Reinforcement Learning of Visual Descriptors for Object Recognition

A dissertation submitted by **Mónica Piñol Naranjo** at Universitat Autònoma de Barcelona to fulfil the degree of **PhD in Computer Science**.

Bellaterra, May 7, 2014

Director: **Ricardo Toledo Morales**
Universitat Autònoma de Barcelona
Dept. Informàtica & Computer Vision Center

Co-director: **Angel D. Sappa**
Computer Vision Center

This document was typeset by the author using $\LaTeX\,2_\varepsilon$.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona.

ISBN 978-84-940902-5-7

Printed by Ediciones Gráficas Rey, S.L.

A la meva família

Acknowledgements

Todo empezó un día que me decidí a entrar en el despacho de Ricardo Toledo para pedirle el proyecto de final de carrera, solo esperaba tener un buen proyecto y acabar mi carrera. Cuando pienso en la carrera solo recuerdo que fue larga, difícil, decepcionante y solo pensaba en acabar y escapar de la ETSE. Aunque para ser sincera encontré grandes amigos y aprendí mucho. Todavía no había acabado el proyecto final de carrera cuando me preguntó que pensaba sobre seguir estudiando, pensé que estaba loco!. Poco después y hablando con él, me di cuenta que sí que me apetecía seguir y que mi estancia en la universidad no acababa sino que empezar una nueva carrera de obstáculos.

Durante mi carrera conocí a mucha gente, pero quiero recordar aquí a las personas más especiales y que todavía, a día de hoy, los tengo en mi vida. Leandro por estar ahí, por todas esas prácticas que hicimos juntos. Juby por estar siempre a mi lado y dar buenos consejos aunque no siempre te he hecho caso. Juan por que aunque tienes un sentido del humor algo raro me caes genial y sobre todo porque me animaste a presentar la tesina cuando no pensaba hacerlo. Alberto, Raúl, Jordi, Esther gracias por nuestras charlas.

Siguiendo con la línea natural del tiempo llegue al CVC donde conocí a mucha gente. Recuerdo el primer día en el CVC sin mesa, ni silla y sin saber donde colocarme así que gracias a D. Aldavert por dejarme tu sitio mientras estabas fuera. D. Gerónimo que te ofreciste a ayudarme en todo y me conseguiste un alargo para el PC (siempre supe que estabas ahí). Gracias a todos los que me habéis acompañado todos estos años, Jorge, Camp, Ivan, David, Jon, Lluis, Toni. Gracias al grupo ADAS por su apoyo y confianza. Y gracias a todas las personas que han pasado por el CVC que de una manera u otra nos ayudan cada día.

También quiero dar las gracias y ahora mucho más personal a mis amigos y familia, por su apoyo incondicional. Nunca imaginé que a los 29 años tendría amigas de cuando empecé el instituto, así que un especial gracias por apoyarme, por ayudarme y alguna vez criticarme Laura, Cristina y Blanca, gracias. A mi padre y a mi madre por que son los que más habéis sufrido mi tesis, con mis neuras y mis problemas. A mis primos Ivan y Miriam, Ferrant y Gemma, Nuria, Esther y Javi, por que siempre os preocupáis por mi. A mis tíos Pilar, Mari, Enrique, Paqui, Nuri, Enric. De entre ellos un especial gracias a mi tío Enric que siempre a sido mi ejemplo a seguir.

Para finalizar, y ahora más formal, quiero dar las gracias a Ricardo Toledo y Angel D. Sappa por apoyarme, ayudarme y sobretodo guiarme durante esta tesis. Al grupo ADAS y sus reading groups que me dieron ideas. Y finalmente, a la UAB por la beca

PIF, con la cual he podido realizar este trabajo de tesis.

Abstract

The human visual system is able to recognize the object in an image even if the object is partially occluded, from various points of view, in different colors, or with independence of the distance to the object. To do this, the eye obtains an image and extracts features that are sent to the brain, and then, in the brain the object is recognized. In computer vision, the object recognition branch tries to learns from the human visual system behaviour to achieve its goal. Hence, an algorithm is used to identify representative features of the scene (detection), then another algorithm is used to describe these points (descriptor) and finally the extracted information is used for classifying the object in the scene. The selection of this set of algorithms is a very complicated task and thus, a very active research field. In this thesis we are focused on the selection/learning of the best descriptor for a given image. In the state of the art there are several descriptors but we do not know how to choose the best descriptor because depends on scenes that we will use (dataset) and the algorithm chosen to do the classification. We propose a framework based on reinforcement learning and bag of features to choose the best descriptor according to the given image. The system can analyse the behaviour of different learning algorithms and descriptor sets. Furthermore the proposed framework for improving the classification/recognition ratio can be used with minor changes in other computer vision fields, such as video retrieval.

Resum

El sistema visual humà és capaç de reconéixe l'objecte que hi ha en una imatge encara que l'objecte estigui parcialment oclòs, des de diferents punts de vista, en diferents colors i amb independència de la distància a la que es troba l'objecte de la càmera. Per poder realitzar això, l'ull obté l'imatge i extreu unes caracterítiques que són enviades al cervell i és allà on es classifica l'objecte per poder identificar-lo. En el reconeixement d'objectes, la visió per computador intenta imitar el sistema humà. Així, s'utilitza un algoritme per detectar característiques representatives de l'escena (detector), un altre algoritme per descriure les característiques extretes (descriptor) i finalment la informació es enviada a un tercer algoritme per fer la classificació (aprenentatge). Escollir aquests algoritmes és molt complicat i tant mateix una àrea d'investigació molt activa. En aquesta tesis ens hem enfocat en la selecció/aprenentatge del millor descriptor per a cada imatge. A l'actualitat hi ha molts descriptors a l'estat de l'art però no sabem quin es el millor, ja que no depèn sols d'ell mateix sinó també depen de les característiques de les imatges (base de dades) i dels algoritmes de classificació. Nosaltres proposem un marc de treball basat en l'aprenentatge per reforç i la bossa de característiques per poder escollir el millor descriptor per a cada imatge. El sistema permet analitzar el comportament de diferents classiicadors i conjunts de descriptors. A més el sistema que proposem per a la millora del reconeixement/classificació pot ser utilizat en altres àmbits de la visió per computador, com per exemple el *video retrieval*

Resumen

El sistema visual humano es capaz de reconocer el objeto que hay en una imagen aún cuando el objeto está parcialmente ocluido, desde varios puntos de vista, en diferentes colores, con independencia de la distancia a la que se encuentre el objeto de la cámara. Para ello, el ojo obtiene la imagen y extrae unas características que son enviadas al cerebro, y es allí donde se clasifica el objeto para poder identificarlo. En el reconocimiento de objetos la visión por computador intenta imitar el sistema humano para obtener la identificación de los objetos. Para ello se utiliza un algoritmo para detectar características representativas de la escena (detector), otro para describir dicha característica descriptiva (descriptor) y finalmente la información extraída se envía a un tercer algoritmo que nos dirá que objeto hay en la imagen (aprendizaje). Elegir este conjunto de algoritmos es una tarea muy complicada y por lo cual es una área de investigación muy activa. En esta tesis nos enfocamos en la selección (aprendizaje) del mejor descriptor para una imagen dada. En la actualidad hay muchos descriptores en el estado del arte pero no sabemos escoger el mejor, ya que no depende solo del descriptor en si mismo sino también de las escenas que va a utilizar (base de datos) y del algoritmo de classificación escogido (aprendizaje). Nosotros proponemos un marco de trabajo basado en aprendizaje por reforzamiento y bolsa de características para poder escoger el mejor descriptor según la imagen dada. El sistema permite analizar el comportamiento de diferentes classificadores y conjuntos de descriptores. Además el sistema que proponemos para la mejora de la classificación/reconocimiento puede ser utilizado con pocas modificaciones en otros ámbitos de la visión por computador, como por ejemplo el *video retrieval*.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In computer vision, the pattern recognition problem has been largely studied during the last years. It aims at an automatic identification of the objects in a given scene. Such a simple and effortless task for the human visual system is even today a difficult and challenging problem for the computer vision domain. Actually, during the last three decades a lot of research effort has been devoted to identify the best pipeline to solve this task. A common one is based on firstly the usage of distinctive elements that could be used to describe in a compact and abstract way a given object and secondly the usage of some learning technique that classify the objects according with their descriptors. Unfortunately, in spite of the large effort there is not yet a dominant and accepted set of algorithms to be used in all the computer vision applications for this basic pipeline: *i*) detection, *ii*) description and *iii*) learning. Every year there are new proposals in the literature that overpass previous ones. Usually, these improvements are related with the increase of the computation capabilities but not with the theory behind the proposals.

The *detectors* find distinctive features in the image, such as: corners, edges, blobs, among others. Different approaches have been proposed in the literature, some of them inspired by the human visual perception system are based on spatial-frequency response, other are more related with the digital representation of the image. The most widely used approaches take *key points* (also referred to as *interest points*) as representative features. These key point features are later on described according to the selected algorithm and using representative information of their neighbourhood.

The second element of the basic pipeline introduced before is related with the description of the distinctive features. In the particular case of having an interest point as a distinctive feature, a *descriptor* generates a representation of this interest point in another space based on its surrounding area. Each of the descriptors proposed in the literature have their own properties; for instance: *i*) easy to compute; *ii*) compactness; and *iii*) robustness to changes in scale, rotation, lighting. All the characteristics make the development of algorithms for feature description a challenging and appealing research domain.

Finally, the *learning* is the third element of the pipeline, which performs the recognition. This process can be implemented through a supervised, an unsupervised or a combination of both techniques. The supervised technique is based on labels, hence each image has a label that is used by the learning algorithm to find the boundaries of the classes being recognized. On the contrary, the unsupervised technique does not have labels, thus, the algorithm finds the boundaries using only the data. Finally, hybrid approaches have been also proposed. These approaches are a combination of supervised and unsupervised techniques in different steps.

During the last decade a large amount of detectors, descriptors and learning algorithms have been proposed in the literature, becoming a very active research field. The performance depends on the selected combination, which at the end is also related with the dataset used for validation. In summary, the question is to select a detector/descriptor/learning for a given dataset; actually, we should select the best combination. This problem becomes difficult to tackle since all the possible combinations should be evaluated. Hence, in general, most of the works are focused on finding the best descriptor for a given detector and learning technique. In the next paragraphs we present examples of contributions, which according to the authors, are the best solution for the given datasets:

> "We demonstrated that visual terrain classification can be performed accurately and precisely using JCD descriptor with ELM classifier" [106]. The authors use two datasets to support their statement, the first dataset was presented in [28] and the other one was DARPA LAGR dataset [76].
>
> "These experiments demonstrate the improved accuracy of GF-HOG over other state of the descriptors art across a multitude of distance measures and affine variations" [36]. The authors use a large Flickr sourced dataset comprising 33 shape categories.
>
> "HMAX performs better than SIFT in all the different experiments" [61] to the Caltech-101 dataset, but it only uses 9 classes and plus the Google things dataset [27] as negative examples.
>
> "Hessian-affine and SIFT form the best combination for object classification method" [47]. The experimental results were extracted using Caltech-101 dataset.
>
> "The MSER and Difference of Gaussian (DoG) detectors with a SIFT or DAISY descriptor are the top performers" [19]. The authors use the DTU Robot dataset.

The five examples above show that, for a concrete setting (i.e., detector, learning dataset), exists a descriptor that reaches the best performance. In all the cases, this "best descriptor" has been found after a trial and error process. However, such a kind

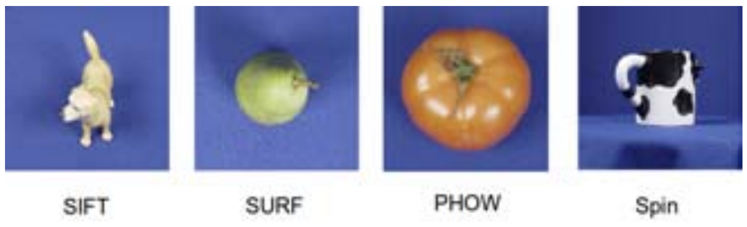


Figure 1.1: Example of four images from the same dataset that are only well classify with a concrete descriptor.

of process can not be usually followed, in particular when big datasets are considered. Trial-and-error process are implemented since on the one hand there is no information about the behaviour of the descriptor for a given dataset; on the other hand we do not have information about the characteristics of the dataset that will help the selection of the descriptor; hence the only option is to use this kind of naïve selection approach. The work in this thesis is focused on this problem by setting as learning technique the *bag of features* approach and using as the detector the one required by the descriptor being evaluated.

More in details, we propose a novel approach that allows the selection of the best descriptor for a given image. In order to do it, a novel framework that merges the classification technique with a learning process is proposed. It is based on the combination of *bag of features* and *reinforcement learning*. These techniques have been chosen due to their flexibility and robustness to be adapted to different domains. The strategy behind the proposed scheme allows to test different descriptors and to select the best one according to the recognition result. This process is performed for every image ensuring results better than the state of the art, which are based on classes but not on elements.

1.1 Motivation & Contributions

This thesis is focused in descriptor selection. As mentioned above, each dataset achieves the maximum performance with a concrete descriptor. This descriptor is generally found in a trial-error process. Unfortunately, after the trial-error process, it finds the “best descriptor”, which does not achieve the 100% of recognition/classification ratio. There are images that are bad classified with the “best descriptor”. On the contrary, the wrongly classified images can be correctly classified with another descriptor. In Fig. 1.1 we can see four images from the ETH dataset where each image is correctly classified just with a concrete descriptor.

Then, we should not search the “best descriptor” for a concrete database but change the objective and search the “best descriptor” for each image. For this reason, this thesis is focused on searching for techniques to select the descriptor given a set of descriptors. We define a set of u descriptors as:

$$D = \{d_0, d_1, ..., d_u\}. \qquad (1.1)$$

Our proposal is to find a function (f) that given an image returns the "best descriptor" as:

$$f : image \longrightarrow d_i. \qquad (1.2)$$

This function maps the image for all descriptors and returns the descriptor that maximize the function. In order to do the mapping, we need to re-define the function f as:

$$f(g(image)) \longrightarrow d_i. \qquad (1.3)$$

where g is a function, which extracts the characteristics of the image to do the mapping. The g function should be faster and easy to compute. This function resume the image in a simple vector (c_{image}). This vector c_{image} introduces more dimensions at the problem but less than a dimensionality of the descriptor. Thus, the most important is to define the g with the characteristics mentioned above.

1.1.1 Objectives

The main goals in this thesis are two:

- One point in this thesis is to maximize the recognition ratio for any database. We are focused on feature extraction, and in concrete, we work to find the best descriptor to maximize the recognition ratio. As mentioned before, each descriptor has a percentage of recognition ratio different and each one has different well/wrong classified images. Thus, we understand that the solution is to use a set of descriptors, but not concatenating all the descriptors in the set because this strategy usually introduces noise. Hence, we need to select the descriptor for each image using a learning process. Then, we propose a framework using the *bag of features* and *reinforcement learning.*
- The second point in this thesis is to extrapolate the idea presented above for other computer vision fields. We propose to apply the framework for object recognition, Chapters 4 and 5, and then, with a video retrieval field in Chapter 6.

1.1.2 Contributions

This section summarizes the contributions in this thesis.

- The first approximation to find the f function is assuming that the f is linear. This hypothesis is explored in Chapter 3 and shows that f function can not be linear function, in other words, we need a learning process to find the boundaries to select the descriptors.

- We propose a framework that uses the bag of features to obtain the classification and the reinforcement learning to do the learning process to select the descriptor for each image.
- In a reinforcement learning technique with a non-deterministic environment, the convergence is assured but does not exist a criterion to stop the execution. In our experiments we need to stop the execution with a criteria, for this reason we propose a formula to determine the convergence.
- We have demonstrate that depending on the g function, the framework of bag of features and reinforcement learning modifies the recognition ratio. Thus, the most challenge in this framework is defining the g function. We propose to use a set of g functions and then, using for each image one of them selected by voting.
- The framework proposed in this thesis can be used for other computer vision tasks, with minor modifications. As an example, we show the improvement of the recognition ratio of the approach in video retrieval.
- The reinforcement learning technique can be used with a human teacher. This approach is known as apprenticeship technique. We introduce the apprenticeship in our framework to improve the classification ratio and minimize the execution time.

1.2 Thesis outline

The reminder of the thesis is organized as follows. Chapter 2 summarizes the state of the art in the two fields addressed in this thesis. First, it presents a set of descriptors and then, in the second part, it explains the reinforcement learning technique and it presents some applications in computer vision. In Chapter 3, the necessity of the reinforcement learning to find the correspondence between the characteristics of the image and the selected descriptor are presented. Then, Chapter 4 and 5 propose a new framework of bag of features with reinforcement learning. Chapter 4 proposes a “simple” framework and then, Chapter 5 improves it with a multi-Qtables. Next, Chapter 6 shows the apprenticeship technique in video retrieval using the MIPRCV-WP6 Video Retrieval Benchmark dataset. Finally, the summary for each chapter and the future research are presented in Chapter 7.

Chapter 2

Related work

This chapter summarizes the states of the art of two different topics inasmuch as this thesis introduces a framework that joints the visual object recognition and the reinforcement learning in order to improve the recognition ratio.

Hence, this chapter first introduces an overview of visual object recognition in Sect. 2.1. In this field one of the most used approaches is the bag of features, which consists of four steps: *i*) feature detection and description; *ii*) dictionary generation; *iii*) image representation; and *iv*) learning process. This thesis is focused in the first step (feature detection and description) and for this reason, the sections 2.1.1 and 2.1.2 are focused in the state of the art of detection and description of the interest points.

As mentioned in the previous chapter, this thesis proposes a novel framework to maximize the recognition ratio. This framework is based on the selection of the best pair detector/descriptor using an artificial intelligence scheme – i.e., the reinforcement learning technique (Sect. 2.2). The reinforcement learning is a technique based on trial-error to learn how to take the actions. In order to select the actions, the technique maximizes the expected reward. Usually, the reinforcement learning technique is formulated as a Markov decision process and solved using dynamic programming.

The remainder of this chapter is organized as follows, Sect. 2.1 introduces the visual object recognition using the bag of features approach. Then, Sect. 2.1.1 and Sect. 2.1.2 summarize the most representative works in detection and description of feature points. Next, Sect. 2.2 introduces the second topic of this thesis, Sect. 2.2.1 summarizes the Markov decision process, the definition and the characteristics. Then, in Sect. 2.2.2 the reinforcement learning technique is introduced. Finally, Sect. 2.2.3 summarizes different applications of computer vision based on the use of reinforcement learning.

2.1 Visual object recognition

Object and scene recognition is one of the most active research fields of the computer vision community. In general, the algorithms proposed in this research field contain several parameters that need to be tuned according to the characteristics of the image to be processed (e.g., light, scale, rotation, occlusions, shadows, etc.).

Popular approaches for object recognition consist of two steps. The first step is focussed on the extraction of features from the given image, which are later on used for training a model to classify the image. These features are computed over interest point of the image. Depending how the feature space is defined we can obtain different invariant points; for this reason, each feature space contains different interest points. In order to transform the image space to feature space we use *detectors* and *descriptors*. The detectors are algorithms that given an image return a set of interest points and then, the descriptors uses the set of these interest points to describe the area around each one of them. There are different detectors and descriptors, actually, during the development of this thesis there were authors working on new approaches. This section summarizes the state of the art on the set of detectors and descriptors that have been used in this thesis. This set of detectors and descriptors was selected because they have the best performance; this explain why they are the most used on the state of the art (e.g., [59], [97], [35], [54], [3], [49], [11]).

The second step of the visual object recognition is the classification. There are different methods to train a model for classifying the content of an image: bag of features, support vector machine, principal component analysis, AdaBoost, etc. Among them, a widely used method is bag of features, which is the one used in this thesis (e.g., [26], [18] and [66]). The bag of features has been proposed for document analysis (e.g., [84] and [58]) and then the method was extended for images to classify textures (e.g., [105] and [16]), action recognition (e.g., [48] and [63]) or objects classification. The training of the bag of features is done in four steps as depicted in Fig. 2.1.

The first step consists in extracting and describing the interest points, which is fundamental for the development of this thesis. Thus, the set of detectors and descriptors that we use in this work are explained in the following sections (2.1.1 and 2.1.2).

Then, the second step is to make a codebook of the interest points. This step is solved by a clustering algorithm such as: k-means [8], k-d tree [85], vocabulary tree [64], etc. All these techniques are unsupervised learning methods. In the current work, we use the k-d tree algorithm that consists in splitting the space in various sub-spaces and obtaining a balanced tree with k-dimensions.

Next, in the third step a histogram is generated. It represents the number of times that an interest point appears in the image using the codebook constructed in the previous step and the interest points extracted in the first step.

Finally, the last step consists of a learning step to do the object classification. In this case, the learning technique is supervised using the ground truth to obtain the labels. This is a complex problem with high dimensional features and multi-class.

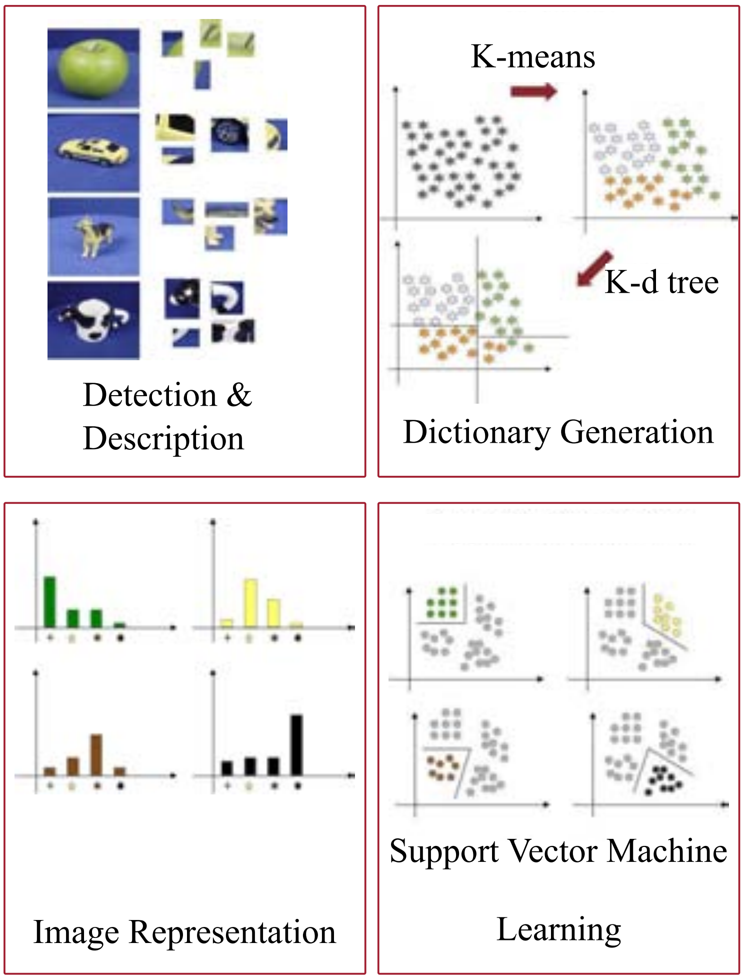


Figure 2.1: Illustration of the training process of the bag of features approach.

Thus, this work uses a support vector machine.

Once the training process has finished, the bag of features scheme can be used for classifying the contents of a given image. This process is referred to as the testing process and it is depicted in Fig. 2.2. In the testing process the first step is the same as in the training process; first, it extracts and describes the interest points. Then, using the dictionary generated during the training process a histogram is obtained like the one obtained in the third step of the training process. Finally, the support vector machine from the last step of the training process is used to obtain the classification.

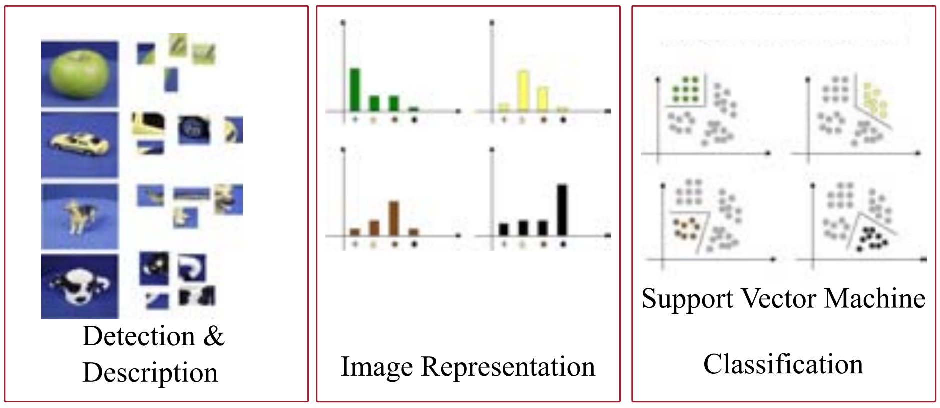


Figure 2.2: Illustration of the testing process of the bag of features approach.

2.1.1 Feature point detection

The detection step search for points that are invariant to some characteristics of the image: rotation, scale, shadows, etc. This section introduces some of the different techniques proposed in the literature to find interest points: Harris corner detector, scale-invariant feature detection and fast Hessian detector.

Harris corner detector

One of the most popular detector is Harris corner detector that was introduced in 1988 [35]. Corners are obtained as the intersection of two or more lines or surfaces, Fig. 2.3 shows some illustration of corners.

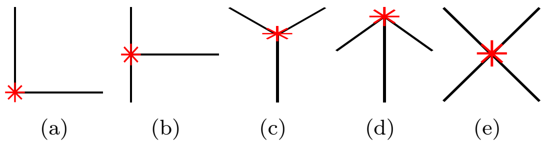


Figure 2.3: Example of corners detected with Harris algorithm.

The Harris corner detector is based on the estimation of the gradient in a local patch. Let's I be the given image, the mathematics definition of the detector is:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2, \tag{2.1}$$

where $w(x,y)$ is the window, which usually is 1. The second part of the equation is the difference of the intensity in a pixel with the shifted one. The detector searches for large difference to find large changes of intensities, so, finds corners.

To calculate the difference we can apply an approximation of the first order of Taylor's series in the shifted intensity:

$$I(x+u, y+v) = I(x,y) + uI_x(x,y) + vI_y(x,y), \tag{2.2}$$

where I_x and I_y correspond to the $\frac{dI}{dx}$ and $\frac{dI}{dy}$; now, if we replace the Taylor's approximation in (2.1), we obtain:

$$E(u,v) = \sum_{x,y} w(x,y)(u^2 I_x^2(x,y) + 2uvI_{x,y}(x,y) + v^2 I_y^2(x,y)), \tag{2.3}$$

this expression can be written as a matrix:

$$E(u,v) = (u,v) \sum_{x,y} w(x,y) M \begin{pmatrix} u \\ v \end{pmatrix}, \tag{2.4}$$

where:

$$M = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}. \tag{2.5}$$

The eigenvalues (λ_1 and λ_2) of this matrix are used to discern whether a point is a corner or not. When $\lambda_2 >> \lambda_1$ or on the contrary $\lambda_1 >> \lambda_2$ the area is an edge. If $\lambda_1 \approx \lambda_2$ and both are large the method returns that the point is a corner (as can be seen in Fig. 2.4).

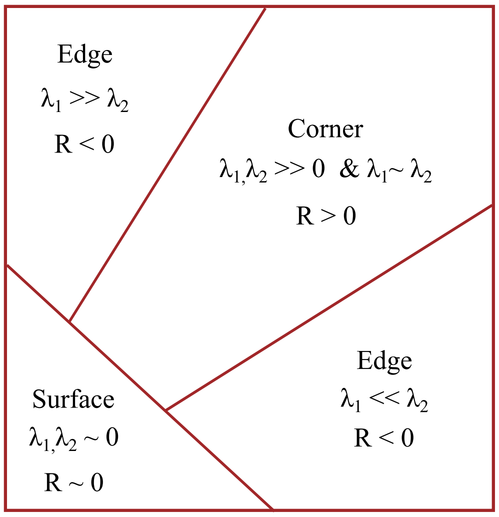


Figure 2.4: Criteria used to define whether a point is a corner or not according to λ_1 and λ_2 values; or according to the value of R.

Actually, the Harris corner detector does not use these criteria, because the eigenvalue computation can be avoided by computing the response function:

$$R(x, y) = det(M(x, y)) - ktrace^2(M(x, y)), \tag{2.6}$$

where $det(M(x, y))$ determines the local structure:

$$det(M(x, y)) = \lambda_1 \lambda_2, \tag{2.7}$$

and the $trace(M(x, y))$ is the trace of the matrix, sum of elements on the main diagonal multiplied by a constant k in the range of $[0.04 - 0.15]$:

$$trace(M(x, y)) = \lambda_1 + \lambda_2. \tag{2.8}$$

Then, the new criteria to find corners depends on the value of the response function R. Hence, the point is considered a corner if the value of $R_{x,y}$ is bigger than a given threshold and there are not greater values in a 3x3 neighborhood (8 neighbors) window in the matrix. So, a high threshold detects only the strong corners and a low threshold detects false positive corners as depicted in the illustration Fig. 2.5(d). Fig. 2.5(b) depicts the basic idea of the process. One of these windows detects flat, another detects a line and the others two detect corners. The third image (Fig. 2.5(c)) shows the expected interest points to be detected; while the last image shows the real points detected by the algorithm (Fig. 2.5(d)).

The quality of the Harris corner detector depends on the threshold. The ideal threshold is a trade-off between detected false positive corners and false negative corners. Also, the threshold depends on the quality of the image, the brightness, the noise, etc.

Scale-invariant feature detection

The scale-invariant feature detector proposes the usage of Difference of Gaussians in order to solve the problem that appears when the image is taken at different scales. This edge detector apply two smoothing filters in an image, each time with different value of σ. Then, a new image is obtained from the difference of the filtered images. The applied filters are low-pass filters, so, these filters decrease the high frequencies of the image:

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.9}$$

Fig. 2.6 shows an illustration of edges obtained from a DoG.

The difference of Gaussians finds the edges of the image as introduced in [56], but, in this section we are focussed on the extraction of feature points, thus, the scale-invariant feature detection does the process to transform the edge information

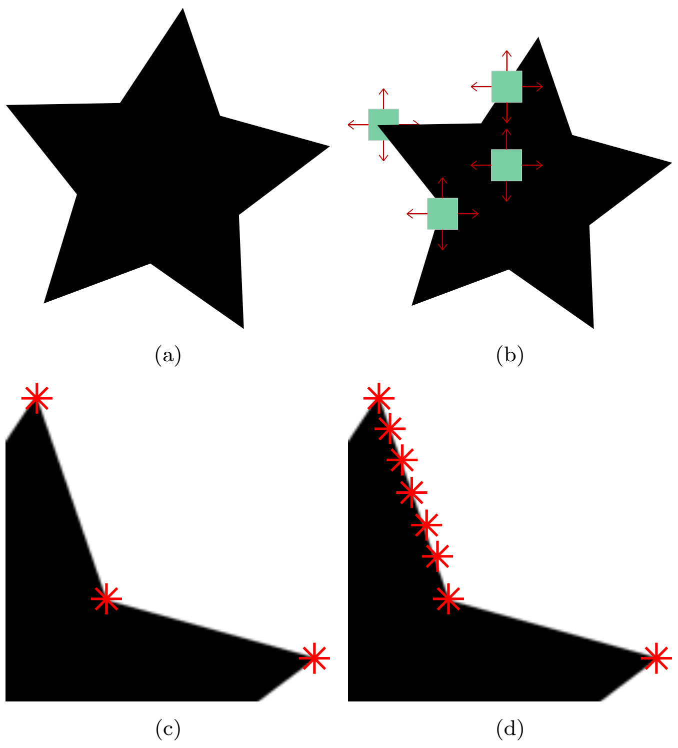


Figure 2.5: Illustration of Harris corner detector algorithm. (a) Original image. (b) Basic idea of the process to find the interested points using Harris corner detector. (c) The expected points after applying the Harris corner detector. (d) The real points obtained after applying the Harris corner detector.

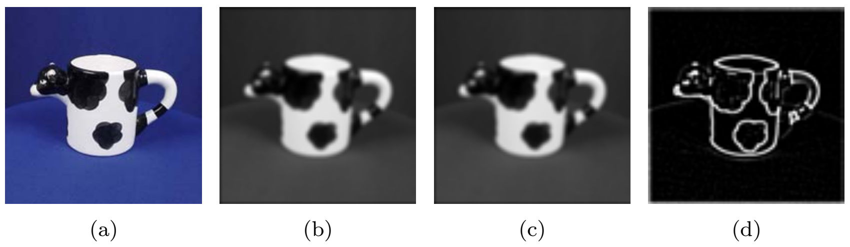


Figure 2.6: Example of difference of Gaussians. (a) Original image. (b) and (c) Image obtained by applying a Gaussian filter with σ_1 and σ_2. (d) The difference of Gaussians from (b) and (c).

to invariant points. Also, the scale-invariant feature detection uses the difference of Gaussians, which is an approximation of Laplacian-of-Gaussians detector but more efficient (as introduced in [13] and [53]). The difference of Gaussians of two adjacent image scales does a scale space using the minimum and maximum values. This detector was introduced by D. Lowe ([54] and [55]) and is defined by two steps.

1. Find scale-space extrema: for a given image $I(x, y)$ obtains the corresponding scale-space representation to find points that are invariant to the scale; thus, to determine the same structure in different scales. For each level of the pyramid applies the Gaussian filter with different values of σ:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.10}$$

and then, the difference of Gaussians is obtained with the difference between two adjacent filtered images separated by the factor k:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma). \tag{2.11}$$

Then, a sampling is applied in order to reduce the image size and the process starts again (see illustration in Fig. 2.7).

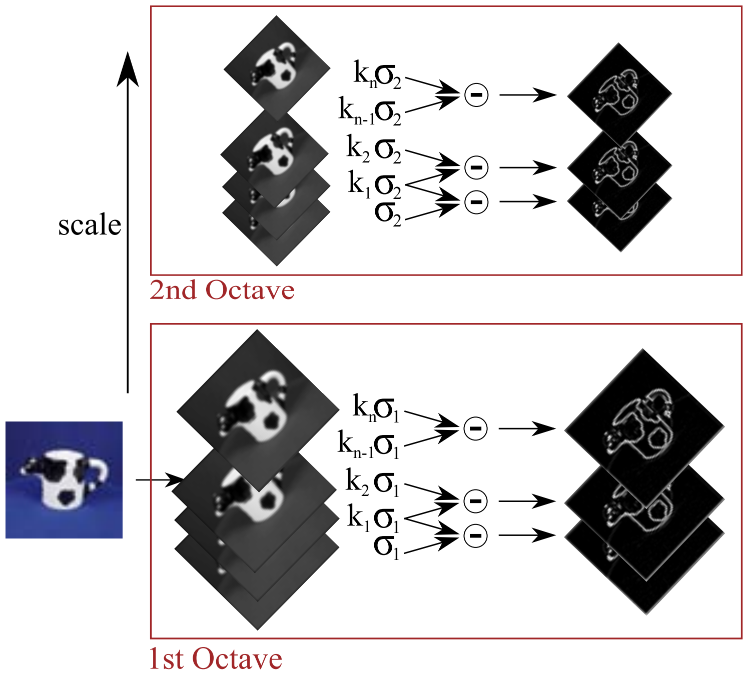


Figure 2.7: Illustration to find the scale-space extrema for SIFT.

Finally, this step returns a set of points that are local maximum or minimum values using the images obtained with the difference of Gaussians. In order to find these interest points the eight neighbors from the current image and nine pixels from the images in the previous and next level are considered; this process is depicted in Fig. 2.8.

2. Keypoint localization: detects the maximum and the minimum values of difference of Gaussians in the scale-space. Figure 2.8 depicts the 26 neighbors to compare the value. The points with low contrast are rejected and also the points that are in an edge.

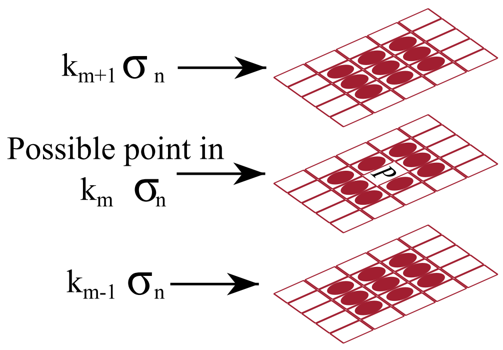


Figure 2.8: Illustration of keypoint localization for SIFT.

The scale-invariant feature detector finds invariant rotation points using the scale-space and applies the difference of Gaussians to simplify the Laplacian of Gaussians.

Fast Hessian detector

The fast Hessian detector is an approximation of Hessian detector that is used to find blobs in the image. This detector has good accuracy and computation time. The Hessian detector uses the determinant based in the matrix $H(x, y, \sigma)$

$$H(x, y, \sigma) = \begin{pmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{pmatrix}, \tag{2.12}$$

where each position (e.g., $L_{xx}(x, y, \sigma)$) is a Laplacian of Gaussian; it is the convolution of second order Gaussian derivative in the point (x, y) in scale σ

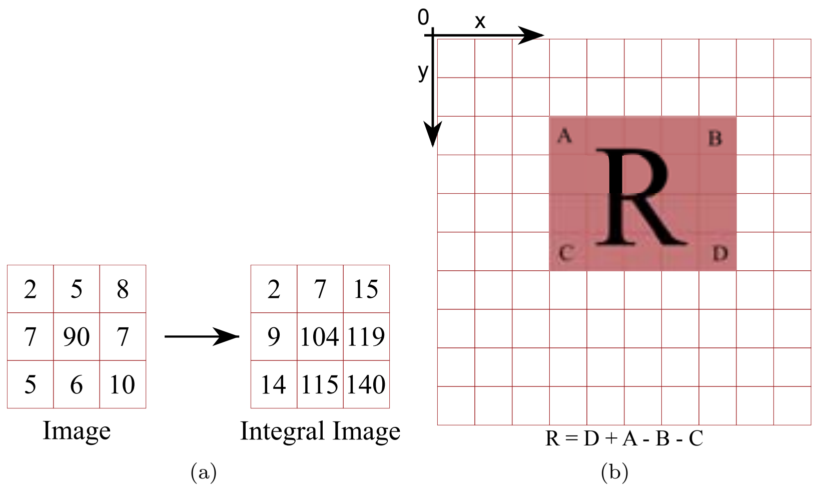
$$L_{xx}(x, y, \sigma) = \frac{d^2}{dx^2} G(x, y, \sigma). \tag{2.13}$$

The detector uses the determinant of the matrix to find the localization and the scale to the interest point. In order to speed up the process, the *integral image* can be used to compute the second order of Gaussian derivative.

The integral image was presented in [17], but the method became popular when Viola and Jones used it for object detection with Haar wavelets (e.g., [100] and [101]). The integral image is an efficient representation of a given image. It consists of accumulating the intensity values of previous pixel by:

$$I(x, y) = \sum_{x' \leqslant x, y' \leqslant y} i(x', y'), \tag{2.14}$$

a simple example is shown in Fig. 2.9.



2	5	8
7	90	7
5	6	10

Image

2	7	15
9	104	119
14	115	140

Integral Image

Figure 2.9: (a) A simple example of the integral image. (b) Process to calculate the area of R from the integral images A, B, C and D.

The simplification of Hessian detector to speed-up the process is by using an integer approximation of the matrix H introduced in Eq. (2.12). Fig. 2.10 illustrates the obtained approximation; in Fig. 2.10(a) a second order derivative of a Gaussian filter in y direction, with values around $[-2, 1]$ is depicted. Fig. 2.10(b) the integer approximation where the black color is -2, gray is 0 and white is 1.

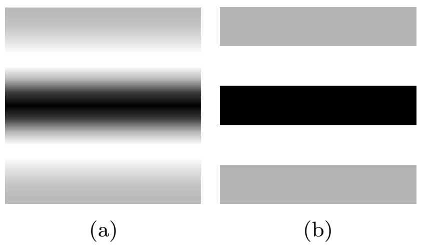


Figure 2.10: Second order derivative Gaussian filter and its approximation.

To obtain the scale-space, on the contrary to the scale-invariant feature detector, the original image is always with the same size and for each level of the pyramid it changes the size of the approximation of the H Hessian matrix (e.g., 9x9, 15x15, 21x21, 27x27, etc.).

The final step of interest point detections is to extract the points that have a local maximum determinant of the Hessian matrix, it is like scale-invariant feature detector. It is obtained by comparing the pixel's values with its 26 neighbors.

The fast Hessian detector finds invariant rotation interest points based on the determinant of the matrix H. In order to speed-up the process it uses the integral image and the integer approximation of the matrix H.

2.1.2 Feature point description

The goal of a feature point description algorithm is to produce a unique description in a fast and robust way to geometric and photometric changes. One of the first approaches has been proposed in 1999 [54] and since then it has been a very active research field. The state of the art of some of them is given below.

This section summarizes the set of descriptors used in this thesis. The descriptors are algorithms that given an interest point returns a description of this point based on its surrounding area.

Scale-invariant feature transform

The scale-invariant feature transform (SIFT) was introduced by David Lowe in ([54] and then [55]). This descriptor uses the scale-invariant feature detection as a detector to find the interest point. This descriptor consists of two steps:

1. Orientation assignment: this step gives the sought invariance to rotation. For each extracted point the gradient directions in all the points in a patch of 16×16 are computed.

$$\theta(x,y) = \tan^{-1} \frac{(L(x,y+1) - L(x,y-1)}{(L(x+1,y) - L(x-1,y)}, \tag{2.15}$$

together with the corresponding magnitude values:

$$m(x,y) = \sqrt{((L(x+1,y) - L(x-1,y))^2 + ((L(x,y+1) - L(x,y-1))^2}, \tag{2.16}$$

where L is obtained from the convolution of the given image with a Gaussian filter (like in Eq. (2.10)). For each region around the interest point an orientation histogram with 36 bins (each bin represents 10 degrees) is generated. The selected orientation corresponds to the peak of the histogram, adding together with that peak those bins that have more than 80% of the magnitude.

2. Keypoint descriptor: this second step gives the resulting features with invariant location and scale. To obtain the feature, the algorithm uses a windows of 16x16 pixels centered in the interest point; this windows is split up into 4x4 regions with 8 bins for each one of these regions (each bin covers 45 degrees) and a

histogram of orientations. The result of this step is a representation of 4x4x8 (128 values) as can be seen in Fig. 2.11.

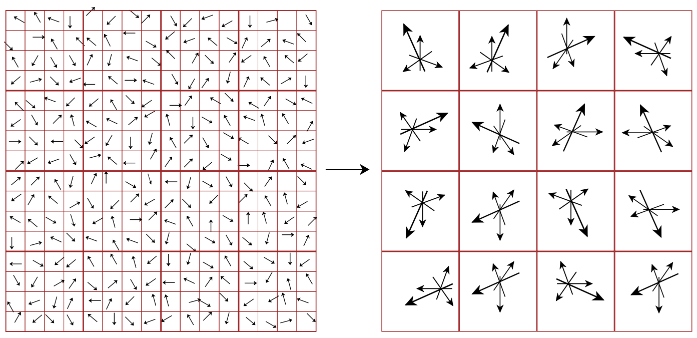


Figure 2.11: Illustration of Keypoint description from SIFT.

Pyramid histogram of visual words

The pyramid histogram of visual words (PHOW) descriptor was introduced in [11]. This descriptor does not use a detector to find the interest points; actually, there are some works supporting the idea that detectors are not needed for visual object recognition. Nowak *et al.* [66] studied the influence of features detector versus dense features and random features.

When the descriptor does not use a detector, the interest points are randomly extracted or densely sampled through the whole image. In this case, this descriptor extracts the interest points using a dense sampling. Thus, from each M pixels the process samples one pixel out of k as interest point. Fig. 2.12(b) shows a dense sampling of interest points from Fig. 2.12(a). Furthermore, in this case, in order to obtain the pyramid, the descriptor is applied several times. The k value is updated according to the level of the pyramid.

Given a set of interest points extracted from the iterative process explained above, the PHOW technique applies the SIFT descriptor explained in Sect. 2.1.2. Thus, the dimensionality of this descriptor is the same as SIFT – each point is described by a vector of 128 dimensions.

Color SIFT

The color SIFT was proposed to find better interest points given that more stable interest point gives better performance [98]. The color information provides an interest point with photometric information.

The first step in this descriptor is to change the RGB space to the opponent color space. The opponent color space has three layers: O_1, O_2 and O_3. O_1 and O_2

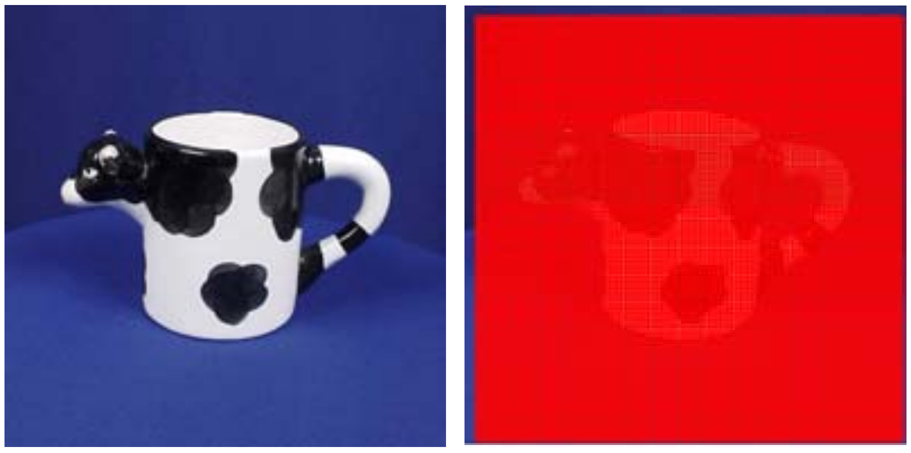

(a) (b)

Figure 2.12: Illustration after applying the PHOW descriptor.

represent the color information and O_3 represents the intensity information. They are defines as:

$$O_1 = \frac{R-G}{\sqrt{2}}; \quad O_2 = \frac{R+G-2B}{\sqrt{6}}; \quad O_3 = \frac{R+G+B}{\sqrt{3}}. \tag{2.17}$$

The first approximation using the SIFT descriptor with invariant color was introduced by [1]. The invariant color problem was solved using the invariant model proposed in [29] where the C-invariant is obtained eliminating the intensity information from the channels O_1 and O_2. Finally, C-SIFT [12] uses the C-invariant with the SIFT descriptor. The method to obtain the normalized opponent color space divides the channels O_1 and O_2 by the O_3 to extract the light intensity. The color with SIFT descriptor was studied in [98]. The feature descriptor consist of 384 values (128 for each channel as explained above in the SIFT case).

Speed-up robust feature

The speed-up robust feature (SURF) is a descriptor introduced by [3]. This descriptor uses the interest points extracted by the Fast Hessian detector. SURF is similar to SIFT, but the feature vector contains less dimensions; it also consists of two steps:

1. Orientation assignment: in order to be invariant to rotations the descriptor needs to know the principal orientation of the information contained in the patch centered in the interest point. That orientation is computed from the response in x and y with Haar wavelets (Fig. 2.13) in a circle with radius $6s$ centered in the interest point, where s is the size of the scale that was used to

obtain the interest point. The size of the Haar wavelet is $4s$. To speed-up the process integral images can be used.

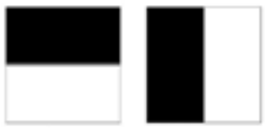


Figure 2.13: Illustration of Haar Wavelet for x and y axis.

Then, the orientation is computed from (x, y) components obtained above. The circle is divided into six regions (each one of $\pi/3$ rads) centered according to the obtained orientation. For each region, it sums up all the responses in the horizontal and vertical. The two summed values make the new vector (see the illustration Fig. 2.14). The dominant orientation assignment is the longest vector.

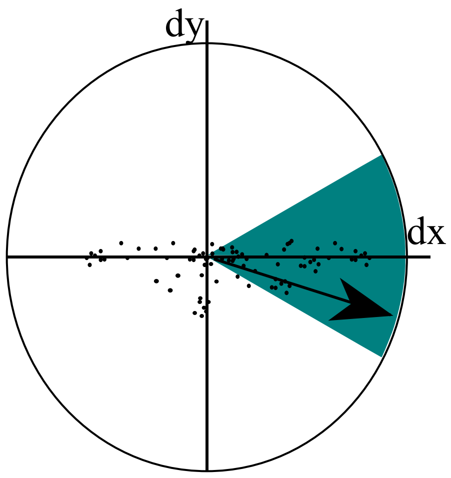


Figure 2.14: Illustracion of orientation assignment for SURF descriptor.

2. Descriptor component: the size of the windows for this descriptor is 20σ centered in the interest point and split up in 4x4 sub-regions. Each sub-region is 5σ and a Haar wavelet of 2σ size is applied to obtain 25 points for each one. For each sub-region extracts four values: $\sum dx$, $\sum dy$, $\sum |dx|$ and $\sum |dy|$. Figure 2.15 depicts an illustration of the process. Thus, the size of the feature vector is $4 \times 4 \times 4$ (64 values).

Spin images

The Spin image was introduced in [40] for matching range images and object recognition. Recently, an adaptation of the approach to the 2D image domain, together

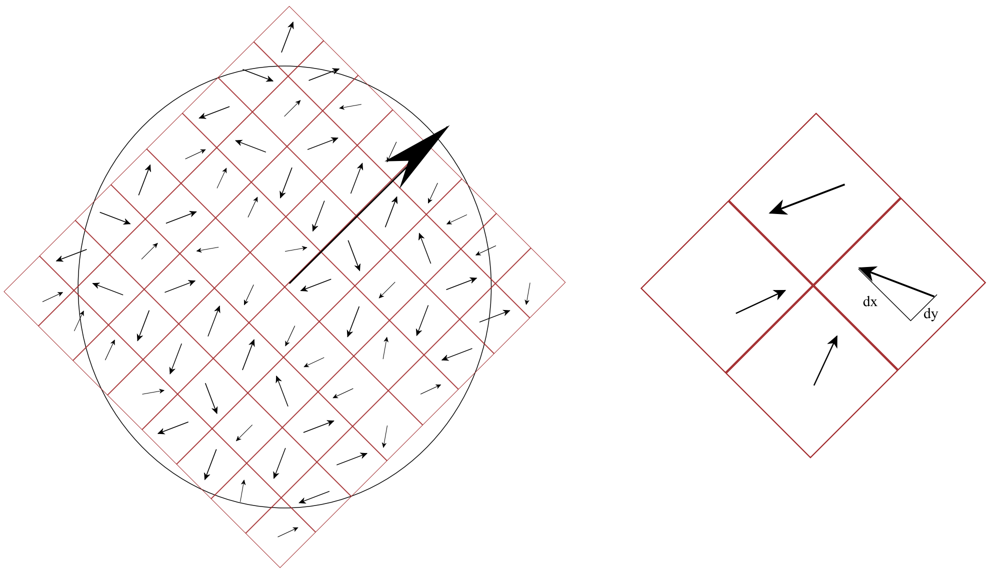


Figure 2.15: Illustration of the process to obtain the feature vector for SURF descriptor.

with a rotation invariance was introduced in [49] and [50]. This descriptor uses the set of interest points extracted by a Harris corner detector. For each point it extracts the feature as follow:

1. From a given normalized patch, the spin image is obtained. The process consists in representing each pixel from the normalized image patch in a polar reference system where: d is the distance between a given pixel to the center and i is the intensity in this pixel. Fig. 2.16 shows the representation of three points: d_0 the center (or the interest point obtained from Harris corner detector), d_1 and d_2 in the spin images space.

2. After obtaining the spin image the description is performed from the intensity values in the spin image. The values of the feature descriptor correspond to five concatenated histograms, where each histogram represents the intensity in a ring. The ring is defined with the distance from the center and the values of this ring are described in a histogram with 10 bins as detailed in [59].

2.2 Machine learning for computer vision

The artificial intelligence field attempts to understand the intelligence of the human in order to know how to build new agents. An agent is anything with *intelligence* that can receive information about the environment and then, act upon the environment with the effectors (Fig. 2.17).

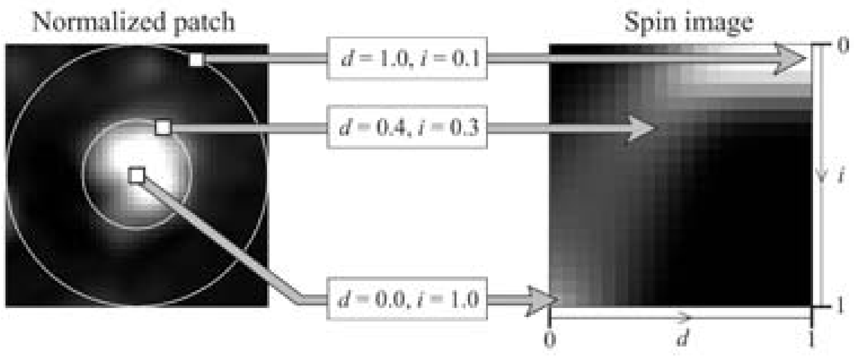


Figure 2.16: Illustration of the transform from a normalized patch to the corresponding spin image. This image was extracted from [49].

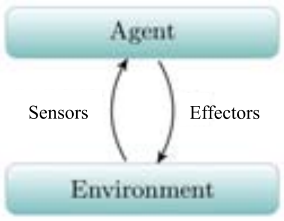


Figure 2.17: Illustration of generic interaction between an agent and the environment.

This section is focused on the machine learning applied to the computer vision problems (e.g., [60], [81]). In other words, how to train a model from a given training data set, which should be able to estimate the right outputs when a new data set is given. Typically, the taxonomy in machine learning is split up into four categories:

- Supervised learning: this category include those techniques that need pairs of labels $< input, output >$ to train. When the output is a continuous value the process is referred to as *regression*; on the contrary, if the output is categorical the process is referred to as *classification*.
- Unsupervised learning: in this category the input data is not labelled.
- Semi-supervised learning: includes hybrid techniques, which are a combination between the two approaches explained above. In this case, the input set is a mixed of data with labels $< input, output >$ and other without labels.
- Reinforcement learning: in the techniques in this category, the agent must learn the behavior through trial-and-error iterations with an environment [89].

This section, first summarize the Markov decision process in Sect. 2.2.1, which is used to define the problem; then, in Sect. 2.2.2 a possible solution using reinforcement learning is introduced. Finally, in Sect. 2.2.3 some examples of computer vision problems using the reinforcement learning technique (e.g., segmentation, face recognition, object recognition, ...) are given.

2.2.1 Markov decision process

The Markov decision process was introduced by Bellman that formulate the subject by functional equations, dynamic programming and the principal of optimality ([5] and [4]). The Markov decision process is a framework to make decisions in complex and with uncertainty problems. The decision maker is obtained from previous actions and random iterations. The Markov decision process is defined by states, actions and the dynamic of the environment in one step. Given a state $s_t \in S$ and an action $a_t \in A$, it obtains a new state $s_{t+1} \in S$. The new state s_{t+1} is selected using the transitions probabilities:

$$P^a_{s,s'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}. \tag{2.18}$$

The state s_{t+1} only depends on the state s_t and the applied action a_t, and not on the history of the process (the states and the actions applied before). The history of the system is the trajectory of the states $s_0 \to s_{t+1}$. This trajectory is defined with booth the states and the actions applied before s_{t+1}, thus, the trajectory has a finite number of rewards r_t until s_{t+1} like $r_0, r_1, ...r_{t-1}, r_t$. The expected reward for that transition is:

$$R^a_{s,s'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}, \tag{2.19}$$

thus, the most important aspects for the dynamic system are the transitions probabilities $P^a_{s,s'}$ and the expected reward $R^a_{s,s'}$.

The Markov decision process defines a π function that maps the states with the actions $\pi(s) \to a$. The π function maximizes the expected reward:

$$E\{r_0 + \gamma^1 r_1 + \gamma^2 r_2 + ...\}, \tag{2.20}$$

where γ is the discount factor for the future rewards ($0 \leq \gamma^i < 1$).

Hence, the main goal of the Markov decision process is to maximize some cumulative function of the reward. In order to find the solution of the Markov decision process, we need to find the optimal policy (π^*) that is defined with the discounted sum of rewards:

$$v_t = r_t + \gamma^1 r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.21}$$

and the array of policies (π). The value function $V(s)$ is the returned expected reward starting from state s and using the policy π.

$$V^{\pi}(s) = E_{\pi}\{v_t | s_t = s\}. \tag{2.22}$$

The value function applies the Bellman equation ([4] and [5]) and obtains:

$$V^{\pi}(s) = E_{\pi}\{v_t | s_t = s\} = E\{r_{t+1} + \gamma V(s_{t+1}) | s_t = s\}, \tag{2.23}$$

which can be solved using (e.g., [89], [81], [60], [77]):

- Dynamic programming: this method solves complex problems by breaking them into simple sub-problems.
- Monte-Carlo evaluation: it solves a complex problem by approximations of mathematics expressions using numerical results by random samplings.
- Temporal-difference learning: this method is the mixed from the two methods explained before. The temporal-difference learning uses the sampling method in the environment using a policy, and uses estimations previously learned to approximate the current estimation. In time t and if s_t is not terminal, the temporal-difference estimates the $V(s_t)$ after s_{t+1} is obtained. The temporal-difference estimates the $V(s_t)$ by:

$$V(s_t) \longleftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]. \tag{2.24}$$

The optimal value function is $V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s)$, which is obtained when in $(V^{\pi}(s))$ the optimal policy π^* is applied:

$$\pi^*(s) = \arg\max_{a}[\tau(s,a) + \gamma V^{\pi^*}(\delta(s,a))]. \tag{2.25}$$

When the δ and/or τ functions are unknown, the agent cannot learn the π^*.

2.2.2 Reinforcement learning

Reinforcement learning is a technique to solve the Markov decision process when the π function is unknown (e.g., [89], [60], [81], [41]). In other words, it is a learning method based on a trial-and-error strategy, where the agent does not have a prior knowledge about which is the correct action to take. For example, the babies learn by trial-and-error a lot of times, they play and they have some sensor-motor connections with the environment. The babies do not have an explicit teacher but exercising this sensor-motor connections produce information about cause and effect (the consequences of the actions) in order to achieve the goals.

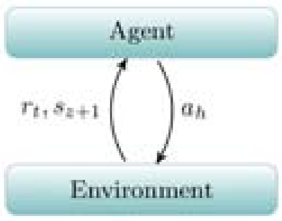


Figure 2.18: Illustration of interaction between an agent and the environment in a framework of reinforcement learning.

Reinforcement learning tries to emulate the learning nature of babies with the computer, in order to reproduce the behaviors of the humans (e.g., [92], [45], [79]). The reinforcement learning technique uses a set of actions and a set of states to learn the optimal action for each situation (state). In reinforcement learning there is an agent that given a state of the environment applies an action. This process produces a new state and a reward/punishment. It is a closed-loop that tries to find the optimal policy to select the action.

The Fig. 2.18 shows the process of the reinforcement learning. The agent interacts with the environment by selecting an action. Applying the action (a_t) at state(s_t), the environment gives a new state (s_{t+1}) and a reward/punishment (r_t). In order to maximize the expected reward, the agent selects the best action a_t based on the $\tau(s_t, a_t)$ provided by $\tau : s \times a \rightarrow \Re$.

The reinforcement learning can be solved using either an on-policy or an off-policy algorithm. In the on-policy, the agent always uses the same policy to select the actions a and a'. On the contrary, using an off-policy, the selection of a and a' could be different. The most popular on-policy algorithm is SARSA (state - action - reward - state - action) (e.g., [89], [80]) and for the off-policy, the Q-learning, is usually the choice.

Summarizing, the reinforcement learning technique dynamically learns for each iteration, hence, the technique does not need any training dataset. The challenge in reinforcement learning approach is to define the set of states and actions. But, once defined the problem (states and actions), the technique is independent of any dataset.

Q-learning

The Q-learning finds the optimal action-selection even when the agent does not have a model of the environment, thus, the agent does not know the effects of its actions on the environment. The Q-learning was introduced in [102] by Watkins to find the optimal action-selection when the δ and/or τ are unknown. The Q-learning algorithm calculates the *quality* of the pair (s, a):

$$Q : S \times A \longrightarrow \Re, \tag{2.26}$$

and uses the Q as an evaluation function to find the π^*, which is defined as:

$$Q(s,a) = \tau(s,a) + \gamma V^{\pi^*}(s)(\delta(s,a)), \tag{2.27}$$

where, Q is related to $V^{\pi^*}(s)$ by:

$$V^{\pi^*}(s) = \max_{a'} Q(s,a'). \tag{2.28}$$

The agent learns the action policy $\pi : S \longrightarrow A$, where π maps the current state s_t into an optimal action a_t to maximize the expected long term reward.

Similarly to the Markov decision process, the main goal is to obtain the optimal policy action-value function Q^{π^*}:

$$Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a). \tag{2.29}$$

The Q-learning proposes a strategy to learn an optimal policy π^*, when the δ and τ functions are a prior unknown. The optimal policy is:

$$\pi^* = \arg\max_a [\tau(s,a) + \gamma V^{\pi^*}(\delta(s,a))] = \arg\max_a Q(s,a). \tag{2.30}$$

Deterministic vs non-deterministic. The environment is deterministic when δ and τ functions are deterministic. Thus, given a s_t and a_t, the δ function always returns the same s_{t+1} and the τ function returns the same reward/punishment. On the contrary, if δ or/and τ are not deterministic, the environment is non-deterministic.

In deterministic environments the equation to obtain the optimal policy uses:

$$\hat{Q}_n(s_t,a_t) \longleftarrow r + \gamma \max_{a'} \hat{Q}_{n-1}(s_{t+1},a'), \tag{2.31}$$

and the nondeterministic environment uses:

$$\hat{Q}_n(s_t,a_t) \longleftarrow (1-\alpha_n)\hat{Q}_{n-1}(s_t,a_t) + \alpha_n[r + \gamma \max_{a'} \hat{Q}_{n-1}(s_{t+1},a')], \tag{2.32}$$

$$\alpha_n = \frac{1}{1 + \mathbf{visits}_n(s_t,a_t)}, \tag{2.33}$$

where $0 \le \gamma < 1$ is a discount factor for future reinforcements. The Eq. (2.33) is the value α_n for a nondeterministic world and **visits** is the number of iterations visiting the **Q**-table at the tuple (s_t,a_t) [60].

Convergence. The convergence in deterministic and non-deterministic Markov decision process is assured [103]. In deterministic Markov decision process, all the pairs (s, a) must be visited repeatedly, where for each visit it reduces the reward by γ and $\forall (s, a), |r(s, a)| \leq c$ there is a bounded reward. This convergence is achieved when $n \longrightarrow \infty$ where:

$$\Delta_n = |\hat{Q}_n(s, a) - Q(s, a)|. \tag{2.34}$$

In a nondeterministic Markov decision process, it converges when achieves the above criteria together with the following one (where $0 < \alpha_n \leq 1$):

$$\sum_i^{\infty} \alpha_n(i, s, a) = \infty, \qquad \sum_i^{\infty} [\alpha_n(i, s, a)]^2 \leq \infty, \tag{2.35}$$

which is true in the ith iteration, when $n \longrightarrow \infty$ with probability 1.

2.2.3 Applications of reinforcement learning in computer vision

Reinforcement learning is a learning technique widely used in the robotics community; recently, some work involving reinforcement learning have been proposed in the computer vision field. In this section, we summarize some works related with the computer vision field that use reinforcement learning to tackle different problems.

Contrast adaptation

In the visual object recognition field, as well as in others computer vision problems, the images need to be preprocessed. In this case, [96] introduces a method using the reinforcement learning to learn how to modify the contrast of the input image. The modifications that uses the method are a simple linear transformation based on histograms of contrast. Also, this process needs the feedback (τ function) from the human observers. When the reinforcement learning technique needs the human feedback the process is referred to as “Apprenticeship learning”.

Image segmentation

The process of image segmentation split up the given image into a set of regions. This process needs the tuning of thresholds. The image segmentation research field is active and there are a lot of authors working in this field. Some of the works are focussed on the interaction with the reinforcement learning technique in order to change the open-loop to closed-loop.

One of the first researchers working on image segmentation based on reinforcement learning was J. Peng and B. Bhanu in 1998 [75]. The authors had done several works on this topic [73], [74] and [6]. They developed a method to recognize objects that consists in three steps: image segmentation, feature extraction and model matching. The authors are focused in the segmentation step. The main goal is to find the best values for the parameters of the algorithm to maximize the matching confidence.

On the same topic as the previous work, G. Taylor [90] did a framework based on image segmentation with reinforcement learning. In this case, the ten optimizing threshold parameters are learned. Also, G. Taylor and C. Wolf propose to use the method explained before for the text detection problem in [91].

F. Sahba *et al.* [83] propose a method to reduce the exploration time using: states s, actions a, opposite-state s_o an opposite-action a_o. For each iteration of the process, the agent receives two states (the state and the opposite one), for each state it extracts the appropriate action and also, the updates of the Q-table is double, one for the state s and other for the opposite s_o.

On the contrary to the previous approaches, M. Shokri and H. R.Tizhoosh in [87] propose to learn not only the optimal threshold but also to select the best technique for each simple image. Finally, in [15] M. Chitsaz and C. S. Woo propose a segmentation method for medical images. In this case, the image is split up into sub-images and for each one there is an agent that learns the optimal threshold.

Edge detection

Edge detection is one of the basics process in most of the computer vision problems. An edge is considered when the intensity of one point and the adjacent pixels changes more than a user defined threshold. In [30] A. Gherega *et al.* propose to find the optimal threshold for a Sobel edge detector algorithm [65] based on a reinforcement learning strategy.

E-learning

A completely different approach than the ones presented above has been proposed by M. Shokri *et al.* in [88]. In this case, the reinforcement learning technique is used to learn the preferred image from a human observer. The training stage work as follow. Initially, for a given word, the algorithm returns a set of images. Then, a human observer provides to the system with the feedback, which is used to learn. Hence, the agent learns the preferred one of the human observer for each word.

Image retrieval

In the image retrieval field, given an image the process returns several images from the dataset that are similar to the query. Reinforcement learning based techniques have been also proposed in this field. For instance, P. Y. Yin *et al.* [104] propose to learn the optimal algorithm to retrieve the required images.

Behaviors recognition

The main goal in this field is to identify the human gestures. In this sense, T. Darrell and A. Pentland introduce a method to use the reinforcement learning to learn active recognition behaviors ([21] and [22]). In this case, the reinforcement learning is applied in a partial observable Markov decision process, thus, there are some hidden states. The previous work is extended in [20], where a multi-spacial scale of the image is used. To recognize the behavior the process uses two images, one corresponds to the complete image in a low resolution and other image that only shows a part of the complete image with higher resolution. The proposed method uses both images to describe the scene to recognize the behaviors.

Face recognition

The face recognition field aims at finding a concrete face in a digital image or video. The most widely extended approaches are based on the use of the eigenfaces and eigenvalues (extracting the principal components). In this field, some works have been also proposed to join face recognition and the reinforcement learning technique. First, M. T. Harandi *et al.* introduced in [33] a method that finds the dominant feature for each individual and then, uses all the dominant features to build a face recognition model. In this method, the formulation of Q-learning is not the usual. This case uses the first order of Markov decision process where the state is only affected by the past but not by the future. The authors [34] propose an extension to learn a set of features. For each individual the agent learns the most discriminant features, and in this case changes the formulation of Q-learning using it as usual.

Object recognition

The visual object recognition field is an active field where a lot of contributions have been recently done. In this section, we summarize the most relevant papers using the reinforcement learning in the object recognition domain (e.g., to select thresholds, to select methods and to select the camera motion to find the most informative viewpoint).

L. Paletta *et al.* propose a method to select the viewpoint which has more information to recognize a given object [70]. The proposed method learns the best position of the cameras, the illumination conditions and the parameters of the visual models to find the optimal viewpoint, thus, the more informative viewpoint to recognize the object. Then, the authors in [71] introduce a neural posterior network; in this paper, the camera motion is learned by the reinforcement learning technique. Finally, [69] introduce a method to learn the saliency in the image by rejecting the irrelevant features extracted by local descriptors and uses the reinforcement learning to find the saliency of the image that characterize the object.

Another way to recognize the object is by using a set of viewpoints. K. Häming and G. Peters [31] introduce a method using first order logic and reinforcement learning. This method is used to learn the optimal camera motion in order to find the visual features to recognize the object.

Then, there are approaches that use the two methods: the optimal viewpoint and also a set of viewpoints to add information. H. Borotschnig *et al.* [10] propose a method to find the best viewpoint and a set of viewpoints to classify the objects. This proposal is more informative and improve the classification ratio.

The previous papers find optimal viewpoints but, they do not use a new framework based on reinforcement learning to recognize the object. Now, we summarize a set of novel approaches, where new frameworks for object recognition based on reinforcement learning are introduced.

An interesting framework is proposed in [24] where B. Draper *et al.* propose a method to dynamically learn the vision procedures for specific tasks. The main goal for these authors is to build methods that can be adapted for any recognition task, but in this concrete paper, the proposed framework works with a dataset of aerial images and learns the best method to find houses in an image.

A complete new learning scheme is proposed by Jodogne *et al.* in a set of papers to learn the features of the objects to classify them. They start the research in [38] by introducing a method to automatically learn the visual classes, the agent only learns by the interactions with the environment and without supervisors. But, the visual features are exponential functions of the size for the images and the agent needs a large number of iterations to converge. The solution proposed in [38] are the supervised reinforcement learning and the reduction of the problem to a sequence of supervised regression. Then, S. Jodogne in [37] proposes a method to iteratively define the classes; then the classes are refined using the update of the Q-learning and the aliasing. Finally, an extension of this work is proposed in [39].

There are other works where the method learns as the cognitive model. An example is [52] that propose a method that uses two techniques: bottom-up and top-down. It consists of a learning technique inspired in the human learning model. The bottom-up level uses the Q(λ)-learning and the top-down level applies ordinal conditional functions (a first order logic technique). Finally, in [7] the reinforcement learning method is used to select the optimal classification approach: matching the features or the bag of features approach using a support vector machine.

Chapter 3

Problem definition

The previous chapter presents the theory of reinforcement learning technique (Sect. 2.2); additionally, the elements of the tuple $< S, A, \delta, \tau >$ that define the problem are introduced. From this tuple the state (S) and action (A) definition are the most difficult problems.

In this chapter we focus on the problem of how to define the state-action space, which is a classical "chicken and egg" problem. The state-action space definitions are close-related and one interferes with the definition of the other. Also, in this chapter we summarize a set of state definitions used for visual object recognition. Finally, a study to show the importance, in reinforcement learning, of finding the π function is given. As mentioned above, the π function maps the states and the actions. In Sect. 3.3 we propose two methods to solve the dimensional problem to find the relationship between the states and actions. Also, we compare the results using two approaches: first, we propose a dimensional reduction for each image and then concatenate all the values or, on the contrary, the second approach modifies the order of the steps. First, we concatenate the values and then, the dimensional reduction is applied to the whole class.

The remaining of this chapter is organized as follow: Sect. 3.1 introduces a problem to define the state and action space. Then, Sect. 3.2 summarizes a set of state definitions from the state of the art, and also the one proposed in the current work. Finally, a study to support the importance of using reinforcement learning in order to find the best π function is presented in Sect. 3.3, also, an empirical study is presented in Sect. 3.4. Finally, Sect. 3.5 gives the conclusions for this chapter.

3.1 Relationship between states and actions

Reinforcement learning has been largely used in game theory to adjust parameters, or in the robotic field for task such as path planning. In reinforcement learning, the agent obtains the state of the environment (see Fig. 2.18). Then, it applies the action (selected from the set of actions) and obtains a new representation of the environment. The set of states (S) represents the environment and the actions are effects on the environment. Thus, the state definition and the set of actions are very close related.

A state (s_t) represents the environment in time t. Thus, the set of states represents a set of possible situations of the environment at any time, while, the set of actions corresponds to the motions of the agent. Usually, the states are the "inputs" of the system and the actions are the "outputs".

The states and actions definitions are close related, actually, the definition of one affect the definition of the other [90]. In order to construct the state space we must know the set of actions. This relation can be seen in the definition of the δ function where given a state s_t and action a_t the function returns a new state s_{t+1}:

$$s_t \xrightarrow[r_t]{a_t} s_{t+1} \xrightarrow[r_{t+1}]{a_{t+1}} s_{t+2} \xrightarrow[r_{t+2}]{a_{t+2}} \dots \tag{3.1}$$

In order to reach a convergence in the reinforcement learning algorithm, we need to define the state space and also we must know how to take the actions. At the same time, to define the set of actions, we must to know the state space definition [2]. In Fig. 3.1 the dependence between the state space and the action space are depicted.

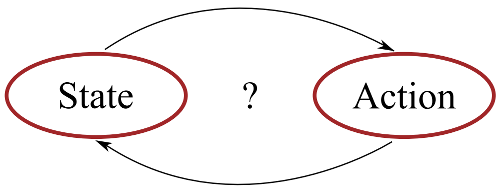


Figure 3.1: Relationship between states and actions.

In fact, we need to define the τ function ($S \times A \longrightarrow \Re$) to know how to evaluate the process. This evaluation has an important role to achieve the convergence. Thus, the agent needs the evaluation to obtain the states and actions [44].

In the current work we are focussed on the selection of the best descriptor for each image, then, the actions are already defined. Hence, the τ function is easy to define, actually it only compares the classification of the image with the ground truth and returns a reward or punishment value ($\Re$). The real problem, in this work, is how to define the state space.

3.2 State space definition

The problem of state space definition has been introduced in the section above. A few contributions on reinforcement learning are focussed on image problems; some examples can be seen in [78], [86] and [91]. The state representation proposed in these related work are briefly described below.

The state representation introduced by I. Qaffou *et al.* [78] uses three values to resume the characteristics of the image. The three values are the ratios between features of the obtained image and the ground truth. The first value is the ratio between the number of obtained edges and the ground truth, the second value is the ratio between the number of white pixels and then, the last value is the ratio between the length of the longest edges.

The second example has been introduced by K. Shibata and M. Iida; they use the combination of the information given by a CCD camera and infra-red sensors [86]. The original image has a size of 320×240; which is reduced to an image of 64×24. Thus, the new image has 1536 pixels that are stored like a vector and also adding four binary values obtained from four infrared sensors. Hence, the vector contains 1540 values that are used as inputs in a neural network. This neural network is trained using the back propagation method. Another example is the work presented in [91] that was also mentioned before in Sect. 2.2.3. In this case the authors propose to use the accumulative gradient filter to define the states.

This section presents the state space definitions proposed in the current work. These state space definitions are based on different characteristics of the image, which are widely used in the literature of computer vision. Usually, these characteristics are used to define and construct most of the descriptors in the state of the art. All these state vectors have the same structure. Each image is summarized in a vector of characteristics of 39 dimensions. As can be seen in Fig. 3.2, each image is split up into 13 squares, and for each square we extract three values. The following sections summarize the space state definitions based on this structure.

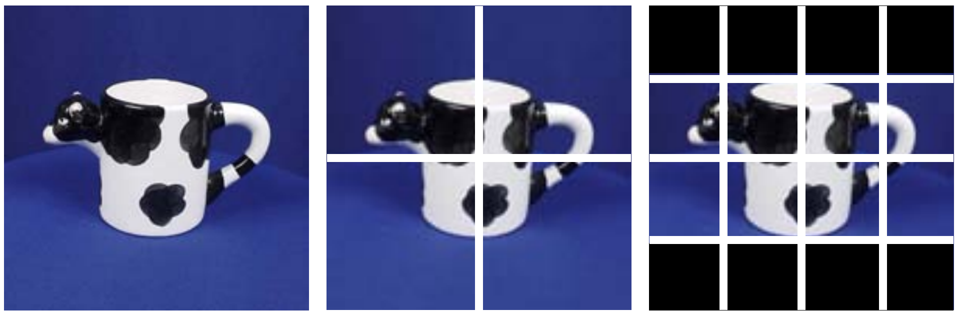

(a) (b) (c)

Figure 3.2: (a) Image from ETH dataset [51] (all the images from the dataset are resized to 128×128). (b) Image split up into four squares. (c) Image split up into sixteen squares, note that only eight of them are used.

3.2.1 $L^*a^*b^*$ based state definition

This state definition uses the $L^*a^*b^*$ color space. This color space is obtained by converting RGB to XYZ and then, XYZ to $L^*a^*b^*$ (see [57] and [82] for more details). The new color image has three layers: L^* in the range of $[0, 100]$ and then a^* and b^* are in the range of $[-110, 110]$, this new image is normalized into $[0, 1]$. Figure 3.3(b) shows a picture in $L^*a^*b^*$ color space while Fig. 3.3(a) depicts the original image in RGB color space. The L^* represents the luminance, a^* represents the difference between the red and green colors, and b^* is the difference between the yellow and blue. This state definition results in a vector of characteristics of 39 elements. This vector is computed using the scheme proposed in Fig. 3.2, which corresponds to 13 squares for every given image, actually for each square this state computes three median values, one for each channel of $L^*a^*b^*$.

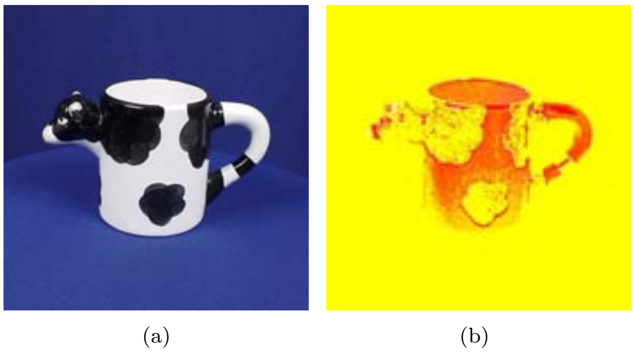


Figure 3.3: (a) Image from ETH dataset. (b) Conversion of RGB image to $L^*a^*b^*$ color space (note that this image is represented using the classical three channel color image representation).

3.2.2 Entropy based state definition

This state definition is based on the uncertainty of the information in the image and the position of the object. First, the state definition uses the grey level representation and, for each element of the partition image, it extracts the entropy as follow:

$$E = -\sum_{i=1}^{N} p_i log_2(p_i), \tag{3.2}$$

where $p(i)$ is the histogram of image and N is the number of pixels in the given region.

Then, it extracts the position of the object and complete the vector with the position of the image by (x, y). In order to extract the position, the process finds the maximum region of the image and assumes that this is the object. After that, it only needs to extract the centroid of this region (as can be seen in Fig. 3.4).

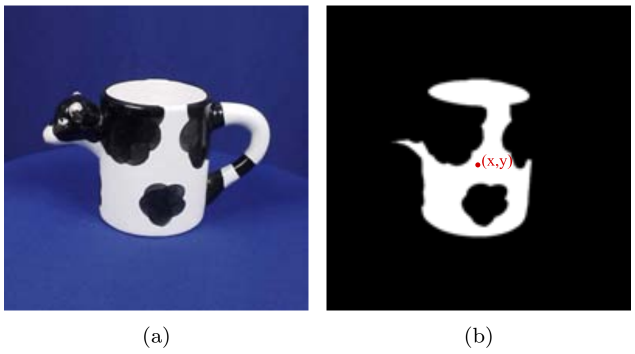


Figure 3.4: (a) Image from ETH dataset [51] (all the images from the dataset are resized to 128 × 128). (b) Maximum area together with its the centroid.

3.2.3 Gradient based state definition

This state definition uses the scheme presented above for a grey scale representation. For each element of the partition image it computes the gradient in x and y direction. Then, the state vector is built with the mean of gradient values and the module of the mean gradient in each image partition (see Fig. 3.2). Figure 3.5(b) shows the edges extracted using the gradient in x direction and Fig. 3.5(c) using the y direction.

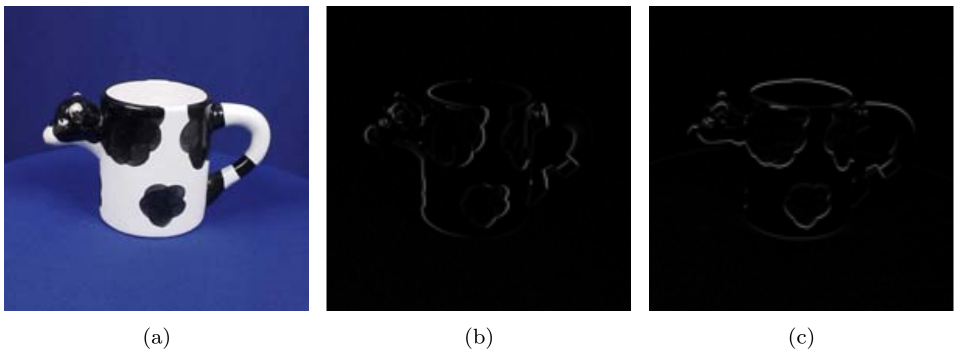


Figure 3.5: (a) Image from ETH dataset. (b) Gradient image in x direction. (c) Gradient image in y direction.

3.3 Is it really necessary the reinforcement learning to find the π function?

In the previous section we have enumerated different state space definitions, however in order to know which is the best one we need to analyse the problem. In other words, we need to find some relationship between the images and the descriptors. On the one hand, the state space is defined based on some characteristics of the image (e.g., gray values, entropies, etc.). On the other hand, the set of actions are the descriptors explained before (Sect. 2.1.2). Both definitions (states and actions space definitions) are working with an intrinsic value of the pixels in the image.

In concrete, the actions that we are using in our experiments are: SIFT, SURF, Spin, C-SIFT. In this case, the PHOW descriptor cannot be used because, in our approach we cannot use pyramids. Without using the pyramids, the result would be the same as the SIFT descriptor.

In the current study, we assume that exist an “intrinsic” relationship between the state and the action definition. In both cases, the same characteristics of the image are used in different ways. Hence, in case these relationships are found these concrete values are the mapping of the states and actions. Thus, we formulate the question whether the reinforcement learning is really necessary or not? In other words, it could be possible that the π function may be found without the reinforcement learning.

In order to evaluate the importance of the reinforcement learning, we search for the relationship between states and actions. The state space definitions are introduced above and the actions in Sect. 2.1.2. Each state and action space definition has different vector size. Thus, all the spaces have different dimension. Hence, we need to unify all the vectors to the same dimensionality before studying their relationship. To do so, we use two methods, first by using a principal component analysis (PCA) scheme (Sect. 3.3.1) and then by using a common vector (CV) formulation (Sect. 3.3.2).

3.3.1 Dimensional reduction with PCA

The principal component analysis is a mathematical technique to reduce the dimensionality of a problem [8]. This technique create a model with the maximum variability of the dataset. Thus, it transforms the data in a new representation with less information. The objective is to study whether it is possible to obtain a model that maps the features extracted by a descriptor and a state. If such a model is obtained for a given state we could obtain a simple feature, which should be similar to the one extracted by the descriptor. We use a linear model where D_{mat} represents a descriptor matrix and S_{mat} represents the state matrix:

$$Model = D_{mat} \times S_{mat}^{-1}. \tag{3.3}$$

In order to obtain this model, first, we need to extract the S_{mat} matrix that resume the values of the state in the training dataset. As can be seen in Fig. 3.6, the process extracts one state vector for each image in the same class from the training dataset. The different state vector definitions were presented in Sect. 3.2, which result for each image in a vector of 39 dimensions. After that, it concatenates all the vectors in a single matrix. Using the PCA algorithm it reduces the matrix in a new matrix with a concrete size. This simple process is repeated for each class of the dataset in order to make the model.

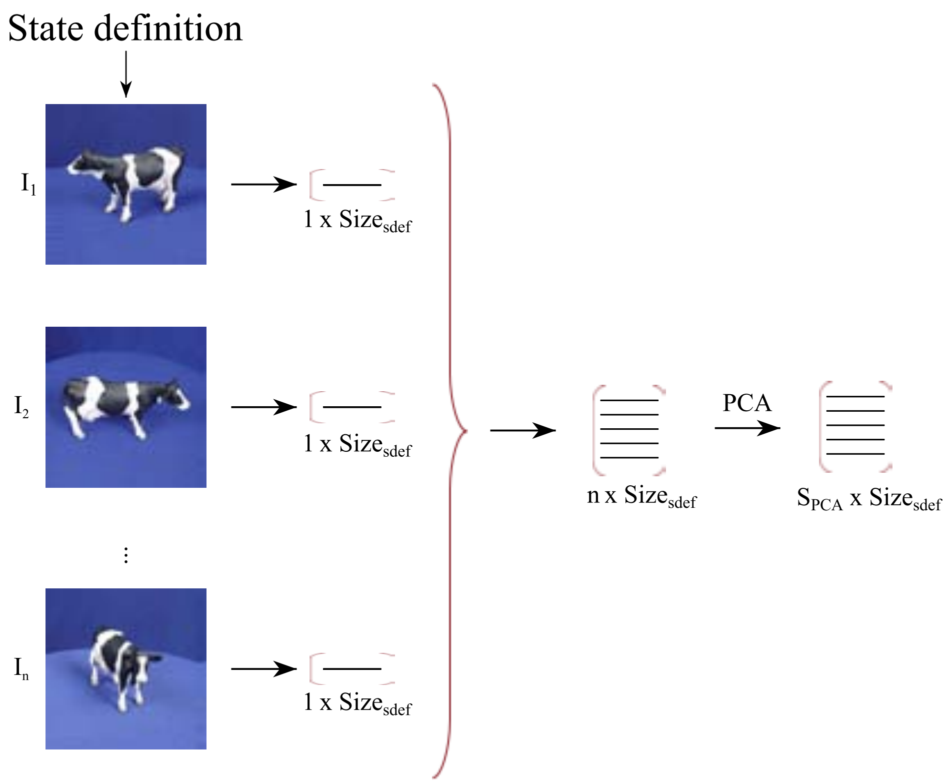


Figure 3.6: Process to resume the characteristics of the image in a matrix with a concrete size. $1 \times Size_{sdef}$ represents a vector with a $Size_{sdef}$ dimensionality for each image. S_{PCA} represents the size obtained by using PCA

This section presents two approaches to extract the D_{mat} matrix in order to build the model: *i*) by applying for each image the dimensionality reduction; *ii*) by applying for each class the dimensionality reduction.

Unify the dimensionality for each image

In this case, given an image it applies the selected detector/descriptor and obtains the interest points (IP_{d_n}). Then, it reduces the matrix obtained with the PCA algorithm to only use the interest points with more variability. This two steps are repeated for each image from the training dataset. Then, it concatenates all the small matrices in a single one; Fig. 3.7 shows an illustration of the whole process.

Unify the dimensionality for each class

This second approach uses the PCA algorithm once the matrix that concatenate all the interest points is obtained (see Fig. 3.8). First, for each image from the training

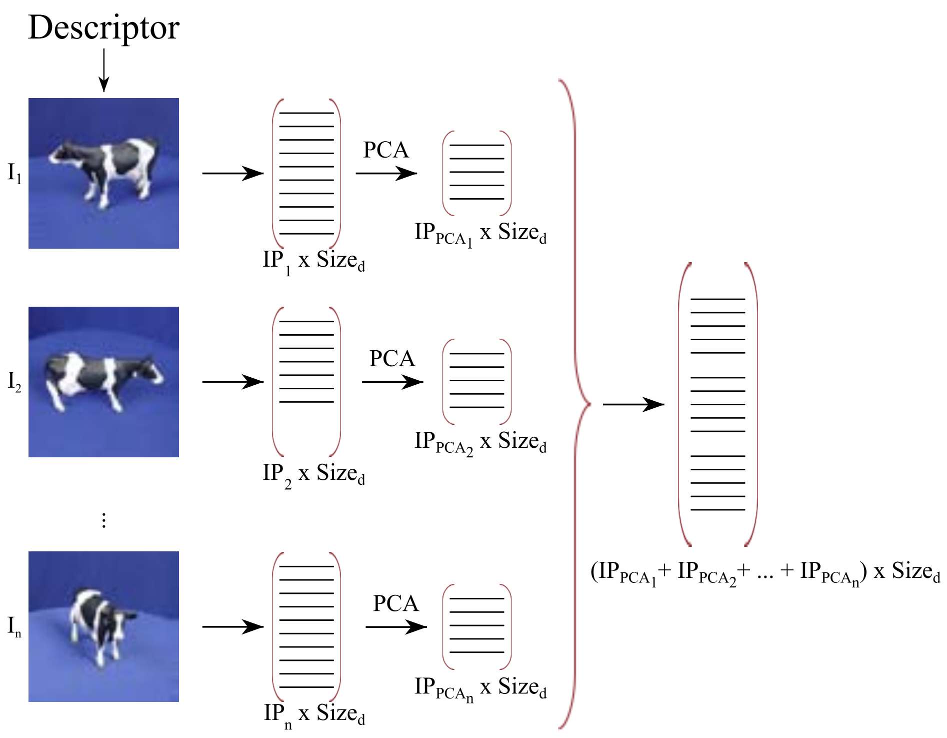


Figure 3.7: Applied process for each image to extract the features and reduce to a single matrix. IP is the number of interest points. $Size_d$ is the size of the selected descriptor. IP_{PCA} is the size of the resulting matrix after apply the reduction dimensional by the PCA algorithm.

dataset it extracts the interest points and their descriptions. Then, all the matrices are concatenated in a single one. Thus, each image has a different number of interest points but all with the same dimension. Finally, it reduces the concatenated matrix in a new one. This new matrix has a smaller size by the PCA algorithm that maximizes the variability.

3.3.2 Dimensional reduction with CV

This section introduces the second approach used for reducing the dimensionality of the model. This second approach applies the common vector (CV) algorithm [14]. The CV algorithm is used to reduce the dimensionality by searching the common information; while the PCA algorithm searches for the maximum variability intra class the CV searches for the maximum repeatability information. Sometimes, the common information is more discriminative than the maximum variability as in the case of the PCA.

The process to reduce the common information in a single matrix is the same pro-

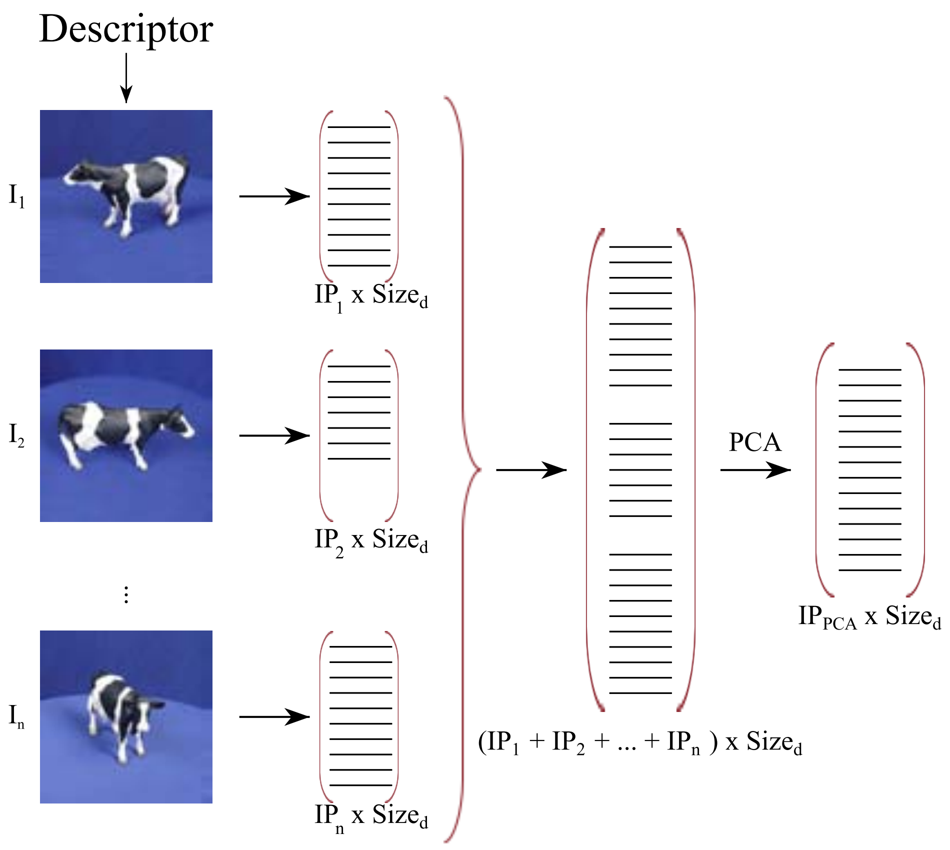


Figure 3.8: Applied process after concatenating all the features to reduce in a single matrix.

cess explained above but changing the PCA algorithm by the CV algorithm. Hence, the CV can be applied after or before the information is concatenated.

3.4 Mapping states and actions: an empirical study

In the experiments performed in this section the states are based on gradients, entropy and color. On the other hand, the descriptors used in this experiments are based on gradient and color. Hence, it is expected that the states based on gradient have the best results with the descriptors based on gradient.

In order to obtain the results, for each image from the testing dataset, we extract the state by the state definition. Then, using the model explained above, we obtain a vector that has an information content similar to the features extracted by the descriptor. So, to compare the descriptor with the obtained vector, the features are reduced to a single vector. The angle between the two vectors is used as a measure

Table 3.1: Results obtained by unifying the values with PCA for each image from ETH dataset. S: state; D: descriptor; E1: mean error; E2: median error; C: $L^*a^*b^*$; G: gradient; E: entropy; Sp: spin; S: SIFT; CS: C-SIFT; SU: SURF; P: PHOW (Bold face corresponds to the lowest error for the given state).

		Apple		Car		Cow		Cowcup		Cup	
S	D	E1	E2	E1	E2	E1	E2	E1	E2	E1	E2
C	Sp	13.97	15.41	7.07	7.74	10.52	13.26	12.2	12.14	16.13	14.91
C	CS	**9.46**	**9.24**	**8.58**	**8.44**	**9.9**	**9.38**	**10.45**	**10.47**	**5.7**	**6.61**
C	S	14.48	15.89	15.62	17.66	14.69	15.83	14.2	14.86	15.5	15.54
C	SU	15.87	15.59	11.19	11.75	15.95	15.95	16.33	16.41	14.94	15.22
C	P	17.55	19.1	25.25	23.75	18.96	17.93	15.77	15.16	19.54	20.17
G	Sp	11.39	13.17	5.6	6.44	8.22	9.91	17.99	17.38	11.65	12.85
G	CS	**7.84**	**7.81**	**9.87**	**8.91**	**6.69**	**6.88**	**8.34**	**8.8**	**5.75**	**6.7**
G	S	16.44	16.21	14.63	14.72	13.35	14.01	13.02	13.82	13.78	14.24
G	SU	14.17	15.52	14.26	13.9	18.17	17.03	16.93	16.82	14.52	14.77
G	P	17.28	18.86	20.86	25.21	17.77	17.71	19.91	19.64	17.87	18.8
E	Sp	13.29	14.84	**6.72**	**6.62**	14.69	15.5	17.44	15.97	15.18	14.86
E	CS	**7.59**	**7.88**	7.98	8.1	**7.18**	**7.14**	**11.94**	**11.12**	**8.44**	**8.47**
E	S	13.76	15.61	20.05	18.41	16.85	16.4	13.83	14.67	16.23	15.87
E	SU	17.65	17.52	15.27	13.7	16.77	16.09	19.87	19.31	16.35	15.65
E	P	24.04	22.49	21.97	21.18	18.99	18.96	16.62	17.21	15.89	16.56

		Dog		Horse		Pear		Tomato	
S	D	E1	E2	E1	E2	E1	E2	E1	E2
C	Sp	9.31	9.38	10.49	13.5	15.86	15.56	15.55	14.84
C	CS	**9.3**	**8.35**	**7.58**	**7.64**	**6.99**	**8.02**	**10.62**	**9.77**
C	S	16.85	17.52	15.39	14.59	12.76	13.64	18.61	19.06
C	SU	18.27	17.17	15.31	16.09	15.94	15.06	15.64	15.79
C	P	18.71	18.72	15.44	18.17	18.29	19.51	25.72	24.3
G	Sp	8.1	9.53	14.01	16.72	14.0	14.23	15.87	15.56
G	CS	**7.82**	**8.74**	**8.57**	**8.69**	**8.48**	**8.5**	**9.85**	**10.62**
G	S	15.89	15.75	16.64	16.43	15.79	15.53	23.54	21.74
G	SU	16.55	16.86	13.06	13.38	14.71	16.72	16.56	15.83
G	P	18.91	19.82	19.77	18.57	18.24	19.77	18.91	21.03
E	Sp	**5.79**	**7.43**	12.42	13.41	13.41	14.23	13.31	13.93
E	CS	9.09	8.87	**7.84**	**7.9**	**8.82**	**8.98**	**9.22**	**9.06**
E	S	15.4	15.93	14.73	14.98	12.09	12.52	17.8	18.94
E	SU	14.26	15.21	14.39	14.11	17.34	16.46	11.32	11.2
E	P	17.94	17.95	16.28	17.0	17.84	16.39	19.52	19.42

of similarity (orthogonal vectors correspond to completely different representations).

The experimental results are extracted from two datasets: ETH and COIL. Each dataset is split up into two sets: training and testing sets, each one with 15 images. The results are depicted in four tables: *i*) by unifying the dimensionality for each image using PCA; *ii*) by unifying the dimensionality for each class using PCA; *iii*) by unifying the dimensionality for each image using CV; *iv*) by unifying the dimensionality for each class using CV.

The obtained results are presented in Table 3.1, Table 3.2, Table 3.3 and Table 3.4. We decide to use the natural clustering to extract the values, in other words, using the classes of the dataset. We need to use a subsets because we need to generalize the study and also, we do not have information to do another clustering. Results from COIL dataset are not included for space limitations, but the conclusions are similar for both datasets.

In general, the error is smaller when the values are unified before joining the

Table 3.2: Results obtained by unifying the values with PCA for each class from ETH dataset. S: state; D: descriptor; E1: mean error; E2: median error; C: $L^*a^*b^*$; G: gradient; E: entropy; Sp: spin; S: SIFT; CS: C-SIFT; SU: SURF; P: PHOW (Bold face corresponds to the lowest error for the given state).

		Apple		Car		Cow		Cowcup		Cup	
S	D	E1	E2	E1	E2	E1	E2	E1	E2	E1	E2
C	Sp	13.87	14.45	**12.95**	**12.87**	19.44	19.23	15.59	**13.35**	19.01	18.11
C	CS	**11.41**	**11.28**	14.25	14.24	**11.76**	**11.66**	**13.66**	13.94	**10.41**	**10.44**
C	S	15.97	16.18	14.49	14.65	15.38	15.28	15.01	14.89	15.77	15.79
C	SU	14.98	14.83	18.42	18.94	18.12	17.97	15.26	15.46	12.11	11.78
C	P	14.07	14.01	14.48	14.63	15.89	15.81	14.7	15.19	18.26	18.3
G	Sp	13.23	13.32	14.24	14.53	14.44	14.45	13.12	13.59	13.82	13.37
G	CS	**8.86**	**8.66**	**8.13**	**8.4**	**7.43**	**7.48**	**8.93**	**9.01**	**5.88**	**5.98**
G	S	14.61	15.58	16.29	16.58	12.76	13.1	14.04	13.98	11.08	10.69
G	SU	14.81	15.67	14.4	14.11	14.71	14.36	14.17	14.19	13.07	13.89
G	P	20.47	20.07	22.25	22.47	15.74	15.77	14.49	15.48	15.02	14.99
E	Sp	14.38	14.56	11.13	11.18	15.55	15.64	11.42	11.47	13.0	13.0
E	CS	**6.67**	**6.67**	**10.18**	**10.41**	**10.17**	**10.5**	**8.84**	**8.8**	**9.35**	**8.96**
E	S	13.96	13.91	15.54	15.61	17.49	17.41	10.64	10.69	13.1	13.18
E	SU	26.19	26.44	13.91	13.88	16.56	16.41	10.42	10.37	16.29	16.28
E	P	13.52	13.43	16.67	16.67	17.53	17.65	16.82	16.75	19.82	19.87

		Dog		Horse		Pear		Tomato	
S	D	E1	E2	E1	E2	E1	E2	E1	E2
C	Sp	15.03	15.16	15.96	13.87	**8.9**	**8.82**	13.59	14.54
C	CS	**11.36**	**11.36**	**12.33**	**12.01**	8.98	8.92	**9.16**	**9.13**
C	S	17.11	16.89	15.31	15.38	17.4	17.68	13.9	13.87
C	SU	17.84	17.38	14.09	13.46	17.47	17.74	20.66	20.97
C	P	17.27	17.21	17.7	17.36	16.81	16.89	16.55	16.12
G	Sp	14.8	14.4	15.57	15.49	14.36	14.29	13.37	12.69
G	CS	**9.28**	**9.21**	**7.44**	**7.26**	**7.78**	**8.38**	**8.73**	**8.53**
G	S	13.28	13.3	12.41	13.34	11.45	11.32	18.52	16.81
G	SU	14.49	14.96	14.67	14.9	14.92	14.57	14.9	15.51
G	P	16.2	15.85	15.71	15.71	19.19	19.48	20.79	19.9
E	Sp	**12.35**	**12.36**	13.62	13.71	15.68	15.47	18.06	18.04
E	CS	15.46	15.47	**9.82**	**9.82**	**10.97**	**11.04**	**11.34**	**11.28**
E	S	18.36	18.17	17.89	18.05	15.89	15.87	14.67	14.61
E	SU	16.46	16.49	16.11	16.13	15.19	15.14	13.35	13.47
E	P	15.32	15.21	16.0	16.06	16.2	16.46	19.99	19.97

Table 3.3: Results obtained by unifying the values with CV for each image from ETH dataset. S: state; D: descriptor; E1: mean error; E2: median error; C: $L^*a^*b^*$; G: gradient; E: entropy; Sp: spin; S: SIFT; CS: C-SIFT; SU: SURF; P: PHOW (Bold face corresponds to the lowest error for the given state).

		Cup		Dog		Horse		Pear		Tomato	
S	D	E1	E2	E1	E2	E1	E2	E1	E2	E1	E2
C	Sp	2.44	2.68	3.12	3.44	4.86	4.96	5.86	5.83	6.5	7.0
C	CS	**1.64**	**1.88**	**2.36**	**2.48**	**3.84**	**3.61**	**4.38**	**4.14**	**4.95**	**5.2**
C	S	2.43	2.54	3.13	3.36	4.8	4.87	5.67	5.5	6.3	6.78
C	SU	2.35	2.92	3.58	3.83	5.19	5.58	6.29	6.17	7.85	7.38
C	P	2.05	2.73	3.38	3.59	4.7	4.96	5.3	5.42	6.44	6.55
G	Sp	3.43	3.97	5.34	5.63	6.55	6.44	7.04	7.13	8.2	8.65
G	CS	**3.24**	**3.11**	**4.33**	**4.21**	**4.77**	**4.66**	**4.93**	**4.96**	**6.05**	**6.19**
G	S	3.8	3.99	5.26	5.44	5.63	5.89	6.26	6.34	8.06	8.03
G	SU	4.45	4.68	5.79	5.95	6.49	6.65	6.88	7.21	8.5	8.48
G	P	3.84	4.2	5.15	5.59	5.42	5.99	6.08	6.4	8.12	7.51
E	Sp	4.07	3.73	5.4	5.18	6.35	6.72	7.61	7.39	8.38	8.04
E	CS	**2.7**	**2.62**	**3.68**	**3.61**	**4.25**	**4.53**	**4.71**	**4.88**	**5.58**	**5.39**
E	S	3.28	3.38	3.98	4.63	5.44	5.82	5.68	6.24	6.05	6.91
E	SU	3.94	3.72	5.23	5.13	6.19	6.54	6.58	7.03	7.43	7.57
E	P	3.49	3.4	4.49	4.72	5.46	5.81	5.37	6.14	6.73	6.64

Table 3.4: Results obtained by unifying the values with CV for each class from ETH dataset. S: state; D: descriptor; E1: mean error; E2: median error; C: $L^*a^*b^*$; G: gradient; E: entropy; Sp: spin; S: SIFT; CS: C-SIFT; SU: SURF; P: PHOW (Bold face corresponds to the lowest error for the given state).

		Apple		Car		Cow		Cowcup		Cup	
S	D	E1	E2	E1	E2	E1	E2	E1	E2	E1	E2
C	Sp	44.09	43.97	22.11	21.26	18.21	18.33	**2.56**	**7.83**	28.18	31.47
C	CS	**12.58**	**12.39**	**13.43**	**13.67**	**12.07**	**12.03**	12.12	11.69	**12.93**	**12.49**
C	S	18.57	19.13	18.94	18.32	18.92	20.07	17.73	17.76	22.44	22.02
C	SU	20.89	20.64	18.1	17.35	20.25	20.31	21.14	21.09	19.47	20.43
C	P	18.87	18.94	18.0	17.91	17.91	17.62	18.85	18.85	19.02	19.19
G	Sp	44.09	43.97	21.72	20.52	20.94	20.84	**2.56**	**8.02**	28.18	31.83
G	CS	**11.94**	**12.38**	**14.4**	**14.83**	**12.07**	**12.03**	12.12	11.69	**12.3**	**12.19**
G	S	18.52	18.94	19.74	20.04	18.92	20.07	17.73	17.76	22.44	22.02
G	SU	22.36	20.98	17.37	17.83	20.25	20.31	21.14	21.09	19.47	20.43
G	P	19.02	19.15	19.85	19.28	17.19	17.3	19.25	19.18	18.94	19.7
E	Sp	44.09	43.97	24.37	22.03	21.73	22.37	**2.56**	**7.52**	28.18	32.13
E	CS	**11.45**	**11.79**	**13.72**	**14.43**	**12.07**	**12.03**	12.12	11.69	**12.3**	**12.22**
E	S	18.52	18.73	19.14	19.45	18.92	20.07	17.73	17.76	22.44	22.02
E	SU	20.5	20.01	17.9	18.18	20.25	20.31	21.14	21.09	19.47	20.43
E	P	18.91	19.23	18.26	18.55	17.28	16.9	19.68	19.62	20.31	20.07

		Dog		Horse		Pear		Tomato	
S	D	E1	E2	E1	E2	E1	E2	E1	E2
C	Sp	22.11	21.43	19.88	22.11	52.16	51.99	31.0	33.36
C	CS	**13.52**	**13.58**	**12.09**	**12.46**	**12.65**	**12.38**	**13.83**	**14.02**
C	S	19.83	21.36	20.85	20.17	20.51	21.05	19.74	19.29
C	SU	21.79	21.93	21.2	20.96	21.57	20.68	19.02	19.13
C	P	17.57	18.64	17.97	18.06	18.69	18.98	19.43	18.67
G	Sp	20.35	19.62	19.03	22.21	52.16	51.99	31.0	34.14
G	CS	**14.43**	**14.64**	**12.09**	**12.46**	**12.65**	**12.38**	**13.83**	**14.32**
G	S	19.83	21.28	20.85	20.17	20.51	21.05	17.3	18.27
G	SU	21.79	21.84	21.2	20.96	21.57	20.68	19.95	19.54
G	P	15.91	16.08	15.61	15.96	19.23	18.86	17.99	18.76
E	Sp	20.5	19.6	20.41	22.3	52.16	51.99	31.0	35.08
E	CS	**14.43**	**15.35**	**12.09**	**12.46**	**12.65**	**12.38**	**13.83**	**14.19**
E	S	19.83	21.31	20.85	20.17	20.51	21.05	19.74	19.8
E	SU	21.79	22.11	21.2	20.96	21.57	20.68	18.74	18.57
E	P	17.27	17.33	17.34	17.42	17.95	17.83	19.39	19.62

values of the class (for each image). The best results, or the results with minimum error, are obtained when the common vectors are used. However, we cannot identify which combination is better, because, the combination changes depending on the class. Usually, the "best descriptor" is C-SIFT but in some cases changes for Spin. The cowcup case uses more times the spin than the C-SIFT. Also, if we study the behaviour for each image the combination also change. Hence, we cannot identify a linear combination between characteristics and descriptors, which some how support the usage of a learning approach to find such combination.

3.5 Discussion

This chapter presents a linear method to find the combination between characteristics of an image and a descriptor. This method tries to select the descriptor for each class using the characteristics of the images. According to this study we can conclude that there is not a linear relationship between them. This study also shows that the combination changes in the same class. So, we need to select the descriptor for each image. Perhaps, the characteristics used to find the descriptor give more information than the class of the object. In this case, the descriptors should be selected according to the information from the pixels of the image, but not according to the information of the classes. In conclusion, we need to build a new subset of images based on the information from the pixels of the image. Also, the selected descriptor depends on the image, thus, we need to learn relationship; since it is not a linear function. For these reasons, we need to propose a new learning framework where the selected descriptor depends on the characteristics of the image.

The next chapters of this thesis present a framework to select the best descriptor using the Q-learning algorithm based on the reinforcement learning technique. This framework propose a method to solve the problems presented above.

Chapter 4

Descriptor's selection

In the computer vision domain the visual object classification (VOC) has attracted the attention of researchers over the last two decades (e.g., [75], [55], [26] and [9]). Generally, VOC is based on the representation of the given scene in a space of features, which were extracted and then described by means of some feature descriptors. These feature descriptions are then used as discriminative elements to characterize the given objects. They are computed using information of interest points together with their neighbourhood; such interest points are pixels with special characteristics (e.g., [35] and [97]). Hence, given an image, the feature descriptors characterize the objects at a higher abstraction level, where classical learning techniques can be used in order to recognize the target object. More elaborated techniques, such as bag of features are becoming nowadays popular for visual object classification (e.g., [26], [18] or [3])

The bag of features architecture is flexible (see Sect. 2.1), so that there are different combinations that can be used to implement the four steps presented before. The final performance of the bag of features depends on the correct algorithm selection.

The work in this thesis is focused on the first step of the bag of features; in particular, the goal is to learn the best algorithm to describe the interest points. From our experience, the performance of the bag of features is strongly influenced by the image feature descriptor, so we state that identifying the best descriptor for each image will improve the classification rate. Actually, a naive approach to solve this problem could be the usage of a concatenation of all the possible descriptors. However, this solution is not always feasible since on the one hand it could take a large amount of resources (e.g., memory, CPU time) and on the other hand this would introduce noise to the solution [25]. The challenge of the problem and the importance of finding the right solution have been recently addressed (e.g., [25]).

The topic of this chapter is the learning of the best descriptor for each image to improve the classification ratio. A priori, we have no information on how to discriminate the descriptors. In the current work we propose to use a reinforcement learning technique where some easy to compute features lead to obtain the best descriptor for a given image.

The proposed framework uses the reinforcement learning with the bag of features to classify the images. Given an image the agent selects the best descriptor (the action that maximize the expected reward) to classify the object. In order to select the best descriptor, we propose a framework that joins two techniques:

- Bag of features: to classify the objects.
- Reinforcement learning: to select the appropriate descriptor.

This chapter presents, first, the whole framework that merges the bag of feature approach and the reinforcement learning technique. Then, it introduces the tuple adaptation of the reinforcement learning to use with images. Additionally, the convergence criteria is presented, which help to have a fast result; and finally, the exploration/exploitation trade-off is summarized.

4.1 Adaptation of the reinforcement learning elements to the bag of features scheme

The theoretical framework of reinforcement learning was introduced in Sect. 2.2, also some of its applications in the computer vision field were introduced in Sect. 2.2.3. In general, the reinforcement learning technique is used in computer vision to solve problems of thresholding, learning behaviours and path planning. In the current work we propose the usage of reinforcement learning in order to learn the best descriptor for each image from a given dataset.

In this section, we present the elements of the Markov decision process $< S, A, \delta, \tau >$ to be used in our computer vision formulation and in concrete, in a bag of features approach. Section 4.1.1 introduces the state definition. Section 4.1.2 proposes the set of actions. In this method, the π function (mapping $s \longrightarrow a$) does not use the future, but only the past. Then, Sect. 4.1.3 explains the δ function. Finally, Sect. 4.1.4 proposes the τ function and the definition of the rewards and punishments.

4.1.1 State definition

As presented in Chapter 2 the set of states S is a representation of the environment. In other words, this set contains k states that can be used by the agent to select the appropriate action:

$$S = \{s_0, s_1, ...s_{k-1}\} = \{s_z\}_{0 \leq z < k}. \quad (4.1)$$

As mentioned before, in this work a state is a representation of an image. In other words, the features used to represent the given image. Many different image features can be used as a state. In this section we propose a state definition based on statistics values, which are then concatenated with corners and blobs.

The proposed state definition consists on first converting the images to a grey scale. Then, the mean, standard deviation and median values from the image intensity values are extracted (Fig. 4.1(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 of the intensity values (as depicted in Fig. 4.1(b)). Additionally, the number of corners (Fig. 4.1(c) shows the corners in this image) and the number of blobs (Fig. 4.1(d) shows the blobs in this image) using the whole grey scale image (Fig. 4.1(a)) are extracted. The corners are obtained using the Harris corner detector [35], while blobs are obtained converting the grey scale image to black and white using OTSU threshold [68] and then, the labelling algorithm presented in [32], with a connectivity of eight neighbours is applied. The result gives a vector of 17 dimensions with the following structure:

$$\langle mean_{L_1}, std_{L_1}, median_{L_1}, mean_{(1,1)L_2}, std_{(1,1)L_2}, \dots, median_{(2,2)L_2}, ncorners, nblobs \rangle \, . \tag{4.2}$$

This state definition is highly discriminative among images. Since the dataset 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 the centroid of each cluster is used as a state ($S = s_{\{0 \leq z \leq k\}}$), instead of using all the vectors' components. Hence, the size of the **Q**-table is determined by the number of clusters.

4.1.2 Action definition

The set of actions contains all the possible actions that the agent could do. In this thesis, the actions are descriptors:

$$A = \{a_0, a_1, a_2, ...a_{u-1}\} = \{a_h\}_{0 \leq h < u}, \tag{4.3}$$

where u is the size of the descriptor set. In other words, our agent learns which is the descriptor that gives the best information for each image. Our architecture is flexible and does not depend on a particular set of descriptors. As mentioned above, in the current work the descriptors used as actions are based on gradients, blobs and patterns, although other combinations could be also used.

4.1.3 δ function

The classical Q-learning formulation involves a δ function, which for a given state (s_z) and an action (a_h) returns a new state ($\delta : S \times A \rightarrow S$). In our case, given an image from the data set (I_{s_z}), the δ function does not return a new state, instead, the output is a new representation of the image I'_{s_z} (the features of the image applying

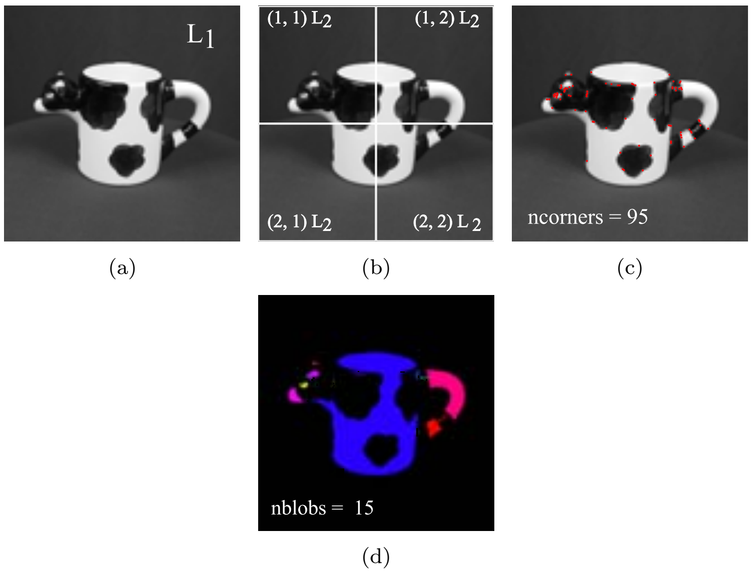


Figure 4.1: Illustration of an image of the dataset. (a) Original gray level image. (b) Image split up into four equally sized squared blocks. (c) Corners detected in the image. (d) Blobs detected in the image.

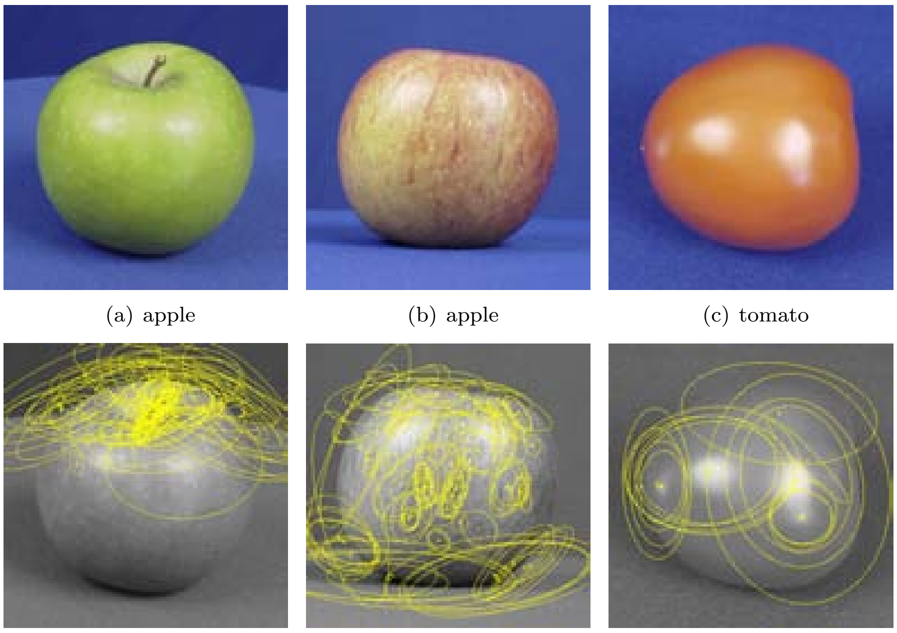
that action). Also, applying the same action to two different images located at the same state s_z, i.e. both images are in the same centroid and applying the same action, could results in different image descriptions (see the example in Fig. 4.2 where the three images are in the same cluster and by applying the same action, the image descriptor representation is different for each one). For this reason, the δ function is nondeterministic.

Once the 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 other three steps. Once the object has been classified, the iteration starts again with a new image from the given dataset.

4.1.4 τ function

The agent classifies a given image using the bag of features approach and the descriptor indicated in the state s_z. 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 learning process we cannot ensure that two images at the same state (s_z) will receive the same reward/punishment $r_t = \tau(s_z, a_h)$ after applying the same action a_h [60]. For example, in Fig. 4.2 using the action SIFT over the three images we obtain the same reward



(a) apple (b) apple (c) tomato

(d) (e) (f)

Figure 4.2: (*top−row*) Illustration of three images from the dataset. Although these images belong to different classes, their centroid lies in the same position. Hence, they belong to the same cluster. (*bottom − row*) The image patches obtained after applying the same action: SIFT descriptor.

because the bag of features correctly classifies all of them. But, if we repeat the same example changing the action SIFT by SURF, the first image (Fig. 4.2(a)) is within the class apple but the bag of features returns tomato. For the second image (Fig. 4.2(b)), the bag of features hits the class. Finally, the third image (Fig. 4.2(c)) is a tomato but the bag of features returns apple. Hence, using SURF, for the image (Fig. 4.2(b)) results in a reward but in the other two images (Fig. 4.2(a) and Fig. 4.2(c)) 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.

4.2 Convergence

The "Convergence of Q-learning for nondeterministic Markov decision processes" theorem [60] shows that a nondeterministic Markov decision process (introduced in Sect. 2.2.1) converges when there is a bounded reward $(\forall(s,a), |r(s,a)| \leq c$ and $0 < \alpha_n \leq 1)$. Eq. (4.4) is true in the ith iteration when $n \longrightarrow \infty$ with probability 1 as can be seen in Sect. 2.2.2:

$$\sum_{i}^{\infty} \alpha_n(i,s,a) = \infty, \qquad \sum_{i}^{\infty} [\alpha_n(i,s,a)]^2 \leq \infty. \tag{4.4}$$

In our framework, the agent interacts with a non-deterministic environment (as introduced in Sect. 4.1.3 and Sect. 4.1.4), so, the convergence is very expensive in time. As the convergence can last for weeks, we need some criteria to stop the training. Thus, in the current work, the framework has a sliding windows to save the last w errors in the Q-table. It is proposed to stop the training when the cumulative of the last w errors is less than a threshold θ, thus, when the following criterion is achieved the convergence is true:

$$\sum_{i=n-w}^{n} |\mathbf{Q}_{i+1}(s,a) - \mathbf{Q}_i(s,a)| < \theta, \tag{4.5}$$

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

4.3 Exploration-exploitation trade-off

In this section the criterion proposed for selecting an action is detailed. In general, it is referred to as exploration-exploitation trade-off. If the agent only uses the exploration strategy it could fall into a local maximum, in other words, it could happen that the agent does not reach the true optimal path. On the contrary, if the agent only uses the exploitation strategy, the agent does not maximize the utility. 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 work, we propose a method to adaptively compute ε. We propose to use the measurement of the error as the parameter for switching the strategy. We define the error as:

$$e = |Q_{it} - Q_{it-1}|, \tag{4.6}$$

and, for each iteration we store this error. The ideal process reduces the error for each iteration, although it could happen that sometimes the error increases (see Fig. 4.3). Hence, we propose to calculate e_{it} and use this value as a switching indicator:

$$e_{it} - e_{it-1} > \mathit{threshold}. \tag{4.7}$$

Additionally, to avoid switches when the error is a small spike, we propose to consider also the error in the neighborhood of current iteration. Hence, a sliding windows (w') is used as indicated below:

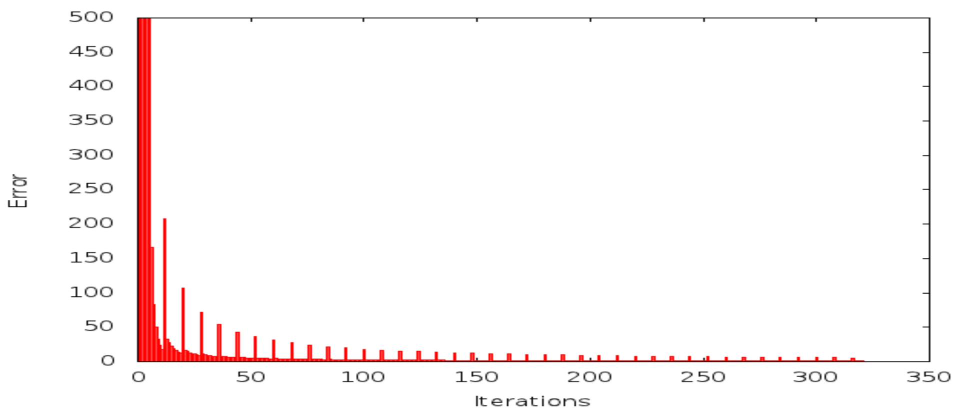


Figure 4.3: Behavior of e_{it}, till convergence is reached, for different iterations.

$$\sum_{i=0}^{w'} e_{it-i} > \sum_{i=1}^{w'+1} e_{it-i}. \tag{4.8}$$

In other words, if the condition is fulfilled the current strategy is switched to exploitation where, like in Eq. (4.5), w' is the size of the sliding windows (in the current implementation $w' = 5$).

4.4 Framework

In order to train and test our approach, we have considered an image dataset and a set of descriptors widely used as indicated by the computer vision literature. The image dataset is split up into three sets: vocabulary tree training set (BoFTS), **Q**-table training set (QTTS), and testing set.

The first training creates a model for each class of the dataset with a unique descriptor using a classical bag of features. This process is repeated for each descriptor, thus, each action of the set of actions (A) has a set of models. The classical bag of features consist of four steps (Sect. 2.1). In our framework, we focus on the first step and the other three steps are used as usual; the k-d tree clustering and support vector machine are considered to classify the objects (e.g., [99]).

The second training starts initializing the Q-table, the set of states S and the set of actions A. Usually, the Q-table is initialized with zeros, but sometimes the initialization can be done with random values or with statistical information. In the current work we initialize the Q-table with zeros.

The next step is to define the set of states S, i.e., for each image in the QTTS compute the characteristics of the image using the state definition explained before

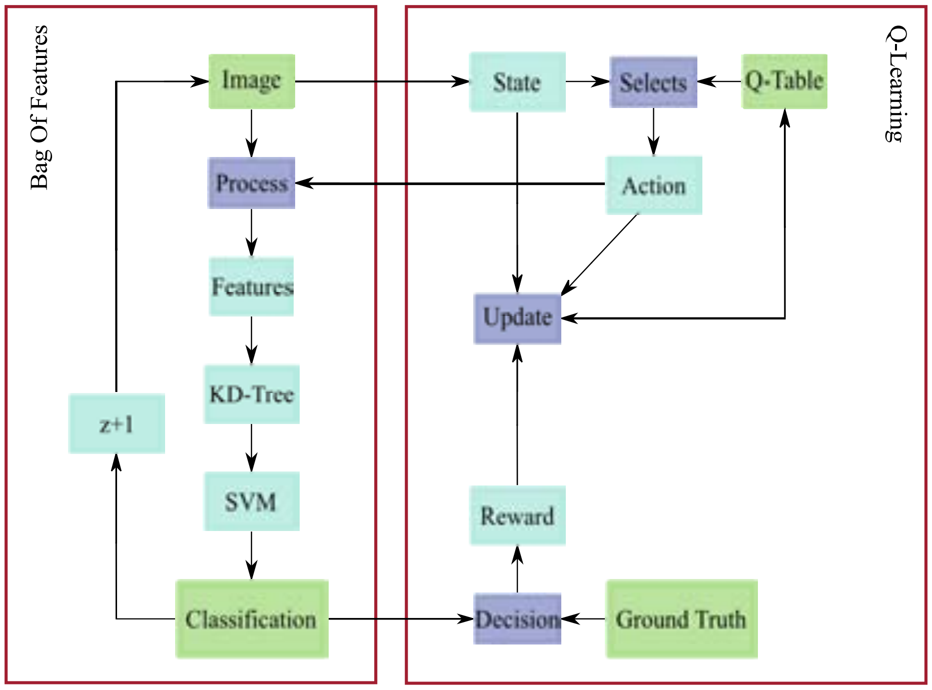


Figure 4.4: Proposed framework linking the bag of features and the Q-learning.

in Sect. 4.1.1 and do a clustering of the characteristics with k-means clustering to obtain the centroids. Each centroid is one state in the set of states.

Then, the last step to initialize the complete system is to define the set of actions A; in the current work the actions are the descriptors that were introduced in Sect. 2.1.2.

The second training process learns the δ and τ functions with the Q-table, which is modified for each iteration of the process. When the system converge the Q-table maps the states with the optimal actions. To do the training process, given an image (I) from QTTS, the agent needs to identify the state of the environment, so, using the state definition extracts the array that contains the characteristics of the image and then, obtains the centroid of the clustering with a NN-neighbours, this centroid is at the same time a concrete state s_z.

Then, the agent using the **Q**-table and the exploration-exploitation strategy decides the action (a_h) that is contained in A. The actions are descriptors, hence, it applies the descriptor a_h to the image I to extract the visual features. Thus, the process continues with the other steps of the bag of features, using the dictionary generates a histogram and applying the models obtains a label with the classification.

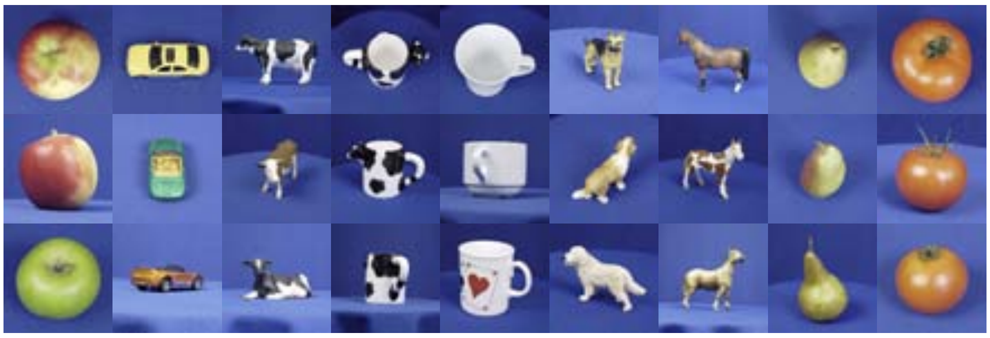


Figure 4.5: Illustration from ETH dataset (three images from the nine classes contained in the dataset).

When the system has a label from the classification, the system compares it with the ground truth to obtain the reward/punishment r_t through $\tau(s_z, a_h)$ as mentioned in Sect. 4.1.4. The agent updates the Q-table with the formula introduced in Sect. 2.2.2 for a nondeterministic environment and computes the error as explained in Sect. 4.3.

Finally, if the convergence criterion is achieved, the process ends, otherwise, the system extracts a new image from QTTS dataset and repeats all the aforementioned steps again.

For the testing time, given an image from the testing set, it extracts the state (as in the second step of the training) and finds the centroid. Finally, it applies to the image the selected descriptor, the agent returns an action that maximizes the expected reward for this states. Then, it classifies the object in the image. Figure 4.4 summarizes the complete process. The scheme is split up into two parts:

- *Left*: The bag of features, this scheme is used in the first training with the BoFTS to obtain the models. The second model uses the scheme to obtain the rewards/punishments. Finally, in testing time, it is used to obtain the classification.

- *Right*: The Q-learning process, this second part is used to train the Q-table, which is then used for the classification process.

4.5 Experimental results

This section provides results obtained with the proposed framework. The results have been validated with the ETH dataset (Fig. 4.5) split up into nine classes: apple, car, cow, cowcup, cup, dog, horse, pear and tomato.

We have used 45 images per class, which were split up into three sets:

- 15 images for training the models (BoFTS).
- 15 images for training the Q-table (QTTS).
- 15 images for testing.

The first training step builds a model using the images from the BoFTS. The output of the second learning process is the Q-table from the images in the QTTS. The experiments have been repeated fifteen times (each time with a new Q-table that was initialized by zero). The Q-table is stable when the convergence criterion is achieved, Sect. 4.2 Eq. (4.5).

Given an image from the testing set, the system extracts the state using the state definition and the k-means to find the concrete state. Then, it selects the action by searching in the Q-table the maximum value in this state. Sometimes the Q-table does not have a unique maximum for a state, for this reason, 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 the classification rate using always the same action (descriptor). In other words, the usual bag of features approach. Table 4.1 shows the performance for each descriptor. The set of descriptors that are in the set of actions A are: Spin, SIFT, SURF and PHOW.

Table 4.1: 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%

The best result using a single descriptor as an action for the ETH dataset is PHOW with a performance of 74.81%. The proposed Q-learning scheme has been evaluated using two different exploration-exploitation strategies (the same result is reached in both cases). The first experiment is using the Q-learning with an ε-greedy exploration-exploitation trade-off. Table 4.2 summarizes the results for different levels of convergence. Figure 4.6 shows the confusion matrix. The second experiment was done with the exploration-exploitation strategy proposed in Sect. 4.3. In this case we show the performance given a value of the converge in Table 4.3.

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 4.4, which shows the number of iterations needed for each exploration-exploitation

Table 4.2: Results for different convergence values, when the $\varepsilon - greedy$ strategy is used in the exploration-exploitation trade-off.

Convergence value	**Number of iterations**	**Classification ratio**
2%	7.149	71.49%
1.8%	13.817	77.77
1.6%	20.386	80.55
1.4%	26.914	81.48
1.2%	33.434	81.48

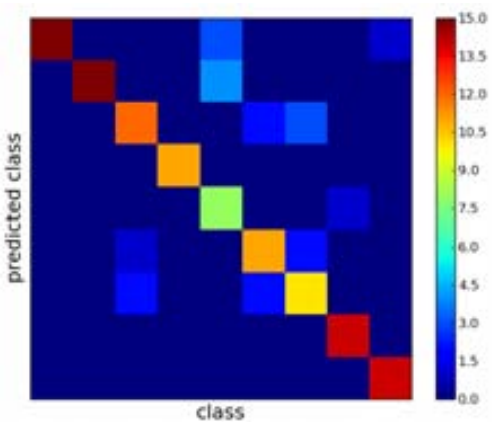


Figure 4.6: Confusion matrix with $\varepsilon = 0.5$ exploration-exploitation strategy; it achieves a performance of 81.48%.

strategy. The usual strategy ($\varepsilon - greedy$) needs 33.434 iterations, more than 1.000 iterations to arrive at the same convergence than using the history of the error strategy.

4.6 Discussion

This chapter proposes a novel method to learn the best descriptor for each image introducing a new architecture joining the Reinforcement Learning and Bag of Features. The proposed approach has been validated using the ETH dataset. Its performance has been compared with respect to a single descriptor scheme. The best descriptor for the ETH dataset 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. Finally, a strategy for exploration-exploitation is proposed that makes the convergence faster than using random switches.

Table 4.3: Results for different convergence values, when the history of the error strategy is used in the exploration-exploitation trade-off.

Convergence value	**Number of iterations**	**Classification ratio**
2%	6.409	70.95%
1.8%	12.921	77.77
1.6%	19.320	77.77
1.4%	25.723	81.48
1.2%	32.124	81.48

Table 4.4: Number of iterations for each strategy using θ less than 10% of error in the Qtable.

Strategy	**Number of iterations**
$\varepsilon = 0.5$	33.434
History of the error	32.124

Chapter 5

Multi-Qtable

In Chapter 4, we propose a novel framework based on the bag of features approach and the reinforcement learning technique. The proposed framework learns the best descriptor for each image from the training dataset. As mentioned before in Chapter 3 the real problem is the state definition. In [25] a method for selecting the best descriptor for every image in the dataset is proposed. In order to select the best descriptor, several attributes of the image (e.g., colourfulness, roughness, shininess, etc.) are taken into account. Although interesting results are presented, their main drawback is the use of a supervised learning scheme where the authors select the descriptors with a subjective criterion. On the contrary, as presented in Chapter 4 our method learns the best descriptor for each image using a reinforcement learning scheme. The reinforcement learning is a simple learning method based on a trial and error strategy. This chapter presents two improvements from Chapter 4.

1. We propose to use several state definitions.

2. A multi-table scheme is introduced in order to exploit the best state definition for each image.

In summary, this work proposes a novel method to learn the best descriptor from a given set. In order to improve the performance, multiple state definitions are used. This scheme works with a BoF approach, and in concrete, the implementation uses a kd-tree in the second step and a support vector machine (SVM) in the fourth step.

The reminder of this chapter is organized as follows. The elements of the tuple of the reinforcement learning are proposed in Sect. 5.1. This section presents the set of states definition (Sect. 5.1.1) and the set of actions (Sect. 5.1.2). Then, in Sect. 5.2 the new framework is introduced. Next, the Sect. 5.3 presents the different experiments and finally, Sect. 5.4 gives the conclusions.

5.1 Elements of the tuple of reinforcement learning

The agent of RL learns the action that maximizes the reward in each state. In order to do this, the RL uses the $\langle S, A, \delta, \tau \rangle$ tuple. This section describes the elements of the Markov decision problem tuple.

5.1.1 State definition

In this work a state is a representation of the image. There are several characteristics of the images that can be used as states and each state definition achieves different results. This section explains the four state definitions used to deal with the visual object classification. The first three state definitions (see below $L^*a^*b^*$ **based state definition, Entropy based state definition and Gradient based state definition**) use the scheme proposed in Fig. 3.2 (Sect. 3.2) that consists of a vector with 13 elements. The first image (Fig. 3.2(a)) contributes with one value, the second image (Fig. 3.2(b)) with 4 and the last image (Fig. 3.2(c)) with 8 elements, resulting in the vector with 13 elements. Then, the last state definition (see below **Histogram of interest point based state definition**) consists of a vector of 50 elements obtained from a representation of visual words.

The steps for computing these states are:

1. For each image in the training set, extract the vector of characteristics (applying one of the next state definitions).

2. Using the K-means algorithm build k clusters. The center of the clusters are the states ($S = \{s_z\}_{0 \leq z < k}$)

The four state definitions mentioned above are explained in this next section. The first state definition is based on $L^*a^*b^*$ color space, next the entropy and finally gradients. These three state definitions are based in the scheme showed in Fig. 3.2 and only extract some characteristics of the image. On the contrary, the last state definition needs some process to obtain the vector of the characteristics.

$L^*a^*b^*$ based state definition

This state definition uses the $L^*a^*b^*$ color space. This color space is obtained by converting RGB to XYZ and then, XYZ to $L^*a^*b^*$. This state definition has been already presented in Sect. 3.2.1.

Entropy based state definition

This state definition is based on the uncertainty of the information in the image. In this case, the vector of characteristics also has 13 elements (Fig. 3.2). This state definition uses the grey level representation and for each element of the partition image, extracts the entropy as follow:

$$E = -\sum_{i=1}^{N} p_i log_2(p_i), \tag{5.1}$$

where $p(i)$ is the histogram of image and N is the number of pixels in the given region.

Gradient based state definition

This state definition uses the grey level representation. The vector of characteristics has 26 elements (13 elements per each direction). For each element of the partition image computes the gradient in x and y direction. Then, the state vector is built with the median of gradient values contained in each image partition (see Fig. 3.2). Figure 5.1(b) shows the edges extracted using the gradient in x direction and Fig. 5.1(c) using the y direction.

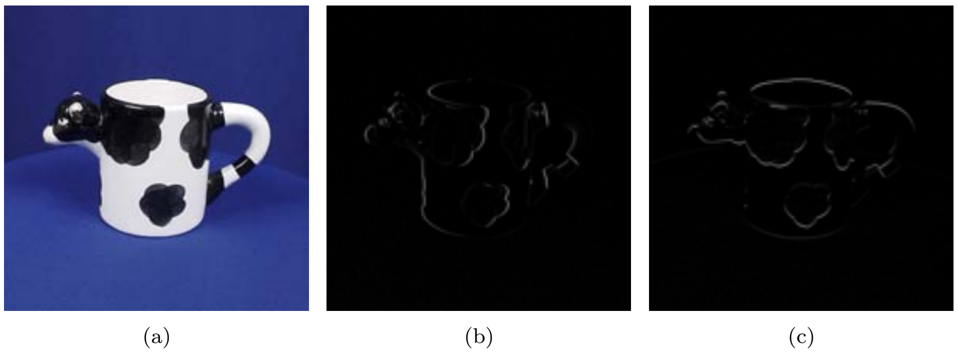


Figure 5.1: (a) Image from ETH dataset. (b) Image gradient in x direction. (c) Image gradient in y direction.

Histogram of interest point based state definition

This state definition uses all the descriptors of the set of actions to compute the vector. This state definition extracts a vector of characteristics containing 50 elements. The elements of this vector are obtained according to the five actions defined in the next section. For each action (descriptor), a dictionary of visual words is built using the first 2 steps of the classical BoF. First, the features are extracted and then, using these features a dictionary of ten visual words is built. The vector results from the process of concatenating the representations of the image using a histogram of ten visual words for each descriptor.

5.1.2 Action definition

In this work the actions came from a set of descriptors that the RL technique learns during the training process; as a result the best descriptor for each image is found. A large number of descriptors have been proposed in the literature in recent years [59]. In this work five frequently used descriptors were selected: SIFT, PHOW, SURF, Spin and C-SIFT (see Sect. 2.1.2).

5.2 Combination of Q-tables

This section presents the architecture proposed for selecting the best descriptor. Our visual object classification scheme aims at using the minimum information and improving the classification rate. The proposed architecture learns the best descriptor for each image. The proposed method is based on BoF and, in concrete, this method is focussed on the first step of the BoF.

In order to train and test our approach, the dataset is split up into three sets: BoF training set (BoFTS), Q-table training set (QTTS) and testing set. The learning process has two steps: The first training step is performed using the BoFTS where, for each descriptor from the set of actions a kd-tree (T_{a_h}) is built and a SVM is used to classify the objects [99]. The second training step is depicted in Fig. 4.4 (Sect. 4.4). Initially, the Q-table is initialized with a "0" in all the cells and then the process starts. Given an image from the QTTS it extracts the characteristics to find the state (s_z). The agent extracts the action (a_h) using the exploration/exploitation trade-off. Applies the action (a_h) to the image (s_z) and obtains the features of the image using the selected descriptor. To obtain the classification the process uses the features and the tree trained above (T_{a_h}, a_z) to obtain a histogram and then, uses the histogram of features and the SVM to return a label (the classification of the image). The agent compares the label of the image with the ground truth and obtains a reward/punishment (r_t). Finally, the Q-table is updated as indicated in Sect. 4.4, the process starts again with a new image from the QTTS.

After explaining the learning process, it should be noticed that in section 5.1.1 four different states definition have been proposed; hence, the next question is how to select the right state definition. A $brute - force$ idea could be to concatenate all the state definitions; however, as will be shown in section 5.3 this strategy not always reaches the best performance.

This section proposes a method to learn the best descriptor from only one state definition for each image (e.g., states 5.1.1). Therefore, the process proposed above is repeated four times (one for each state definition) and it finishes with a Q-table for each state definition. Now the objective is to decide the action for each image using the information from the Q-tables. In this work, we propose a simple voting strategy for combining the four Q-tables. The strategy consists in selecting the best action proposed by the Q-tables for each image. The best action from the Q-tables is the action that maximizes the reward (Fig. 5.2 illustrates this multi-table scheme).

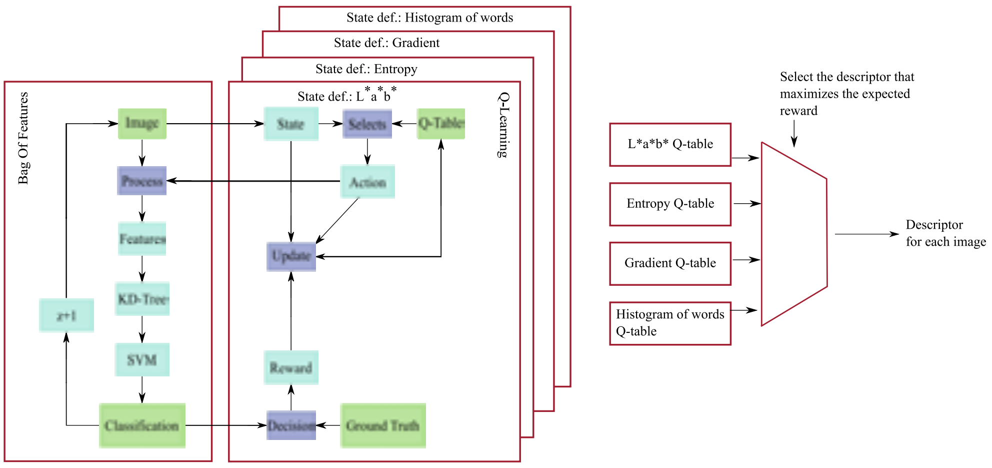


Figure 5.2: Multi-table combination.

5.3 Experiments

The proposed method has been evaluated using two different datasets (ETH [51] and COIL [62]). The evaluation framework compares the results using:

- A unique descriptor for the whole dataset.
- All the descriptors concatenated in a single one.
- The RL-based approach presented in Chapter 4.
- The RL-based approach with different state definitions.
- All the states concatenated.
- The information provided by the combined Q-tables (Fig. 5.2).

These experiments have been performed with the first dataset (ETH dataset) and then repeated with the second one to validate them.

In Chapter 4 a single Q-table is considered and a state definition using image content statistics is proposed. This state definition is based on the use of the grey scale image information together with additional image detectors as presented below. The resulting state (vector) contains 17 elements obtained from the following four groups.

1. Mean, standard deviation and the median grey values for each image (Fig. 4.1) (this contributes with three elements to the vector).
2. Mean, standard deviation and the median grey values for each of the four equally sized squared blocks (resulting in 12 elements in the vector).

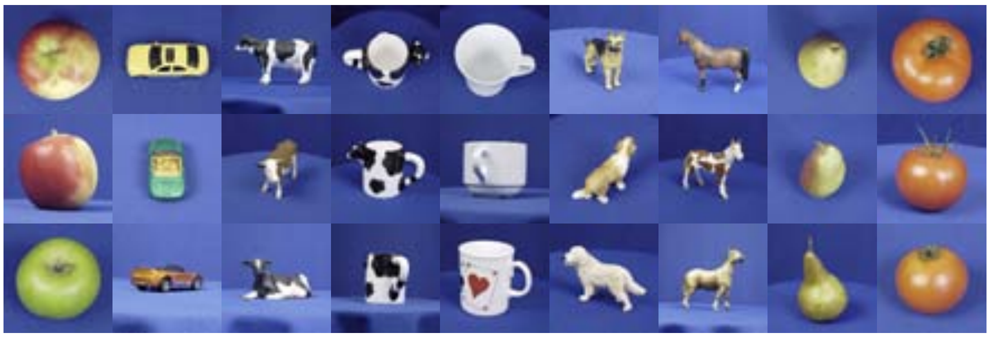


Figure 5.3: Some of the objects contained in the nine classes of ETH dataset.

3. The number of corners obtained from Harris corner detector [35] (it contributes just with 1 element to the vector).
4. The number of blobs obtained converting grey scale to black and white using OTSU threshold [68], and then, a labelling algorithm with a connectivity of 8 neighbours [32] (this is the last element in the vector).

The first experiments were performed using the ETH dataset. Figure 5.3 shows the nine classes of the dataset (i.e., apple, car, cow, cow-cup, cup, dog, horse, pear and tomato). In order to do the experiments, we randomly select 45 images from each class. As mentioned above, the dataset was split up into three sets: 15 images for training the BoF, 15 images for training the Q-table and finally, 15 images for testing. The value of γ is 0.9 and the process uses a ε-greedy strategy of exploration-exploitation trade-off ($\varepsilon = 0.2$).

In order to compare the results obtained using the proposed approach, first of all, the performance for each descriptor and the combination of all them are computed (see Fig. 5.4). It can be appreciated that the best single descriptor is PHOW with a performance of 74.81% of correctly recognized objects and, using the combination of all the descriptors, this ratio is increased up to 76.3% of classification. In this work, the *oracle* means the best performance that can be reached only if each image were described by its best descriptor. In this particular dataset, the oracle improves the results in 19% reaching 95.6% of performance.

Figure 5.5 shows the performance using the proposed RL based scheme with the different state definitions. The best performance is obtained using the $L^*a^*b^*$ state reaching 82.2% of classification rate. Figure 5.6 summarizes the proposed evaluation. The first two bins are simple BoFs, the first bin corresponds to the result obtained if the best single descriptor (PHOW 74.81%) is considered; the second one corresponds to the results obtained when the whole set of descriptors is considered together; it reaches 76.3% of classification rate. Next, the result from the best single state definition, in this case the $L^*a^*b^*$, is presented. Then, the next two bins correspond to the results obtained concatenating all the states presented in Sect. 5.1.1 and the results obtained combining the states using the strategy presented in Sect. 5.2. In

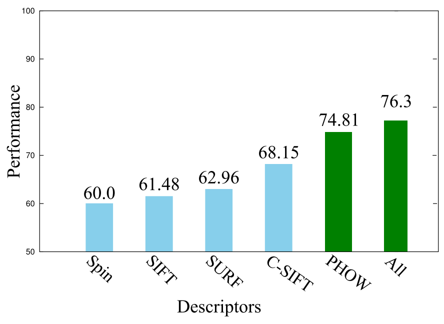


Figure 5.4: Performance of single descriptor and a combination of all descriptors for ETH dataset.

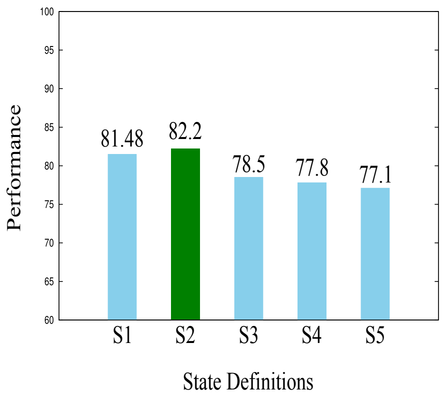


Figure 5.5: Performance using BoF with RL and different state definitions for ETH dataset. S1: state definition from Chapter 4. S2: $L^*a^*b^*$. S3: Entropy. S4: Gradient. S5: Histogram of interest point.

the case of state concatenation an 82.2% of classification rate is obtained while with the proposed approach an 83.7% is reached. Finally, the last bin corresponds to the result from the oracle in this dataset. Fig. 5.7 depicts the performance D1, D3 and D5 of Fig. 5.6 as confusion matrices. The first confusion matrix is obtained using the PHOW descriptor (Fig. 5.7(a)). Figure 5.7(b) shows the performance using BoF with RL and $L^*a^*b^*$ state definition. Figure 5.7(c) shows the results obtained when the proposed combination of Q-tables is used.

In order to validate the proposed approach, the experiments are repeated with a second dataset. This dataset is COIL [62]. The COIL dataset contains 100 classes. Fig. 5.8 shows one image per class. Each class contains 45 images and was split up into three sets: 15 images for training the BoF, 15 images for training the Q-table and 15 images for testing.

The performance of BoF using a single descriptor is depicted in Fig. 5.9 where the best option is PHOW with 98.3% of classification rate. Figure 5.10 shows the performance for each of the state definition described above (section Sect. 5.1.1) and

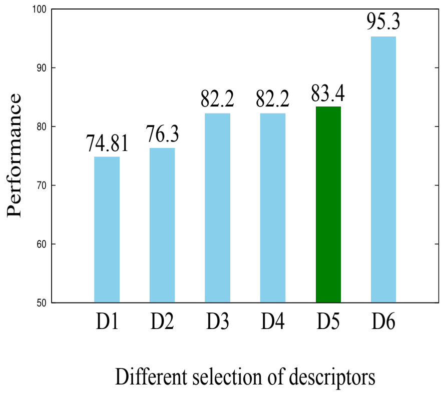


Figure 5.6: Comparison of different methods to select the descriptors for ETH dataset. D1: PHOW descriptor. D2: All descriptors concatenated in a single vector. D3: The best descriptor selected by RL with $L^*a^*b^*$ as state. D4: The best descriptor selected by RL with a concatenation of states: S2, S3, S4 and S5. D5: The proposed multi-table approach. D6: Oracle.

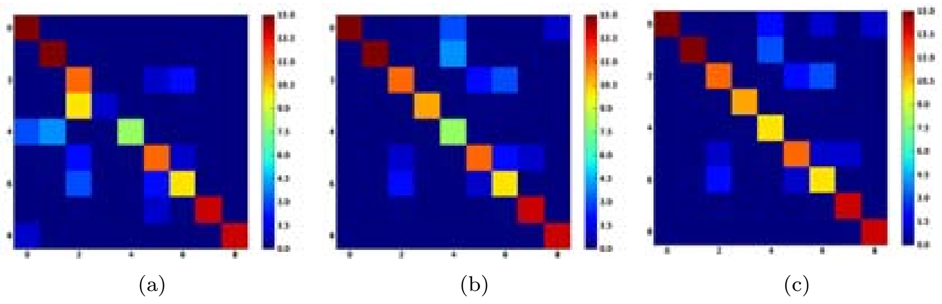

(a) (b) (c)

Figure 5.7: Different confusion matrices for ETH dataset: (a) Using only the PHOW descriptor (74.81%). (b) Using $L^*a^*b^*$ state definition (82.2%). (c) Using the proposed approach (83.7%).

Fig. 5.11 summarizes the proposed evaluation. It can be appreciated that in this case the best state definition corresponds to the gradient, which reaches a 98.8% of classification rate. It can be seen that the combination of Q-tables reaches 99.0% of classification rate (see Fig. 5.11).

Fig. 5.12 shows the confusion matrices with the best results. Figure 5.12(a) shows the confusion matrix of PHOW as a descriptor (98.3% of classification rate). The second confusion matrix corresponds to BoF with RL and gradient as a state definition, in this case the classification rate is 98.8%. Finally, Fig. 5.12(c) shows the confusion matrix for the proposed approach that combines multiples Q-tables (99.0% of classification rate).

In this second experiment the behaviour of the approaches are similar, because all state definitions improve the classification rate with respect to the single descriptor

Figure 5.8: Illustration from COIL dataset, one object per class is depicted.

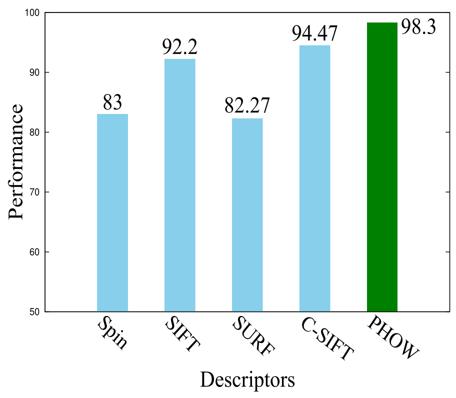


Figure 5.9: Performance of BoF using a single descriptor for COIL dataset.

case. Also, the combination of all descriptors reaches the same classification rate as the combination of Q-tables, but we work with less information from the image, thus, we use less computer resources (space and time). Note that although there is still space for improving, the obtained result is quite near to the best result, which is only reached by the oracle (99.7% of classification rate).

5.4 Discussion

This chapter presents a novel framework for visual object classification. In particular, it is focussed on the selection of the best image feature descriptor. It is based on the combined use of a bag of features scheme together with a reinforcement learning technique, implemented trough the Q-learning approach. Note that any visual classification method (based on image descriptors) can substitute for BoF in this approach.

The proposed method combines different state definitions in a multi-table strategy that guarantee the selection of the best action (image descriptor). Experimental results using two public datasets and comparisons with state of the art are provided showing the performance of the proposed approach.

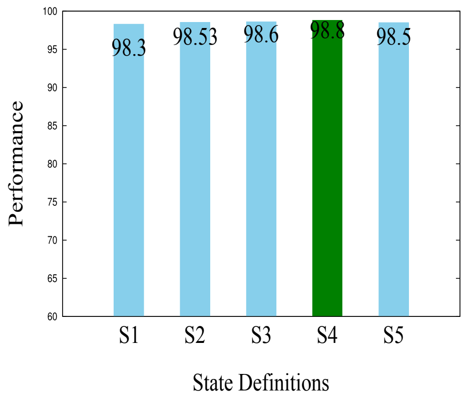


Figure 5.10: Performance using BoF with RL and different state definitions for COIL dataset. S1: state definition from Chapter 4. S2: $L^*a^*b^*$. S3: Entropy. S4: Gradient. S5: Histogram of interest point.

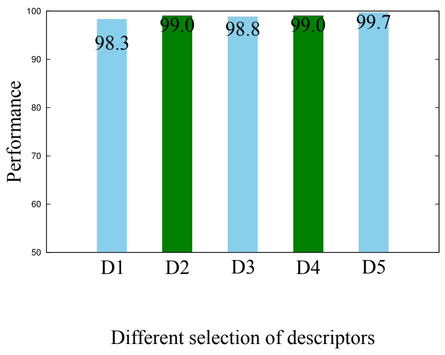


Figure 5.11: Comparison of different methods to select the descriptors for COIL dataset. D1: PHOW descriptor. D2: All descriptors concatenated in a single vector. D3: The best descriptor selected by RL with Gradient as state. D4: Multi-table approach. D5: Oracle.

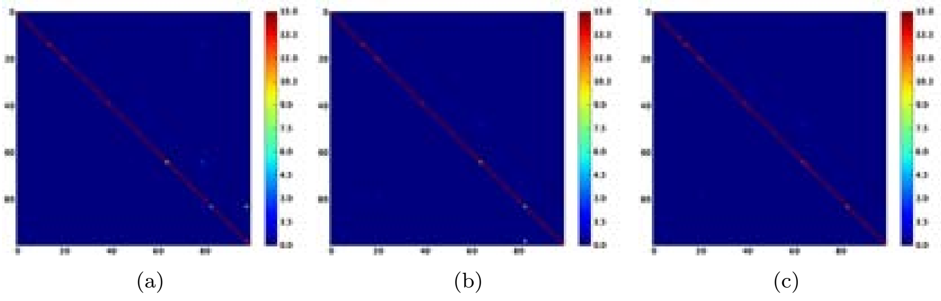


Figure 5.12: Different confusion matrices for COIL dataset: (a) Using only PHOW as a descriptor (98.31%). (b) Using gradient as a state definition (98.8%). (c) Using the proposed method (99.0%).

Chapter 6

Opening the loop of reinforcement learning

In general, closed-loop techniques are used to continuously propose the model according to the feedback. This kind of approaches are widely used in control theory where there are two modules (controller and sensor) to introduce in the system the feedbacks.

On the other hand, the reinforcement learning is a closed-loop technique based on the reward/punishment signal to introduce the feedback in the system. In concrete, in Q-learning algorithm, the reward signal is used to update the Q-table in each iteration. Actually, the closed-loop is not the unique solution for the reinforcement learning, there are some contributions based on an open-loop strategy or apprenticeship learning. This chapter proposes an apprenticeship learning technique applied in a video retrieval application.

The rest of this chapter is organized as follows: Sect. 6.1 summarize of some applications of apprenticeship learning. Then, Sect. 6.2 describes the video retrieval problem and introduce a possible solution. After that, we propose an apprenticeship learning as a new solution to the video retrieval problem in Sect. 6.1. This apprenticeship learning is improved reducing the interactions with the human expert. The last contribution in this chapter is presented in Sect. 6.4 where the deterministic environment is changed to a non-deterministic one. Finally, Sect. 6.5 present some experimental results and discussions are given in Sect. 6.6

6.1 Apprenticeship learning

The apprenticeship is a reinforcement learning technique with an open-loop (learning from expert demonstrations). In the open-loop approach, there is a human teacher to improve the model. The framework can be modified by the teacher in two ways:

i) modify the reward; and *ii*) modify the selected action.

J. Zico *et al.* propose a method to use the apprenticeship learning in a quadruped motion [46]. In that paper, the authors propose a framework for the robot to learn the motion for walking doing the experiments with the two apprenticeship learning approaches. The paper split in two the apprenticeship learning paradigm: high vs. low level. The high level uses the expert to select the optimal policy $- \pi$ function. On the contrary, the low level uses the expert to suggest the best action to move the action s to s'.

On the contrary to the previous work, the next examples were evaluated just using a simulator. A. L. Thomaz *et. al* propose a kitchen simulator where the human expert modifies the reward signal to the agent, which learns how to do a cake [94]. Then, A. L. Thomaz and C. Breazeal present a study of the human teacher and the interaction with the agent [93]. Also, this second paper introduces a new possibility of interaction with the kitchen simulator. In this case, the human teacher can guide with the mouse the object that the agent needs in the next action. An extension of this study is presented in [95] where A.L. Thomaz *et al.* propose a new study of the reward signal. These experiments investigate the best way to teach the reinforcement learning.

An other example of interactive reinforcement learning is introduced in [42] by W. B. Know and P. Stone. They propose a tetris simulator and use the human teacher to return the reward in the system. An extension of this work has been presented in [43] where they combine the usual reward SARSA(λ) signal with the human interaction by eight different formulas. In all the cases the guidance of the human teacher is for each iteration. In this chapter we propose a method to reduce the number of interactions with the human teacher.

6.2 Video retrieval with reinforcement learning

A video retrieval system is a computer system for retrieving frames from a large dataset. The retrieval system compares a query frame f_i with each dataset frame F=(f1, f2, ..., ft) and returns the top L similar dataset frames. The retrieval system identify the relevant or nonrelevant frames and modify the weight of the frame. The process is repeated n times (in our particular case 5 times). The selection of the L similar images are based on some characteristics of the image. Usually, the characteristics are extracted by the combination of one detector and one descriptor. In the current work the selected descriptor is CSIFT [98] that represents the color of the image with a pyramid.

In the video retrieval, as in the bag of features, there are some points where the expert needs to use the experience to select the appropriate algorithm. In previous chapters, the reinforcement learning was a tool to select the descriptor. In the current chapter the reinforcement learning is used to select the combination of search algorithms. P.Y. Yin *et al.* propose a framework to learn the best combination of algorithms [104]. The proposed framework uses the reinforcement learning technique and specifically the Q-learning algorithm [102]. The elements of the tuple to use the reinforcement learning are defined as follow:

- The states are defined as a combination of 3 variables: index of the frame (x), the number of iterations (y) and the last action(z), $S_{x,y,z}$.
- The actions are the algorithms for frame retrieval. To maximize the result, the agent selects the best action to maximize the reward provided by $\tau : S \times A \rightarrow R$.
- The transition function is deterministic because, given a frame, the state $s_{x,y,z}$ and the action a_h, maximizing the reward, the new state $(s_{x,y+1,h})$ is obtained.
- The reward/punishment function $\tau(s_{x,y,z}, a_n)$ is deterministic because, given a frame, the state $s_{x,y,z}$ and the action a_h that maximizes the reward, the value of reward/punishment is always the same.

In this framework, the delta and reward/punishment functions are deterministic, thus, the environment is deterministic and the Qlearning uses the formula previously introduced in Chapter 2.2.2:

$$\mathbf{Q}(s_n, a_n) = r(s_n, a_n) + \gamma \max_{a'} \mathbf{Q}(s_{n+1}, a'), \tag{6.1}$$

to update the Qtable. A deterministic world assures the convergence of the process, but in concrete in this problem, the convergence is very slow. Hence, we propose a new strategy to reduce the computational cost.

6.2.1 Convergence

In a deterministic world, as can be seen in Sect. 2.2.2, the formula used to reach the convergence is:

$$|\mathbf{Q}_{n+1}(s, a) - \mathbf{Q}_n(s, a)| = \gamma \triangle_n. \tag{6.2}$$

In our framework, when the state changes from $s_n \rightarrow s_{n+1}$, really the only change is the frame to analyse. In reinforcement learning, one loop can change all the path of the convergences. Then, due to the added difficulty of our framework, we need to modify the formula of deterministic world using a new formulation. In [104], the authors use the following equation for convergences:

$$E(s_{x,y,z}) = -\sum_{h=1}^{u} p(a_h|s_{x,y,z}) \log_2(p(a_h|s_{x,y,z})), \tag{6.3}$$

but this formula is very time consuming. We propose to solve the problem of the convergences faster by using the following equation:

$$\begin{aligned} |\mathbf{Q}_{n+1}(s,a) - \mathbf{Q}_n(s,a)|_{it} < \gamma\triangle_{n,it} \\ it = 0,1,2..w \end{aligned} \tag{6.4}$$

which uses a sliding window to control the history of the error. The size of the sliding window (w) provides a restricted convergence.

6.3 Apprenticeship learning in video retrieval

This method proposes to use the apprenticeship learning as a tool to improve the video retrieval ratio. As introduced in Sect. 6.1, the reinforcement learning framework can be opened to introduce an expert. In general, this expert is used to modify the reward/punishment function, but in the current work, the expert modify the selection of the action (see illustration in Fig. 6.1). Actually, the framework introduces a novel idea to reduce the interaction with the expert using the history of the error in the Qtable.

6.3.1 Expert introduction

In the previous section the human expert is introduced to modify the reward/punishment function. As can be seen in Fig. 6.1 the interaction of the expert with the framework can be possible in two places: *i*) by modifying the action selection; and *ii*) modifying the reward value. Usually, in these cases, the expert change the value of the reward or punishment for other one according to some criteria. In other cases, the framework calculates a relation of the expert value and the value that gives the system after the iteration to update the Qtable.

The current work is focused on modifying the action selection by the expert. The criteria selected by the expert in this work is to select the action that was less selected. Thus, the expert provokes most exhaustive search modifying the π function and increases the iterations of the algorithm.

6.3.2 Minimum interaction with the framework

It is difficult to have a human expert all the time interacting with the framework. Usually, in frameworks similar to the current one, the expert spends hours to teach the agent. On the contrary to previous approaches, the proposed apprenticeship learning try to minimize the number of interactions. The expert only interacts with the framework when it is necessary. This new control module is inspired by a Proportional-Integral-Derivative controller (PID). In a PID controller [67], the Derivative part takes care of the "speed" of the error in one point:

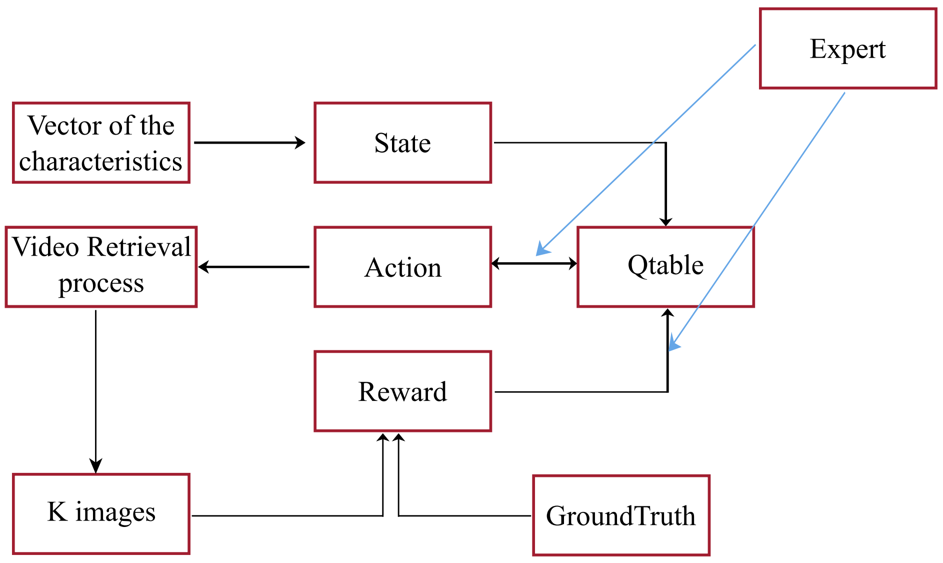


Figure 6.1: Illustration showing the places where the expert can do the interactions with the framework.

$$D = K_d \frac{de(t)}{dt}. \tag{6.5}$$

The Integral part takes into account the “history” of the error (the magnitude and also the duration):

$$I = K_i \int_0^t e(t)dt. \tag{6.6}$$

Also, the expression (Eq. (6.6)) deals with error spikes. Finally, the Proportional (Eq. (6.7)) multiplies the error by a constant.

$$P = K_p e(t). \tag{6.7}$$

The semi-automatic algorithm introduces two new variables: Error and Memory. The interactive agent behaves like a “closed-loop” while the error is below a given threshold (including error spikes):

$$D \approx \frac{\Delta Error}{\Delta Iterations} = \varepsilon. \tag{6.8}$$

The memory is a windows accounting for the last le errors:

$$I \approx \sum_{i}^{le} \varepsilon. \tag{6.9}$$

The system asks for a teacher intervention (behaves as an “open-loop”), if and only if, the error is higher than a given threshold and it is not an spike.

6.4 Apprenticeship with non-deterministic environment

The previous sections introduce the video retrieval with reinforcement learning. This framework is improved in cost time by reducing the training time. Finally, the loop is open to use an expert for improving the results. In this section, we propose to modify the definition of the elements of the tuple to be in non-deterministic environment.

In order to have a non-deterministic environment we modify the state definition. In the problem introduced above, each image is defined by a unique state and this state is unique for this image. Hence, in training time we can reach a 100% video retrieval ratio. But, in testing time, the system does not reach this percentages. This section proposes to have a state that include some images. The most easy way to obtain states with different images is using the K-means clustering. Thus, using the training dataset we extract K clusters and use the index of the cluster as state, in other words, the state was defined as a combination of 3 variables: index of the image (x), the number of iterations (y) and the last action(z) but now, change the index frame to the index of the clustering center.

6.5 Experiments

The experimental results are obtained using the MIPRCV-WP6 Video Retrieval Benchmark dataset [1] (table 6.1). We use eight concepts: building, crowd, outdoor, office, person, sky, urban and vegetation.

For each frame, the image is represented by a histogram and a descriptor is used to describe the interest points. In this work, we use a pyramidal SIFT-based color descriptor [98]. We use the set of actions (A) that consist consist of: Baseline, Rocchio and Relevance Score [23, 72] and the set of states (S) is introduced in Sect. 6.2.

In all the experiments, the process of relevance feedback is repeated five times and we want to obtain the ten most similar frames of the dataset with respect to the given query. In order to obtain the results we apply different strategies: *i*) just only using one algorithm; *ii*) using the algorithm introduced by P.Y. Yin *et al.* [104]; *iii*) adaptation of [104]; *iv*) apprenticeship learning approach. Finally, the experiment *iv*) is repeated for a non-deterministic environment.

[1] http://www.vision.uji.es/consolider/public/campaign2.php3

dataset	Labels	Number of Samples
Building	2	1585
Crowd	2	1575
Office	2	1556
Outdoor	2	1460
Person	2	1478
Sky	2	1541
Urban	2	1413
Vegetation	2	1508

Table 6.1: MIPRCV-WP6 Video Retrieval Benchmark.

dataset	Closed-loop [104]	Proposed convergence	Open-loop	Non-deterministic environment
Building	94.67	95.31	97.61	95.04
Crowd	96.06	96.3	97.66	95.0
Office	96.11	96.18	96.42	95.6
Outdoor	96.58	98.6	100	97.58
Person	93.67	95.83	99.28	95.22
Sky	94.31	96.56	99.50	95.21
Urban	94.63	94.91	96.91	94.98
Vegetation	94.68	95.74	98.26	94.25

Table 6.2: Percentage reached in the last iteration of the training process.

Incrementally, the experiments have been modified to improve the results. First, we use the algorithm presented in [104], where the image retrieval is working with Q-learning. This algorithm has a slow convergence, we decide to use a usual convergence for Q-learning [102]. As can be seen in the values depicted in Table 6.3, the performance improves with our convergence in all the cases. Finally, we propose to improve the results by using an expert teacher. Table 6.2 shows the values for each algorithm and Fig. 6.2 depicts the results of the training process for each dataset using a deterministic environment. Table 6.3 shows the percentage values reached in the last iteration of the testing process and Fig. 6.3 depicts the video retrieval ratio in testing time. In all the cases, the video retrieval ratio increase using the human expert (Sect. 6.3).

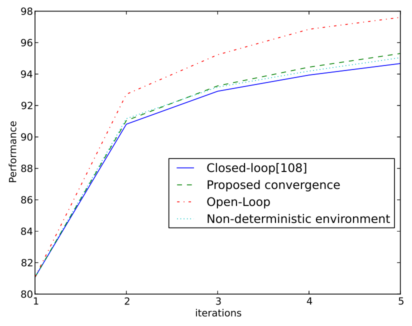
6.6 Discussion

In [104], the authors propose a method to uses the Q-learning algorithm for finding the best combination of relevance feedback for each iteration. Our approach is based on that framework and propose a new formula for the convergence to reduce the consuming time. Also, we open the loop to improve the results and we propose a new method of the interaction of the expert with the framework. This new method can

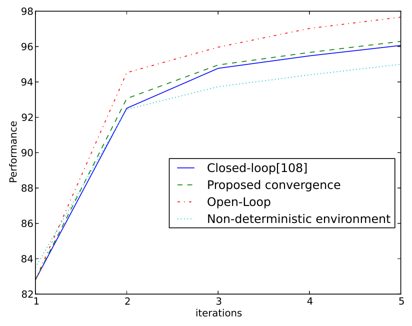
dataset	Closed-loop [104]	Proposed convergence	Open-loop	Non-deterministic environment
Building	94.78	95.12	95.38	95.31
Crowd	96.18	96.38	96.45	96.26
Office	95.88	96.21	96.31	96.32
Outdoor	96.93	98.35	98.74	98.76
Person	93.3	95.4	96.4	95.81
Sky	93.95	96.41	96.60	96.56
Urban	95.03	95.71	95.5	95.67
Vegetation	95.24	96.0	96.6	95.8

Table 6.3: Percentage reached in the last iteration of the testing process.

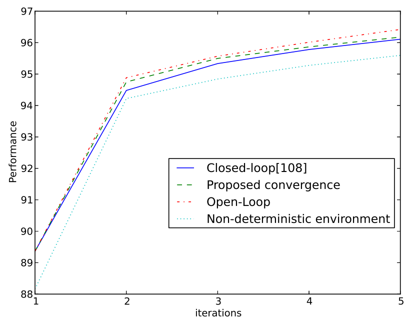
modify the action selected but only when the system needs help.



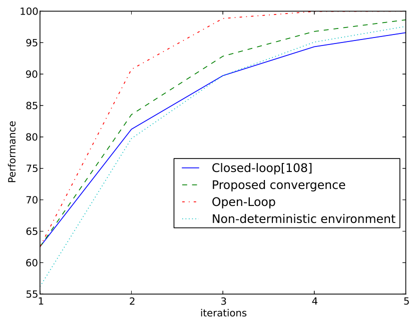
(a) Building dataset.



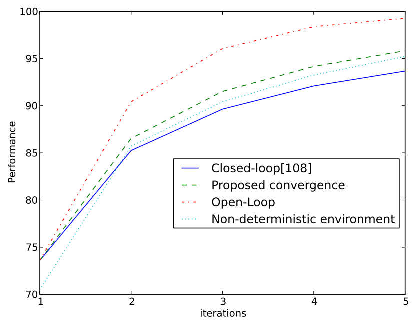
(b) Crowd dataset.



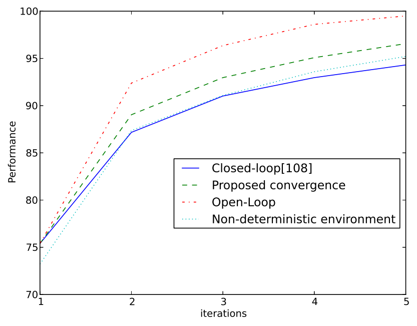
(c) Office dataset.



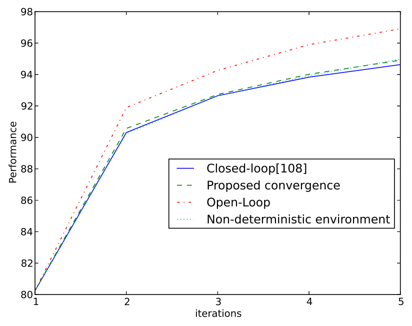
(d) Outdoor dataset.



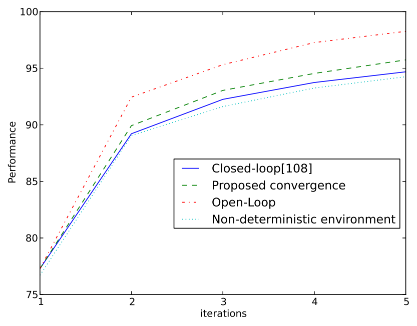
(e) Person dataset.



(f) Sky dataset.

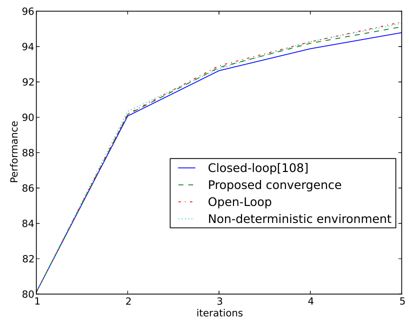


(g) Urban dataset.

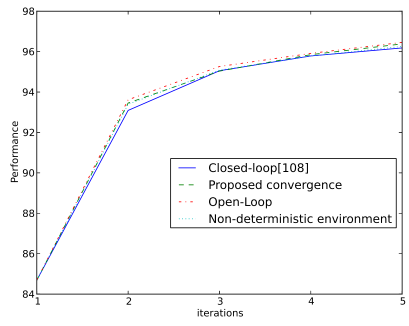


(h) Vegetation dataset.

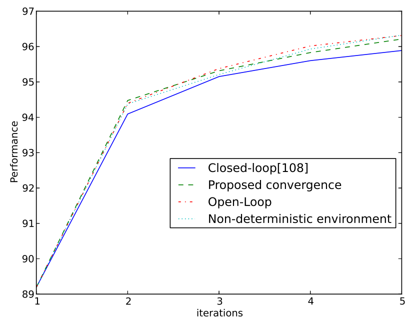
Figure 6.2: Performance evolution during the training process (values reached in the last iteration are depicted in Table 6.2).



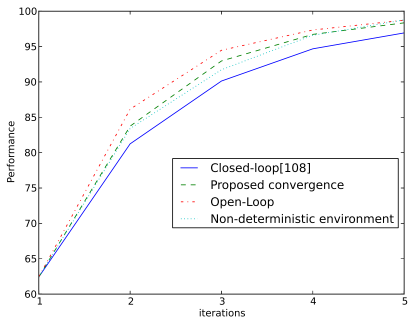
(a) Building dataset.



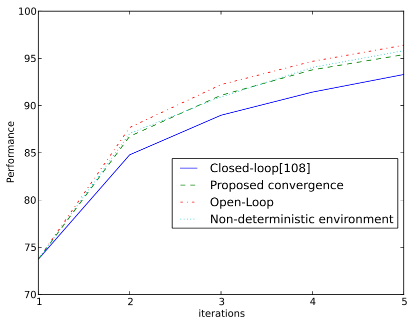
(b) Crowd dataset.



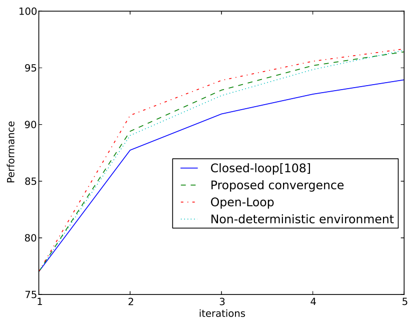
(c) Office dataset.



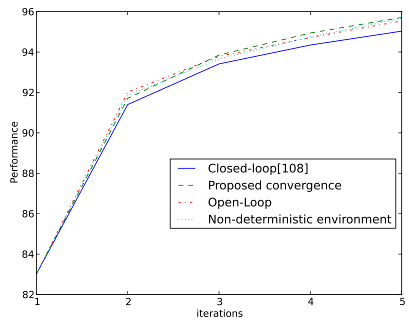
(d) Outdoor dataset.



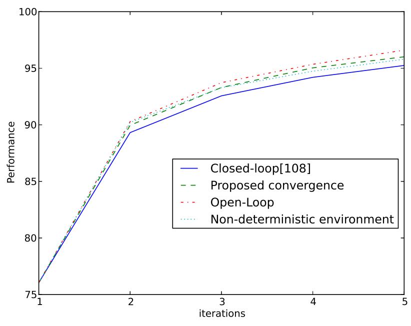
(e) Person dataset.



(f) Sky dataset.



(g) Urban dataset.



(h) Vegetation dataset.

Figure 6.3: Performance evolution during the testing process (values reached in the last iteration are depicted in Table 6.3).

Chapter 7

Summary and Conclusions

7.1 Summary

In this thesis we explore the advantage of the usage of reinforcement learning in computer vision. In concrete, we use the Q-learning algorithm in two cases. First, we introduce the Q-learning in visual object classification using the bag of features as well as for video retrieval. The contributions from the thesis are discussed per chapter in the following paragraphs:

Chapter 1: Introduction

The first chapter introduces the topic of this thesis. First, it details the two fields used to evaluate our framework. On the one hand, the bag of features approach where the proposed reinforcement learning is used to improve the classification performance. On the other hand, the video retrieval field used to evaluate the apprenticeship approach, which open the loop of the reinforcement learning approach.

Then, the motivations of the proposed work, together with main challenges, are introduced. Finally the objectives and structure of this thesis are presented.

Chapter 2: Related work

The second chapter introduces the *state of the art* it is split-up into two parts. The first part summarizes the visual object classification, in concrete, the bag of features. Also, it resumes a set of detectors (Sect. 2.1.1) and descriptors (Sect. 2.1.2). This set of detectors and descriptors is used in Chapter 3, 4 and 5. The set of detectors is based on corners, such as Harris corner detector, or based on blobs, such as scale-invariant feature detection, or fast Hessian detector. In the descriptors set, we summarize the scale-invariant feature transform, pyramid histogram of visual words, color SIFT, speed-up robust feature and the spin image.

The second part of Chapter 2 summarizes the machine learning in computer vision. This section is focused on Markov decision process and reinforcement learning to solve it. The reinforcement learning is a technique and the most popular algorithm is the Q-learning algorithm. This entire theoretical introduction is to use the Q-learning algorithm in computer vision as can be seen in the last part of this chapter where we summarize a set of examples. We present some examples of contrast adaptation, image segmentation, edge detection, e-learning, image retrieval, behavior recognition, face recognition, and finally, object recognition.

Chapter 3: Problem definition

The third chapter is focused on the definition of the states and the actions. The challenge in reinforcement learning is how to define the four variables of the tuple that define the problem. Also, we know that exist an intrinsic relation between the states and actions definition. This chapter tries to create a model to select the action given a state without using learning algorithm.

To create a model, in this concrete work, the set of actions is defined for the descriptors and the unique problem is how to do the state definition. It proposes a set of states definition based on characteristics of the image ($L^*a^*b^*$ color space, entropy and gradient). For each state definition, we propose a reduction of this information in a vector to simplify the problem. Finally, to build the lineal model, we propose two strategies: *i*) dimensional reduction by principal component analysis; and *ii*) using common vectors.

Chapter 4: Descriptor's selection

In previous chapters, we introduce the bag of features to classify the objects and the Q-learning algorithm as a reinforcement learning technique. In this chapter, we propose a framework to use both techniques for learning the best descriptor. Thus, we propose a definition for the elements of the tuple to adapt the Q-learning to the bag of features (set of states, set of actions, δ function and τ function). Also, in this chapter, we introduce the convergence problem with a non-deterministic environment and propose a possible solution. Then, we propose a new exploration-exploitation trade-off based on the control of the error in each iteration to switch the strategy. Finally, we propose the framework using the bag of features, and the Q-learning with all the improvements.

Chapter 5: Multi-Qtable

This chapter is an extension of the previous one. As mentioned before, the most difficult problem to use the reinforcement learning technique is to define the tuple, and in concrete, how to define the set of states and actions. In this work, we have fixed the set of actions (a set of descriptors), thus, the problem is how to define the states. Also, if the state definition is wrong the algorithm does not converge. We find a set of states definition that converges, but there are images that are only well classified with a concrete state definition. In order to obtain the best classification rate, we propose a system to select for each image the state definition using a simple voting system to select the Qtable.

This chapter introduces four state definitions based on the states proposed in Chapter 3 and 4 ($L^*a^*b^*$ color space, entropy, gradient and histogram of interest point). In this chapter, the color SIFT is introduced in the actions set. Finally, we propose this new framework and a set of experiments to show the improvements.

Chapter 6: Open loop of reinforcement learning
This is the last chapter and uses the video retrieval problem to validate the proposal. This chapter summarize the *state of the art* of apprenticeship learning to open the loop of the reinforcement learning in video retrieval. A system to introduce the human expert to improve the Q-learning algorithm is proposed. This system only interacts with the Q-learning when the error becomes large to modify the trend. Finally, we propose to modify the tuple definition to change the deterministic to non-deterministic environment.

7.2 Future directions

The thesis works in a novel topic that research includes the reinforcement learning technique in computer vision. After this four years of work in bag of features with Q-learning, we think in new areas to include the reinforcement learning, and also, new improvements for the frameworks proposed in this thesis.

This section introduces some new ideas to be implemented using the Q-learning. Firstly, and continuing with the work of this thesis, we propose the combination of descriptors for each image. Then, we propose to use the contributions of Chapter 6 in other fields such as the bag of feature approach. In other words, the intervention of the human expert in the bag of features approach to select the actions and/or the modification of the reward/punishment. Finally, a big problem to adapt the Q-learning in the bag of features approach is the tuple definition. The most difficult problem is the state definition (as mentioned before). During this thesis, we propose a set of state definitions based in the same idea, extract some characteristics of the image and reduce them by k-means. We need to do exhaustive experiments to find the best unsupervised learning algorithm to change for the k-means, or maybe, introduce the reinforcement learning to do the clustering. The last point in this section proposes a method to select the best classification algorithm by Q-learning.

Object Recognition for parts
The contributions presented in this thesis are based on full images. In this new line we propose to split up the given image in different parts and for each part to select the best descriptor. In this case, we propose to extract the characteristics from each part, which are used as states. Then, for each part it should apply the descriptor selected by the Q-learning.

Combination of descriptors
All works proposed in this thesis are based on the usage of only one descriptor for each image. Studying the images, we have noted that some images can only be correctly classified with the concatenation of a set of descriptors. Sometimes this concatenation is with two, three or more descriptors. Hence, as a future

research work we propose to explore the possibility of using the Q-learning to select the best combination of descriptors for each image.

In order to do this new implementation, we need to modify the τ function. If the image is correctly classified continues the process as in Chapter 3. On the contrary, if the image is not correctly classified, the process continues with the same image and uses a new Qtable to select the best descriptor, which is concatenated with the previous one. This new part of this process is repeated until obtaining a concatenation that recognize the image or when the concatenation uses all the descriptors of the set.

Expert introduction in Bag of Features

In Chapter 6, we propose a new framework of video retrieval using an apprenticeship learning system based on reinforcement learning with a human expert. In that chapter, we explain that the human expert can interact in two sites of the framework. First modifying the selection of the action and the other possibility is to modify the reward/punishment.

This apprenticeship learning can be also used in the framework proposed in Chapter 4. The human expert can modify the actions, the usual strategy in Q-learning is to select the action that maximizes the expected reward. The human expert can modify this selection with other strategies, a strategy to modify the action can be seen in [93].

The other interaction of the framework with the human expert can be to modify the reward/punishment. The τ function is very strict and the expert can introduce some flexibility depending on the difficulty of the image to be correctly classified.

Clustering on-line

The clustering approach is an unsupervised learning technique. We think that is possible to do a clustering on-line using the reinforcement learning technique. For a given image from the training dataset, the framework assigns a state depending on the similarity of the vector of the characteristics of the image. Then, the behaviour given an action can modify the assigned state. Thus, the system learns the best action for each image and also, it does the clustering about the behaviours of the image. We assume that this system can improve the classification rate because the values of the states are classified by the behaviour and not for the distance between the values of the characteristics of the image.

Meta-Classifier

On the contrary to the research topics mentioned above, in this last point we propose to explore a new framework where the Q-learning is outside the bag of features approach. In this case, the Q-learning can be used to learn the best classification system for each dataset. For example, the state definition is some characteristics of the dataset and the actions are the different approaches.

Appendix A

Publications

A.1 Journals

- M. Piñol, A.D. Sappa, R. Toledo. Adaptive Feature Descriptor Selection based on a Multi-Table Reinforcement Learning Strategy. *Neurocomputing* (accepted)

A.2 International Conferences and Workshops

- M. Piñol, A.D. Sappa, A. López, R. Toledo. Feature Selection Based on Reinforcement Learning for Object Recognition. In *Autonomous Agents and Multiagent system Workshop on Adaptive Learning Agents*. Valencia, Spain, June 4-8, 2012, pp. 33-39
- M. Piñol, A.D. Sappa, R. Toledo. Multi-Table Reinforcement Learning for Visual Object Recognition. In *International Conference on Signal and Image Processing*. LNEE 221, Springer Verlag, Coimbatore, India, December 13-15, 2012, pp. 469-480.

Bibliography

[1] A. E. Abdel-Hakim and A. A. Farag. CSIFT: A SIFT descriptor with color invariant characteristics. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1978–1983, New York, NY, USA, June 2006.

[2] M. Asada, S. Noda, and K. Hosoda. Action-based sensor space categorization for robot learning. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, pages 1502–1509, Osaka, Japan, November 1996.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proc. the European Conference on Computer Vision*, pages 404–417, Graz, Austria, May 2006.

[4] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, USA, 1957.

[5] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.

[6] B. Bhanu and J. Peng. Adaptive integrated image segmentation and object recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 30(4):427–441, 2000.

[7] R. Bianchi, A. Ramisa, and R. de Mántaras. Automatic selection of object recognition methods using reinforcement learning. *Advances in Machine Learning I*, 262:421–439, 2010.

[8] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning.* springer New York, 2006.

[9] L. Bo, X. Ren, and D. Fox. Depth kernel descriptors for object recognition. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, pages 821–826, San Francisco, CA, USA, September 2011.

[10] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance-based active object recognition. *Image and Vision Computing*, 18(9):715–727, 2000.

[11] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Rio de Janeiro, Brazil, October 2007.

[12] G. J. Burghouts and J. M. Geusebroek. Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113(1):48–62, 2009.

[13] P. Burt and E. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[14] H. Cevikalp, B. Barkana, and A. Barkana. A comparison of the common vector and the discriminative common vector methods for face recognition. In *Proc. World MultiConference on Systemics, Cybernetics and Informatics*, Orlando, Florida, USA, July 2005.

[15] M. Chitsaz and C. S. Woo. Software agent with reinforcement learning approach for medical image segmentation. *Journal of Computer Science and Technology*, 26(2):247–255, 2011.

[16] M. Crosier and G. D. Lewis. Using basic image features for texture classification. *International Journal of Computer Vision*, 88(3):447–460, 2010.

[17] F. C. Crow. Summed-area tables for texture mapping. *SIGGRAPH Computer Graphics*, 18(3):207–212, 1984.

[18] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshopon Statistical Learning in Computer Vision*, pages 1–22, Prague, Czech Republic, May 2004.

[19] A.L. Dahl, H. Aanæs, and K.S. Pedersen. Finding the best feature detector-descriptor combination. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 318–325, Hangzhou, TBD, China, May 2011.

[20] T. Darrell. Reinforcement learning of active recognition behaviors. Technical report, Interval Research Technical Report 1997-045, Palo Alto, CA, USA, 1997.

[21] T. Darrell and A. Pentland. Active gesture recognition using learned visual attention. In *Advances in Neural Information Processing Systems*, pages 858–864, Denver, CO, USA, November 1995.

[22] T. Darrell and A. Pentland. Active gesture recognition using partially observable markov decision processes. In *Proc. International Conference in Pattern Recognition*, pages 984–988, Vienna, Austria, August 1996.

[23] T. Deselaers, R. Paredesand E. Vidal, and H. Ney. Learning weighted distances for relevance feedback in image retrieval. In *International Conference on Pattern Recognition*, pages 1–4, Tampa, Florida, USA, December 2008.

[24] B. A. Draper, J. Bins, and K. Baek. Adore: adaptive object recognition. In *International Computer vision systems*, pages 522–537, Las Palmas, Gran Canaria, Spain, January 1999.

[25] I. Everts, J. C. van Gemert, and T. Gevers. Per-patch descriptor selection using surface and scene properties. In *Proc. the European Conference on Computer Vision*, pages 172–186, Florence, Italy, October 2012.

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

[27] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 380–387, San Diego, CA, USA, June 2005.

[28] L. Gagliardini, X. Tian, F. Gao, C. Qi, C. Chevallereau, and X. Zhao. Modelling and trajectory planning for a four legged walking robot with high payload. In *Social Robotics, Lecture Notes in Computer Science*, pages 552–561. 2012.

[29] J. M. Geusebroek, R. Van den Boomgaard, A. W. M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, 2001.

[30] A. Gherega, M. Radulescu, and M. Udrea. A q-learning approach to decision problems in image processing. In *The Fourth International Conference on Advances in Multimedia*, pages 60–66, Chamonix / Mont Blanc, France, April 2012.

[31] K. Häming and G. Peters. Learning scan paths for object recognition with relational reinforcement learning. In *Proc. of International Conference on Signal Processing, Pattern Recognition and Applications*, Innsbruck, Austria, February 2010.

[32] R. M. Haralick and L. G. Shapiro. *Computer and robot vision.* Addison-Wesley, 1992.

[33] M. T. Harandi, M. N. Ahmadabadi, and B. N. Araabi. Face recognition using reinforcement learning. In *Proc. IEEE International Conference on Image Processing*, pages 2709–2712, Singapore, October 2004.

[34] M. T. Harandi, M. N. Ahmadabadi, and B. N. Araabi. Optimal local basis: A reinforcement learning approach for face recognition. *International Journal of Computer Vision*, 81(2):191–204, 2009.

[35] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, pages 147–151, Manchester, UK, August 1988.

[36] R. Hu and J. Collomosse. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. *Computer Vision Image Understanding*, 117(7):790–806, 2013.

[37] S. Jodogne. Reinforcement learning of perceptual classes using q learning updates. In *Proc. International Multi-Conference on Artificial Intelligence and Applications*, pages 445–450, Innsbruck, Austria, February 2005.

[38] S. Jodogne and J. H. Piater. Interactive selection of visual features through reinforcement learning. In *Proc. Internal Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 285–298, Cambridge, UK, December 2004.

[39] S. Jodogne, F. Scalzo, and J. H. Piater. Task-driven learning of spatial combinations of visual features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pages 48–55, San Diego, CA, USA, June 2005.

[40] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[41] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[42] W. B. Knox and P. Stone. Tamer: Training an agent manually via evaluative reinforcement. In *IEEE International Conference on Development and Learning*, pages 292–297, Monterey, CA, USA, August 2008.

[43] W. B. Knox and P. Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proc. International Conference on Autonomous Agents and Multiagent Systems*, pages 5–12, Toronto, Canada, May 2010.

[44] Y. Kobayashi, J. Ota, K. Inoue, and T. Arai. State and action space construction using vision information. In *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, pages 447–452, Tokyo, Japan, October 1999.

[45] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2619–2624, New Orleans, LA, USA, April 2004.

[46] J. Zico Kolter, P. Abbeel, and A. Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In *Advances in Neural Information Processing Systems*, pages 769–776, Vancouver, B.C., Canada., December 2007.

[47] J. Lankinen, V. Kangas, and J.K. amarainen. A comparison of local feature detectors and descriptors for visual object categorization by intra-class repeatability and matching. In *International Conference on Pattern Recognition*, pages 780–783, Tsukuba, JAPAN, Nov 2012.

[48] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, Alaska, USA, June 2008.

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

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

[51] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 409–415, Madison, WI, USA, June 2003.

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

[53] T. Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[54] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. International Conference on Computer Vision*, pages 1150–1157, Kerkyra, Greece, September 1999.

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

[56] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[57] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[58] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Association for the Advancement of Artificial Intelligence, Workshop on learning for text categorization*, pages 41–48, Madison, Wisconsin, USA, July 1998.

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

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

[61] P. Moreno, M.J. Marín-jiménez, R. Bernardino, and N. Pérez De La Blanca. A comparative study of local descriptors for object category recognition: Sift vs hmax. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 515–522, Girona, Spain, June 2007.

[62] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). Technical report, Febrary 1996.

[63] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

[64] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, New York, NY, USA, June 2006.

[65] M. Nixon and A. S. Aguado. *Feature extraction & image processing*. Academic Press, 2008.

[66] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proc. the European Conference on Computer Vision*, pages 490–503, Graz, Austria, May 2006.

[67] K. Ogata. *Modern control engineering*. Prentice Hall, 2009.

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

[69] L. Paletta, G. Fritz, and C. Seifert. Q-learning of sequential attention for visual object recognition from informative local descriptors. In *Proc. International Conference on Machine Learning*, pages 649–656, Bonn, Germany, August 2005.

[70] L. Paletta, M. Prantl, and A. Pinz. Reinforcement learning of 3-d object recognition from appearance. In *Conference on Automated Learning and Discovery*, Pittsburgh, PA, USA, June 1998.

[71] L. Paletta and E. Rome. Reinforcement learning of object detection strategies. In *International Symposium on Intelligent Robotic Systems*, pages 85–92, Reading, England, July 2000.

[72] R. Paredes, T. Deselaers, and E. Vidal. A probabilistic model for user relevance feedback on image retrieval. In *Machine Learning for Multimodal Interaction – Workshop*, pages 260–271, Utrecht, The Netherlands, September 2008.

[73] J. Peng and B. Bhanu. Closed-loop object recognition using reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):139–154, 1998.

[74] J. Peng and B. Bhanu. Delayed reinforcement learning for adaptive image segmentation and feature extraction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 28(3):482–488, 1998.

[75] J. Peng and B. Bhanu. Local reinforcement learning for object recognition. In *Proc. International Conference in Pattern Recognition*, pages 272–274, Brisbane, Australia, August 1998.

[76] Michael J. Procopio. Hand-labeled DARPA LAGR datasets. Available at `http://www.mikeprocopio.com/labeledlagrdata.html`, 2007.

[77] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, Inc., 1994.

[78] I. Qaffou, M. Sadgal, and A. Elfazziki. A multi-agents architecture to learn vision operators and their parameters. *International Journal of Computer Science Issues*, 9(1):140–149, 2012.

[79] J. Randlov and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proc. International Conference on Machine Learning*, pages 463–471, Madison, Wisconsin, USA, July 1998.

[80] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems.* University of Cambridge, Department of Engineering, 1994.

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

[82] M. A. Ruzon and C. Tomasi. Edge, junction, and corner detection using color distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1281–1295, 2001.

[83] F. Sahba, H. R. Tizhoosh, and M. Salama. Aplication of opposition-based reinforcement learning in image segmentation. In *IEEE Symposium on Computational Intelligence in Image and Signal Processing*, pages 246–251, Honolulu, HI, USA, April 2007.

[84] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[85] H. Samet. *Foundations of multidimensional and metric data structures.* Morgan Kaufmann, 2006.

[86] K. Shibata and M. Iida. Acquisition of box pushing by direct-vision-based reinforcement learning. In *Society of Instrument and Control Engineers – SICE Annual Conference*, pages 2322–2327, Fukui, Japan, August 2003.

[87] M. Shokri and H. R.Tizhoosh. A reinforcement agent for threshold fusion. *Applied Soft Computing*, 8(1):174–181, 2008.

[88] M. Shokri, H. R. Tizhoosh, and M. Kamel. Reinforcement learning for personalizing image search. In *LORNET Annual E-Learning Conference on Intelligent Interactive Learning Object Repositories*, Montreal, QC, Canada, 2006.

[89] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT Press, Cambridge Univ Press, MA(USA), 1998.

[90] G. W. Taylor. *Reinforcement Learning for Parameter Control of Image-Based Applications.* Ph. D. Thesis, University of Waterloo, Ontario, Canada, 2004.

[91] G. W. Taylor and C. Wolf. Reinforcement learning for parameter control of text detection in images from video sequences. In *Proc. International Conference on Information and Communication Technologies: From Theory to Applications*, pages 517–518, Damascus, Syria, April 2004.

[92] G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[93] A. L. Thomaz and C. Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Association for the Advancement of Artificial Intelligence*, pages 8–6, Boston, Massachusetts, USA, July 2006.

[94] A. L. Thomaz, G. Hoffman, and C. Breazeal. Real-time interactive reinforcement learning for robots. In *Association for the Advancement of Artificial Intelligence – Workshop on Human Comprehensible Machine Learning*, Pittsburgh, Pennsylvania, USA, July 2005.

[95] A. L. Thomaz, G. Hoffman, and C. Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 352–357, Hatfield, United Kingdom, September 2006.

[96] H. R. Tizhoosh and G. W. Taylor. Reinforced contrast adaptation. *International Journal of Image and Graphics*, 6(3):377–392, 2006.

[97] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[98] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

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

[100] P. Viola and M. J. Jones. Robust real-time object detection. In *Proc. International Conference on Computer VisionWorkshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing and Sampling*, Vancouver, Canada, July 2001.

[101] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

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

[103] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[104] P. Y. Yin, B. Bhanu, K. C. Chang, and A. Dong. Reinforcement learning for combining relevance feedback techniques. In *Proc. International Conference on Computer Vision*, pages 510–515, Nice, France, October 2003.

[105] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.

[106] Y. Zou, W. Chen, L. Xie, and X. Wu. Comparison of different approaches to visual terrain classification for outdoor mobile robots. *Pattern Recognition Letters*, 38(0):54 – 62, 2014.