Concurrent Learning of Visual Codebooks and Object Categories in Open-ended Domains*

Miguel Oliveira[1], Luís Seabra Lopes[12], Gi Hyun Lim[1],
S. Hamidreza Kasaei[1], Angel D. Sappa[34] and Ana Maria Tomé[12] .

***Abstract*— In open-ended domains, robots must continuously learn new object categories. When the training sets are created offline, it is not possible to ensure their representativeness with respect to the object categories and features the system will find when operating online. In the Bag of Words model, visual codebooks are usually constructed from training sets created offline. This might lead to non-discriminative visual words and, as a consequence, to poor recognition performance. This paper proposes a visual object recognition system which concurrently learns in an incremental and online fashion both the visual object category representations as well as the codebook words used to encode them. The codebook is defined using Gaussian Mixture Models which are updated using new object views. The approach contains similarities with the human visual object recognition system: evidence suggests that the development of recognition capabilities occurs on multiple levels and is sustained over large periods of time. Results show that the proposed system with concurrent learning of object categories and codebooks is capable of learning more categories, requiring less examples, and with similar accuracies, when compared to the classical Bag of Words approach using codebooks constructed offline.**

I. Introduction

In order to be effective, service robots need to robustly recognize the objects present in the environment in which they operate. This is a challenging task because of the very large number of objects present in household environments and, moreover, due to the infinite variety in appearance of those objects. Given this, the appropriate strategy to solve the problem is to make robots capable of learning on site, rather than to exhaustively program them before deployment. This calls for online and incremental learning capabilities, in the sense that the representation of known object models should be enhanced (e.g., augmented, corrected or replaced) over time. Furthermore, the need to learn previously unknown object categories implies that the learning system is open-ended, as in a problem of multinomial classification in which the number of classes is continuously increasing. Thus, open-ended learning comprises both online and incremental learning, but it also requires the ability to learn additional categories regularly [1], [2], [3], [4].

Note that the specific requirements of open-ended domains rule out many of the state of the art machine learning techniques used in *Visual Object Recognition* (VOR). One reason is that computer based VOR approaches are typically structured in two phases, learning and recognition, which are often secluded from each other. Deep artificial neural networks [5], for example, are incremental in nature (every new training observation can be used for updating the weights in the network), but not open-ended, since the inclusion of novel categories enforces a restructuring in the topology of the network. In humans there is no evidence of a partition between learning and recognition. On the contrary, several studies suggest that VOR learning is a protracted process that starts in early childhood and follows through to adolescence [6], [7]. Experiments in children from 6 to 11 years old, designed to assess their object recognition skills, suggested a maturation process of complex visual perceptual abilities, possibly related to the development of the cerebral processes involved in object recognition [8]. Given that children progressively learn more object categories, it appears there is a concurrent development of both the object recognition skills as well as of the underlying visual capabilities used in that recognition.

Bag of Words (BOW) representations are suitable for open-ended domains because they generate compact representations of the object views, which enables the categories to be represented as a set of histograms of views. This is known as an instance based approach (see [2] and [9] for examples of open-ended systems using instance based approaches). Objects may be recognized by computing the nearest neighbour between the histogram of the target view and the histograms of all the known views. Note that as an alternative, more advanced classification rules can be used, provided they can handle open-ended domains (see [10] for a survey of classifiers used in BOW).

The codebooks used in the BOW models are constructed *offline* by feeding a sample set of features (the codebook training set) to a clustering algorithm, e.g., K-means [11], hierarchical K-means [12], Mean-shift [13], randomized clustering forests [14] or Gaussian Mixture Models [15].

*This work was funded by the EC 7th FP theme FP7-ICT-2011-7, grant agreement no. 287752 (project RACE - Robustness by Autonomous Competence Enhancement), by FEDER, grant FCOMP-01-0124-FEDER-022682, and by FCT, grant Incentivo/EEI/UI0127/2013. The work of Dr. A. Sappa has been partially supported by the Spanish Government under Project TIN2014-56919-C3-2-R and PROMETEO Project of the "Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación de la República del Ecuador".

Emails: {mriem,lsl,lim,seyed.hamidreza}@ua.pt, asappa@cvc.uab.es, ana@ua.pt

[1]IEETA - Instituto de Engenharia Electrónica e Telemática de Aveiro, Universidade de Aveiro. {mriem,lsl,ana}@ua.pt

[2] Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro.

[3] Computer Vision Center, Edificio O, Campus UAB 08193 Bellaterra Barcelona, España.

[4] Facultad de Ingeniería Eléctrica y Computación (FIEC), Escuela Superior Politécnica del Litoral (ESPOL), Campus Gustavo Galindo, Km 30.5 vía Perimetral, 09-01-5863. Guayaquil-Ecuador.

Because clustering is done offline, the representativeness of this training set, i.e., how well the features in the set describe the features that the system will find when running online, becomes critical to the clustering process and, consequently, to the performance of the system. This is most noticeable in open-ended domains, because the categories to be learned are not known a priori and may yield feature patterns which were not included in the codebook training set.

The contribution of this paper is to present a method to *concurrently* learn both the object category models and the visual word codebook used to encode those category models. The codebook is learned in an unsupervised and online fashion, in parallel with an open-ended supervised learning of the categories. Thus, in this work, the feature-level knowledge, i.e., the codebook, is also dynamic, and changes at this level imply a restructuring on the object-level knowledge, including object view representations and object category models. The approach contains similarities with the human vision system, because it is not only suited for learning novel object categories continuously, but is also capable of learning novel resources, i.e., codebook words, which can be used to better discriminate those categories. The concurrent learning of categories and codebooks endows the system with more proficiency in coping with novel object categories, in particular in cases where the data contains unprecedented patterns in the feature space.

The remainder of this paper is organized as follows: Section II reviews related work, Sections III, IV and V describe the proposed approach. Results are given in Section VI, followed by conclusions in Section VII.

II. Related Work

The BOW model proposes a simplified representation of the features extracted from an object view, through the grouping of those features into clusters [13], [15]. These clusters are commonly referred to as *words*, and the set of words used to describe the object is called the *dictionary* or *visual codebook*. Each feature in an object view is assigned to a codebook word using a nearest neighbor strategy, or its weight is distributed over the set of words in a probabilistic manner [16]. This produces a histogram, which contains the relative frequency of features in each word.

One of the critical challenges in any BOW approach is how to define a good codebook. Some authors have addressed this by proposing to bond together the codebook construction with the training of object category models. The rationale is that, if the clustering is disconnected from the classification, the representation will not capture the aspects of the data that are most useful for the classification problem. In [17], a framework is proposed that unifies the clustering and classification problems using a Discriminative Cluster Refinement approach [18]. Results show this is especially useful when the size of the training set is limited. A generative model based on latent aspects for explaining images at feature descriptors level is proposed in [19]. In this case, the codebook is a built-in component of the model learned simultaneously with other parameters. In [20], an optimization method is proposed that takes into account the category information as additional information for the construction of the codebook. These methodologies have interesting recognition performances, which shows that the construction of the codebook is a critical component for the recognition performance. However, it is not straightforward how these approaches could be applied to open-ended domains.

An alternative approach to the problem is to start with a very large offline codebook and then refine it. In [21], an initially large codebook is compressed into a compact representation learned from training data, by pair-wise merging of visual words. Results show that the refined codebook often displays better recognition accuracy when compared to the initial codebook. In [15], object categories are represented by bipartite histograms. These histograms contain the category signature in an universal codebook as well as in codebooks specifically adapted to each of the categories. However, all codebooks are constructed offline during a training stage. In addition, since the adaptive codebooks are derived from the universal one, this approach has the constraint that the number of words in the adaptive codebooks must be the same as in the universal codebook, which in turn, must be defined a priori.

There are some works which propose online updating of BOW codebooks: in [22], the codebook update mechanism is limited to the merging of clusters or creation of new ones. That means that a complete restructuring of the codebook (if the data suggest that is the best way to proceed) is not possible using this approach; in [23], an online optimization algorithm based on stochastic approximations is presented which can be applied to codebook learning; in [24], a recursive Least Squares learning algorithm is proposed to continuously update the codebook, in order to avoid the (many times unfeasible) batch processing of large training sets. While [22] focuses on scene recognition for SLAM applications, the works in [23] and [24] only approach the mechanisms for updating the codebooks online, without evaluating the impact on the performance of object recognition. None of those works targets concurrent learning, nor the problem of open-ended learning of object categories.

Previous work from the authors on this topic focused on the design and evaluation of open-ended learning systems, e.g., [1], [2], [9].

III. System Overview

The proposed system is illustrated in Fig. 1. Note that the visual words are defined in an unsupervised way as Gaussians over high-dimensional feature spaces. For that reason, there is no guarantee that they are as meaningful as those used in the example of Fig. 1, e.g., legs or tails. A new view of a bird is given to the system for learning a novel category *Bird*. At that time, the codebook (Previous Codebook in Fig. 1) contained three words: *heads*, *legs* and *tails*. The system also knew two categories: Category *Fish* which contained 1 head and 1 tail, and category *Cat*, which yielded 1 head, 4 legs and 1 tail. During the training of a new

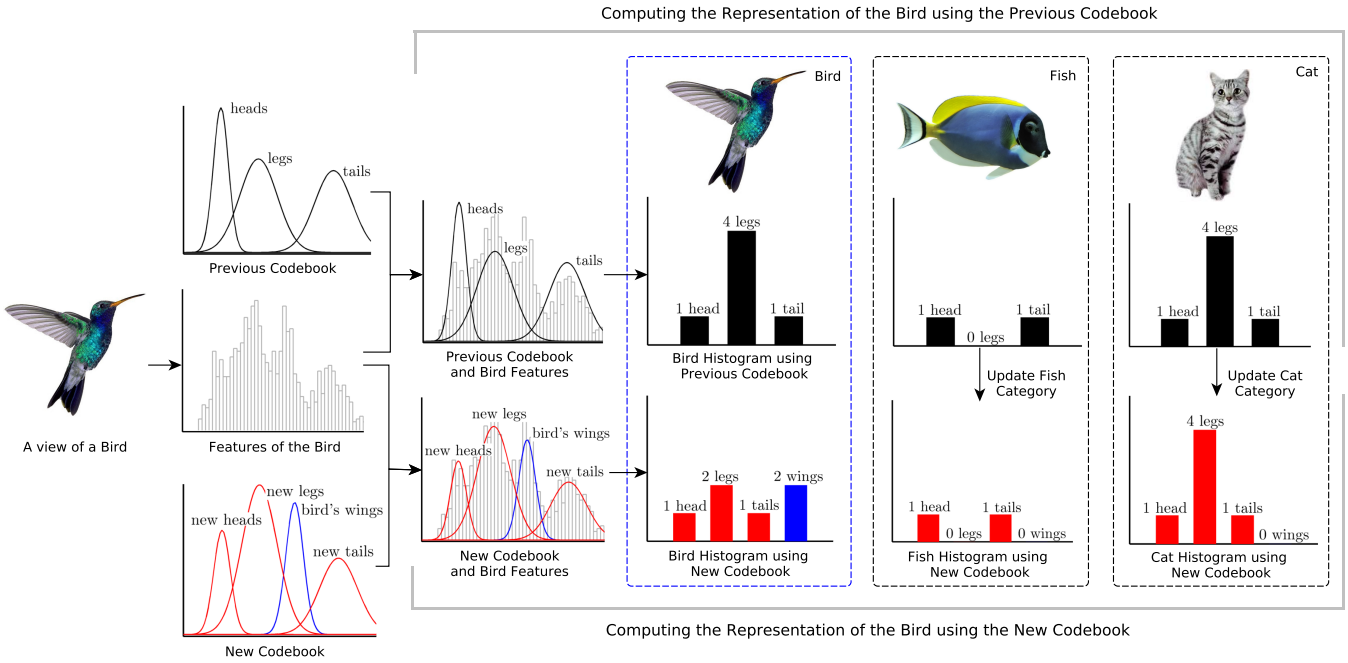


Fig. 1. An example which shows the advantages of learning the codebook online: because the bird's wings are included in the *legs* word of the previous codebook, the training of a new *bird* category leads to a bird histogram which is the same as the *cats* histogram, i.e., 1 *head*, 4 *legs* (2 legs plus 2 wings) and 1 *tail*; On the other hand, if a codebook update is conducted before training the bird category (see Fig. 2), a new word *bird's wings* is created and the bird's histogram is distinct from the *cats*.

category *Bird* without updating the codebook, the system must describe that category based on the amount of heads, legs and tails the bird displays. The features belonging to the bird's wings fall outside all words (see in Fig. 1 that there are some features between the Gaussians that correspond to the legs and tails). Nonetheless, these features are assigned to the nearest words (the legs in this case). As a result, the total number of legs in the bird category is 4, 2 actual bird's legs plus the two wings included as legs. The result is that the representation of the *Bird* is the same as the one that describes the *Cat*. This will lead to poor classification performance.

The problem in the example above is that the new view contains unprecedented features (those describing the wings) which cannot be adequately described by the previous codebook. The solution we propose is to run a codebook update before training the new category (the details are presented in the next Sections). In Fig. 1, the new codebook contains one additional word, the *bird's wings*, and the histogram of the bird created using this codebook (i.e., birds have 1 head, 2 legs, 1 tail and 2 wings) is distinct from the one from the cats (1 head, 4 legs, 1 tail, 0 wings). This example unfolds some of the advantages of using dynamic codebooks.

IV. Features, Objects and Categories

In this work an instance-based approach is used. That means, a category is represented by a set of histograms of views of objects of that category. The instance-based approach is used because it is a baseline method for category representation. However, more advanced methodologies can be easily plugged into the dynamic codebooks approach, provided that they can learn incrementally and in an open-ended fashion. Similarly, we use a baseline recognition mechanism in the form of a nearest neighbor classification rule (Euclidean distance). That is, the system will compare the view to be recognized against all the views stored, and output the category label of the view which was more similar to the unknown view.

A. Modelling the Feature Space using GMM

To model the distribution of the observed features over the feature space we use *Gaussian Mixture Models* (GMMs). GMMs are parametric finite mixture models, i.e., they assume the observations are explained by a linear combination of a finite set of components in the mixture, and that those components can be represented by known distributions, in this case Gaussian distributions. Mixture models are interesting because it is relatively easy to fit these models to data. Since mixture models assume the data is a composite of several components, it is forthright to consider that these components can be used in the task of clustering, i.e., that each component corresponds to a cluster. This approach is often referred to as model-based clustering.

Let $\{x_1,\ldots,x_n\}$ denote a set of key points, typically obtained from an object view. The distribution over the feature space can be seen as a multidimensional hyper surface $f(x)$ where x is a d-dimensional point in the feature space. We

approximate $f(x)$ by fitting a GMM to $\{x_1,\ldots,x_n\}$, thus:

$$\hat{f}(x) = \sum_{i=1}^{Q} \omega_l \cdot p(x,\mu_i,\Sigma_i), \tag{1}$$

where ω_i denotes the weight associated to the corresponding i component in the Q number of components GMM and $p(\cdot)$ denotes the value of the probability density function at point x, given a multivariate Gaussian with d dimensional mean μ_i and $d \times d$ covariance matrix Σ_i. The fitting is executed using an *Expectation Maximization* (EM) framework [25], which searches the values of the model parameters ω_i, μ_i and Σ_i, $\forall i \in \{1,2,\ldots,Q\}$, that define a GMM which gives the features the greatest probability. One drawback of this method is that, since GMMs are finite models, the number of components in the mixture must be given as an input parameter (Q in eq. (1)). In this particular case, however, since we are using EM for fitting a transient GMM, which will them be simplified, Q may be set to a high value because the simplification step will collapse overlapping components in the mixture. Another problem with fitting GMMs using EM is that the values of all observations must be available to the algorithm. This is not problematic because, in our approach, EM fitting is used to model the features of a single object view; an additional GMM (which is computed as detailed in Section V) is used to represent the distribution of all the observed features over multiple categories and views. This global GMM defines the codebook, i.e., each component in the GMM corresponds to a word in the codebook.

B. Histograms as Representations of Object Views

An object view is represented by a histogram which contains the normalized frequency of the occurrences for each word in the codebook. This is the standard representation as defined by the BOW. Let $\mathscr{C}$ denote a codebook composed of K words w, such that $\mathscr{C} = \{w_1, w_2, \cdots, w_K\}$ and $v^{(\mathscr{C})}$ denote the histogram computed using codebook $\mathscr{C}$, defined as $v^{(\mathscr{C})} = \{v_1, v_2, \ldots, v_K\}^{(\mathscr{C})}$, where v_k denotes the normalized frequency of the occurrences of features in the view which are included in codebook word k. Let a given object view be composed of N key points. Each key point i produces a d dimensional feature vector which represents a point in the feature space and is represented as x_i. Then, v_k is computed as the sum of the measures of membership of x_i for word k:

$$v_k = \sum_{i=1}^{N} P(x_i \in w_k \mid x_i), \tag{2}$$

where the measure of membership is estimated as the conditional probability $P(x_i \in w_k \mid x_i)$. Note that, rather than assigning each x_i to a single word, the procedure estimates a probability that each x_i belongs to word k. This is known as soft clustering or model-based clustering, since it makes use of the GMM. It was shown in [16], [26] that soft clustering techniques outperform hard clustering methods. Using Bayes' theorem, we can formulate $P(x \in w \mid x_i)$ as follows:

$$P(x_i \in w_k \mid x_i) = \frac{P(w_k) \cdot P(x_i \mid x_i \in w_k)}{P(x_i)}, \tag{3}$$

where $P(w_k)$ is the prior associated to the *kth* component of the Gaussian mixture (ω, in eq. (1)), $P(x_i)$ is the total probability of x_i as given by the mixture model (see eq. (1)) and $P(x_i \mid x_i \in w_k)$ is given by the probability density function of the Gaussian of the corresponding mixture component denoted as $p(\cdot)$. Thus, eq. eq. (2) can be rewritten as:

$$v_k = \sum_{i=1}^{N} \left(\frac{\omega_k \cdot p(x_i,\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \omega_j \cdot p(x_i,\mu_j,\Sigma_j)} \right). \tag{4}$$

Figure 2 shows an example of EM fitting. The features extracted from the image of the bird are fitted to a four component view GMM.

V. Online Visual Codebook Learning

Suppose at time t_0 there is a codebook $\mathscr{C}^{(t_0)}$, represented by a GMM which models the distribution of key points observed up to time t_0, denoted as $g(x)^{(t_0)}$:

$$g(x)^{(t_0)} = \sum_{i=1}^{K^{(t_0)}} \omega_i^{t_0} \cdot p(x,\mu_i^{t_0},\Sigma_i^{t_0}). \tag{5}$$

A new object view is collected at time t_1, which generates a new set of feature vectors $\{x_1,\ldots,x_n\}^{(t_1)}$. Let the distribution of the features in this view be represented by $h(x)$. This distribution is modelled by an additional *view* GMM, which is fitted to the features of the view (see section IV-A for a detailed description):

$$h(x) = \sum_{i=1}^{Q} \alpha_i \cdot p(x,v_i,A_i), \tag{6}$$

where Q denotes the number of components of the view GMM and α_i, v_i and A_i denote the weight, mean and covariance respectively, of the *ith* component of the GMM fitted to $\{x_1,\ldots,x_n\}^{(t_1)}$. Note that using the EM algorithm as described in section IV-A, we obtain the parameters α_i, v_i and A_i, $\forall i \in \{1,\ldots,Q\}$. This means that there are two known distributions $g(x)^{(t_0)}$ and $h(x)$. The updated codebook $\mathscr{C}^{(t_1)}$ should reflect a new distribution computed from the merging of those distributions. For the merging of two GMMs, we use a simplified version of the method presented in [27]. The simplification is designed to improve the computational efficiency of the algorithm, since our approach is designed to handle high dimensional feature vectors computed after large amounts of sensor data. The merging of two GMMs is defined in two steps: a concatenation which is followed by a simplification. Thus, the output distribution $g(x)^{(t_1)}$ can be written as follows:

$$g(x)^{(t_1)} = \mathrm{simp}\left(\mathrm{conc}\left(g(x)^{(t_0)}, h(x)\right)\right), \tag{7}$$

where $\mathrm{conc}(\cdot)$ denotes the concatenation of two GMMs and $\mathrm{simp}(\cdot)$ the simplification of a GMM. The concatenation of two GMMs is a trivial combination of the two GMMs,

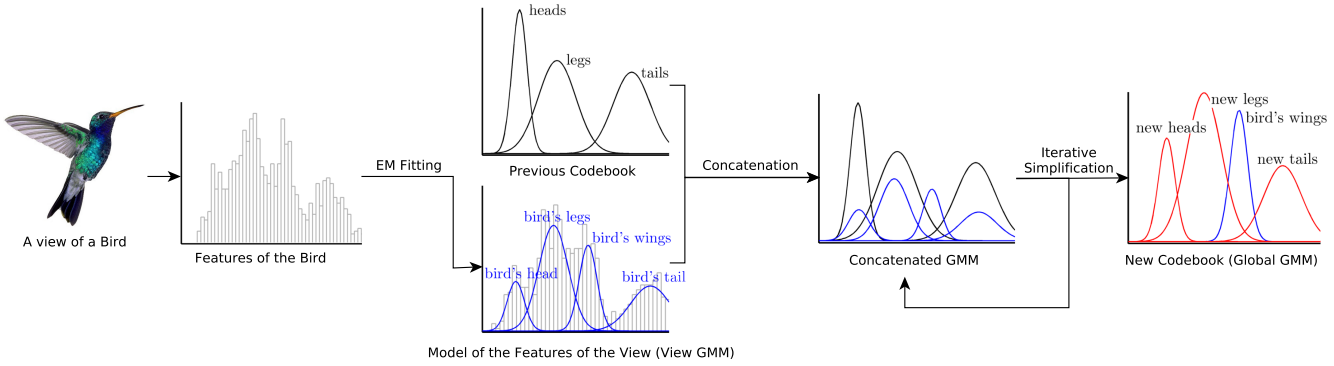


Fig. 2. The codebook update is composed of an EM fitting of the features of a view, a concatenation and an iterative simplification.

resulting in a concatenated distribution which abides to the following:

$$q(x) = s_{g(x)} \cdot g(x)^{(t_0)} + s_{h(x)} \cdot h(x) \tag{8}$$

where $s_{g(x)}$ and $s_{h(x)}$ are factors that rescale the prior of each of the input GMMs, such that:

$$s_{g(x)} = \frac{N_{g(x)}}{N_{g(x)} + N_{h(x)}}, \tag{9}$$

$$s_{h(x)} = \frac{N_{h(x)}}{N_{g(x)} + N_{h(x)}}, \tag{10}$$

where N is the total number of key-points each distribution represents. Thus, the concatenated GMM contains $K^{(t_0)} + Q$ mixture components and is defined as follows:

$$q(x) = \sum_{i=1}^{K^{(t_0)}+Q} \beta_i \cdot p(x, w_i, B_i), \tag{11}$$

where w_i and B_i are the mean and covariance of the concatenated GMM, and β_i the priors obtained from eq. (9) or eq. (10) as appropriate.

After concatenation, the GMM is simplified. The goal of simplification is to reduce the complexity of the model, namely by decreasing the number of components in the mixture, i.e., $K^{(t_1)} \leq K^{(t_0)} + Q$. Note that after concatenation, some components in the mixture may be very similar. While fusing similar components will certainly reduce the complexity of the model, a more evident advantage could come from the fact that overlapping components result in ambiguous encoding of the features, which degrades the classification performance. The simplification algorithm runs iteratively, fusing a pair of components at each iteration. The fusion of two Gaussian components is defined as follows: let β_i, w_i, B_i and β_j, w_j, B_j be the parameters of the Gaussians associated with the ith and jth components in the concatenated GMM. The goal is to produce a new Gaussian with parameters $\beta_{\langle i,j \rangle}$, $w_{\langle i,j \rangle}$, $B_{\langle i,j \rangle}$ which results from the fusion of components i and j. They are given by the usual equations for the combination of cumulative densities [28]:

$$\beta_{\langle i,j \rangle} = \beta_i + \beta_j, \tag{12}$$

$$w_{\langle i,j \rangle} = \frac{\beta_i \cdot w_i + \beta_j \cdot w_j}{\beta_{\langle i,j \rangle}}, \tag{13}$$

$$B_{\langle i,j \rangle} = \frac{\beta_i \left(B_i + \left(w_i - w_{\langle i,j \rangle} \right)' \left(w_i - w_{\langle i,j \rangle} \right) \right)}{\beta_{\langle i,j \rangle}} + \frac{\beta_j \left(B_j + \left(w_j - w_{\langle i,j \rangle} \right)' \left(w_j - w_{\langle i,j \rangle} \right) \right)}{\beta_{\langle i,j \rangle}}. \tag{14}$$

At each iteration of the algorithm, all pairwise combinations of Gaussian components are analysed and an hypothetically fused Gaussian $\langle i, j \rangle$ is estimated for all $i < j \leq K^{(t_0)} + Q$. Then, the best hypothesis for fusing is defined by measuring the Kullback-Leibler divergence of the discrete probability distributions of the combined two input Gaussians against the output fused Gaussian. The best candidate is fused if the following holds:

$$\underset{i,j}{\operatorname{argmin}} \, D_{KL}(R_{i,j}(X) \| R_{\langle i,j \rangle}(X)) < T, \tag{15}$$

where T is the *maximum divergence threshold*, D_{KL} denotes the Kullback-Leibler divergence, X represents a set of O points sampled using the distribution of the hypothetically fused Gaussian, and $R_{i,j}(X)$ is the discrete probability distribution obtained by the sum of the values of Gaussian components i and j at sample points X:

$$R_{i,j}(X) = \beta_i \cdot P(X, w_i, B_i) + \beta_j \cdot P(X, w_j, B_j), \tag{16}$$

where $P(X, w, B)$ denotes the evaluation of probability of the Gaussian with parameters w and B, $\forall x \in X$, and $R_{\langle i,j \rangle}(X)$ is the discrete probability obtained by the fused Gaussian hypothesis:

$$R_{\langle i,j \rangle}(X) = \beta_{\langle i,j \rangle} \cdot P(X, w_{\langle i,j \rangle}, B_{\langle i,j \rangle}). \tag{17}$$

The simplification continues to iterate until eq. (15) is not verified, in which case the simplification is completed. When a pair $\langle i, j \rangle$ is fused and a simplification occurs, components i and j are deleted from the list of components in the mixture, a new component corresponding to the fused Gaussian is inserted at the end of the list, and the number of components

L is decremented. In the next iterations, only a subset of the possible combinations must be updated, i.e., $\langle i, j\rangle, \forall i \in \{1, L-1\} \wedge j = L$. This makes the simplification in the subsequent iterations considerably faster.

Figure 2 shows an example where a codebook is updated. In the example, the previous codebook contained words *heads*, *legs* and *tails*. The fitting of the view of the bird resulted in four new words, *birds' head*, *birds' legs*, *birds' tail* and *birds' wings*. After, simplification, similar words were merged (e.g., *head* is merged with *bird's head* resulting in *new head*), and the resulting codebook contains one additional word which can describe the birds wings. This example shows that the mechanism is capable of fusing similar words and creating words which can describe unprecedented features. When a new codebook is learned, the histograms of known categories must be updated. At the moment, we do this by storing the features of each view and using them to recompute the histograms.

VI. Results

Although there are well established methodologies to evaluate learning systems, e.g., k-fold cross validation, leave-one-out, etc, these approaches follow the classical train-and-test procedure, i.e., two separate stages, training followed by testing. Training is accomplished offline, and once it is complete the testing is performed. These methodologies are not well suited to evaluate open-ended learning systems, because they do not abide to the simultaneous nature of learning and recognition and because those tests imply that the number of categories must be predefined.

An evaluation protocol for open-ended learning systems was proposed in [1], [2]. The idea is to emulate the interactions of a recognition system with the surrounding environment over large periods of time. The protocol interfaces with a recognition system by continuously asking it to recognize new views of known object categories. The teaching protocol progressively estimates the recognition performance of the system and, in case this performance exceeds a given threshold, introduces an additional object category [1]. The protocol uses three basic actions to interact with a VOR system: *teach*, used for teaching a new object category, *ask*, used to ask the system what is the category of an object view and *correct*, used for providing the system with an additional (labelled) view of an existing category. The idea is to continuously ask the system to recognize the categories of previously unseen views of known categories and provide corrections when needed. This way, the system is trained, and at the same time the recognition accuracy of the system is continuously estimated. To operate, the protocol must be connected to a dataset of object views. In this work, we use the *RGB-D Object Dataset* described in [29]. This dataset contains 300 common household objects, organized into 51 categories.

[1]See `https://www.youtube.com/watch?v=N-Vgc7wPuxM` and `https://www.youtube.com/watch?v=pEYxx9O7288` for examples of the teaching protocol evaluations

To evaluate open-ended learning systems, one must assess the performance of the system continuously over the multiple stages of the learning process. That is to say that the performance of these systems is also dynamic. Moreover, the performance of an open-ended learning system is not limited to the accuracy with which the system recognizes learned objects. In fact, we consider three distinct questions for assessing the overall performance of an open-ended learning system, and propose ways of using the teaching protocol experiments to measure them: **How much does it learn?** We measure this as the number of categories learned in a teaching protocol experiment (see [1], [2] for early studies of this kind); **How well does it learn?** Standard evaluations for offline learning such as accuracy, precision and recall are well suited for this[2]; **How fast does it learn?** We measure this as the number question/correction iterations in a teaching protocol experiment up to a certain number of learned categories[3].

Since our goal is to see if dynamic codebooks improve the performance of learning systems, we compare our approach with the static codebook approach, where the codebook is constructed offline, using an EM fitting procedure with a set of features obtained from multiple views as input (see section IV-A). It is necessary to define the number of words of the initial codebook as well as the examples (object views) that are given for clustering the features, i.e., the representativeness of the data used to compute the static codebook. This is defined as a percentage of the number of object categories in the dataset. For example, if a codebook has a 0.1 representativeness, it means views from $0.1 \times 51 \sim 5$ random object categories were used to create the static codebook. Representativeness is used to reproduce the fact that it is very hard (if not impossible) to come up with a subset of object views that is sufficient to represent all the possible variations the system will encounter when running online. The dynamic codebooks of the proposed approach are also initialized with codebooks containing different number of words. The representativeness of all dynamic codebooks is 0.05. Ten protocol evaluation experiments were made for each of the approaches, i.e., static and dynamic codebook approaches. Figure 3 shows one protocol evaluation experiment for one of the dynamic codebook approaches. Teaching of new categories and points where the codebook is updated are marked in the figure. In these experiments, the codebook is updated whenever the system does not learn a new category after 75 iterations.

A simple color-based feature descriptor defined by the α, β components of the $l\alpha\beta$ colorspace is used (see [30] for a survey of color descriptors used in VOR). Although this is a simple feature descriptor, it serves the purpose of proving the concept and is applied without loss of generality in the

[2]We use the global accuracy, measured as the ratio between correct answers and number of questions, since the beginning of the teaching protocol experiment.

[3]We do not refer to the computational complexity of the algorithms, which is often estimated by measuring the time an algorithm takes to learn or recognize. Rather, the focus here is on how many examples the algorithm requires to accurately learn the object category.

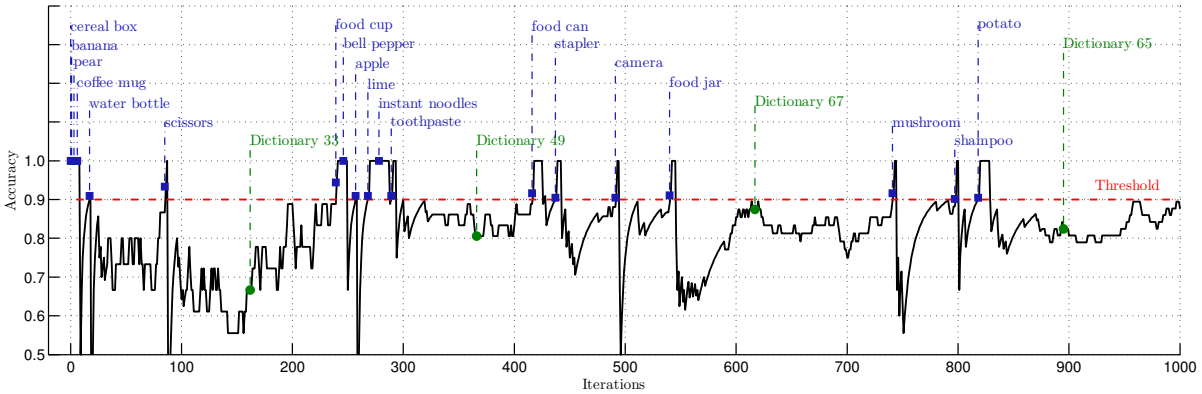
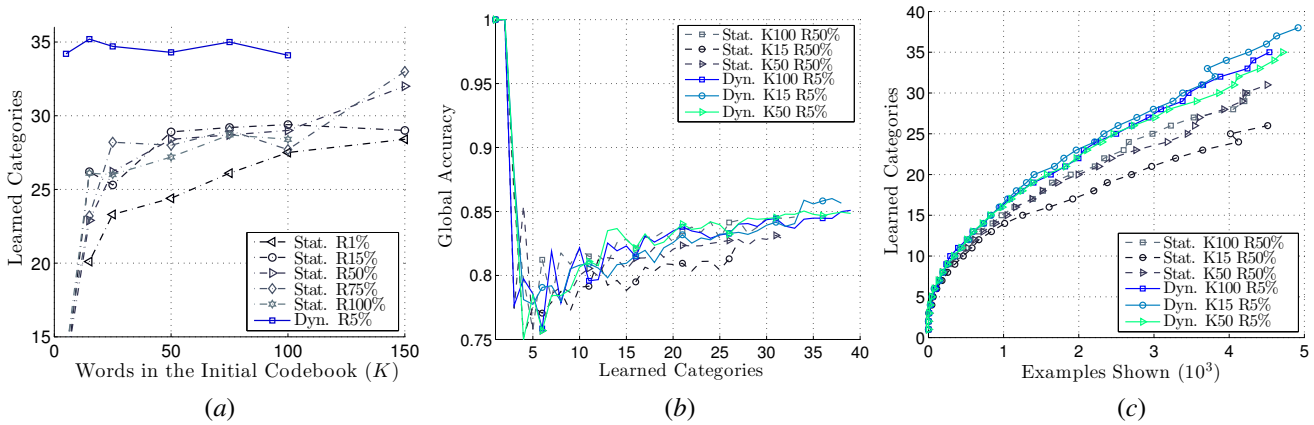


Fig. 3. Open-ended evaluation protocol. One experiment with a dynamic codebook initialized with 8 words. For better visualization, only the first 1000 iterations of the experiment are shown.



(*a*) (*b*) (*c*)

Fig. 4. Comparison between dynamic and static codebooks, using different configurations for the initial number of words (K) and representativeness of the codebook (R): (*a*) total number of learned categories as a function of K and R; (*b*) recognition accuracy as a function of the number of learned categories; (*c*) number of categories learned as a function of the number of examples shown i.e., the number of iterations in the experiment.

proposed mechanisms, since the technique is applicable to any d-dimensional feature space.

Fig. 4 (*a*) shows the number of learned categories at the end of the protocol evaluation experiment (average of the ten experiments per approach). Note that the number of words is small, which could be explained by the simple (2D) feature descriptor we are using. The dynamic codebook approaches are able to learn about 35 categories, while the static codebooks typically learn less than 30 categories. The best static codebook is configured with representativeness 0.75 and $K = 150$. In general, the larger the representativeness, the higher the number of learned categories. As expected, codebooks which are created using more views are better. In the static codebooks, the increase in the number of words leads to an increase in the number of learned categories. This improvement is observable only until a certain number of words (50 words). This makes sense: codebooks with very few words are not discriminative, enough but there is a point where increasing the number of words brings no advantage. One interesting observation is that dynamic codebooks seem to be insensible to the number of words in the initial codebook. That means that the initialization does not affect the performance of the dynamic codebooks. This is a relevant observation because constructing offline codebooks is sometimes a complex, time consuming procedure.

Fig. 4 (*b*) shows the global accuracy obtained by the tested approaches as a function of the number of learned categories. In all approaches, the accuracy decreases up to a certain number of categories learned, and then starts to slightly increase as more categories are learned. With the increase in the number of categories, it is expected that the accuracy decreases. However, as the number of learned categories learned increases, also the number of examples per category increases, which improves the performance of the system. Accuracy values are similar both for the dynamic and static codebooks, which means both alternatives learn the categories with similar accuracy.

Fig. 4 (*c*) shows the number of question/correction iterations required to learn a certain number of categories. Dynamic codebooks learn faster that static codebooks, i.e., they require less examples to learn the same number of categories. This is interesting, since it shows that sometimes is might be better to operate a larger restructuring of knowledge, eventhough, at first sight, it could appear to be a step back. Results show that, in the long term, the speed of learning benefits.

VII. Conclusions

This paper proposed a methodology for the concurrent learning of BOW codebooks as well as of visual object categories. The codebook learning algorithm is based on the concatenation and simplification of GMMs which can be used for creating, maintaining and updating the codebooks used in BOW models. The work focuses on open-ended learning scenarios, where the usage of static codebooks is inappropriate. The proposed approach can be seen as an extension of the BOW approach for open-ended learning scenarios, thus making recognition systems more adaptable. The dynamic codebooks were compared with static codebooks using an open-ended evaluation protocol. Results show that dynamic codebooks are capable of learning more categories, with similar accuracy, requiring less examples.

Future work should include the testing of this approach with higher-dimensional feature descriptors, with which we expect that the higher dimensionality of the feature space should make the advantages of dynamic codebooks even more evident. Similarly, more advanced encoding mechanisms such as Fisher [31] or Super Vectors [32] should be tested. In addition to that, the focus should be on how to transfer category knowledge when a codebook update occurs, in particular on how to do this without storing the features of the observed views. Another open issue is how to establish criteria used to define the moments in which the codebooks should be updated.

References

[1] L. Seabra Lopes and A. Chauhan, "How many words can my robot learn?: An approach and experiments with one-class learning," *Interaction Studies*, vol. 8, no. 1, pp. 53 – 81, 2007.

[2] A. Chauhan and L. Seabra Lopes, "Using spoken words to guide open-ended category formation," *Cognitive processing*, vol. 12, no. 4, pp. 341–354, 2011.

[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, and T. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the Conf. on Artificial Intelligence*, 2010.

[4] S. Kirstein, H. Wersing, H.-M. Gross, and E. Körner, "A life-long learning vector quantization approach for interactive learning of multiple categories," *Neural Networks*, vol. 28, pp. 90–105, 2012.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview," *CoRR*, vol. abs/1404.7828, 2014. [Online]. Available: http://arxiv.org/abs/1404.7828

[6] M. Juttner, A. Muller, and I. Rentschler, "A developmental dissociation of view-dependent and view-invariant object recognition in adolescence." *Behav Brain Res*, vol. 175, no. 2, pp. 420–4, 2006.

[7] M. Nishimura, S. Scherf, and M. Behrmann, "Development of object recognition in humans," *F1000 Biology Reports*, 2009.

[8] S. Bova, E. Fazzi, A. Giovenzana, C. Montomoli, S. Signorini, M. Zoppello, and G. Lanzi, "The development of visual object recognition in school-age children." *Develop. in Neuropsychology*, vol. 31, no. 1, pp. 79–102, 2007.

[9] M. Oliveira, G. H. Lim, L. Seabra Lopes, H. Kasaei, A. Tome, and A. Chauhan, "A perceptual memory system for grounding semantic representations in intelligent service robots," in *Intelligent Robots and Systems. IEEE/RSJ Int. Conf. on*. IEEE, 2014.

[10] J.-X. Huang, Y.-Y. Qu, C.-H. Li, and M.-J. Hu, "The comparison of classifiers for object categorization based on bag-of-word technology," in *Pattern Recognition, Chinese Conf. on*, November 2009, pp. 1–5.

[11] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," in *Computer Vision, IEEE Int. Conf. on*, Oct 2003, pp. 1470–1477 vol.2.

[12] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Computer Vision and Pattern Recognition. IEEE Conf. on*, 2006, pp. 2161–2168.

[13] F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition," in *Computer Vision, IEEE Int. Conf. on*, vol. 1, Oct 2005, pp. 604–610 Vol. 1.

[14] F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," in *Neural Information and Processing Systems. Int. Conf. on.*, 2007.

[15] F. Perronnin, "Universal and adapted vocabularies for generic visual categorization," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 30, no. 7, pp. 1243–1256, July 2008.

[16] J. Van Gemert, C. Veenman, A. W. M. Smeulders, and J.-M. Geusebroek, "Visual word ambiguity," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 32, no. 7, pp. 1271–1283, July 2010.

[17] L. Yang, R. Jin, R. Sukthankar, and F. Jurie, "Unifying discriminative visual codebook generation with classifier training for object category recognition," in *Computer Vision and Pattern Recognition. IEEE Conf. on*, June 2008, pp. 1–8.

[18] L. Yang, R. Jin, C. Pantofaru, and R. Sukthankar, "Discriminative cluster refinement: Improving object category recognition given limited training data," in *Computer Vision and Pattern Recognition. IEEE Conf. on*, June 2007, pp. 1–8.

[19] D. Larlus and F. Jurie, "Latent mixture vocabularies for object categorization and segmentation," *Image and Vision Computing*, vol. 27, no. 5, pp. 523 – 534, 2009.

[20] C. Zhang, J. Liu, Y. Ouyang, Q. Tian, H. Lu, and S. Ma, "Category sensitive codebook construction for object category recognition," in *Image Processing. IEEE Int. Conf. on*, November 2009, pp. 329–332.

[21] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *Computer Vision. IEEE Int. Conf. on*, vol. 2, Oct 2005, pp. 1800–1807.

[22] T. Nicosevici and R. Garcia, "Automatic visual bag-of-words for online robot navigation and mapping," *Robotics, IEEE Trans. on*, vol. 28, no. 4, pp. 886–898, August 2012.

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756008

[24] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *Signal Processing, IEEE Transactions on*, vol. 58, no. 4, pp. 2121–2130, April 2010.

[25] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of the American Statistical Association*, vol. 97, pp. 611–631, 2002.

[26] B. Fernando, E. Fromont, D. Muselet, and M. Sebban, "Supervised learning of gaussian mixture models for visual vocabulary generation," *Pattern Recognition*, vol. 45, no. 2, pp. 897 – 907, 2012.

[27] P. Hall, Y. Hicks, T. Robinson, and U. of Bath. Department of Computer Science, *A Method to Add Gaussian Mixture Models*, ser. Technical report (University of Bath. Department of Computer Science).

[28] A. Declercq and J. H. Piater, "Online learning of gaussian mixture models - a two-level approach." in *VISAPP (1)*, pp. 605–611.

[29] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation. IEEE Int, Conf. on*, 2011, pp. 1817–1824.

[30] K. Van de Sande, T. Gevers, and C. Snoek, "Evaluating color descriptors for object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 32, no. 9, pp. 1582–1596, September 2010.

[31] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 143–156.

[32] X. Zhou, K. Yu, T. Zhang, and T. Huang, "Image classification using super-vector coding of local image descriptors," in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6315, pp. 141–154.