**IEEE Transactions on System, Man and Sybernetics – B**

# Clustering in Image Space for Place Recognition and Visual Annotations for Human-robot Interaction

**Aleix M. Martínez** *(Corresponding author)*
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
aleix@ecn.purdue.edu

**Jordi Vitrià**
Centre de Visió per Computador
Universitat Autònoma de Barcelona
Edifici O, 08193 Bellaterra (Barcelona)
jordi@cvc.uab.es

## Abstract

*The most classical way of attempting to solve the vision-guided navigation problem for autonomous robots corresponds to the use of 3D geometrical descriptions of the scene; what is known as model-based approaches. However, these approaches do not facilitate the user's task because they require that geometrically precise models of the 3D environment be given by the user. In this paper, we propose the use of "annotations" posted on some type of blackboard or "descriptive" map to facilitate this user-robot interaction. We show that using this technique user commands can be as simple as "go to label 5".*

*To build such a mechanism, new approaches for vision-guided mobile robot navigation have to be found. We show that this can be achieved by means of mixture models within an appearance-based paradigm. Mixture models are more useful in practice than other pattern recognition methods such as PCA (Principal Component Analysis) or FDA (Fisher Discriminant Analysis - also known as Linear Discriminant Analysis, LDA), because they can represent non-linear sub-spaces.*

*However, given the fact that mixture models are usually learned using the EM (Expectation-Maximization) algorithm which is a gradient ascent technique, the system cannot always converge to a desired final solution, due to the local maxima problem. To resolve this, a genetic version of the EM algorithm is used. We then show the capabilities of this latest approach on a navigation task that uses the above describe "annotations".*

**Keywords:** Mobile robot navigation, computer vision, pattern recognition, user-robot interaction, EM algorithm and genetic algorithms.

## 1 Introduction

One of the ultimate goals in robotics is to create autonomous robots. The word autonomous means that robots must accept high-level descriptions (orders) of tasks and execute them without any further human help. In this context, the goal of autonomous robotics is: to create systems that accept high-level input descriptions specifying *what* the user wants done rather than *how* to do it [17].

The building of visual systems that allow a mobile robot to autonomously navigate, interact and analyze its surroundings is of major interest in making progress towards this goal. In this paper we review the most common approaches of the appearance-based paradigm for vision-guided navigation problems and propose two new methods based on clustering data in the image space.

The appearance-based methods for robot navigation are not the most common approaches encountered in the literature though. Indeed, the most classical way of attempting to solve the vision-guided navigation problem is the use of 3D geometrical descriptions of the surroundings, i.e. model-based approaches. To do that, the system must first generate a three-dimensional representation of its surroundings and, second, recover the transformation that relates this model and the incoming image. In pioneering work, Moravec [25, 26] developed a vision-guided navigation system capable of recovering depth maps from a stereo system while this was moved along a track. Its main disadvantage, however, was that simple image distortions (or changes) would dramatically make the system fail in its identification task (i.e. it could not handle uncertainty). Both Ayache and Faugeras [2] and Kosaka and Kak [15], solved this problem by using a new approach based on Kalman filters. However, whereas the work of Ayache and Faugeras was more related to the construction of 3D representations from scenes of the robot surroundings, Kosaka and Kak focused their interest on how to use these maps to autonomously navigate around an already known (or learned) environment. In so doing, they built a navigation system able to navigate at the speed of 8 to 10 m/min (using an ordinary computer of that time – around 16 MIPS power).

However, although several of these systems can navigate in indoor environments and some of them can even handle some types of uncertainty, they require geometrically precise models of the 3D environment to be available. Moreover, all these model-based systems seem too complicated compared to how the human being's navigation system works. It is widely agreed now, that a navigation system should be able to learn and recognize its environment in a much simpler manner. In the context of this contribution, navigation is only involved in the act of "moving around a previously unknown environment". [1] It is important that we require these actions to be fast, allowing other "high-level" tasks (e.g. reasoning) to achieve other goals. These systems (generally) allow easier interaction between the system (robot) and the user, because it is easier to say "go to the third door of this hallway", than having to precisely describe a 3D map of the entire environment of the robot.

Following these ideas we propose a new vision-guided system based on annotations. Annotations, in the context of this article, are defined as: "specific locations (areas) of the environment of the robot which are labeled with a specific (*symbolic*) name. These names are used to explain the actions that the robot performs and can be used to accept high-level user commands".

Our main motivation is the construction of an autonomous navigation system able to interact with other tasks or with the user in an easy manner. Annotations simplify the act of moving a robot from the current point to a final position, say *office C-03*, because they allow the use of sentences such as: "go to office C-03", as shown in Figure 1 for clarity. Annotations also allow the system to describe what the robot is doing, has been doing recently, or will shortly do, using high-level sentences of the kind: "I am going from *office C-15* to *office C-03*. Currently, I am next to *the library*". Obviously, this kind of communication is more desirable than having to precisely search for a 3D-position in a 3D-map.

[1] This contribution is focused on the indoors problem exclusively.

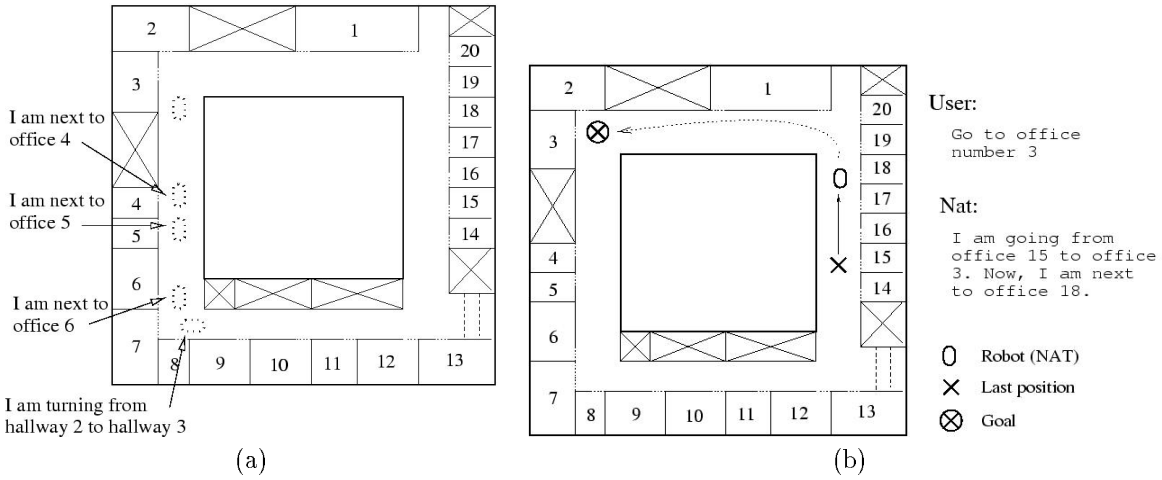



Figure 1: (a) Annotations involve the use of labels. The robot annotates all actions it is doing once a subgoal or sub-state has been reached. (b) An example of navigation using annotation. First the user specifies to the robot the place it has to go next. Second, the robot starts to go in this direction. While, the robot is moving, it makes annotations, i.e. it explains (using high-level descriptions) what it is doing and where it currently is within its environment.

To achieve such a goal, our robot has to be designed as a system able to recognize its current position and infer the following direction to go (in order to reach a given final goal). The system must also be able to recognize the final goal once there and announce this. Additionally, the user's task should be as simple as driving the robot around its environment in order to allow the robot to learn its working-space. Annotations are to be learned by the robot while this learning process takes place. The user can name these annotations (labels) with symbolic names that he/she is comfortable with.

As discussed above, in order to achieve such a learning mechanism of visual data, different approaches (to that of model-based) should be defined. Recent studies of appearance-based methods applied to robotics have shown that different well-known pattern recognition techniques, such as PCA (Principal Component Analysis) and FDA (Fisher Discriminant Analysis – also known as Linear Discriminant Analysis, LDA), can successfully and efficiently represent real visual data [31, 39]. The great advantage of appearance-based methods is that it is not necessary to define a representation or model for a particular scene since the surroundings are implicitly defined by the selection of the sample images.

Nayar *et al.* [31] used the eigenspace for learning the visual data obtained by an industrial robot. The applications reported are positioning, tracking and inspection (to ensure that all components of a manufactured object are present and correctly positioned). Weng [39] proposed the use of PCA and FDA on top of PCA to autonomously learn to navigate along a hallway. Notice that PCA is an unsupervised technique, whereas FDA is supervised.

Unfortunately, the use of these techniques (as will be shown and discussed in the following sections) is not always well founded, because (in practice) the representative spaces obtained are normally embedded in non-linear manifolds. Given the fact that PCA and FDA only search for features that can be obtained by a linear transformation of the original feature space, PCA and FDA will never achieve the learning of non-linear representations correctly. One way to resolve this problem consists in using density model representations. In this context, we can describe (and so discriminate) by fitting a separated probability density model to each class and then picking the class that assigns the highest density to the test data. A common way to do this is by means of mixture models [21] learned using the Expectation-Maximization (EM) algorithm [9]. However, a limitation stems from the fact that the EM algorithm, which is a hill-climbing method, only converges to a local maximum. This problem is critical to our application, because a local maximum usually corresponds to a wrong outcome of the mixture model (and this implies incorrect learning and, therefore, incorrect recognitions). For this reason

a new EM algorithm based on GA (Genetic Algorithms) [19] is used to learn the group of mixture models that describe a certain location of the working-space. Each of the models of the mixture will then represent a linear subspace embedded in the original space. Each of these linear subspaces can now be assigned to a label (symbol), and this can be used as an annotation to facilitate the user-robot communication as described above.

In this communication we show comparisons between the classical PCA and FDA methods and our two new clustering methods. We show that when the training dataset is sufficiently large and correctly samples the information of the working-space, the two new proposed clustering methods outperform both the FDA and PCA approaches. We also show that FDA is superior to PCA for the case of large and representative datasets; as it has also been discussed in [18] in the context of face recognition.

We finally show experimental results of the human-robot communication based on annotations. We have run experiments in 4 different days. In the first day, the system learned the working-space area. For the second, third and fourth days the robot was asked to perform several actions based on the visual annotations learned during the first session. Of the experimental results reported in this manuscript, 96.67% were satisfactory.

This article is organized as follows. In section 2 we introduce PCA and FDA and their use for mobile robot navigation. Section 3 introduces mixture models and the genetic version of the EM algorithm that allows the learning of those mixtures. The use of these two methods for learning and recognition purposes is also reported in this section. Results comparing the four described methods (i.e. PCA, FDA, classical mixture models and genetic mixture models) are reported and discussed. Section 4 describes the new proposed architecture that uses "annotations" to allow easier communications between the navigation system and the user. Experimental results are also reported. We conclude in section 5.

## 2 Appearance-Based Methods for Robot Navigation

As correlation techniques are computationally expensive and require a great amount of storage, it is natural to pursue dimensionality reduction schemes. A technique now commonly used for dimensionality reduction in computer vision is PCA [12]. PCA has been previously used in face recognition [16, 38, 23, 20], lip reading [8], character recognition [30], data compression [34], and generic object recognition [29, 32]. Recently, PCA was proposed as a tool to represent visual data acquired from industrial robots [31] and mobile robots [39].

### 2.1 PCA for visual data representation

PCA, also known as Karhunen-Loéve expansion methods, searches for those features of the original (high-dimensional) space that best describe the data. However, a limitation stems from the fact that we seek only features which can be obtained by a linear transformation of the original variables.

More formally, let us consider a set of $N$ sample images $\{\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_N\}$ taking values in an $n$-dimensional image space, i.e. each raw image, $I_i$ of $c$ columns and $r$ rows, is represented in a vector, $\hat{\mathbf{x}}_i$, of $n$ pixels (where $n = c \cdot r$), by reading pixel brightness values in a raster scan manner. It is assumed that the imaging sensor used for recognition has a linear response, i.e. image brightness is proportional to scene radiance. It is also desirable that our recognition system be unaffected by variations in the intensity of the illumination or the aperture of the imaging acquisition system. This can be achieved by normalizing each image in such a way that the total energy contained in the entire image is unity, i.e. $||\mathbf{x}_i|| = 1$. This brightness normalization transforms each measured image $\hat{\mathbf{x}}_\mathbf{i} = [\hat{x}_{i,1}, ..., \hat{x}_{i,n}]^T$ to a normalized image $\mathbf{x}_i = [x_{i,1}, ..., x_{i,n}]^T$, where $x_{i,j} = \frac{\hat{x}_{i,j}}{A_i}$ and $A_i = \sqrt{\sum_{j=1}^{n} \hat{x}_{i,j}^2}$. Also the average $\bar{\mathbf{x}}$ of all vectors in the set is subtracted from each intensity illumination normalized vector $\mathbf{x}_i$. This ensures that the eigenvector with the highest eigenvalue represents the dimension in the eigenspace in which variance of vectors is maximum in a correlation sense. These new normalized vectors describe a new set of $N$ normalized images $\mathbf{X} = \{\mathbf{x}_1 - \bar{\mathbf{x}}, ..., \mathbf{x}_N - \bar{\mathbf{x}}\}$ taking values in a $n$-dimensional space. We can define the *covariance* matrix as:

$$\mathbf{Q} = \mathbf{X}\mathbf{X}^{\mathbf{T}} \tag{1}$$

The $m$ eigenvectors, $\mathbf{e}_i$, associated with the $m$ highest eigenvalues, $\lambda_i$, of this huge matrix of $N \times N$ dimensions, will define the linear transformation $\mathbf{E}$ mapping the original $n$-dimensional space into the low $m$-dimensional feature space (normally, $m \ll n$). These eigenvectors and associated eigenvalues of the covariance matrix $\mathbf{Q}$ can be determined by solving the well-known eigenstructure decomposition problem:

$$\lambda_i \mathbf{e}_i = \mathbf{Q}\mathbf{e}_i \,. \tag{2}$$

Then, $\mathbf{E} = [\mathbf{e}_1, ..., \mathbf{e}_m]^T$.

Computing the eigenvectors of a large *covariance* matrix can be computationally intractable in reality with modest hardware requirements. However, efficient algorithms do exist to solve this problem, e.g. [28, 29]. This leads to our first learning and recognition algorithm, named the PCA-Algorithm:

**PCA-Algorithm: Learning**

1. **Obtain images of the surroundings**: This can be achieved by acquiring images on different days, at different hours within the same day, and within the same hour. It is well-known that due to slippage of the wheels or any other environmental distortion, the robot will almost never do the same run.[2] Grab an image for each small displacement, $\epsilon$, of the robot. The value of $\epsilon$ should be kept small, so that consecutive images are strongly correlated and in such a way that the use of pattern recognition techniques (such as PCA) are best founded.
2. **Vectorize and normalize**: Let us denote each vectorized image as $\hat{\mathbf{x}}_i^{(x,y)}$, where $(x, y)$ represents the 2D-environmental location from where the image was grabbed (we assume $z = 0$), and $i$ represents the $i$th time we obtain an image from this position. Normalize all vectors, such that $\|\mathbf{x}_i^{(x,y)}\| = 1$ $(\forall\, i,\ (x, y))$, and compute the average $\bar{\mathbf{x}}$
3. **Computing the covariance matrix**: Create the set of all $N$ obtained vectors $\mathbf{X} = \{\mathbf{x}_1^{(x_1,y_1)} - \bar{\mathbf{x}},\ ...,\ \mathbf{x}_{N_1}^{(x_1,y_1)} - \bar{\mathbf{x}},\ \mathbf{x}_1^{(x_2,y_2)} - \bar{\mathbf{x}},\ ...,\ \mathbf{x}_{N_q}^{(x_q,y_q)} - \bar{\mathbf{x}}\}$. Where $N_1 + ... + N_q = N$, and typically $N_i = N_j\ \forall\, i, j$. Then, build the *covariance* matrix $\mathbf{Q}$ from equation (1)
4. **Eigenspace**: Calculate the eigenvalues and eigenvectors of $\mathbf{Q}$ by solving equation (2). Store in memory the projecting matrix $\mathbf{E}$. Project all $N$ normalized vectors to this obtained eigenspace and store the $m$-dimensional samples in memory.

**PCA-Algorithm: Recognition**

1. Grab an image of the environment from the current (unknown) position and vectorize it, $\mathbf{x}^{(x_{unk},y_{unk})}$.
2. Obtain the normalized vector $\mathbf{y} = \mathbf{x}^{(x_{unk},y_{unk})} - \bar{\mathbf{x}}$
3. Project this image onto the $m$-dimensional eigenspace by computing the product of the resulting vector $\mathbf{y}$ with the projecting matrix $\mathbf{E}$, i.e. $\mathbf{g} = [\mathbf{e}_1, ..., \mathbf{e}_m]^T\mathbf{y}$.

[2] Notice that what are difficult problems in many other systems (e.g., slippage of the wheels and environmental distortions), is used here to obtain a large diversity of images (which ultimately aid to the learning system).

4. Find the closest sample to $\mathbf{g}$ in the eigenspace, i.e. the nearest neighbor based on the $L_2$-norm. To do that we use a k-d tree, a recording cases method described in [40]. This decreases the time need to search the closest sample.

## 2.2 From the most descriptive data to the most discriminant data

As seen above, PCA searches for those features of the space that best describe the data obtained from the environment. That implies that the distance between any two images in the original (raw) space is proportional to the distance obtained in the new low-dimensional eigenspace [12]. However, this has been shown to not be optimal for classification. For example, in the context of human face images, it has been proven [1] that images of different people taken under the same illumination conditions are closer to one another than images of the same individual taken at different illumination conditions.

When large samples are available, Fisher Discriminant Analysis (FDA or Linear Discriminant Analysis, LDA) should be a better choice, because FDA searches for those features (or combination of features) which best discriminate between classes (in a linear sense) [10, 12]. FDA has been previously used for face recognition [37, 5], and robot navigation [39]. It has been shown that although FDA is not guaranteed to outperform PCA where the training dataset is small or under-samples the true distributions of the classes, FDA is expected to outperform PCA where large and representative datasets are available [18].

Although the data obtained from the robot surroundings is generally taken under uncontrolled environmental conditions, the position $(x, y)$ of the robot within the environment (working-space) can approximately (or even quite precisely with the help of a teacher) be known. The class at which each sample belongs, can be readily inferred then. One option consists of (automatically) assigning a distinct class value at each $a \cdot \epsilon\ cm$ of displacement of the robot (where $a$ is a constant). Another option would imply the use of a teacher, who would manually assign a class label for each acquired sample.

Formally, given a number of independent features relative to which data is described, FDA creates a linear combination of these which yields the largest mean difference between the desired classes. Mathematically speaking, for all samples of all classes, we define a measure called *between-class* and a measure called *within-class*. While the between-class measure attempts to maximize the distances among classes, the within-class scatter attempts to minimize the distances among the samples of the same class. If $\mathbf{x}_i^{(x_j,y_j)}$ is the normalized vector corresponding to the $i$th image obtained from the $(x_j, y_j)$ 2D-world position (that is to say, the $i$th example of class $(x_j, y_j)$) and, $\mu_j$ is the class mean of this class, we can define the within-class scatter matrix as:

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (\mathbf{x}_i^{(x_j,y_j)} - \mu_{(x_j,y_j)})(\mathbf{x}_i^{(x_j,y_j)} - \mu_{(x_j,y_j)})^T \tag{3}$$

where $\mu_{(x_j,y_j)}$ is the mean of class $(x_j, y_j)$, $c$ is the total number of classes, and $N_j$ the number of samples in class $(x_j, y_j)$. The between-class scatter matrix is defined as:

$$S_b = \sum_{j=1}^{c} (\mu_{(x_j,y_j)} - \mu)(\mu_{(x_j,y_j)} - \mu)^T \tag{4}$$

where $\mu$ represents the mean of all classes.

Now, we want to make $S_w$ as small as possible, and $S_b$ as large as possible. A possible way to do this is to maximize the ratio: $\frac{det|S_b|}{det|S_w|}$. The advantage of using this ratio is that it has been proven [10] that if $S_w$ is

a non-singular matrix then this ratio is maximized when the column vectors of the projection matrix are the eigenvectors of:

$$S_w^{-1} S_b \,. \tag{5}$$

Thus, we define the projecting matrix from the original space to the discriminant space as $\hat{\mathbf{D}} = [\hat{\mathbf{d}}_1, ..., \hat{\mathbf{d}}_m]^T$, where, the set $\{\hat{\mathbf{d}}_1, ..., \hat{\mathbf{d}}_m\}$ corresponds to the $m$ eigenvectors associated with the highest eigenvalues of equation (5). Notice that there are at the most $c-1$ non-zero generalized values, and so an upper bound on $m$ is $c-1$, i.e. the number of classes minus 1.

To be able to calculate these eigenvectors, we need to ensure that the within-class scatter matrix is non-singular. Unfortunately, the rank of $S_w$ is at the most $N-c$, and, in general $N \ll n$. For example, when we vectorize images of $256 \times 256$ pixels, they become vectors of $65,536$ dimensions.

A solution to this problem consist of either grab smaller images or make the robot navigate around the environment for many different days until the total number of images is obtained. Obviously, a totally impractical solution. A better approach to this consists of first projecting the original sample data onto an intermediate, $d$-dimensional, space before projecting the data onto the FDA space. By making this intermediate space small enough (i.e. $m < d << n$), we can guarantee that the within-class scatter matrix does not become singular. Weng [39] solved that by first projecting the data onto the PCA space, where data is optimally represented, and then *re*-projecting this eigen-data onto the FDA space (other works have also applied this approach successfully for several different applications, e.g. [5, 37]). The final projecting matrix is $\mathbf{D} = \hat{\mathbf{D}}\mathbf{E}$. That allows us to define a new learning and recognition algorithm, named the DA-Algorithm:

**DA-Algorithm: Learning**

(Items 1 to 3 as in the PCA-Algorithm)

4. **Eigenspace**. Also, project all $N$ normalized vectors to this obtained $d$-dimensional eigenspace and build the set $\mathbf{Y} = \mathbf{E}\,\mathbf{X}$.
5. **Compute the DA-space**: Obtain the $m$ first eigenvectors of equation (5), and store the projecting matrix $\mathbf{D}$ in memory. Project all $N$ sample vectors to this final discriminant space.

**DA-Algorithm: Recognition**

1. Obtain an image of the environment from the current unknown position, $\hat{\mathbf{x}}^{(x_{ukn}, y_{ukn})}$.
2. Obtain the normalized vector $\mathbf{y} = \mathbf{x}^{(x_{unk}, y_{unk})} - \bar{\mathbf{x}}$
3. Project this image onto the $m$-dimensional space by finding the product of the resulting vector $\mathbf{y}$ with the projecting matrix $\mathbf{D}$, i.e. $\mathbf{g} = [\mathbf{d}_1, ..., \mathbf{d}_m]^T \mathbf{y}$.
4. Find the closest point to $\mathbf{g}$ in the discriminant space. As above, we use the k-d tree algorithm.

## 3 Learning Visual Data Using Normal Mixture Models

In many cases, the data obtained from a real environment is embedded in a non-linear space. Figure 2a shows two simple non-linear subspaces representing two different classes. In such a case, although the original

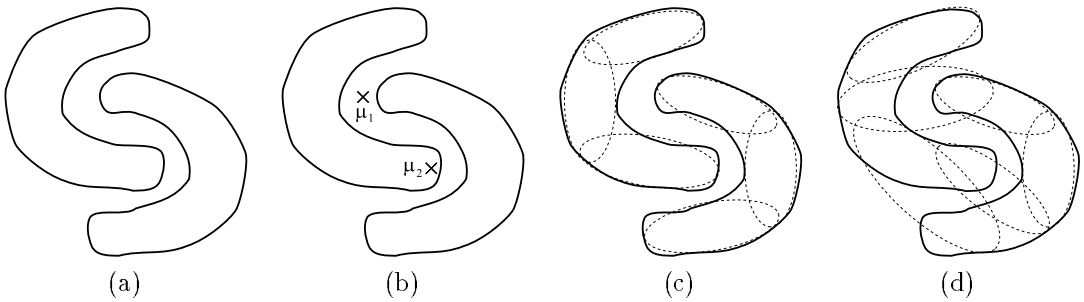



Figure 2: (a) The original space. Two different subclasses embedded in separated subspaces. (b) FDA does not work in this case. No linear discriminant projection will be found. (c) Using normal mixture models, we can readily describe the two classes in a discriminant manner. (d) A possible local maxima of the *a priori* probability of the EM algorithm.

space has clearly two discriminant subspaces, FDA will not compute a correct discriminant representation (Figure 2b). This problem might be overcome with the use of the non-parametric DA [11, 12], or by training a classifier to output one of the $c$ classes based on the input data (e.g. neural networks). However, it is also possible to discriminate by fitting a separate probability density model to each class and then picking the class that assigns the highest density to the test data. This is a more natural way of learning, because the discriminant subspaces within the representational space can be chosen to best fit the training data. Paying attention to our previous case of Figure 2a, we see that three subclasses per class suffice for discrimination (Figure 2c). Let us call each of these subclasses subspaces, hence each of them represents a subspace defined by the local model. In such a framework, the best classifier will be the one that best approximates (describes) the data [22]. Mixture models (e.g. mixture of Gaussians) are within this framework, and they have been successfully used for several applications within the appearance-based paradigm, as for example: face recognition [36], hand-writing recognition [13], lip reading [8] and image compression and data representation [6]. An important advantage of mixture models is that they combine much of the flexibility of non-parametric methods with certain of the analytical advantages of parametric methods [21].

We propose a third algorithm based on learning a mixture of normal models, named the NMM-Algorithm. When learning mixture models, the Expectation Maximization (EM) [9] algorithm is typically used. Unfortunately, in many cases, the solution reached for the classical EM algorithm does not suffice. As previously indicated, the learning method needs to fit the data as well as possible, but unfortunately this is not guaranteed. For instance, a solution that does not fit the data as well as 2c is the result shown in Figure 2d.

The EM algorithm is an iterative procedure that can only guarantee a local maximum of the *a posteriori* probability distribution, $p(\theta \mid \mathbf{X})$ ($\mathbf{X} \in \Re^{\mathbf{n}}$ represents the $n$-dimensional observed data set, and $\theta$ the unknown means and covariances of the normal models of the mixture) [22]. A possible improvement to this problem would be to calculate $v$ solutions by randomly initializing the parameters $\theta$ of the mixture model to several different values, and then selecting that with the highest (final) likelihood. That would be a brute force approach. We use another method based on using evolutionary strategies, which allows an efficient search within the parameter-space [19].

We first introduce our approach within a classical mixture of normal models framework, and later, we forward the genetic version of the EM algorithm and its use in our mixture model implementation.

### 3.1 Mixture models and the EM algorithm

Again let us consider a set of $N$ normalized sample images $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ taking value in an $n$-dimensional space. Under the finite mixture model, each $\mathbf{x}_i$ can be viewed as arising from a superpopulation $G_1, ..., G_t$ of probabilistic models with their correspondent associated probabilities $\pi_1, ..., \pi_t$ (such that, $\sum_{i=1}^{t} \pi_i = 1$). Then, a mixture model is defined by its probability density function (pdf) $f(\mathbf{x}; \phi)$, where $\phi = \{\pi, \theta\}$ represents the

unknown parameters that describe the pdf. We can decompose this pdf as a group of particular pdf's for each particular model as $f(\mathbf{x};\phi) = \sum_{i=1}^{t} \pi_i f_i(\mathbf{x};\theta)$.

Normal mixture model estimation is a common form of *soft* clustering. The data $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is assumed to be given by $\mathbf{x}_i \sim \mathbf{N}(\mu_j, \mathbf{\Sigma}_j)$ (where each normal model $\mathbf{N}_j$ belongs to a superpopulation $G_i$ with probability $\pi_i$). The means $\mu_1$, ..., $\mu_t$ and the covariance matrices $\mathbf{\Sigma}_1$, ..., $\mathbf{\Sigma}_t$, constitute the parameters $\theta$, associated with the density function.

The problem of estimating mixture densities is itself viewed as a missing data problem. The typical algorithm used to estimate the missing values is the Expectation Maximization (EM) algorithm [9]. The EM algorithm ([24, 22] are recommended for a review) is a widely-used probabilistic technique that provides a quite general approach to learning in the presence of unobservable or missing variables. The EM calculates the optimal estimators using the ML hypothesis, $\theta_{ML} = arg\, max_{\forall i}\ p(\mathbf{x} \mid \theta_\mathbf{i})$.

Mathematically speaking, let $\mathbf{z} \in \Re^c$ be the complete data (we assume that all $\mathbf{z}$ outcome from an underlying space $\zeta$ in $\Re^c$), and let $\mathbf{x} \in \Re^\mathbf{n}$ be the observable (or incomplete) data ($n < c$). The complete data $\mathbf{z}$ is not directly observable, but only by means of $\mathbf{x}$ as $\mathbf{x} = \mathbf{x}(\mathbf{z})$, This expression describes a mapping from many-to-one, that makes the problem difficult to be solved. However, we can assume that this mapping can be modeled by means of a probability density function (pdf) which depends on a parameter $\theta$, $f(\mathbf{z} \mid \theta)$ (where $\theta \in \Theta \subset \Re^r$ is the set of the parameters of the density). If we assume this function to be continuous through $\theta$ and appropriately differentiable, the ML estimate of $\theta$ can be assumed to lie within the region $\Theta$, and we can write the pdf of the incomplete data as:

$$g(\mathbf{x} \mid \theta) = \int_{\zeta(\mathbf{x})} f(\mathbf{z} \mid \theta) d\mathbf{z}$$

Let $L_x(\theta) = \log\, g(\mathbf{x} \mid \theta)$ denote the log-likelihood function. Now, we want to find the $\theta$ that maximizes the $f(\mathbf{z} \mid \theta)$. Unfortunately, we do not have the data $\mathbf{z}$ to compute the log-likelihood. To solve this, the EM algorithm estimates the $log\, f(\mathbf{z} \mid \theta)$ given $\mathbf{x}$ and the current estimation of $\theta$,

$$Q(\theta \mid \theta^{[k]}) = E[log\, f(\mathbf{z} \mid \theta) \mid \mathbf{x}, \theta^{[k]}]$$

and, then, calculates the argument $\theta$ that maximizes the previous estimation,

$$\theta^{[k+1]} = arg\, max_\theta\ Q(\theta \mid \theta^{[k]})\,.$$

These two steps are called the *Expectation* and *Maximization* step respectively. An initial value (normally at random) is given to $\theta$ and, then, both steps are iterated until convergence. It has been proven that any EM sequence of an exponential family always increases the likelihood and, if bounded above, converges to some local maximum [41]. Under this statement, convergence is reached when the ML hypothesis does not change or is smaller than a threshold, i.e. $\| p(\mathbf{x} \mid \theta^{[\mathbf{k}]}) - \mathbf{p}(\mathbf{x} \mid \theta^{[\mathbf{k-1}]}) \| < \varepsilon$.

In our case, we consider the probability of each data element $\mathbf{x}_i$ to belong to class $j$ as the missing data (which we denote as $h_{ij}$), and the sample set $\mathbf{X}$ as the observable data. In this case, the E-step corresponds to the expectation that an observable value $\mathbf{x}_i$ belongs to a normal model $\theta_j$, e.g. $E[h_{ij} \mid \mathbf{x}_i, \theta_j]$. We temporally assume that the current estimation of each model, $\theta_j$, is correctly known. This can be written as,

$$h_{ij}^{[k+1]} = \frac{\left|\boldsymbol{\Sigma}_j^{[k]}\right|^{-1/2} exp\{-\frac{1}{2}(\mathbf{x}_i - \mu_j^{[k]})^T \boldsymbol{\Sigma}_j^{[k]\,-1}(\mathbf{x}_i - \mu_j^{[k]})\}}{\sum_{l=1}^{t}\left|\boldsymbol{\Sigma}_l^{[k]}\right|^{-1/2} exp\{-\frac{1}{2}(\mathbf{x}_i - \mu_l^{[k]})^T \boldsymbol{\Sigma}_l^{[k]\,-1}(\mathbf{x}_i - \mu_l^{[k]})\}} \tag{6}$$

where $h_{ij}$ represents the probability of sample $x_i$ to belong to the normal model $\mathbf{N}_j(\mu_j, \boldsymbol{\Sigma}_j)$. The M-step will, then, re-estimate the means and the covariances of the normal models assuming that the $h_{ij}$ above estimated are correct,

$$\mu_j^{[k+1]} = \frac{\sum_{i=1}^{N} h_{ij}^{[k]} \mathbf{x}_i}{\sum_{i=1}^{N} h_{ij}^{[k]}} \tag{7}$$

$$\boldsymbol{\Sigma}_j^{[k+1]} = \frac{\sum_{i=1}^{N} h_{ij}^{[k]} (\mathbf{x}_i - \mu_j^{[k+1]})(\mathbf{x}_i - \mu_j^{[k+1]})^T}{\sum_{i=1}^{N} h_{ij}^{[k]}} \tag{8}$$

where $N$ is the number of samples, and $t$ the number of models used in the mixture. Moreover, we are assuming that all models are equi-probable (i.e. $\pi_i = \pi_j \; \forall i, j$). In any other case the above equations will differ slightly (see [21] for details). The initial values (i.e. $\mu_j^{[0]}$ and $\boldsymbol{\Sigma}_j^{[0]} \; \forall j$) are taken at random.

### 3.2 The NMM-Algorithm

Assume we are given a vast convoluted manifold in the space of pixel intensities (i.e. our original feature space), where every vector that represents a possible view of our robot working-space lie. This (theoretical) manifold accounts for all possible local variations of the environment, e.g. view-point, lighting, partial occlusions, etc. It is plausible that locally this manifold is low-dimensional relative to the number of pixels, because our environment (the robot's working-space) is a small subset of all possible images that the space of pixel intensities can represent. Whereas PCA and FDA approximate this non-linear low-dimensional manifold with a linear subspace, mixture models can represent the non-linear changes more accurately (Figure 2).

However, due to the high-dimensionality of this original space of pixel intensities, mixture models (e.g. mixture of Gaussians) will find it difficult to approximate the non-linearities of the environment manifold. We resolve this (in a similar manner as for the DA-Algorithm), by first projecting the original space of pixel intensities to an intermediate space (in $\Re^d$); the PCA-space. The intermediate space is chosen to be the PCA because it is the way to represent the data onto a lower dimensional space with the smallest structural data change; i.e., it finds a projection where the structure of the data suffers the smallest geometric change relative to the data structure encountered in the original feature space.

The normal mixture models algorithm, named the NMM-Algorithm, is described as:

**NMM-Algorithm: Learning**

(Items 1 to 4 as in the PCA-Algorithm)

5. **EM algorithm**: Evaluate the EM algorithm $v$ times and select the best outcome.
   (a) Initialize, randomly, all $t$ normal models of the mixture.
   (b) While (not convergence)

i. Evaluate the EM algorithm, use formulas (6) to (8).

(c) end while.

6. **Memory**: Store in memory the resulting subspaces (i.e. the means $\mu_i$ and covariance matrices $\mathbf{\Sigma}_i$, of all normal models of the mixture) and the samples (samples are projected onto a lower-dimensional space using each of the local normal models).

**NMM-Algorithm: Recognition**

1. Obtain an image of the environment from the current unknown position, $\hat{\mathbf{x}}^{(x_{ukn}, y_{ukn})}$.
2. Compute the normalized vector $\mathbf{y} = \mathbf{x}^{(x_{unk}, y_{unk})} - \bar{\mathbf{x}}$
3. Project the vector onto the PCA-space, $\mathbf{g} = \mathbf{E}\,\mathbf{y}$
4. Find the closest normal model to $\mathbf{g}$. As in all other cases, we use the k-d tree algorithm. However, whereas in the previous cases the Euclidean distance was used, in this case, the Mahalanobis distance must be used instead. This is due to the fact that Gaussian distributions were obtained in the learning stage, which impose a deformation of the space that has to be considered here. The Mahalanobis distance is considered a better option, because it represents the deformation of the space in relation to the training data, whereas the Euclidean distance does not take the data structure into account. Finally, we find the closest sample within the closes Gaussian distribution.

### 3.3 Learning mixture models with the EM and GA

In order to improve the results one would obtain with the above described system, we define a new method that uses a population of mixture models (rather than a single mixture). For details and experimental results using synthetic data see [19]. Formally, let us define a normal mixture model as $w_i = \{\mathbf{N}(\mu_{i1}, \mathbf{\Sigma}_{i1}),\ ...,\ \mathbf{N}(\mu_{it}, \mathbf{\Sigma}_{it})\}$, where $\mu_{ij}$ and $\mathbf{\Sigma}_{ij}$ are the mean and covariance matrix of the $j$th normal model of the $i$th mixture, $\mathbf{N}(\cdot)$ represents the normal distribution function, and $w_i$ the $i$th mixture model involving all $t$ normal models. In the sequel, when referring to any of these models, we shall use the following notation: $\mathbf{O}_{ij}$ (i.e. $\mathbf{O}_{ij} = \mathbf{N}(\mu_{ij}, \mathbf{\Sigma}_{ij})$).

In a GA, we are given an initial population $W = \{w_1,\ ...,\ w_p\}$, of $p$ different normal mixture models. Each individual of the population is initialized at random within a normal distribution with zero mean and identity matrix as covariance matrix, i.e. $\mathbf{N}(\mathbf{0}, \mathbf{I})$.

Then a $(p, o)$-strategy is used to select those individuals which best represent our data (that is to say, those ones which are best adapted). The notation $(p, o)$ indicates that $p$ parents create $o$ offspring by means of recombinations and mutations, and only $p$ of these $o$ offspring individuals are deterministically selected to replace the previous parents. Notice that the method is not *elitistic*, in the sense that the best member of the population at generation $g + 1$ can perform worse than the best individual at generation $g$. Thus, we facilitate the system to accept temporary deterioration that might help to leave the region of attraction of a local maximum and reach a better final state. This strategy has been defined as being more efficient that the (p+o)-strategy, where $p$ offspring are selected from the union of parents and offspring [3].

The mutation process is done at a rate of 40% and, even then, only 25% of the parameters of each model are mutated. It has been shown that this is a good rate that generally converges into a good solution [19]. The mutation operator uses a normally distributed random vector $\mathbf{N}(\mathbf{0}, \mathbf{1})$. Moreover, we impose the standard deviation ($\sigma$) to be identical for all object variables. To achieve that, all object variables are mutated as follows:

$$\begin{aligned} \sigma' &= \sigma \cdot exp\{(\sqrt{n})^{-1} \cdot \mathbf{N}(\mathbf{0}, \mathbf{1})\} \\ o_{k,ij} &= o_{k,ij} + \sigma' \cdot \mathbf{N}(\mathbf{0}, \mathbf{1}) \end{aligned}$$

where $o_{k,ij}$ represents the $k$-component of the $\mathbf{O_{ij}}$ model, and $\sigma$ is a constant.

The rest of the offspring individuals (i.e., 60% of them) are obtained by recombination. To do that, we use the classical recombination operator, crossover. We first select two individuals at random, let us say individual $A = (a_1, ..., a_{m(1+m)})$ and individual $B = (b_1, ..., b_{m(1+m)})$. After that, we select a random integer number *pos* from the range $[1, ..., m(1+m)]$ (where $m(1+m)$ is the length of the unknown variables – precisely, $m$ is the dimensionality of the mean vector and $m^2$ the dimensionality of the covariance matrix). Then the two individuals are replaced by both their offsprings $(a_1, ..., a_{pos}, b_{pos+1}, ..., b_t)$ and $(b_1, ..., b_{pos}, a_{pos+1}, ..., a_{m(1+m)})$.

The evaluation process consists of applying the EM algorithm for normal mixture models described above (in formulas from (6) to (8)).

After obtaining the $o$ offspring, a selection operator based on the Maximum-Likelihood (ML) hypothesis is used to select those $p$ individuals that best fit the data. To do this, we calculate the *a posterior* probability of each model of the mixture,

$$p(w_i \mid \mathbf{x}) = \sum_{j=1}^{t} p(\theta_{ij} \mid \mathbf{x}) \tag{9}$$

and then, select those $p$ individuals associated with the highest *a posteriori* probabilities obtained.

The reader should keep in mind that many other alternative to the selection mechanism described above exist. Among others, the reader might be interested in some of the following alternative: stochastic sampling (also known as the roulette wheel selection method), stochastic universal sampling, rank-based fitness assignment (linear or non-linear rankings) and several local selection mechanisms. For details on some of these alternative and on comparison results see [7, 27].

Also a lucky-factor has been added to the system to help maintaining diversity in the population [40]. In 20% of the generations (iterations of the GA), $q$ randomly selected parents also survive, where $q \in [1, p/4]$ (obviously, when this is the case, only $p - q$ individuals can survive from the selection process).

The new genetic version of the algorithm, named the GA-NMM-Algorithm, follows exactly as the one reported before. Only the EM step changes. This is described as follows:

**GA-NMM-Algorithm: Learning**

5. **EM algorithm**: Evaluate the EM algorithm $g$ times and select the individual with best *a posteriori* probability of the output population:
   (a) Initialize (randomly) the population of mixture of normal models, $W$.
   (b) Evaluate all mixture models.
   (c) While (not convergence)
      i. Mutate the current mixture models, $W' = Mutate\,(W)$.
      ii. Recombine the current mixture models, $W' = W' \cup Recombine\,(W)$.
      iii. Evaluate the new mixture models, $W'$.
      iv. Select those mixtures with highest *a posteriori* probability as inputs for the next generation, $W = Select\,(W')$. Use the lucky factor, if applicable.
   (d) end while.

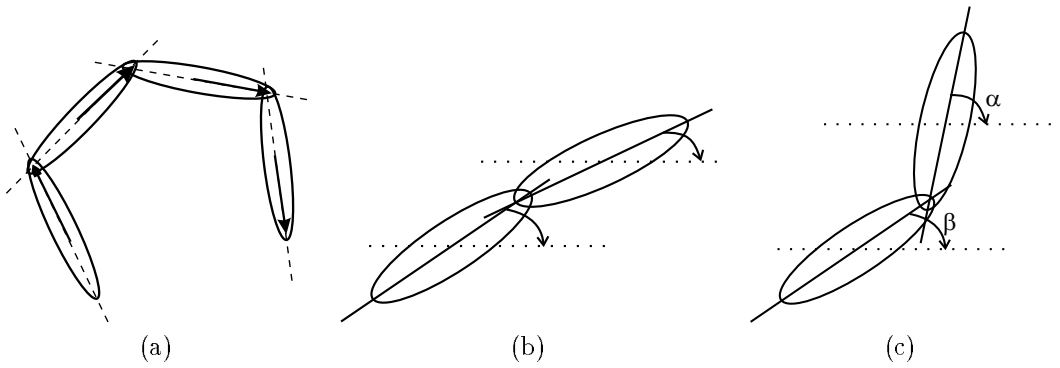



Figure 3: (a) A mixture of Gaussians and their corresponding eigenvectors associated to the largest eigenvalue of each model (hard lines represent the first eigenvector of each model, discontinuous lines represent the axes defined for each of these eigenvectors). The case of figure (b) is much more desirable than the case of figure (c), since the approximation made is more smooth.

### 3.4 Using knowledge to learn

The search for a good fit for the mixture of normal models is a complex problem mainly because of the large dimensionality of the searching-space. GAs attempt to resolve this by combining and mutating local solutions obtained earlier. It is clear though, that if some heuristics could be included into the searching mechanism, better results could be obtained (or, at least, the final –desirable– results could be reached faster).

As indicated in the previous section, we know that if consecutive images are grabbed after small displacement of the robot, their vector representations will be strongly correlated, and thus their projections onto the low-dimensional space can be assumed to be close to one another. This fact allows us to assume that the manifold where data is described is continuous.[3] Hence this manifold is continuous and can be assumed to have local linear parts (depicted by our models), it is logical to expect these models (of the mixture) to be aligned (i.e. interconnected to one another). This is visually described in Figure 3a. The greatest advantage of this constraint is that it is very easy to implement in practice by forcing consecutive models to have their first eigenvector (the one associated to the highest eigenvalue) as similar to each other as possible; as schematically depicted in Figure 3a.

If we think of those first eigenvectors of each model as local linear approximations of the subspaces where samples take value, we can readily see that when good approximations are searched, a smooth factor of the angle described between consecutive eigenvectors must be imposed [8]. The mixture of models space is an approximation of the original manifold, and, should therefore be as similar to it as possible. Obviously, the more models are added to the mixture, the better the (linear) approximations should be. However, in general, and under the same conditions (that is to say, same number of models), those that describe the data most precisely are those that best approximate the non-linearity of the original manifold and, in consequence, those that have the first eigenvector most similar to its predecessor and successor models. For example, Figure 3b depicts a more desirable case than Figure 3c.

Before computing the difference of angle value between two consecutive models, we must determine which are the predecessor and successor models for each normal distribution $\mathbf{O}_{ij}$. The optimal way of doing that consists of calculating the geodesic distance from the mean of the current model to all other models. The geodesic distance is defined as the shortest path that unifies two points. Depending on the metric used, the geodesic distance can be easy or complex to compute. Unfortunately, the case of mixture of Gaussians is know to be on the complex side. A possible approach consists of using the Mahalanobis distance between each pair of models;

[3] The assumption of continuity is not a new concept. It has been used in the appearance-based domain for a long time. See, for example, [29].

Figure 4: Some images of a run along a hallway of the first floor of the building where the experiments were performed (only few images are shown here).

i.e. we can use both, the Mahalanobis distance that goes from model $\mathbf{O}_{ij}$ to model $\mathbf{O}_{ik}$ and the Mahalanobis distance that goes from $\mathbf{O}_{ik}$ to $\mathbf{O}_{ij}$. While the first measurement takes into account the space deformation created by model $\mathbf{O}_{ij}$, the second one will consider the deformation caused by model $\mathbf{O}_{ik}$. Finally, the average between both distances is taken as the measure of proximity between both models. Mathematically speaking, we define this measure of proximity as $D_{ij,ik} = (d_{ij,ik} + d_{ik,ij})/2$, where $d_{ij,ik}$ is the Mahalanobis distance from $\mathbf{O}_{ij}$ to $\mathbf{O}_{ik}$, and $d_{ik,ij}$ the Mahalanobis distance from $\mathbf{O}_{ik}$ to $\mathbf{O}_{ij}$. The Mahalanobis distance is expressed as $d_{ij,ik} = \sqrt{(\mu_{ij} - \mu_{ik})^T \Sigma_{ik}^{-1} (\mu_{ij} - \mu_{ik})}$ (where $\Sigma_{ik}$ is the covariance matrix of model $\mathbf{O}_{ik}$).

Once the predecessor and successor of each model $\mathbf{O}_{ij}$ have been found, we can calculate the difference between their first eigenvectors (i.e. the angle). This can be formally expressed as $q_{ij} = 1/(|\alpha - \beta| + |\alpha - \gamma| + 1)$, where $\alpha$ is the clockwise angle between the current model $\mathbf{O}_{ij}$ and the horizontal, $\beta$ is the clockwise angle between the predecessor and the horizontal, and $\gamma$ is the clockwise angle between the successor and the horizontal. The value of $q_{ij} \in [1, \frac{1}{721})$. Now, $\sum_{j=1}^{t} q_i$ defines a new quality factor for the individual $w_i$ of the population $W$.

The final selection function should be proportional to both the quality of the mixture (i.e. $q_i$) and the *a posterior* probability (i.e. $p(w_i \mid \mathbf{x})$). We can write this selecting function as:

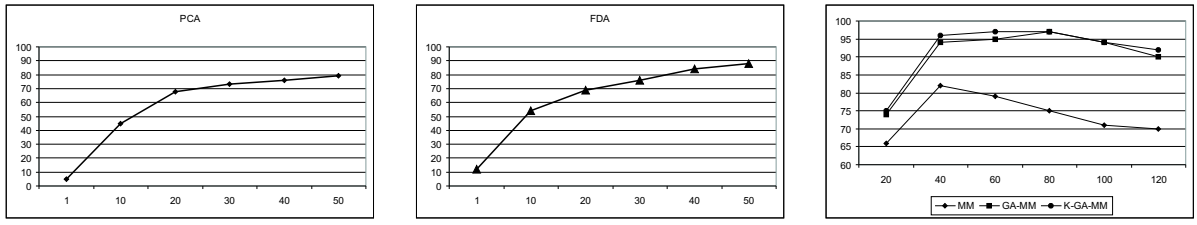
$$p(w_i \mid q_{ij}, \mathbf{x}) = \tau \, (\sum_{j=1}^{t} q_{ij}) + (\tau - 1) \, p(w_i \mid \mathbf{x}) \tag{10}$$

where $\tau$ is the value that determines which of the two factors have more relevance (the smoothing factor or the *a priori* probability).

This last algorithm, named the K-GA-NMM-Algorithm, slightly differs from the GA-NMM-Algorithm, in that the selection function uses (10) instead of (9).

### 3.5 Experimental results

In this sub-section, we first describe the data obtained for the experiments discussed here, and second, report results on all four algorithms defined above.




(a) (b) (c)

Figure 5: Results for all algorithms described in this contribution. Results are from (a) PCA, (b) FDA, and (c) mixture models (i.e. NMM, GA-NMM and K-GA-NMM algorithms).

### 3.5.1 Experimental data

The environment used in our test is schematically depicted in Figure 1. This figure illustrates the first floor of a research institution, where the test images were obtained. The floor is divided into four different hallways.

Two different (independent) sub-sets of images were obtained from this environment (working-space). The first set was used for learning purposes, while the second was intended for testing. The learning images were captured on two consecutive days. On each day, we manually drove our robot for a total of six different runs around the whole environment (which includes all four hallways). All of them were obtained in the evening. Some of the images obtained are shown in Figure 4. $\epsilon$ was equal to 20 $cm$. A complete run contains $\sim$ 350 images. Notice that these twelve different groups of samples represent the same environment viewed from *slightly* different points, with different illumination conditions, etc. This set was built so that it represented the working-space from many independent points of view. This is useful, because large datasets are needed to train the approaches discussed in this paper.

The testing images were acquired during the two consecutive days after those obtained for learning. Again, we manually drove our robot along all four hallways ($\epsilon = 20cm$). For learning, six runs around the enier working-space were made (three on each day). Again, all images were obtained in the evening, which prevents the lighting from changing drastically (since the building had large windows on all hallways).

### 3.5.2 PCA, DA, NMM, GA-NMM and K-GA-NMM tests

The same learning data was used to train all five described algorithms. However, small differences exist for each method. For each of them, we tested different settings of its parameters and depicted the best results for comparison.

To analyze which dimensionality is most adequate for the intermediate (i.e. PCA) space, different values of the dimensionality were tested. The results are shown in Figure 5a. The best results were obtained when $m = 50$, although no big differences exist for the values ranging from 30 to 50.

For the DA algorithm, recall that we are using a supervised method and, therefore, our data has to be labeled. We chose 120 different classes; i.e. $c = 120$. Then, the DA algorithm has two dimensionality reduction stages: a first step for the PCA (i.e. $d$) and a second one for the FDA (i.e. $m$). We tested values of $m$ from a low of 1 to a high of 50. For all these dimensions $d$ was kept fixed at a value of $d = 100$ (this is supposed to be adequated from the above results). Results are shown in Figure 5b.

Although the NMM algorithm does not require the data to be labeled, a number of classes has to be assumed (i.e. number of super-populations, $t$). For this purpose we tested different settings, from a low of 20 to a high of

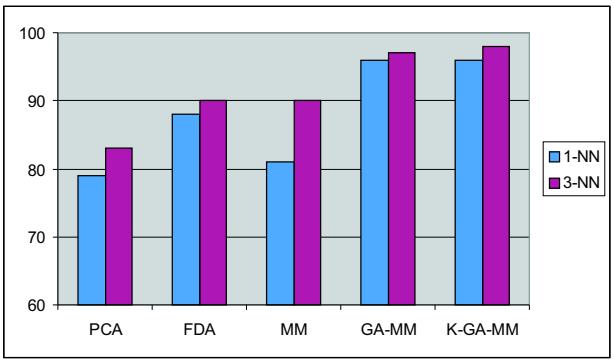



Figure 6: Successful rate results for all five different algorithms. Notation: 1-NN means that the solution is within the first nearest neighbor, 3-NN means that the solution is within the three first nearest neighbors.

120 (recall that in this algorithm $t$ –the number of models– equals $c$ –the number of classes). The intermediate space was $d = 100$. Algorithms GA-NMM and K-GA-NMM also use these values. Every clustering was then reduced to a 50 dimensional eigenspace, where Euclidean searches can be applied to search the closest match. Thus, the search implies two steps: (i) finding the closest normal model and (ii) searching for the closest sample within this local (linear) deformation. Results are plotted in Figure 5c (in this figure the horizontal axes represents the value of $t$).

Figure 6 depicts the best results obtained when using PCA, DA, NMM, GA-NMM and K-GA-NMM. Results are shown using two different searching methods, the nearest neighbor and the three nearest neighbors. While in the former we only search for the closest sample (using $L_2$-norm), in the latter we search for the three closest points. In this latter case, if the correct sample to be selected is within these three closest samples, a successful search is recorded.

## 4 Annotation: using high-level/symbolic communications

So far, we have shown how using mixture models and genetic algorithms, the localization problem using appearance-based methods becomes more robust. In this section, we describe the earlier introduced platform: "annotations".

When building this platform, we must first learn the environment by means of either the GA-NMM or K-GA-NMM algorithm (because, these are the ones that best describe the environment as shown above). To do this, we have developed a tool that allows the user to drive the robot around the environment in a simple manner. This procedure must be completed several times (as stated earlier), because this facilitates the learning of the surroundings under different environmental conditions (such as different illumination conditions, occlusions, viewpoint positions, etc.). Recall that in such a learning mechanism, the representative space is divided into many different *sub*-spaces representing different areas of the working-space. If $t$ normal models are used for learning, $t$ different areas of the real environment will be discriminated. The mean of each normal model of the mixture will represent a 2D-world position (or, equivalently a 3D-world-position if $z$ is not assumed to be zero) of the surroundings (or working-space). These positions can be plotted on a blackboard or relative map, to facilitate the interaction with the user. The user's task is as easy as naming each of these labels (which will finally allow the *symbolic* human-robot communication).

### 4.1 Learning the surroundings

As Figure 7a shows, the user (teacher) has to move the robot around the environment it is supposed to learn. The user is responsible for correctly initializing the system at position zero, i.e. $(x, y) = (0, 0)$. Then, at

each $\epsilon$ centimeters of displacement the robot stops and grabs an image from its current location. The value of $\epsilon$ does not need to be constant but it can be changed by the user at specific locations of the environment that require more attention or that are known to be more difficult to learn. For example, turning from one hallway to another is one of these cases. To cope with this problem, we define $\epsilon$ as a variable that changes with time, i.e. $\epsilon(t)$, as illustrated in Figure 7b.

Once the learning process has been concluded, the robot can ask the user to label those areas that the learning system finds relevant (discriminant). To solve this, the robot makes a new run along all hallways and stops at those places in the environment that represent the mean of each of the normal models of the mixture, Figure 7c. The user can then easily give a name to each of these models. If some type of map of the environment is supplied, the labeling step becomes even simpler.

### 4.2 Qualitative localization

We built a system commander that allows the user to employ the labels described earlier so as to indicate where the robot has to go next. These commands could also come from another module of the robot that asks the navigation system to move the system in the desired direction so as to achieve a goal. Figure 1b shows an example of that.

Finally, the robot uses high-level descriptions to inform either the user or other tasks of the system, where the robot is within its environment, what it has been doing or what it is expecting to do next. These high-level descriptions are posted on a qualitative map of the environment as annotations. Notice that the map in not quantitative, in the sense that does not precisely represent the 2D or 3D environment of the robot, but it only represents the *symbolic* idea of the surroundings. Figure 7d shows an example of these annotations. This *symbolic* or qualitative map was exclusively designed for the task reported here. When learning new environments, new maps must be supplied.

While moving around its environment the robot works as follows. It first grabs an image, vectorizes it and projects the high-dimensional vector obtained onto the PCA-space. Once in the eigenspace, the system searches for the closest normal model (which infers the label). Recall that in our experiments, we use 100 dimensional eigenspaces. Once a closest normal model has been selected, the system searches for the nearest sample. When doing that, we only use the largest 50 dimensions of the sub-space described by the selected local normal model, because this ensures good results (as justified in Figure 8) with a low computational cost. This latest step can be seen as a second dimensionality reduction stage, where samples belonging to different models are projected onto different local linear sub-spaces.

Once the nearest sample has been found, the robot is moved in the same manner as that used by the teacher (supervisor) when in the same position (notice that each sample does not only represent a 2D-environmental-position, but it also represents a specific orientation, illumination conditions, etc.). A simple improvement of this system, consists of searching for the closest $k$ samples, and then interpolating their motor commands (i.e. k-NN). The experimental results described in this contribution use $k = 2$.

### 4.3 Experimental results

The final experiments were conducted in the environment described above. To do this test, the robot was manually driven by a user twelve different times during two consecutive days. Then, the system learned the environment using GA-NMM and K-GA-NMM. Excluding the PCA step (common for both methods), K-GA-NMM converged faster than GA-NMM.

During the next three days, the robot was asked to perform different tasks (10 tasks per day). The robot used the navigation algorithm to move towards the desired position and used the localization algorithm to recognize where to stop. The successful rates are shown in detail in the table 1.

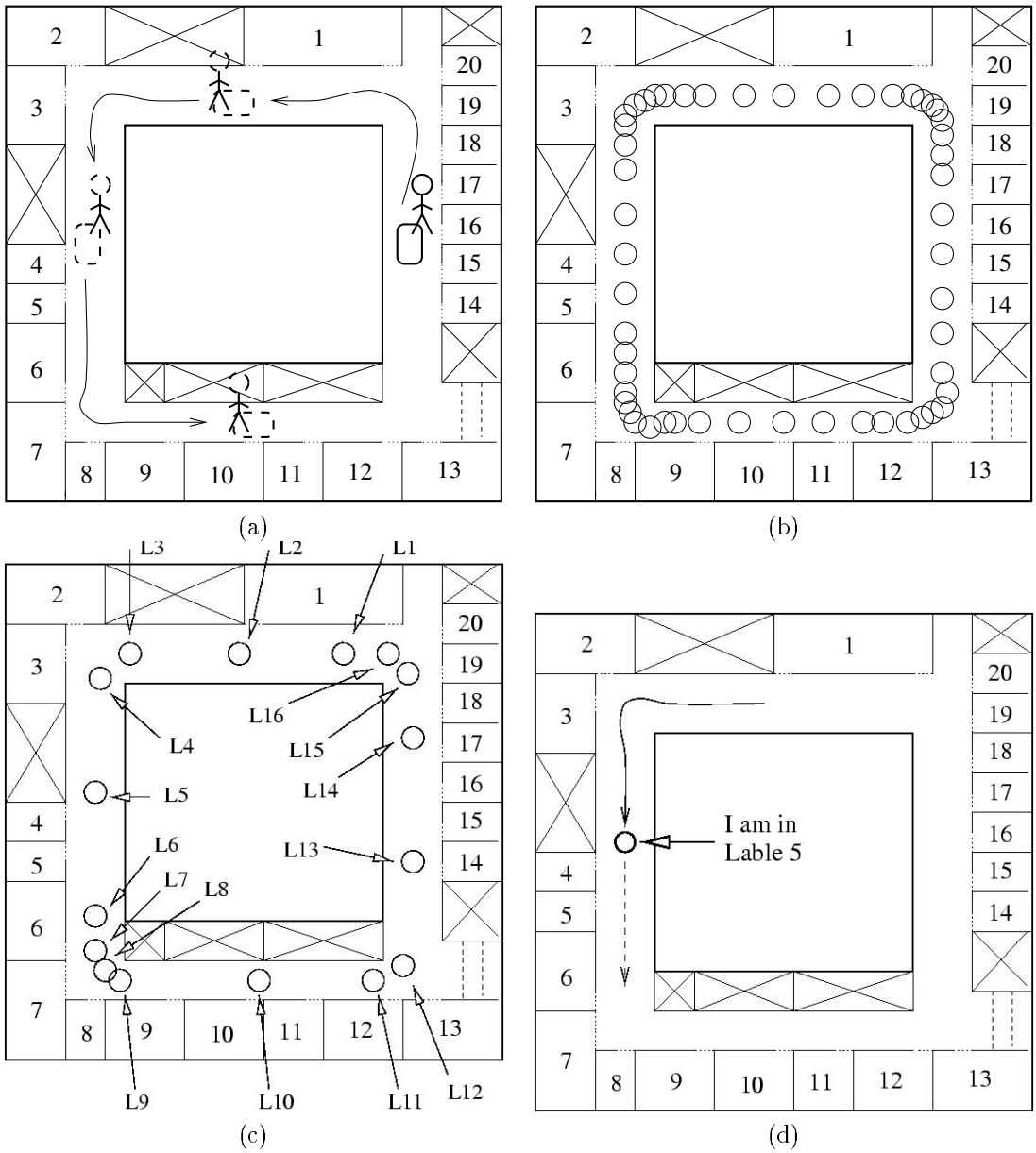



Figure 7: (a) The user (supervisor) has to manually drive the robot along all the hallways the robot is supposed to learn. (b) It is adequate to use smaller values of $\epsilon$ for those cases where non-continuous spaces are expected to be obtained. (c) The selected labels obtained by the system. The user is asked to specify a name for each of the labels. Each label corresponds to the *mean* of each normal model. (d) The robot has reached label 5. It makes an annotation on the *map* to specify this fact. If any other task (or the user) is interested in what the robot is trying to do next, the answer would be: "going to label 6". Furthermore, the robot can answer: "I am going from label 2 to label 6. Currently, I am in label 5".

| **Experimental Results** | | | | |
|---|---|---|---|---|
| | Day 1 | Day 2 | Day 3 | Average |
| GA-NMM | 90% | 90% | 100% | 93.33% |
| K-GA-NMM | 100% | 90% | 100% | 96.67% |

Table 1: Successful recognition rates for our final experimental results.

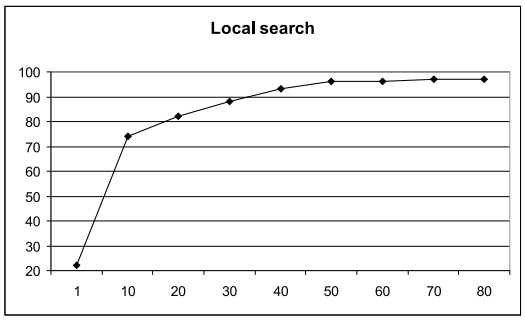



Figure 8: 1-NN search results as a function of dimensionality (horizontal axes).

Clearly both algorithms are almost equivalent, but K-GA-NMM has faster learning times. For those cases where the robot could not finish the required task correctly, the robot stopped because it reached a point that could not be recognized with a high enough probability factor. For obvious reasons the learning of more data would increase the probability of success for further recognitions, but it would also make the user's task less *simple*. A drawback of the appearance-based methods is that it is still not clear how to use dimensionality reduction techniques to achieve generalization. To date, systems require large and representative datasets [14, 18]. Further efforts along these lines are still needed. Note however that if the PCA (the dimensionality reduction technique used here) could be replaced with a *better* mapping technique (which better achieves generalization in our context), the rest of the system would still be valid, and better results could probably be obtained.

## 5 Conclusions

It is clear that when building autonomous robots, the user-robot interaction becomes a crucial issue. However, the classical way to build vision-guided autonomous robots requires the user to supply a 3D geometrical description of the surroundings (model-based approach). These models are usually difficult to be built because precise descriptions of the robot working-space are needed for the algorithms used to recognize the environment.

In this paper, we have proposed the use of appearance-based approaches instead. The great advantage of appearance-based methods is that it is not necessary to define a representation or model for a particular class (e.g. a 2D-world position) since the class is implicitly defined by the selection of the test images.

However, we have shown that already existing approaches within the appearance-based paradigm, such as PCA and FDA, do not suffice. For this reason a new method that uses mixture models (density models) to learn the training data-set has been introduced. Unfortunately, since mixture models are typically learned using the EM (Expectation-Maximization) algorithm, a good final convergence is not guaranteed. This is due to the fact that the EM algorithm is a hill-climbing method that only reaches a local maximum. To resolve this, two new approaches that use genetic algorithms have been presented. It has been empirically proven that both methods better learn and identify visual data acquired from a mobile robot than other appearance-based methods such as PCA and FDA.

The first approach uses a population of $i$ different mixture models that evolves over time (in $g$ generations). This has been proven to obtain better results than the brute force approach (i.e. randomly initialize the EM algorithm $v$ different times and choose the best final result, where $v = g \cdot i$).

The second approach assumes that the data obtained from the environment is embedded into a continuous manifold (even though this representation can still be non-linear). This idea has been successfully used before by other researchers (e.g. [29]). In this second case the results did not improve, but the convergence velocity of the EM algorithm was faster.

These two new approaches were then used to build a platform that uses "annotations" to facilitate the user-robot interaction. Annotations are defined as "specific locations (areas) of the environment of the robot which are labeled with a specific (*symbolic*) name. These names are used to explain the actions that the robot performs and can be used to accept high-level user commands (using these symbols)". The name annotations comes from the fact that these labels are posted onto a blackboard or qualitative map. This has been proven to facilitate the user-robot interaction. Now, the user's task is as simple as asking the robot to "go to label 5". Also the robot can easily give qualitative (or symbolic) information of what it is doing (the user no longer has to analyze 3D-maps of the surroundings). The results obtained by our methods were above 90% correct.

An important feature of the systems under discussion here is that they require of large and representative data-sets. It is not always the case that one can obtain a large enough training set. Even where this is feasible some important environmental features might not be present in the sample set, which will ultimately make the system fail. Further research will focus on how to cope with such a problem.

## Acknowledgments

The authors wish to thank Soledad Garcia, Robert Benavente and Jordi Lluis for their help in different parts of the project. This project was partially financially supported by grant TAP94-0401 from CICYT. We thank the referees for their comments.

## References


[1] Y. Adini, Y. Moses and S. Ullman, "Face Recognition: The Problem of Compensating for Changes in Illumination Direction," IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7):721-732, 1997.

[2] A.V. Ayache and O.D. Faugeras, "Maintaining representations of the environment of a mobile robot," IEEE Transaction on Robotics and Automation 5(6):804-819, 1989.

[3] T. Bäck and H. Schwefel, "Evolutionary Computation: An Overview," Proc. IEEE Conf. Evolutionary Computation (ICEC'96), Nagoya, Japan, 1996.

[4] D.H. Ballard, "Animate Vision," Artificial Intelligence 48:57-86, 1991.

[5] P.N. Belhumeour, J.P. Hespanha and D.J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," Transactions on Pattern Analysis and Machine Intelligence 19(7):711-720, 97.

[6] C.M. Bishop and M.E. Tipping, "A Hierarchical Latent Variable Model for Data Visualization," IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3):281-293, 1998.

[7] T. Blickle and T. Lothar, "A Comparison of Selection Schemes Used in Genetic Algorithms," Intl. Journal on Evolutionary Computation, Vol. 4, No.4, 1996.

[8] C. Bregler and S.M. Omohundro, "Surface Learning with Applications to Lipreading," Advances in Neural Information Processing Systems 6 (J.D.Cowan, G.Tesauro and J.Alspector, Eds.), Morgan Kaufmann Publishers, pp. 43-50, 1994.

[9] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," Journal Royal Statistical Society, 30(1):1-38, 1977.

[10] R.A. Fisher, "The Statistical Utilization of Multiple Measurements," Annals of Eugenics, 8:376-386, 1938.

[11] K. Fukunaga and M. Mantock, "Nonparametric discriminant analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence 5:671-678, 1983.

[12] K. Fukunaga, "Introduction to Statistical Pattern Recognition (second edition)," Academic Press, 1990.

[13] G.E. Hinton, P. Dayan and M. Revow, "Modeling the Manifolds of Images of Handwritten Digits," IEEE Transactions on Neural Networks, 8(1):65-74, 1997.

[14] A.K. Jain, R.P.W. Duin and J. Mao, "Statistical Pattern Recognition: A Review," IEEE Trans. Pattern Analysis and Machine Intelligence 22(1):4-37, 2000.

[15] A. Kosaka and A.C. Kak, "Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties," CVGIP: Image Understanding 56(3):271-329, 1992.

[16] M. Kirby and L. Sirovich, "Application of the Karhunen-Love Procedure for the Characterization of Human Faces," IEEE Transactions on Pattern Analysis and Machine Intelligence 12(1):103-108, 1990.

[17] J. Latombe, "Robot Motion Planning," Kluwer Academic Publishers, 1991.

[18] A.M. Martínez and A.C Kak, "PCA versus LDA," IEEE Transactions on Pattern Analysis and Machine Intelligence 23(2):228-233, 2001.

[19] A.M. Martínez and J. Vitrià, "Learning Mixture Models Using a Genetic Version of the EM Algorithm," Pattern Recognition Letters 21, pp.759-769, 2000.

[20] A.M. Martínez, "Recognition of Partially Occluded and/or Imprecisely Localized Faces Using a Probabilistic Approach," In Proc. of Computer Vision and Pattern Recognition, Vol. I, pp. 712-717, Hilton-Head Island, June 2000.

[21] G. McLachlan and K. Basford, "Mixture Models: Inference and applications to clustering," Marcel Dekker, 1988.

[22] G. McLachlan and T.Krishnan, "The EM algorithm and Extensions," Wiley, 1997.

[23] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Representation," IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7):696-710, 1997.

[24] T.K. Moon, "The Expectation-Maximization Algorithm," IEEE Signal Processing Magazine, 13(6):47-60, 1996.

[25] H.P. Moravec, "Robot Rover Visual Navigation," UMI Research Press, 1981.

[26] H.P. Moravec, "The Stanford Cart and the CMU Rover," Proceedings of IEEE 71(7):872-884, 1983.

[27] H. Mühlenbein and D. Schlierkamp-Voosen, "Analysis of Selection, Mutation and Recombination in Genetic Algorithms," Neural Network World 3, pp. 907-933, 1993.

[28] H. Murakami and V.Kumar, "Efficient Calculation of Primary Images from a Set of Images," IEEE Transactions on Pattern Analysis and Machine Intelligence 4(5):511-515, 1982.

[29] H. Murase and S. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," International Journal of Computer Vision, 14: 5-24, 1995.

[30] H. Murase, F. Kimura, M. Yoshimura and Y. Miyake, "An improvement of the auto-correlation matrix in pattern matching method and its application to handprinted 'HIRAGANA'," Transactions IECE J64-D(3), 1981.

[31] S.K. Nayar, N.A. Nene and H. Murase, "Subspace Methods for Robot Vision," IEEE Transactions on Robotics and Automation 12(5):750-758, 1996.

[32] S.K. Nayar and T. Poggio (Eds.), "Early Visual Learning," Oxford University Press, 1996.

[33] R. Redner and H. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm," SIAM Rev., 26(2):195-239, 1984.

[34] A. Rosenfeld and A.C. Kak, "Digital Picture Processing (second edition)," Academic Press, 1982.

[35] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," J. Opt. Soc. Am. A, 4:519-524, 1987.

[36] K.K. Suang and T. Poggio, "Example-Based Learning for View-Based Human Face Detection," IEEE Transaction on Pattern Analysis and Machine Intelligence 20(1):39-51, 1998.

[37] D.L. Swets and J.J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval," IEEE Transaction on Pattern Analysis and Machine Intelligence 18, No. 8:831-836, 96.

[38] M. Turk and A. Pentland, "Eigenfaces for Recognition," Journal Cognitive Neuro-science, 1991.

[39] J.J. Weng, "Crescepton and SHOSLIF: Towards Comprehensive Visual Learning," in [32], pp. 183-214, 1996.

[40] P.H. Winston, "Artificial Intelligence (third edition)," Addison-Wesley, 1992.

[41] C.F.J. Wu, "On the convergence properties of the EM algorithm," Annals of Statistics 11(1):95-103, 1983.