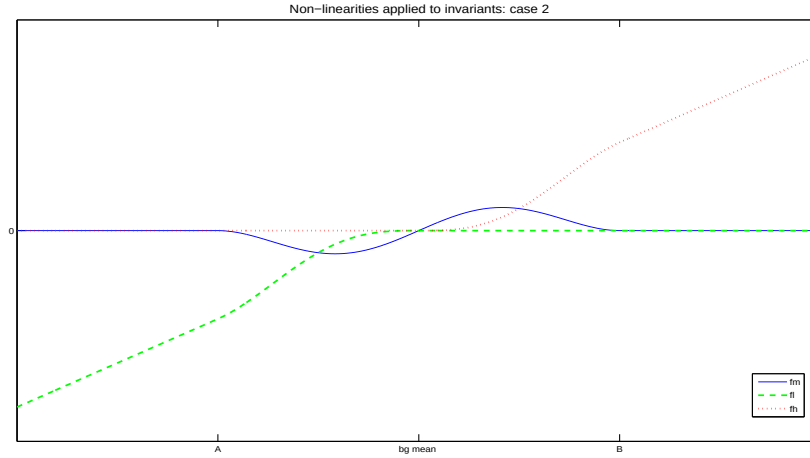


Figure 6.5: Second non-linearity applied to invariants before the average: f_l is sensitive to low values, f_m to values around the background mean, f_h is sensitive to high values.

Finally, we describe the last transformation. With respect to the estimated mean value of the background, each invariant is split into three parts: positive, negative and absolute value. In order to avoid any loss of information we add to this last component the background mean. The mapping functions are shown in Figure 6.6.

Note that the second and third non-linearities increase the dimension of the feature space from 36 to 108.

In order to assess the performance of these features we have done some experiments on urine database. We have considered a classifier modeling the training data with a mixture of Gaussians (see par. 7.1). Each time we have considered a different set of features:

1. the ones that are computed simply averaging invariants without any non-linearity
2. the ones that are computed applying the first non-linearity
3. the ones that are computed applying the second non-linearity

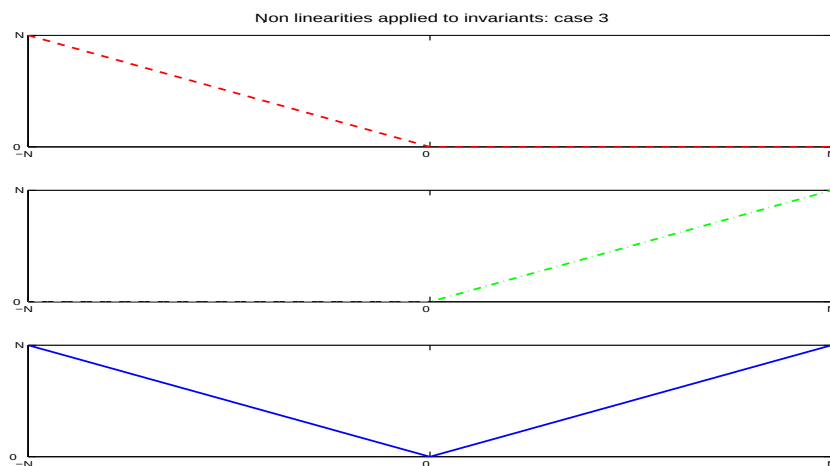


Figure 6.6: Third non-linearity applied to invariants before the average: each plane is split into positive, negative and absolute value with respect to the background mean.

4. the ones that are computed applying the third non-linearity

For each case, we have found the optimal parameters running experiments for many different values. The results are summarized in tab. 6.1.

Since the third non-linearity has given the best performance, we will refer to this kind of non-linearity in the following paragraphs.

feature	parameters	error rate
1) averaged invariants		
D	18	18
MoG	4 full	
2) 1st non-lin.		
N	0.5	16.3
D	8	
MoG	4 full	
3) 2nd non-lin.		
N	0.5	11.3
D	18	
MoG	3 diag.	
4) 3rd non-lin.		
D	12	10.8
MoG	4 diag.	

Table 6.1: Parameters and performance of classifier using the different kind of features described in this section. D is the number of dimension of the feature space. MoG says how many components are in the mixture and the structure of their covariance matrix. N is the number of standard deviation (of the background brightness) that are used to compute a certain feature (only for 1st and 2nd non-linearity). We have run experiments for $D \in (8...20)$, $N \in (0.1, 0.5, 1)$, covariance structure spherical, diagonal and full. We trained on 480 images and tested on 20 images. The number of components in the MoG was chosen in order to have at least three samples per parameter.

6.1.2 Decreasing the dimensionality

In training, we aim to model the distribution of feature vectors belonging to all the images of a certain class with a mixture of Gaussians (MoG) or with support vector machines (SVM) (see par. 7.1, 7.4). However, it is not possible to work in a 108 dimensional feature space because the number of parameters grows very quickly with the space dimensions. For example, in Table 6.2 you can find these relations as a function of the choice of covariance matrix structure and as a function of the number of components in MoG.

	SPHERICAL	DIAGONAL	FULL
<i>nr. param.</i>	$N(D + 1) + N - 1$	$2ND + N - 1$	$N(D + \frac{D(D+1)}{2}) + N - 1$

Table 6.2: Number of parameters in MoG using N components in a D dimensional feature space as a function of the covariance matrix structure.

For instance, if $D = 108$ and $N = 2$ then, we should estimate 220, 433, 11989 parameters in accordance to the chosen covariance matrix structure. On the other hand, a common rule of thumb says that you need at least 3 or 4 points to estimate a single parameter. Since our database has not a so huge number of images we have to reduce the dimension of feature space.

In order to reduce the feature space dimensionality we could apply PCA [17] [7]. Experimentally, we found that linear discriminants analysis works better than PCA. Particularly, given the whole set of training feature vectors we compute Fisher's Linear Discriminants (FLD) [7] [25].

We now introduce this technique. Suppose for simplicity to have data $\mathbf{x} \in \mathbb{R}^D$ belonging only to two classes C_1 and C_2 . We want to find the vector $\mathbf{w}$ such that the projected data

$$y_n = \mathbf{w}^T \mathbf{x}_n \quad (6.-7)$$

are optimally separated. The mean vectors of the two classes are

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \mathbf{x}_n \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \mathbf{x}_n \quad (6.-7)$$

where N_i is the number of items in class i -th. Then, we define

$$m_k = \mathbf{w}^T \mathbf{m}_k$$

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2 \quad (6.-7)$$

Fisher's discriminants maximize a function which is in the two-class problem,

$$\frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \quad (6.-7)$$

that is, we are trying to maximize the distance between the means of the projected clusters and minimize the spread of the projected data. While PCA seeks the sub-space that best *represents* the data, FLD seeks the sub-space that maximizes the *separability* between the classes.

The generalization of FLD to several classes follows the same reasoning. It turns out that if you have K classes you can have no more than $K - 1$ directions where to project the data. For example, if we think to work with the 12 categories of urine database, we should have the constraint to reduce the feature space dimension from 108 to less than 12. In order to reduce the feature space dimensions without any constraint, we split randomly the data of each class in a proper number of sub-categories ².

6.2 Image and Power Spectrum Principal Components

We describe a feature extraction based on principal component analysis (PCA), [17] [7].

Given data in a N dimensional space, the goal of PCA is to find a D dimensional subspace such that the projected data are the closest in L_2 norm (mean square error) to the original data. This means that we are looking for the subspace which gives the best *representation* of the original data. This subspace is spanned by the eigenvectors of the data covariance matrix having the highest corresponding eigenvalues. Often the full covariance matrix can not be reliably estimated from the number of example available, but the approximate highest eigenvalue basis vectors can be computed using singular value decomposition (SVD). Thus, if Σ is the covariance matrix of the data, its SVD decomposition is

²We tried also to divide each cluster applying *k-means* algorithm [7] and estimating a MoG but without any meaningful improvement.

$$\Sigma \stackrel{svd}{=} USV^T \quad (6.-7)$$

where if $\Sigma \in \mathbb{R}^{N \times N}$, then $U \in \mathbb{R}^{N \times N}$, $V \in \mathbb{R}^{N \times N}$ and $UU^T = I$, $VV^T = I$. S is a diagonal matrix with the elements on the diagonal (the singular values) in descending order. The first columns of U are the eigenvectors with the highest eigenvalues and can be used as basis for the original data.

In our specific case, let be $i(x, y)$ the intensity distribution of the image along spatial variables x and y with $x, y \in [1, N]$. Let be $\mathbf{i}$ the values of image pixels rearranged in a column vector, then the covariance matrix is $\Sigma = E[(\mathbf{i} - \mathbf{m})(\mathbf{i} - \mathbf{m})^T]$ with $\mathbf{m} = E[\mathbf{i}]$. If we call image principal components IPC_n the eigenvectors of Σ (computed by singular value decomposition) and $IPC_n(x, y)$ the same eigenvectors reshaped into a matrix $N \times N$, then we can write the following decomposition,

$$i(x, y) = \sum_{n=1}^P v_n IPC_n(x, y) \quad (6.-7)$$

with $P \leq N^2$ and v_n the coefficients used for describing the image $i(x, y)$ in this new basis. Obviously, v_n is the projection of the image along the direction given by IPC_n .

Figure 6.7 shows 36 IPCs (with the highest corresponding eigenvalues) computed from the first 500 images of each class belonging to the urine database. In our experiments we used always less than 10 IPCs and you can note from Figure 6.8 that the first components are the most important in the data representation. Before any computation, we subtract the background mean estimated from the border of the image (a frame 10 pixels wide) in order to normalize the data with respect to the background brightness.

If we compute the discrete Fourier transform (DFT) $I(k_x, k_y)$ of an image, we can approximate its power spectrum with the squared magnitude of I :

$$R(k_x, k_y) = |I(k_x, k_y)|^2 \quad (6.-7)$$

with

$$I(k_x, k_y) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} i(x, y) e^{-j \frac{2\pi}{N} (xk_x + yk_y)} \quad (6.-7)$$

and $f_x = k_x/N$, $f_y = k_y/N$ spatial frequencies. The power spectrum encodes the energy density for each spatial frequency and orientation over the whole image. PCA applied to power spectra gives the main components

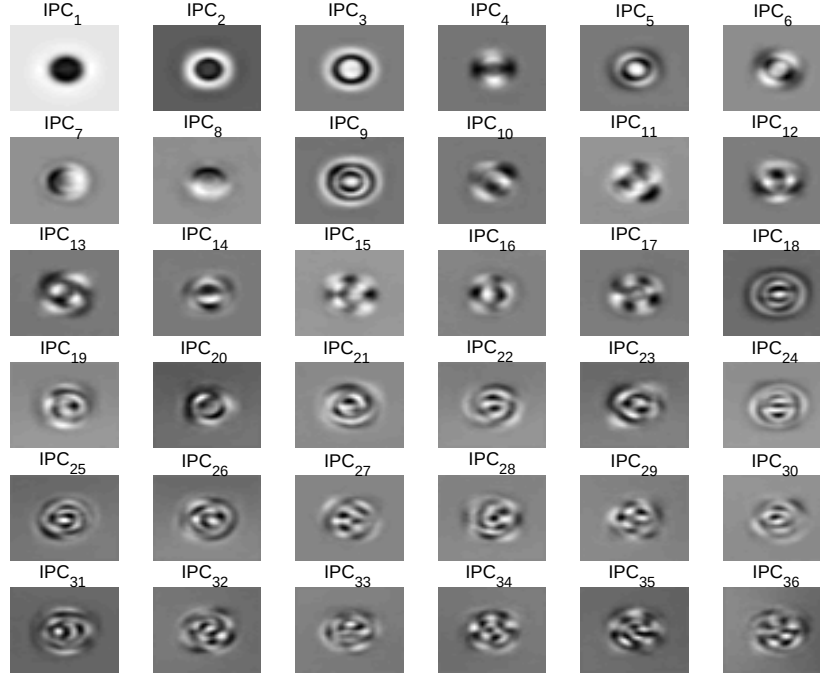


Figure 6.7: First IPCs derived collecting 500 images from each class in urine database. Each image was normalized to a common resolution of 40x40 pixels, thus the space dimension is 1600 and there are 1600 IPCs and singular values.

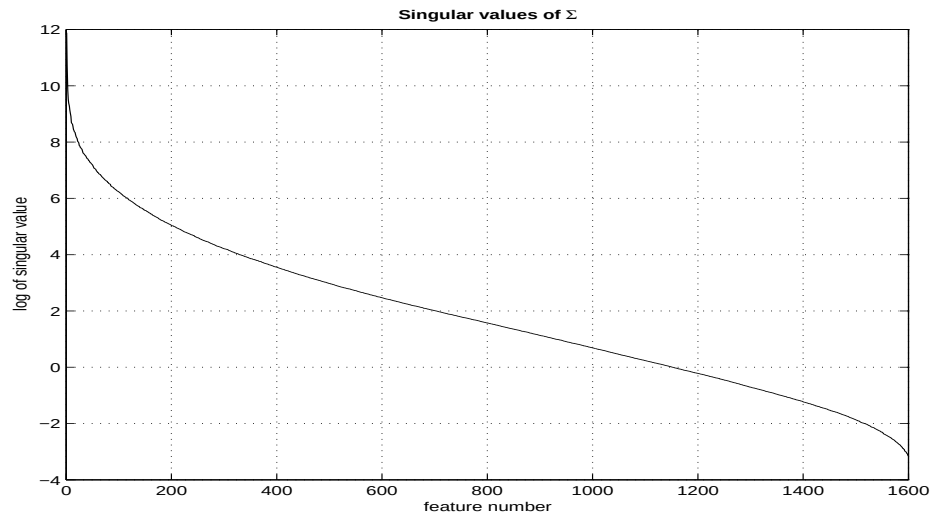


Figure 6.8: The singular values corresponding to IPCs indicate the importance of each component in representing the data.

that take into account the structural variability between images.

First, we normalize the power spectrum with respect to its variance for each spatial frequency: $R^*(k_x, k_y) = R(k_x, k_y) / \text{std}[R(k_x, k_y)]$, with $\text{std}[R(k_x, k_y)] = \sqrt{E[(R(k_x, k_y) - E[R(k_x, k_y)])^2]}$. Then, we apply PCA as done before for the image, we find the spectral principal components (SPCs) and we decompose the normalized power spectrum in

$$R^*(k_x, k_y) = \sum_{n=1}^P u_n \text{SPC}_n(k_x, k_y) \quad (6.-7)$$

Figure 6.9 shows the SPCs (with the highest corresponding eigenvalues) computed from the first 500 images of each class belonging to the urine database.

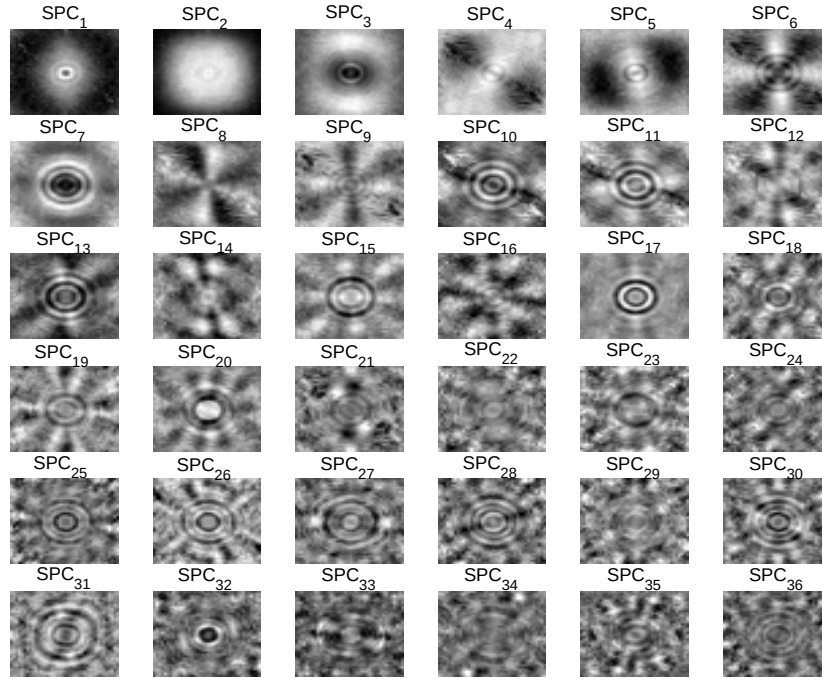


Figure 6.9: First SPCs derived collecting 500 images from each class in urine database. Each image was normalized to a common resolution of 40x40 pixels.

Till now, we have described the basic approach as presented in [31]. In our research, we added a new step.

After the normalization of the image with respect to the background mean that makes this mean equal to zero, we split the image in three derived images: the first one is its positive part I^+ , the second one its negative part

I^- and the third one the absolute value I^a . This operation is equivalent to the application of three non-linear functions that are selective of high, low and mid-range responses. Then, we apply for each of these derived images the PCA and we get IPC_+ , IPC_- , IPC_a . A set of these principal components are shown in Figure 6.10 and are computed from the same data used to calculate the basis shown in Figure 6.7.

It is experimentally proved that the classification is improved when this splitting is done; evidently, particles belonging to different categories behave differently with respect to this operation. Note that the split in positive, negative and absolute value parts can be done very efficiently. We are adding complexity only in training stage because we double the number of PCA required (from two to four).

With this trick, we decrease the error rate from nearly 19% to 16% with reference to the classification of urine database.

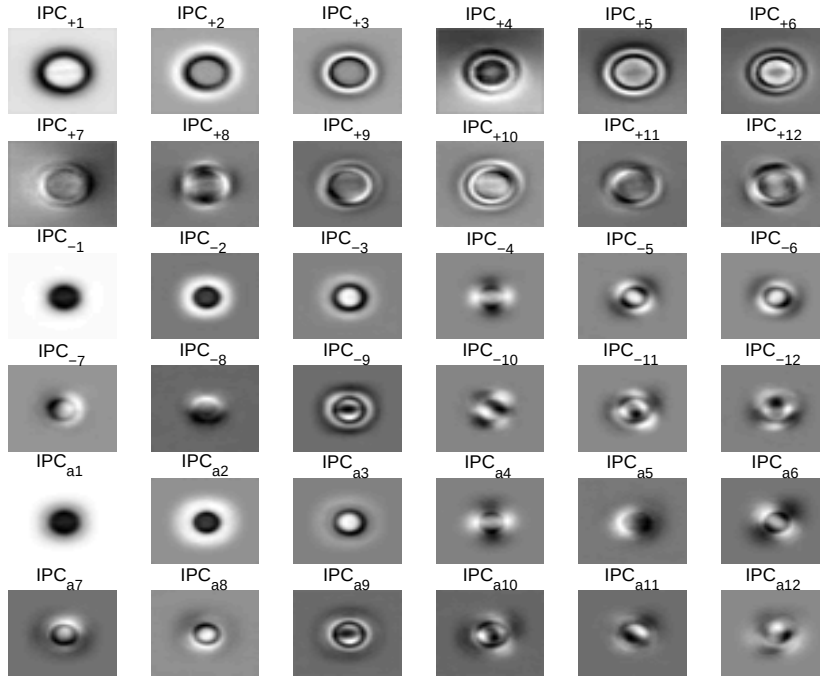


Figure 6.10: First 12 IPC_+ , IPC_- , IPC_a derived collecting 500 images from each class in urine database. Each image was normalized to a common resolution of 40x40 pixels.

If we consider the first N_1^i, N_2^i, N_3^i, N^s coefficients computed by the projection of I^+ along IPC_+ , I^- along IPC_- , I^a along IPC_a and R^* along SPC we get a feature vector from each image

$$\mathbf{f} = [v_{11}, \dots, v_{1N_1^i}, v_{21}, \dots, v_{2N_2^i}, v_{31}, \dots, v_{3N_3^i}, u_1, \dots, u_{N^s}] \quad (6.-7)$$

where v_{ij} is the projection of the image I^i along j -th IPC_i for $i \in \{+, -, a\}$.

How do we choose values for N_1^i, N_2^i, N_3^i, N^s ? In preliminary experiments, we found that the nearly same performance can be achieved considering

$$\begin{aligned} N_1^i &= N_2^i = N_3^i \\ N^s &= 3N_j^i, \quad \text{for } j = 1, 2, 3 \end{aligned} \quad (6.-7)$$

and applying Fisher's linear discriminants to the whole set of principal components in order to reduce the dimension of feature space to $N = N_1^i + N_2^i + N_3^i + N^s$. In the next chapter, we will only consider the first option because of its simplicity.

6.3 Interpretation

The two sets of features described in this chapter are similar, in spite of the different techniques used to compute them. Features based on PCA give the average information content of the signal at frequencies higher and higher when it is considered the projections on eigenvectors with smaller and smaller eigenvalues. Similarly, features based on invariants give the global amount of information at different bands when it is taken the average of these invariants at different scales. This is proved experimentally. The implementation of classifiers based on these features skipping the non-linearity application gave nearly the same performance, with an error rate of 20% on the twelve categories of urine database.

The application of non-linearities boosts the performance; in particular, it nearly halves the error rate of classifier using the features based on local jets.

Chapter 7

Classification

Classification is the last stage of a recognition system. Given a patch with - hopefully - one centered object, a feature vector is extracted from the image. In training, the system learns how to distinguish among features of images belonging to different categories. In test phase, according to the test image feature, a decision is made. If the object in the image is considered a particle of interest, then its class is also identified, otherwise it is discarded because the measurement does not match any training model.

This chapter begins with the description of training and test stage of a Bayesian classifier using Gaussian mixture models. This classifier gave the best performance with an error rate of 7.6% on urine database and 10.2% on “pure” pollen database.

However, following sections present slightly different approaches. First, two ways of combining both kinds of features already presented in the previous chapter will be discussed. Second, a customized version of AdaBoost.M1, an algorithm to boost the performance of any classifier, will be defined. Finally, a classifier based on support vector machines (SVM) will be introduced.

This chapter will conclude with experiments to optimize the system and some final experiments covering all the data set.

7.1 Mixture of Gaussians

This classifier is the simplest and the most powerful system we developed. With the only exception of the classifier using SVM, all the other classifiers are based on it.

We now describe its two main stages: training and test.

7.1.1 Training

In training phase, we can model the distribution of feature vectors belonging to a certain class with a mixture of Gaussians (MoG) [7]. If the number of categories is K , then we have to estimate K MoGs. The estimation is done applying expectation maximization (EM) algorithm [7].

Generally, our aim is to estimate the *class-conditional distribution* $f(\mathbf{x} | C_k)$, that is the probability density function (pdf) of the data, given our knowledge that they belong to a certain class C_k . In order to simplify the notation we now make implicit the class condition. However, it is a straightforward generalization to consider it.

The mixture model decomposes the pdf of the data $f(\mathbf{x})$ in a linear combination of M component densities $f(\mathbf{x} | j)$ in the form

$$f(\mathbf{x}) = \sum_{j=1}^M f(\mathbf{x} | j)P(j) \quad (7.0)$$

The mixing parameters $P(j)$, also called *priors probabilities*, satisfy the constraints

$$\sum_{j=1}^M P(j) = 1 \quad (7.1)$$

$$0 \leq P(j) \leq 1 \quad (7.2)$$

moreover, the conditional probability density functions verify

$$\int f(\mathbf{x} | j) d\mathbf{x} = 1 \quad (7.2)$$

An important property of mixture models is that they can approximate any continuous pdf to arbitrary accuracy with a proper choice of number of components and parameters of the model.

In our system, we use mixture of Gaussians. Thus, we have to choose the number of components and the structure of their covariance matrix. The best choice of parameters is found experimentally. We make use of a package for MATLAB called NETLAB in which there are functions to estimate the MoG by the application of EM algorithm.

The goal of training is to build for each class a model of the distribution of the feature vectors belonging to the class images. This model is achieved using MoG and it describes the pdf of the class feature vectors.

7.1.2 Test

Given a test image, we compute its feature vector $\mathbf{f}$. Then, we apply the Bayes' rule.

We are interested on the probability of the class C_k given the test data $P[C_k | \mathbf{x}]$, because we want to apply the following decision rule:

$$x \mapsto r(x) = \operatorname{argmax}_k P[C_k | \mathbf{x}] \quad (7.2)$$

The extension of Bayes' rule to continuous conditions assure that

$$P[C_k | \mathbf{x}] = \frac{f(\mathbf{x} | C_k)P[C_k]}{f(\mathbf{x})} = \frac{f(\mathbf{x} | C_k)P[C_k]}{\sum_{j=1}^M f(\mathbf{x} | C_j)P[C_j]} \quad (7.2)$$

where f is pdf and P is probability. So, in hypothesis of equal distribution for the class probabilities, $P[C_k] = 1/K$ for $k = 1, \dots, K$, we can simplify

$$P[C_k | \mathbf{x}] = \frac{f(\mathbf{x} | C_k)}{\sum_{j=1}^M f(\mathbf{x} | C_j)} \quad (7.2)$$

Moreover, we can observe that $\max P[C_k | \mathbf{x}]$ is obtained for the same k of $\max f(\mathbf{x} | C_k)$ because the factor $\sum_{j=1}^M f(\mathbf{x} | C_j)$ is constant for all j . The term $f(\mathbf{x} | C_k)$ is called *likelihood* because it is a conditional probability density function with implicit dependence on the model parameters. Because our decision rule looks for the maximum of this term, it is referred as *maximum likelihood* decision rule.

We also add to this framework a test. When we train, we collect for each class its resolution mean and standard deviation, m_{r_k} and σ_k . Then in test, we compute the likelihood of the test data given the class C_k only if the following condition holds:

$$r \in [m_{r_k} - N\sigma_k, m_{r_k} + N\sigma_k] \quad (7.2)$$

where, r is the resolution of the test image and N is a parameter to be chosen; in our experiments we chose $N = 3$.

This test is mainly done to save time avoiding useless computation. If an image resolution is out of the range of every class, then it is not classified and put in an *unknown* class.

7.2 Combination of features

We will show in the experimental part that the previous classifier using features based on principal component analysis and on local jets gives good

results with an error rate below 15% on urine database. In this section, we present two methods to combine these features.

If you look at the confusion matrices of the classifier working with these features (see Figure 7.14 and Figure 7.15), you can notice a certain correlation among misclassifications which suggests dependence between the two kinds of feature.

However, we can simplify the problem of feature combination assuming *independence* between the two sets. If we do not make any hypothesis between these features we have to solve the more general problem of combination of experts' response.

7.2.1 Independence hypothesis

Given an image, we apply the two techniques described in the previous chapter to extract features based on principal component analysis $\mathbf{x}_1$ and on local jets $\mathbf{x}_2$. In hypothesis of independence between these two vectors, we can write the class conditional density function as

$$f(\mathbf{x}_1, \mathbf{x}_2 | C_i) = f(\mathbf{x}_1 | C_i)f(\mathbf{x}_2 | C_i) \quad (7.2)$$

Thus, as we did in the previous section, we look for the maximum likelihood of the test image feature to make a decision and thanks to eq. (7.2.1), we can write

$$P[C_i | \mathbf{x}_1, \mathbf{x}_2] \text{ is proportional to } f(\mathbf{x}_1 | C_i)f(\mathbf{x}_2 | C_i) \quad (7.2)$$

To summarize, with reference to the classifier using MoG we make two separate trainings using the two set of features and two separate tests. Then, we multiply the class conditional densities and we choose the class with highest value.

7.2.2 Without hypothesis

This approach was suggested by the good performance achieved modeling data with MoG. We now do not make any assumption on features.

We divide the training set into two sets. The first one is used to train separately the classifier using the two different kinds of features and so to get $f(\mathbf{x}_i | C_j, \theta_n)$, where $\mathbf{x}$ is a feature vector and θ_n is the model used. We refer to model as the technique for feature extraction, namely the method based on PCA or local jets. Then, for each image of the second training set we compute the vector

$$(P[C_1 \mid data, \theta_{pca}], \dots, P[C_1 \mid data, \theta_{lj}], \dots) \quad (7.2)$$

and we model with a MoG these probability vectors of all the images belonging to each class.

In test, given an image we compute the two feature vectors and the vector of probabilities of eq. (7.2.2). Then, we make a decision by computing the maximum likelihood of this vector in the last training model.

In this way, we take into account the performance of each classifier and implicitly we give more weight to the “expert’s” response with better performance in the given class (even if in test we do not know the true class to which the particle belongs).

The drawback of this method, is that it requires a lot of training data because it needs to build MoG models. Because our data set is quite small, the parameter estimation cannot be very precise and thus, the performance is not improved (see the experimental section).

7.3 Boosting

We studied the multi-class extension of AdaBoost called “AdaBoost.M1”, [29] [16].

Given a weak learner, let be X the set of input data and Y the set of the possible labels for the samples in X . The learner receives examples (x_i, y_i) , with $x_i \in X$ and $y_i \in Y$. These examples are chosen randomly according to some fixed but unknown distribution P on $X \times Y$. The goal is to learn to predict the label y given an instance x . We assume that the sequence of N training examples is $(x_1, y_1), \dots, (x_N, y_N)$. At each iteration or round, the weak learner generates hypothesis which assigns to each instance one of the K possible labels. At the end, the boosting algorithm presents an output hypothesis $h_f : X \rightarrow Y$ with lower error rate. It is required that each weak hypothesis has prediction error less than $1/2^1$.

We define for any predicate p , $\vdash p \dashv$ to be 1 if p holds and 0 otherwise.

We summarize the general algorithm in pseudo-code in Figure 7.1.

Let’s try to boost the performance of the classifier described in par. 7.1 with reference to the previous algorithm.

We have to decide how to provide our classifier with the distribution $\mathbf{p}$ at each round. We could go inside EM algorithm and modify the weights of the sum

¹The error-correcting output code technique [28] has not this constraint but it is slightly more difficult. Our classifier has an error rate below 15% and so, this constraint is not a problem.

ALGORITHM AdaBoost.M1**Input:**

- sequence of N 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
- weak learner algorithm
- integer T specifying the number of iterations

Initialize the weight vector: $w_i^1 = D(i)$ for $i = 1, \dots, N$.**DO for** $t = 1, 2, \dots, T$

1. Set

$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$

2. Call weak learner, providing it with the distribution $\mathbf{p}^t$; get back the hypothesis $h_t : X \rightarrow Y$.
3. Calculate the error of $h_t : e_t = \sum_{i=1}^N p_i^t \vdash h_t(x_i) \neq y_i \dashv$.
4. Set $\beta_t = \frac{e_t}{1-e_t}$.
5. Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta_t^{1 \dashv h_t(x_i) \neq y_i \dashv}$$

Output the hypothesis

$$h_f(x) = \arg \max_{y \in Y} \sum_{t=1}^T \left(\log \frac{1}{\beta_t} \right) \vdash h_t(x_i) = y_i \dashv$$

Figure 7.1: Pseudocode of algorithm AdaBoost.M1.

in the log-likelihood computation. However, it seems simpler to extract from each class set a number of images proportional to the class weight. Note that in this way, $\mathbf{p}$ becomes a distribution over the classes and not any more over the single training images. This method has also another drawback because the MoG estimation needs a lot of examples to find convergence. So, we have to reduce the number of training images belonging the best classified categories and on the other hand, to keep a sufficient number for the MoG estimate. Thus, our implementation of AdaBoost.M1 is a slightly different version.

We choose the weights in a way that allows to the worst classified class to train on all the maximum number of images while the best classified class has available a minor number of images; however this number is sufficient for the estimate of parameters in the MoG. In particular, at each round the minimum weight is decreased if the class is still the best classified, while the maximum weight is always equal to the maximum if the class is still the worst classified. We show the pseudo-code of this customized version of AdaBoost.M1 in Figure 7.2.

In the experimental section, we will show that this method is able to achieve a slightly better performance than the one of the basic classifier.

ALGORITHM AdaBoost.M1 customized version**Input:**

- sequence of $N = M * K$ 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 l

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

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

1. Call classifier, providing it with w_k^t images for each class; get back the hypothesis $h_t : X \rightarrow Y$.
2. Calculate the overall error rate e_t .
3. Set $\beta_t = \frac{e_t}{1-e_t}$.
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

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

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 $l \times mw$ and MW .

Output the hypothesis

$$h_f(x) = \arg \max_{y \in Y} \sum_{t=1}^T \left(\log \frac{1}{\beta_t} \right) \vdash h_t(x_i) = y_i \dashv$$

Figure 7.2: Pseudocode of modified version of algorithm AdaBoost.M1.

7.4 Support Vector Machines

The support vector machine (SVM) approach does not attempt to model the distribution of features but it works directly on them.

Let us start considering the simplest case of a *linear machine* trained on *separable data*, [11] [8]. The training input data are a set of labeled features $\{\mathbf{x}_i, y_i\}$, $i = 1, \dots, l$, $y_i \in \{-1, 1\}$, $\mathbf{x}_i \in \mathbb{R}^d$. Suppose we have some hyperplane which separates the positive from the negative examples. The points $\mathbf{x}$ which lie on the hyperplane satisfy $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is the normal to the hyperplane. Let be d_+ and d_- the shortest distance from the separating hyperplane to the closest positive and negative example respectively. For the linearly separable case, the support vector algorithm looks for the separating hyperplane with the largest *margin*, defined as $d_+ + d_-$. Thus we can write that each training point satisfy:

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b &\geq +1, & \text{for } y_i = +1 \\ \mathbf{x}_i \cdot \mathbf{w} + b &\leq -1, & \text{for } y_i = -1 \end{aligned} \quad (7.2)$$

that is,

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \quad (7.2)$$

Choosing a proper scale for $\mathbf{w}$ and b , we can have points for which the equality in eq. (7.4) holds and $d_+, d_- = 1/\|\mathbf{w}\|$; these points are called *support vectors*. Their removal would change the solution and they live in the hyperplanes $H_1 : \mathbf{x}_i \cdot \mathbf{w} + b = +1, H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$. No training point falls between them.

It can be shown that in order to find $\mathbf{w}$ and b , we have to solve a quadratic optimization problem. Once these parameters are found, given a test point $\mathbf{x}$, we simply determine on which side of the decision boundary $\mathbf{x}$ lies and assign the corresponding class label.

If the training data are not separable, we have to introduce positive *slack* variables ξ_i , $i = 1, \dots, l$ in the previous equations which become:

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b &\geq +1 - \xi_i, & \text{for } y_i = +1 \\ \mathbf{x}_i \cdot \mathbf{w} + b &\leq -1 + \xi_i, & \text{for } y_i = -1 \\ \xi_i &\geq 0 \quad \forall i \end{aligned} \quad (7.1)$$

Thus, for an error to occur, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound on the number of training errors. It can be shown [8] that the quadratic programming problem can be solved introducing a new user defined parameter C in order to assign the desired penalty to errors. An interpretation of C and support vectors is that only support vectors exert a force on the decision sheet and C is an upper bound on this force.

Because a support vector machine only realizes a linear classifier, non-separable data are projected in a very high dimensional feature space in which the data become separable, and this means that in the original feature space the classifier becomes highly non-linear.

The only way in which training data appear in the optimization problem is in the form of dot products, $\mathbf{x}_i \cdot \mathbf{x}_j$. Then, we can map the data in another (generally higher dimensional space) H using the function Φ :

$$\Phi : \mathbb{R}^d \mapsto H \quad (7.1)$$

Now, the training depends only on the data through dot products in H , i.e. on $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Moreover, we are interested to find a *kernel function* K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ in order to use only a function. The most powerful and simple kernels are:

- **linear:** $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- **polynomial** of order p : $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p$
- **Gaussian radial basis function** (RBF) of parameter σ : $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$

Note that the dimension of space H is $\frac{(D+p-1)!}{p!(D-1)!}$ and ∞ when using the polynomial and the RBF kernel respectively. During the tuning stage, we have only to choose good values of two parameters: the penalty C and the degree p of the polynomial kernel or the width σ of the RBF kernel. In Figure 7.3, you find a simple example of binary classification using these kernels for different values of their parameters.

Finally, we have to discuss how to use this algorithm to solve a multiclass classification problem [13]²: we use *error-correcting output codes* technique. Each class is assigned a unique binary string of length n called *codeword*. Then, n binary functions are learned, one for each bit position in these binary strings. During training for an example of class i , the desired outputs of these n binary functions are specified by the codeword for class i . New values $\mathbf{x}$ are classified by evaluating each of the n binary functions to generate an n -bit

²We could apply this method also to AdaBoost algorithm.

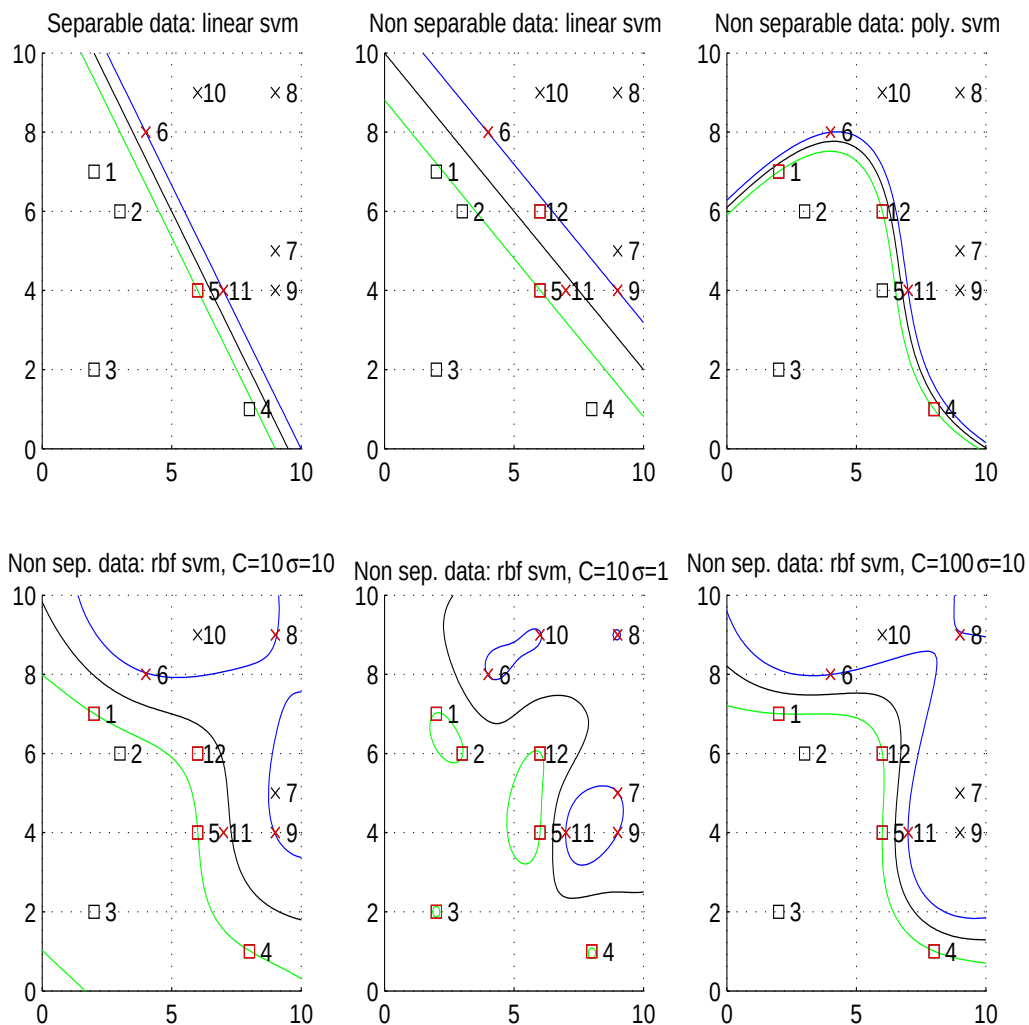


Figure 7.3: Example of classification using SVM for separable and non-separable synthetic data with different choices of kernel. Support vectors are drawn in red. The decision boundary is more shattered for high values of degree of polynomial kernel, low values of σ in RBF kernel, high values of C . A balance between generalization and performance in training set has to be found.

string s . This string is then compared to each of the k codewords, and $\mathbf{x}$ is assigned to the class whose codeword is closest³ to the generated string s .

The error-correcting code should satisfy the two properties:

- each codeword should be well-separated in Hamming distance from each of the other codewords
- each bit-position function should be uncorrelated with the functions to be learned for the other bit positions

For our experiments we derived the error-correcting code by T. Dietterich's collection of code matrices [3], see Figure 7.4.

```

*****
*
* 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 1 0 0 0 0 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 1 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 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 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 1 0 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 1 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 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 0 *
* 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 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 0 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 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 0 0 0 *
* 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 0 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 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 *
*
*****

```

Figure 7.4: Error-correction codes used in SVM experiments to solve the multi-class classification problem: 12 classes, codewords with 47 bits.

We did not implement SVM algorithm but we downloaded [4] a software in Matlab written by Anton Schwaighofer (2002). Its main features are:

- Except for the QP solver, all parts are written in plain Matlab.
- Extension to multi-class problems via error correcting output codes is included.

7.5 Experiments

We divide this last section into two parts. First, we describe the method to tune the parameters and then, we show the final results.

³In our experiments, we use Hamming distance.

7.5.1 Tuning

We describe which parameters must be optimized and the optimization method. We will refer to the urine database because the reasoning is the same for the pollen database.

MoG

The classifier using MoG, no matter the kind of feature used, needs to be optimized with respect to the choice of:

1. dimension of the feature space after application of Fisher's Linear Discriminants
2. number of components in each mixture
3. structure of the covariance matrix of each component
4. initial condition for EM algorithm used to estimate the mixture

In order to achieve the best reliable (and hopefully also the best) result, we proceed with a systematic method. The urine database has available twelve classes and nearly one thousand images for each category. Mucus class has only 546 images, Nhyal 860 and NSE 999. We want to work during this optimization with the same number of images in test and training for each class. Then, we divide the total number of available images in all classes in two sets: the first 500 will be used for only training purposes while the complementary set for only test. From the former set, we extract randomly 30 images in order to build a validation set. All training models will be shown in the following pages, are done on the rest 470 images of training set and the evaluation of the performance is done on the validation set. Then, when optimal parameters are found, a final test is done on 30 randomly chosen images of the test set.

As you can see from the curves of error rate (see Figure 7.5 and following ones), the performance is strongly dependent not only on the number of parameters used in the MoG (see Table 6.2) but also on the kind of parameters. For example, in previous simulations we found that even if you keep constant the number of parameters, the error rate sometimes increases if you choose to increase the dimension of the space instead of increasing the number of Gaussians in the mixture. For this reason, we run a lot of simulations for different values of the feature space dimension. While this value is maintained constant, several numbers of components are tried for different structures of covariance matrix (spherical, diagonal and full).

In order to estimate a parameter we need at least 3 points. In training we use 470 images and so we should stop our search when we reach a number of 150 parameters. On the other hand, our analysis wants to be exhaustive and so we try to perform simulations up to 300 parameters. However, when we run simulations with spherical covariance matrix, there are usually more than ten Gaussians (the maximum value was 25). Unfortunately, EM sometimes does not find convergence for a so high number of Gaussians even if the number of parameters is acceptable. Indeed, we checked the starting number of points assigned to each Gaussian by “k-means” algorithm and not surprisingly we found that some Gaussian has only few points (less than ten) and it is likely that these functions easily collapse in a single point or they can give rise to local minima in the error function which may give poor representations of the true distribution. Our assessment is strengthened by some warning (probability of data given the model nearly zero) of EM program and by the results of our test on the validation set.

As you can see, for example, in Figure 7.5 there are some irregularities in the trend of spherical covariance matrix because for a high number of Gaussians EM doesn’t find a model for one or two classes (giving an acceptable global error rate but for those categories the error rate is 100%!).

When we run experiments for the combination of classifiers we avoid these situations stopping the program at a lower number of Gaussians in the spherical case because we are looking for a stable system and we need a reasonable estimate of $P[C_j \mid data, mod_i]$.

The classifier using features based on PCA needs to work with dimensions of space multiple of six because each IPCs is splitted in three parts and we require to get the same number of dimensions coming from image and spectrum principal analysis⁴. For this reason, the following experiments are done for a number of dimensions equal to 12, 18 and 24.

The same argument is applied to the combination of classifiers.

The combination of classifiers modeling their outcomes needs a special care. Here, we have two trainings. In previous studies we found that a good MoG for the second training (dummy test to model outcomes) has two spherical Gaussians in each MoG. We use this model to optimize the first training and then with these good values we run another simulation to find good parameters for the second training. This new analysis confirms the previous results.

We chose to run the first training on 470 images per class (the full data training set) and then to select from this set 170 images to perform the

⁴A justification for this choice is given at the end of par. 6.2.

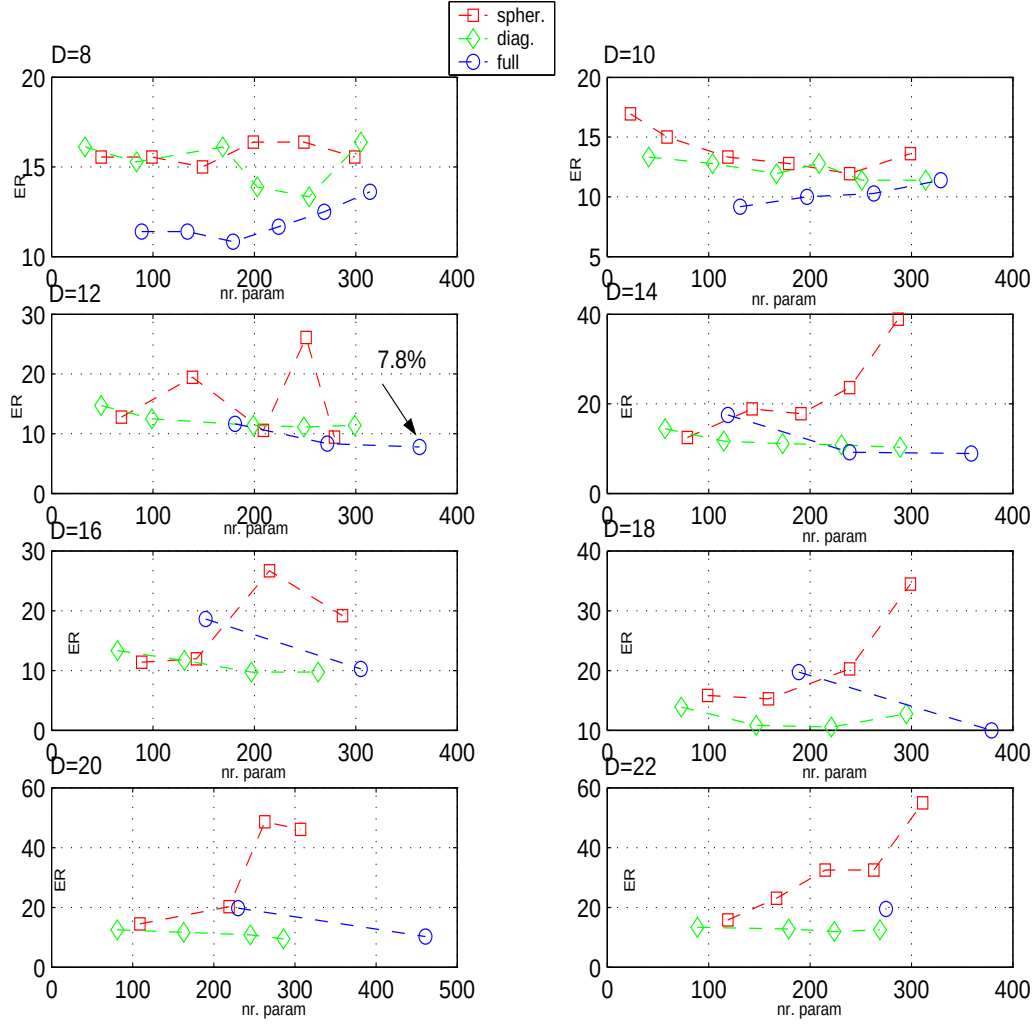


Figure 7.5: Classifier using MoG and feature based on local jets: ER as a function of the parameter number for different choices of feature space dimensions and structure of covariance matrix.

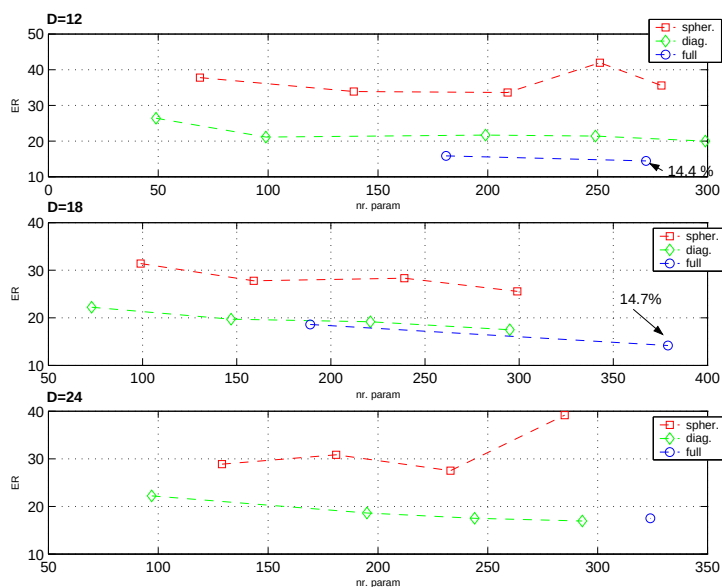


Figure 7.6: Classifier using MoG and feature based on image and spectrum principal components: ER as a function of the parameter number for different choices of feature space dimensions and structure of covariance matrix.

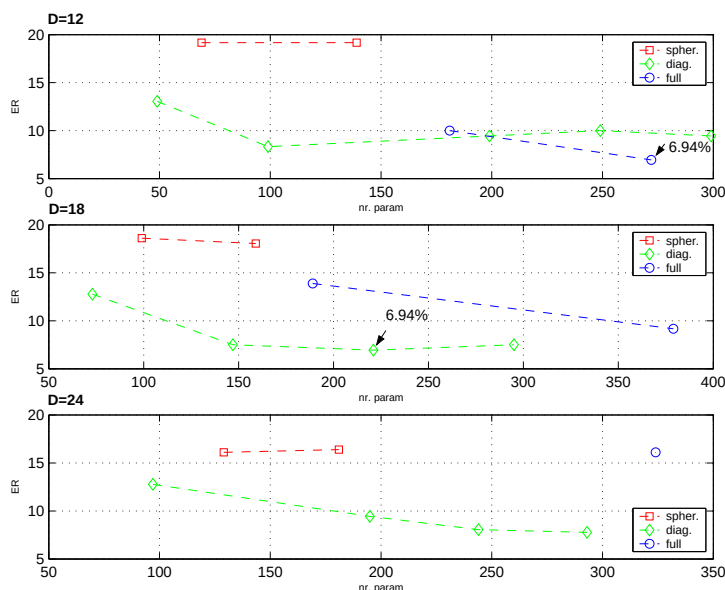


Figure 7.7: Classifier using MoG and combination of feature in independence hypothesis: ER as a function of the parameter number for different choices of feature space dimensions and structure of covariance matrix.

second training.

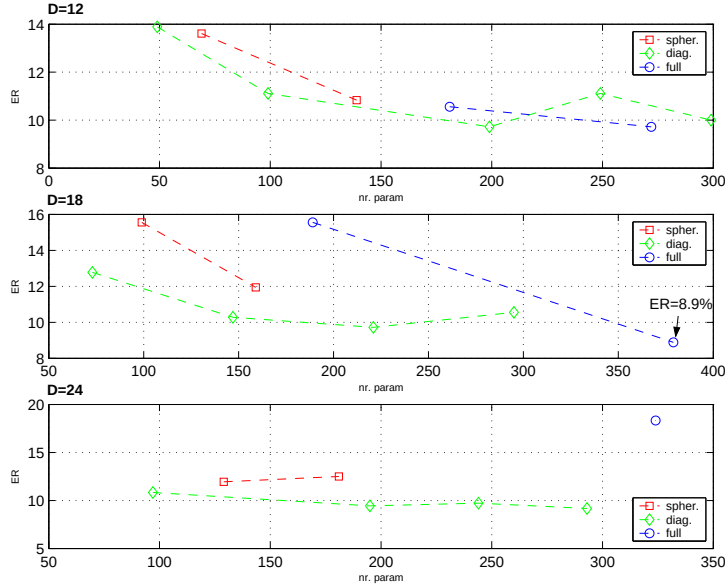


Figure 7.8: Classifier using MoG and making a combination of experts' response, first training: ER as a function of the parameter number for different choices of feature space dimensions and structure of covariance matrix.

features	gauss.	dimens.
local jets	4 full	12
PCA	3 full	12
comb. feat. (indep.)	6 diag.	18
experts' combination	2 full	18

Table 7.1: Best parameters found for MoG in urine database.

The last set of experiments has the goal to find the optimal initial conditions for EM. Our program calls some functions of NETLAB package to estimate the MoG; we use “gmm”, “gmminit” and “gmmem”. The first routine uses the Matlab function “randn” (generator of normally distributed random numbers), the second one calls “randn” and “rand” (generator of uniformly distributed random numbers). The initial state of these two functions is a vector with two components for “randn” and with 35 components for “rand”. So, for each mixture (that is for each class model) we have to

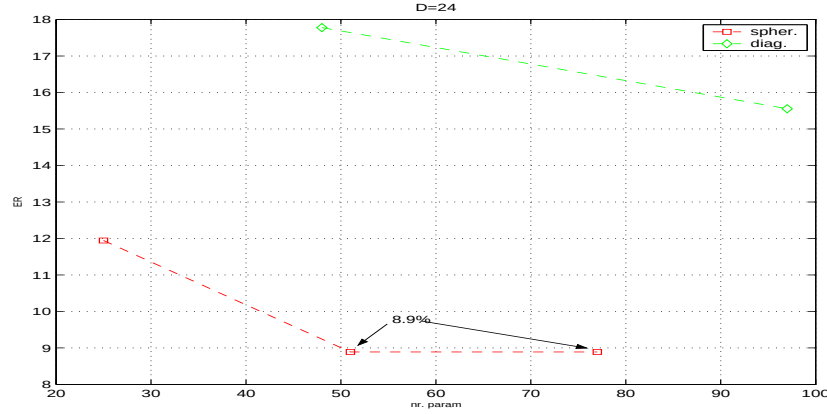


Figure 7.9: Classifier using MoG and making a combination of experts' response, second training (or modeling of results in dummy test): ER as a function of the parameter number for different choices of feature space dimensions and structure of covariance matrix.

find three vectors. We run one hundred experiments for each classifier leaving initial conditions free. At each iteration, we stored these vectors and the performance on the validation set. At the end, we force the initial state to that value which gave the lowest ER on the validation set.

Obviously, the number of Gaussians, kind of covariance matrix and number of space dimensions are the ones found in the previous simulations.

We show the results of these experiments in the form of scatter plots. At each iteration in which a different initial condition was chosen, a training model was estimated and the test was done on 30 images per class randomly extracted from the training set, test set and validation set. You can note the generally stronger correlation between the test and validation error rate versus test and training error rate.

In the classifier which combines the experts' outcomes, as you can see in Figure 7.12 (a similar situation happens also in Figure 7.11), there is no dependence of the performance on the initial conditions of the random number generator. We should not be surprised about it since we chose the maximum number of iteration of EM equal to one thousand and in this classifier we have to estimate the parameters of only two spherical Gaussians. For the classifier using feature based on image and spectrum principal components, three states are possible.

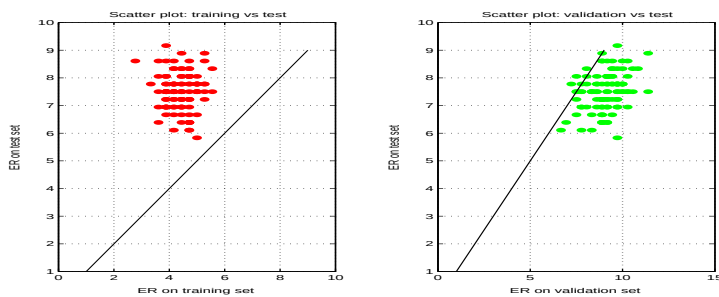


Figure 7.10: Classifier using MoG and feature based on local jets: each point in each scatter plot shows the error rate; 100 experiments were run with different initial conditions of random number generator.

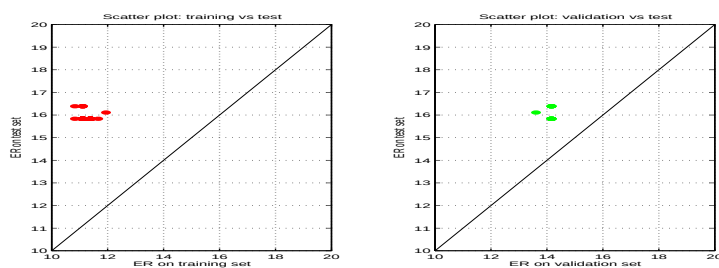


Figure 7.11: Classifier using MoG and feature based on image and spectrum principal components: each point in each scatter plot shows the error rate; 100 experiments were run with different initial conditions of random number generator (a lot of points are overlapped).

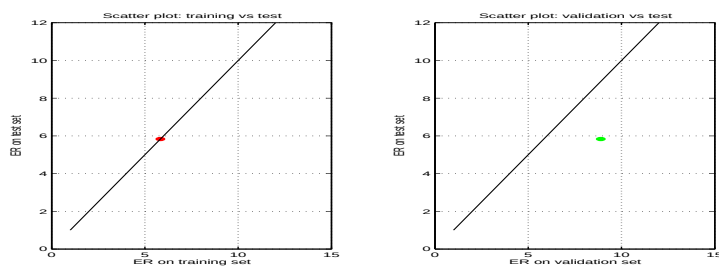


Figure 7.12: Classifier using MoG and making a combination of experts' response: each point in each scatter plot shows the error rate; 100 experiments were run with different initial conditions of random number generator (all points are overlapped).

AdaBoost

We considered 500 images from each class and extracted randomly 20 of them for validation. The complementary set is used for the final test. In Figure 7.13, the training, validation and test error rates are shown at each round. Then, we combined the 10 hypothesis in the final one as suggested in the pseudocode of Figure 7.2.



Figure 7.13: AdaBoost.M1: test, validation and training error rate as a function of the number of rounds. Training done on 480 images, each test on 20 images (randomly chosen).

SVM

The parameters we aim to find are:

1. kernel type
2. number of dimensions of feature space after reduction with Fisher's Linear Discriminants
3. parameter C
4. parameter of kernel, i.e. degree of polynomial kernel or width of Gaussian radial basis function

From our first experiments, it turns out that RBF kernels is able to achieve a lower error rate than polynomial kernel. So, we will refer only to the results of RBF kernel.

We summarize the results in Table 7.2 and following. The experiments are done taking 500 images for each class and extracting (only once) 30 images for validation purpose.

σ	D 10	D 12	D 14	D 16	D 18	D 20	D 22
0.25	10.83	11.67	12.78	15.00	16.94	19.17	20.28
0.5	9.17	8.89	9.17	9.44	10.56	10.00	10.83
1	10.83	10.83	10.00	10.56	11.67	10.56	11.11
1.5	11.39	11.11	11.39	12.22	12.78	11.94	12.22
2	11.94	12.22	12.50	13.06	13.06	13.33	13.89
3	14.17	13.61	13.61	14.44	14.17	15.00	14.44
4	15.28	14.72	15.28	15.00	15.83	15.83	14.72
5	16.11	15.56	15.56	16.11	16.39	16.11	17.22
10	18.33	16.94	18.61	18.61	18.33	18.89	18.33
15	20.28	18.33	18.33	18.89	19.44	19.44	20.28
20	22.50	19.44	20.00	20.56	20.83	21.11	21.94
50	26.11	23.06	24.44	25.28	27.22	28.33	28.89

Table 7.2: SVM with RBF kernel and penalty of errors $C = 1$: error rate as a function of the width σ and dimension D of feature space. The best performance is highlighted in thick black letters.

σ	D 10	D 12	D 14	D 16	D 18	D 20	D 22
0.25	12.50	12.78	12.50	15.28	16.39	18.33	19.44
0.5	10.83	10.00	8.61	9.44	9.44	8.89	10.28
1	10.00	10.56	9.72	9.72	9.44	8.61	9.17
1.5	11.39	10.83	11.39	10.56	11.67	10.00	9.72
2	11.39	11.67	11.39	11.39	12.22	10.83	10.83
3	11.67	11.39	11.94	11.94	12.22	11.67	11.94
4	11.94	11.94	12.22	13.33	12.78	12.22	12.22
5	11.94	10.28	9.44	10.28	11.39	11.11	10.56
10	12.78	11.39	11.67	11.94	12.78	12.78	13.06
15	13.33	13.06	12.78	13.89	14.17	13.89	14.72
20	14.44	14.44	14.72	14.72	14.72	14.44	15.28
50	20.56	18.06	18.89	18.89	19.44	20.00	20.56

Table 7.3: SVM with RBF kernel and penalty of errors $C = 5$: error rate as a function of the width σ and dimension D of feature space. The best performance is highlighted in thick black letters.

σ	D 10	D 12	D 14	D 16	D 18	D 20	D 22
0.25	10.56	11.39	11.94	11.67	13.61	13.33	16.11
0.5	13.33	10.28	10.00	10.56	12.78	11.11	11.39
1	10.83	10.28	11.39	9.44	8.89	9.72	9.17
1.5	10.00	9.72	9.44	9.44	10.83	8.89	9.17
2	10.28	8.89	10.00	10.28	10.56	8.61	9.72
3	10.28	10.00	11.11	11.67	10.28	10.56	9.72
4	11.11	10.83	12.22	13.06	11.11	10.83	10.83
5	11.67	11.39	11.94	12.50	11.67	11.94	11.94
10	12.22	12.50	11.39	12.50	12.78	13.61	13.06
15	13.89	13.06	13.33	12.78	14.17	14.44	15.00
20	15.00	14.72	13.33	12.50	14.44	14.44	15.00
50	15.83	16.67	17.78	18.33	18.89	18.89	19.00

Table 7.4: SVM with RBF kernel and penalty of errors $C = 20$: error rate as a function of the width σ and dimension D of feature space. The best performance is highlighted in thick black letters.

σ	D 10	D 12	D 14	D 16	D 18	D 20	D 22
0.25	12.78	12.22	12.50	14.17	15.56	18.61	19.17
0.5	12.22	10.28	9.44	10.83	10.28	10.83	10.28
1	10.28	10.00	9.72	9.72	10.56	9.17	8.61
1.5	10.28	10.28	10.83	9.17	10.56	9.17	9.72
2	10.56	10.00	9.72	10.00	9.44	9.72	9.72
3	9.72	10.28	11.39	9.72	10.83	9.44	10.00
4	10.83	10.83	11.94	10.28	10.56	10.28	10.28
5	11.39	10.83	11.39	11.67	11.94	10.00	11.39
10	12.22	12.22	12.50	11.94	12.78	12.50	12.78
15	13.33	12.50	12.78	13.61	12.78	13.89	13.89
20	14.17	13.89	14.72	15.00	13.89	15.00	14.44
50	16.94	16.11	16.39	16.39	17.22	18.61	18.89

Table 7.5: SVM with RBF kernel and penalty of errors $C = 50$: error rate as a function of the width σ and dimension D of feature space. The best performance is highlighted in thick black letters.

σ	D 10	D 12	D 14	D 16	D 18	D 20	D 22
0.25	11.67	11.67	14.17	15.00	16.11	18.89	18.89
0.5	8.89	8.89	10.28	10.28	11.11	11.11	11.67
1	8.06	8.33	7.50	7.50	8.33	8.89	8.89
1.5	7.50	7.78	7.50	5.83	8.06	6.67	8.61
2	6.94	7.50	7.78	6.94	6.67	6.39	7.78
3	7.50	8.61	7.50	8.06	7.22	7.50	7.50
4	8.61	8.06	7.78	8.06	8.06	7.22	7.50

Table 7.6: SVM with RBF kernel and penalty of errors $C = 100$: error rate as a function of the width σ and dimension D of feature space. The best performance is highlighted in thick black letters.

7.5.2 Final experiments

We show the final results for urine and pollen database. In order to assess the performance, a confusion matrix is built. Each row is the class to which the test images belong, the columns are the classes to which the classifier assigns the test images. Thus, in the intersection between the j -th row and the i -th column we find the following estimate:

$$P[outcome \in C_i \mid image \in C_j] = \frac{\text{nr. of items of } C_j \text{ classified as belonging to } C_i}{\text{total nr. of tested images belonging to } C_j} \quad (7.1)$$

In order to evaluate the overall performance we compute the error rate defined as,

$$ER = \frac{\text{nr. of not correctly classified test images}}{\text{total nr. of test images}} \quad (7.1)$$

Because not all the categories in our databases have the same number of images, we train and test on a uneven number of images for class. Then, we define a new error rate which is the normalized sum of single class error rates

$$ERp = \frac{1}{K} \sum_{j=1}^K \frac{\text{nr. of misclassified test images belonging to } C_j}{\text{total nr. of test images belonging to } C_j} \quad (7.1)$$

For example, when we consider the whole system for pollen recognition we will see that the classifier has to analyze also particles of junk. The ratio between pollen and junk particles is 1 out of 10. If we test 100 pollen patches and 1000 junk patches it is possible that the classifier has an $ER = 10\%$ which can be considered quite good. However, it is possible that most of junk particles are classified as junk and most of pollen particles are misclassified, the ER simply can not say if the less numerous classes are well classified. Instead when you consider also the ERp you can check if the error is evenly or proportionally distributed among the categories.

Urine database

First, we consider the classifier using mixture of Gaussians with the different kinds of features studied.

During the tuning of parameters, we focused on data belonging to the training and validation set. Now, we show some results of experiments done in the following way:

1. training and test using the full data set available and with different number of images per class; we pay attention to put in the training set all the images belonging to the previous training and validation set and we choose randomly (from the complementary set) 10% of the total amount of images for test; the result will be the average of 10 experiments
2. we run 100 experiments on the full data set available; at each iteration we randomly extract about 10% of images for test purpose.

In Figure 7.15, Figure 7.14, Figure 7.16 and Figure 7.17 we show the graphical representation of confusion matrix using the first method for the four different kinds of classifiers based on MoG modeling.

Note that the numbers in the plot are in percentage with respect to the total number of images tested for a certain class. Moreover, we show the average result on 10 experiments.

The experiments with the classifier that combines the experts' outcomes, need a comment. We have to choose the subdivision of the training set for the first and second training. When we used 7/9 of the training set for the first training and 4/9 for the second one (so, there is a partial overlap of 2/9) we got an ER equal to 9.8%; a second time we took all the images of the training set for the first training and we extracted randomly from this set half of them for the second training and we got an ER equal to 8%. The trend was confirmed by the last experiment, shown in Figure 7.17, when we used for both trainings all the images available in the training set; in this way we got an ER of 7.7%. Even if the second training models the training outcomes (or the training errors), the general performance is improved.

The best result is achieved with the classifier using features based on local jets, its error rate is equal to 6.8% (the proportional error rate is 6.5%).

Finally, we show the average results of 100 experiments on the full data set taking randomly 10% of images for test, see Figure 7.18, Figure 7.19, Figure 7.20.

Why did we choose to perform 100 experiments? Because we want a reliable estimate of the average ER with a low standard deviation. Now, we give a brief overview of the theoretic study for the choice of the experiment number.

If x is a stochastic binary variable with alphabet $A_x = \{0, 1\}$ and $P[x = 1] = p(1) = p$ it is easy to find that the mean of x is $m_x = p$ and its variance $\sigma_x^2 = p(1 - p)$. We can interpret this variable as the outcome of the following event: "the classifier takes the right decision on the test image".

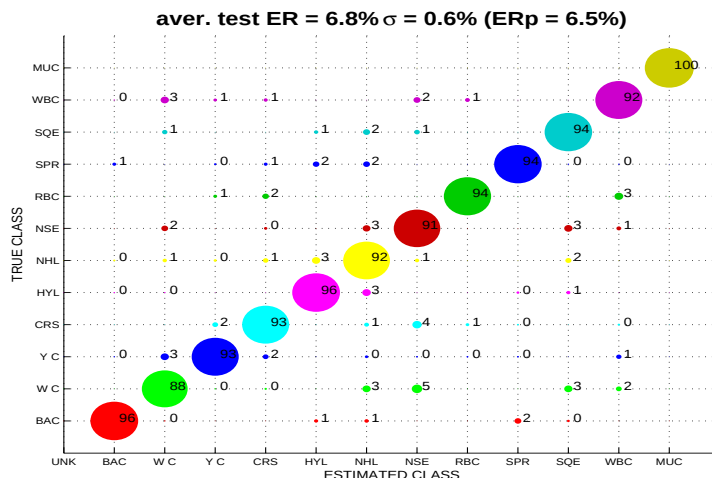


Figure 7.14: Classifier using MoG and feature based on local jets: averaged confusion matrix for experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test; there is no overlapping between the test items and previous trained images.

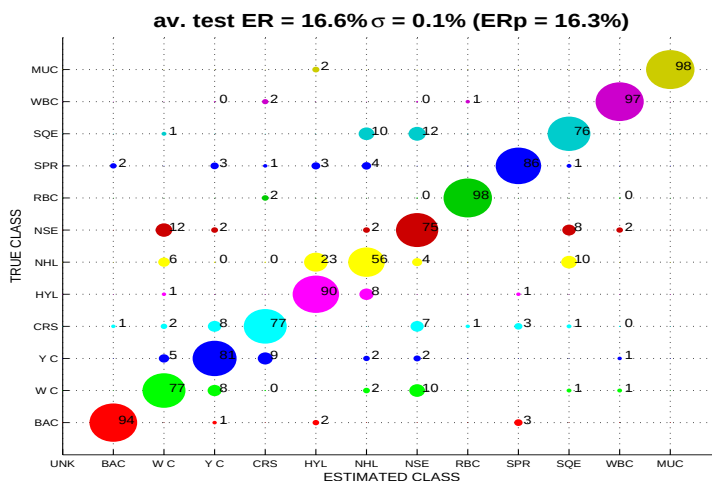


Figure 7.15: Classifier using MoG and feature based on image and spectrum principal components: averaged confusion matrix for experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test; there is no overlapping between the test items and previous trained images.

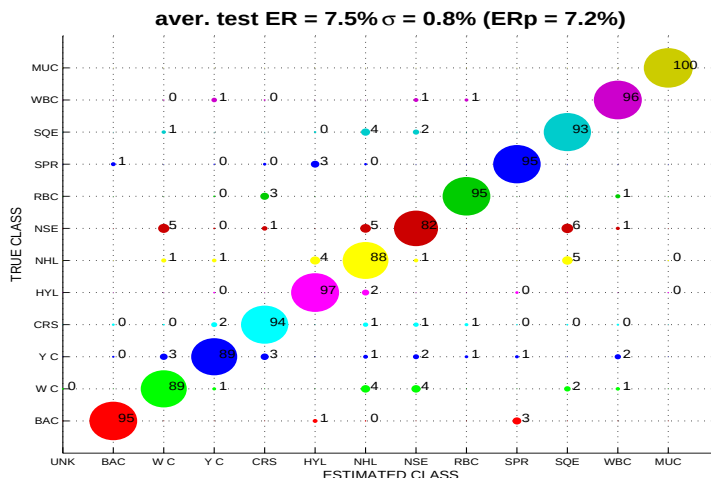


Figure 7.16: Classifier using MoG and feature combination in indep. hyp.: averaged confusion matrix for experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test; there is no overlapping between the test items and previous trained images.

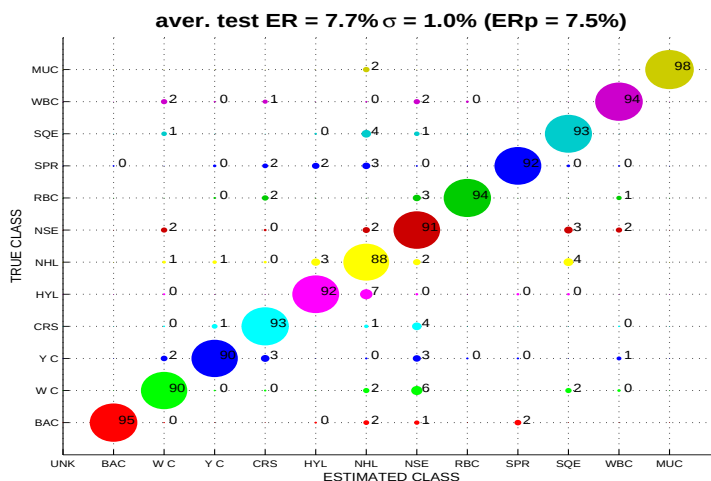


Figure 7.17: Classifier using MoG and making a combination of experts' response: averaged confusion matrix for experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test; there is no overlapping between the test items and previous trained images.

If you consider n stochastic binary variables and all these variables have the same parameter p and are independent, we can define the sum variable $y = \sum_{i=1}^n x_i$ which is a binomial one with parameters n and p ; its mean and variance are: $m_y = np = nm_x$, $\sigma_y^2 = np(1-p) = n\sigma_x^2$. Let be $z = y/n$, then $m_z = m_y/n = p = m_x$, and $\sigma_z^2 = \sigma_y^2/n^2 = \sigma_x^2/n$. We can think that y describes the statistic of n decisions of the classifier on n test images, and z the performance when considering n test images. You can note that increasing n you can decrease σ_z^2 .

In order to have a reliable result for the performance of our classifier we could average the results on many experiments. Given the (estimated) performance if we choose properly the number of images to test and the number of experiments, we can control the variability of the error rate.

Let be m the number of experiments we want to run and $w = \frac{1}{m} \sum_{i=1}^m z_i$, then it can be shown that $m_w = m_z = m_x$ and $\sigma_w^2 = \frac{1}{m} \sigma_z^2 = \frac{1}{mn} \sigma_x^2$.

Finally, we can approximately estimate the standard deviation of the ER in our experiment doing the following hypothesis: if the probability to misclassify an image is nearly $p = 0.1$ (even if it depends on the belonging class), m is equal to 100 and n is about $100 \cdot 12$ (in practice it is less), then

$$\sigma_w = \frac{\sigma_x}{\sqrt{mn}} \simeq \frac{\sqrt{9 \cdot 10^{-2}}}{\sqrt{12 \cdot 10^4}} \simeq 7 \cdot 10^{-3}$$

With this choice of parameters, we can be confident to find an ER with a standard deviation of about 0.7% (below 1%).

These result is confirmed by our experiments as you can check in Figure 7.18, Figure 7.19, Figure 7.20.

As you can see for example in Figure 7.18 and expect, the confusion matrix has an high symmetry with respect to the diagonal. For example, 2.5% of Hyal are classified as Nhyal and vice versa, 4% of NSE are classified as white cell clumps and 3.9% of these particles are classified as belonging to NSE class.

The best performance is again achieved using the classifier with features based on local jets, the error rate is equal to 7.6%. If you look at Figure 7.18, the best classified class is mucus with an ER equal to 2%; the worst classified particle is NSE. Most of errors are done between white cell clumps and NSE, white cell clumps and squame epithelium, hyal and nhyal.

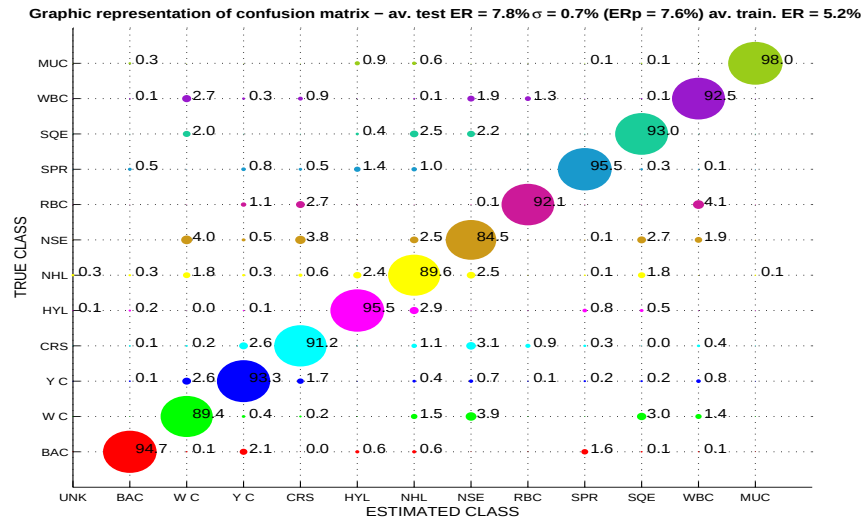


Figure 7.18: Classifier using MoG and feature based on local jets: averaged confusion matrix of test errors for 100 experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test.

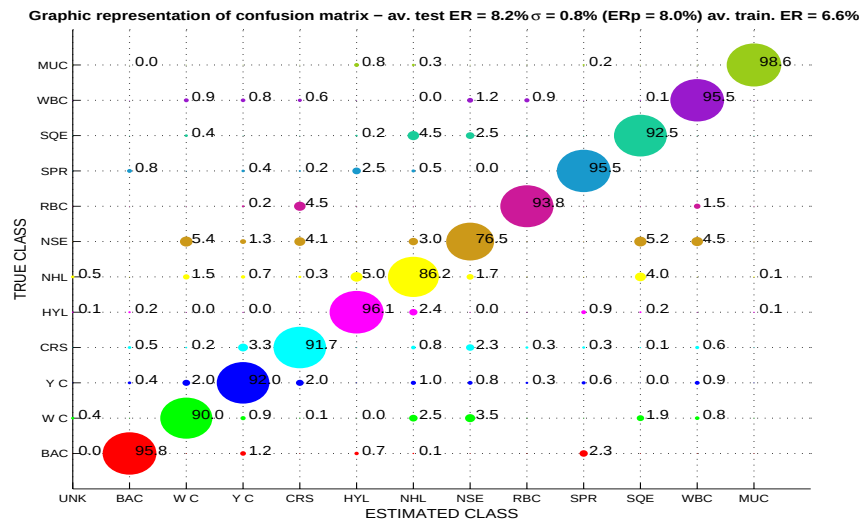


Figure 7.19: Classifier using MoG and feature combination in indep. hyp.: averaged confusion matrix of test errors for 100 experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test.

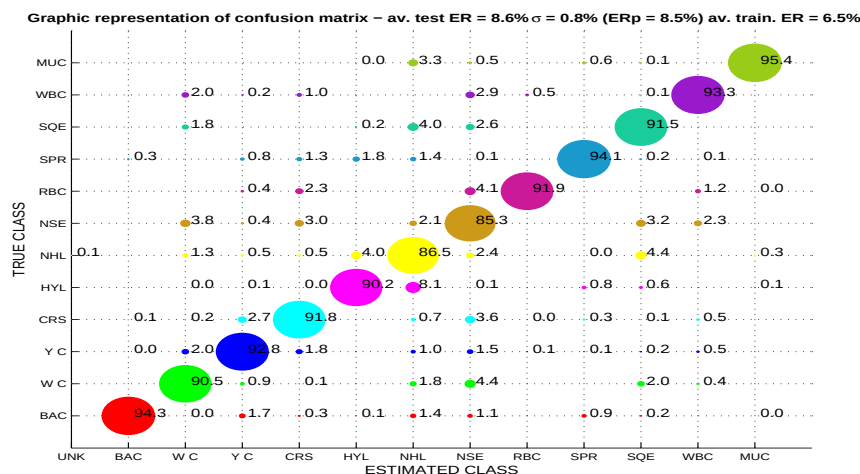


Figure 7.20: Classifier using MoG and making a combination of experts' response: averaged confusion matrix of test errors for 100 experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test.

We now show the results of AdaBoost inspired algorithm. With the hypothesis found during the experiments of Figure 7.13, we run a test on 10% of images in the data set (not belonging to previous training or validation set). In Figure 7.21 and Figure 7.22, the averaged performance on 10 experiments is shown for the boosted classifier and the standard one (that is taking only the first hypothesis). As you can notice there is an improvement of 0.5% in the error rate.

Figure 7.23 shows the performance when optimal parameters are found for the standard classifier that performs the training with 480 images.

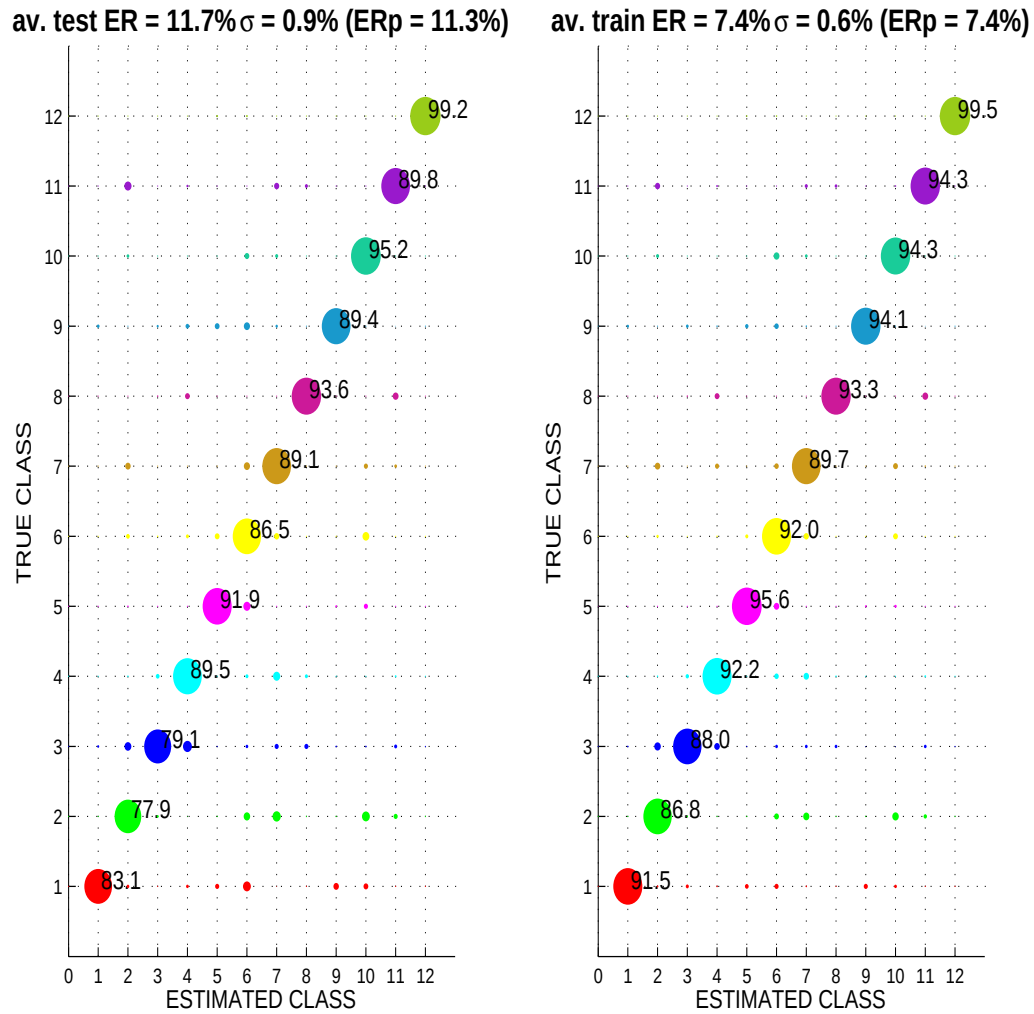


Figure 7.21: AdaBoost.M1: test and training confusion matrices. The training is performed on 480 images and the test on 10% of full data set. The result is averaged on 10 experiments.

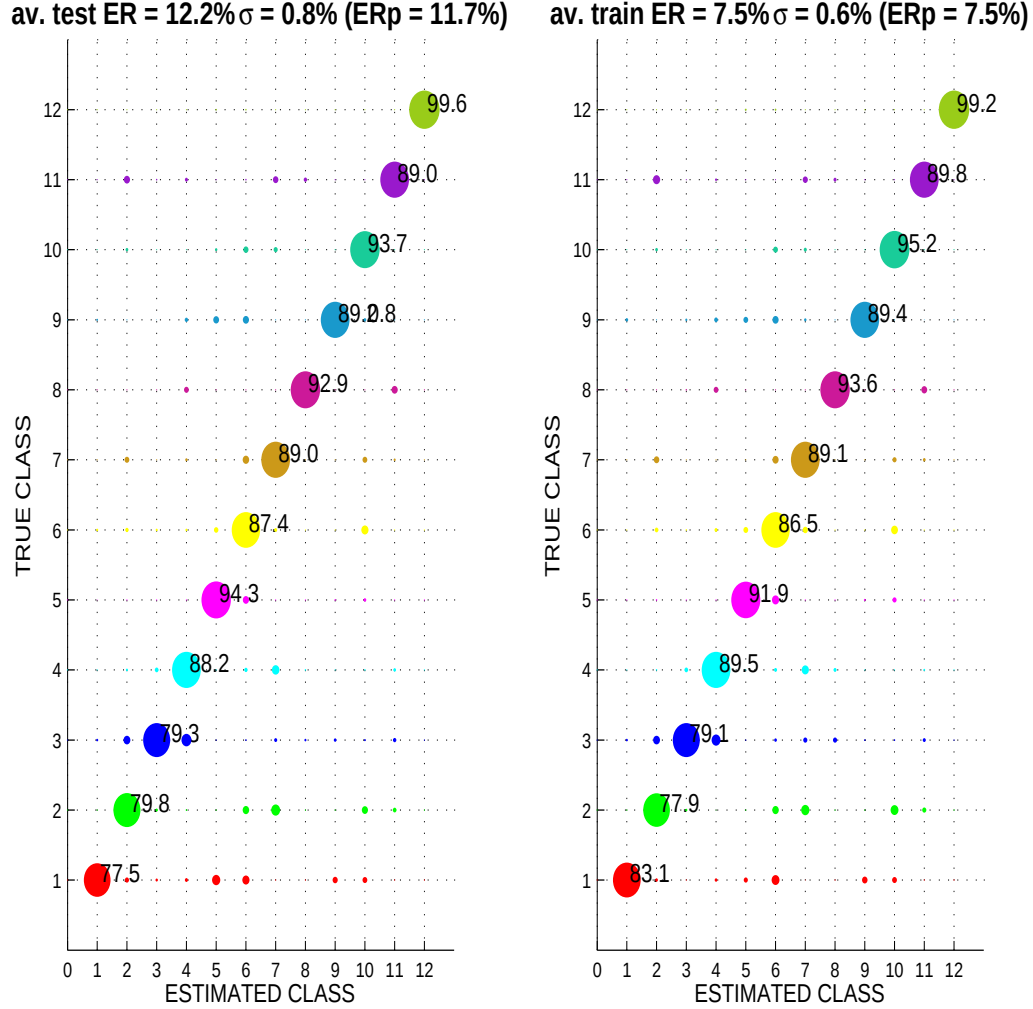


Figure 7.22: Test and training confusion matrices for the original classifier using MoG and local jet based features. The first hypothesis was used and thus, the feature space dimension is 12 and we are modeling each class with a mixture of 4 full covariance Gaussians. The training is performed on 480 images and the test on 10% of full data set. The result is averaged on 10 experiments. In this figure and in the following ones, the classes are identified with numbers in accordance to Table 3.1. The numbers in the diagonals are the percentages of correct classification in each class.

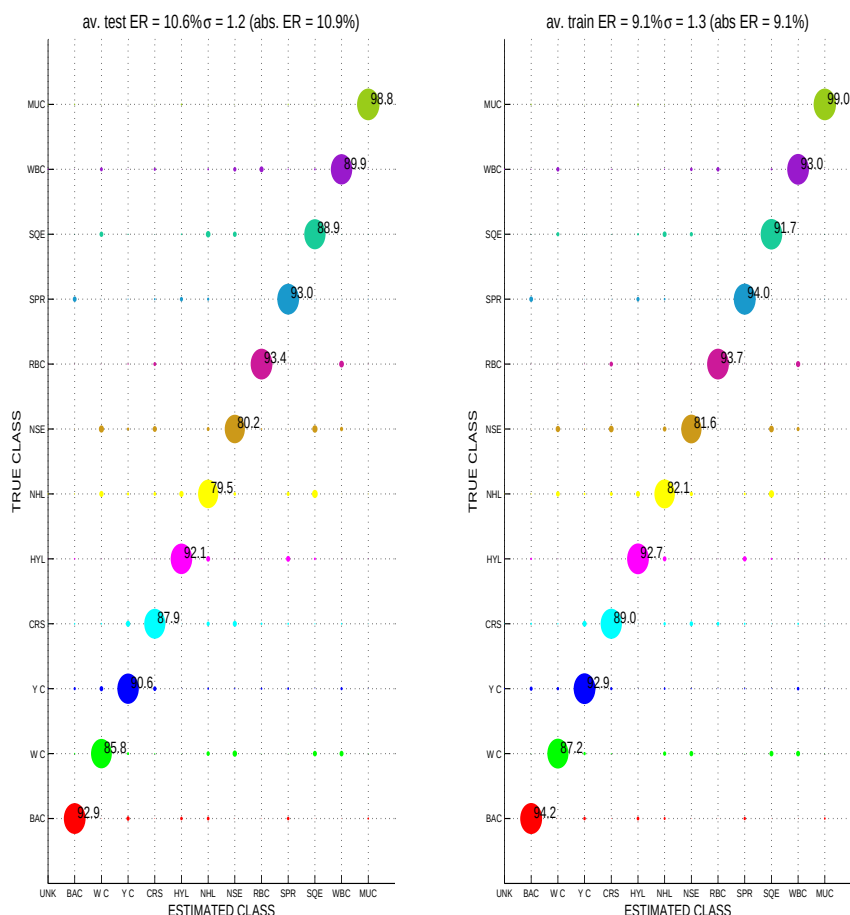


Figure 7.23: Test and training confusion matrices for the original classifier using MoG and local jet based features. The best parameters have been found for a training with 480 images: the feature space dimension is 12 and we are modeling each class with a mixture of 4 diagonal covariance Gaussians. The training is performed on 480 images and the test on 10% of full data set. The result is averaged on 10 experiments. The numbers in the diagonals are the percentages of correct classification in each class.

Taking the best achieved performance on validation set, we run experiments using SVM following the methodology above-mentioned for AdaBoost. Unfortunately, we achieved an error rate of 12.2% in test and below 2% in training. This is caused by training data overfitting. We have to choose parameters for SVM to find a sort of balance between accuracy attained on a particular training set, and the capacity of the machine to learn the general properties of our data. With this choice of parameters, the system seems to have a little generalization. For this reason, we tried other “good” but sub-optimal values for the parameters and the best performance is reported in Figure 7.24. There is still a big gap between the performance on training and test set however there is an improvement of 0.5% and 1% in the absolute and proportional error rate with respect to the classifier using MoG (first hypothesis of AdaBoost).

The last experiment was done using SVM to test the 10% of images when training on the 90% of images. We performed 13 experiments and we have randomly extracted the test images at each iteration. As you can see in Figure 7.25, the system seems to have still a poor generalization given the gap between the test and the training error rate and the performance is a little bit worse than the one we get using the Bayesian classifier (see for a comparison Figure 7.18).

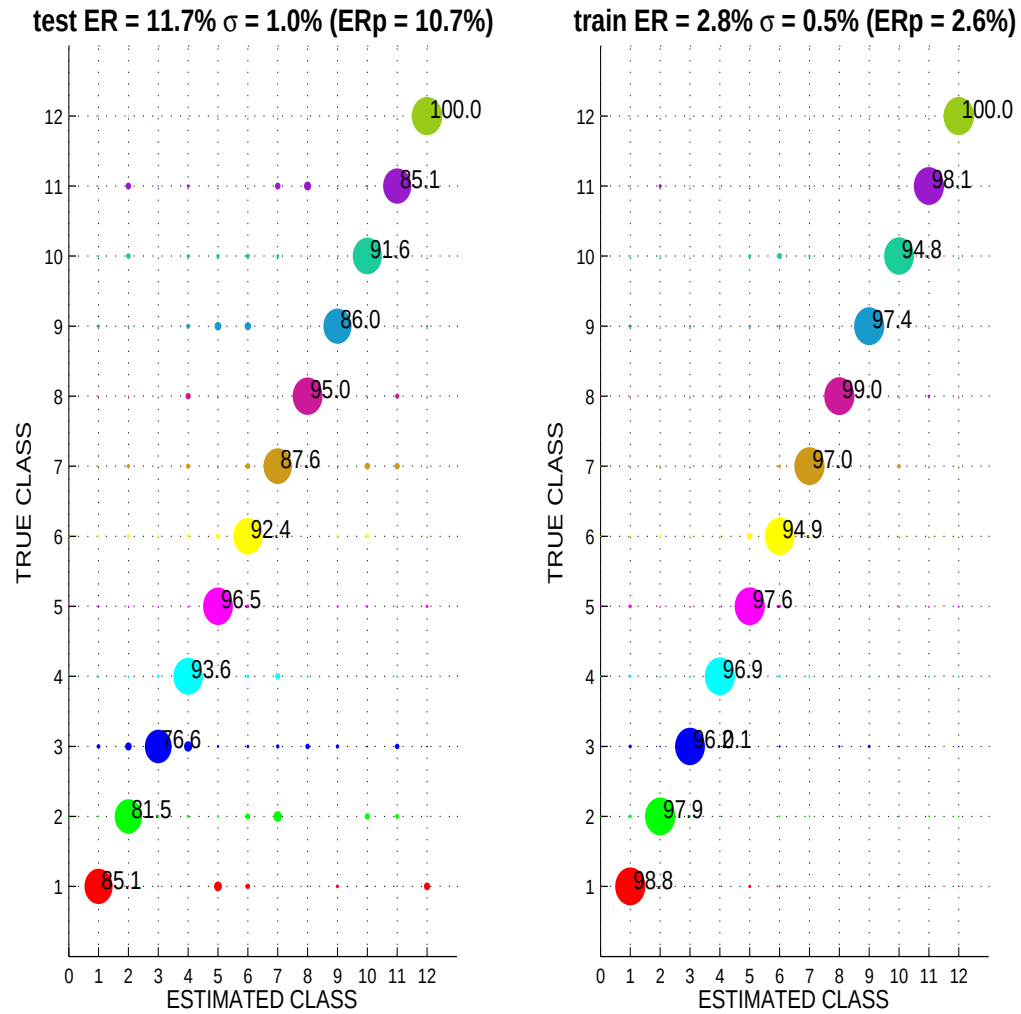


Figure 7.24: SVM with rbf kernel $C = 5$, $\sigma = 1$ in a 20 dimensional feature space: test and training confusion matrices. The training is performed on 500 images and the test on 10% of full data set. The result is averaged on 10 experiments.

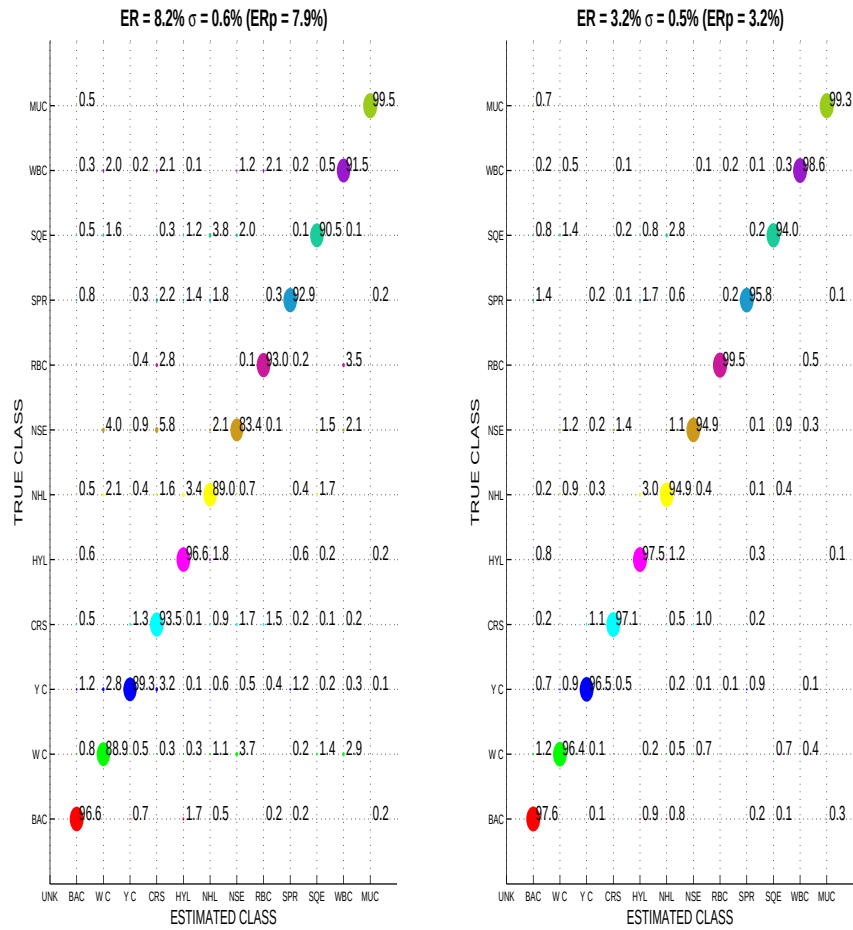


Figure 7.25: Classifier using SVM: averaged test (left panel) and training (right panel) confusion matrices for 13 experiments in which 90% of images in the full data set is used for training and 10% is randomly extracted for test.

Pollen database

We show the performance of classifier using MoG with feature based on local jets. The databases we are considering is the database of extracted patches derived from the expert's reference list and the database built while running the detector using DoG images. This last set of images has been checked in order to remove samples with bad segmentation (see par.5.4).

The pollen particles we have used for classification are listed in tab.7.7.

genus	nr. expert's patches	nr. good detector patches
alder	416	355
ash	329	303
chin. elm	522	466
jacaranda	282	223
mulberry	111	74
oak	413	329
olive	326	220
pine	705	606
<i>TOTAL</i>	<i>3104</i>	<i>2576</i>

Table 7.7: List of pollen particles belonging to the most numerous classes in the databases of extracted patches. The second column refer to patches extracted thanks to the expert's list, the last column refers to patches extracted by our detector.

Because of the shortage of images in these databases, we did not consider a training, validation and test set. We randomly extracted from each class the 10% of images to perform the tuning. Then, we run 100 experiments to assess the performance of the system. At each iteration, the 10% of images was randomly selected for the test stage.

Classification using expert's patches

We have computed the performance using the expert's patches in order to find a lower bound for the error rate of the classifier as a part of the system for real-time pollen count. Why are we getting a lower bound? First, experiments with detector patches use less samples to train the system. Secondly, the automatic segmentation of grains is not always perfect (see par.5.4). Last but not least, we have to deal with a lot of false alarms in any real situation.

In order to tune the system, we have run many experiments varying the number of parameters: dimensions of feature space (from 6 to 18), number of components in each mixture of Gaussians (from 1 to 10 when appropriate⁵), structure of covariance matrix of each component, initial conditions for EM algorithm. The optimal parameters can be found in tab.7.8. Finally, we have run 100 experiments. Each time, we have randomly extracted from each class the 10% of samples for test, the rest was used to perform the training. The average result on 100 experiments is shown in Figure 7.26.

parameters	value
dimensions	14
nr. components	2
covariance	diag.

Table 7.8: Parameters found thanks to tuning stage which also provided us a good set of initial conditions for the random number generator used to perform EM algorithm.

⁵This means that we tried to keep the number of parameters to less than 50 because *mulberry* class has about 100 samples.

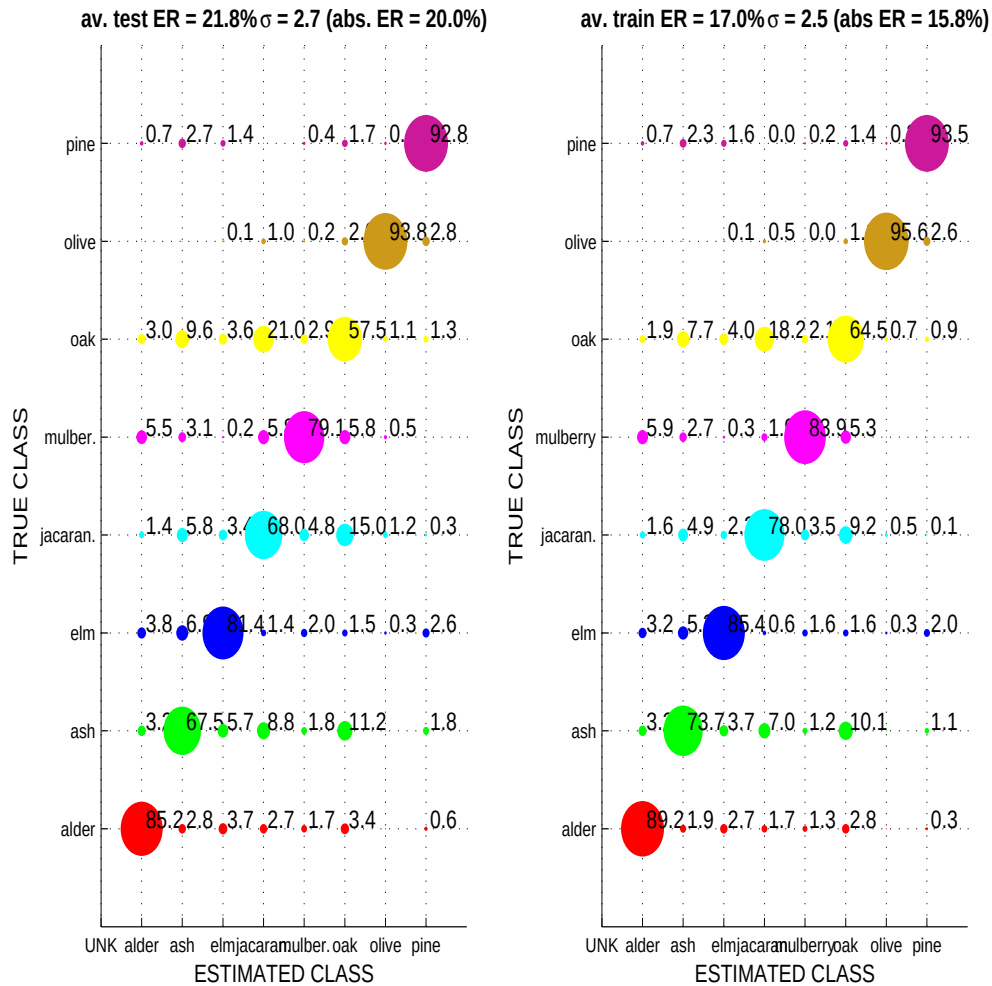


Figure 7.26: Average test (left side) and training (right side) confusion matrices of 100 experiments in which expert's patches were used (see second column of tab.7.7). The proportional error rate is the sum of the error rates of each single class; these error rates are the ratio between the number of misclassified patches belonging to that class and the product of the number of tested classes by the total number of samples belonging to that class. The error rate shown in parenthesis is the ratio between the total number of misclassified images and the total number of tested images. The proportional error rate is useful when we run experiments using a different number of test samples for each class. In these experiments, we have randomly extracted the 10% of images for test at each iteration.

Classification using detector patches without false alarms

The tuning stage gave us the parameters that are shown in tab.7.9. We present the result of classification using detector patches without the false alarm class in Figure 7.27. This is useful to assess the quality of our automatic segmentation, even though the performance is a little worse due to the minor number of images used in training. We observe that the tuning stage provided us the same parameters that were found using the expert's patches, and that also the performance is very close to that one (see Figure 7.26). Thus, we can conclude that the automatic segmentation is good. Finally, given the strong similarity of pollen grains belonging to certain classes (i.e. ash, jacaranda, oak) we can be satisfied about an error rate of nearly 20% because this classification has been performed using a classifier developed for the cells in urine.

parameters	value
dimensions	14
nr. components	2
covariance	diag.

Table 7.9: Parameters found thanks to tuning stage which also provided us a good set of initial conditions for the random number generator used to perform EM algorithm.

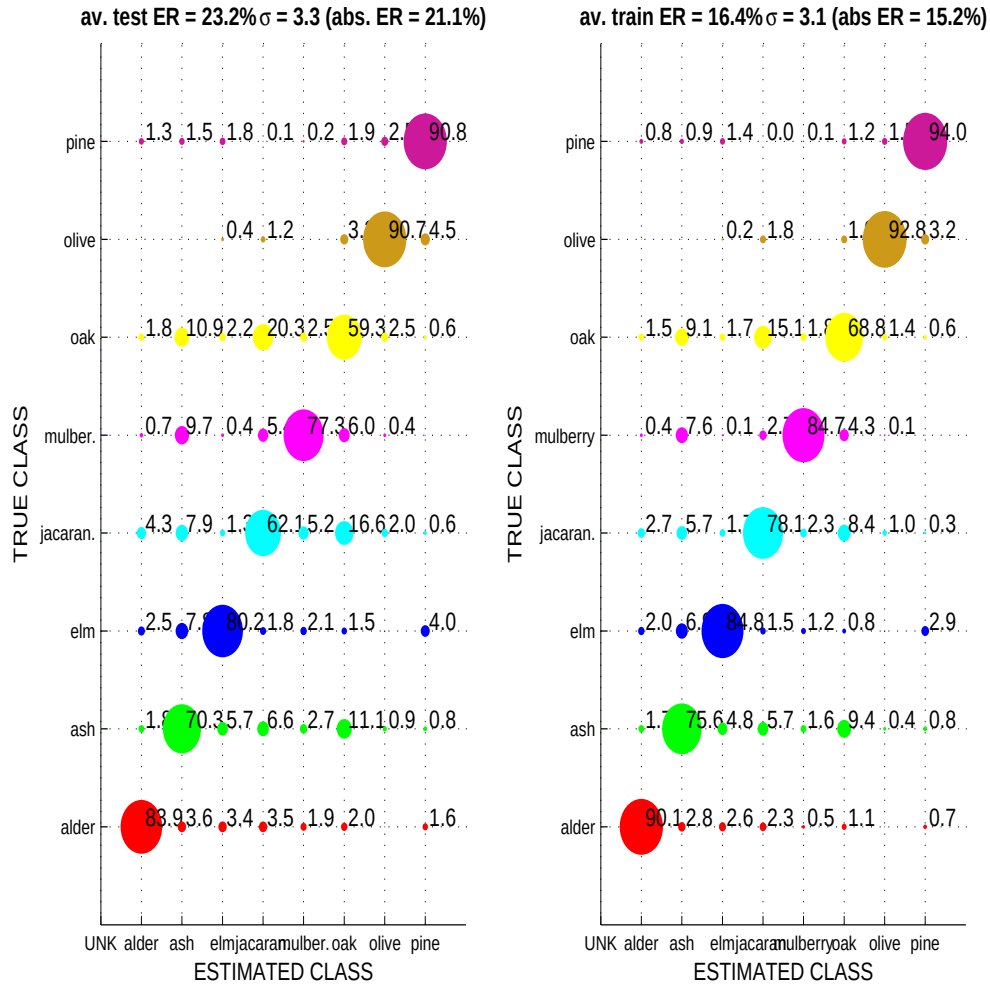


Figure 7.27: Average test (left side) and training (right side) confusion matrices of 100 experiments in which detector patches were used (see last column of tab.7.7). In these experiments, we have randomly extracted the 10% of images for test at each iteration.

Chapter 8

Analysis of errors

In this chapter, we analyze errors made by detector and classifier. We aim to find where the mistakes are made and to understand why they are made. We conclude each section with some ideas to overcome these problems.

8.1 Detector

A missed detection may occur when:

- pollen particle is very low contrast in a very noisy image
- pollen particles are grouped in a single blob in a way that we get only an interest point for all these grains, this usually produce also a bad automatic segmentation
- dust is very close or attached to a pollen causing a bad segmentation.

These limitations are intrinsic of this system. We could improve only the algorithm for the determination of bounding boxes in order to allow junk around the pollen. On the other hand, this could slow down the detection.

False alarms could be avoided considering information about color, about average brightness, about texture. Junk particles usually are black or pink (i.e. bubbles of glue) while pollen grains are brown, cyan, red and green. In the current system we do not have a lower threshold of brightness that could discard black particles of dust. Pollen grains have always a kind of texture that makes them different from other particles.

8.2 Classifier

In this section, we will refer to experiments done on urine database. The classification errors will be analyzed with reference to the Bayesian classifier using feature based on local jets.

In particular, we refer to experiments done on full data set taking 10% of images for test. These images are randomly extracted by a set never used for training and validation.

In Figure 8.1, we show the misclassified images with the estimated class and the assigned probability (in percentage). This probability is the probability of the estimated class given the extracted feature vector. We note that for most of these images the probability is below 90%. This is a good point because in a real application we could assign these items to an unknown class for further analysis.

Let us try to give a deeper insight on the previous misclassification. In Figure 8.2, we show a collage in which in the central column there are some (misclassified) images of Figure 8.1 with assigned high probability. In the first three columns there are images belonging to the true class that are the nearest to the one in central column in the euclidean norm of the feature space (after the projection on FLD). In the last three columns, there are images belonging to the estimated class that are the nearest in the same feature space.

If you follow a row you can see the smooth transition in the features space from one class to the other; with this interpretation the misclassified image is on the boundary between the two classes and it can be thought like a transition item.

We note that some errors could be avoided by considering features with information on shape. For example, in the second row of Figure 8.2, it seems clear that the average of local jets doesn't allow to separate clearly the image on the fourth column from the image in the fifth column but we should overcome this difficulty considering shape and texture of these two images.

On the other hand, there are images which are really hard to distinguish because of a certain intrinsic ambiguity of the database.

What is the difference between the images on the fourth and fifth columns in the intersection with the fifth, sixth, tenth and eleventh row? It is an embarrassing question, which was also posed to the experts who provided us this database, and they could not say it! This intrinsic ambiguity will constitute a lower bound for the error rate of any classifier on this data set and this bound could be found with a test on the human (expert's) error rate on this

database.

Analogous analysis can be done on pollen database. Errors are made because our features do not capture all the essence of texture in particles and because we are not using color information.

(true class, estimated class, prob.)											
(1,9,9)	(2,11,82)	(3,4,57)	(4,3,72)	(5,9,99)	(6,5,64)	(7,2,88)	(8,4,99)	(9,1,96)	(10,5,79)	(11,2,50)	
(1,5,80)	(2,11,73)	(3,11,100)	(4,3,93)	(5,6,100)	(6,2,42)	(7,2,3)	(8,4,63)	(9,6,82)	(10,6,99)	(11,2,51)	
(1,9,62)	(2,7,89)	(3,11,78)	(4,7,53)	(5,10,89)	(6,5,94)	(7,2,76)	(8,11,85)	(9,5,97)	(10,7,36)	(11,2,98)	
	(2,11,59)	(3,4,81)	(4,9,100)	(5,6,54)	(6,10,75)	(7,2,88)	(8,11,89)	(9,3,39)	(10,2,97)	(11,3,65)	
	(2,7,93)	(3,2,67)	(4,7,100)	(5,6,100)	(6,10,1)	(7,2,55)	(8,4,72)			(11,2,75)	
	(2,7,92)	(3,7,51)	(4,7,3)	(5,9,15)	(6,3,62)	(7,2,46)	(8,4,0)			(11,7,85)	
	(2,10,53)	(3,4,58)		(5,6,96)	(6,5,62)	(7,4,59)	(8,11,99)			(11,1,65)	
	(3,2,61)					(7,6,14)	(8,11,68)			(11,2,97)	
						(7,6,13)	(8,4,91)			(11,4,65)	
										(11,8,96)	
1: Bact	2: Wh. Cell Clumps	3: Clumps Ysts	4: Crystal	5: Hyal	6: Nhyal	7: NSE	8: Red Bl. Cells	9: Sperm	10: Sq. Epitel.	11: Wh. Bl. Cells	12: Mucus

Figure 8.1: Analysis of classification errors: misclassified images by Bayesian classifier using features based on local jets.



Figure 8.2: Analysis of classification errors: nearest images in the euclidean norm of the feature space. In central column the misclassified images, in the first three columns the closest images belonging to the true class, in the last three columns the closest images belonging to the estimated class; each row is referred to the image in its central column. Above each image, there is the class number and its index in the database.

Chapter 9

Whole system

This chapter has for its main aim to combine detector and classifier in order to assess the performance of the whole system.

We can identify particles only in the database of pollen captured by spore trap and so, we will refer to this database throughout the following discussion.

We consider the detector described in chapter 5 using DoG images. Given its performance, we assume that for each found pollen there are 10 false alarms (see tab. 5.8 and 5.9).

The first step was to build a database with those patches individuated by this detector; each image was put in the proper folder according to the experts' reference list. Then, we removed from this set only those samples having a bad segmentation that is mainly due to junk around the grain or to other pollen particles attached to the grain. In par. 5.4 we showed some examples of patches that are in this database and some patches that were removed, (see Figure 5.13 and following ones). Finally, we added a new class to gather the false alarm patches. Note that during classification we also refer to another class: "*unknown*". This is the class where we put the images whose maximum posterior probability is less than a certain threshold. Tab. 9.1 summarizes the number of pollen samples available in each category.

We apply the Bayesian classifier using features based on local jets. This classifier is one of the best we have found but it needs a lot of images to estimate parameters in the MoG. For this reason, we consider only categories with more than 70 images, namely: alder, ash, elm, jacaranda, mulberry, oak, olive, pine and obviously the false alarm class.

The first experiment aims to find the performance of the system considering all the available pollen categories. This can be done modeling the false alarm class or just only the pollen categories. We tried both solutions.

Finally, we tried to assess the performance of the whole system when using only few pollen classes. This is meaningful since in each season only few

genus	nr. good detector patches
alder	355
ash	303
chin. elm	466
jacaranda	223
mulberry	74
oak	329
olive	220
pine	606
<i>TOTAL</i>	<i>2576</i>

Table 9.1: List of pollen particles belonging to the most numerous classes in the database of patches extracted by our detector. We are considering only those samples having a good segmentation.

species can be released in the air [5].

9.1 Classification of 8 pollen categories with false alarms

In order to simulate the performance of the recognition system, we assume that the detector is able to identify 83% of pollen grains and that for each pollen grain he gives 10 false alarms, see tab. 5.9. Thus, we are using in our experiments 2576 pollen samples and 26000 samples of false alarms.

The false alarm class is very variable. Then, we can assume that a sample of false alarm will give a low posterior probability in each model of pollen class. This means that we can think to classify the false alarms simply using a threshold on the maximum posterior probability. In this case, a correct classification of false alarms must put these samples in the *unknown* class.

Keeping the parameters and the models found during the tuning of only the pollen classes (see par. 7.5.2 and tab. 9.2), we have only to find a proper threshold on the posterior probability. We have run experiments varying this threshold and the achieved results are shown in Figure 9.1.

The final result is given in Figure 9.2; the performance is not good because we have to use an high value for the threshold on the posterior probability in order to recognize the false alarms but this penalizes the pollen classification. With respect to the performance without the false alarm class, the error rate is nearly doubled.

parameters	value
dimensions	14
nr. components	2
covariance	diag.
prob. thresh.	0.9

Table 9.2: Classification of 8 pollen species without modeling the false alarm class: parameters found thanks to tuning stage which also provided us a good set of initial conditions for the random number generator used to perform EM algorithm.

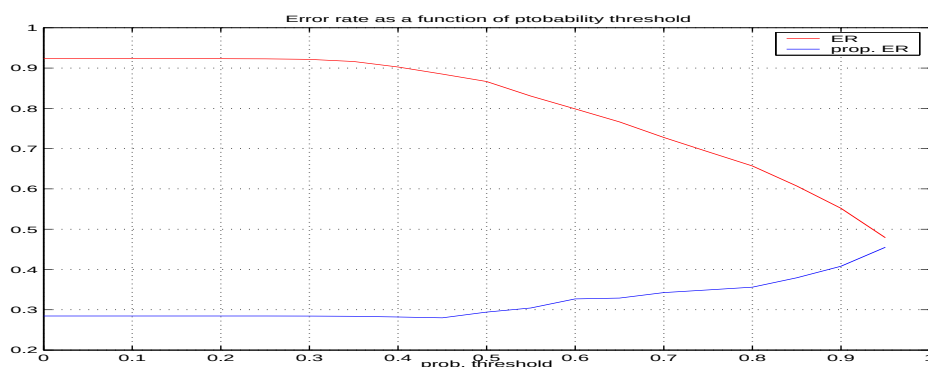


Figure 9.1: Classification of 8 pollen species without modeling the false alarm class: error rate as a function of the threshold on the posterior probability. The red line shows the usual error rate defined as the ratio between the number of misclassified images and the number of tested images (in percentage). This is mainly given by the classification performance on the false alarm class since we are testing 10 false alarms per each pollen grain. The blue line shows the proportional error rate defined as the sum of each class error rate. This is more sensitive to misclassification of pollen samples. In our experiment, we have chosen a threshold of 0.9.

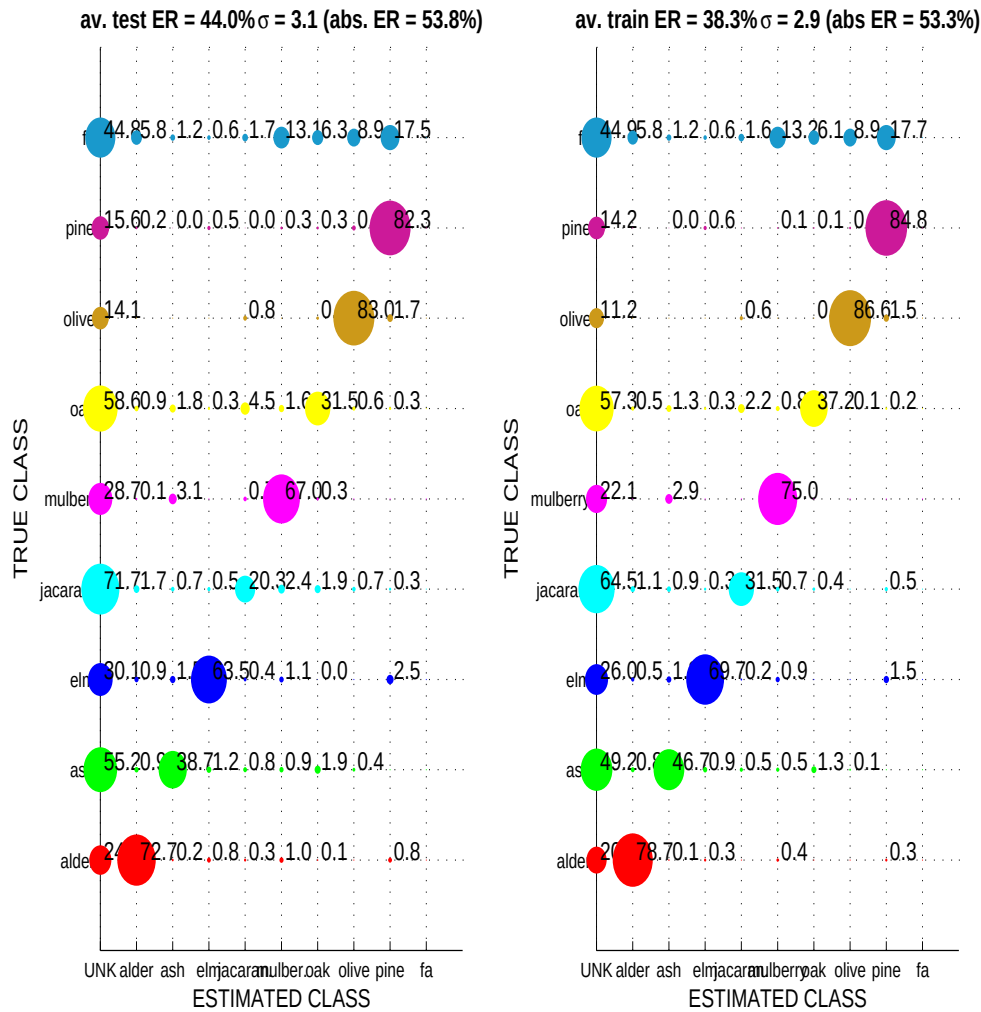


Figure 9.2: Classification of 8 pollen species without modeling the false alarm class: average test (left side) and training (right side) confusion matrices of 100 experiments in which good detector patches were used. In these experiments, we have randomly extracted the 10% of images for test at each iteration.

Finally, we can try to model the false alarm class. Since we have available a lot of samples belonging to this class and since these samples show an extreme variability in appearance, we can build for this class a more complicate model. The tuning proceeds in the following steps:

1. we run experiments to find a good number of dimensions, number of components in the MoGs, structure of covariance matrix for the pollen classes keeping the same model for the false alarm class; we used for this class a mixture with 4 full covariance Gaussians (the number of dimensions was varied during the experiments)
2. keeping the best parameters for the pollen classes, we tried many values for the number of components in the mixture model of the false alarm class
3. we run 100 experiments in order to find the best initial conditions for EM algorithm
4. we varied the threshold on the posterior probability (see Figure 9.3)

The best performance was achieved with parameters listed in tab. 9.3. The last step is to find the average error rate of 100 experiments. Note that we assume that a correct classification for a false alarm sample occurs when it is assigned either to the false alarm class or to the unknown class. As usual for the last experiment, at each iteration we randomly select the 10% of images for test. The promising performance is shown in Figure 9.4.

parameters	value
dimensions	14
components poll. classes	2
covariance poll. classes	spher
components f.a. class	4
covariance f.a. class	full
prob. thresh.	0.5

Table 9.3: Classification of 8 pollen species when modeling the false alarm class: parameters found thanks to tuning stage which also provided us a good set of initial conditions for the random number generator used to perform EM algorithm.

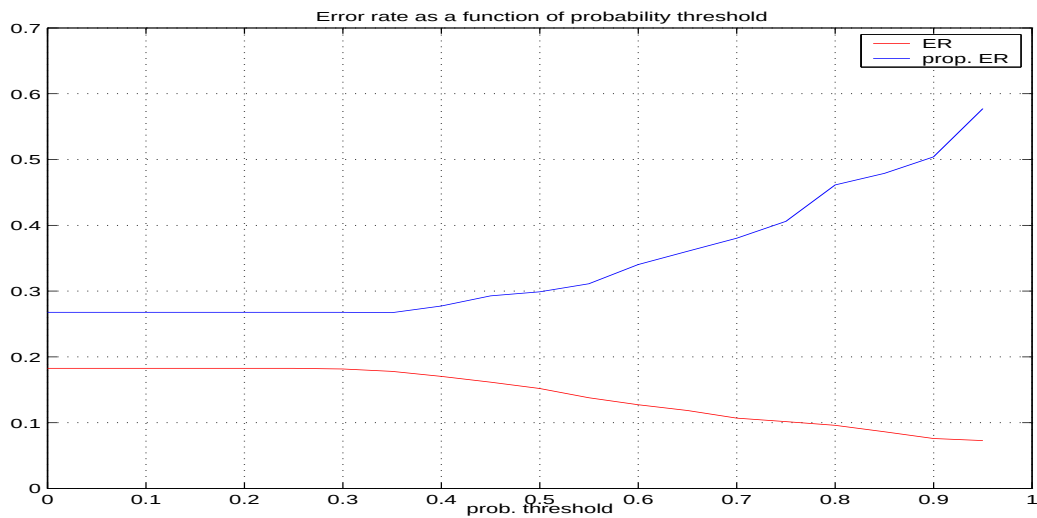


Figure 9.3: Classification of 8 pollen species when modeling the false alarm class: error rate as a function of the threshold on the posterior probability. The red line shows the usual error rate defined as the ratio between the number of misclassified images and the number of tested images (in percentage). This is mainly given by the classification performance on the false alarm class since we are testing 10 false alarms per each pollen grain. The blue line shows the proportional error rate defined as the sum of each class error rate. This is more sensitive to misclassification of pollen samples. In our experiment, we have chosen a threshold of 0.5.

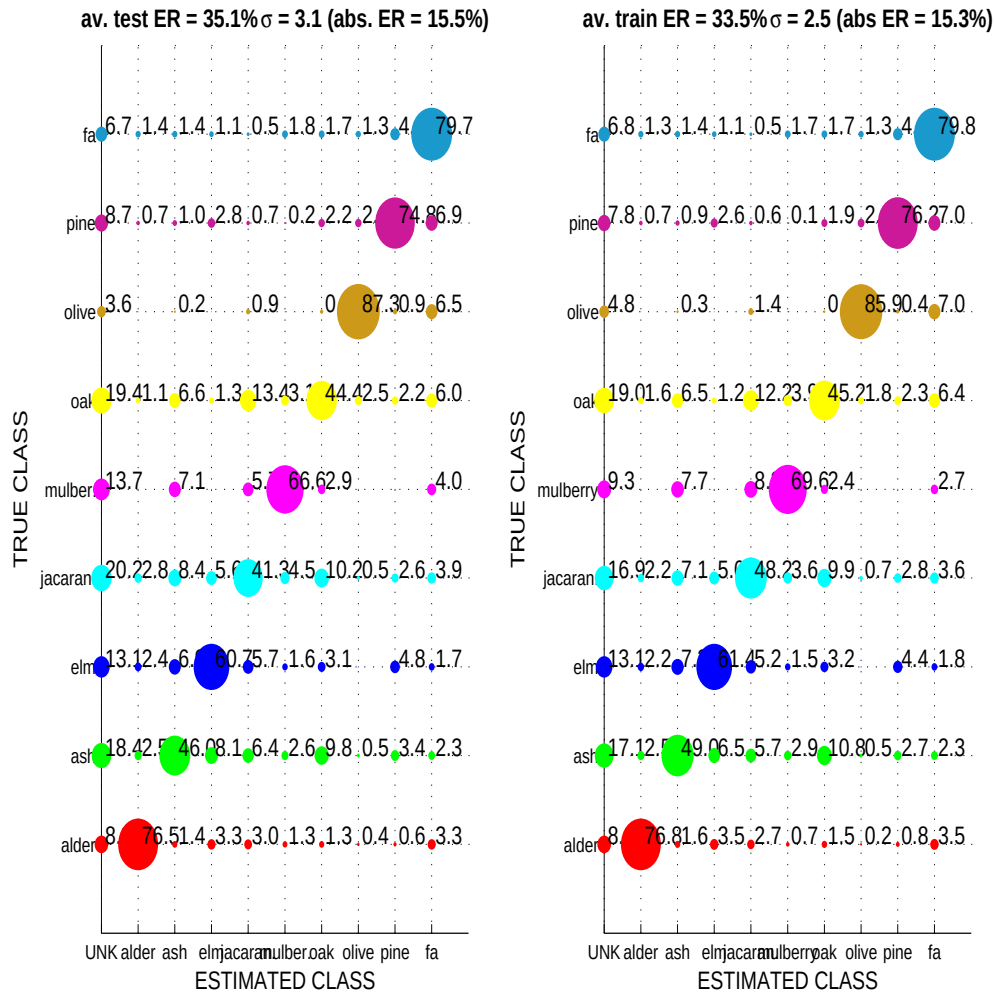


Figure 9.4: Classification of 8 pollen species when modeling the false alarm class: average test (left side) and training (right side) confusion matrices of 100 experiments in which good detector patches were used. In these experiments, we have randomly extracted the 10% of images for test at each iteration.

9.2 Classification during specific seasons

In any practical application of airborne pollen recognition, it is useful to know what types occur together at various times of the year in the area of interest. In Pasadena area, in September there is chinese elm and almost nothing else; in January, ash, alder, pine pollen grains are commonly found [5]. We will never encounter a period of the year with all the 8 kinds of pollen considered in the previous section.

Then, it is interesting to see what the performance would be in a particular season.

In September, we have to distinguish between chinese elm and false alarms. Since we have 466 samples of this pollen, we take 4660 samples belonging to the false alarm class. We proceed like in the previous section modeling the false alarm class. The tuning stage provided us the parameters listed in tab. 9.4. Chosing a threshold value of 0.65, we achieved an error rate of about 6% as shown in Figure 9.6.

parameters	value
dimensions	8
components poll. classes	1
covariance poll. classes	diag
components f.a. class	5
covariance f.a. class	full
prob. thresh.	0.65

Table 9.4: September month: parameters found thanks to tuning stage which also provided us a good set of initial conditions for the random number generator used to perform EM algorithm.

In January, we have to distinguish among alder, ash, pine and obviously false alarms. Since we have 1264 samples of these pollen grains, we take 12600 samples belonging to the false alarm class. The tuning stage provided us the paramenters listed in tab. 9.5. Chosing a threshold value of 0.7 (see Figure 9.7), we achieved an error rate of about 17% as shown in Figure 9.8.

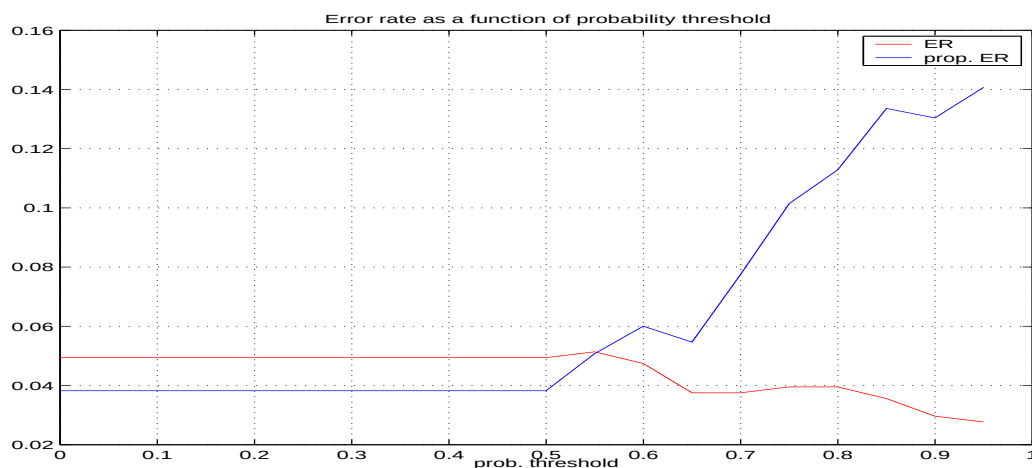


Figure 9.5: September month: error rate as a function of the threshold on the posterior probability. The red line shows the usual error rate defined as the ratio between the number of misclassified images and the number of tested images (in percentage). This is mainly given by the classification performance on the false alarm class since we are testing 10 false alarms per each pollen grain. The blue line shows the proportional error rate defined as the sum of each class error rate. This is more sensitive to misclassification of pollen samples. In our experiment, we have chosen a threshold of 0.65.

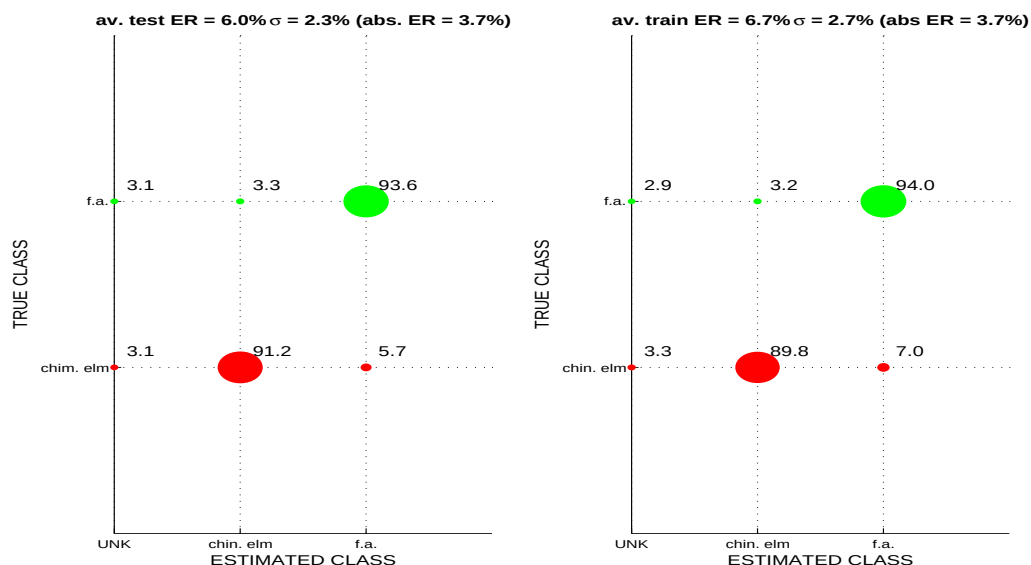


Figure 9.6: September month: average test (left side) and training (right side) confusion matrices of 100 experiments in which good detector patches were used. In these experiments, we have randomly extracted the 10% of images for test at each iteration.

parameters	value
dimensions	8
components poll. classes	1
covariance poll. classes	full
components f.a. class	10
covariance f.a. class	full
prob. thresh.	0.7

Table 9.5: January month: parameters found thanks to tuning stage which also provided us a good set of initial conditions for the random number generator used to perform EM algorithm.

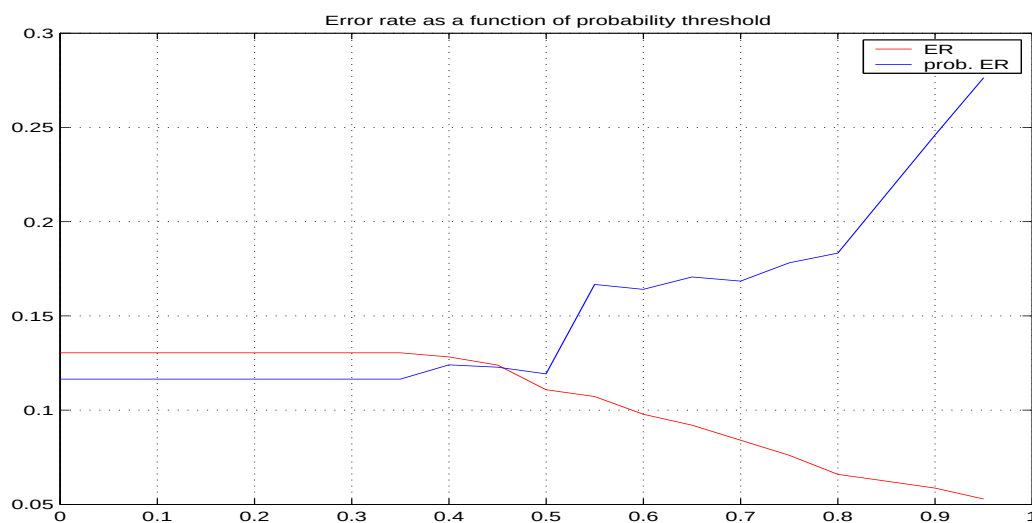


Figure 9.7: January month: error rate as a function of the threshold on the posterior probability. The red line shows the usual error rate defined as the ratio between the number of misclassified images and the number of tested images (in percentage). This is mainly given by the classification performance on the false alarm class since we are testing 10 false alarms per each pollen grain. The blue line shows the proportional error rate defined as the sum of each class error rate. This is more sensitive to misclassification of pollen samples. In our experiment, we have chosen a threshold of 0.65.

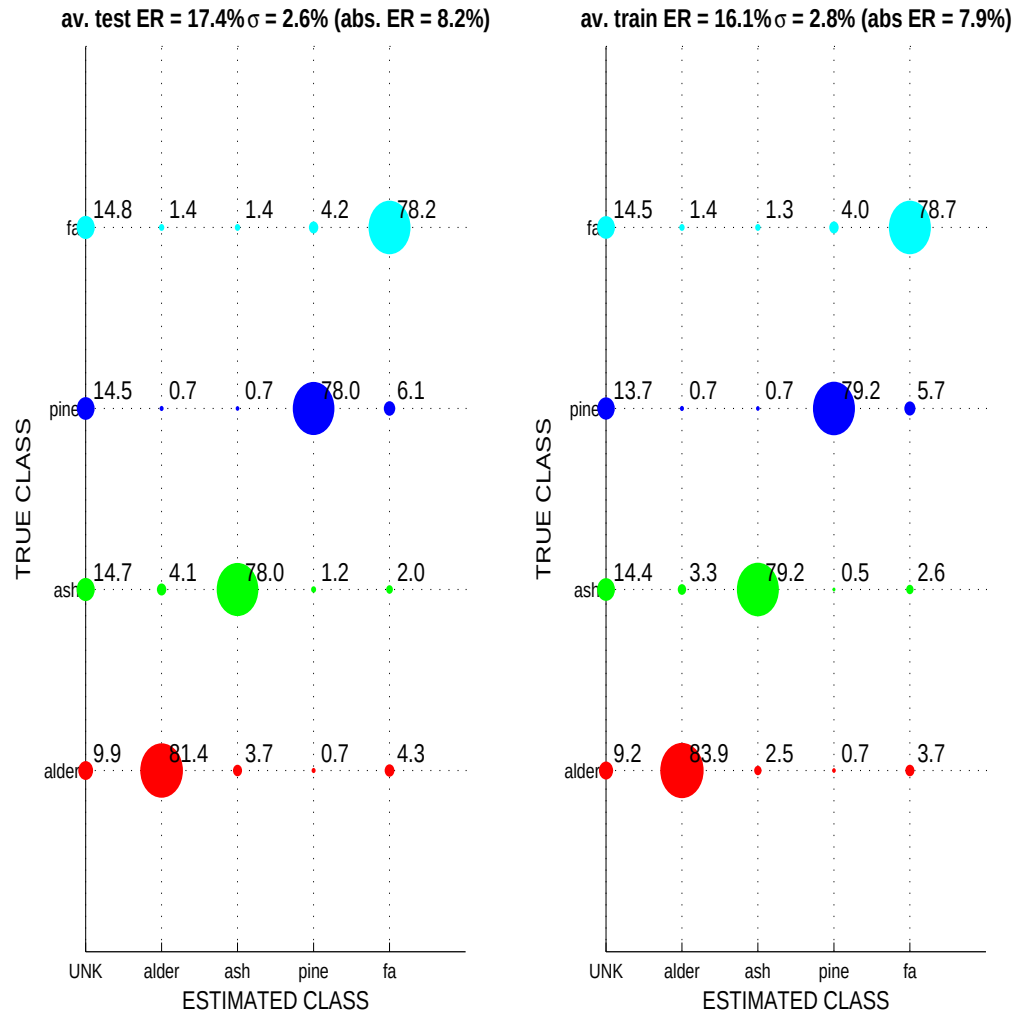


Figure 9.8: September month: average test (left side) and training (right side) confusion matrices of 100 experiments in which good detector patches were used. In these experiments, we have randomly extracted the 10% of images for test at each iteration.

9.3 Conclusion

In this chapter it was described the performance of the whole system on images of pollen captured by spore trap.

The detector is able to detect with probability of 0.94 all pollen grains. It is able to determine a proper bounding box for the 83% of the pollen grains that were labeled by experts. It detects also for each pollen ten other particles that are not pollen. With the spatial resolution of our images this results are quite good.

The next stage is classification. When the system learns to classify not only the pollen categories but also the junk class, the error rate is about 35% when considering 8 pollen classes. The false alarm class is described by a more complex model and we achieve on this category the best correct classification rate: nearly 90%.

The experiments are done in conditions which are very close to a real situation. Particularly, we keep the ratio between pollen and junk particles at one out of ten and the classifier receives objects in the right proportion. In this way, we overcome the difficulty to select images for training and to classify pollen particles that are not modeled because their number is too low.

This system has a better performance when considering few pollen classes. Thanks to the counts of this last years in Pasadena area, we know what kind of pollen particles can be released in each season and we can use this information to improve the performance of the system. For example, in September there is mainly a release of chinese elm and we achieve an error rate of about 6%; in January there are only three kinds of pollen and the error rate is around 17%. These promising results make the system already attractive for a practical application in airborne pollen recognition.

Chapter 10

Conclusion

In this report, it was discussed a system for visual recognition of biological particles. Our interest is justified by the need to automatically do microscopic analysis. This is because manual analysis is slow, expensive, not precise and not standardized. We worked on two databases: the first one is a collection of particles in urine, the second one is a collection of sticky tape images containing many genera of pollen.

The recognition system can be divided into two stages: detection and classification. Detection aims to find points of interest in an image, in which it exists at least a small probability of a useful particle presence represented by those spots. Classification receives patches from the detector; particularly, it has to skip particles that are not of interest and for the rest to determine their category.

We developed a detector based on a filtering approach which gave very good results on pollen database. It is able to detect with probability 9/10 the 23 species of pollen in sticky tape pictures taken by light microscope. The percentage of false alarm is about 1000%. Looking at the images in our database, this performance seems very good. Even an expert has to focus his attention on nearly 10 points in each image to establish the presence of a pollen. On the other hand, if we look at the patches that this detector is able to extract, we have to decrease the percentage of detection. When dust is close to a pollen grain or when pollen grains are attached in a single blob, the detection algorithm can fail and the same is true for the algorithm we are using to determine a proper bounding box. If we consider only the patches that can be reasonably classified, we can say that the 83% of pollen grains have been detected and have been properly segmented. This result is reliable because we have used 1429 images with 3686 pollen grains. Looking at some very noisy images in this database, we can be satisfied of such performance. Furthermore, this technique is quite fast since the analysis of one image takes

up 8 seconds on a current 2.5GHz Pentium processor computer.

The simplest classifier with the best performance is the Bayesian classifier using mixture of Gaussians to model the training data. We developed a new kind of features based on local jets [10]; these features prove to capture almost all the essence of input images because we were able to achieve an error rate of about 7.6% on urine database and 20% on 8 kinds of pollen categories. This approach is interesting because it is segmentation free, very general and able to handle very low contrast particles.

On the other hand, the system could be further improved using more complex technique, which extracts from points in foreground information about texture, shape and color.

Moreover in a real application, the system often has to handle particles that are not of interests, for example in pollen recognition most of the patches contain junk particles. This class should be very well classified if we admit an high number of false alarms in detection. This is a challenge to consider in the design of a good classifier, and in our experiments we took into account this class when we considered the pollen classification task. We achieved an error rate of about 35% when classifying 8 pollen categories and the false alarm class, and an error rate of 17% when classifying 3 pollen categories and the false alarm class.

Our contribution in classification of urine particles relies on the definition of a new set of powerful features that are able to extract information from patches of about 60x60 pixels without any segmentation. Thanks to the simplicity of our system, we are able to classify very quickly particles; nearly 10ms are required to analyze a particle on a current 2.5GHz Pentium processor computer.

In pollen recognition, our work is the first research towards a feasible system for real-time pollen count in which detection and classification are combined to recognize several kinds of pollen.

We used images taken with a common light microscope and our database is built using the common equipment for pollen collection, namely the volumetric spore trap. Unfortunately, the collection of airborne pollen grains is very slow and the available database is still quite small. However, these first results seem very promising. The performance can be acceptable if we take into account the seasonal information about pollen release in the air because this can reduce the number of classes involved in the classification.

In our opinion, urinalysis could be improved in the future if a general and segmentation free approach is applied, and if we reserve customized

operations only in a second stage for the most difficult samples. Pollen recognition needs further research but several steps have been made in the right direction. We think that in the next months a new software could be written based on this work to detect pollen particles and make hypothesis of classification. This should be a tool to help experts, who are building the current database and speed-up the process of labelling. It is reasonable to foresee that in the next few years a completely autonomous system for automatic and reliable recognition of pollen could be built using as input the sticky slides of volumetric spore trap. In the future, a new kind of collection technique able to produce input images with less debris is also possible, and will certainly improve the performance of the visual recognition system. This will yield to a real-time and low cost measurement instrument for pollen count which will definitely make easier and better the life of many allergic people.

Bibliography

- [1] <http://www.aaaai.org>.
- [2] <http://medlib.med.utah.edu/WebPath/TUTORIAL/URINE/>.
- [3] <http://web.engr.oregonstate.edu/~tgd/software/ecoc-codes.tar.gz>.
- [4] <http://www.cis.tugraz.at/igi/aschwaig/software.html>.
- [5] <http://www.che.caltech.edu/groups/rcf/DataFrameset.html>.
- [6] The iq200 automated urinalysis system. Brochure IRIS.
- [7] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford Univ. Press, 2000.
- [8] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [9] M.C. Burl, L. Asker, P. Smith, U. Fayyad, P. Perona, L. Crumpler and J. Aubele. Learning to recognize volcanoes on venus. In *Machine Learning*, volume 30, pp. 165–194, Boston, Feb-Mar 1998. Kluwer Academic Publisher.
- [10] R. Mohr C. Schmid. Local grayvalue invariants for image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, (19(5)):530–535, 1997.
- [11] N. Cristianini and J. Shawe-Taylor. *Support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.
- [12] J. Dahmen, J. Hektor, R. Perrey and H. Ney. Automatic classification of red blood cells using gaussian mixture densities. In *Proc. Bildverarbeitung für die Medizin*, pp. 331–335, Munich, Germany, 2000.

- [13] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 1995.
- [14] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, New York, 1973.
- [15] I. France, A.W.G. Duller, H.F. Lamb and G.A.T. Duller. A comparative study of approaches to automatic pollen identification. *BMCV*, 1997.
- [16] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, 1996.
- [17] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, second edition, 1990.
- [18] C. Harris and M. Stephens. A combined corner and edge detector. Manchester, 1998. 4th Alvey Vision Conference, pp. 189-192.
- [19] J.J. Koenderink and J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, (55):367–375, 1987.
- [20] M. Langford. *Some applications of digital image processing for automation in palynology*. Ph. D. Thesis, University of Hull, 1988.
- [21] M. Langford, G.E. Taylor and J.R. Flenley. The application of texture analysis for automated pollen identification. Conference on identification and pattern recognition, 1986.
- [22] D.G. Lowe. Object recognition from local scale-invariant features. In *International conference on computer vision*, Corfù, Sept. 1999.
- [23] D.G. Lowe. Distinctive image features from scale-invariants keypoints. submitted for publication, June 2003.
- [24] J.R. Flenley M. Langford, G.E. Taylor. Computerised identification of pollen grains by texture. *Review of paleibotany and palynology*, 1990.
- [25] P. Perona. Note on fisher linear discriminants, 2000.
- [26] M. Ranzato. *Automatic visual recognition of biological particles*. Masterthesis, Univ. di Padova, 2004.

- [27] O. Ronneberger, U. Heimann, E. Schultz, V. Dietze, H. Burkhardt and R. Gehrig. Automated pollen recognition using gray scale invariants on 3d volume image data. Vienna, Austria, Sept 2000. Second European Symposium on Aerobiology.
- [28] R.E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth Conference*, 1997.
- [29] R.E. Schapire. The boosting approach to machine learning an overview. In *MSRI workshop on nonlinear estimation and classification*, 2002.
- [30] E. Grant Smith. *Sampling and identifying allergenic pollens and molds*. Blewstone Press, San Antonio, Texas, 2000.
- [31] A. Torralba and A. Oliva. Statistics of natural image categories. *Network: Computation in neural systems*, (14):391–412, 2003.
- [32] E.L. Vezey and J.J. Skvarla. Computerised feature analysis of exine sculpture patterns. *Review of paleibotany and palynology*, 1990.