Feature Selection Based on Reinforcement Learning for Object Recognition

Monica Piñol
Computer Science Dept.
Computer Vision Center
Universitat Autònoma de Barcelona
08193-Bellaterra
Barcelona, Spain
mpinyol@cvc.uab.es

Angel D. Sappa
Computer Vision Center
Campus UAB
08193-Bellaterra
Barcelona, Spain
asappa@cvc.uab.es

Angeles López
Computer Science and Engineering Dept.
Universitat Jaume I
12071-Castellón de la Plana
Castellón, Spain
lopeza@icc.uji.es

Ricardo Toledo
Computer Science Dept.
Computer Vision Center
Universitat Autònoma de Barcelona
08193-Bellaterra
Barcelona, Spain
ricard@cvc.uab.es

ABSTRACT

This paper presents a novel method that allows learning the best feature that describes a given image. It is intended to be used in object recognition. The proposed approach is based on the use of a Reinforcement Learning procedure that selects the best descriptor for every image from a given set. In order to do this, we introduce a new architecture joining a Reinforcement Learning technique with a Visual Object Recognition framework. Furthermore, for the Reinforcement Learning, a new convergence and a new strategy for the exploration-exploitation trade-off is proposed. Comparisons show that the performance of the proposed method improves by about 6.67% with respect to a scheme based on a single feature descriptor. Improvements in the convergence speed have been also obtained using the proposed exploration-exploitation trade-off.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis —*Object recognition*

General Terms

Algorithms, Performance, Experimentation

Keywords

Reinforcement Learning, Object Recognition, Q-Learning, Visual Feature Descriptors

1. BACKGROUND

Usually, in Visual Object Recognition (VOR), the scenes are represented in feature spaces where the classification and/or recognition tasks can be done more efficiently. There are several representations with different levels of performance. Often, the feature space is built around "interest points" of the scene. The interest points are obtained with detectors, then, for each "interest point" a descriptor is computed.

Currently, researchers are beginning to work with reinforcement learning in computer vision. For instance, several approaches have been proposed in the image segmentation field [19, 21, 22]. In all these cases, the reinforcement learning has been used to tackle the threshold tuning reaching a similar performance than the state of the art approaches.

There are also some works in the face recognition problem. For instance, in [8] the authors proposes to learn the set of dominant features for each image to recognize the faces using a reinforcement learning based technique.

In the VOR domain, [13] presents a method using bottom-up and top-down strategies joining Reinforcement Learning and Ordinal Conditional Functions. A similar approach has been proposed in [6], which joins Reinforcement Learning with First Order Logic. In [10] and [11], a new method for extracting image features is presented. It is based on "interest points" detection, and uses reinforcement learning and aliasing to distinguish the classes. Finally, in [2] the reinforcement learning method is used to select the best classification in a Bag of Features approach. On the contrary to previous works, in the current work we propose the use of a reinforcement learning based method to learn the best descriptor for each image.

One of the most recent and widely used approach for VOR is Bag of Features (BoF) [4, 5]. Figure 2($left$) shows an illustration of the BoF architecture. In general, a BoF approach consists of four steps detailed next. In the first step, feature extraction is carried out by image feature descriptors. In [15] the performance of descriptors is analysed. The second step is the creation of a dictionary using visual words from a training set; and the third step is the image representation. Both steps are solved with a Vocabulary Tree (VT)

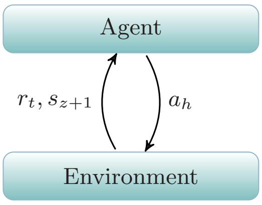

Figure 1: Scheme of interaction between the agent and the environment

[17]. The VT defines a hierarchical tree by k-means clustering. In the algorithm, k defines the branch factor (number of children of each tree-node) and the process is recursive until reaching a maximum depth. At any tree level, each cluster is defined by a dictionary of visual words. The fourth step is the classification.

The BoF approach does not specify the descriptors and the classification algorithms. One solution for determining the descriptors is concatenating all of them, but this solution introduces noise as well as increase the CPU time. Furthermore, since each image could have different characteristics of color, texture, edges, etc.; images from the same database could behave differently when we apply the same descriptor.

This paper proposes a new approach for enhancing the BoF. The key idea is the introduction of the Reinforcement Learning technique [20] to learn the best descriptor for each image. Then, these descriptors are used in a VT scheme, which is used by a Support Vector Machine (SVM) algorithm for the classification. The reminder of the paper is organized as follows. Section 2 summarizes the Reinforcement Learning technique. Then, Section 3 presents the proposed method. Experimental results are provided in Section 4. Finally, conclusions and future work are given in Section 5.

2. REINFORCEMENT LEARNING

Reinforcement learning (RL) is a learning method based on trial and error, where the agent does not have a prior knowledge about which is the correct action to take. The underlying model that RL learns is a Markov Decision Process (MDP). A MDP is defined as a tuple $\langle S, A, \delta, \tau \rangle$, where: S is the set of states; A is the set of actions; δ is a transition function δ:Sx$A \rightarrow S$; and τ is a reward/punishment function τ:Sx$A \rightarrow \Re$.

The agent interacts with the environment and selects an action. Applying the action (a_h) at state(s_z), the environment gives a new state (s_{z+1}) and a reward/punishment (r_t) (see Fig. 1). In order to maximize the expected reward, the agent selects the best action a_h based on the $\tau(s_z, a_h)$ provided by τ:Sx$A \rightarrow \Re$.

Different methods have been proposed in the literature for solving the RL problem: dynamic programming, Monte Carlo methods, and temporal difference learning. In the current work a temporal difference learning based method has been selected since it does not require a model and it is fully incremental [23]. In concrete, our framework is based on the *Q-learning* algorithm [25]. In this case, the agent learns the action policy π:$S \longrightarrow A$, where π maps the current state s_z into an optimal action a_h to maximize the expected long term reward.

The Q-learning proposes a strategy to learn an optimal policy π^*, when the δ function (δ:Sx$A \rightarrow S$) and τ function (τ:Sx$A \rightarrow \Re$) are not known a priori. The optimal policy is $\pi^* = argmax_{a'} \mathbf{Q}(s, a')$, where $\mathbf{Q}$ is the evaluation function the agent is learning. Note that since δ and/or τ are nondeterministic, the environment should be represented in a nondeterministic way. Hence, we can write the $\mathbf{Q}$ in a nondeterministic environment as follows:

$$\mathbf{Q}_n(s_z, a_h) \longleftarrow (1-\alpha_n)\mathbf{Q}_{n-1}(s_z, a_h) + \alpha_n[r + \gamma \max_{a'} \mathbf{Q}_{n-1}(s_{z+1}, a')], \quad (1)$$

$$\alpha_n = \frac{1}{1 + \mathbf{visits}_n(s_z, a_h)}, \quad (2)$$

where $0 \leq \gamma < 1$ is a discount factor for future reinforcements. The Eq. (2) is the value α_n for a nondeterministic world and **visits** is the number of iterations visiting the **Q**-table at the tuple (s_z, a_h) [16].

Since, in this problem is not feasible to update the Q-table using the Eq. (1), because the current state (s_z) is only affected by the previous visits (first order MDP), the expression in Eq. (1) is modified by Eq. (3).

$$\mathbf{Q}_n(s_z, a_h) \longleftarrow (1-\alpha_n)\mathbf{Q}_{n-1}(s_z, a_h) + \alpha_n[r + \gamma \max_{a'} \mathbf{Q}_{n-1}(s_z, a')], \quad (3)$$

where $0 \leq \gamma < 1$ and α_n is defined as in Eq. (2).

3. PROPOSED METHOD

This paper proposes a new method where the agent learns the best descriptor for each image. Figure 2 shows an illustration of the proposed scheme. The most important elements of the proposed approach are detailed in this section. First, the elements defining the tuple are introduced. Then, the proposed convergence strategy is presented. Next, the exploration-exploitation trade-off to select the actions is introduced. Finally, the training stage is summarized. Note the proposed method does not follow a classical RL based scheme where an action (a_h) and a reward/punishment (r_z), for a given state, produce a new state. In our case, after applying a given action, the **Q**-table is updated but it does not result in a new state, hence a new image is considered. The different stages are detailed next.

3.1 Tuple definition

The tuple that defines our MDP is $\langle S, A, \delta, \tau \rangle$. The variables of the tuple are:

3.1.1 State definition, S

In our case, the state is a representation of the image. Many different image features can be used as a state. In our experiments, we convert the color image to grey scale image. Then, we extract mean, standard deviation and median values from that image (Fig. 3(a)). After that, the process is applied again by splitting the image into four equally sized squared blocks, and, for each sub-image, we extract again the mean, standard deviation and median values (as depicted in Fig. 3(b)). Additionally, the number of corners (Fig. 3(c)) and the number of blobs (Fig. 3(d)) using the whole grey scale image (Fig. 3(a)) are extracted. The corners are obtained using a Harris corner detector [9], while blobs are obtained converting grey scale image to black and

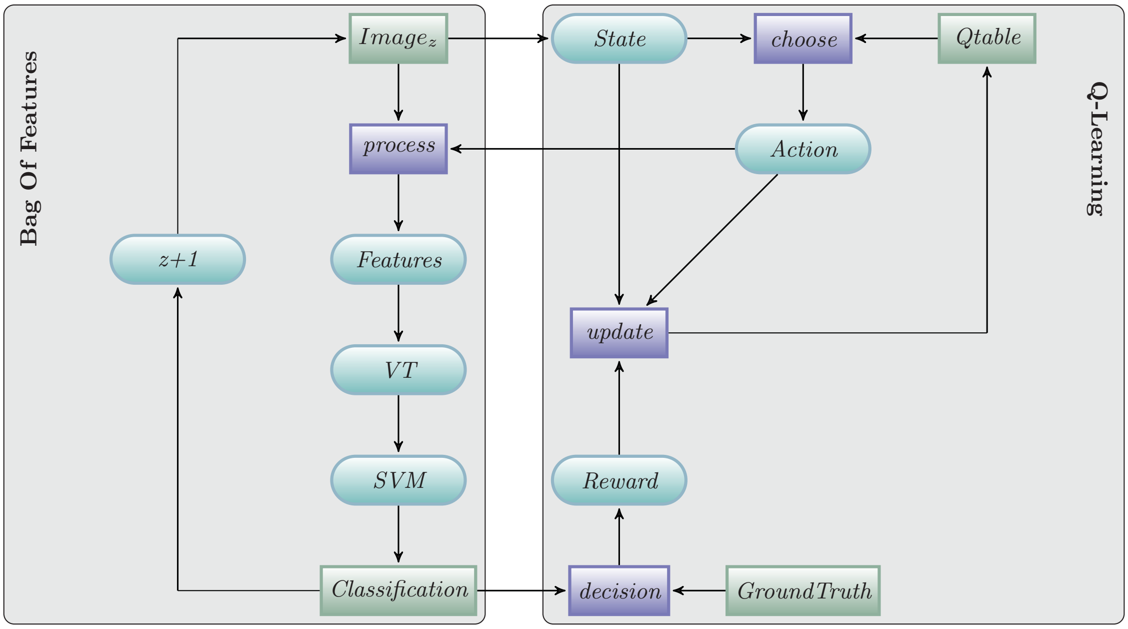

Figure 2: Illustration of the proposed scheme for image classification using Q-learning

white using OTSU threshold [18] and then, the labelling algorithm (bwlabel) with a connectivity of 8 neighbours [7] is applied. The result gives a vector of 17 dimensions with the following shape:

$\langle$ $mean_{L_1}$, std_{L_1}, $median_{L_1}$, $mean_{(1,1)L_2}$, $std_{(1,1)L_2}$, ... , $median_{(2,2)L_2}$, ncorners, nblobs $\rangle$.

This state definition is highly discriminative among images. Since the database could contain thousand of images, the size of the **Q**-table could be huge. To solve this problem, we propose a k-means clustering of the vectors and use the centroid of each cluster as a state ($S = \{s_z\}_{0<z<ncentroids}$), instead of using all the vectors' components. The size of the **Q**-table is determined by the number of clusters.

3.1.2 Action definition, A

In the current work, the actions $A = \{ a_h\}_{0<h<u}$ are descriptors, where u is the size of the descriptor set. In other words, our agent learns which descriptor gives the best information for each image. Our architecture is flexible and does not depend on a particular set of descriptors. In the current work, the descriptors used as actions are based on gradients, blobs and patterns, although other combinations could be also used. The four descriptors selected to test the proposed architecture are as follow:

- SIFT (Scale-Invariant Feature Transform): finds scale invariant regions using the magnitude of the gradient [14].
- Spin: the descriptor makes a histogram of quantized pixels locations and intensity values. This descriptor finds textures [12].
- SURF (Speeded Up Robust Feature): is based on sums of 2D Haar wavelet responses and is efficiently computed using integral images [1].

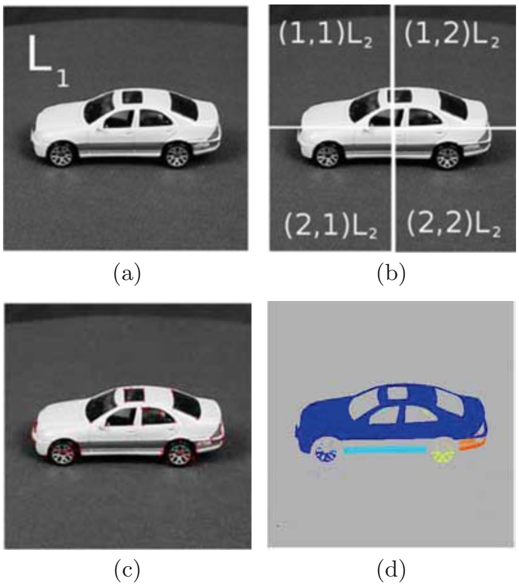

Figure 3: Illustration of an image of the database. (a) Original gray level image. (b) Image split up into four equally sized squared blocks. (c) The corners detected in the image. (d) The blobs detected in the image.

- PHOW (Pyramid Histogram Of visual Words): is a variant of dense SIFT descriptors, extracted at multiple scales [3].

3.1.3 δ function

The classical Q-learning formulation involves a δ function, which for a given state (s_z) and an action (a_h), it returns a new state ($\delta : SxA \rightarrow S$).

In our case, given an image from the given data set (I_{s_z}), the δ function does not give a new state, instead, the output is a new representation of the image (I'_{s_z}). Also, two different images at the same state (centroid) and applying the same action, usually, leads to different representation (see the example in Fig. 4). For this reason, the δ function is nondeterministic.

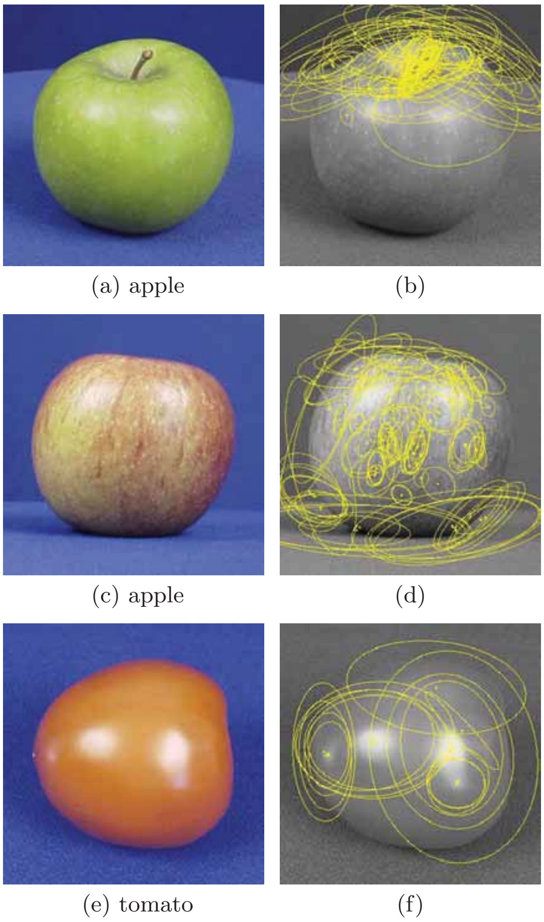
(a) apple (b)
(c) apple (d)
(e) tomato (f)

Figure 4: ($left - column$) Illustration of three images from the database. Although these images belong to different classes, their centroid lies in the same position. Hence, they are classified into the same cluster. ($right - column$) The image patches obtained applying the same action: SIFT.

Once δ function is applied the learning process continues with the classification step. The BoF uses the new representation of the image(I'_{s_z}) to classify the object through the VT and SVM. Once the object has been classified, the iteration is finished and the process continues through a new image as a new iteration.

3.1.4 τ function

The agent decides the class of the image. The τ function returns a reward when the decision of the agent matches the ground truth, and, when the decision of the agent differs, the function returns a punishment. The τ function is also a nondeterministic function. During the process, we cannot ensure that two images at the same state (s_z) and doing the same action (a_h) lead to the same $\tau(s_z, a_h)$ [16].

For example, in Fig. 4 using the action SIFT over the three images gives the same reward because the VT correctly classifies all of them. But, if we repeat the same example changing the action SIFT by SURF, the first image (Fig. 4(a)) is within the class apple but the VT returns tomato. For the second image (Fig. 4(c)), the VT hits the class. Finally, the third image (Fig. 4(e)) is a tomato but the VT returns apple. Hence, using SURF, for the image (Fig. 4(c)) results in a reward but in the other two images (Fig. 4(a) and Fig. 4(e)) the process gives a punishment. In the current implementation τ is defined as (+1000) when the image is correctly classified and (−1000) when it is wrongly classified.

3.2 Convergence

The theorem “Convergence of Q-learning for nondeterministic Markov decision processes” [16] shows that a nondeterministic MDP converges when there is a bounded reward ($\forall(s,a), |r(s,a)| \leq c$ and $0 < \alpha_n \leq 1$). Eq. (4) is true in the ith iteration when $n \longrightarrow \infty$ with probability 1.

$$\sum_{i}^{\infty} \alpha_n(i,s,a) = \infty, \qquad \sum_{i}^{\infty} [\alpha_n(i,s,a)]^2 \leq \infty. \tag{4}$$

In our framework, the agent interacts with a nondeterministic environment, so, the convergence is very expensive in time. For example, in [16] we can see that the Tesauro’s TD-GAMMON needs for training 1.5 million of backgammon games iterations and each of them contains tens of state-action transitions. As the convergence can last for weeks, we need some criteria to stop the training. Thus, in the current work it is proposed to stop the training when the following criterion is achieved:

$$\sum_{i=n-w}^{w} |\mathbf{Q}_{n+1}(s,a) - \mathbf{Q}_n(s,a)|_i < \theta, \tag{5}$$

the sliding window provides a restricted convergence. Where, w is the size of the sliding window and θ is the admitted error.

3.3 Exploration-exploitation trade-off

This section presents the action selection; in general, it is referred to as exploration-exploitation trade-off. If the agent uses only the exploration strategy it could fall into a local maximum. To avoid this problem, the RL learns some steps with an exploitation strategy.

An ε-greedy scheme is generally used as an exploitation strategy. In this paper, we propose a method to compute ε adaptively. We propose the measurement of the error as the parameter for switching the strategy. We define the error as $e = f(it)$, where f is defined as: $f(it) = |Q_{it} - Q_{it-1}|$; and, for each iteration we store this error. The ideal process reduces the error for each iteration, but sometimes the error increases (see Fig. 5).

We propose to calculate e_{it} and use this value as a switching indicator: $e_{it} - e_{it-1} > threshold$. However, to avoid switches when the error is a small spike, we propose to consider also the error in the neighborhood of current iteration.

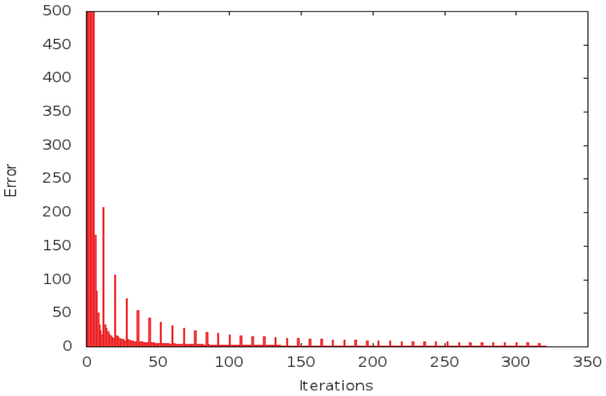

Figure 5: Behavior of e_{it}, till convergence is reached, for different iterations.

Hence, a sliding windows (w') is used as indicated in Eq. (6). In other words, if the following condition is fulfilled current strategy is switched to exploitation trade-off.

$$\sum_{i=0}^{w'} e_{it-i} > \sum_{i=1}^{w'+1} e_{it-i}, \qquad (6)$$

where, like in Eq. (5), w' is the size of the sliding windows (in the current implementation $w' = 5$).

3.4 Training

In order to train and test our approach, we have considered an image database and a set of descriptors widely used in the computer vision literature [1, 3, 12, 14]. The image database is split up into three sets: vocabulary tree training set (VTTS), **Q**-table training set (QTTS), and testing set.

The first training is using the VTTS, a tree is built for each descriptor (T_{a_h}) using a BoF approach with VT and SVM [24].

The second training starts initializing the **Q**-table and the method is depicted in Fig. 2. Given an image from QTTS, the agent extracts the state (s_z) and using the **Q**-table and the exploration-exploitation strategy decides the action (a_h). The agent extracts the features using the descriptor (a_h) and classifies the image into one class. To do the classification, we use the pair (T_{a_h}, a_h). When the agent obtains a class, the agent compares this class with the ground truth and obtains the reward/punishment r_t through $\tau(s_z, a_h)$. Finally, the agent computes the convergence (Eq. (5)) and then, the agent updates the **Q**-table using Eq. (3).

4. RESULTS

The ETH database has been used to evaluate the proposed approach. Nine classes from that database have been selected: apple, car, cow, cowcup, cup, dog, horse, pear and tomato. Figure 6 shows some images of the database.

We have used 45 images per class, which were split into three sets: 15 images for training VT, 15 images for training the **Q**-table and finally, 15 images for testing. We have repeated the experiments fifteen times, using this three image sets. The process of testing starts when the **Q**-table achieves the convergence (Eq. (5), with $\theta = 0.4$).

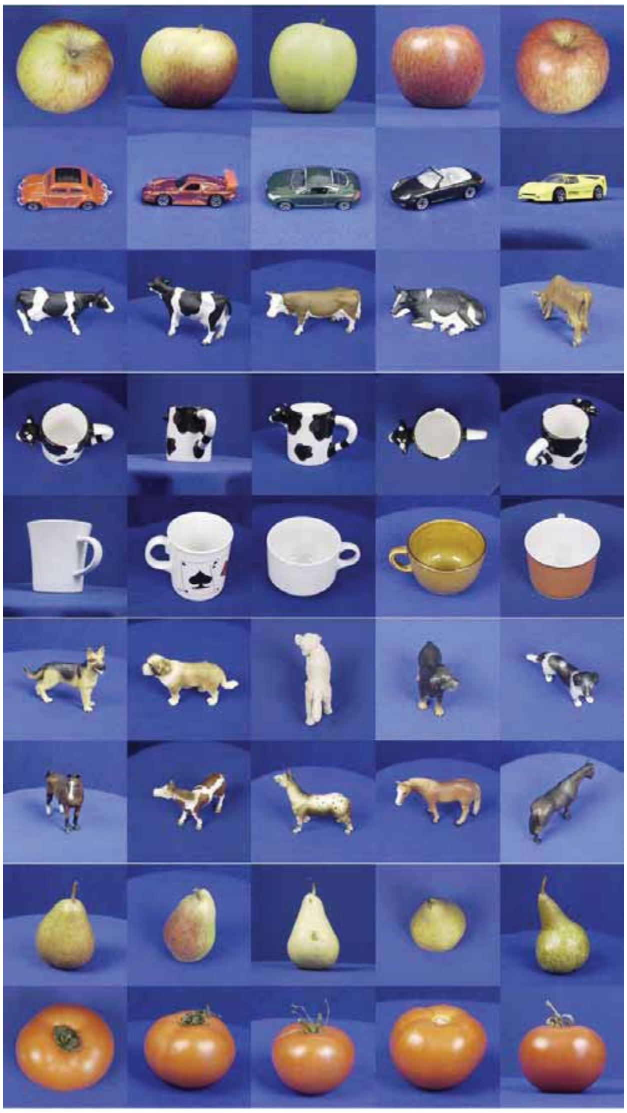

Figure 6: Illustration from ETH database

Given a testing image, the process extracts the state and selects the action. To select the best action, the process searches the maximum value in the **Q**-table for this state. But sometimes, the **Q**-table does not have an unique maximum, to solve this problem, we introduce a vector with weights. The vector of weights is multiplied by the **Q**-table to obtain more distance between the actions.

In order to compare the results, we have measured the performance of each action (Table 1). Firstly, using as an action always the same descriptor. Secondly, the descriptor resulting from Q-learning is used. Note that other strategies could be considered for the comparisons, for instance considering as an action all the descriptors at once. However, this kind of strategy will introduce noise as well as will increase the CPU time.

The best result using a single descriptor as an action for the ETH database is PHOW with a performance of 74.81%. The proposed Q-learning scheme has been evaluated using

Table 1: Performance for each action and using the Q-table (Performance: percentage of success during the classification)

Action	Performance
Spin	60.00%
SIFT	61.48%
SURF	62.96%
PHOW	74.81%
Q-learning	81.48%

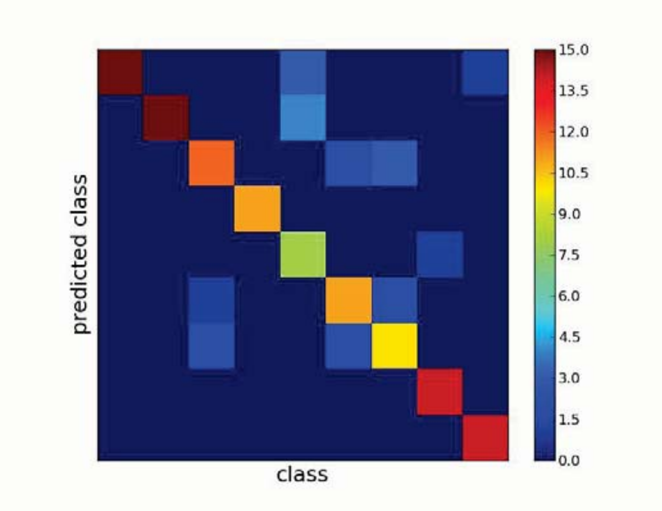

Figure 7: Confusion matrix with $\varepsilon = 0.5$ exploration-exploitation strategy; it achieves a performance of 81.48%

two different exploration-exploitation strategies. Firstly, an exploration-exploitation trade-off with $\varepsilon = 0.5$ has been used. It reaches a performance of 81.48%. The value ε has been set empirically; actually, other values have been tested (e.g., 0.2) reaching the similar results but increasing the number of iterations and consequently the CPU time. Figure 7 shows the confusion matrix.

The second experiment was done with the exploration-exploitation strategy presented in Section 3.3; in this case we also reach the same performance (81.48%). However, it should be noticed that the proposed approach converges in less iterations as depicted in Table 2, which shows the number of iterations needed for each exploration-exploitation strategy. The usual strategy needs more than 40.000 iterations to arrive at the same convergence state.

Table 2: Number of iterations for each strategy using $\theta = 0.4$

Strategy	Number of iterations
$\varepsilon = 0.5$	168.419
History of the error	127.710

5. CONCLUSIONS AND FUTURE WORK

In this paper a novel method to learn the best descriptor for each image in a database has been presented. A new architecture joining the Reinforcement Learning and Bag of Features is proposed. Additionally, a new exploration-exploitation strategy is introduced. The proposed approach has been validated using the ETH database. Its performance has been compared with respect to a single descriptor scheme. The best descriptor for the ETH database is PHOW with 74.81% of performance, while the proposed method reaches 81.48%. Therefore, the results are improved in almost 7% using the proposed method. Additionally, a strategy for exploration-exploitation is proposed that makes the convergence faster than using random switches.

Future work will be focused on the exploration of other state definition, and also, on the increase of the set of descriptors (e.g., H-mat, GIST, Centrist, etc. are some of the descriptors to be considered). The proposed method will be tested with databases containing different backgrounds. Finally, the possibility of selecting the actions without random values will be studied.

6. ACKNOWLEDGMENTS

This work was partially supported by the Spanish Government under Research Program Consolider Ingenio 2010: MIPRCV (CSD2007-00018) and Project TIN2011-25606. Monica Piñol was supported by Universitat Autònoma de Barcelona grant PIF 471-01-8/09.

7. REFERENCES

[1] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *Proc. the European Conference on Computer Vision*, pages 404–417, 2006.

[2] R. Bianchi, A. Ramisa, and R. de Mántaras. Automatic Selection of Object Recognition Methods using Reinforcement Learning. *Advances in Machine Learning I*, pages 421–439, 2010.

[3] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. *Proc. International Conference on Computer Vision*, 2007.

[4] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *In Workshop on Statistical Learning in Computer Vision, Proc. the European Conference on Computer Vision*, pages 1–22, 2004.

[5] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 524–531, 2005.

[6] K. Häming and G. Peters. Learning scan paths for object recognition with relational reinforcement learning. *Signal Processing, Pattern Recognition and Applications*, 2010.

[7] R. Haralick and L. Shapiro. *Computer and robot vision*, volume 1. Addison-Wesley, 1992.

[8] M. T. Harandi, M. N. Ahmadabadi, and B. N. Araabi. Face recognition using reinforcement learning. *Proc. IEEE International Conference on Image Processing*, IV:2709–2712, 2004.

[9] C. Harris and M. J. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, -:147 – 152, 1988.

[10] S. Jodogne. Reinforcement learning of perceptual classes using q learning updates. *Proc. of the 23rd IASTED International Multi-Conference on Artificial Intelligence and Applications*, pages 445–450, 2005.

[11] S. Jodogne and J. H. Piater. Interactive selection of visual features through reinforcement learning. *24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 285–298, 2004.

[12] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[13] T. Leopold, G. Kern-Isberner, and G. Peters. Belief revision with reinforcement learning for interactive object recognition. *Proc. the European Conference on Artificial Intelligence*, pages 65–69, 2008.

[14] D. Lowe. Distinctive image features from scale invariant keypoints. *International Journal on Computer Vision* , 2:91–110, 2004.

[15] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[16] T. M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[17] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2:2161 – 2168, 2006.

[18] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:62–69, 1979.

[19] J. Peng, , J. Peng, and B. Bhanu. Local reinforcement learning for object recognition. *Pattern Recognition* , 1:272–274, 1998.

[20] S. Russell, P. Norvig, J. Canny, J. Malik, and D. Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, NJ, 1995.

[21] F. Sahba, H. R. Tizhoosh, and M. Salama. Aplication of opposition-based reinforcement learning in image segmentation. *IEEE Computational Intelligence in Image and Signal Processing*, pages 246 – 251, 2007.

[22] M. Shokri and H. R.Tizhoosh. A reinforcement agent for threshold fusion. *Applied Soft Computing*, 8:174 – 181, 2008.

[23] R. Sutton and A. Barto. *Reinforcement learning: An introduction*, volume 28. Cambridge Univ Press, 1998.

[24] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms, 2008. http://www.vlfeat.org/.

[25] C. J. C. H. Watkins. *Learning from delayed rewards*. Ph.D. thesis, King's College, Cambridge UK, 1989.