Task-Driven Learning of Spatial Combinations of Visual Features

Sébastien Jodogne*, Fabien Scalzo and Justus H. Piater
Institut Montefiore (B28), Université de Liège
B-4000 Liège, Belgique
{S.Jodogne,FScalzo,Justus.Piater}@ULg.ac.be

Abstract

Solving a visual, interactive task can often be thought of as building a mapping from visual stimuli to appropriate actions. Clearly, the extracted visual characteristics that index into the repertoire of actions must be sufficiently rich to distinguish situations that demand distinct actions. Spatial combinations of local features permit, in principle, the construction of features at various levels of discriminative power. We present an algorithm for selecting relevant spatial combinations of visual features by exercising a given task in a closed-loop learning process based on Reinforcement Learning. The algorithm operates by progressively splitting the perceptual space into distinct regions. Whenever the agent detects perceptual aliasing of distinct world states, it constructs a spatial combination of visual features that disambiguates the aliased states. We demonstrate the efficacy of our algorithm on a version of the classical "Car on the Hill" control problem where position and velocity are presented to the agent visually, in a way that the task is unsolvable using individual point features.

1. Introduction

Strong neuropsychological evidence suggests that human beings learn to extract useful information from visual data in an *interactive* fashion, without any external supervisor [5]. 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 certainly *task-driven*, since different tasks do not necessarily need to make the same distinctions [17]. This way, as we interact with the outside world, we gain more and more expertise on our tasks [19].

Reinforcement Learning (RL) is a successful approach for modeling the behavior of an artificial agent that learns how to perform its task through its interactions with the environment [2, 18]. In RL, the agent learns to connect its sensory inputs to the appropriate actions. It is never told what action it should take; rather, when it does a good or a bad action, it only receives a reward or a punishment, the *reinforcement signal*. Schematically, RL lies between supervised learning (where an external teacher gives the correct action to the agent) and unsupervised learning (in which no clue about the goodness of the action is given). Unfortunately, RL is highly sensitive to the number of distinct percepts as well as to the noise that results from the sensing process. This makes RL inapplicable for the direct closed-loop learning of image-to-action mappings.

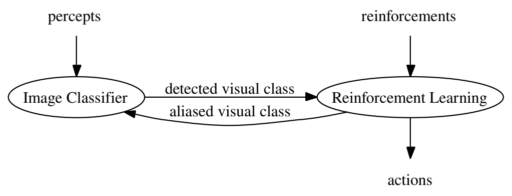


Figure 1: The information flow in Reinforcement Learning of Visual Classes.

We have recently proposed a new RL paradigm, called *Reinforcement Learning of Visual Classes* (RLVC) [8], that makes such a learning possible. In RLVC, an image classifier is introduced in front of a classical RL algorithm. This classifier partitions the perceptual space into a finite set of distinct regions according to the local-appearance paradigm, by focusing the attention of the agent on highly distinctive local descriptors located at interest points of the visual stimuli [15]. The detected class number is given as input to the embedded RL algorithm, as shown in Figure 1.

Initially, the classifier maps all the possible percepts to the same perceptual class. Of course, this introduces *perceptual aliasing* [10, 21], since, in general, two states requiring different reactions are merged into this single class. Whenever the agent identifies perceptual aliasing of distinct world states, it dynamically selects a local-appearance feature that disambiguates the aliased states. The selected feature is used to refine the classifier. When no further aliasing is detected, this iterative process is stopped, and the resulting image classifier is used to control the system. Thus, RLVC can be thought of as performing an adaptive discretization of the perceptual space.

*Aspirant – Fonds National de la Recherche Scientifique Belge (FNRS)

The efficacy of RLVC clearly depends on the discriminative power of the visual features. If their power is insufficient, the algorithm will not be able to remove completely the aliasing, which will produce a sub-optimal control law. Practical experiments on simulated visual navigation tasks have exhibited this deficiency, as soon as the number of detected visual features is reduced or as features are made more similar by using a less sensitive metric.

Now, most objects encountered in the world are composed of a number of distinct constituent parts (e.g., a face contains a nose and two eyes, a phone possesses a keypad). These parts are themselves recursively composed of other subparts (e.g., an eye contains an iris and eyelashes, a keypad is composed of buttons). Such a hierarchical physical structure certainly imposes strong constraints on the spatial disposition of the visual features.

In this paper, we show how highly informative spatial combinations of visual features can be iteratively constructed using only reinforcement feedback. Our algorithms are general, since they can be applied to any visual control problem that can be defined in the RL framework. This result is promising for it permits the construction of features at increasingly higher levels of discriminative power, enabling us to tackle visual tasks that are unsolvable using individual point features alone. To the best of our knowledge, this paper appears to be the very first attempt to build visual feature hierarchies in a closed-loop, interactive and task-driven learning process.

2. Reinforcement Learning

In RL, the environment is traditionally modeled as a *Markov Decision Process* (MDP). A 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 $\mathbb{R}$. A 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 $\mathcal{T}(s_t, a_t, s_{t+1})$.

In general, the agent does not have a direct access to the state s_t it lies in, neither to the state s_{t+1} it reaches. Rather, it perceives the current and the next state through its sensors. Let us define the *perceptual space* P as the set of possible percepts 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 system is defined as a quadruple $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$.

If each percept can only be generated by one state, the perceptual space is said *fully observable*, and the agent is able to distinguish between the states of the environment using only its sensors. Otherwise, the perceptual space is called *partially observable*. Throughout this paper, we will only consider fully observable perceptual spaces. This contrasts our scenario from established work on *Partially Observable Markov Decision Processes* (POMDPs) [9].

A *percept-to-action mapping* is a fixed probabilistic function $\pi : P \mapsto A$ from the percepts to the 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:

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}, \tag{1}$$

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 maximizing 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:

$$(HQ)(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{T}(s, a, s') \max_{a' \in A} Q(s', a'), \tag{2}$$

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 [1]:

$$HQ^* = Q^*. \tag{3}$$

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 $s \in S$. Hence, DP reduces the Markov Decision Problem to the solution of the non-linear system of Equations (3). However, the direct solution is seldom used in practice, in favor of incremental algorithms such as *Value Iteration* [2].

Reinforcement Learning (RL) [2] is a set of algorithmic methods for solving Markov Decision Problems when the underlying MDP is unknown. 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. The most popular RL algorithm is certainly *Q Learning* [20].

3. Dynamic Selection of Features

In this section, we give a short description of the *Reinforcement Learning of Visual Classes* (RLVC) algorithm. We invite the interested reader to refer to our previous paper for a more thorough discussion [8]. As stated in the Introduction, RLVC iteratively refines an image classifier by successively selecting new visual features. After k refinement steps, the classifier, which will be denoted $\mathcal{C}_k$, partitions the visual perceptual space P into a finite number m_k of *visual classes* $\{V_1^k, \ldots, V_{m_k}^k\}$. $\mathcal{C}_k$ is a binary decision tree, each internal node of which tests the presence of a given visual feature, and each leaf of which corresponds to a visual class. Here is the outline of the algorithm:

Algorithm 1 RLVC General Structure

```
1:  $k \leftarrow 0$
2:  $m_k \leftarrow 1$
3:  $\mathcal{C}_k$ is a binary decision tree with only one leaf
4:  repeat
5:      Collect $N$ interactions $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$
6:      Apply an arbitrary RL algorithm to obtain $Q_k^*$
7:      $\mathcal{C}_{k+1} \leftarrow \mathcal{C}_k$
8:      for all $i \in \{1, \ldots, m_k\}$ do
9:          if aliased($V_i^k$) then
10:             $v \leftarrow$ select($V_i^k$)
11:             if $v \neq \perp$ then
12:                 In $\mathcal{C}_{k+1}$, replace the leaf corresponding to $V_i^k$
                    by an internal node that tests the presence of
                    $v$, and that leads to two new leaves corre-
                    sponding to visual classes $V_i^{k+1}$ and $V_{m_k+1}^{k+1}$
13:                 $m_k \leftarrow m_k + 1$
14:             end if
15:         end if
16:     end for
17:     $k \leftarrow k + 1$
18: until $\mathcal{C}_k = \mathcal{C}_{k+1}$
19: Control the system with the image-to-action mapping
    $\pi^*$ defined as $\pi^*(p) = \text{argmax}_{a \in A} Q_k^*(p, a)$ for all $p$
```

In order to turn this learning structure into a working algorithm, we still have to describe how to detect aliasing and how to select a visual feature that removes this aliasing. It is important to notice that Q_k^* depends on the image classifier $\mathcal{C}_k$ through which it was computed. In the absence of aliasing, the agent could act optimally, and $\mathcal{C}_k$ would correspond to Q^*, by Bellman's theorem that states the uniqueness of the optimal Q function. By Equation (3), 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 known as *Bellman's residual* of the function Q_k^*. The basic idea behind RLVC is to refine the visual classes that have a non-zero Bellman's residual.

The underlying MDP, and hence the H transform, are generally unknown in the RL framework. It is therefore impossible for the agent to compute Bellman's residuals directly. However, if the transition function $\mathcal{T}$ of the MDP is deterministic, it is still possible to compute the value of $B_k(p, a)$ by acquiring a single interaction $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$ such that $p_t = p$ and $a_t = a$. Using Equation (2), we get:

$$B_k(p, a) = HQ_k^*(p, a) - Q_k^*(p, a)$$

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

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

According to the previous discussion, a non-zero $B_k(p_t, a_t)$ for some time stamp t indicates the presence of aliasing in the visual class $\mathcal{C}_k(p_t)$ with respect to the action a_t. This leads us to the following algorithm for detecting aliasing:

Algorithm 2 Aliasing Criterion: aliased(V_i^k)

```
1: for all $a \in A$ do
2:     for all $t$ such that $\mathcal{C}_k(p_t) = V_i^k$ and $a_t = a$ do
3:         Compute $B_k(p_t, a_t)$ through Equation (4)
4:         if $B_k(p_t, a_t) \neq 0$ then
5:             return true
6:         end if
7:     end for
8: end for
9: return false
```

Once aliasing has been detected in some visual class V_i^k, we are interested in discovering a visual feature that contributes to the removal of this aliasing, i.e. that best explains the variations in Bellman's residuals. Ideally, such a feature would split the percepts classified as V_i^k in the same way as a threshold on the residuals. Of course, since the transition function is assumed deterministic, such variations can also come from the non-deterministic nature of the environment. Therefore, for each selected visual feature, a χ^2 hypothesis test is used to decide whether the split it implies is significantly different from a random spit. This technique is quite close to the construction of decision trees with pruning [14], and is summarized in Algorithm 3.

4. Learning Spatial Relationships

As motivated in the Introduction, we propose to extend RLVC by constructing a hierarchy of spatial arrangements of individual point features. This structure is built simultaneously with the image classifier. As soon as no sufficiently informative visual feature can be extracted, the algorithm tries to combine two visual features in order to construct a higher level of abstraction, which is hopefully more distinctive and more robust to noise. Our method subsumes the co-existence of two different kinds of visual features:

Algorithm 3 Feature Selection: select(V_i^k)

```
1: $v^* \leftarrow \perp$        {*Best feature found so far*}
2: $s^* \leftarrow -\infty$     {*Information-theoretic score of $v^*$*}
3: **for all** $a \in A$ **do**
4:   $\Delta \leftarrow \{\}$
5:   $F \leftarrow \{\}$
6:   **for all** $t$ such that $\mathcal{C}_k(p_t) = V_i^k$ and $a_t = a$ **do**
7:     $\Delta \leftarrow \Delta \cup \{B_k(p_t, a_t)\}$
8:     $F \leftarrow F \cup \{v \mid v$ is a feature exhibited by $p_t\}$ [1]
9:   **end for**
10:  **for all** cutpoints $c$ in the sequence $\Delta$ of residuals **do**
11:    $F' \leftarrow F$     {*Used when combining features*}
12:    **for all** visual feature $v \in F'$ **do**
13:      $s \leftarrow$ the information gain provided by $v$ on the
         partition induced by $c$
14:      **if** $s \geq s^*$ and the $\chi^2$ test succeeds **then**
15:        $v^* \leftarrow v$
16:        $s^* \leftarrow s$
17:      **end if**
18:    **end for**
19:  **end for**
20: **end for**
21: **return** $v^*$
```

Primitive Features: They correspond to the individual point features, i.e. to the local-appearance descriptors. They are simply vectors of reals describing the appearance of the neighborhood around interest points of the images. Our algorithms are independent of the choice of the interest point detector [16] and of the local description technique [11].

Composite Features: They consist of spatial combinations of lower-level visual features. There is no *a priori* bound on the maximal height of the hierarchy. Therefore, a composite feature can be combined with a primitive one, or with a composite one.

4.1. Detection of Visual Features

A natural way to represent such a hierarchy is to use a directed acyclic graph $G = (V, E)$, in which each vertex $v \in V$ corresponds to a visual feature, and in which each edge $(v, v') \in E$ models the fact that v' is a part of the composite feature v. Thus, G must be *binary*, i.e. any vertex should have either no child, or exactly two children. The set V_P of the leaves of G corresponds to the set of primitive features, while the set V_C of its internal vertices represents the set of composite features.

[1]Note that at Step 8 of the algorithm, it is implicitly assumed that it is possible to compare the features between them, i.e. that a metric (like Euclidean or Mahalanobis') has been defined on the set of features.

Each leaf vertex $v_P \in V_P$ is annotated with a local descriptor $D(v_P)$. Similarly, each internal vertex $v_C \in V_C$ is annotated with constraints on the relative position between its parts. In this work, we consider only constraints on the distances between the constituent visual features of the composite features, and we assume that they should be distributed according to some Gaussian law $\mathcal{G}(\mu, \sigma)$ of mean μ and standard deviation σ. More precisely, let v_C be a composite feature, the parts of which are v_1 and v_2. In order to trigger the detection of v_C in an image p, there should be an occurrence of v_1 and an occurrence of v_2 in p such that their relative Euclidean distance has a sufficient likelihood τ of being generated by a Gaussian of mean $\mu(v_C)$ and standard deviation $\sigma(v_C)$. To ensure symmetry, the location of the composite feature is then taken as the mid point between the locations of v_1 and v_2.

The occurrences of a visual feature v in a percept p can be found using recursive Algorithm 4. At Step 8 of Algorithm 3, the composite features that belong to the current hierarchy and that appear in p_t (i.e., such that occurrences(v, p) $\neq \phi$) must be added to F.

Algorithm 4 Detecting Features: occurrences(v, p)

```
1: **if** $v$ is primitive **then**
2:   **return** $\{(x, y) \mid (x, y)$ is an interest point of $p$, the
     local descriptor of which corresponds to $D(v)\}$
3: **else**
4:   $O \leftarrow \{\}$
5:   $O_1 \leftarrow$ occurrences(subfeature$_1(v), p)$
6:   $O_2 \leftarrow$ occurrences(subfeature$_2(v), p)$
7:   **for all** $(x_1, y_1) \in O_1$ and $(x_2, y_2) \in O_2$ **do**
8:     $d \leftarrow \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
9:     **if** $\mathcal{G}(d - \mu(v), \sigma(v)) \geq \tau$ **then**
10:      $O \leftarrow O \cup \{((x_1 + x_2)/2, (y_1 + y_2)/2)\}$
11:    **end if**
12:  **end for**
13:  **return** $O$
14: **end if**
```

4.2. Generation of Composite Features

In this section, we describe how composite features are generated. The general idea behind our algorithm is to accumulate statistical evidence from the relative positions of the detected visual features in order to find "conspicuous coincidences" of co-occurrences of visual features. The generation of composite features takes place at Step 11 of Algorithm 3. Rather than iterating only over the visual features in F, we add the possibility of enriching F with spatial combinations of two features from F.

At Step 11 of Algorithm 3, let I be the set of visual percepts that are mapped to the considered visual class V_i^k, and that are associated with the considered action a:

$$I = \{p_t \mid \mathcal{C}_k(p_t) = V_i^k \text{ and } a_t = a\}.$$

Each cutpoint $c \in \mathbb{R}$ in the sequence of Bellman's residuals Δ induces a binary partition of the set of images I: Those images p_t for which the residuals are above c, and the others. Thus, $I = \{I_1, I_2\}$, where:

- $I_1 = \{p_t \in I \mid B_k(p_t, a) \leq c\}$, and
- $I_2 = \{p_t \in I \mid B_k(p_t, a) > c\}$.

As stated at the end of Section 3, we wish to extract a composite feature that splits the set I into the sets I_1 and I_2.

4.2.1 Identifying Spatial Relations

To achieve this goal, we first identify the pairs of visual features the occurrences of which are highly correlated within the set of images I_1 or I_2. This simply amounts to counting the number of co-occurrences for each pair of features in F, then only keeping the pairs the corresponding count of which exceeds a fixed threshold.

Let now v_1 and v_2 be two features that are highly correlated in the set I_i (with $i = 1$ or $i = 2$). A search for a meaningful spatial relationship between v_1 and v_2 is then carried out in the images of I_i that contain occurrences of both v_1 and v_2. For each such co-occurrence, we accumulate in a set Λ the distances between the corresponding occurrences of v_1 and v_2. Finally, a clustering algorithm is applied on the distribution Λ in order to detect typical distances between v_1 and v_2. For the purpose of our experiments, we have used *hierarchical clustering* [7]. For each cluster, a Gaussian is fitted by estimating a mean value μ and a standard deviation σ. Then, a new composite feature v_C is introduced in F', that has v_1 and v_2 as parts and such that $\mu(v_C) = \mu$ and $\sigma(v_C) = \sigma$.

In summary, we replace Step 11 of Algorithm 3 by a call to Algorithm 5. It is worth pointing out that, clearly, the feature combination stage is liable to combinatorial explosion, since $\mathcal{O}(N^2)$ clusterings are necessary if there are N visual features in the set of images I. Future work will address this problem.

4.2.2 Feature Validation

Algorithm 5 can generate several composite features for a given visual class V_i^k. However, at the end of Algorithm 3, at most one composite feature is to be kept.

It is important to notice that the performance of the clustering method is not critical for our purpose. Indeed, irrelevant spatial combinations are automatically discarded, thanks to the information-theoretic checks of the discriminative power of the constructed feature at Steps 13 and 14 of Algorithm 3. In fact, the reinforcement signal helps to direct the search for a good feature, which is an advantage over unsupervised methods of building feature hierarchies.

Algorithm 5 Generation of Composite Features

```
1: $I \leftarrow \{p_t \mid \mathcal{C}_k(p_t) = V_i^k$ and $a_t = a\}$
2: $I_1 \leftarrow \{p_t \in I \mid B_k(p_t, a) \leq c\}$
3: $I_2 \leftarrow \{p_t \in I \mid B_k(p_t, a) > c\}$
4: for all $i \in \{1, 2\}$ do
5:   for all $(v_1, v_2) \in F \times F$ do
6:     if enough co-occurrences of $v_1$ and $v_2$ in $I_i$ then
7:       $\Lambda \leftarrow \{\}$
8:       for all $p \in I_i$ do
9:         for all occurrences $(x_1, y_1)$ of $v_1$ in $p$ do
10:          for all occurrences $(x_2, y_2)$ of $v_2$ in $p$ do
11:            $\Lambda \leftarrow \Lambda \cup \{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\}$
12:          end for
13:        end for
14:      end for
15:      Apply a clustering algorithm on $\Lambda$
16:      for each cluster $C = \{d_1, \dots d_m\}$ in $\Lambda$ do
17:        $\mu = \mathrm{mean}(C)$
18:        $\sigma = \mathrm{stddev}(C)$
19:        Add to $F'$ a composite feature $v_C$ composed
           of $v_1$ and $v_2$, annotated with a mean $\mu$ and a
           standard deviation $\sigma$
20:      end for
21:    end if
22:  end for
23: end for
```

5. Visual "Car on the Hill" Problem

5.1. General Description

In this section, we demonstrate the efficacy of our algorithms on a version of the classical "Car on the Hill" control problem [12], where the position and velocity information is presented to the agent visually.

In this episodic task, a car (modeled by a mass point) is riding without friction on a hill, the shape of which is defined by the function:

$$H(p) = \begin{cases} p^2 + p & \text{if } p < 0, \\ p/\sqrt{1 + 5p^2} & \text{if } p \geq 0. \end{cases}$$

The goal of the agent is to reach as fast as possible the top of the hill, i.e. a location such that $p \geq 1$. At the top of the hill, the agent obtains a reward of 100. The car can thrust left or right with an acceleration of ± 4 Newtons. However, because of gravity, this acceleration is insufficient for the agent to reach the top of the hill by always thrusting toward the right. Rather, the agent has to go left for while, hence acquiring potential energy by going up the left side of the

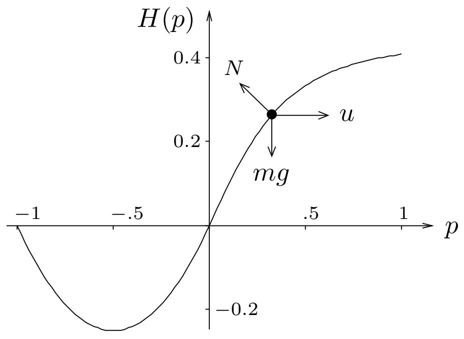


Figure 2: The "Car on the Hill" control problem.

hill, before thrusting rightward. There are two more constraints: The agent is not allowed to reach locations such that $p < -1$, and a velocity greater than 3 in absolute value leads to the destruction of the car.

5.2. Formal Definition of the Task

Formally, the set of possible actions is $A = \{-4, 4\}$, while the state space is $S = \{(p, s) \mid |p| \leq 1 \wedge |s| \leq 3\}$. The system has the following continuous-time dynamics:

$$\begin{aligned} \dot{p} &= s \\ \dot{s} &= \frac{a}{M\sqrt{1 + H'(p)^2}} - \frac{gH'(p)}{1 + H'(p)^2}, \end{aligned}$$

where $a \in A$ is the thrust acceleration, $H'(p)$ is the first derivative of $H(p)$, $M = 1$ is the mass of the car, and $g = 9.81$ is the acceleration due to gravity. These continuous-time dynamics are approximated by the following discrete-time state update rule:

$$\begin{aligned} p_{t+1} &= p_t + h\dot{p}_t + h^2\dot{s}_t/2 \\ s_{t+1} &= \dot{p}_t + h\dot{s}_t, \end{aligned}$$

where $h = 0.1$ is the integration time step. The reinforcement signal is defined through this expression:

$$\mathcal{R}((p_t, s_t), a) = \begin{cases} 100 & \text{if } p_{t+1} \geq 1 \wedge |s_{t+1}| \leq 3, \\ 0 & \text{otherwise.} \end{cases}$$

In our setup, the discount factor γ was set to 0.75.[2]

5.3. Inputs of the Agent

In previous work [4, 12], the agent was always assumed to have direct access to a numerical measure of its position and velocity. In our experimental setup, the agent is provided with two cameras, one looking at the ground underneath, the second at a velocity gauge. This way, the agent cannot directly know its current position and velocity, but has to suitably interpret its visual inputs to derive them.

Some examples of the pictures the sensors can return are presented in Figure 3. The ground is carpeted with a color image of 1280×128 pixels that is a montage of pictures from the COIL-100 database [13]. It is very important to notice that using individual point features is insufficient for solving this task, since the set of features in the pictures of the velocity gauge are always the same. To know its velocity, the agent has to generate composite features sensitive to the distance of the primitive features on the cursor with respect to the primitive features on the digits.

5.4. Results

In this experimental setup, we used color differential invariants as primitive features [6]. Among all the possible visual inputs (both for the position and the velocity sensors), there were 88 different primitive features. The entire image of the ground includes 142 interest points, whereas the images of the velocity gauge include about 20 interest points.

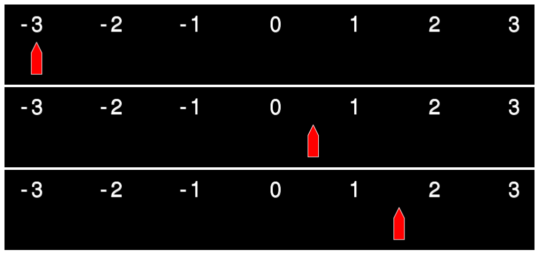
The output of RLVC is a decision tree with 157 leaves, thus defining 157 visual classes. Each internal node of this tree tests the presence of one visual feature, taken from a set of 91 highly discriminant features selected by RLVC. This implies that a given visual feature can be tested in more than one node, and that RLVC makes distinct visual features work together to delimit regions in the state space corresponding to different values of Q^*. Among the 91 selected visual features, there were 56 primitive and 26 composite features. Two examples of composites features that were selected by RLVC are depicted in Figure 5. The computation stopped after $k = 38$ refinement steps in Algorithm 1.

To show the efficacy of our method, we compare its performance with the scenario in which the agent has a direct perception of its current (p, s) state. In the latter scenario, the state space was discretized in a grid of 13×13 cells. The number 13 was chosen since it corresponds to the square root of 157, the number of visual classes that were produced by RLVC. This way, RL is provided an equivalent number of perceptual classes in the two scenarios.

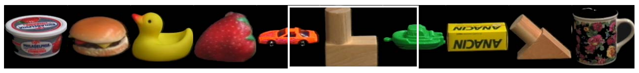
In RL, it is generally instructive to study the *optimal value function* $V^*((p, s))$ of each state $(p, s) \in S$, that corresponds to the expected discounted return when the agent starts from the state (p, s), and then always chooses the optimal action in each state, i.e. $V^*((p, s)) = \max_{a \in A} Q^*((p, s), a)$. Figure 4 compares the optimal value function of the direct-perception problem with the one obtained through RLVC. The similarity between the two pictures indicates the soundness of our approach.

We have also evaluated the performance of the optimal

[2]This definition is actually a mix of two coexistent formulations of the "Car on the Hill" task [4, 12]. The major differences with the initial formulation of the problem [12] is that the set of possible actions is discrete, and that the goal is at the top of the hill (rather than on a given area of the hill), just like in the definition from Ernst et al. [4]. To ensure the existence of an interesting solution, the velocity is allowed to remain less than 3 (instead of 2), and the integration time step is set to $h = 0.1$ (instead of 0.01).



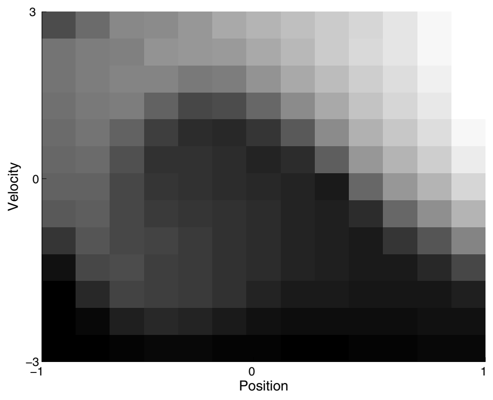
(a)



(b)

Figure 3: (a) Visual percepts corresponding to pictures of the velocity gauge when $s = -3$, $s = 0.5$ and $s = 1.5$. (b) Visual percepts returned by the position sensor. The region framed with a white rectangle corresponds to the portion of the ground that is returned by the sensor when $p = 0.1$. This portion slides back and forth as the agent moves.

(a)



(b)

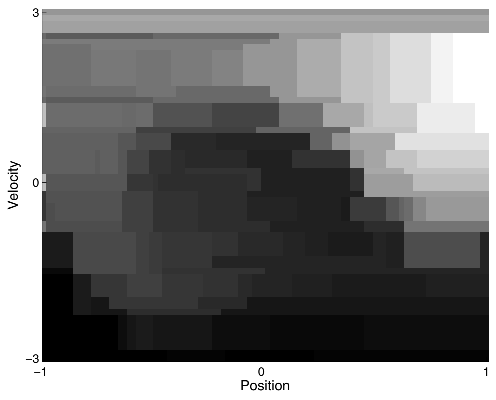


Figure 4: (a) The optimal value function, when the agent has a direct access to its current (p, s) state, and when the input space is discretized in a 13×13 grid. The brighter the location, the greater its value. (b) The value function obtained by RLVC.

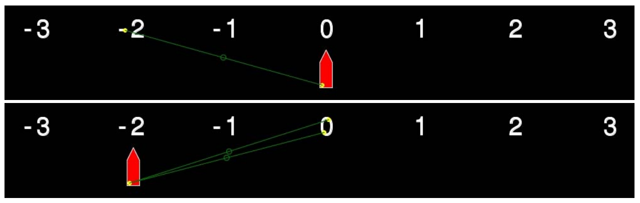


Figure 5: Two composite features that were generated, in green. The primitive features of which they are composed are marked in yellow. The first feature triggers for velocities around 0, whereas the second triggers around -2.

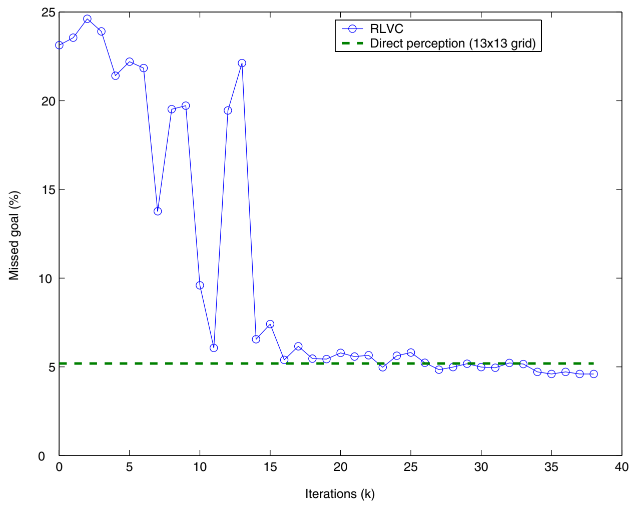


Figure 6: Evolution of the number of times the goal was missed over the iterations of RLVC.

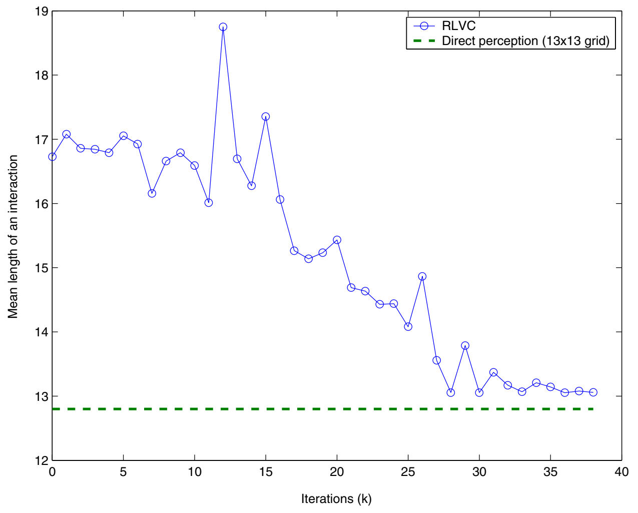


Figure 7: Evolution of the mean lengths of the successful trials over the iterations of RLVC.

image-to-action mapping $\pi^* = \text{argmax}_{a \in A} Q^*((p, s), a)$ obtained through RLVC. For this purpose, the agent was placed randomly on the hill, with an initial velocity of 0. Then, it used the mapping π^* to choose an action, until it reached a final state. A set of 10,000 such trials were carried out at each step k of Algorithm 3. Figure 6 compares the proportion of trials that missed the goal (either because of leaving the hill on the left, or because of acquiring a too high velocity) in RLVC and in the direct-perception problem. When k became greater than 27, the proportion of missed trials was always smaller in RLVC than in the direct-perception problem. This advantage in favor of RLVC is due to the adaptive nature of its discretization. Figure 7 compares the mean lengths of the successful trials. The mean length of RLVC trials clearly converges to that of the direct-perception trials, while staying slightly larger.

To conclude, RLVC achieves a performance close to the direct-perception scenario. However, the mapping built by RLVC directly links visual percepts to the appropriate actions, without considering explicitly the physical variables.

6. Conclusions

We have presented new algorithms that allow the closed-loop, reinforcement-driven generation of a hierarchy of visual features, while simultaneously learning an optimal image-to-action mapping. Highly distinctive spatial arrangements of visual features are detected in a sequence of attempts to iteratively remove perceptual aliasing. Our algorithms are flexible and defined independently of any given task, hence their potential field of application is broad. They permit the solution of visual control problems in which the use of individual point features alone would produce sub-optimal control policies.

Future directions include the demonstration of the applicability of our algorithms in a robotic application, such as grasping objects by combining visual and haptic feedback [3]. This necessitates the extension of our techniques to continuous action spaces, for which no fully satisfactory solutions exist to date. Secondly, spatial arrangements do not currently take into consideration the relative angles between the parts of a composite feature. Doing so would further increase the discriminative power of the composite features, but requires non-trivial techniques for clustering in circular domains. Finally, it would be interesting to study the behavior of our algorithms on tasks that require the construction of a hierarchy with more than two levels.

References

[1] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[2] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[3] J. Coelho, J.H. Piater, and R. Grupen. Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems, special issue on Humanoid Robots*, 37(2–3):195–218, 2001.

[4] D. Ernst, P. Geurts, and L. Wehenkel. Iteratively extending time horizon reinforcement learning. In *Proc. of the 14th European Conference on Machine Learning*, pages 96–107, Dubrovnik, 2003.

[5] E.J. Gibson and E.S. Spelke. The development of perception. *Handbook of Child Psychology Vol. III: Cognitive Development*, chapter 1, pages 2–76. Wiley, 4th edition, 1983.

[6] V. Gouet and N. Boujemaa. Object-based queries using color points of interest. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 30–36, Kauai, 2001.

[7] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

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

[9] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[10] R.A. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, New York, 1996.

[11] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 257–263, Madison, 2003.

[12] A. Moore and C. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21, 1995.

[13] S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image library (COIL-100). Technical Report CUCS-006-96, Columbia University, New York, 1996.

[14] J.R. Quinlan. The effect of noise on concept learning. In *Machine Learning: An Artificial Intelligence Approach: Volume II*, pages 149–166. Kaufmann, Los Altos, 1986.

[15] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[16] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[17] P.G. Schyns and L. Rodet. Categorization creates functional features. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 23(3):681–696, 1997.

[18] R.S. Sutton and A.G. Barto. *Reinforcement Learning, an Introduction*. MIT Press, 1998.

[19] M.J. Tarr and Y.D. Cheng. Learning to see faces and objects. *Trends in Cognitive Sciences*, 7(1):23–30, 2003.

[20] C.J.C.H. Watkins. *Learning From Delayed Rewards*. PhD thesis, King's College, Cambridge, 1989.

[21] S.D. Whitehead and D.H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83, 1991.