Efficient Parameter Estimation in Image Processing using a Multi-Agent Hysteretic Q-Learning Approach

Issam QAFFOU
ISI Laboratory-Department of Computer Science-Faculty of Sciences Semlalia, Cadi Ayyad University
Boulevard Prince My Abdellah B.P. 2390 | 40000 Marrakech. Morocco

***Abstract*—Optimizing image processing parameters is often a time-consuming and unreliable task that requires manual adjustments. In this paper, we present a novel approach that utilizes a multi-agent system with Hysteretic Q-learning to automatically optimize these parameters, providing a more efficient solution. We conducted an empirical study that focused on extracting objects of interest from textural images to validate our approach. Experimental results demonstrate that our multi-agent approach outperforms the traditional single-agent approach by quickly finding optimal parameter values and producing satisfactory results. Our approach's key innovation is the ability to enable agents to cooperate and optimize their behavior for the given task through the use of a multi-agent system. This feature distinguishes our approach from previous work that only used a single agent. By incorporating reinforcement learning techniques in a multi-agent context, our approach provides a scalable and effective solution to parameter optimization in image processing.**

***Keywords*—Parameter estimation; reinforcement learning; cooperative agents; hysteretic q-learning; optimistic agent; object extraction**

I. Introduction

Image processing tasks often require the application of one or more image processing operators that are parameterized, requiring assignment of values to the parameters. However, changing parameter values can significantly impact the quality of the processing result. Non-expert users may face challenges in manually computing optimal parameter values, particularly when multiple operators are involved. For instance, the Deriche filter [1] is frequently used to detect contours in an image, and its parameter α represents the size of the filter. Non-expert users may need to make several attempts to find satisfactory results, wasting time and computing resources.

To address this issue, we propose a novel approach to automatically estimate parameters in image processing using a multi-agent system and reinforcement learning. Our approach leverages cooperative learning between agents to outperform centralized learning. The main contribution of this paper is the use of a multi-agent system to tackle the challenge of parameter estimation in image processing.

In this paper, we begin by discussing related works in Section II. Section III introduces preliminaries, while Section IV details the proposed approach. In Section V, we present experiments and results to validate our approach's effectiveness. Finally, in Section VI, we conclude the paper and discuss potential future work.

II. Related Works

The problem of parameter estimation in image processing has garnered interest from various researchers. In their work [2], Elie Zemmour et al. proposed an automatic method for estimating the parameters required for adaptive thresholding to detect peppers and apples in varying lighting conditions. The authors focused on the adjustment of light level threshold, stop splitting conditions, and classification rule direction for detecting the specific objects (apples and peppers) under consideration. However, the proposed algorithm's adaptability to other image processing tasks was not discussed and is likely to depend on the specific object characteristics. Rafael et al. [3] introduced a method that uses a racing algorithm to tune the parameters required for document image binarization. Their approach is based on a statistical method for determining the optimal parameter values for two algorithms used in binarization tasks: the perception of objects by distance and its combination with a Laplacian energy-based method. Meanwhile, in a study aimed at improving image and video codecs that widely employ uniform quantization schemas, Miguel et al. [4] identified the size of the dead zone and the reconstruction point location as the critical parameters affecting the image R/D coding. The authors proposed a parameterized Uniform Variable Dead Zone Quantizer (UVDZQ) for encoding using wavelets, and evaluated its performance against the most popular quantizers used in video and image coding - USQ (Uniform Scalar Quantizer) and USDZQ (Uniform Scalar Dead Zone Quantizer). The optimal display of an image requires an optimal gamma transformation. In [5], Wang et al. proposed a method based on the location of the Zero-Value Histogram Bin (ZVBH) to estimate the gamma transformation parameter. This approach leverages the relationship between the parameter and the number of ZVBHs to approximate the optimal parameter value and its associated interval. When it comes to biological applications, the choice of parameter values can impact the results obtained. Diana B. et al. [6] proposed using Gaussian process learning to estimate the best biological parameters from non-quantitative and noisy image data. They validated their approach on a parametric function and applied it to estimate parameters in a biological setting by adjusting artificial ISH (in-situ hybridization) data of the developing murine limb bud. Machine learning has become a popular solution to optimization problems in image processing, including parameter estimation. In their work [7], Qaffou et al. proposed an automatic solution for adjusting parameters in an object recognition task using the Q-learning algorithm [54]. This algorithm allows the agent to identify the optimal

combination of parameters for two vision operators - GLCM (Gray Level Co-occurrence Matrix) and k-means. In another study [8], the authors proposed a general framework for optimizing the process of operator and parameter estimation independently on a specific image processing task using reinforcement learning. Furthermore, in [9], Qaffou et al. explored a multi-agent architecture for modeling the interaction between the three components of the proposed architecture. Their solution was successfully applied to a segmentation task, and the results demonstrated its ability to adapt to user preferences [10]. However, the multi-agent architecture proposed in [9, 10] only models different types of agents but has only one agent that learns the optimal values. Qingang et al. [11] proposed a decoupled learning methodology that dynamically fits the weights of a deep network as most existing trained models rely on the configuration of a single parameter. Jinming et al. [12] proposed a simple method for learning local parameter tuning in adaptive image processing by extracting local characteristics from an image and learning the relationship between them and the optimal filtering parameters, optimizing any metric that defines the image's quality.

III. PRELIMINARIES

Most image processing tasks involve the use of one or more vision operators, which are typically parameterized. Each parameter has a range of possible values, and in order to execute an operator, the user must assign a value to each parameter. If the resulting output is unsatisfactory, the user may try other parameter values or even switch to a different operator altogether. In this paper, we focus on the parameter estimation process assuming that the operators to be used have already been fixed. The approach proposed in this paper requires the introduction of certain concepts, which are explained in the following subsections.

A. Overview

We assume that image processing tasks require the use of multiple operators, each of which is parameterized with a range of values. The processing is guided by the ground truths provided for the input images. Our proposed approach in this paper is based on the concepts of multi-agent systems, and requires the following components for the agents to function effectively:

- The combination of operators to use.
- The range of possible values for each parameter.
- The input images.
- The image reference which serves as the ground truth for the desired result. For example, in the case of segmentation, the ground truth is a manual segmentation done by an expert, and it is used for evaluation purposes.

Fig. 1 provides a summary of the inputs required for the multi-agent system used, as well as the output it generates.

B. Multi-agent System

An agent is any autonomous entity that interacts with its environment through sensors and actuators [15]. When such an agent tries to optimize its performance measure, it is called a rational agent. Although intelligent agents are autonomous, they may sometimes need to collaborate and cooperate to complete tasks that require integration or are time-consuming. Agents may or may not cooperate, share knowledge with each other, depending on the task at hand and users' preferences. A system composed of a group of agents capable of interacting with each other is called a multi-agent system (MAS), which constitutes the core of distributed artificial intelligence (DAI) that emerged in the 1980s [13, 14]. Since then, researchers have used MAS to solve problems of distributed or parallel processing in image processing [16-21].

In this study, the multi-agent system used is composed of a team of agents that cooperate using reinforcement learning to speed up the process of finding the optimal values for parameters in image processing tasks. The integration of MAS and reinforcement learning to solve such an optimization problem is an innovative idea.

1) Reinforcement Learning (RL): RL is the technique that forms the basis of the solution proposed in this paper to solve the problem of parameter estimation for a combination of vision operators. We chose this technique because of its adaptability to the dynamicity of environments due to the balance it allows between exploring the environment and exploiting possible solutions [22]. RL was originally developed for Markov Decision Processes (MDPs). RL defines a type of interaction between an agent and its environment, as shown in Fig. 2. In a real state s of the environment, the agent chooses and performs an action a which causes a transition to the state s'. The agent receives a reinforcement signal "r" that is a reward if the action is beneficial or a punishment if not; a null signal means an inability to award a penalty or reward. The agent then uses this signal to improve its strategy, which is the sequence of its actions, in order to maximize the accumulation of future rewards.

To achieve this, the agent must find a balance between exploration and exploitation. Exploration consists of testing new actions that can lead to higher gains, but with the risk that they will be lower, while exploitation consists of applying the best strategy acquired until then (which may not be optimal). Fig. 2 shows the general architecture of an RL agent. There are several methods to find the optimal policy corresponding to the maximum value of the state/action value function. In this paper, the proposed MAS exploits the idea of the Q-learning algorithm, where the agents cooperate and update their Q-values optimistically.

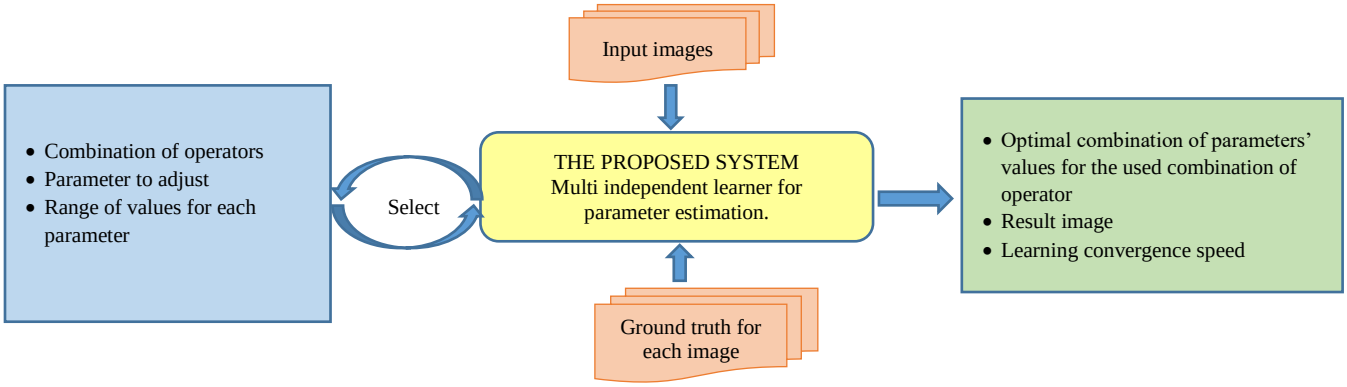

Fig. 1. External architecture of the proposed solution.

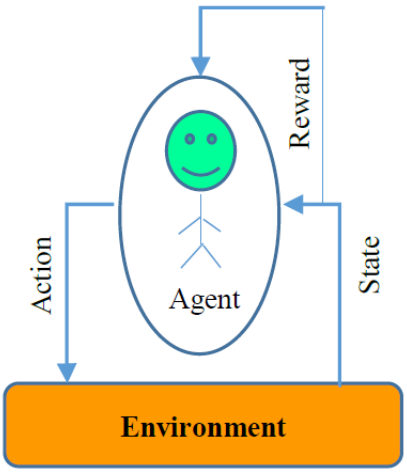

Fig. 2. RL agent general architecture.

2) Q-learning algorithm: The Q-learning algorithm was proposed by Watkins in 1989 [23]. In this model, the agent learns to act optimally in Markov domains by testing action sequences. It selects an action in a particular state and uses the immediate reward or punishment to estimate the value of that state. By trying different actions in different states, the agent learns which is the best by referring to the long-term update of the rewards [24]. The agent must determine an optimal policy and maximize the total expected rewards. Algorithm 1 gives the procedural form of the Q-learning algorithm [56].

Alg. 1: Q-learning algorithm

1. *Initialize* $Q(s,a)$ *for all* $s \in S$, $a \in A(s)$ *randomly*
2. *Observe the initial state* s
3. *Repeat until termination:*
   4. *Select an action* a *using a policy derived from* Q
   5. *Take action* a*, observe reward* r *and new state* s'
   6. *Update*
   $$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$
   7. $s \leftarrow s'$
8. *Until* s *is a terminal state*

In this algorithm, S is the set of possible states, $A(s)$ is the set of possible actions in state s, α is the learning rate, and γ is the discount factor used to balance immediate and future rewards. The value of $Q(s,a)$ is updated based on the reward received and the estimated value of the next state-action pair. The policy derived from Q is used to select the action in step 4.

3) Cooperative learning: G Although reinforcement learning (RL) has promising applications in multi-agent systems (MAS), there are still several challenges to extending RL to a MAS [25]. One major difficulty is the lack of theoretical guarantees, as convergence hypotheses that hold for a single agent may not be valid for a MAS due to the presence of several learners, which makes the environment non-stationary and challenging for multi-agent learning systems [26-29]. Additionally, defining a good learning goal for all RL agents and enabling communication between them pose further challenges for learning in a MAS [30]. Despite these obstacles, researchers have successfully integrated RL into MAS and addressed coordination problems between agents [31-36]. One algorithm that addresses the challenge of cooperation between agents is hysteretic Q-learning, which focuses on optimistic agent behavior and has been shown to perform well in multi-agent environments [29]. We use this algorithm to include RL in our cooperative MAS. The next section provides more details on how we model our cooperative MAS.

IV. The Proposed Approach Modeling

To accomplish a task in image processing, such as filtering, image enhancement, segmentation, object recognition, etc., a sequence of operators must be applied. A task could be divided into several sub-tasks that may require a sub-sequence of vision operators to be used. To run an operator, its parameters must be adequately tuned. This problem has been solved using a single agent [7], but it consumes much time to converge. The main contribution of this paper is to propose a quicker and more precise solution to this problem. We model our solution as a multi-agent system using reinforcement learning. Each agent takes one operator and learns to find the optimal values of its parameters depending on the user's preferences. These preferences include the type of process the user wants to perform, the operators they want to use, and the desired result. The agents work together to find the best choice that brings them the highest reward. These agents have a joint action, and each one has its

individual action in the whole process. An agent's individual actions are formed by all possible choices of the values that can be assigned to its parameters. Any value changed is a new action. Formally speaking, if an operator has, for instance, k parameters $(p_1, p_2, \dots, p_k)$ and each parameter p_i has a set of possible values $V_i = \{v_{i_1}, v_{i_2}, \dots, v_{i_p}\}$, an individual action is then:

$$a_j = (u_1, u_2, \dots, u_k) \in V_1 \times V_1 \times \dots \times V_k \quad (1)$$

Each agent chooses its action independently of the choices of the other agents. For the multi-agent system, we talk about a joint action which is a combination of the individual actions. It is the action that the multi-agent system applies on the input image, which represents the environment with which the agents are interacting. A state of the environment is a set of features extracted from the image. For each image, we provide a ground truth to evaluate the result found by the proposed multi-agent system. The reward is calculated by comparing the features of the output image with those of its ground truth. These components, which are the set of actions, states, and the reward function, are the principal elements required to define the reinforcement learning process. Fig. 3 shows the global schema of the proposed multi-agent system. The proposed multi-agent system applies a joint action on the input image, compares the obtained result with the ground truth, and receives a return signal in the form of a reward or punishment. During learning, the system reinforces actions that have been beneficial in the past, and after convergence, it selects the action with the maximum Q-value according to the principle of Q-learning. The definition of actions, states, and reward function in our approach depends on the task to be accomplished and the execution environment. An adaptive definition is provided in Section IV. The use of RL in this system is challenging because the final return concerns all agents, and each agent may question their share of the return. Additionally, since an agent cannot see their teammates' choices, they may be punished for a bad choice made by another agent. To address this issue, we consider all agents to be independent learners. The main challenge for these agents is coordination; how to ensure that all agents choose their individual actions consistently to achieve a Pareto-optimal joint action, where no other strategy benefits any of the agents. This is a complex problem resulting from the combined actions of several factors. Since agents must cooperate, they have no interest in threatening each other, but they must change their policy to improve their rewards and adapt to this change. We can model this setting as a repeated game where the same agents play the same game repeatedly. As the agents share the common goal of achieving the best end result, this game is cooperative and the reward is shared. A simple extension of centralized Q-Learning [56], case of a single learner, to stochastic games takes into account common actions in the calculation of Q-values. Thus, the update equation according to a centralized view of a system of n agents is:

$$Q(s, a_1, \dots, a_n) \leftarrow (1-\alpha)Q(s, a_1, \dots, a_n) + \alpha^*[r + \gamma^* \max Q(s', a'_1, \dots, a'_1)] \quad (2)$$

Where s' is the new state, α is the learning rate and γ∈ [0, 1] is the discount factor [23].

In this model, the reinforcement perceived by an agent depends on the actions chosen by the group. Therefore, an agent does not know exactly its share in the total reward, or at least the influence of the received return (positive or negative) on its individual action. Even if an agent executes a good action, it could still be punished because of a bad choice made by the group. It is therefore preferable for an agent to give little importance to a punishment received after choosing an action that has satisfied it in the past. However, the agent must not be completely blind to sanctions, as this could result in a sub-optimal equilibrium or prevent coordination on an optimal joint action [29]. To solve this problem, we use the hysteretic Q-learning algorithm. This algorithm considers that an agent with an optimal individual action should not be punished because of a bad choice made by the group, but it must remain optimistic in order to reduce variations in the learned policy. The equation for updating the hysteretic Q-learning proposed by L. Matignon [29] for an agent i executing the action a_i from state s to transit to state s' is:

$$\delta \leftarrow r + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a_i) \quad (3)$$

$$Q_i(s, a_i) \leftarrow \begin{cases} Q_i(s, a_i) + \alpha\delta & if\ \delta \geq 0 \\ Q_i(s, a_i) + \beta\delta & otherwise \end{cases} \quad (4)$$

Where α and β are two coefficients corresponding respectively to the increase or the decrease of the Q-value of a joint action. To have optimistic learners α must be greater than β. The main goals of using these two coefficients is to minimize the shadowed equilibria's effect and to manage stochasticity of the environment [29]. The Hysteretic Q-Learning algorithm is decentralized; each agent builds his own Q-table whose the size is independent on the number of agents and is linear according to his own actions. Algorithm 2 summarizes the hysteretic Q-learning.

Alg.2: Hysteretic Q-learning algorithm

Begin

Initialize arbitrarily $Q_i(s, a_i)$ for each (s, a_i) from $S \times A_i$
Initialize the initial state s
While s is not an absorbent state do
In the state s, choose the action a_i / phase of decision */*
Apply the action a_i and observe the new state s' and the return r
$q \leftarrow r + \gamma\ max_{b \in A_i} Q_i(s', b)$
/ Hysteretic update */*
if $q \geq Q_i(s, a_i)$ ***then***
$Q_i(s, a_i) \leftarrow (1-\alpha)Q_i(s, a_i) + \alpha q$
else
$Q_i(s, a_i) \leftarrow (1-\alpha)Q_i(s, a_i) + \beta q$
if $Q_i(s, arg\ max_{u \in A_i} \pi_i(s, u)) \neq max_{u \in A_i} Q_i(s, u)$ ***then***
choose randomly $a_{max} \in arg\ max_{u \in A_i} Q_i(s, u)$
$\forall\ b \in A_i\ \ \pi_i(s, b) \leftarrow \begin{cases} 1\ if\ b = a_{max} \\ 0\ otherwise \end{cases}$
/ equilibria selection */*
$s \leftarrow s'$

End

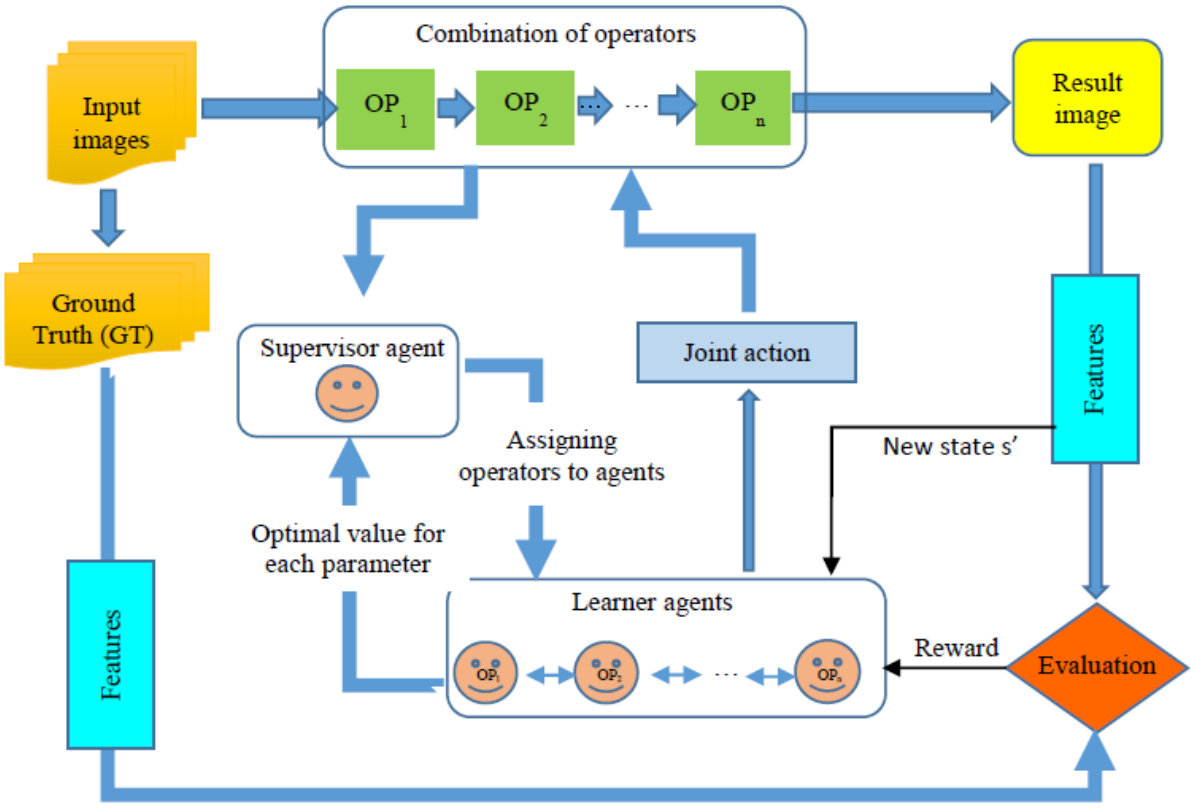

Fig. 3. Global schema of the proposed solution.

The proposed multi-agent system consists of two types of agents: learner agents and a supervisor agent. Each learner agent takes one operator and proceeds to estimate its parameters, with each agent representing one of the operators in the combination. The learner agents work together to learn the best values of the parameters for the entire combination of operators. These agents use a hysteretic Q-learning algorithm, which means that they give little importance to the penalties received but are not completely blind to them [37, 38].

The supervisor agent is responsible for distributing the operators to the learner agents and receiving the optimal value of each parameter.

V. Experiments and Analysis

In this section, we evaluate the effectiveness of our approach by applying it to estimate parameters in an object recognition task. Object recognition is a vital area of computer vision with numerous applications such as object tracking, facial recognition, and autonomous driving. The main goal of object recognition is to detect objects in an image by locating, classifying, and framing them with rectangles. Object detection methods can be broadly categorized into two categories: the first approach divides the image into regions and classifies them into object categories, while the second approach treats object detection as a classification or regression problem, producing final results directly. Examples of methods in the first category include R-CNN [39], SPP-net [40], Fast R-CNN [41, 42], Faster R-CNN [43], R-FCN [44], FPN [45] and Mask R-CNN [46]. For the second category we find MultiBox [47], AttentionNet [48], G-CNN [49], YOLO [50], SSD [51], YOLOv2 [52], DSSD [53] and DSOD [54]. While our work does not aim to propose a new object detection method, we demonstrate how our approach can provide valuable assistance to users seeking to determine the optimal values of each parameter in a combination of operators. We use object detection as a case study to demonstrate the effectiveness of our approach.

A. Experiments' Environment

We conducted an experiment to evaluate our approach using a dataset of 60 mixed textured images, where a disc (also textured) was inserted into 40 images while the remaining 20 did not contain it. The objective was to recognize and extract the disc from these images using a combination of two operators in two phases. In the first phase, a filter operator with two parameters needed to be estimated, and in the second phase, an operator with two parameters was applied to segment textures and classify them into clusters. For each parameter, a set of possible values was proposed, and a system of two agents was assigned to adjust the parameters for each operator. The results obtained using our proposed approach were compared with those found in [7], where the learning was centralized. To facilitate a comprehensive comparison, we implemented our approach using Matlab, which has a rich toolbox of image processing operators and allows for parallel programming using "Workers."

B. Parameter's Value Learning

The operators used in this experiment are "imfilter" and "GLCM_KMeansFct." The "imfilter" operator is an existing operator in the Matlab toolbox and is used for image filtering.

The "GLCM_KMeansFct" operator is a function that we implemented by combining two Matlab operators: "graycomatrix," which computes the gray-level co-occurrence matrix (GLCM) of an image, and "kmeans," which classifies the obtained textures into clusters.

The operator “imfilter” has two parameters:

- the type of filtering with a single value: {'unsharp'}
- alpha (smoothing coefficient) with 2 possible values: {0.2, 0.6}

The operator “GLCM_KMeansFct” has two parameters to adjust:

- the size of the sliding window with 7 possible values: {3, 5, 7, 9, 11, 13, 15}
- the number of clusters with 4 possible values: {2, 3, 4, 5}

The proposed approach in this paper consists of assigning one agent AG1 to “imfilter” and another AG2 to “GLCM_KMeansFct”.

C. RL Configuration

The three principal components: actions, states and the return function must be defined adaptively to our approach.

1) Actions: An action of the proposed multi-agent system is a joining of individual actions of all the agents. For instance, an individual action for the agent AG1 is {unsharp, 0.2} or {unsharp, 0.6}, and for the agent AG2 is every combination between a size of the sliding window and a number of possible clusters. Thus, a joint action may be, for example, {unsharp, 0.2, 3, 2}.

2) States: A state is defined according to the features of the image obtained after the execution of an action. In this paper, we consider four characteristics to define a state.

$$S = [\, x_1, x_2, x_3, x_4] \tag{6}$$

x_1: the number of objects in the result image.

x_2: the ratio between the area of the result object and the area of the entire input image.

x_3: the ratio between the area of the result object and the area of the reference object.

x_4: the average of the values of the textural metrics: energy, correlation, entropy and contrast [55].

3) Return function: The return can be a punishment or a reward, depending on the quality criterion representing how well the object has been detected. A simple method is to use an objective assessment by comparing the obtained result with the ground truth. This comparison is made between features of the two images to generate a value determining a reward or a punishment. The value calculated from this comparison is a weighted sum of the difference between the features extracted from the two images. The weights reflect the importance of a feature in the final decision.

$$D = \sum_i w_i D_i \tag{7}$$

Where w_i are the weights assigned to each of the following differences:

D1: difference in number of objects;

D2: difference in size of objects;

D3: difference in area of objects;

D4: difference in values of entropy;

If D is greater than a given threshold, the return is a reward. Otherwise it is a punishment.

$$if\ (D \leq \varepsilon)\ r = +10;$$
$$else\ r = -10;$$

Where ε is the threshold. In this experiments, we fix it in 0.15.

D. Results

The experience is based on 40 textured images containing four different textures. In these images we inject an object of interest, which is a disk. Fig. 4 shows some examples of these images.

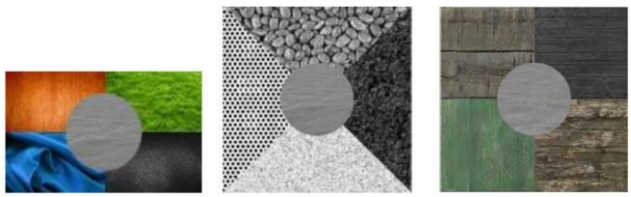

Fig. 4. Examples of used images.

The process of our approach needs a ground truth. In this experiment, the ground truth is the disc shown in Fig. 5.

Fig. 5. The ground truth in our experiment.

The multi-agent system we propose in this paper, uses some features to compare the obtained result with the ground truth. These features are the same for both images. The reference features are the following data:

NbrTargetArea = 5: represents the number of target zones

TotalSize = 20800: the total size of the image

AreaObjet = 2573: the area of the object of interest.

TextureMesure = 4.550863e+000: the entropy value (GLCM)

The desired state is [1.000 1.000 1.000 1.000] according to the equation (6).

The size and the area are measured in pixels.

For the process of exploration/exploitation, we use the hysteretic Q-learning with the values: Number of episodes and steps (iterations) are respectively 900 and 60, $\alpha = 0.5, \beta = 0.01, \gamma = 0.9, \varepsilon = 0.008$.

After running the proposed multi-agent system, we obtain the extracted object shown in Fig. 6.

The corresponding features of the result disc are:

NbrTargetArea = 6

TotalSize = 23810

AreaObjet = 2371

TextureMesure = 4.3448e+000

Fig. 6. Disc obtained by the proposed MAS.

To compare the two objects, the differences in equation (7) are calculated. Their obtained values are:

D1 = 2.000000e-001

D2 =1.447115e-001

D3 = 7.850758e-002

D4 = 2.06063e-001

The difference between the result disc and the ground truth is given by:

D = 0.2*D1+0.2*D2+0.3*D3+0.3*D4 = 9.8672e-002

The optimal joint action (most rewarding) proposed by our MAS is then {'unsharp', 0.6, 5, 3} and its corresponding state is:

[1.1000 1.1000 0.9000 1.0000]

E. Discussion

The value of D is small; this means that the obtained disc is very closer to the reference. This result demonstrates that the proposed multi-agent system succeeds to extract the object of interest. The main contribution of this paper is not only to propose a method that finds the optimal parameters' values, that is already done in [7], but to propose an approach that outperforms the solution proposed in [7] in terms of speed and accuracy. Fig. 7 shows the curves of learning and the reward gain for the multi-agent system proposed in this paper.

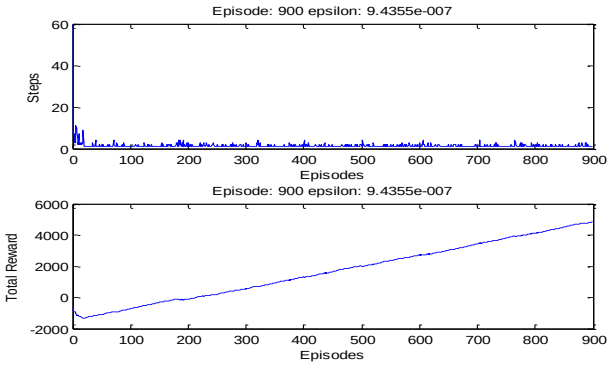

Fig. 7. Results of the proposed approach. Top: Learning curve (steps/episodes). Bottom: Corresponding cumulative reward.

The curve at the top shows the learning process. During the first 10 episodes, the multi-agent system executes several iterations to reach a convergence. A correspondence is clear in the curve below showing the cumulative returns. Indeed, during the first 10 episodes the multi-agent system receives only punishments, then the curve increases to show that the multi-agent system accumulates rewards. This is very logical with the principle of reinforcement learning, and in particular with hysteretic Q-learning.

To show the performance of the proposed multi-agent system as well as its main contribution, we run the one-agent approach [7] in the same environment with the same criteria. Fig. 8 shows the obtained curves.

In the one-agent approach, the top curve shows the execution of many iterations during the first 100 episodes with a high cumulative punishment as the bottom curve shows.

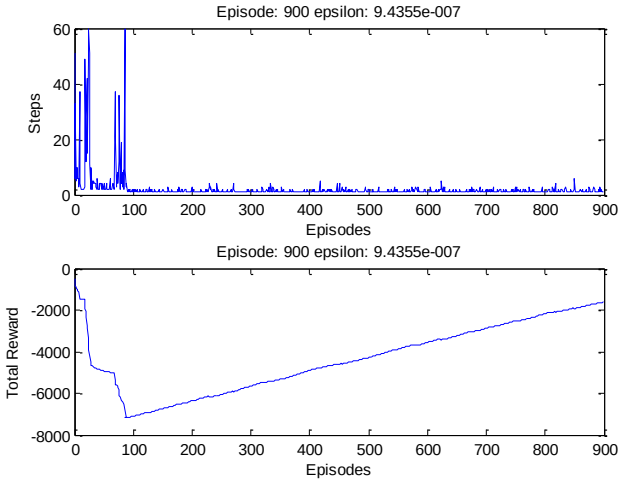

Fig. 8. Results of one-agent approach. Top: Learning curve (steps/episodes). Bottom: Corresponding cumulative reward.

Table I summarizes a comparison between the two approaches, by running them in the same environment and under the same conditions cited above.

TABLE I. COMPARISON BETWEEN MULTI-AGENT AND MONO-AGENT APPROACHES.

	Multi-agent & Hysteretic Q-Learning	One-agent & Q-learning
Quality Result (difference between the result and the ground truth)	10%	12%
Convergence (at which episode)	10	100
Reward (900 episodes)	4200	-
Punishment (900 episodes)	-1200	-7200

The results obtained using the multi-agent approach are significantly better than those obtained by the single-agent approach. The criteria used to evaluate the quality of the

extracted disc include the episode at which convergence starts, the values of punishment (which should be small), and the reward (which should be high). Based on these criteria, we can determine whether one approach is superior to another. In this paper, the multi-agent approach outperforms the single-agent approach. However, it is important to note that the difference between the extracted disc and the reference disc is still relatively high. This is mainly due to the choice of operators and not the proposed approach, which aims to adjust the parameters of the selected operators rather than propose a new method for object extraction.

VI. Conclusion

In this paper we have proposed an approach based on multi-agent system and reinforcement learning to automatize the process of parameter selection in image processing. In this work, adjusting parameters is seen as a decision process over time, where experiences gained from past decisions affect future decisions. The solution we have proposed, attributes one agent to an operator to adjust its parameters. We have used hysteretic Q-learning for multi-agent learning to find the best parameters for the given operators and test it to extract an object of interest. The complexity of the images, the speed of convergence and the quality of the results show the potential of the new approach and its adaptability. The results show that the proposed approach outperforms the use of one agent especially in terms of speed. Future works aim to include the operator selection also. In spite of using a predefined operator combination, we can suggest among several possible operators which are the best to use and furthermore what are their optimal parameters' values. Also, we think about using deep reinforcement learning instead of classical reinforcement learning.

References

[1] R. Deriche. Fast Algorithms for Low-Level Vision. IEEE Transactions on Pattern Anal. and Machine Intell. ,vol .PAMI-12, no.1 (1990), 78-87.

[2] E. Zemmour, P. Kurtser, and Y. Edan. Automatic parameter tuning for adaptive thresholding in fruit detection. Sensors, 19(9) (2019), 2130.

[3] G. Rafael, M. Ricardo, A. Carlos, B. Péricles. Parameter tuning for document image binarization using a racing algorithm. Expert Systems with Applications 42 (2015), 2593–2603.

[4] O. Miguel, P. Pablo, M. Otoniel, P. Manuel. Optimizing the image R/D coding performance by tuning quantization parameters. Journal of Visual Communication and Image Representation 49 (2017), 274–282.

[5] P. Wang, F. Liu, C. Yang, X. Luo. Parameter estimation of image gamma transformation based on zero-value histogram bin locations. Signal Processing: Image Communication, (2018).

[6] B. Diana, D. Michael, I. Dagmar. Global optimization using Gaussian processes to estimate biological parameters from image data. Journal of Theoretical Biology 481 (2019), 233–248.

[7] I. Qaffou, M. Sadgal, A. Elfazziki. A New Automatic Method to Adjust Parameters for Object Recognition. International Journal of Advanced Computer Science and Applications, Vol. 3, No. 9 (2012), 213-217.

[8] I. Qaffou, M. Sadgal, A. Elfazziki. Selecting Vision Operators and Fixing Their Optimal Parameters Values Using Reinforcement Learning. Lecture Notes in Computer Science Volume 7340 (2012), 103-112. Springer-Verlag Berlin, Heidelberg ©2012.

[9] I. Qaffou, M. Sadgal, A. Elfazziki. A Multi-Agents Architecture to Learn Vision Operators and their Parameters. International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1 (2012), 140-149.

[10] I. Qaffou, M. Sadgal, A. Elfazziki. Q-learning optimization in a multi-agents system for image segmentation. International Journal of Advanced Studies in Computer Science and Engineering, Volume 2, Theme based issue 3 (2013), 41-47.

[11] F. Qingnan, C. Dongdong, Y. Lu, H. Gang, Y. Nenghai, C. Baoquan. Decouple Learning for Parameterized Image Operators. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2019.

[12] J. Dong, L. Frosio, J. Kautz. Learning Adaptive Parameter Tuning for Image Processing, Proc. EI, Image Processing: Algorithms and Systems XVI, Burlingame (2018).

[13] A.H. BOND and L.GASSER. Reading in distributed artificial intelligence. Morgan Kaufmann publishers, Inc, 1988.

[14] J. FERBER. Les systèmes multi-agents: vers une intelligence collective, InterEdition. 1997.

[15] S.J. Russell and P. Norvig. Artificial Intelligence: a Modern Approach. Prentice Hall, Englewood Cliffs, NJ, 2nd edition (2003).

[16] P. Hofmann P. Lettmayer, T. Blaschke, M. Belgiu, S. Wegenkittl, R. Graf, T. Lampoltshammer, V. Andrejchenko. Towards a framework for agent-based image analysis of remote-sensing data. International Journal of Image and Data Fusion, 6:2 (2015), 115-137.

[17] S.Y. Wai, C. WaiShiang, M. bin Khairuddin. Multi Agent Object Recognition: A Preliminary Study. ICIIP (2019), China.

[18] T. Inguère, F. Carlier, V. Renault. Flexible image processing in Embedded Systems using Multi-agents Systems. IFAC-PapersOnLine Volume 49, Issue 25 (2016), 164-169.

[19] A. Maudet, G. Touya, C. Duchêne, S. Picault. Patterns multi-niveaux pour les sma. Journées Francophones sur les Systèmes Multi-Agents (2015).

[20] L. Males,D. Marcetic, S. Ribaric. A multi-agent dynamic system for robust multi-face tracking. Expert Systems with Applications 126 (2019), 246–264.

[21] J. Paulin, A. Calinescu, M. Wooldridge. Agent-based modeling for complex financial systems. IEEE Intelligent Systems, 33 (2) (2018), 74–82.

[22] R. S. Sutton and A. G. Barto. Reinforcement learning: an introduction. Adaptive computation and machine learning. MIT Press, Cambridge, Mass., 1998.

[23] C. J. C. H. Watkins. Learning from Delayed Rewards. PhD thesis, Cambridge University, 1989.

[24] S. Sehad. Contribution à l'étude et au développement de modèles connexionnistes a apprentissage par renforcement : application a d'acquisition de comportements adaptatifs. Thèse génie informatique et traitement du signal. Montpellier: Université de Montpellier II, 1996, 112 p.

[25] E. Yang and D. Gu. Multiagent reinforcement learning for multi-robot systems: A survey. Tech. rep., Department of Computer Science, University of Essex, (2004).

[26] M. Bowling and M. Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Tech. rep., Computer Science Department, Carnegie Mellon University, (2000).

[27] B. Banerjee, S. Sen, J. Peng. On-policy concurrent reinforcement learning. Journal of Experimental & Theoretical Artificial Intelligence, 16(4) (2004), 245–260.

[28] S. Abdallah and V. Lesser. A multiagent reinforcement learning algorithm with non-linear dynamics. Journal of Artificial Intelligence Research, 33 (2008), 521–549.

[29] L. Matignon, J. G. Laurent, N. Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. Knowledge Engineering Review, Cambridge University Press (CUP), 2012, 27 (1), pp.1-31.

[30] L. Busoniu, R. Babuska, B. De Schutter. A comprehensive survey of multiagent reinforcement learning. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 38(2) (2008), 156–172.

[31] C. Watkins and P. Dayan. Technical note: Q-learning. Machine Learning. 8 (1992), 279–292.

[32] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In Proc. 17th International

Conf. on Machine Learning, pp. 535–542. Morgan Kaufmann, San Francisco, CA, (2000).

[33] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. Artificial Intelligence, 136 (2002), 215–250.

[34] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems. Washington, DC, USA. IEEE Computer Society. (2004). pp. 1258–1259.

[35] J.Hao, D. Huang, Y. Cai, H.-f. Leung. The dynamics of reinforcement social learning in networked cooperative multiagent systems, Engineering Applications of Artificial Intelligence 58 (2017), 111–122.

[36] W. Zemzem and M. Tagina. Cooperative multi-agent reinforcement learning in a large stationary environment. IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), IEEE, 2017, pp. 365–371.

[37] J. A. Swets. Signal Detection Theory and Roc Analysis in Psychology and Diagnostics. Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[38] Chunyu, L., & Gang, L. Learning Multiple Instance Deep Representation for Objects Tracking. Journal of Visual Communication and Image Representation (2020), Volume 71, 102737.

[39] R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

[40] K. He, X. Zhang, S. Ren, J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans. PatternAnal. Mach. Intell., vol. 37, no. 9 (2015), pp. 1904–1916.

[41] R. Girshick. Fast r-cnn. In ICCV, 2015.

[42] Rossi L., Karimi A., Prati A. Self-Balanced R-CNN for instance segmentation. Journal of Visual Communication and Image Representation (2022), Volume 87, 103595.

[43] S. Ren, K. He, R. Girshick, J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, 2015, pp: 91–99.

[44] Y. Li, K. He, J. Sunet al., "R-fcn: Object detection via region-basedfully convolutional networks," inNIPS, 2016, pp. 379–387.

[45] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, S. J.Belongie. Feature pyramid networks for object detection. In CVPR, 2017.

[46] K. He, G. Gkioxari, P. Dollar, R. B. Girshick. Mask r-cnn. In ICCV, 2017.

[47] D. Erhan, C. Szegedy, A. Toshev, D. Anguelov. Scalable object detection using deep neural networks. In CVPR, 2014.

[48] D. Yoo, S. Park, J.-Y. Lee, A. S. Paek, I. So Kweon. AttentionNet: Aggregating weak directions for accurate object detection. In CVPR, 2015.

[49] M. Najibi, M. Rastegari, and L. S. Davis. G-cnn: an iterative grid based object detector. In CVPR, 2016.

[50] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You only look once: Unified, real-time object detection. In CVPR, 2016.

[51] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector," in ECCV, 2016.

[52] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger" arXiv: 1612.08242, 2016.

[53] C. Y. Fu, W. Liu, A. Ranga, A. Tyagi, A. C. Berg. Dssd: Deconvolutional single shot detector. arXiv: 1701.06659, 2017.

[54] Z. Shen, Z. Liu, J. Li, Y. G. Jiang, Y. Chen, X. Xue. Dsod: Learning deeply supervised object detectors from scratch. In ICCV, 2017.

[55] R. M. Haralick, K. Shanmugan, I. Dinstein. Textural Features for Image Classification. IEEE Transactions on Systems, Man and Cybernetics, Vol. 3, 1973, No 6, 610-621.

[56] C. J. C. H. Watkins and P. Dayan. Q-learning. Machine Learning, 8: 279–292, 1992.