Combining Structural Descriptions and Image-based Representations for Image, Object, and Scene Recognition*

Nicolas Do Huu†, Williams Paquier and Raja Chatila
LAAS-CNRS
7, avenue du Colonel Roche 31077 Toulouse Cedex 4 - France
nicolas.dohuu@laas.fr, william.paquier@laas.fr, raja.chatila@laas.fr

Abstract

Object and scene learning and recognition is a major issue in computer vision, in robotics and in cognitive sciences. This paper presents the principles and results of an approach which extracts structured view-based representations for multi-purpose recognition. The structures are hierarchical and distributed and provide for generalization and categorization. A tracking process enables to bind views over time and to link consecutive views. Scenes can also be recognized using objects as components. Illustrative results are presented.

1 Introduction

Object and scene learning and recognition is a major issue in computer vision, in robotics, in neuroscience, and in cognitive sciences as well. And one of the main questions to this respect is how to extract knowledge from 2D patterns of light in the camera or the retina.

For the recognition of objects, two models were proposed in the literature. In the 'image-based' or 'view-based' model, an object is represented as a collection of view-specific local features [Poggio and Edelman, 1990; Ullman, 1998; Tarr and Bülthoff, 1998; Riesenhuber and Poggio, 1999]. Representations are organized in trees in which a set of view-tuned units constitutes the weighted inputs to a higher level object-invariant unit. Each unit measures the similarity between the input image and its stored view, and the higher level unit computes the weigthed sum from its incoming connections. If the resulting value reaches a given threshold, then the learned object is recognized. Riesenhuber's HMAX model of object recognition in the ventral visual stream of primates also proposes a similar grouping method where higher level cells compute the maximum response of view-tuned cells [Riesenhuber and Poggio, 1999].

The second model is based on structural descriptions. One of the most important approaches is the "Recognition-By-Components" (RBC) model [Biederman, 1987] in which each object is represented as a collection of volume parts. Thus, there is no need of multiple view-tuned representations since 3D models can be virtually rotated and compared to the input image. Moreover, the use of such a model can make the recognition invariant to illumination and color.

We can identify certain properties of the modeling process for efficient learning and recognition. Firstly, the training and thus the construction of new representations must be sufficiently fast. Lack of speed is the principal cutoff of models based on structural descriptions because building 3D models from images remains a non trivial task. Secondly, an efficient modeling process must organize knowledge in a structured way. A first means of organizing the data is to build categorical representations. Categorical structures make it possible to obtain capacities of class generalization at low cost. Indeed, to effectively exploit the extracted data, those must be describable at various levels of specificity: a bottle can be described as a container, a plastic object for recycling, etc. Such a structure can also accelerate recognition that deals with large collections of stored objects by reducing the number of candidate object models. Another mode to structure meanings is the decomposition of a representation as the set of its parts. Thanks to structural descriptions, RBC offers more robustness, in particular with respect to noise and occlusion. Moreover, it is interesting to share components among several structural descriptions to save memory: one could use the representation of a wheel to describe either a car, a truck or a bike. Thirdly, a learning system for knowledge extraction and structuring must allow the addition of new representations at non-prohibitive memory cost and without explosion of complexity. This open-endedness property can be approached by using the two kinds of data structures referred to above, in which a representation can be factorized in categories or shared as components.

While considering object recognition one must mention important results achieved by template-based approaches for object classification and recognition [LeCun *et al.*, 2004]. Nearest Neighbor methods, Support Vector Machines and Convolutional Networks provide efficient solutions but the needs of structured knowledge, reusability, incremental and autonomous learning are several points addressed here which are not, to the best of the authors' knowledge, well dealt with by the pre-cited methods.

This paper presents an efficient approach [Paquier, 2004]

*The work described in this paper was partially conducted within the EU Integrated Project COGNIRON ("The Cognitive Companion") funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

†supported by the European Social Fund.

(which is partially implemented) for building structured representations without using 3D primitives and exhibiting the properties mentioned above. The paper is organized as follows. Section 2 presents the model we choose to both extract and organize representations. Each essential property is presented and illustrated by an example produced by the implemented system. Section 3 introduces the view-binding algorithm and the incremental building of object-invariant detectors. The use of objects as landmarks for buidling structured representation of scenes is presented in section 4. We conclude in section 5.

2 Architecture for View-based Recognition

We will first describe the *map* structure which constitutes the basic building block of our model, and discuss its inherent shift-invariant property. In the learning process, the maps will specialize in certain features, and we will present the way we associate them in order to prevent redundancy in this specialization and structure extracted features.

2.1 *Maps* and Inter-*maps* Connectivity

The *Map*: a Set of Local Classifiers

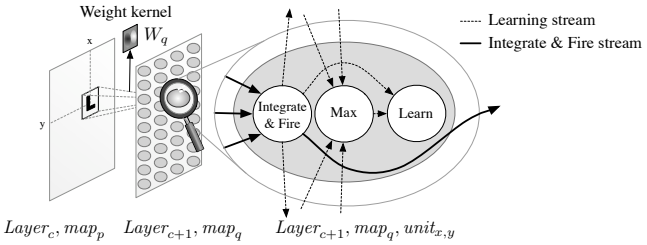

Figure 1: A *map* is a collection of *units* sharing the same set of weights (*kernel*). Each unit is composed as a three stage calculation pipeline. It receives signals incoming from previous layers for integration and lateral signals from *maps* situated on the same layer for competition.

In order to satisfy real-time constraints, the learning and recognition system must be able to update its representations at a frequency that is compatible with the modifications of the environment, and process images at a rather high frequency (e.g., 15 fps). With this end in view we chose to implement an integrate-and-fire model in local classifiers built as calculation pipelines (figure 1).

We call a *map* a collection of local classifiers called *units*, organized in a retinotopic[1] way. Each *map* is associated with one or more afferent *maps* which are the locations of its *units*' input domains (or their receptive fields). Receptive fields of each *unit* are static and each *unit* is thus associated to a set of afferent *units* in each afferent *map*. We denote Ω_i the set of afferent *units* of a given *unit* U_i.

At time t and for a given unit U_i, information flows from its receptive fields through a set of learning weights $W_i(t)$. Learning weights are organized in *kernels*, and there are as many weight kernels as afferent *maps*. All the *units* of a *map* are sharing the same set of weights, thus they can detect and learn a pattern which is at different positions in the input *maps*. As shown in figure 1, the coordinates of a unit in a given map and its receptive field in the afferent *map* are the same. Therefore the position of an active unit corresponds to the position of the detected pattern, which provides for a shift-invariance property.

This type of mapping has been previously introduced by Fukushima's *Neocognitron* and was successfully applied to handwritten digit recognition [Fukushima, 2003]. Figure 2 illustrates this intrinsic *map* feature. In this experiment, we produce an image containing a set of four randomly distributed letters (k, p, s, u). After less than a minute, the system has learned autonomously its own distributed representation of the input image. Each resulting *map* can extract its learned feature at any location in the input image. To obtain this result we must also provide our system with an inter-*maps* communication channel, so that *maps* don't learn the same feature several times. This procedure is called "local competition" and is introduced next.

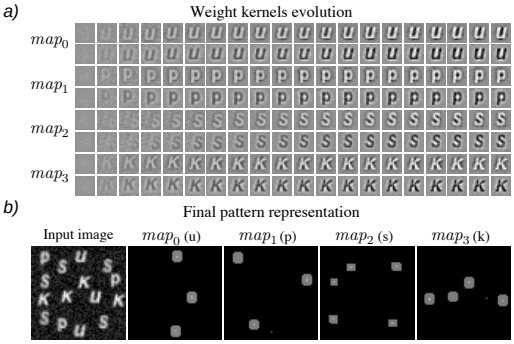

Figure 2: Letter extraction from noisy image input (20% random noise, 20% scale and angle variation, 70 steps, 2 images per second). a) weight kernels evolution across time. b) input image and associated *maps* activations.

Preventing Redundancy with Competition

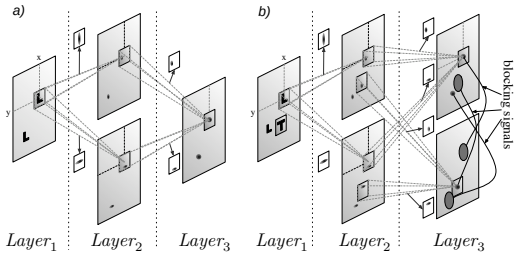

Figure 3: Hierarchical *maps* connectivity and local competition. a) The letter "L" is detected as a relative positioning of horizontal and vertical bars. b) input image containing the letters "L" and "T": inter-*maps* competition with local inhibition leads to distinct specialisations.

One previously mentioned requirement concerns the struc-

[1] A retinotopic mapping means that stimuli that are adjacent to each other in the visual world are processed by adjacent sets of *units*.

tural decomposition into several components. To achieve such a decomposition, different maps observing the same inputs must "decide" to specialize into distinct component detectors. The undergoing process is illustrated in figures 3.a and 3.b. In this example, we want to train two *maps* to detect the letters "L" and "T" in the input image. A first step of extraction is composed of a layer of two *maps* which respectively detect vertical and horizontal bars in their inputs. When the letter "L" is present in the image (figure 3.a), the pattern is decomposed as the positions of these two local orientations. The relative positioning of the two patterns is then detectable and can be learned by a dedicated *map* on the next layer ($Layer_3$).

From this point, if we add another letter ("T") to the input image and a second learning *map* to the third layer to learn it (figure 3.b), then we must prevent this new *map* from learning the "L" pattern one more time. This differentiation is achieved by using a local competition in which we compare *map* detection values and choose the best fitted (See section 2.2 for the computation of the detection value).

Right after calculating its detection value, every unit of each *map* broadcasts it to all other *units* on *maps* of the same layer and at the same coordinates. Thus, every *unit* whose coordinates correspond to the position of the letter receives incoming values at the 'Max' stage of its pipeline architecture (see figure 1). The 'Max' stage, as its name indicates, computes the maximum incoming value and sends it to its own 'Learn' stage. The learning process is then able to compare the local *unit* activation incoming from the 'Integrate & Fire' stage to distant activations. By allowing learning only to the best fitted *unit*, we ensure that no *map* could learn one pattern if another is already specialized to detect it. We illustrate this inter-*maps* competition on Figure 3.b with the dark discs which represent the blocking signal.

Connecting *Maps* to Build Distributed and Hierarchical Representations

We adopted a layered hierarchical architecture for several reasons. Firstly we need this architecture for *categorization*. As our units achieve linear separation in their input set, complex features cannot be extracted with a single layer. This is a known limitation of the single layer perceptron that cannot simulate an exclusive disjunction (logical XOR). The second reason is *reusability*. A huge number of extracted features are encountered in different shapes: oriented segments, arcs of circles or color blobs are building blocks for more complex images. Since extracted information is shared in the network, we avoid redundancy and the resulting computational overhead. Following the same idea, similar meanings should be encoded using shared sets of units. For example, the internal representations of a truck and of a car should intersect. This intersection representing the shared meaning could contain the internal representation of the four wheels, the steering wheel, etc. Thanks to this distributed representation architecture, it is easier to pool similar objects into cross categories (e.g., wheeled vehicules).

This capability of building hierarchical and shared representations is shown in figure 4. In this experiment we trained a network to recognize faces using a set of ten different images of each of 40 distinct subjects[2]. In the same way as in the previous example of letter extraction, $Layer_2$ learns to detect the position of each eye in one *map* and the position of the mouth in the other one, sharing the same segments extraction in the previous layer ($Layer_1$). In $Layer_3$, we train another *map* which receives as inputs the detection values of the mouth and eyes specialized *maps*. As a result, this last *map* learns the representation of the face with relative positioning of eyes and mouth. We also see in this example that the resulting face recognition is robust to high variations in subject posture and morphology. Nevertheless, this kind of recognition only applies to images with limited face orientations. In section 3, we will propose additional methods to recognize 3D objects based on the pooling of overlapping views.

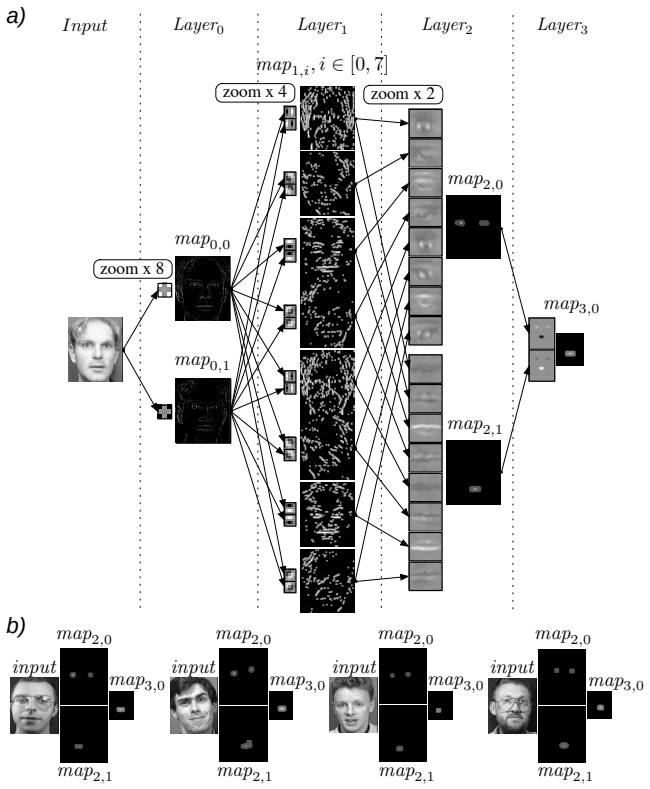

Figure 4: An example of a hierarchical representation of faces. a) network connectivity showing weight kernels and *maps* activities. $Layer_0$, $Layer_1$, $Layer_2$, $Layer_3$ respectively extract contrasts, oriented segments, eyes and mouth, and face. b) recognition robustness for different subjects and variable postures.

2.2 Integration, Firing and Learning

Now that we have presented the global mechanisms involving the computing *units*, we present the internal workings of the system more precisely and the computation of the detection values mentioned in section 2.1.

[2]We used the faces database of the AT&T Laboratories, Cambridge http://www.uk.research.att.com/facedatabase.html

Integration
Given a *unit* U_i, we denote $\beta_i(t) \in [0,1]$ its calculated output burst at time t. The burst is the detection value mentioned before. Let $w_{ij}(t) \in W_i(t)$ be the learning weight associated to the connection between *units* U_i and U_j, and $\Omega i^+(t)$ a subset of afferent Ω_i defined as:

$$W_i(t) = \{w_{ij}(t), U_j \in \Omega_i\} \quad (1)$$

$$\Omega_i^+(t) = \{U_j \in \Omega_i, \beta_j(t) > 0\} \quad (2)$$

We also define three weight sums as follows :

$$S_i^\star(t) = \sum_{w_{ij}(t) \in W_i(t)} \mid w_{ij}(t) \mid \quad (3)$$

$$S_i^{\beta^+}(t) = \sum_{U_j \in \Omega_i^+(t)} w_{ij}(t) \quad (4)$$

$$S_i^+(t) = \sum_{w_{ij}(t) \in W_i^+(t)} w_{ij}(t) \quad (5)$$

where $W_i^+(t) = \{w_{ij}(t), U_j \in \Omega_i \; and \; w_{ij}(t) > 0\}$.
Then the integrated potential P_i at time $t+1$, which expresses a level of similarity between the learnt pattern and the inputs, is given by:

$$P_i(t+1) = \frac{\alpha_P P_i(t) + (1-\alpha_P) S_i^{\beta^+}(t)}{S_i^+(t)} \quad (6)$$

Where $\alpha_P \in [0,1]$ is a fixed potential leak.

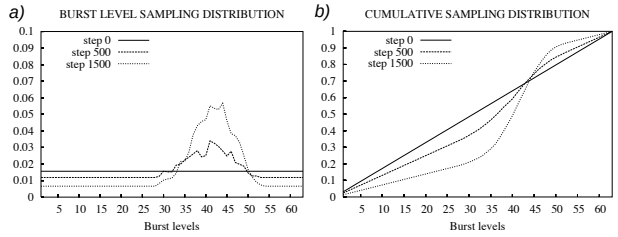

Figure 5: Burst sampling distribution and cumulative distribution evolution. a) burst sampling distributions computed at steps 0, 500 and 1500. Burst values are discretized within 64 levels. b) corresponding cumulative distribution used as a transfer function for integration.

Firing
Output bursts β_i generated by a unit U_i are produced both for integration processes situated on the the next layer and for its own "Learn" stage (see figure 1). In the first case, burst levels are not taken into account since all positive bursts participate to integration (see equation 2). In the second case, we previously explained that we need to compare *units* activation in order to find the best fitted *map* in a competition process. The burst level is thus computed as follows:

$$\beta_i(t) = \begin{cases} 0 & \text{if } P_i(t) < T_i, \\ F_i(t, P_i(t)) & \text{otherwise.} \end{cases} \quad (7)$$

Where T_i is a fixed threshold and F_i is an adaptive transfer function whose variations across time are illustrated by figure 5.b. The underlying idea of this function is that, as units are learning, their output function must converge to a binary-like response corresponding to a more strict linear separation. At each step we compute a sample distribution of the input burst by discretizing the output burst value in 64 levels. The transfer function F_i is in fact the associative cumulative distribution at time t:

$$F_i(t,x) = \sum_{0<b\leq x} f_i(t,b) \quad (8)$$

Where function f_i is a sampling distribution of burst levels updated at each step (figure 5.a). We thus obtain a sigmoidal transfer function which permits a binary-like classification behavior and a more effective competition process. Until now, object detection is internally represented as the activation of a limited set of computing *units*. Due to our connectivity scheme, recognitions are indeed only reflected at positions located near the centre of the learned pattern. However patterns presence are more than just their center since the whole surface they cover is relevant. So we increase the pool of activated units to cover a wider surface. As a first approach, we chose a simple disc shape whose surface is given by $S_i^\star(t)$.

Learning
We use in our model a hebbian-like learning rule in which *units* tend to learn patterns responsible for their activation. In other words, a learning process occurs when one *unit* has both fired and won the competition in its layer. Moreover, we compute a stochastic standard deviation σ_{ij} for each weight w_{ij} which is computed and used as coefficient in the learning calculation as follows:

$$\mu_{ij}(t+1) = (1-\alpha_W)\mu_{ij}(t) + \alpha_W \mid \gamma_{ij}(t) \mid$$
$$\sigma_{ij}(t+1)^2 = (1-\alpha_W)\sigma_{ij}(t) + \alpha_W[\mu_{ij}(t+1) - \gamma_{ij}(t+1)]^2$$
$$w_{ij}(t+1) = (1-\alpha_W)w_{ij}(t) + \alpha_W[1 - 2\sigma_{ij}(t+1)]\gamma_{ij}(t+1)$$

Where μ_{ij} is a stochastic mean and α_W the learning rate. Hence we ensure that weights corresponding to noisy inputs (with high standard deviation) will tend to 0 and then won't take part in the representation. γ_{ij} is a function of input burst of *unit* i which takes its values in $\{-1, 0, 1\}$. *Maps* among the same layer are in competition as shown in figure 3. If at time t, an input burst from *unit* j is received $\gamma_{ij}(t) = 1$, else if a burst is received from any other afferent *unit* then $\gamma_{ij}(t) = -1$. Finally if no burst was present $\gamma_{ij}(t) = 0$.

3 From Views to Objects

In this section, we consider the 3D-object recognition problem. Let us consider a camera rotating around an object. For a given orientation we can train a dedicated *map* and obtain enough robustness to detect this view, say within about a 45 degrees angle interval centered on the learnt prototype. Wider intervals can be obtained with "simple" objects, such as balls or paper cups, whose views are relatively invariant during view-point modification, while turning around them. Our approch is to pool overlapping view detectors. Then, a view-invariant (or object-tuned) detector can be obtained simply by computing the maximum view-tuned *map* response [Riesenhuber and Poggio, 1999]. This will provide for continuous object recognition.

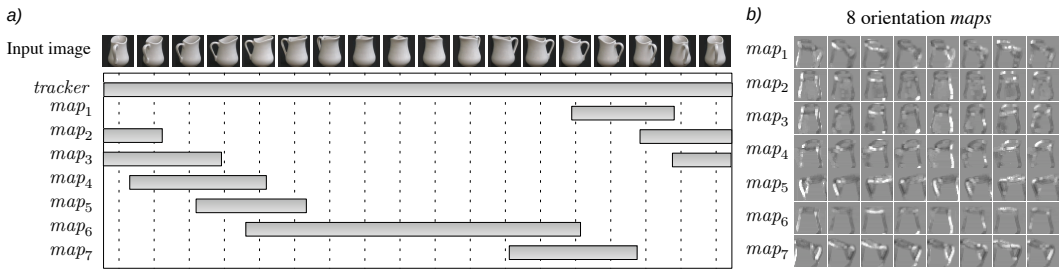

Figure 6: Object invariant detection with view-tuned *maps*. a) *maps* activities overlap to produce continuous detection. b) resulting learning weights. These are composed of 8 kernels for each *map* which correspond to 8 afferent oriented segment detectors situated on previous layer (see figure 4).

Assuming the support of an external module producing the initial object detection (this could be simply a user's selection on the input image, but this module could be based on saliency, motion, etc...), we must resolve several remaining problems. Firstly, the object-tuned *map* must keep track of the object during point of view modifications (temporal binding). Secondly, as we have no initial information about the object's complexity and, therefore cannot anticipate the exact number of required view-tuned *maps*, we must find a method which incrementally adds new *maps* to the network and associate them together.

Map-tracker : Short-term Memory for Temporal Binding

The different views of the same object tend to occur close together in time and space [Tarr and Bülthoff, 1998]. Several authors proposed to take advantage of this property [Stinger and Rolls, 2002; Wallis, 1996]. We particularly want to mention Stinger and Rolls's hypothesis about the functional architecture and the operation of the ventral visual system tested in their model called VisNet . In this model, each activation of a view-invariant neuron is maintained during a short period of time. The pooling is then achieved by using an associative memory to link temporal memory trace and current view detection. Although this method seems biologically plausible, it seems difficult to apply to real-time robotics. As the view-to-object membership is not *a priori* known, there is no criteria to decide whether or not a newly specialized *map* corresponds indeed to the view of an object. If no preselection is applied, the size of the associative memory could exceed computable capacities. So, it is crucial to restrict *map* specialization to relevant views. A solution comes from a tracking approach. In this domain, temporal coherence is also used but in another way. The aim of tracking is not so much extracting knowledge but rather following the position of a collection of pixels. Temporal coherence can here be exploited by considering that persitent informations are preserved between two frames and can thus be extracted. We propose to use the *map* architecture for pixel tracking in order to follow the position of the object and use this information as a first step for an incremental view-tuned *map* creation.

We can obtain the desired tracking capability by increasing the learning rate α_W of a map (see section 2.2), which we will call tracker-*map*, to a value near 1. The tracker-map is trained at the time the first object position is produced by the external module. We call this process "one-shot learning" because of the use of a high learning rate. In the next frame, thanks to the robustness of our model, the tracker-map can detect the object even if it has started to move. According to the hebbian rule, this detection is followed by a learning phase. Then, another learning occurs which almost completly renews the stored prototype due to the high learning rate. From now the process can restart allowing continous detection. Tracker-*maps* can be viewed as short-term memories that never reach stable representations.

Temporal Binding

Now that we can use temporal coherence to track the object position, we must resolve the remaining problem of building incrementally view-specific representations of the tracked object. To reach this goal we granted tracker-*maps* the capability of creating new maps during runtime. As we are in the context of moving objects and/or moving point of views, one-shot learning (learning with high learning rate will also be used to catch object's views during movement. The very specific resulting prototype will be refined as view-tuned maps compute standard learning rates. The binding algorithm develops as follows: if no view-tuned *map* recognizes the object at the position of the tracker-*map*'s detection during a short period of time (we use a 5 frames time-out), which is initially the case, then the tracker creates a new *map* and forces it to learn the unknown view with a one-shot-learning signal. If a view is detected, because a *map* has previously learnt it, then a standard learning process occurs in this *map* and the detection produces a one-shot learning signal for the tracker-*map* in order to focus tracking on this view. This process ensures that long-term memory of a specific view has a higher priority during object detection than the tracker. This algorithm has been used in the experiment of figure 6 in which the image of a rotating object has been used to train a three-layers network. In figure 6.a, we see that the tracker always detects the object, thanks to the feedback from long-term to short-term memory, and that activity of map_6 is about 3 times longer than the others. Indeed, this *map* learned to detect orientations that are rather invariant during rotation. Our model can thus adapt the number of required *maps* depending on the complexity of objects.

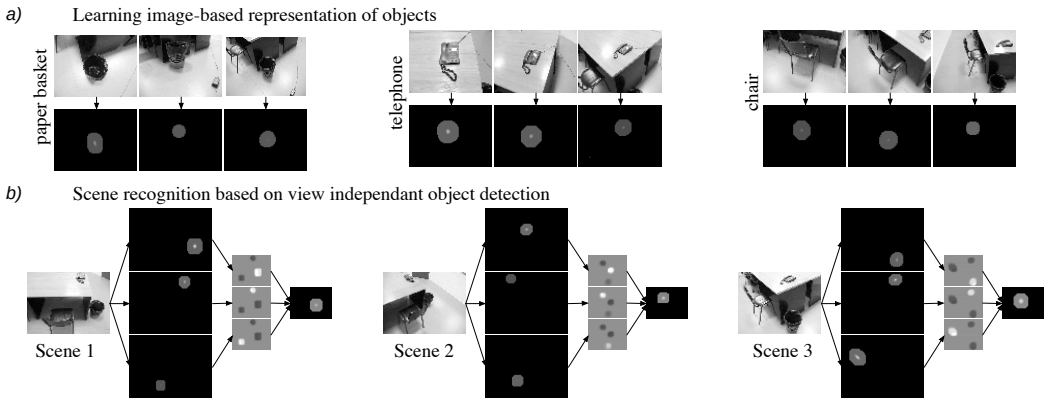

Figure 7: Representing scenes as arrangements of objects in an office environment. a) examples of object-invariant detections from different points of view learn. b) results of scene recognition by relative positioning of object-invariant detections.

4 Objects as landmarks for scene recognition

We present in this section experimental results of our system in the context of scene recognition. We can define a scene as a particular arrangement of objects. We demonstrated previously with face recognition that our architecture is able to construct hierarchical representations of an image. We can use this same technique in order to learn a scene as relative arrangement of objects. For this experiment we observed a standard office environment including 3 basic objects: a paper basket, a phone and a chair. In a first phase, object representations are learnt on the fly using temporal coherence by the means of tracker-*maps* (section 3). The resulting network then detects each object simultaneously in view-specific *maps* and in the corresponding object-invariant tracker-*map*. In a second phase, three higher level *maps* connected to all object-invariant detectors learn representations of the scene from three different points of view. Figure 7.a shows the outputs of the three object-invariant detectors built as presented in section 3 with natural images. Without needing any additional algorithm, our model extracts a view-point specific representation of the scene in a hierarchical structure (figure 7.b). From now, a location-invariant *map* can be trained thanks to tracker-*maps* for scene-invariant *maps* pooling as explained in section 3. These multi-layered architecture allowed by parallel-pipeline calculations is the key property which permits to combine view-based representations and structural descriptions.

5 Conclusion

In this paper we have introduced several methods and algorithms to extract knowledge from visual data. Our model is able to build representations by decomposition of image features into local components structured in a global hierarchy of concepts. Such decompositions are minimal requirements for both sharing representation in order to save memory and computational time, and providing generalization capabilities due to the fact that extracted components can be used as description blocks to recognize new elements. The main advantage of our model is that one does not need any prior knowledge or model to build robust dedicated detectors of simple to complex features in a unified language. Thus, building heterogeneous libraries of meanings becomes feasible. These representations could be then provided as input to a symbolic reasoning level.

References

[Biederman, 1987] I Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[Fukushima, 2003] K. Fukushima. Neocognitron for handwritten digit recognition. *Neurocomputing*, 51:161–180, April 2003.

[LeCun *et al.*, 2004] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR'04*, 2004.

[Paquier, 2004] W. Paquier. *Apprentissage Ouvert de Représentations et de Fonctionnalités en Robotique : Analyse, Modèle et Implémentation*. PhD thesis, 2004.

[Poggio and Edelman, 1990] T. Poggio and S. Edelman. A network that learns to recognize three dimensional objects. *Letters to Nature*, 343:263–266, 1990.

[Riesenhuber and Poggio, 1999] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 28:1019–1025, 1999.

[Stinger and Rolls, 2002] S. M. Stinger and E. T. Rolls. Invariant object recognition in the visual system with novel views of 3d objects. *Neural Computation*, 14(11):2585–2596, November 2002.

[Tarr and Bülthoff, 1998] M. Tarr and H. Bülthoff. Image-based object recognition in man, monkey and machine. *Cognition*, (67):1–20, 1998.

[Ullman, 1998] S. Ullman. Three-dimensional object recognition based on the combination of views. *Cognition*, 67:21–44, 1998.

[Wallis, 1996] G. Wallis. How neurons learn to associate 2d-views in invariant object recognition. Technical Report 37, Max-Planck-Institute für Biologishe Kybernetik, Tübingen, Germany, 1996.