Chapter 4

Higher-Order Dependencies

4.1 Introduction

A fundamental problem of computer vision is to detect and recognize objects from cluttered backgrounds. The task can be especially difficult when objects cannot be easily segmented because their edges are merged with the contours of background elements and their texture and color cannot be used for segmentation. In this context, the most successful object recognition approaches are based on the coupling of local image measurements with global knowledge of the object shape. If we assume that such process must rely on statistical properties of the image, the statistical characterization of local visual features constitutes its first step.

The human visual system has shown a high proficiency at recognizing complex objects in cluttered backgrounds [84]. Recent experiments have shown that this characteristic includes perfect recognition capabilities even without accurate segmentation [15]. It has been argued that this fact is based, among other reasons, on the ability of unsupervised statistical learning of higher-order spatial structures [42] and the use of visual features of intermediate complexity [143].

Regarding computer vision, the dominant approach during the last years for designing object recognition systems has been the appearance-based approach. This approach, based on the statistical modeling of raw image data, has shown good performance in a variety of problems, but it requires the segmentation of the object and its performance is very limited in the presence of occlusion or cluttered backgrounds. One possible solution to this problem is the use of a bottom-up approach based on feedforward conjunctive feature detectors [87]. Depending on the application we can use different image features: local color histograms [18], local Gaussian derivatives [120] and geometric invariant features derived from them [123], etc. But in all these cases we must face a common problem: how to learn from data a suitable model for recognition. In most cases, it has been assumed that simple statistical descriptors like histograms [137] or Gaussian models [90] are a sufficient model for recognition. In other cases, more complex statistical models have been proposed [153].

In chapter 2, we evaluated different local feature descriptors which are usually used for current recognition approaches. In section 2.3 we presented several methods which make use of local descriptors to perform object recognition. And as seen, we have categorized all these local approaches into 3 different categories: (i) Feature based schemes, (ii) Higher-order schemes and (iii) Parts-based schemes. The main difference between these three categories is the spatial relationship among features. Thus, when we do not take into account the geometrical relationship among local features, a moderately noncomplex framework is obtained. As seen in chapter 2, the main problem of this kind of object representations is the potential ambiguity. Higher-order schemes appeared in order to solve this ambiguity by introducing information about the spatial relationship among local feature. However, a higher-order scheme produces a more complex framework. Finally, a parts-based scheme is considered when a subset of local features is grouped and they form an entity with a particular behaviour. Of course, a parts-based scheme contains high-level information about local features and this produces a highly complex framework.

In chapter 3 we have analyzed different object representation methods. Some of them adapted only to local positive data representations and some of them adapted to negative and non-negative data representations. Thus, chapter 3 contains feature based schemes since no information about spatial geometry and arrangement among local features is considered. Furthermore, experiments of chapter 3 have been done in closed environments where we do not have cluttered scenes or real occlusions (we only evaluated objects with artificial occlusions). In contrast to this, in this chapter we propose a novel local appearance modeling method for object detection and recognition in cluttered scenes. The approach is mainly based on modeling the joint distribution of local feature vectors at multiple salient points. The initial intractability of this distribution can be managed by factorizing it. In this way, the estimation of the statistical properties of the object image is reduced to the estimation of a number of low dimensional density functions that represent higher order dependencies among local features.

In section 4.2 we describe the general methodology for obtaining a reliable estimation of the joint distribution of local feature vectors at multiple salient points. We define the concept of k-tuple in order to represent the local appearance of the object at k different points as well as the statistical dependencies among them. We show how Independent Component Analysis (ICA) [64] (or section 3.6.1) can be used to transform local feature vectors to a new representation space that allows a reliable estimation of the k-tuples density function by factorizing it. Recognition is then performed using the Maximum *a posteriori* criterium. However, this method can be used for modeling a small set of objects, but degrades when we are interested in recognizing a large set of objects (or an object with very different appearances). In this case the best approach is to consider a specific model for each object (including its different appearances). For this case we introduce Class-Conditional ICA (see section 3.6.2) in order to be able of comparing among different models.

In section 4.3 we have tested our method in a closed environment where we recognize objects taken under different points of view (COIL-100 [97] database) and with different levels of handmade occlusions. Furthermore, our technique is also generalizable to real, complex and cluttered environments and we present some results of object detection in these scenarios with promising results. Finally, a very large statistically significant test (using the MNIST database) is used to illustrate the generality of feature representations in our scheme as well as explicitly demonstrating the advantage of modeling higher-order statistics using factorized joint distributions.

4.2 Methodology

We propose to model an object as a collection of local visual features. This choice is motivated by the need of getting a maximally invariant object description with respect to partial occlusion or nearby clutter. Features must be computed on the (a priori) most discriminative points of the object. In our implementation, we used the Harris operator [55] to detect interest points. We have used two kinds of visual features to represent objects: local color distributions [137] and the first 9 differential invariant jets [73] at each point. Using these features results in a local appearance model which is not only invariant to in-plane rotation (and translation) but is also robust with respect to partial occlusions as we shall see later. We must emphasize however that our methodology is not restricted to color and differential invariant jets and can be used for any local set of features, for example, curvature, edge-intensity, texture moments or even shape descriptors (see section 4.3.5).

Let i be an index corresponding to an interesting point in the image. Let $\mathbf{x}_i$ denote the feature vector of dimension n extracted at location i. Let $T_j = \{\mathbf{x}_i\}$ a set of features found in an image or region j. We will follow a Bayesian model-based object recognition strategy, so that we will assign a sample object T_j to a particular class M_l using the probability of misclassification as an error measure. It is well known that the solution to this problem is to assign T_j to the class that maximizes the *posterior probability*. This is called the Maximum a Posteriori or MAP solution. Using the Bayes rule we can formulate the posterior probability in terms of quantities which are easier to estimate, and the MAP solution takes the form:

$$C_{MAP} = \arg_{l=1,\ldots,k} \max\{P(T_j|M_l)P(M_l)\} \tag{4.1}$$

where $P(M_l)$ is usually called the *prior* probability, $P(T_j|M_l)$ is referred to as the *class-conditional probability* or, when seen as a function of the parameters, the *likelihood*.

For the class-conditional density in equation (4.1), it is intractable to model dependencies among all $\mathbf{x}_i$'s (even if correspondence is solved), yet to completely ignore these dependencies is to severely limit the modeling power of the probability densities. Objects frequently distinguish themselves not by individual regions (or parts), but by the relative location and comparative appearance of these regions. A tractable compromise between these two modeling extremes (which does not require correspondence) is to model the joint density of all k-tuples $\mathbf{X} = (\mathbf{x}^1, \ldots, \mathbf{x}^k)$'s in T_j, that is, of the features that are elements of a possible subset of cardinality k of T_j (with $k < n$ and $k << I$) [91]. We can consider all possible k-tuples or just a subset of them that fulfills a given condition.

4.2.1 Joint Distribution of k-tuples

Given a model object that is represented by I n-dimensional feature vectors, instead of modeling the total joint likelihood of $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_I$, which is an $(I \times n)$-dimensional distribution, we assume that the alternative distribution of all k-tuples is a good approximation. But for a given k-tuple, its joint density function, $P(\mathbf{x}^1, \ldots, \mathbf{x}^k)$ is a $(k \times n)$-dimensional distribution, which is still intractable. For example, in the case of using multi-dimensional histograms as an approximation of the joint distribution of image features with 20 histogram bins along each dimension, such a framework would require $20^{(k \times n)}$ bins [91]. A possible solution to this problem is to factorize this distribution into a product of low-dimensional distributions. We can achieve this factorization by transforming $\mathbf{x}$ into a new feature vector $\mathbf{s}$ whose components are (mostly) independent, and this transformation can be computed using Independent Component Analysis (ICA).

4.2.2 Density Factorization with ICA

ICA originated in the context of blind source separation [32, 67] to separate "independent causes" of a complex signal or mixture. The ICA of a n dimensional random vector $\mathbf{x}$ is the linear transform which minimizes the statistical dependence between its components. This representation in terms of independence proves useful in an important number of applications such as data analysis and compression, blind deconvolution, denoising, etc. Assuming the random vector we wish to represent through ICA has no noise, the ICA model can be expressed as

$$\mathbf{W}(\mathbf{x} - \bar{\mathbf{x}}) = \mathbf{s} \tag{4.2}$$

where $\mathbf{x}$ corresponds to the random vector representing our data, $\bar{\mathbf{x}}$ its mean, $\mathbf{s}$ is the random vector of independent components with dimension $m \leq n$, and $\mathbf{W}$ is the *filter* or *projection* matrix. This model can be also presented in terms of $\mathbf{A}$, the pseudoinverse of $\mathbf{W}$, called the *mixture* matrix. If the components of $\mathbf{x}$ are independent, at most one is Gaussian, and their densities are not reduced to a point-like mass, it can be proved that $\mathbf{W}$ is completely determined [64].

In practice, the estimation of $\mathbf{W}$ can be performed through the optimization of several objective functions such as likelihood, network entropy or mutual information [64]. Since mutual information is the Kullback-Leibler difference between a distribution and its marginal densities, we are obtaining a representation that best approximates (in the sense of Kullback-Leibler) the following rule [20]:

$$P(x_1, x_2, \ldots x_n) = |\det \mathbf{W}| P(s_1, s_2, \ldots s_n) \approx |\det \mathbf{W}| \prod_{j=1}^{n} P(s_j) \tag{4.3}$$

The original high-dimensional distribution is now factorized into a product of n one-dimensional distributions, with only small distortions expected.

In our case, by using ICA and given a set of feature vectors $\mathbf{x}$, we can learn the following linear mapping:

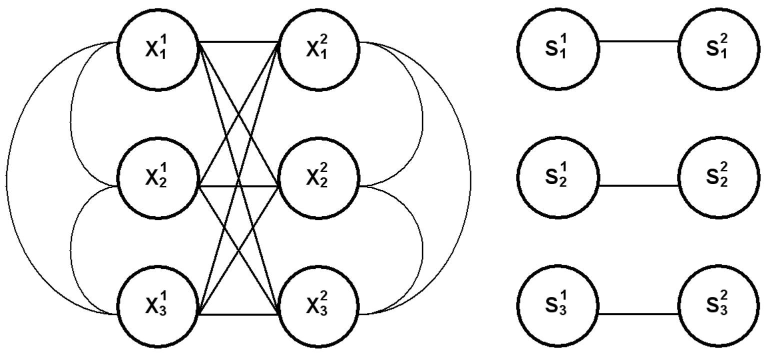
$$(\mathbf{x} - \bar{\mathbf{x}}) = \mathbf{A}\mathbf{s} \tag{4.4}$$

Under this mapping the density function of a feature vector $P(\mathbf{x})$ can be approximated by a product of n 1-dimensional density functions $P(\mathbf{s})$, and if we consider the distribution of a set of feature vectors which correspond to a k tuple,

$$P(\mathbf{x}^1, \ldots, \mathbf{x}^k) = \omega P(\mathbf{s}^1, \ldots, \mathbf{s}^k) \approx \omega \prod_{j=1}^{n} P(s_j^1, \ldots, s_j^k) \tag{4.5}$$

where ω is a normalizing constant. After factorization, we must estimate n k-dimensional density functions, and this becomes manageable if k is small, e.g., $k = 2$ or 3.

This factorization can be better understood from the point of view of a graphical model [14]. Figure (4.1) is a graphical model showing the dependencies between a pair of 3-dimensional data points $\mathbf{x}^1, \mathbf{x}^2$. The joint distribution over all nodes is 6-dimensional and all nodes are (potentially) interdependent. The basic approach towards obtaining a tractable distribution is to remove intra-component dependencies (vertical and diagonal links) leaving only inter-component dependencies (horizontal links). Ideally, a perfect ICA transform results in the graphical model shown in the right diagram where the s_1, s_2 and s_3 only have pair-wise inter-component dependencies. Therefore, the resulting factorization can be modeled by using parametric or non parametric k dimensional density models, as for example Gaussian mixture models.



(a) (b)

Figure 4.1: Graphical models: (a) fully-connected graph denoting no independence assumptions (b) the ICA-factorized model with pair-wise only dependencies.

Factorization of expression (4.5) can be seen under the point of view of graphical models. In figure (4.1) we see how intra-component dependencies are removed using an ICA transformation leaving only inter-component dependencies (horizontal links). Graphical model of figure (4.1.b) is the same as the one presented in figure (4.2.a). Graphical models are a mixture of graph theory and probability theory and offer an elegant formalism in which problems can be formulated and conditional relationships evaluated. We have an undirected graphical model (figure (4.2.a)) and we need to know the conditional relationships between its elements in order to provide a joint probability density function. A possible graphical model with conditional probabilities is shown in figure (4.2.b). Each arrow indicates the direction of causation. If we only consider nodes A and B of figure (4.2), the undirected graphical model is equivalent to a joint density of $P(A, B)$ and the directed graphical model is equivalent to $P(B|A)P(A)$. Arrows can be used in the other way round and the joint density would be equivalent to $P(A|B)P(B)$.

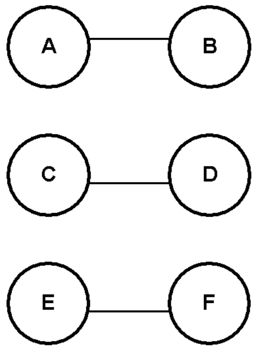
So that, if we have the graphical model of figure (4.2.b) and we want to know the joint density of all variables $P(A, B, C, D, E, F)$ there is a theorem in graphical models that,

$$P(\text{all variables}) = \prod_i P(v_i|\text{Pa}(v_i)) \tag{4.6}$$

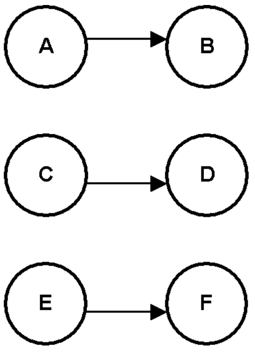
where v_i is each variable of the graphical model, and $\text{Pa}(v_i)$ means *the parents of node v_i*. For example, in figure (4.2.b), the father of node B is A but A has no father. As seen, this expression only takes into account probabilities where each node is conditioned on its parents. If we apply expression (4.6) with the graphical model of figure (4.2.b), we find that

$$\begin{aligned} P(A, B, C, D, E, F) &= P(B|A)P(A)P(D|C)P(C)P(F|E)P(E) \\ &= P(A, B)P(C, D)P(E, F) \end{aligned} \tag{4.7}$$

Now, we see that expression (4.5) is the same as the one obtained using graphical models (expression (4.7)). Thus, we have used graphical models to explain expression (4.5).



(a) Undirected graphical model.



(b) Directed graphical model.

Figure 4.2: Two graphical models. (a) Undirected graphical model corresponding to the graphical model of figure (4.1.b). (b) A directed version of the graphical model in (a) where we can use conditional probabilities and the arrow indicates the direction of causation.

A general scheme of our methodology is shown in figure (4.3). In this figure we have an object and its corresponding keypoints. Then, we extract a set of $k = 3$ tuples and we use ICA to obtain a reduced space. Then, m mixture of Gaussians in a k dimensional space are used to model joint distributions of $k = 3$ tuples.

Finally, we should note that since we have a large collection of objects, we will use the Class-Conditional ICA introduced in section 3.6.2 to compare different ICA spaces.

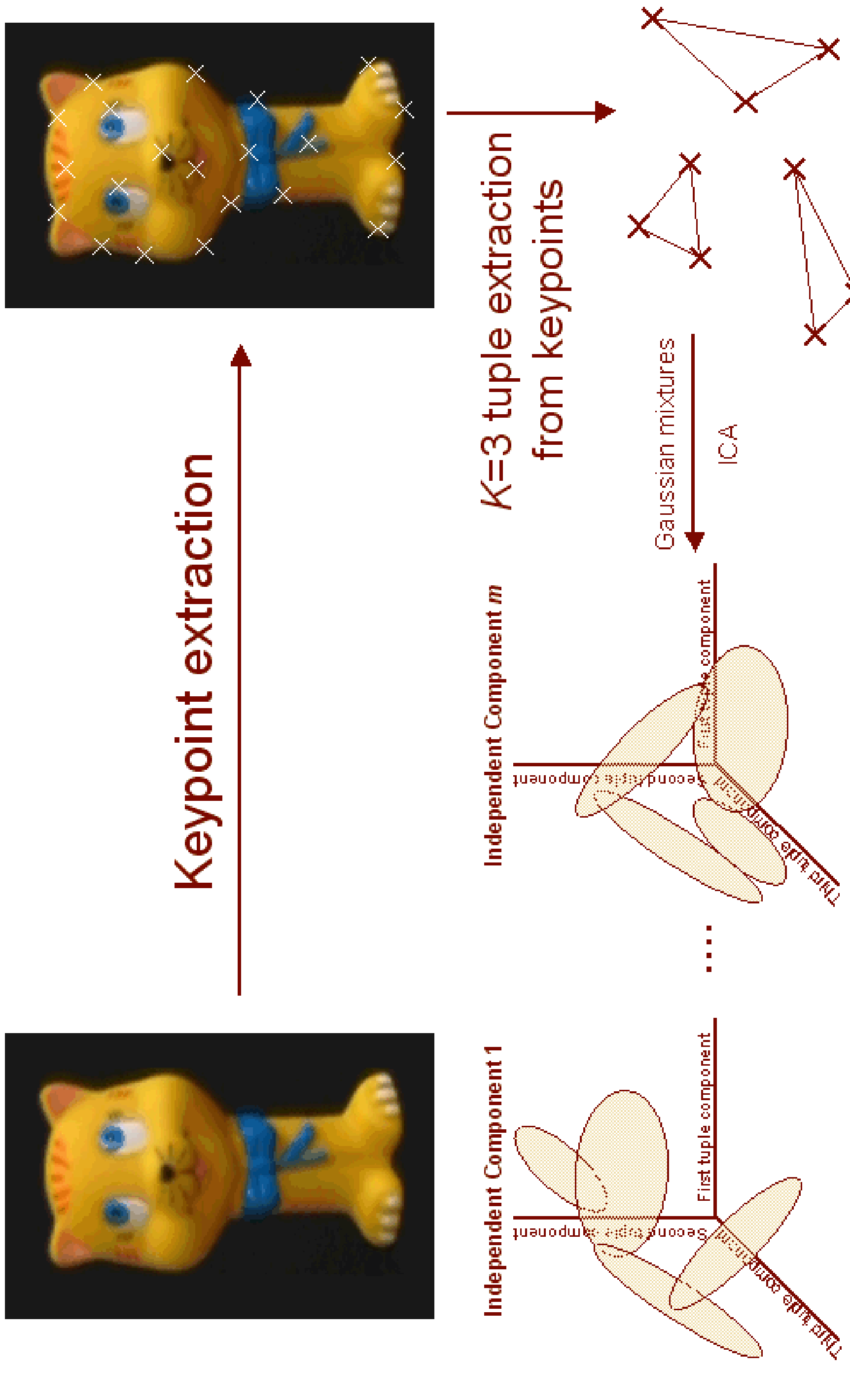


Figure 4.3: System diagram for k-tuple density factorization using ICA and Gaussian mixture models.

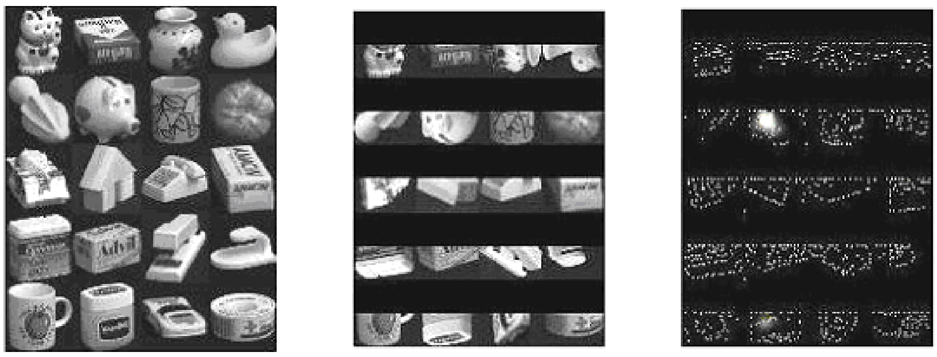
4.3 Experimental Results

Our experiments are based on extracting local information from a set of interest points using the well-known Harris operator [55, 124] and modeling possible high-order dependencies of local features. In the first experiment we extracted the first 9 differential invariant jets [73] at each point as the corresponding feature vector x. Using these jets as our feature vectors results in a local appearance model which is not only invariant to in-plane rotation (and translation) but is also robust with respect to partial occlusions as we shall see later. We must emphasize however that our methodology is not restricted to differential invariant jets and can in principal be used for any local set of features, for example, color, curvature, edge-intensity, texture moments or even shape descriptors (see section 4.3.6). We then performed ICA to get $m < 9$ independent components for the feature vectors (jets). In our experiments, we considered the set of $k = 1, 2, 3$ order tuples, resulting in a set of 1D, 2D and 3D Gaussian mixture models which were used to model 1-tuple, 2-tuple and 3-tuple joint component densities. Initial experiments were done using multi-dimensional histograms as a non-parametric approximation of the joint distribution of tuples but results were not as satisfactory as parametric mixture models. Once an ICA space is defined, we used the definition of class-conditional ICA of Equation (3.73) in order to perform object classification.

4.3.1 From Histograms to Mixture Models

A simple and direct method to estimate a joint distribution of k-tuples is through the use of histogram representations. As seen in section 2.1.3, histograms have been widely used as a first approximation of a probability density function. So that, representing $k = 2$ tuples and using 32 histogram bins, we require of about $32^{9\times 2}$ bins. Such a representation of $32^{9\times 2}$ bins is very expensive in terms of computational costs. Also, if we assume a histogram based representation our model might not be extended to higher-order models such as $k = 3$ or $k = 4$ because the computational costs required for these representations would be tremendous. In order to deal with these representations, we used PCA to reduce the dimensionality of the problem and ICA to obtain "independent" vectors. Such dimensionality reduction produces a tractable space in terms of computational costs. Thus, our principal need is to determine a correct subspace dimension (dimension m) in terms of reducing the original data vectors (dimension n) preserving the maximum amount of information. We conducted an experiment in order to find a correct ICA dimension which preserves the data of the original space and obtains the best performance in terms of recognition and detection. Here, we should note that we first apply PCA to reduce the dimensionality of the original space and then we apply ICA to obtain its independence. However, we apply PCA and ICA at the same time and we use the term *ICA dimension* instead of *PCA dimension*.

Our experimental results are mainly based on the COIL100 object database. In order to evaluate which ICA dimension performs good in our framework, we used 20 objects from the Columbia Object Image Library (COIL) and we constructed an artificial situation where objects are transformed by pose change, a planar rotation and followed by 50% occlusion. We tested our method under this set of unfavourable conditions and we present our initial results in figure (4.4) where the raw output for object "piggy bank" can be seen.



(a) 20 Objects (b) Testing artificial objects (c) Likelihood map

Figure 4.4: Synthetic "cluttered" scene and a detection example. (a) The synthetic test image of 20 objects from COIL; (b) The rotated and occluded version of (a); (c) The likelihood map for detecting "piggy bank" in (b). The white dots are the interest points and high-likelihood points are highlighted.

Objects of figure (4.4.a) have been used in order to determine which ICA dimension perfoms better in our framework. We evaluated from $m = 1$ to $m = 9$ Independent Components of our feature vectors (9 dimensional feature jets). As said before, PCA is used for dimensionality reduction and after we apply ICA to obtain independence. For the original $n = 9$ dimensional jets, histogram factorization along feature components is not a valid solution since the independence assumption on the differential invariant jets does not hold in general. ICA is introduced to obtain a factored model but we can obtain a reduced space lower than $n = 9$ dimensions. Detection performance was measured by the average rank of the accumulated regional likelihood for the object model (the ground truth object location was used in this particular case). Figure (4.5) depicts the clear improvement introduced by ICA with respect to the use of the original jets in one particular case. This experiment demonstrates that our nine dimensional data vectors can be represented using only 3 dimensional projected vectors obtained using PCA and ICA. And we have seen that this new representation is able to obtain 100% in terms of recognition rates. Since our nine dimensional data vectors are obtained from nine invariant jets, we think that the intrinsic dimension of our problem is 3 because the invariant jets are built by different convolution filters and they might contain repeated and redundant information. Results of figure (4.5) are presented in terms of ranks where, for example, rank one means that a target object was correctly classified at first place. Rank two means that a target object was correctly classified at second position. So that, figure (4.5) contains the average detection rank according to the ICA dimension used.

Experimental results are based on a training and a testing set. Each object model is trained only using one instance per object. For testing purposes, we have generated

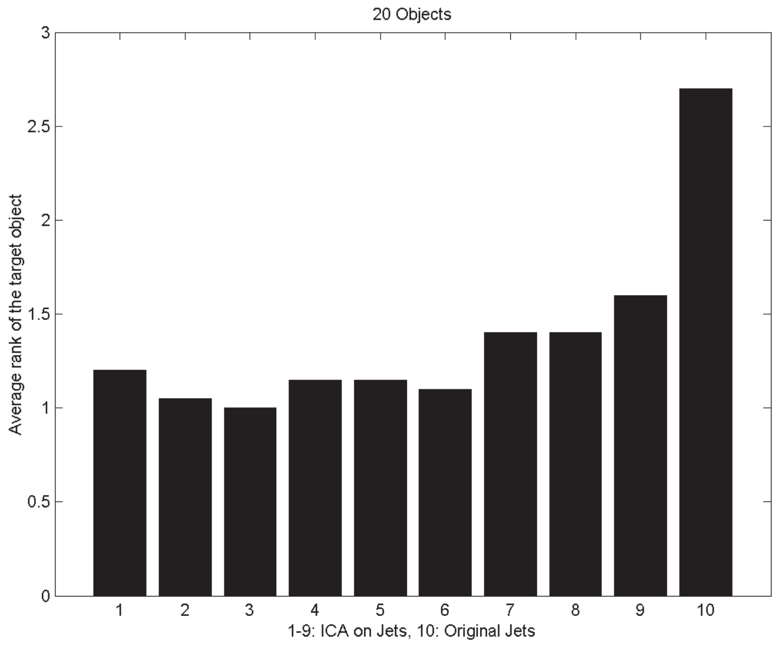


Figure 4.5: The average detection rank of 20 objects of figure (4.4.a) using different Independent Components (m=1,2,... 9) versus original $n = 9$ dimensional jets (shown as the rightmost bar). Dataset used in this particular example: COIL 20 objects.

a testing set consisting of four new instances of each object captured from other points of view (each testing instance of the object is rotated 5 degrees in azimuth apart from the other). Thus, the following experiments will show different recognition rates. Tables containing these results can be understood as follows: row 1 with label *Instance 1* is the training instance used to create each object model and from row 2 to row 5 are four new testing instances of each object — each of them rotated by 5 degrees from the previous instance. Then, we introduce a new row containing a percentage for the training set with label *Train* and a label *Test* which indicates the recognition rate obtained with the testing set

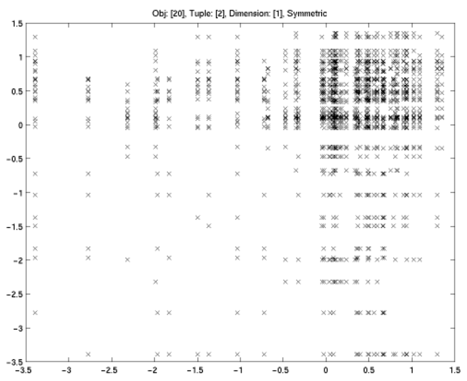
We want to compare the performance of our method using the original data vectors with respect to the reduced data vectors obtained using an ICA transformation. To do this, we took the first 20 objects of the COIL100 and we compared both performances with the training and testing datasets. We took one instance per object and we created our object models using a $k = 2$ tuple histogram representation. We took four new object instances each of them rotated by 5 degrees from the previous instance to build up our testing dataset. We calculate a set of keypoints per object and we extract $k = 2$ tuples. From all the possible $k = 2$ tuples, we have selected those tuples which their two keypoints are closer (the distance between them is less than 25 pixels, see section 4.3.5). Objects contain about 100 keypoints and the number of possible $k = 2$ tuples is extremely huge (about 5000 tuples). So that, using tuples with closest keypoints we reduced this huge number of tuples to 1000. Choosing tuples with the closest keypoints produces a robust technique that can naturally deal with different levels of occlusions (see section 4.3.5). The original space composed of the responses of 9 invariant jets is modeled using one histogram per dimension. Since we are modeling $k = 2$ tuples, each histogram is bidimensional. In this particular experiment, 32 histogram bins are used for each dimensional histogram (9 histograms). Thus, each histogram requires of $32^2 = 1024$ bins. In the previous experiment, we found that a correct ICA dimension is $m = 3$. Thus, we reduced the $n = 9$ dimensional feature vectors to an ICA dimension of $m = 3$. Then, we used 3 histograms to model each $m = 3$ subspace dimension. Results of these two approaches are shown in table (4.1) where we can also see the effect of increasing the tuple order (from $k = 1$ to $k = 3$).

	9D k tuples			3D k tuples		
Instances	$k=1$	$k=2$	$k=3$	$k=1$	$k=2$	$k=3$
Instance 1	3	17	20	16	20	20
Instance 2	1	4	4	8	7	6
Instance 3	1	4	4	6	5	2
Instance 4	1	4	5	4	5	4
Instance 5	1	4	3	6	4	3
Train	15%	85%	100%	80%	100%	100%
Test	5%	20%	20%	30%	26.25%	18.75%

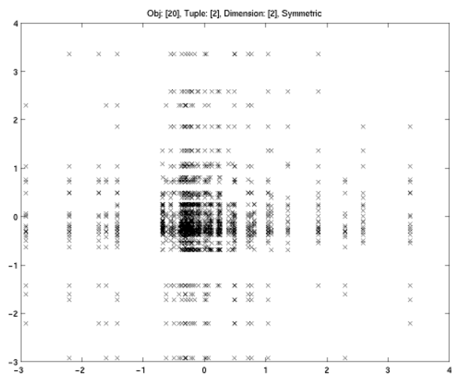
Table 4.1: Recognition rates when we consider the original $9D$ invariant jets with independence assumption and the independent $3D$ ICA projected vectors. First instance is used for training, and 4 new instances are used for testing.

Table (4.1) reflects two differents behaviours: (i) Results obtained using 9 dimensional jets are poor, (ii) Even working with independent distributions, recognition results are not very convincing. As expected, it is clear that if we work in the original space of 9 dimensional vectors and if we assume statistical independence between vectors, we are implicitly assuming a bad framework for object recognition. It is clear that the original space is not independent and results should be poor. When we introduce an ICA transformation, recognition rates are slightly increased but only when recognition is done using the same training instance. Under new unseen object instances of the learned objects, our method fails and it does not reflect robustness under unlearned instances. Table (4.1) also reflects a $k = 3$ tuple model but its results are worse than the $k = 2$ tuple model. From table (4.1), when we increase the order of our tuples (k), recognition results are automatically diminished. This bad behaviour has been deeply analyzed in order to find a justification. For this, we have found that our tuple distributions are graphically represented as shown in figure (4.6).

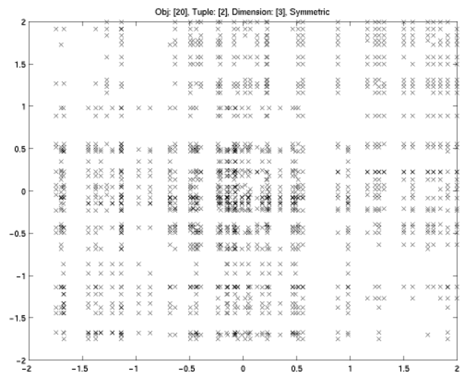
Figure (4.6) shows three tuple distributions that are difficult to model. As seen in this figure, points are not concentrated in specific regions and they are not organized as an homogeneous cloud of elements. This spatial organization difficults its modelization. Figure (4.6.a) shows a big region that concentrates a large number of points and two different tails. Figure (4.6.b) also reflects a similar behaviour. However, figure (4.6.c) reflects an equally distributed region of points. These 3 figures



(a) Dimension 1.



(b) Dimension 2.



(c) Dimension 3.

Figure 4.6: Three 2 dimensional spaces representing the tuple space of one object.

show that the spatial configuration of points corresponding to our tuple space is very important. Thus, this is the main reason that histograms are not well suited to model these highly complex tuple spaces. As we will see later, this particular spatial configuration of points in conjunction with a histogram representation is the reason that our higher-order model of $k = 3$ tuples does not improve the $k = 2$ case.

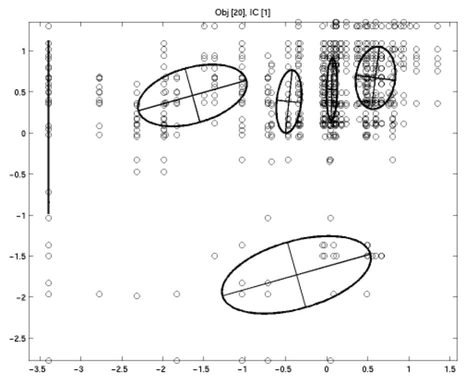
Since histogram representations need to tune a bunch of internal parameters and we have highly complex spatial distributions of points, we used another representation more adaptive. If we try to represent these spaces with histograms, one possible solution would be to increase the number of histogram bins in order to capture tails or complex point distributions. Then, we will obtain a histogram resolution where we do not lose information. Instead of increasing the number of histogram bins we decided to use a mixture of Gaussians to represent this space. The main reason to introduce a mixture of Gaussians is that we will need less computational space to store our models. Also, another important reason is that our data distributions contain a great diversity of behaviours (big regions, tails, sparse regions, etc.), so that, each of them can be captured using a Gaussian or a combination of them.

	5 Mixtures 3D k tuples			10 Mixtures 3D k tuples		
Instances	$k = 1$	$k = 2$	$k = 3$	$k = 1$	$k = 2$	$k = 3$
Instance 1	20	17	16	20	19	18
Instance 2	13	14	14	14	14	15
Instance 3	11	14	15	10	13	14
Instance 4	12	12	12	9	11	12
Instance 5	9	10	12	7	10	12
Train	100%	85%	80%	100%	95%	90%
Test	56.25%	62.5%	66.25%	50%	60%	66.25%

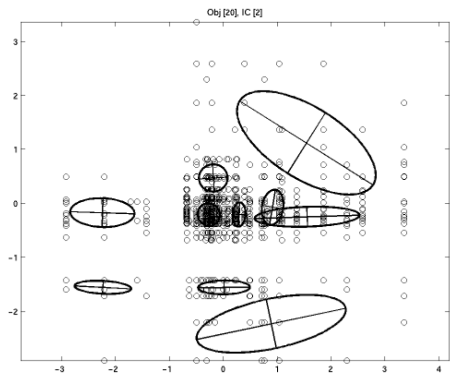
Table 4.2: Recognition rates when we consider a mixture of Gaussians. Experiments use 3 independent components that were obtained using an ICA transformation. We present a mixture of Gaussians using 5 and 10 Gaussians.

In table (4.2) we expose a comparison of 2 different Gaussian mixture models. We used 5 and 10 Gaussians to model all tuple spaces (from $k = 1$ to $k = 3$) and results are significantly better than the ones presented in table (4.1). We should note that a $k = 2$ tuple model produces data redundancy because we have to consider two tuples $(\mathbf{x}^1, \mathbf{x}^2)$ and $(\mathbf{x}^2, \mathbf{x}^1)$ where $\mathbf{x}^1$ and $\mathbf{x}^2$ are two feature vectors which are extracted from two keypoints of an object. However, a $k = 3$ tuple model introduces a major degree of data redundancy because we have to consider 6 tuples (all possible combinations between 3 local keypoints): $(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3)$, $(\mathbf{x}^1, \mathbf{x}^3, \mathbf{x}^2)$, $(\mathbf{x}^2, \mathbf{x}^1, \mathbf{x}^3)$, $(\mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^1)$, $(\mathbf{x}^3, \mathbf{x}^2, \mathbf{x}^1)$ and $(\mathbf{x}^3, \mathbf{x}^1, \mathbf{x}^2)$. Thus, we are modeling symmetric data and the amount of data redundancy that can be generated with a $k = 3$ tuple model is extremely important. Then, we will have problems to learn these data distributions using Gaussian mixture models. To avoid this data redundancy and increase the speed of the learning process, we decided to learn our joint distributions using a fixed order of the elements of each tuple. If we are able to assign an order to each element of a tuple, we will have a simpler tuple point instead of 6 (in a $k = 3$ tuple model) and it will be easier to learn the whole tuple space. Since our extracted keypoints are obtained using the well-known Harris operator [55] and it is based on the first order Gaussian derivatives, we are able to determine the order of the keypoints of a tuple. Given a specific keypoint in an image, if G_x is the first Gaussian derivative extracted from this keypoint along x axis, G_y the first Gaussian derivative extracted from the same keypoint along the y axis and G_{xy} is the second order Gaussian derivative extracted from the same keypoint along x and y axes, then we are able to obtain a value $\tau = G_x^2 \times G_y^2 - G_{xy}^2$. τ usually reflects whether a keypoint is located in an homogeneous region or in an abrupted region because it is based on the Gaussian derivatives [123]. So that, each keypoint in an image will have its corresponding τ value which will be used to determine the specific order of the elements of a given tuple. Now, our joint distributions contain less tuples and they should be easier to learn.

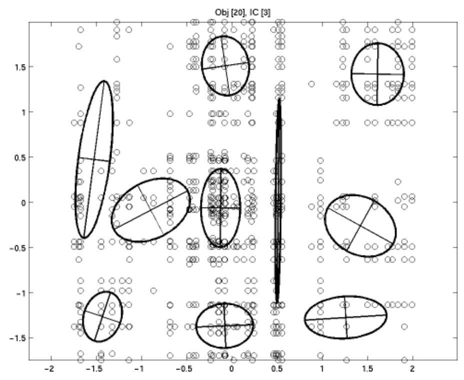
The three joint distributions presented in figure (4.6) are also shown in figure (4.7) assuming that tuples are sorted according to the above explained method. In this particular case, the number of $k = 2$ tuples is reduced significantly. Table (4.3) shows the recognition results obtained using tuples with ordered keypoints. If we compare results of tables (4.2) and (4.3) we see that results do not change significantly. However, in the last case, we learned the Gaussian mixture models faster than using unordered tuples.



(a) Dimension 1.



(b) Dimension 2.



(c) Dimension 3.

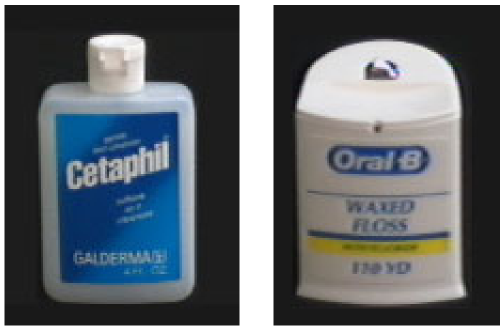
Figure 4.7: Three 2 dimensional spaces representing one of the objects of the database where keypoints of tuples are ordered. We also show how a mixture of Gaussians can be adapted to our tuple distributions.

Using a mixture of Gaussians to model joint distributions imply that we should be aware of the number of components (Gaussians). Thus, a highly complex joint distribution would require more Gaussian components than a noncomplex joint distribution. In our case, this is very important because our data distributions are not easy to model (see figure (4.7)) and if we choose an inappropiate number of Gaussians we will generate an incorrect model. As example of this important fact, we present the tuples of one object with respect to the tuples of another object. Figure (4.9) shows two tuple spaces merged in one space. One tuple space is modeled using a mixture of Gaussians. We are able to appreciate that there are several tuples which

	5 Mixtures 3D k tuples			10 Mixtures 3D k tuples		
Instances	$k = 1$	$k = 2$	$k = 3$	$k = 1$	$k = 2$	$k = 3$
Instance 1	20	19	19	20	20	20
Instance 2	13	13	15	11	13	14
Instance 3	13	13	14	12	12	13
Instance 4	12	12	13	8	10	12
Instance 5	10	11	11	9	8	13
Train	100%	95%	95%	100%	100%	100%
Test	60%	61.25%	66.25%	50%	53.75%	65%

Table 4.3: Recognition rates when we consider a mixture of Gaussians. Experiments use 3 independent components which were obtained using an ICA transformation. We present a mixture of Gaussians using 5 and 10 Gaussians. This experiments contains ordered tuples.

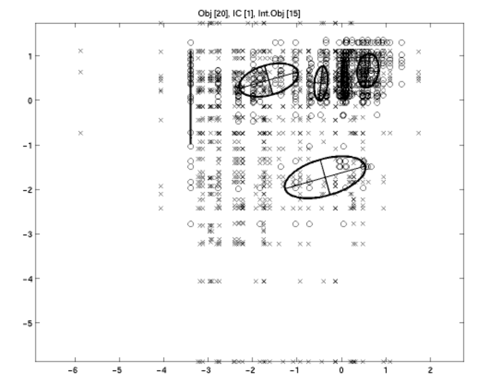
share common regions but they belong to different objects. Thus, there are intersecting regions between both objects. Of course, this behaviour is justified because both objects have similar local appearance regions leading to generate similar or even identical tuples. We present in figure (4.8) the two objects which have been used to extract the above presented tuple spaces (figure (4.9)). Now, we see that both objects are similar in terms of local appearance and this fact produces intersecting regions in their joint distributions.



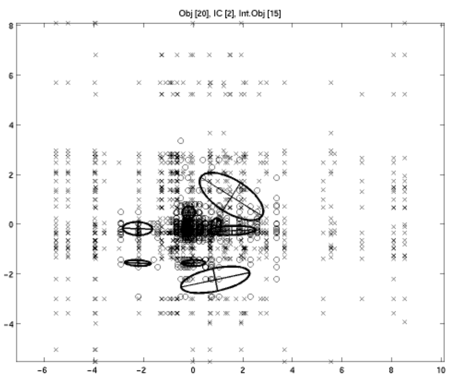
(a) Object 1. (b) Object 2.

Figure 4.8: Two objects of our database. These two objects are similar in terms of local appearance and their respective tuple spaces are shown in figure (4.9).

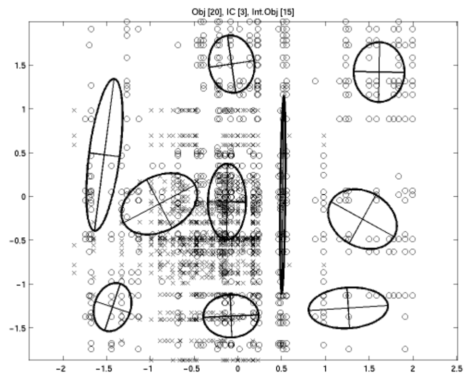
As seen, our objects contain local appearance regions which might difficult the learning of Gaussian mixture models. The central problem of using a mixture of Gaussians as a model is, of course, the choice of the number of components (also known as "model-order selection"). In our experiments, we used an adaptive mixture model based on the Minimum Description Length (MDL) [139] optimality criterion used to fit the data in each case with the "right" number of components. Results using the MDL criterion are presented in table (4.4). This table shows that a MDL criterion to choose the number of Gaussians definitely enhances the recognition performance,



(a) Dimension 1.



(b) Dimension 2.



(c) Dimension 3.

Figure 4.9: Three joint distributions of tuples of two objects corresponding to each $m = 3$ independent components. One of the objects is modeled using a mixture of Gaussians (circles) that is also seen in figure (4.7). Crosses represent tuples of the non-modeled object. We see a high level of intersection between these two tuple spaces.

undoubtedly through an increase in accuracy in modeling the joint distributions.

Even though results from table (4.4) show an enhancement with respect to all previous results, we can not state that our technique is robust because the best obtained performance is about 71.25%. Analyzing our object database we found that nearly all errors are localized when we compare one homogeneous object with a complex one. Since our method is based on the extraction of keypoints, an homogeneous object contains less keypoints than an object with abrupted regions. Thus, these objects contain less tuples and our mixture of Gaussians can not generalize from a reduced set of keypoints. A possible solution to this problem is to fix a number of keypoints per object in order to force a minimum number of keypoints for homogeneous objects (i.e. at least 50 keypoints per object). If we consider at least 50 keypoints per object, recognition results are increased as shown in table (4.5).

	MDL Estimation 3D k tuples		
Instances	$k = 1$	$k = 2$	$k = 3$
Instance 1	20	20	20
Instance 2	13	14	16
Instance 3	12	12	14
Instance 4	10	11	14
Instance 5	11	10	13
Train	100%	100%	100%
Test	57.5%	58.75%	71.25%

Table 4.4: Recognition rates when the number of Gaussians is estimated using the Minimum Description Length (MDL).

	MDL Estimation 3D k tuples			MDL Estimation 3D k tuples	
Instances	$k = 1$	$k = 2$	$k = 3$	$k = 2$	$k = 3$
Instance 1	20	20	20	20	20
Instance 2	14	15	16	16	18
Instance 3	13	14	15	14	16
Instance 4	12	13	15	13	15
Instance 5	11	12	14	13	14
Train	100%	100%	100%	100%	100%
Test	62.5%	67.5%	75.0%	70%	78.75%

Table 4.5: Recognition rates when the number of Gaussians is estimated using the Minimum Description Length (MDL) and homogeneous objects contain a significant number of keypoints (at least 50 keypoints per object). The 3 columns in the left show results obtained using the product of probabilities, and the 2 columns in the right show results when a voting scheme is used.

The first left-most 3 columns of table (4.5) show how results are slighty improved when we force to obtain more keypoints with homogeneous objects. We have also experimentally tested that we can improve classification results using a voting scheme instead of using the usual product of probabilities as seen in equation (4.5). Objects are usually described using 100 keypoints approximately and we can work with about 2000 $k = 2$ tuples or 20000 $k = 3$ tuples. Since we have a large amount of tuples in both cases, we can classify an object according to the number of tuples that are correctly classified. Thus, introducing a voting scheme where each tuple votes for one candidate training object, we can avoid some effects derived from the mixture of Gaussians model. A model based on a mixture of Gaussians can suffer from an overflow of probabilities since some covariance matrices can be very narrow (see figure (4.9)). Thus, this scheme might generate a very high probability value that can be propagated over other tuples of the model and the final probability of the object can be severely affected. A voting scheme reduces this effect because each tuple is treated independently. Using this voting scheme, each training object will have its corresponding counter and when all testing tuples have voted we will choose the most voted training object as the candidate for the classification. The two right-most columns of table (4.5) show classification results when we consider this voting scheme. We have to take in mind that a voting scheme can only be used when we have a fixed number of training objects and the number of tuples of a test object is high.

4.3.2 Appearance + Color Models

Previous section shows results using appearance information (9 invariant jets) but the final performance seems not to be very convincing. In this section we want to introduce more information in our appearance model in order to improve appearance based results. Furthermore, we introduce more objects in our analysis. In this section, experiments are based on 100 objects from the Columbia Object Image Library (COIL-100) [97]. We repeated the same experiment of the previous section but using 100 objects and we find that appearance-based models (ie. using monochrome-based invariant jets) are not very satisfactory (see table (4.6)) therefore we introduced a hybrid appearance/color model by introducing the mean color of each normalized channel (R,G,B) obtained from a circular region defined around each interest point. Although color histograms [137] can also be used, given the local nature of the representation, we limited it to the main dominant color in the surrounding region. Recognition rates using $k = 1$ tuples using appearance, color and a hybrid appearance/color model are presented in table (4.6).

$k = 1$ tuples				
	Appearance (3D)	Color (3D)	Appearance + Color (3D)	Appearance + Color (4D)
Instance 1	85	31	93	89
Instance 2	38	25	52	66
Instance 3	33	23	50	55
Instance 4	28	18	49	57
Instance 5	28	17	42	52
Train	85%	31%	93%	89%
Test	31.75%	20.75%	48.25%	57.5%

Table 4.6: Recognition rates with 100 objects using $k = 1$ tuples and considering all the possible models. All spaces are modeled using a mixture of 10 Gaussians.

As noted in Table (4.6), feature vectors of the appearance model (first column of results) are reduced from a $n = 9$ dimensional space to a $m = 3$ dimensional ICA space since this ICA dimension is the best one as seen in the previous experiment (see figure (4.5)). In light of the previous experiments (see figure (4.5)), we strongly believe that our 9-dimensional jets have an intrinsic dimensionality of 3 components. The addition of (dominant) color introduces essentially one degree of freedom (information) to the model and we would expect that $m = 4$ dimensional ICA spaces would be the best (as in fact they were found to be). In Table (4.6) we present recognition results considering a projected ICA space of 4 dimensions. Since considering $k = 1$ tuples is the same as evaluating the probability of a single point to appear in one object model, recognition results are poor. Tables (4.7) and (4.8) show recognition rates when considering higher-order models with $k = 2$ and $k = 3$ tuples.

$k = 2$ tuples				
	Appearance (3D)	Color	Appearance + Color (3D)	Appearance + Color (4D)
Instance 1	98	54	99	99
Instance 2	44	32	70	73
Instance 3	33	30	68	71
Instance 4	28	33	58	63
Instance 5	29	29	62	64
Train	98%	54%	99%	99%
Test	33.5%	31%	64.5%	67.75%

Table 4.7: Recognition rates with 100 objects using $k = 2$ tuples and considering all the possible models. All spaces are modeled using a mixture of 10 Gaussians.

$k = 3$ tuples				
	Appearance (3D)	Color	Appearance + Color (3D)	Appearance + Color (4D)
Instance 1	100	83	100	100
Instance 2	53	68	69	85
Instance 3	42	63	67	81
Instance 4	41	60	64	78
Instance 5	36	58	58	74
Train	100%	83%	100%	100%
Test	43%	62.25%	64.5%	79.50%

Table 4.8: Recognition rates with 100 objects using $k = 3$ tuples. All spaces are modeled using a mixture of 10 Gaussians.

It is quite clear as seen in tables (4.6), (4.7), and (4.8), that as the number of interest points per tuple is increased and hence more mutual information about the local appearance jets are modeled, the recognition rates are improved.

4.3.3 Incorporating Local Geometry

When considering $k = 3$ tuples, we can also take into account the geometry of a tuple — *i.e.* how the three points in a tuple are arranged spatially and use this information to perhaps increase our recognition capacity. Rank ordering the interest points in a tuple as (p_1, p_2, p_3), we can use the following geometrical descriptors:[1]

- Distance L_{12}: Distance between keypoint p_1 and keypoint p_2.
- Distance L_{13}: Distance between keypoint p_1 and keypoint p_3.
- Angle α: Angle between the connecting line p_1, p_2 and the connecting line p_1, p_3.

However, a feature space using these three geometric descriptors alone (L_{12}, L_{13}, $\cos(\alpha)$), results in a rather poor recognition rate of 50%. We thus conclude that

[1]Keypoints are ordered according to the output of the Harris operator, which is proportional to the principal curvatures of the intensity surface.[55]

geometry information alone is not sufficient for recognition. However, tuple geometry might help to disambiguate confusing appearance-based cases and therefore it helps to create a hybrid appearance/geometry mixture model.

Since the combined color and appearance model was found to be the best in previous experiments, we use this model as a first step classifier in our experiments and then used the geometric model as a second step classifier to resolve any possible ambiguities found by the first classifier. Table (4.9) shows the recognition rates when considering this new two-stage classifier design. We can see that the resulting (two-stage) recognition rates are better than those obtained with the (single stage) appearance/color model alone (see table (4.8)).

	Classifier 1: Appearance + Color Model (4D) Classifier 2: Geometry Model	
	No MDL estimation	MDL estimation
Instance 1	100	100
Instance 2	88	96
Instance 3	86	94
Instance 4	81	91
Instance 5	77	86
Train	100%	100%
Test	83%	91.75%

Table 4.9: Recognition rates with 100 objects using $k = 3$ tuples. This table reflects a two-stage classifier: the first for the appearance/color model and the second for the geometry model. First column is an appearance/color model using a mixture of 10 Gaussians and the geometry model used 2 Gaussians. Second column uses a MDL estimator.

Comparing both columns of table (4.9), we see that MDL has definitely enhanced the recognition performance of our system, undoubtedly through an increase in accuracy in modeling the joint distributions.

4.3.4 Invariance to Partial Occlusion

As an illustration of our object classification framework, a representative visual example is shown in figure (4.10) where different likelihood detection maps (based on joint density functions) are shown when the particular object model of figure (4.10.a) is the search target. Note that the hybrid appearance/color model correctly localizes the target object from among the 20 object candidates arranged in the test image.

Object detection and classification techniques should be robust under the presence of occlusions. Since our technique is based on local tuples obtained from a set of interest points, occlusions should be easy to deal with. In this particular case, we occluded parts of the test objects using various quadrants assuming that the rest of the object would be sufficient to recover the original identity. Results are presented in table (4.10) where we use a first step classifier model based on appearance and color information and a second step classifier based on geometry, both using the MDL criterion to set the number of Gaussians. Graphical results can be seen in figure (4.11) and taking as a reference figure (4.10) we see that objects with occlusions can

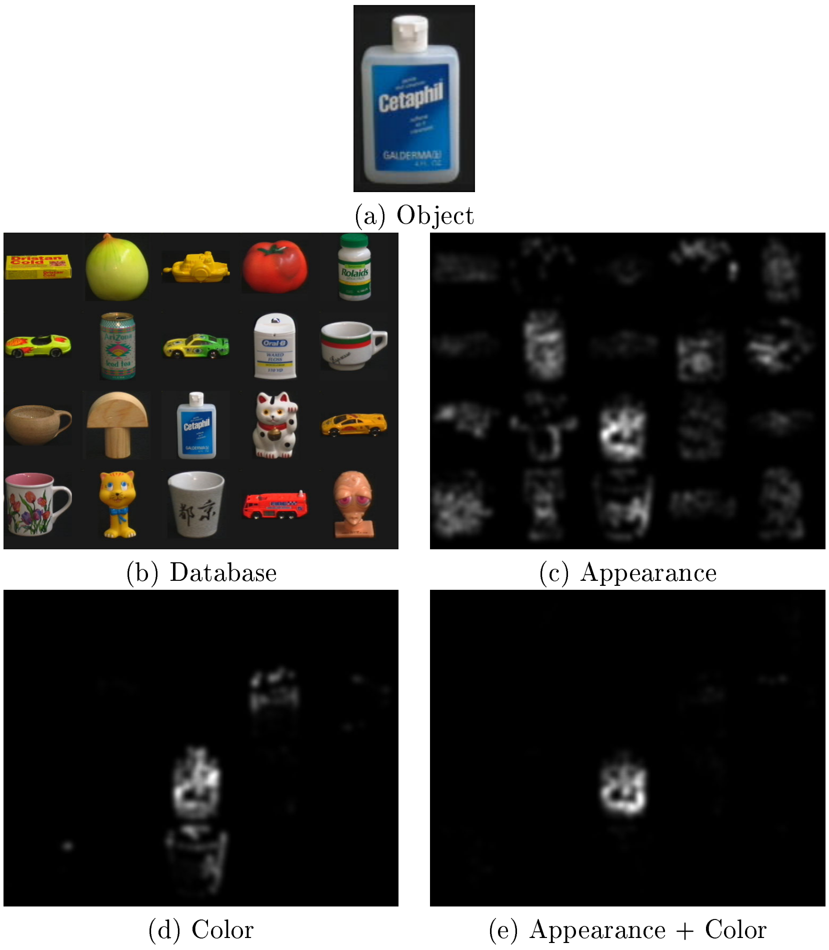


Figure 4.10: Likelihood maps (c), (d) and (e) of different models for the input image (b) when using object (a) as the search target. Note: We are only showing the first 20 objects of COIL-100.

also be detected.

Table (4.10) shows that when one quadrant is missing, recognition rates are acceptable since we obtain an average rate of 75%. When two quadrants are missing, recognition rates decrease to an average of 52% but this is still a good trade-off between the level of occlusion and recognition rates. We should take into account that we are testing object instances that are not previously learned (ie. images taken from other points of view).

We also evaluated our approach in real laboratory scenes where deformable objects can appear under various configurations, poses and occlusions. Figure (4.12) shows the objects and images used for this experiment. Two different objects with similar colors but different shapes were learned in order to detect them in a complex environ-

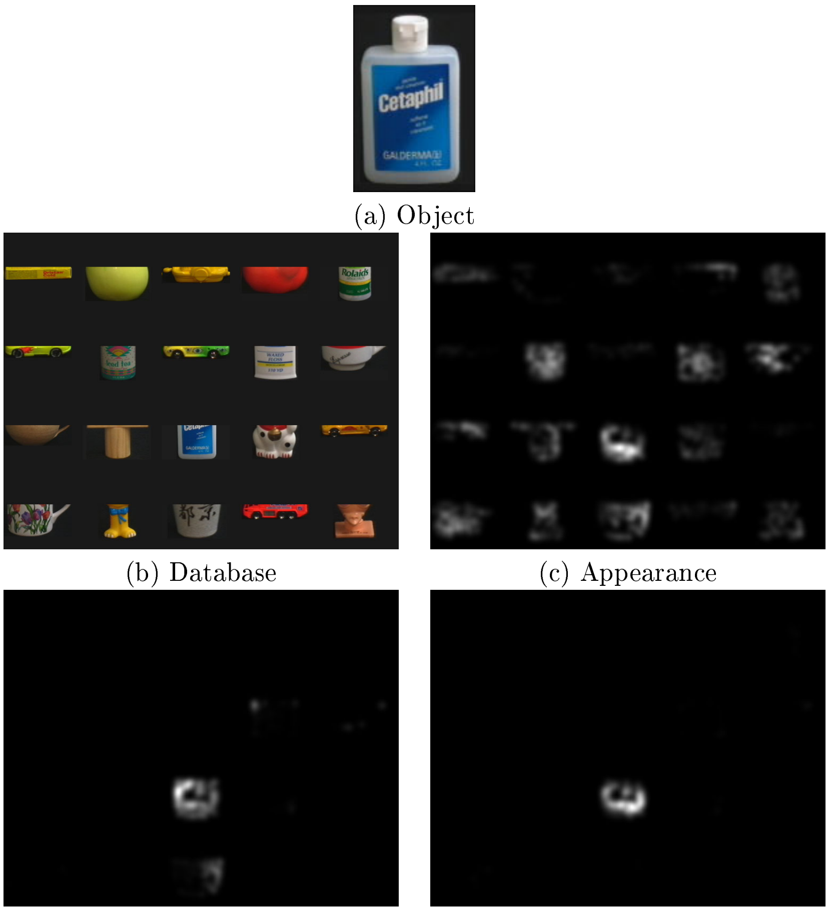


Figure 4.11: Experiment with occlusions. Likelihood maps (c), (d) and (e) of different models for the input image (b) when using object (a) as the search target. Note: We are only showing the first 20 objects of COIL-100.

ment. As noted in this figure (4.12), objects can be difficult to recognize since they contain different levels of occlusions and can be seen under different poses. Despite these difficulties, objects are correctly detected indicating a good level of robustness of our system.

Finally, we have also tested our system with real and cluttered scenes where objects can be affected by different natural factors. This is the case presented in figure (4.13) which shows the modeling and subsequent detection of the US Pentagon building before and after the September 11 terrorist attack. Figure (4.13.a) presents a real image of a pentagon building and figure (4.13.b) shows the extracted building used for our learning and modeling. Figure (4.13.c) depicts a test image which was taken after the bombing debris was cleared away by the cleanup crew (leaving a whole

	Occlusions considered							
	Q_1	Q_2	Q_3	Q_4	Q_1+ Q_2	Q_2+ Q_3	Q_3+ Q_4	Q_4+ Q_1
Instance 1	100	100	100	100	94	97	92	93
Instance 2	83	83	70	80	60	55	62	54
Instance 3	80	82	67	82	55	51	49	49
Instance 4	75	74	59	76	57	57	54	58
Instance 5	71	65	65	67	51	48	45	54
Train	100%	100%	100%	100%	94%	97%	92%	93%
Test	77.25%	76%	65.25%	76.25%	55.75%	52.75%	52.5%	53.75%

Table 4.10: Recognition rates with 100 objects using $k = 3$ tuples and various occlusions according to different quadrants (Q_1,Q_2,Q_3 and Q_4) and combinations of two quadrants.

section of the building missing). This test image was also taken at a different time of day and under different weather conditions. Figure (4.13.d) shows the graphical likelihood map thresholded and multiplied by the original test image in order to visualize the detected region (where the model likelihood is very high). We can see that our hybrid color/appearance model (which is quite general in formulation) is found to be satisfactory for such satellite/aerial imagery.

4.3.5 Tuple Selection

Image from figure (4.13) depicts a pentagon building that is composed of several keypoints. In this particular case, the object is composed of nearly 250 keypoints. All possible $k = 3$ tuples that we can generate from 250 keypoints is extremely huge (like $250 \times 249 \times 248 = 15438000$ if redundancy is present in tuples). We can not use a mixture of Gaussians or other techniques to learn from this huge dataset. We should say that the experiment of figure (4.13) has been done using only a subset of the possible $k = 3$ tuples. In case of using a technique to model this space, it would be very expensive in terms of computational costs. Previous experiments have been done taking in mind that not all the generated tuples from one object are really necessary. Our idea is to select a subset of the generated tuples in order to find a representative set of tuple candidates and learn a Gaussian mixture model to obtain a good representation of the natural object. In the particular case of dealing with a natural object as the pentagon object building and in order to be able to manage with partial and natural occlusions, tuples must be carefully selected. Thus, we defined a tuple radial threshold (R_{thr}) and we only consider those tuples that the distance between each keypoint of the tuple with respect to the middle point of tuple (the middle point of a tuple is obtained considering the three keypoints of the tuple) is lower than R_{thr}, a certain threshold previously fixed. This idea is represented in figure (4.14) where we can see three local features ($\mathbf{x}^1$, $\mathbf{x}^2$ and $\mathbf{x}^3$) and the middle point of the tuple. The middle point of a tuple is obtained considering the spatial location of all three keypoints of the tuple. When all the three distances (R_1, R_2 and R_3) between each local feature and the middle point are lower than R_{thr}, the tuple is considered for training. As can be seen, this idea comes out in order to consider

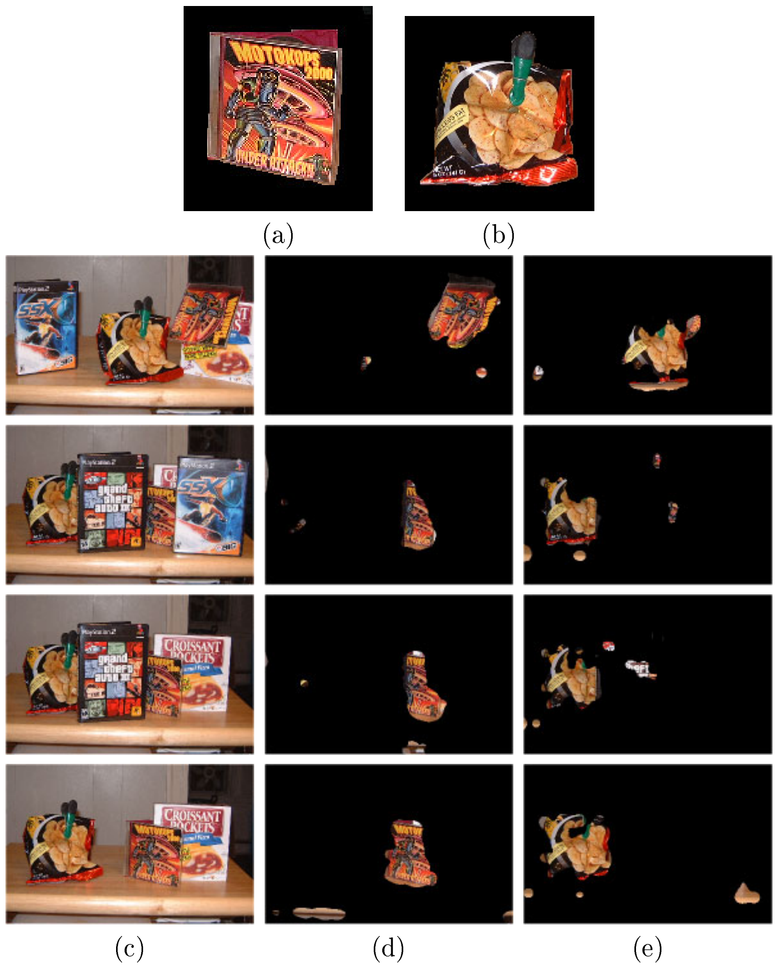


Figure 4.12: Two objects (a) and (b) with similar colors but differing in shape used to train our models. First column (c) contains 4 testing images where the two learned objects are present under different occlusions and poses. Columns (d) and (e) show the detection maps for objects (a) and (b), respectively.

tuples with close keypoints and mantain the object structure.

Now, in figure (4.15) we present different likelihood maps obtained from the pentagon object building when considering different radial thresholds for the tuple selection. Our pentagon object used for learning is about 120×120 pixels and, as seen in figure (4.13), it consists of several structured parts but repeated along the object. After obtaining all the pentagon keypoints, we have considered a set of learning tuples limited to a radial threshold of 25, 30, 35, 40 and 45 pixels in order to mantain the structure of the object. Results are shown in figure (4.15) where we can appreciate that low radial thresholds produce bad detection maps and high radial thresholds produce good (or acceptable) detection maps. However, the number of training tu-

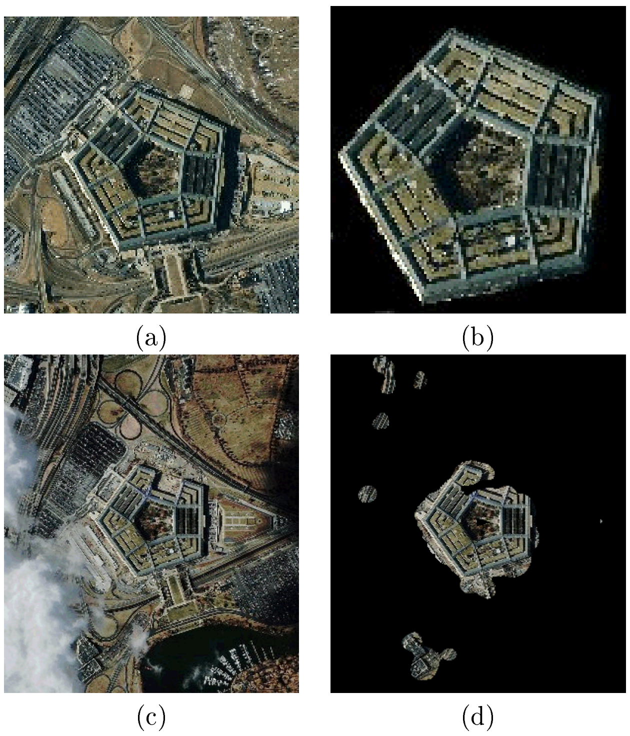


Figure 4.13: (a) Satellite image of the US Pentagon building (prior to 9/11/01). (b) extracted building region used for learning. (c) a new test image of the same region taken after 9/11/01 under different natural conditions and with the damaged portion of the building missing (removed after site cleanup). (d) the highest probability target area of the test image given our local appearance model of (b). (Note: All images have been rescaled for display purposes.)

ples when we use high radial thresholds are really huge and our adaptive Gaussian mixture model needs a considerable amount of computational resources. Since our method is tested with an object with a missing part, detection maps obtained are understandable in the sense that a part of the pentagon building might not be recognized properly. When using a $R_{thr} = 25$ pixels, results are not acceptable since the object is not correctly detected and a lot of external regions are considered as the pentagon. But, when using $R_{thr} = 40$ pixels, pentagon is correctly detected and only a few external regions are considered as being part of the object. Note that the best detected pentagon shown in figure (4.15) using $R_{thr} = 40$ pixels is shown in figure (4.13).

Finally, we present a last experiment where we also show how our technique can be successfully applied to detect objects in semi-cluttered scenes. Figure (4.16) shows a complex scene where a toy object appears three times and each instance of the object is captured under a different pose. As with the pentagon object, the goal is to learn the toy object from one instance and detect the other instances. The toy

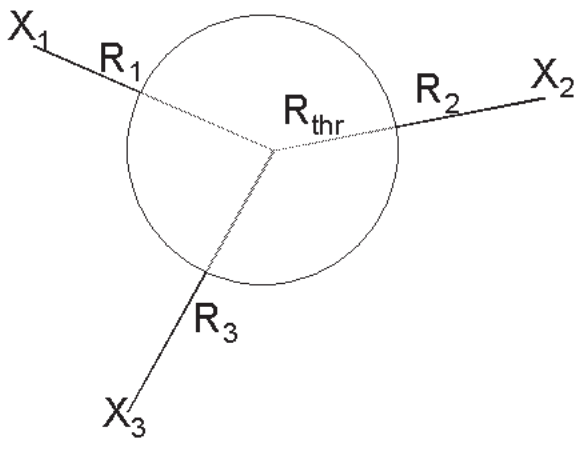
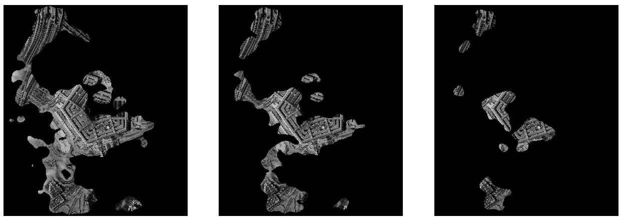


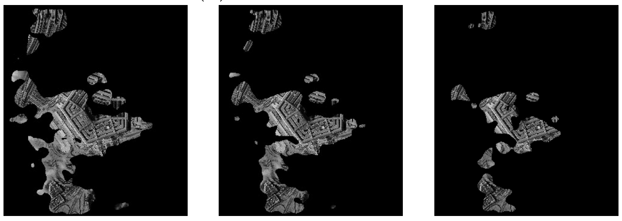
Figure 4.14: Given a tuple $(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3)$, we obtain the middle point using the spatial location of each keypoint. When the three distances between each keypoint and the middle point (R_1, R_2 and R_3) are lower than a predefined radial value (R_{thr}), this tuple is considered for learning.

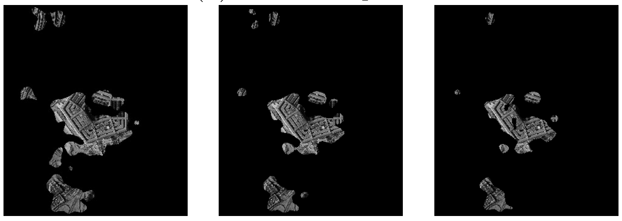
instance used for learning is shown in figure (4.16.a). As it can be seen, this object instance has been extracted from the complex scene and we have manually removed all the background.

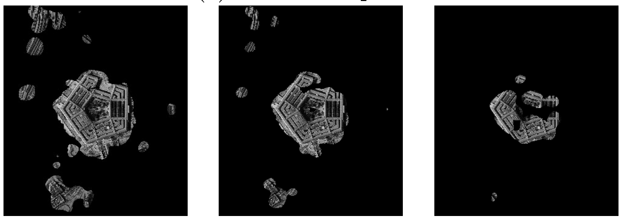
The toy object contains 99 keypoints and depending on the radial area (R_{thr}) of influence, we can work with large number of $k = 3$ tuples. A first detection test was performed in order to obtain the likelihood maps of the complex scene when trying to detect the learned object. Figure (4.17) shows the likelihood maps obtained using 3 different models. As seen in figure (4.17.a), appearance model is not a good candidate for this specific task because there are a lot of regions that do not belong to the learned object that are detected as belonging to the original toy object. Color model seems to obtain good results but only detecting the same instance of the object that has been previously learned and it seems that it is not able to generalize to other object poses. The combined appearance and color model demonstrates that is able to detect the three instances of the learned object but, as seen in figure (4.17.c), the inferior part of the object is not perfectly detected. This behaviour is completely justified by the fact that we have learned our object against a black background and the inferior part of the object is small and homogeneous meaning that the number of keypoints of this zone is relatively small and not really meaningful. We have experimentally demonstrated that the combined appearance and color model is the best for this specific problem. So that, we take this particular framework to show different likelihood maps of this scene when we modify the radial tuple threshold (R_{thr}) that is used to control the number of $k = 3$ tuples. Figure (4.18) shows different likelihood maps that have been generated according to different values of the radial tuple threshold (R_{thr}).

It is clear that when the radial tuple threshold (R_{thr}) is increased, we are taking into account new relations between keypoints in an object. So that, it is expected that $R_{thr} = \infty$ should provide the best results in terms of object recognition and detection.


(a) $R_{thr} = 25$ pixels


(b) $R_{thr} = 30$ pixels


(c) $R_{thr} = 35$ pixels


(d) $R_{thr} = 40$ pixels

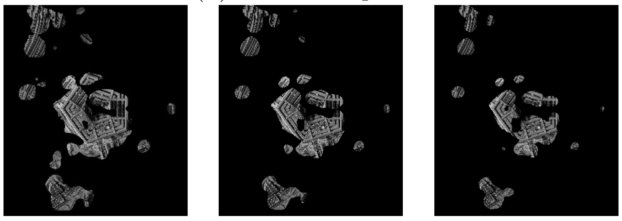
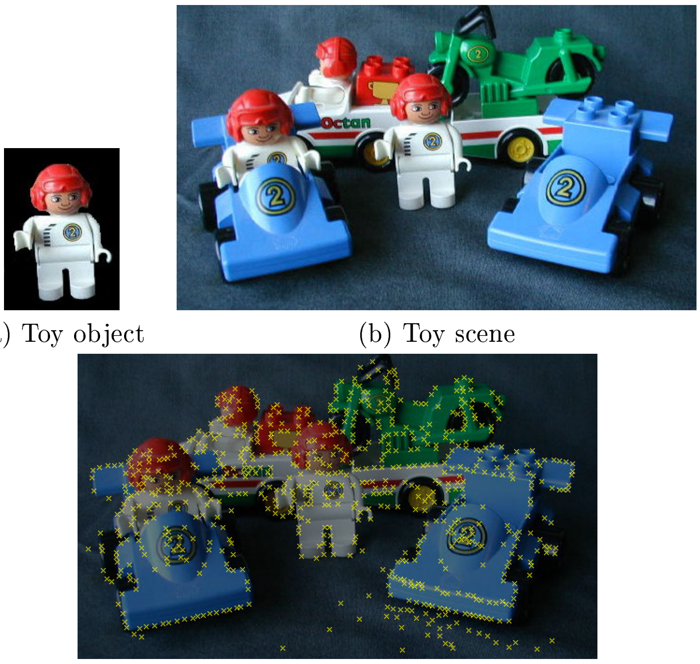

(e) $R_{thr} = 45$ pixels

Figure 4.15: Detection maps corresponding to different radial thresholds (from $R_{thr} = 25$ to $R_{thr} = 45$ pixels).

However, if we assume the whole set of tuples our problem becomes computationally expensive. But, one of the advantages of using only a subset of the total number



(a) Toy object (b) Toy scene

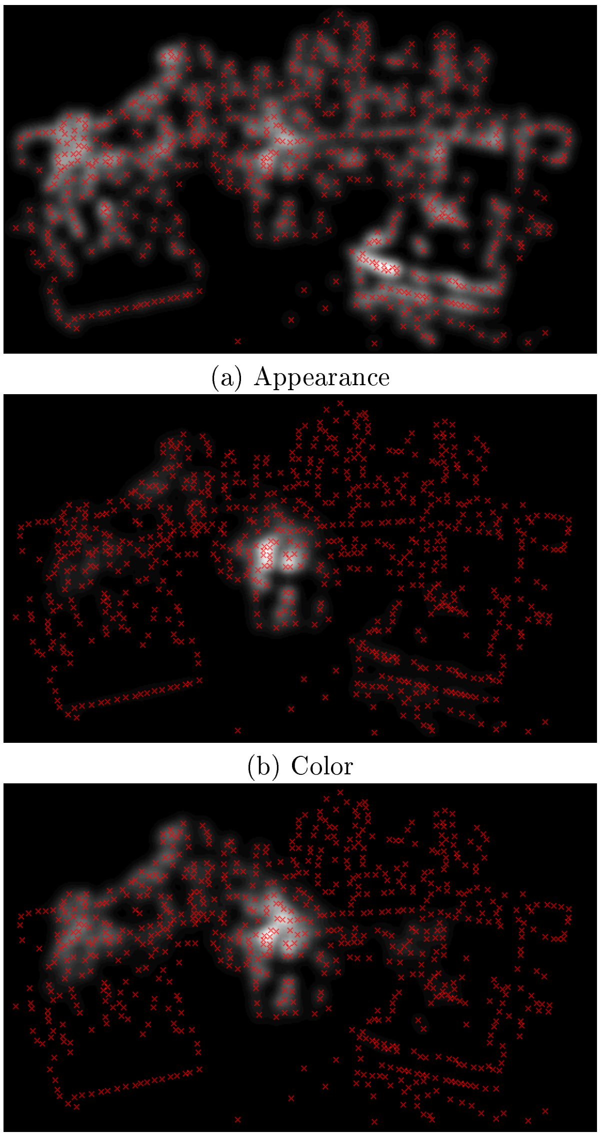
(c) Toy scene with keypoints

Figure 4.16: Toy example used to test our technique in cluttered scenes. Having a complex scene as shown in (b), we have extracted a toy object as shown in (a). Note that this toy object appears in (b) three times. One instance is the same instance as in (a), another instance of the object appears occluded and there is another instance where the object has a completely different pose. Interesting points of this scene are reflected in (c). Size of toy object is 100×140 pixels.

of $k = 3$ tuples is that our method is indirectly robust to partial occlusions. If we consider the whole set of tuples, we will have a holistic representation of the local behaviours of all keypoints of an object. So that, in this case and having an image with occlusions, recognition and detection will be difficult to manage. In order to be robust under occlusions, tuples do not have to be randomly selected. They have to be selected in order to preserve the spatial compactness of an object, so that, tuples with close keypoints should be used for object recognition and detection task.

4.3.6 Modeling Higher-Order Dependencies

We next apply our object recognition scheme in a totally different context in order to demonstrate (1) how to integrate mutiple instances of objects into a single model, (2) that our scheme can be used with other kinds of features and data representations and (3) that increasing the tuple order does in fact lead to improved performance. In this experiment, we chose the MNIST [76] (see appendix C.4) digit database because it contains a huge number of training and testing samples (60,000 training samples and 10,000 testing samples), so we can statistically verify that incrementing the order



(a) Appearance

(b) Color

(c) Appearance + Color

Figure 4.17: Likelihood maps obtained when we try to search for the object defined in figure (4.16.a) when using a radial tuple threshold of $R_{thr} = 25$ pixels. As seen, the color and appearance model can detect all the three instances of the learned object under different object poses.

of our models will lead to better recognition rates. We must note that our scheme is not especially adapted to work with the MNIST database (which for one thing, is not even in color and has little in the way of appearance texture) rather it is a general

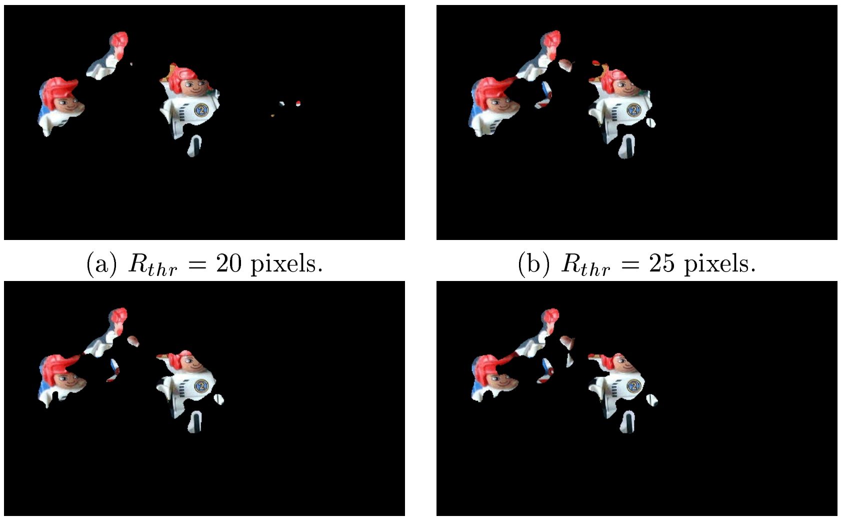

(a) $R_{thr} = 20$ pixels. (b) $R_{thr} = 25$ pixels.
(c) $R_{thr} = 30$ pixels. (d) $R_{thr} = 35$ pixels.

Figure 4.18: Different likelihood maps that have been thresholded in order to visually perceive what parts of the scene are detected. These likelihood maps have been obtained according to the radial tuple threshold (R_{thr}) considered in each particular case.

technique for use in complex and cluttered scenes with the presence of occlusions. Our main goal here is to explore how increasing tuple order affects to the recognition rates using a well-known and large database.

In particular, features were extracted from hand-written MNIST digits using the same technique as in [12] where they obtain a set of shape histograms for each digit. In our case, each digit is represented by a set of 75 points sampled from the shape contour (75 pixel locations sampled from the output of the Canny dectector). Having 75 pixel locations, we have represented each location using a shape histogram (exactly the same as in [12]), so that each digit is represented by 75 shape histograms of 60 dimensions. In order to find the "right" ICA dimension to reduce our feature vectors, we did a k-NN (with k=5) based classification using the original shape histograms taking a reduced set of training and testing samples (200 training samples per each digit and the first $5,000$ testing samples) using the χ^2 test statistic (as in [12]) as a distance metric. Also, a k-NN (with k=5) based classification was done using the ICA projected feature vectors between $m = 5$ to $m = 50$ ICA dimensions with the same training and testing sets as before using the L_1 norm as a metric distance in order to evaluate which is the ICA dimension that preserves the same recognition rates of the original space. The dimension found by the experiments to be the most suitable one to be used with our ICA scheme was 25, which was used thereafter.

We have tested two different approaches: (1) learn an adaptive mixture model per each training instance and (2) learn an adaptive mixture model per each digit class. Our factored $k = 2$ and $k = 3$ high-order models generate a huge number of tuples. In this particular case, when using $k = 2$ tuples, we generate an order of

$5,000$ tuples per each digit and when using $k = 3$ tuples, $100,000$ possible tuples are generated. We have not tested higher order dependencies because the number of possible tuples is really huge. Thus, having a huge number of tuples, we had to choose a reduced number of them in order to train our factored models in a reasonable period of time. In the previous section, we selected tuples composed of neighbor keypoints to build $k = 3$ tuples. Here, we have tested three different approaches: (1) a random selection of tuples, (2) tuples composed of neighbor keypoints and (3) tuples composed of distant keypoints. We have selected tuples composed of distant keypoints because MNIST digits are homogeneous representations and the information extracted from neighbor keypoints is almost the same. Thus, two neighbor keypoints contain almost the same local information. This is due because digits do not contain changes in texture or color and the neighborhood of two close keypoints does not change significantly. It is clear that since digits do not contain natural occlusions, tuples composed of distant keypoints can be used. So that, shape histograms extracted from two neighbor keypoints do not contain relevant changes as the shape histograms extracted from two distant keypoints.

Our tuple random selection method consists of randomly selecting $1,000$ $k = 2$ tuples and $5,000$ $k = 3$ tuples to learn our adaptive Gaussian mixture models. When creating tuples composed of neighbor keypoints, we take the $1,000$ $k = 2$ tuples and $5,000$ $k = 3$ tuples with the closest keypoints. Finally, when selecting tuples with distant keypoints, we take the $1,000$ $k = 2$ tuples and $5,000$ $k = 3$ tuples with the most distant keypoints. For our experimental tests, we used 500 training samples per each digit ($5,000$ in total) and all the testing MNIST set ($10,000$ digits). Experimental results are shown in table (4.11) where we can clearly see that incrementing the order of our models leads to an improvement in the recognition rates. Interestingly enough, we note also that there seems to be little difference between the two different approaches of handling multiple training instances: using one model/instance *vs.* one model/class.

k tuples		1 Model / Instance	1 Model / Class
$k = 1$ tuples		74.23%	71.85%
$k = 2$ tuples	Random	83.14%	82.03%
	Near	78.36%	75.47%
	Distant	85.09%	83.71%
$k = 3$ tuples	Random	91.57%	91.13%
	Near	88.67%	87.92%
	Distant	93.03%	91.85%

Table 4.11: Experiments done using 500 digits per each class as training and $10,000$ testing digits. First column of results indicates that each training digit instance is represented using one joint distribution (a total of $5,000$ training models) and second column indicates that each class is represented using only one model (10 training models in total). Recognition rates are represented according to the tuple order and tuple selection technique used.

Each training instance can be modeled using one joint distribution. This is what is reflected in the first column of table (4.11). If we take all the tuples of a given class, i.e. tuples of digit 2, and we create a model using all this set of tuples, we are considering a model per class. This is what is reflected in the second column of table (4.11). As seen in table (4.11) it is better to use a model for each training digit instance instead of using a model for the whole set of tuples of a given class. But there is not a big difference in terms of recognition rates. Table (4.11) also shows what is the effect of using different keypoints to create candidate tuples for learning. It seems clear that for this particular experiment using digits, the selection of tuples based on the neighbor keypoints is not very appropiated. We believe that the reason of this behaviour is due to the fact that the shape contexts between two neighbor keypoints is nearly the same and they do not contain relevant information. In contrast to this, if we take into account one tuple built from three distant keypoints, shape contexts are totally different and they can help to build a robust classifier. Thus, this is the reason that recognition rates based on tuples built from distant keypoints are higher than all the other cases.

Using the nearest neighbor classifier (k-NN with k=3) in the original space of shape histograms with the χ^2 test statistic, we obtain a recognition rate of 75.87%. We do not use any kind of point matching technique between our features as in [12] and it should be obvious that our method is not well-suited to the MNIST database (that is not the point here) but we do notice the improvement of our factored distribution models from $k = 1$ to $k = 3$. We should emphasize that even though we do not achieve the best reported recognition rates for the MNIST, our factored models with $k = 3$ are not only significantly better than $k = 1$ but also better than using k-NN in the original space of shape histograms (a recognition rate of 75.87%).

With this last experiment we can make the following observations: (1) Our technique can be extended to different data representations but in doing so, because it is a general technique, we can not expect it to obtain the best recognition rates, (2) by incrementing the order of our factored models, recognition rates can almost certainly be expected to improve, (3) our technique might be more suited to complex and cluttered scenes than to recognition of objects in closed object databases such as MNIST or COIL-100.

4.4 Conclusions

The contribution in this chapter is related with the previous chapter in the sense we present a technique which works with local features. However, spatial arrangement of local features is taken into account as well as the local information of each local feature. In this sense, a novel probabilistic modeling scheme was proposed based on factorization of high-dimensional distributions of local image features using Independent Component Analysis (ICA). We define the concept of k-tuple in order to represent the local appearance of an object at k different keypoints.

This chapter has been presented as an evolution of several experimental tests in order to find an optimal configuration of the method. We started modeling our joint distributions of image features using histogram representations. We realized that our joint distributions of k tuples form a complex space which is difficult to be modeled using histogram representations. Then, a parametric model is introduced to improve initial histogram representation. So that, our factored distributions were modeled using Gaussian mixture models based on the Minimum Description Length optimality criterion to fit our data, obtaining satisfactory results.

Our framework was tested using local appearance and local color information as well as using geometry information between local image features of tuples. An hybrid classifier based on all these local image features achieves the best recognition results. COIL-100 object database was tested with and without occlusions. And once we tuned the internal parameters of the model using the well-known COIL-100 object database, we did more experiments with complex and cluttered scenes in order to demonstrate that this technique is adapted to object detection and localization tasks in natural environments. We experimented with real scenes containing objects of different shapes and colors and detection results are satisfactory. Also, we experimented with an aerial image of the pentagon object where different natural factors are considered: (i) test image was taken under a different time of day, (ii) test image was taken under different weather conditions, (iii) the pentagon object has a missing portion due to the September 11 terrorist attack. Even all these natural factors, our method is able to detect the pentagon building. Finally, an image containing several instances of a toy object is tested to evaluate its robustness to several pose configurations. As a result of these experimental tests under several natural environments, our method performs very good in object detection.

Our method is based on k-tuples reflecting high-order dependencies between k points in an object. However, the number of tuples can be extremely huge, so that, we propose a first attempt to select the most important tuples. A final experiment with the MNIST digit database was performed in order to evaluate different tuple selection methods. We tried (i) a random selection of tuples, (ii) tuples composed of neighbor keypoints and (iii) tuples composed of distant keypoints. Objects composed of textured regions as objects from COIL100, pentagon object or toy objects, extracted tuples should be composed of neighbor keypoints in order to mantain the object structure. However, objects that do not suffer occlusions and do not contain changes in texture (as MNIST digits) are better represented using tuples with distant keypoints. Since k tuples are composed of k keypoints, if these k keypoints are selected from the neighborhood and no relevant changes are manifested between these neighbors points, the joint distribution is not discriminant. In this experiment, we have joined all the tuples of a given class into a single model. Then, results are compared with joint distributions which use a model per each object instance. A single model per object instance is the best choice in front of using a model per class. This last experiment also shows a statistical improvement of increasing the high-order dependencies of our factored distributions and demonstrates that our technique can be adapted to a wide variety of computer vision problems.