Interactive Learning of Mappings from Visual Percepts to Actions

Sébastien Jodogne[1] S.Jodogne@ULg.ac.be
Justus H. Piater Justus.Piater@ULg.ac.be
Montefiore Institute (B28), University of Liège, B-4000 Liège, Belgium

Abstract

We introduce flexible algorithms that can automatically learn mappings from images to actions by interacting with their environment. They work by introducing an image classifier in front of a Reinforcement Learning algorithm. The classifier partitions the visual space according to the presence or absence of highly informative local descriptors. The image classifier is incrementally refined by selecting new local descriptors when perceptual aliasing is detected. Thus, we reduce the visual input domain down to a size manageable by Reinforcement Learning, permitting us to learn direct percept-to-action mappings. Experimental results on a continuous visual navigation task illustrate the applicability of the framework.

1. Introduction

Many reactive robotic tasks can be solved by "black-box" mappings from an input space to an output space, that directly connect percepts to the appropriate actions through a given computational mechanism. Such mappings are usually hard to derive by hand, especially when the input space contains images, since visual domains are high dimensional and noisy. This paper introduces algorithms suitable for building image-to-action mappings using a fully automatic and flexible learning protocol.

Strong neuropsychological evidence suggests that human beings learn to extract useful information from visual data in an *interactive* fashion, without any external supervisor (Gibson & Spelke, 1983). By evaluating the consequence of our actions on the environment, we learn to pay attention to visual cues that are behaviorally important for solving the task. This process is *task-driven*, since different tasks do not necessarily need to make the same distinctions.

One plausible framework to learn image-to-action mappings in such an interactive protocol is *Reinforcement Learning* (RL) (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998). RL is a generic framework for modeling the behavior of an agent that learns a percept-to-action mapping through its interactions with the environment. The agent is never told what action it should take. Rather, when it does a good or a bad action, it only receives from the environment a reward or a punishment, the *reinforcement signal*. The major advantages of the RL protocol are that it is fully automatic, and that it imposes very weak constraints on the environment. However, the price to pay for the flexibility of RL is poor performance when the input space of the agent is high dimensional or noisy, which makes it unsuitable for building image-to-action mappings.

However, to take the right decision, it is often sufficient for the robotic agent to focus its attention on robust and highly informative patterns in the percepts. This is actually the basic postulate behind *local-appearance methods*, that have had much success in Computer Vision applications such as image matching, image retrieval and object recognition (Lowe, 1999; Schmid & Mohr, 1997). They rely on the detection of discontinuities in the visual signal thanks to *interest point detectors* (Schmid et al., 2000). Similarities in images are thereafter identified using a *local description* of the neighborhood around the interest points (Mikolajczyk & Schmid, 2003): If two images share a sufficient number of matching local descriptors, they are considered as belonging to the same visual class. Matches are detected through a suitable metric (e.g., Mahalanobis' or Euclidean distance), as local descriptors are nothing else than vectors of real numbers. Such techniques are at the same time powerful and flexible, as they

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

[1]Research fellow of the Belgian National Fund for Scientific Research (FNRS).

are robust to partial occlusions, and do not require segmentation or 3D models of objects.

It seems therefore promising to introduce, in front of the RL algorithm, an image classifier that translates the raw visual input to a visual class according to the presence of some selected local descriptors at the interest points in the image. This preprocessing step is intended to reduce the size of the input domain, thus enhancing the rate of convergence, the generalization capabilities as well as the robustness of RL to noise in visual domains. The central difficulty is the dynamic selection of the discriminative local descriptors.

We propose the following algorithm, that will be called *Reinforcement Learning of Visual Classes* (RLVC). Initially, the agent just knows about one visual class, so that all images are mapped to this class. Of course, this introduces a kind of *perceptual aliasing* (Whitehead & Ballard, 1991): The optimal decisions cannot always be made, since percepts requiring different reactions are associated with the same class. Then, the agent isolates the aliased classes. Since there is no external supervisor, the agent can only rely on a statistical analysis of the earned reinforcements. Once some aliased class has been detected, the agent selects a new local descriptor that is *distinctive*, i.e. that best explains the observed variations in the reinforcements for the considered class. The extracted descriptor is finally used to refine the classifier. New local descriptors are learned until perceptual aliasing vanishes.

RLVC is quite different from classical RL techniques for discretizing the input space (Bertsekas & Tsitsiklis, 1996), since previous methods assume that the interesting features (i.e., the local descriptors) or their numbers are known in advance. In fact, RLVC can be thought of as performing adaptive discretization of the perceptual space on the basis of an image processing algorithm. It allows us to select a subset of highly relevant visual features in a fully closed-loop, task-driven learning process.

2. Theoretical Background

2.1. Reinforcement Learning

In RL, the environment is traditionally modeled as a *Markov Decision Process* (MDP). An MDP is a tuple $\langle S, A, \mathcal{T}, \mathcal{R} \rangle$, where S is a finite set of states, A is a finite set of *actions*, $\mathcal{T}$ is a probabilistic *transition function* from $S \times A$ to S, and $\mathcal{R}$ is a *reinforcement function* from $S \times A$ to $\mathbb{R}$. An MDP obeys the following discrete-time dynamics: If at time t, the agent takes the action a_t while the environment lies in a state s_t, the agent perceives a numerical reinforcement $r_{t+1} = \mathcal{R}(s_t, a_t)$, then reaches some state s_{t+1} with probability $\mathcal{T}(s_t, a_t, s_{t+1})$.

The definition of MDPs assumes the *full observability* of the state space, which means that the agent is able to distinguish between the states of the environment using only its sensors. Let us define the *perceptual space* P as the set of possible percepts that the sensors can return. In visual tasks, P is a set of images. So, from the point of view of the agent, an *interaction* with the environment is defined as a quadruple $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$, where p_t (resp. p_{t+1}) is the percept furnished by its sensors in the presence of the state s_t (resp. s_{t+1}).

A *percept-to-action mapping* is a fixed probabilistic function $\pi : P \mapsto A$ from percepts to actions. A percept-to-action mapping tells the agent the probability with which it should choose an action when faced with some percept. In RL terminology, such a mapping is called a *stationary Markovian control policy*. For an infinite sequence of interactions starting in a state s_t, the *discounted return* is $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$, where $\gamma \in [0, 1[$ is the *discount factor* that gives the current value of the future reinforcements. The *Markov Decision Problem* for a given MDP is to find an optimal percept-to-action mapping that maximizes the expected discounted return.

MDPs can be solved using *Dynamic Programming* (DP). Let us call $Q^\pi(s, a)$, the function giving for each state $s \in S$ and each action $a \in A$ the expected discounted return obtained by starting from state s, taking action a, and thereafter following the mapping π: $Q^\pi(s, a) = \mathrm{E}^\pi \{R_t \mid s_t = s,\ a_t = a\}$, where E^π denotes the expected value if the agent follows the mapping π. Let us also define the H *transform* from Q functions to Q functions as:

$$(HQ)(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{T}(s, a, s') \max_{a' \in A} Q(s', a'), \tag{1}$$

for all $s \in S$ and $a \in A$. All the optimal mappings for a given MDP share the same Q function, denoted Q^*, that always exists and that satisfies Bellman's so-called optimality equation (Bellman, 1957):

$$HQ^* = Q^*. \tag{2}$$

Once Q^* is known, an optimal deterministic percept-to-action mapping π^* is easily derived by letting $\pi^*(s) = \mathrm{argmax}_{a \in A} Q^*(s, a)$ for each $s \in S$.

RL is a set of algorithmic methods for solving Markov Decision Problems when the underlying MDP is unknown (Bertsekas & Tsitsiklis, 1996). The input of RL algorithms is basically a sequence of interactions

$\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$ of the agent with its environment. RL techniques are often divided in two categories: (i) *model-based methods* that first build an estimate of the underlying MDP (e.g., by computing the relative frequencies that appear in the sequence of interactions), and then use classical DP algorithms, and (ii) *model-free methods* such as $SARSA$, $TD(\lambda)$, and the popular *Q Learning* (Watkins, 1989), that do not compute such an estimate.

2.2. Local-Appearance Methods

Besides RL, the local-appearance paradigm in Computer Vision is the other technique required by our algorithms. Formally, let us call F the infinite set of local descriptors that can be generated by the chosen description method (Mikolajczyk & Schmid, 2003). The elements of F will be equivalently referred to as *visual features*. Usually, F corresponds to $\mathbb{R}^n$ for some $n \geq 1$. We assume the existence of a *visual feature detector*, that is a Boolean function $\mathcal{F} : P \times F \mapsto \mathcal{B}$ testing whether a given image exhibits a given local descriptor at one of its interest points (Schmid et al., 2000). Any suitable metric can be used to test the similarity of two visual features, e.g. Mahalanobis' or Euclidean distance.

3. Learning Visual Mappings

As discussed in the Introduction, we propose to insert an image classifier before the RL algorithm. The classifier will be iteratively refined. At any step k, the classifier, which will be denoted $\mathcal{C}_k$, partitions the visual perceptual space P into a finite number m_k of *visual classes* $\{V_1^k, \ldots, V_{m_k}^k\}$. The classification is done according to the local-appearance paradigm, by focusing the attention of the agent on highly distinctive local descriptors detected at interest points in the images. The goal of our visual learning algorithms is to iteratively refine a sequence of classifiers by successively selecting new visual features, starting with a classifier $\mathcal{C}_0$ that maps all of its input images in a single visual class $V_{0,1}$.

3.1. Image Classifier

Because of this incremental process, a natural way to implement the classifiers is to use binary decision trees. The visual classes correspond to the leaves of the trees. Each internal node is labeled by the visual feature, the presence of which is to be tested in that node. To classify an image, the system starts at the root node, then progresses down the tree according to the result of the feature detector $\mathcal{F}$ for each visual feature found during the descent, until reaching a leaf. To refine a visual class using a feature, it is sufficient to replace the leaf corresponding to this class by an internal node testing the presence or absence of this feature.

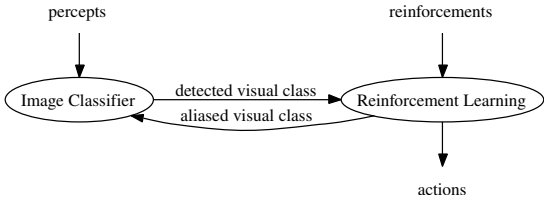


Figure 1. The information flow in RLVC.

3.2. Learning Architecture

We are interested in computing an image-to-action mapping $\pi : P \mapsto A$. At step k, given a visual percept $p \in P$, rather than applying RL algorithms directly on the raw values of the pixels of p, the RL algorithm is supplied with the output of $\mathcal{C}_k(p)$. The resulting architecture will be referred to as *Reinforcement Learning of Visual Classes* (RLVC), and is depicted in Figure 1.

RLVC consists of two interleaved, different learning processes: (i) the selection of the distinctive visual features, and (ii) the Reinforcement Learning of a mapping from the visual classes to the actions. The two main challenges in RLVC are therefore (i) to identify when a refinement of the image classifier is needed, and (ii) to select visual features that are suitable to refine the classifier.

Until the agent has learned the visual classes required to complete its task, visual percepts needing different reactions may be mapped to the same class. Therefore, the right decisions cannot always be made by the embedded RL algorithm on the basis of its inputs. From the viewpoint of the embedded RL algorithm, the input space is only partially observable (cf. Section 2.1). This phenomenon is known as the *perceptual aliasing* (or *hidden state*) problem (Whitehead & Ballard, 1991). Hence, detecting when a refinement is needed is actually equivalent to detecting in which visual classes perceptual aliasing occurs. We discuss how this detection is achieved in RLVC in the following section.

3.3. Detecting the Aliased Visual Classes

3.3.1. Projecting MDPs through a Classifier

Formally, any image classifier $\mathcal{C}_k$ converts a sequence of N interactions $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$ to a *mapped sequence* of N quadruples $\langle \mathcal{C}_k(p_t), a_t, r_{t+1}, \mathcal{C}_k(p_{t+1}) \rangle$. Let use define the *mapped MDP* $\mathcal{M}_k$ as the MDP

$\langle S_k, A, \mathcal{T}_k, \mathcal{R}_k \rangle$ obtained from the mapped sequence, where S_k is the set of visual classes that are known to $\mathcal{C}_k$, and where $\mathcal{T}_k$ and $\mathcal{R}_k$ have been computed using the relative frequencies in the mapped sequence.

Consider two visual classes $V, V' \in \{V_1^k, \ldots, V_{m_k}^k\}$ and one action $a \in A$. We define the following functions:

- $\delta_t(V, a)$ equals 1 if $\mathcal{C}_k(p_t) = V$ and $a_t = a$, and 0 otherwise;
- $\delta_t(V, a, V')$ equals 1 if $\mathcal{C}_k(p_t) = V$, $a_t = a$ and $\mathcal{C}_k(p_{t+1}) = V'$, and 0 otherwise;
- $\eta(V, a)$ is the number of t such that $\delta_t(V, a) = 1$.

Using this notation, we can write:

- $S_k = \{V_1^k, \ldots, V_{m_k}^k\}$;
- $\mathcal{T}_k(V, a, V') = \sum_{t=1}^{N} \delta_t(V, a, V')/\eta(V, a)$;
- $\mathcal{R}_k(V, a) = \sum_{t=1}^{N} r_t \delta_t(V, a)/\eta(V, a)$.

3.3.2. Optimal Q Function for a Mapped MDP

Each mapped MDP $\mathcal{M}_k$ induces an optimal Q function on the domain $S_k \times A$ that will be denoted $Q_k'^*$. Computing $Q_k'^*$ can be difficult: In general, there may exist no MDP defined on the state space S_k and on the action space A that can generate a given mapped sequence, since the latter is not necessarily Markovian anymore. So, if some RL algorithm is run on the mapped sequence, it might not converge toward $Q_k'^*$, or not even converge at all. However, when applied on a mapped sequence, any model-based RL method (cf. Section 2.1) can be used to compute $Q_k'^*$ if $\mathcal{M}_k$ is used as the underlying model. Under some conditions, Q Learning also converges to the optimal Q function of the mapped MDP (Singh et al., 1995).

In turn, the function $Q_k'^*$ induces another Q function on the initial domain $S \times A$ through the relation $Q_k^*(s, a) = Q_k'^*(\mathcal{C}_k(p_s), a)$, where p_s is an arbitrary percept that s can generate. In the absence of aliasing, the agent could perform optimally, and Q_k^* would correspond to Q^*, according to Bellman's theorem that states the uniqueness of the optimal Q function (cf. Section 2.1). By Equation (2), the matrix $B_k = HQ_k^* - Q_k^*$ is therefore a measure of the aliasing induced by the image classifier $\mathcal{C}_k$. In RL terminology, B_k is *Bellman's residual* of the function Q_k^*. The basic idea behind RLVC is to refine the states that have a non-zero Bellman's residual.

3.3.3. Measuring Aliasing

Unfortunately, in practice, the H transform is unknown, because the agent does not have direct access to the state space of the MDP modeling the environment. We propose a different approach, by assuming temporarily that the transition function $\mathcal{T}$ of the environment is deterministic. In this case, Bellman's residual of Q_k^* can be rewritten as:

$$B_k(s, a) = \mathcal{R}(s, a) + \gamma \max_{a' \in A} Q_k^*(\mathcal{T}(s, a), a') - Q_k^*(s, a)$$

$$= \mathcal{R}(s, a) + \gamma \max_{a' \in A} Q_k'^* \left(\mathcal{C}_k \left(p_{\mathcal{T}(s,a)}\right), a'\right) - Q_k'^*(\mathcal{C}_k(p_s), a) \quad (3)$$

Consider an action $a \in A$ and two states $s, s' \in S$ of the environment. Interestingly, Equation (3) allows to compute Bellman's residual for the pair (s, a), if we have an interaction $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$ such that p_t has been generated by s, p_{t+1} has been generated by s', and a_t corresponds to a. Indeed, in the latter case, $B_k(s, a)$ equals:

$$\Delta_t = r_{t+1} + \gamma \max_{a' \in A} Q_k'^*(\mathcal{C}_k(p_{t+1}), a') - Q_k'^*(\mathcal{C}_k(p_t), a_t). \quad (4)$$

According to the previous discussion, the residuals Δ_t measure the perceptual aliasing induced by $\mathcal{C}_k$.[2] A nonzero Δ_t indicates the presence of perceptual aliasing in the visual class $V_t = \mathcal{C}_k(p_t)$ with respect to action a_t. Our criterion for detecting the aliased classes consists in computing the $Q_k'^*$ function, then in sweeping again all the $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$ to identify nonzero Δ_t.

3.4. Extracting Distinctive Visual Features

Once aliasing has been detected in some visual class $V \in S_k$ with respect to an action a, we need to discover a new local descriptor that best explains the variations in the set of Δ_t values corresponding to V and a. This is a classification problem, for which we suggest an adaptation of a popular splitting rule used with decision trees (Quinlan, 1993).

Let us denote T the set of time stamps t such that $\mathcal{C}_k(p_t) = V$ and $a_t = a$. Each threshold $c \in \mathbb{R}$ induces a binary partition $\{T_1^c, T_2^c\}$ of the set T, where:

$$\begin{aligned} T_1^c &= \{t \in T \mid \Delta_t \leq c\}, \\ T_2^c &= \{t \in T \mid \Delta_t > c\}. \end{aligned}$$

Similarly, any visual feature $f \in F$ splits T into two subsets:

$$\begin{aligned} T_1^f &= \{t \in T \mid \mathcal{F}(p_t, f)\}, \\ T_2^f &= \{t \in T \mid \neg\mathcal{F}(p_t, f)\}. \end{aligned}$$

[2] It is worth noticing that $\alpha_t \Delta_t$ corresponds to the updates that would be applied by Q Learning (Watkins, 1989), where α_t is known as the *learning rate* at time t.

Ideally, we would like to identify a visual feature f that splits T the same way a threshold c would.

Therefore, for each cutpoint in the sorted sequence of Δ_t such that $t \in T$, we select the visual feature that maximizes a given information-theoretic score for the partition of images induced by the feature detector $\mathcal{F}$. This is done by iterating over the local descriptors that describe the neighborhood of all the interest points of the images in $\{p_t \mid t \in T\}$. Finally, the visual feature that has the maximal score among all the extracted features is used to refine the classifier.

Our algorithms assume that the transition function $\mathcal{T}$ behaves deterministically to be able to directly target Bellman's residuals. Of course, interesting environments are in general non-deterministic, which generates variations in Bellman's residuals that are not a consequence of perceptual aliasing. RLVC can be made somewhat robust to such a variability by introducing a statistical hypothesis test: For each candidate split, a χ^2 test is used to decide whether it is significantly different from a random split. This approach is inspired from decision tree pruning.

3.5. Algorithm

Putting it all together, RLVC can be described as:

1. Begin with a counter $k \leftarrow 0$ and a classifier $\mathcal{C}_k$ that maps all the percepts to the same visual class.
2. Construct an optimal function Q'^*_k for the classification induced by $\mathcal{C}_k$, e.g. using model-based methods. Record in a database the interactions $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$ encountered during this RL process.
3. Let $\mathcal{C}_{k+1} \leftarrow \mathcal{C}_k$. For each $(i, a) \in \{1, \ldots, m_k\} \times A$:
   (a) Select all the interactions $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$ in the database such that $\mathcal{C}_k(p_t) = V_i^k$ and $a_t = a$.
   (b) For each of them, compute the value:
   $$\xi_t = r_{t+1} + \gamma \max_{a' \in A} Q'^*_k(p_{t+1}, a').$$
   Note that since $Q'^*_k(\mathcal{C}_k(p_t), a_t) = Q'^*_k(V_i^k, a)$ is a constant c for all the selected interactions, ξ_t is just the Δ_t from Equation (4), offset by c.
   (c) If some of the ξ_t are different from c, select a distinctive feature that best explains the variations in the set of ξ_t. If the hypothesis test succeeds, refine the classifier $\mathcal{C}_{k+1}$ by splitting the class V_i^k according to the newly selected feature.
4. If $\mathcal{C}_k \neq \mathcal{C}_{k+1}$, let $k \leftarrow k + 1$ and go to step 2.

At the end of the algorithm, the classifier $\mathcal{C}_k$ is assumed free of perceptual aliasing, and can be subsequently used to control the system.

4. Experiments on a Navigation Task

We have evaluated our system on an abstract task that closely parallels a real-world scenario while avoiding any unnecessary complexity. RLVC has succeeded at solving the continuous, noisy visual navigation task depicted in Figure 2. The goal of the agent is to reach as fast as possible one of the two exits of the maze. The set of possible locations is continuous. At each location, the agent has four possible actions: Go up, right, down, or left. Every move is altered by a Gaussian noise, the standard deviation of which is 2% the size of the maze. Glass walls are present in the maze. Whenever a move would take the agent into a wall or outside the maze, its location is not changed.

The agent earns a reward of 100 when an exit is reached. Any other move, including the forbidden ones, generates zero reinforcement. When the agent succeeds in escaping the maze, it arrives in a terminal state in which every move gives rise to a zero reinforcement. In this task, γ was set to 0.9. Note that the agent is faced with the delayed reward problem, and that it must take the distance to the two exits into consideration for choosing the most attractive one.

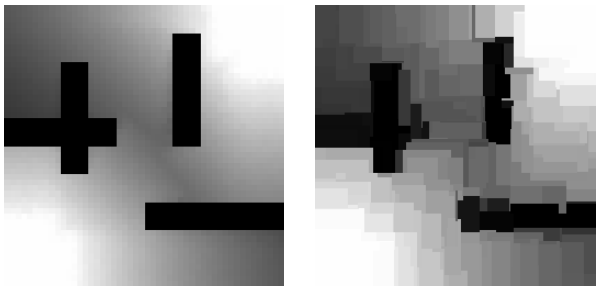
The maze has a ground carpeted with a color image of 1280×1280 pixels that is a montage of pictures from the COIL-100 database[3]. The agent does not have direct access to its (x, y) position in the maze. Rather, its sensors take a picture of a surrounding portion of the ground. This portion is larger than the blank areas, which makes the input space fully observable. Importantly, the glass walls are transparent, so that the sensors also return the portions of the tapestry that are behind them. Therefore, there is no way for the agent to directly locate the walls. It is obliged to identify them as the regions of the maze in which an action does not change its location.

In this experiment, we have used color differential invariants as visual features (Gouet & Boujemaa, 2001). The entire tapestry includes 2298 different visual features. RLVC selected 200 features, corresponding to a ratio of 9% of the entire set of possible features. The computation stopped when k reached 84, which took 35 minutes on a 2.4GHz Pentium IV using databases of 10,000 interactions. 205 visual classes were identified. This is a small number, compared to the number of perceptual classes that would be generated by a dis-

[3] `http://www.cs.columbia.edu/CAVE/coil-100.html`

Figure 2. A continuous, noisy navigation task. The exits of the maze are indicated by boxes with a cross. Walls of glass are identified by solid lines. The agent is depicted at the center of the figure. Each one of the four possible moves is represented by an arrow, the length of which corresponds to the resulting move. The sensors return a picture that corresponds to the dashed portion of the image.

Figure 3. The resulting deterministic image-to-action mapping $\pi^* = \text{argmax}_{a \in A} Q_k^*(s,a)$, sampled at regularly-spaced points. It manages to choose the correct action at each location.



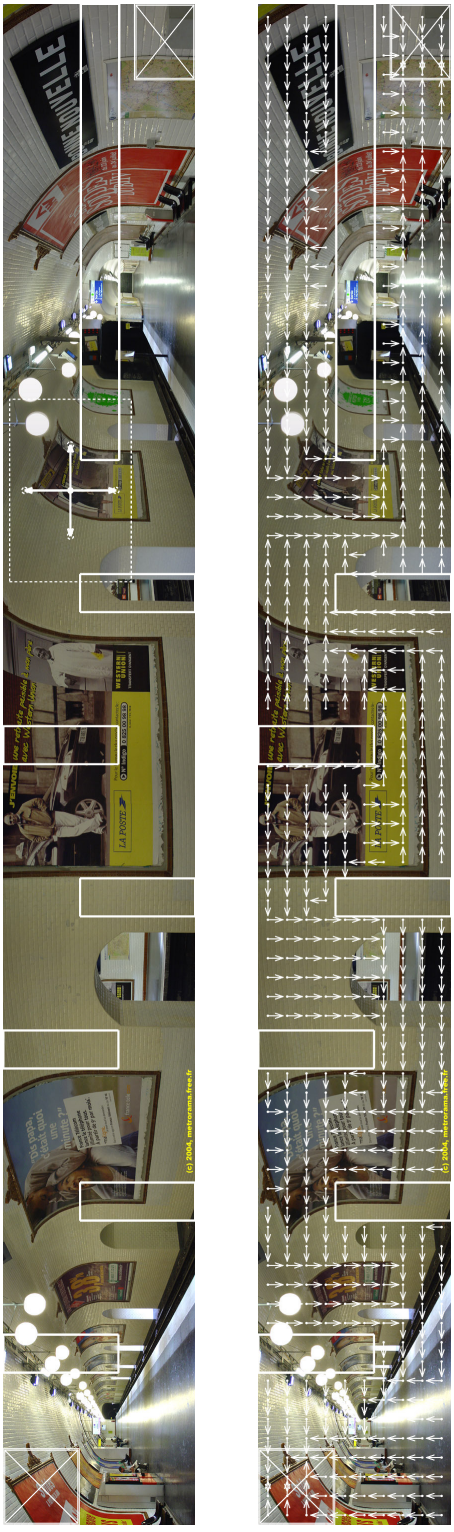
(a) (b)

Figure 4. (a) The optimal value function, when the agent has direct access to its (x, y) position in the maze and when the set of possible locations is discretized into a 50×50 grid. The brighter the location, the greater its value. (b) The optimal value function obtained by RLVC.

cretization of the maze when the agent knows its (x, y) position. For example, a reasonably sized 20×20 grid leads to 400 perceptual classes.

The optimal, deterministic image-to-action mapping that results from the last obtained image classifier $\mathcal{C}_k$ is shown in Figure 3. In RL, it is generally instructive to study the *optimal value function* $V^*(s)$ of each state, that corresponds to the expected discounted return when the agent always chooses the optimal action in each state, i.e. $V^*(s) = \max_{a \in A} Q^*(s,a)$. Figure 4 compares the optimal value function of the discretized problem with the one obtained through RLVC. The similarity between the two pictures indicates the soundness of our approach. Importantly, RLVC operates with neither pre-treatment, nor human intervention. The agent is initially not aware of which visual features are important for its task. Moreover, the interest of selecting descriptors is clear in this application: A direct, tabular representation of the Q function considering all the Boolean combinations of features would have $2^{2298} \times 4$ cells.

The behavior of RLVC on real-word images has also been investigated. The navigation rules were kept identical, but the tapestry was replaced by a panoramic photography of 3041×384 pixels of a subway station, as depicted in Figure 5. RLVC took 101 iterations to compute the mapping at the right of Figure 5. The computation time was 159 minutes on a 2.4GHz Pentium IV using databases of 10,000 interactions. 144 distinct visual features were selected among a set of 3739 possible ones, generating a set of 149 visual classes. Here again, the resulting classifier is fine enough to obtain a nearly optimal image-to-action mapping for the task. Our framework therefore seems applicable to realistic Computer Vision applications.



(a) (b)

Figure 5. (a) A navigation task with a real-world image, using the same conventions than Figure 2. (b) The deterministic image-to-action mapping computed by RLVC.

5. Related Work

The idea of building a decision tree that tests the presence of Boolean features to partition a large state space into a piecewise constant value function goes back to the G Algorithm (Chapman & Kaelbling, 1991). McCallum's U Tree algorithm builds upon this idea by combining this so-called "selective attention" mechanism with a short-term memory that enables the learning agent to deal with partially observable environments (McCallum, 1996). Decision trees have also been used to deal with continuous input spaces in the context of Continuous U Tree (Uther & Veloso, 1998) and Variable Resolution Grids (Munos & Moore, 2002). However, our work appears to be the first to relate Reinforcement Learning with the local-appearance paradigm in Computer Vision. The use of visual features in fact appears to be a natural and powerful tool for solving visual, reactive tasks.

A large number of criteria for deciding the presence of aliasing have been proposed in the literature (Whitehead & Ballard, 1991; Chrisman, 1992; McCallum, 1996; Munos & Moore, 2002). The aliasing criterion used in RLVC is defined independently of any fixed RL algorithm, and is very close to that used by Continuous U Tree (Uther & Veloso, 1998), with the major exception that RLVC deals with Boolean features, whereas Continuous U Tree works in a continuous input space.

6. Conclusions

We have introduced *Reinforcement Learning of Visual Classes* (RLVC). RLVC is designed to learn mappings that directly connect visual stimuli to output actions that are optimal for the surrounding environment. The learning process is closed-loop and flexible. It consists in taking lessons from interactions with the environment, which is similar to the way living beings learn to solve everyday tasks. RLVC focuses the attention of an embedded Reinforcement Learning algorithm on highly informative and robust parts of the inputs by testing the presence or absence of local descriptors at the interest points of the input images.

The relevant visual features are incrementally selected in a sequence of attempts to remove perceptual aliasing. Importantly, this selection is task-driven: Different tasks will not necessarily lead to the selection of the same subset of features. This is similar to human visual learning, for which there is strong evidence that new visual classes are learned when the task requires it (Schyns & Rodet, 1997). In this sense, the paradigm of RLVC can also be motivated from a neuropsychological point of view, and not only as an *ad-hoc* machine

learning algorithm.

The area of applications is wide, since nowadays robotic agents are often equipped with CCD sensors. Our long-term goal would be to construct a robotic system that autonomously learns to structure its visual system to solve a reactive task, such as grasping objects (Coelho et al., 2001). RLVC could also be potentially be applied to Human-Computer Interaction, as the actions need not be physical actions.

However, many questions are still open. For example, it is unclear how RLVC could be applied on continuous action spaces, which is required for the example of grasping. It would also be interesting to add post-pruning operations to get rid of selected features that subsequently prove useless for the task, and that generate overfitting effects. Finally, the efficacy of RLVC clearly depends on the discriminative power of the visual features. If this power is insufficient, the algorithm will not be able to completely remove the aliasing, which will produce a sub-optimal control law. For future research, we therefore suggest the combination of RLVC with techniques for disambiguating between aliased percepts using a short-term memory (McCallum, 1996), or for building higher-level geometrical combinations of visual features that are more robust to noise (Piater, 2001; Scalzo & Piater, 2005).

References

Bellman, R. (1957). *Dynamic programming*. Princeton University Press.

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Athena Scientific.

Chapman, D., & Kaelbling, L. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. *Proc. of the 12th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 726–731). Sydney.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. *National Conference on Artificial Intelligence* (pp. 183–188).

Coelho, J., Piater, J., & Grupen, R. (2001). Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems*, *37*, 195–218.

Gibson, E., & Spelke, E. (1983). The development of perception. *Handbook of child psychology vol. iii: Cognitive development*, chapter 1, 2–76. Wiley.

Gouet, V., & Boujemaa, N. (2001). Object-based queries using color points of interest. *IEEE Workshop on Content-Based Access of Image and Video Libraries* (pp. 30–36). Kauai (HI, USA).

Lowe, D. (1999). Object recognition from local scale-invariant features. *International Conference on Computer Vision* (pp. 1150–1157). Corfu, Greece.

McCallum, R. (1996). *Reinforcement learning with selective perception and hidden state*. Doctoral dissertation, University of Rochester, New York.

Mikolajczyk, K., & Schmid, C. (2003). A performance evaluation of local descriptors. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 257–263). Madison (WI, USA).

Munos, R., & Moore, A. (2002). Variable resolution discretization in optimal control. *Machine Learning*, *49*, 291–323.

Piater, J. (2001). *Visual feature learning*. Doctoral dissertation, University of Massachusetts, Computer Science Department, Amherst (MA, USA).

Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc.

Scalzo, F., & Piater, J. (2005). Task-driven learning of spatial combinations of visual features. *Proc. of the IEEE Workshop on Learning in Computer Vision and Pattern Recognition*. San Diego (CA, USA).

Schmid, C., & Mohr, R. (1997). Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 530–535.

Schmid, C., Mohr, R., & Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, *37*, 151–172.

Schyns, P., & Rodet, L. (1997). Categorization creates functional features. *Journ. of Experimental Psychology: Learning, Memory and Cognition*, *23*, 681–696.

Singh, S., Jaakkola, T., & Jordan, M. (1995). Reinforcement learning with soft state aggregation. *Advances in Neural Information Processing Systems* (pp. 361–368). MIT Press.

Sutton, R., & Barto, A. (1998). *Reinforcement learning, an introduction*. MIT Press.

Uther, W. T. B., & Veloso, M. M. (1998). Tree based discretization for continuous state space reinforcement learning. *Proc. of the 15th National Conference on Artificial Intelligence (AAAI)* (pp. 769–774). Madison (WI, USA).

Watkins, C. (1989). *Learning from delayed rewards*. Doctoral dissertation, King's College, Cambridge.

Whitehead, S., & Ballard, D. (1991). Learning to perceive and act by trial and error. *Machine Learning*, *7*, 45–83.