Learning Visual Representations for Interactive Systems

Justus Piater, Sébastien Jodogne, Renaud Detry, Dirk Kraft, Norbert Krüger, Oliver Kroemer, and Jan Peters

Abstract We describe two quite different methods for associating action parameters to visual percepts. Our *RLVC* algorithm performs reinforcement learning directly on the visual input space. To make this very large space manageable, RLVC interleaves the reinforcement learner with a supervised classification algorithm that seeks to split perceptual states so as to reduce perceptual aliasing. This results in an adaptive discretization of the perceptual space based on the presence or absence of visual features. Its extension RLJC also handles continuous action spaces. In contrast to the minimalistic visual representations produced by RLVC and RLJC, our second method learns structural object models for robust object detection and pose estimation by probabilistic inference. To these models, the method associates grasp experiences autonomously learned by trial and error. These experiences form a non-parametric representation of grasp success likelihoods over gripper poses, which we call a *grasp density*. Thus, object detection in a novel scene simultaneously produces suitable grasping options.

1 Introduction

Vision is a popular sensory modality for autonomous robots. Optical sensors are comparatively cheap and deliver more information at higher rates than other sensors. Moreover, vision appears intuitive as humans heavily rely on it. We appear to

J. Piater, S. Jodogne (now at Euresys s.a., Belgium) and R. Detry
INTELSIG Laboratory, Université de Liège, Belgium, e-mail: Justus.Piater@ULg.ac.be, Renaud.Detry@ULg.ac.be

D. Kraft and N. Krüger
University of Southern Denmark, e-mail: kraft@mip.sdu.dk, norbert@mip.sdu.dk

O. Kroemer and J. Peters
Max Planck Institute for Biological Cybernetics, Tübingen, Germany, e-mail: oliverkro@tuebingen.mpg.de, Jan.Peters@tuebingen.mpg.de

think and act on the basis of an internal model of our environment that is constantly updated via sensory – and mostly visual – perception. It is therefore a natural idea to attempt to build autonomous robots that derive actions by reasoning about an internal representation of the world, created by computer vision.

However, experience shows that it is very difficult to build generic world models by sensory perception. For example, detailed shape recovery from passive optical sensors is hard, and the appearance of a scene – that is, its raw image representation – varies about as greatly with imaging conditions as it does with semantic content.

Paraphrasing Vapnik's Main Principle of inference from a small sample size [23], it may thus not be a good idea to try to solve the hard intermediate problem of building a reliable world model in order to solve the easier problem of extracting just the information the robot needs to act appropriately. This leads to the idea of linking perception more-or-less directly to action, without the intermediate step of reasoning on a world model, whose roots go back at least to the learned value functions of Samuel's famous checker player [20].

In this chapter, we give an overview of two examples of our own research on learning visual representations for robotic action within specific task scenarios, without building generic world models. The first problem we consider is the direct linking of visual perception to action within a reinforcement-learning (RL) framework (Sect. 2). The principal difficulty is the extreme size of the visual perceptual space, which we address by learning a percept classifier interleaved with the reinforcement learner to adaptively discretize the perceptual space into a manageable number of discrete states. However, not all visuomotor tasks can be reduced to simple, reactive decisions based on discrete perceptual states. For example, to grasp objects, the object pose is of fundamental importance, and the grasp parameters depend on it in a continuous fashion. We address such tasks by learning intermediate object representations that form a direct link between perceptual and action parameters (Sect. 3). Again, these representations can be learned autonomously, permitting the robot to improve its grasping skills with experience. The resulting probabilistic models allow the inference of possible grasps and their relative success likelihoods from visual scenes.

2 Reinforcement Learning of Visual Classes

Reinforcement learning [2, 22] is a popular method for learning perception-action mappings, so-called *policies*, within the framework of Markov Decision Processes (MDP). Learning takes place by evaluating *actions* taken in specific *states* in terms of the *reward* received. This is conceptually easy for problems with discrete state and action spaces. Continuous and very large, discrete state spaces are typically addressed by using function approximators that permit local generalization across similar states. However, for reasons already noted above, function approximators alone are not adequate for the very high-dimensional state spaces spanned by images: Visually similar images may represent states that require distinct actions, and very dissimilar images may actually represent the exact same scene (and thus state) under different imaging conditions. Needless to say, performing RL directly on the combinatorial state space defined by the image pixel values is infeasible.

The challenge therefore lies in mapping the visual space to a state representation that is suitable for reinforcement learning. To enable learning with manageably low numbers of exploratory actions, this means that the state space should either consist of a relatively small number of discrete states, or should be relatively low-dimensional and structured in such a way that nearby points in state space mostly admit identical actions that yield similar rewards.

The latter approach would be very interesting to explore, but it appears that it would require strong, high-level knowledge about the content of the scene such as object localization and recognition, which defeats our purpose of learning perception-action mappings without solving this harder problem first. We therefore followed the first approach and introduced a method called Reinforcement Learning of Visual Classes (RLVC) that adaptively and incrementally discretizes a continuous or very large discrete perceptual space into discrete states [9, 12].

2.1 RLVC: a Birdseye View

RLVC decomposes the end-to-end problem of learning perception-to-action mappings into two simpler learning processes (Fig. 1). One of these, the *RL agent*, learns discrete state-action mappings in a classical RL manner. The state representation and the determination of the current state are provided by an *image classifier* that carves up the perceptual (image) space into discrete states called *visual classes*. These two processes are interleaved: Initially, the entire perceptual space is mapped to a single visual class. From the point of view of the RL agent, this means that a variety of distinct world states requiring different actions are lumped together – *aliased* – into a single perceptual state (visual class). Based on experience accumulated by the RL agent, the image classifier then identifies a visual feature whose presence or absence defines two distinct visual subclasses, splitting the original visual class into two. This procedure is iterated: At each iteration, one or more perceptually-aliased visual classes are identified, and for each, a feature is determined that splits it in a way that maximally reduces the perceptual aliasing in both of the resulting new visual classes (Fig. 2). Thus, in a sequence of attempts to reduce perceptual aliasing, RLVC builds a sequence $\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2, \ldots$ of increasingly refined, binary decision trees $\mathscr{C}_k$ with visual feature detectors at decision nodes. At any stage k, $\mathscr{C}_k$ partitions the visual space S into a finite number m_k of visual classes $\{V_{k,1}, \ldots, V_{k,m_k}\}$.

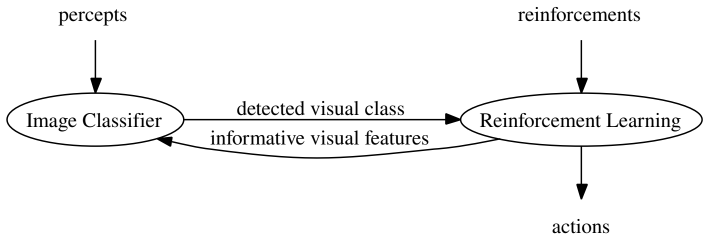


Fig. 1 RLVC: Learning a perception-action mapping decomposed into two interacting subproblems.

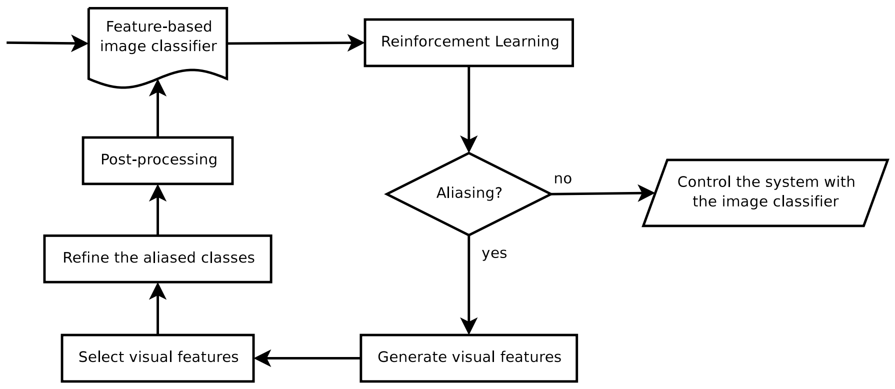

Feature-based image classifier
Reinforcement Learning
Post-processing
Aliasing?
no
Control the system with the image classifier
Refine the aliased classes
yes
Select visual features
Generate visual features

Fig. 2 Outline of the RLVC algorithm.

2.2 Reinforcement Learning and TD Errors

An MDP is a quadruple $\langle S, A, \mathscr{T}, \mathscr{R} \rangle$, where S is a finite set of states, A is a finite set of actions, $\mathscr{T}$ is a probabilistic transition function from $S \times A$ to S, and $\mathscr{R}$ is a scalar reward function defined on $S \times A$. From state s_t at time t, the agent takes an action a_t, receives a scalar reinforcement $r_{t+1} = \mathscr{R}(s_t, a_t)$, and transitions to state s_{t+1} with probability $\mathscr{T}(s_t, a_t, s_{t+1})$. The corresponding quadruple $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ is called an *interaction*. For infinite-horizon MDPs, the objective is to find an optimal *policy* $\pi^* : S \to A$ that chooses actions that maximize the expected *discounted return*

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \qquad (1)$$

for any starting state s_0, where $0 \leq \gamma < 1$ is a discount factor that specifies the immediate value of future reinforcements.

If $\mathscr{T}$ and $\mathscr{R}$ are known, the MDP can be solved by dynamic programming [1]. Reinforcement learning can be seen as a class of methods for solving unknown MDPs. One popular such method is Q-learning [24], named after its state-action value function

$$Q^{\pi}(s,a) = \mathrm{E}^{\pi}\left[R_t \mid s_t = s, a_t = a\right] \tag{2}$$

that returns the expected discounted return by starting from state s, taking action a, and following the policy π thereafter. An optimal solution to the MDP is then given by

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} Q^*(s,a). \tag{3}$$

In principle, a Q function can be learned by a sequence of α-weighted updates

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(\mathrm{E}[R_t \mid s = s_t, a = a_t] - Q(s_t,a_t)\right) \tag{4}$$

that visits all state-action pairs infinitely often. Of course, this is not a viable algorithm because the first term of the update step is unknown; it is precisely the return function (2) we want to learn. Now, rewards are accumulated (1) by executing actions, hopping from state to state. Thus, for an interaction $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$, an estimate of the current return $Q(s_t,a_t)$ is available as the discounted sum of the immediate reward r_{t+1} and the estimate of the remaining return $Q(s_{t+1},a_{t+1})$, where $s_{t+1} = \mathscr{T}(s_t,a_t)$ and $a_{t+1} = \pi(s_t)$. If the goal is to learn a value function Q^* for an optimal policy (3), then this leads to the algorithm

$$\begin{aligned} Q(s_t,a_t) &\leftarrow Q(s_t,a_t) + \alpha \varDelta_t & (5)\\ \varDelta_t &= r_{t+1} + \gamma \max_{a' \in A} Q(s_{t+1},a') - Q(s_t,a_t) & (6) \end{aligned}$$

that, under suitable conditions, converges to Q^*. $\varDelta_t$ is called the *temporal-difference error* or *TD error* for short.

2.3 Removing Perceptual Aliasing

RLVC is based on the insight that if the world behaves predictably, $r_{t+1} + \gamma \max_{a' \in A} Q(s_{t+1},a')$ approaches $Q(s_t,a_t)$, leading to vanishing TD errors (6). If however the magnitudes of the TD errors of a given state-action pair (s,a) remain large, this state-action pair yields unpredictable returns. RLVC assumes that this is due to perceptual aliasing, that is, the visual class s represents distinct world states that require different actions. Thus, it seeks to split this state in a way that minimizes the sum of the variances of the TD errors in each of the two new states. This is an adaptation of the splitting rule used by CART for building regression trees [3].

To this end, RLVC selects from all interactions collected from experience those whose visual class and action match s and a, respectively, along with the resulting TD error $\varDelta$, as well as the set $F_{\oplus} \in F$ of features present in the raw image from which the visual class was computed. It then selects the feature

$$f^* = \operatorname*{argmin}_{f \in F} \left\{ p_f \sigma^2\{\varDelta_f\} + p_{\neg f} \sigma^2\{\varDelta_{\neg f}\} \right\} \tag{7}$$

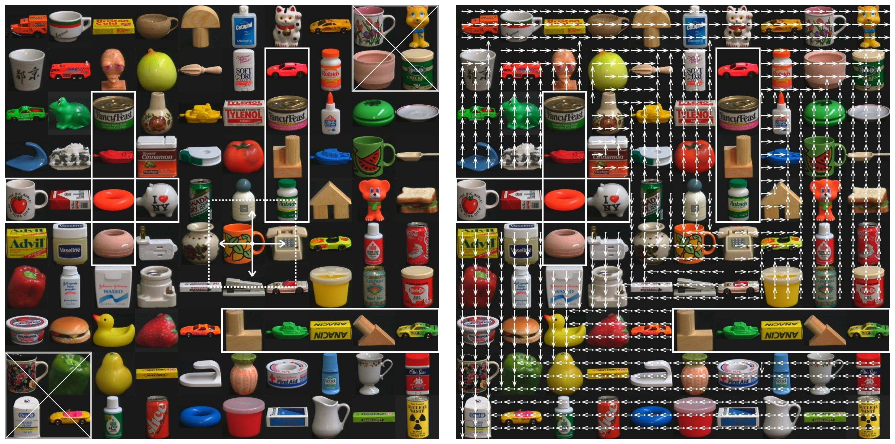


Fig. 3 A continuous, noisy navigation task. The exits of the maze are marked by crossed boxes. Transparent obstacles are identified by solid rectangles. The agent is depicted near the center of the left-hand figure. Each of the four possible moves is represented by an arrow, the length of which corresponds to the resulting move. The sensor returns a picture that corresponds to the dashed portion of the image. The right-hand figure shows an optimal policy learned by RLVC, sampled at regularly-spaced points.

that results in the purest split in terms of the TD errors. Here, p_f is the proportion of the selected interactions whose images exhibit feature f, and $\{\Delta_f\}$ is the associated set of TD errors; $\neg f$ indicates the corresponding entities that do not exhibit feature f.

Splitting a visual class s according to the presence of a feature f^* results in two new visual classes, at least one of which will generally exhibit lower TD errors than the original s. However, there is the possibility that such a split turns out to be useless because the observed lack of convergence was due to the stochastic nature of the environment rather than perceptual aliasing. RLVC partially addresses this by splitting a state only if the resulting distributions of TD errors are significantly different according to a Student's t test.

2.4 Experiments

We evaluated our system on an abstract task that closely parallels a real-world, reactive navigation scenario (Fig. 3). The goal of the agent is to reach one of the two exits of the maze as fast as possible. 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 Gaussian noise, the standard deviation of which is 2% of the size of the maze. Invisible obstacles are present in the maze. Whenever a move would take the agent into an obstacle or outside the maze, its location is not changed.

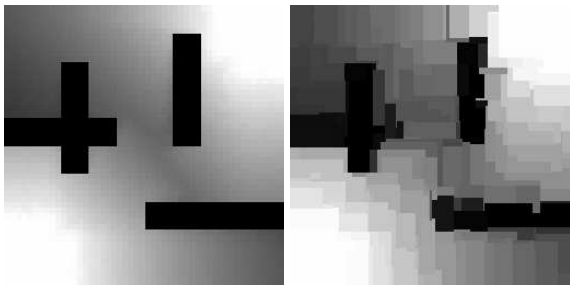


Fig. 4 Left: The optimal value function, when the agent has direct access to its (x, y) position in the maze and when the set of possible locations is discretized into a 50×50 grid. The brighter the location, the greater its value. Right: The final value function obtained by RLVC.

The agent earns a reward of 100 when an exit is reached. Any other move generates zero reinforcement. When the agent succeeds at escaping the maze, it arrives in a terminal state in which every move gives rise to a zero reinforcement. The discount factor 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 when choosing the most attractive exit.

The raw perceptual input of the agent is a square window centered at its current location, showing a subset of a tiled montage of the COIL-100 images [17]. There is no way for the agent to directly locate the obstacles; it is obliged to identify them implicitly as regions of the maze where certain actions do not change its location.

In this experiment, we used color differential invariants as visual features [8]. The entire tapestry includes 2298 different visual features, of which RLVC selected 200 (9%). The computation stopped after the generation of $k = 84$ image classifiers, which took 35 minutes on a 2.4 GHz 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 discretization of the maze when the agent knows its (x, y) position. For example, a reasonably-sized 20×20 grid leads to 400 perceptual classes. A direct, tabular representation of the Q function in terms of all Boolean feature combinations would have $2^{2298} \times 4$ cells. Figure 4 compares the optimal value function of a regularly-discretized problem with the one obtained through RLVC.

In a second experiment we investigated RLVC on real-word images under identical navigation rules (Fig. 5). RLVC took 101 iterations in 159 minutes to converge using databases of 10,000 interactions. 144 distinct visual features were selected among a set of 3739 possibilities, 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.

2.5 Further Developments

The basic method described in the preceding sections admits various powerful extensions, some of which are described in the following.

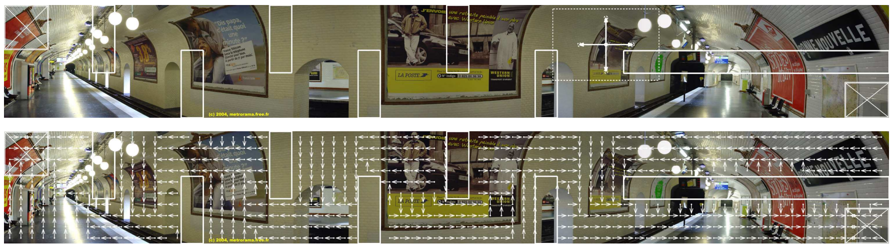


Fig. 5 Top: a navigation task with a real-world image, using the same conventions as Figure 3. Bottom: the deterministic image-to-action mapping computed by RLVC.

2.5.1 On-the-fly Creation of Visual Feature

As described above, the power of RLVC to resolve action-relevant perceptual ambiguities is limited by the availability of precomputed visual features and their discriminative power. This can be overcome by creating new features on the fly as needed [13]. When the state-refinement procedure fails to identify a feature that results in a significant reduction of the TD errors, new features are created by forming spatial compounds of existing features. In this way, a compositional hierarchy of features is created in a task-driven way. Compounds are always at least as selective as their individual constituents. To favor features that generalize well, features are combined that are frequently observed to co-occur in stable spatial configurations.

We demonstrated the power of this idea on a variant of the popular mountain-car control problem, a physical simulation where an underpowered car needs to accumulate energy to climb out of a parabolic valley by alternatingly accelerating forwards and backwards. Contrary to its classical versions [16, 7], our agent has no direct access to its position and velocity. Instead, the road is carpeted with a montage of images, and the velocity is communicated via a picture of an analog gauge (Fig. 6). In particular, the velocity is impossible to measure using local features only. However, RLVC succeeded at learning compound features useful for solving the task. The performance of the policy learned by RLVC was about equivalent to classical Q-learning on a directly-observable position-velocity state space discretized into an equivalent number of states. Evidently, the disadvantage of having to learn relevant state parameters indirectly through visual observations was compensated by the adaptive, task-driven discretization of the state space (Fig. 7).

2.5.2 Merging Visual Classes

In its original form, RLVC is susceptible to overfitting because states are only ever split and never merged. It is therefore desirable to identify and merge equivalent states. Various ways of defining this equivalence are possible. Here, we say that two states are equivalent if (a) their optimal values $\max_a Q(s,a)$ are similar, and (b) their

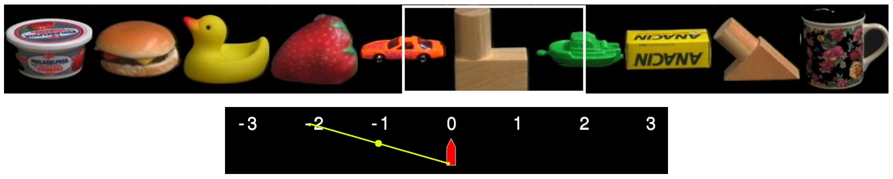
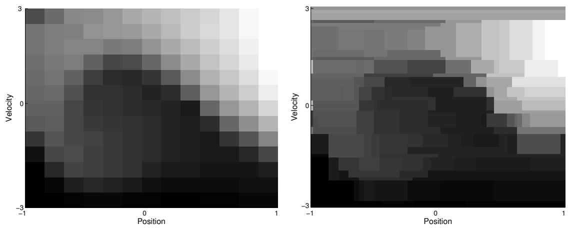


Fig. 6 Top: road painting. The region framed with a white rectangle corresponds to an instantaneous percept; it slides back and forth as the agent moves. Bottom: velocity gauge as seen by the agent, and a visual compound feature learned by RLVC that triggers at near-zero velocities.

Fig. 7 Left: The optimal value function, when the agent has direct access to its current (p,s) state discretized into a 13×13 grid. The brighter the location, the greater its value. Right: The value function obtained by RLVC.



optimal policies are equivalent, that is, the value of one state's optimal action $\pi^*(s)$ is similar if taken in the other state.

A second drawback of basic RLVC is that decision trees do not make optimal use of the available features, since they can only represent conjunctions of features.

We modified RLVC to use a Binary Decision Diagram (BDD) [4] instead of a decision tree to represent the state space [10]. To split a state, a new feature is conjunctively added to the BDD as before to the decision tree. Periodically, after running for some number of stages, *compaction* takes place: equivalent states are merged and a new BDD is formed that can represent both conjunctions and disjunctions of features. In the process, feature tests are reordered as appropriate, which may lead to the elimination of some features.

The benefits are demonstrated by a reactive outdoor navigation task using photographs as perceptual input, and a discrete action space consisting of quarter turns and a forward move (Fig. 8). Compared to the original RLVC, the numbers of visual classes and features was greatly reduced (Fig. 9), while achieving a slight improvement of generalization to unseen percepts.

2.5.3 Continuous Actions: Reinforcement Learning of Joint Classes

RLVC addresses high-dimensional and continuous perceptual spaces, but the action space is practically limited to a small number of discrete choices. Reinforcement Learning of Joint Classes (RLJC) applies the principles of RLVC to an adaptive discretization of the joint space of perceptions and actions (Fig. 10) [11]. The Q function now operates on the joint-class domain encompassing both perceptual and action dimensions. Joint classes are split based on *joint features* that test the pres-

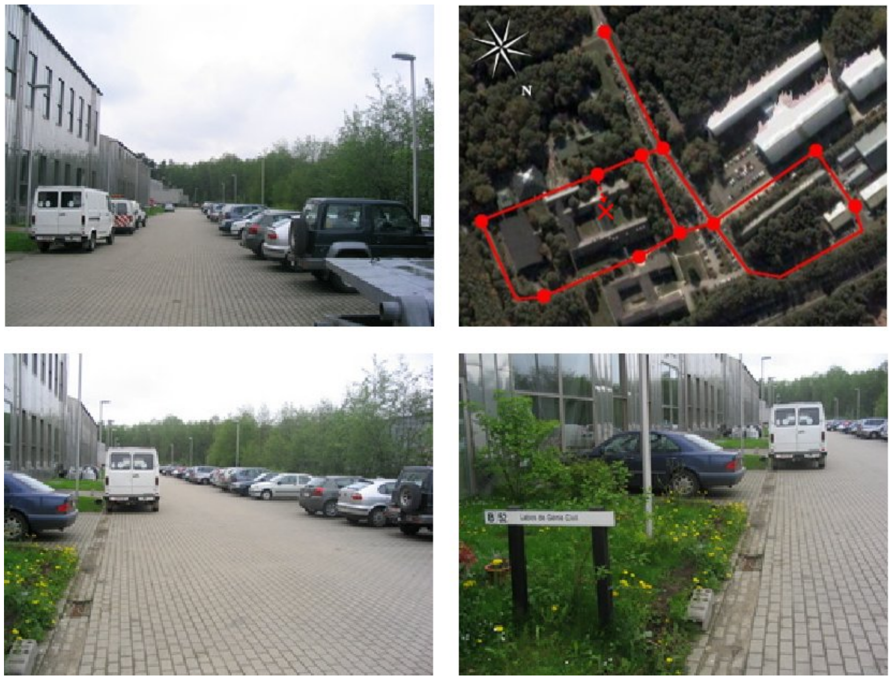


Fig. 8 Navigation around the ULg School of Engineering; three example percepts corresponding to the same world state.

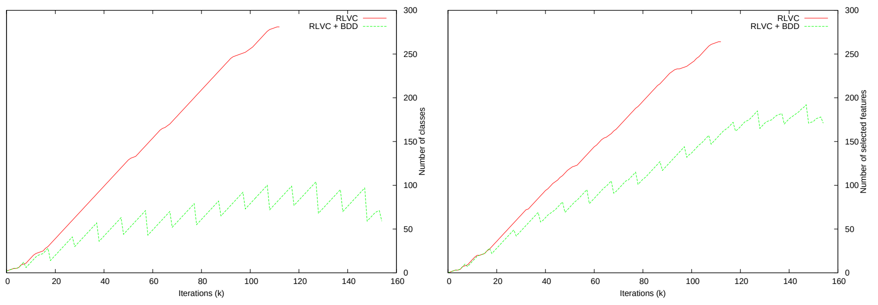


Fig. 9 Comparison of the number of generated classes and selected visual features between the basic and BDD versions of RLVC. The number of visual classes (resp. selected features) as a function of the stage counter k is shown on the left (resp. right). The sawtooth patterns are due to the periodicity of the compaction phases.

ence of either a visual feature or an *action feature*. An action feature (t,i) tests whether the ith component of an action $a \in \mathbb{R}^m$ falls below a threshold t. This relatively straightforward generalization of RLVC results in an innovative addition to the rather sparse toolbox of RL methods for continuous action spaces.

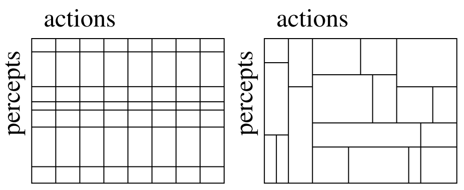


Fig. 10 Intuition of the adaptive discretization performed by RLVC (left) and RLJC (right).

We evaluated RLJC on the same task as shown in Fig. 3 but allowing the agent to choose a real-valued direction to go. In this case, the number of resulting joint classes roughly corresponds to the number of perceptual classes generated by RLVC on the discrete version, multiplied by the number of discrete actions.

3 Grasp Densities

RLVC, described in the preceding section, follows a minimalist approach to perception-action learning in that it seeks to identify small sets of low-level visual cues and to associate reactive actions to them directly. There is no elaborate image analysis beyond feature extraction, no intermediate representation, no reasoning or planning, and the complexity of the action space that can be handled by RL is limited. In this section we describe a different approach to perception-action learning that is in many ways complementary to RLVC. Its visual front-end builds elaborate representations from which powerful, structured object representations are learned, and multidimensional action vectors are derived via probabilistic inference.

We describe this method in the context of grasping objects [5], a fundamental skill of autonomous agents. The conventional robotic approach to grasping involves computing grasp parameters based on detailed geometric and physical models of the object and the manipulator. However, humans skillfully manipulate everyday objects even though they do not have access to such detailed information. It is thus clear that there must exist alternative methods. We postulate that manipulation skills emerge with experience by associating action parameters to perceptual cues. Then, perceptual cues can directly trigger appropriate actions, without explicit object shape analysis or grasp planning. For example, to drink, we do not have to reason about the shape and size of the handle of a hot teacup to determine where to place the fingers to pick it up. Rather, having successfully picked up and drunk from teacups before, seeing the characteristic handle immediately triggers the associated, canonical grasp.

Our method achieves such behavior by first learning visual object models that allow the agent to detect object instances in scenes and to determine their pose. Then, the agent explores various grasps, and associates the successful parameters that emerge from this grasping experience with the model. When the object is later detected in a scene, the detection and pose estimation procedure immediately produces the associated grasp parameters as well. This system can be bootstrapped in

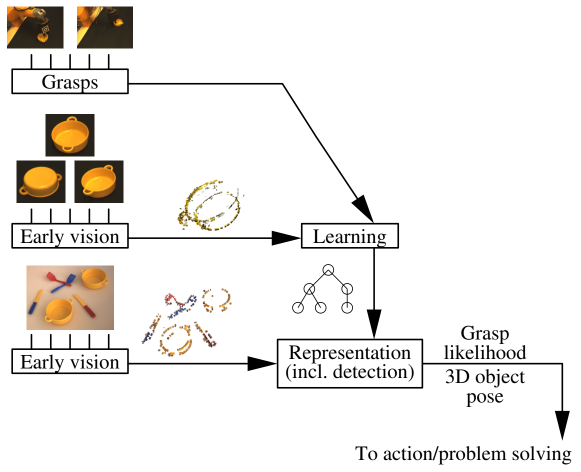


Fig. 11 Overview of learning to grasp. Learning produces graph-structured object representations that combine experienced grasps with visual features provided by early vision. Subsequently, instances can be detected in new scenes provided by early vision. Detection directly produces pose estimates and suitable grasp parameters.

the sense that very little grasping experience is already useful, and the representation can be refined by further experience at any time. It thus constitutes a mechanism for learning grasping skills from experience with familiar objects.

3.1 Learning to Grasp: a Birdseye View

Figure 11 presents an overview of our grasp learning system. Visual input is provided by an early-vision front-end that produces 3D oriented patches with appearance information. On such sparse 3D reconstructions, the object-learning component analyzes spatial feature relations. Pairs of features are combined by new parent features, producing a hierarchically-structured Markov network that represents the object via sparse appearance and structure. In this network, a link from a parent node to a child node represents the distribution of spatial relations (relative pose) between the parent node and instances of the child node. Leaf nodes encode appearance information.

Given a reconstruction of a new scene provided by early vision, instances of such an object model can be detected via probabilistic inference, estimating their pose in the process.

Having detected a known object for which it has no grasping experience yet, the robot attempts to grasp the object in various ways. For each grasp attempt, it stores the object-relative pose of the gripper and a measure of success. Object-relative gripper poses are represented in exactly the same way as parent-relative child poses of visual features. In this manner, grasping experience is added to the object representation as a new child node of a high-level object node. From then on, inference of object pose at the same time produces a distribution of gripper poses suitable for grasping the object.

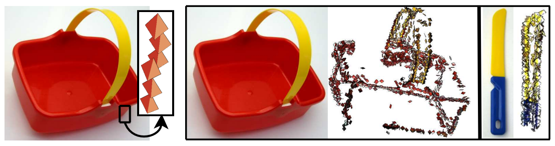


Fig. 12 ECV descriptors. Left: ECV descriptors are oriented appearance patches extracted along contours. Right: object-segmented and refined ECV descriptors via structure from motion.

3.2 Early Vision

Visual primitives and their location and orientation in space are provided by the Early-Cognitive-Vision (ECV) system by Krüger et al. [15, 19]. It extracts patches – so-called ECV descriptors – along image contours and determines their 3D position and a 2D orientation by stereo techniques (the orientation around the 3D contour axis is difficult to define and is left undetermined). From a calibrated stereo pair, it generates a set of ECV descriptors that represent the scene, as sketched at the bottom of Fig. 11.

Objects can be isolated from such scene representations by motion segmentation. To this end, the robot uses bottom-up heuristics to attempt to grasp various surfaces suggested by combinations of ECV descriptors. Once a grasp succeeds and the robot gains physical control over the grasped structure, the robot can pick it up and turn it in front of the stereo camera. This allows it to segment object descriptors from the rest of the scene via coherent motion, and to complete and refine the object descriptors by structure-from-motion techniques, as illustrated in Fig. 12 [14].

3.3 Markov Networks For Object Representation

Our object model consists of a set of generic *features* organized in a hierarchy. Features that form the bottom level of the hierarchy, referred to as *primitive features*, are bound to visual observations. The rest of the features are *meta-features* that embody relative spatial configurations of more elementary features, either meta or primitive. At the bottom of the hierarchy, primitive features correspond to local parts that each may have many *instances* in the object. Climbing up the hierarchy, meta-features correspond to increasingly complex parts defined in terms of constellations of lower-level parts. Eventually, parts become complex enough to satisfactorily represent the whole object. Here, a primitive feature represents a class of ECV observations of similar appearance, e.g. an ECV observation with colors close to red and white. Any primitive feature will usually have hundreds of instances in a scene.

Figure 13 shows an example of a hierarchy for a traffic sign. Ignoring the nodes labeled Y_i for now, the figure shows the traffic sign as the combination of two features, a bridge pattern (feature 4) and a triangular frame (feature 5). The fact that the bridge pattern has to be in the center of the triangle to form the traffic sign is encoded in the links between features 4-6-5. The triangular frame is encoded in terms

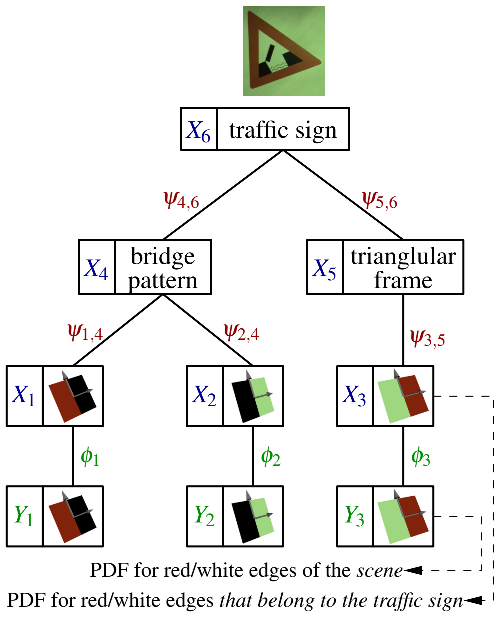


Fig. 13 An example of a hierarchy for a traffic sign. X_1 through X_3 are primitive features; each of these is linked to an observed variable Y_i. X_4 through X_6 are meta-features.

of a single (generic) feature, a short red-white edge segment (feature 3). The link between feature 3 and feature 5 encodes the fact that many short red-white edge segments are necessary to form the triangular frame, and the fact that these edges have to be arranged along a triangle-shaped structure.

Here, a “feature” is an abstract concept that may have any number of instances. The lower-level the feature, the larger generally the number of instances. Conversely, the higher-level the feature, the richer its relational and appearance description.

The feature hierarchy is implemented as a Markov tree (Fig. 13). Features correspond to hidden nodes of the network. When a model is associated to a scene (during learning or instantiation), the pose of feature i in that scene will be represented by the probability density function of a random variable X_i, effectively linking feature i to its instances. Random variables are thus defined over the pose space, which corresponds to the Special Euclidean group $SE(3) = \mathbb{R}^3 \times SO(3)$.

The relationship between a meta-feature i and one of its children j is parametrized by a *compatibility potential* $\psi_{ij}(X_i, X_j)$ that reflects, for any given relative configuration of feature i and feature j, the likelihood of finding these two features in that relative configuration. The (symmetric) potential between i and j is denoted by $\psi_{ij}(X_i, X_j)$. A compatibility potential is equivalent to the spatial distribution of the child feature in a reference frame that matches the pose of the parent feature; a potential can be represented by a probability density over $SE(3)$.

Each primitive feature is linked to an observed variable Y_i. Observed variables are tagged with an appearance descriptor that defines a class of observation appearance. The statistical dependency between a hidden variable X_i and its observed variable Y_i

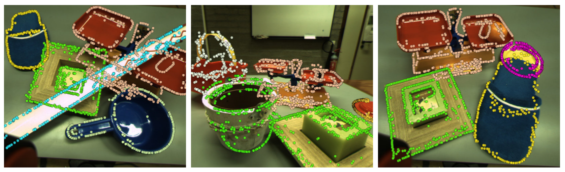


Fig. 14 Cluttered scenes with pose estimates. Local features of object models are back-projected into the image at the estimated pose; false colors identify different objects.

is parametrized by an *observation potential* $\psi_i(X_i, Y_i)$. We generally cannot observe meta-features; their observation potentials are thus uniform.

Instantiation of such a model in a scene amounts to the computation of the marginal posterior pose densities $p(X_i|Y_1, \ldots, Y_n)$ for all features X_i, given all available evidence Y. This can be done using any applicable inference mechanism. We use nonparametric belief propagation [21] optimized to exploit the specific structure of this inference problem [6]. The particles used to represent the densities are directly derived from individual feature observations. Thus, object detection (including pose inference) amounts to image observations probabilistically voting for object poses compatible with their own pose. The system never commits to specific feature correspondences, and is thus robust to substantial clutter and occlusions. During inference, a consensus emerges among the available evidence, leading to one or more consistent scene interpretations. After inference, the pose likelihood of the whole object can be read out of the top-level feature. If the scene contains multiple instances of the object, this feature density will present multiple major modes. Figure 14 shows examples of pose estimation results.

3.4 Grasp Densities

Having described the visual object representation and pose inference mechanism, we now turn to our objective of learning grasp parameters and associating them to the visual object models. We consider parallel-gripper grasps parametrized by a 6D gripper pose composed of a 3D position and a 3D orientation. The set of object-relative gripper poses that yield stable grasps is the *grasp affordance* of the object.

A grasp affordance is represented as a probability density function defined on $SE(3)$ in an object-relative reference frame. We store an expression of the joint distribution $p(X_o, X_g)$, where X_o is the pose distribution of the object, and X_g is the grasp affordance. This is done by adding a new "grasp" feature to the Markov network, and linking it to the top feature (see Fig. 15). The statistical dependency of X_o and X_g is held in a compatibility potential $\psi_{og}(X_o, X_g)$.

When an object model has been aligned to an object instance (i.e. when the marginal posterior of the top feature has been computed from visually-grounded bottom-up inference), the grasp affordance $p(X_g \mid Y)$ of the object *instance*, given all observed evidence Y, is computed through top-down belief propagation, by sending a message from X_o to X_g through $\psi_{og}(X_o, X_g)$:

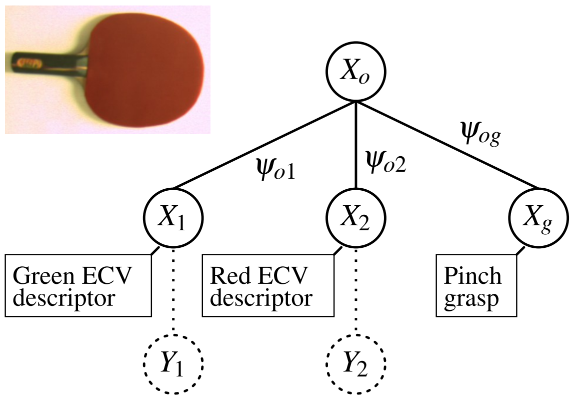


Fig. 15 Visual/affordance model of a table-tennis bat as a 2-level hierarchy. The bat is represented by feature o (top). Feature 1 represents a generic green ECV descriptor. The rectangular configuration of green edges around the handle of the paddle is encoded in ψ_{o1}. Y_1 and Y_2 are observed variables that link features 1 and 2 to the visual evidence produced by ECV. X_g represents a grasp feature, linked to the object feature through the pinch grasp affordance ψ_{og}.

$$p(X_g \mid Y) = \int \psi_{og}(X_o, X_g) p(X_o \mid Y) \, \mathrm{d}X_o \tag{8}$$

This continuous, probabilistic representation of a grasp affordance in the world frame we call a *grasp density*. In the absence of any other information such as priors over poses or kinematic limitations, it represents the relative likelihood that a given gripper pose will result in a successful grasp.

3.5 Learning Grasp Densities

Like the pose densities discussed in Sect. 3.3, grasp densities are represented nonparametrically in terms of individual observations (Fig. 16). In this case, each successful grasp experience contributes one particle to the nonparametric representation. An unbiased characterization of an object's grasp affordance conceptually involves drawing grasp parameters from a uniform density, executing the associated grasps, and recording the successful grasp parameters.

In reality, executing grasps drawn from a 6D uniform density is not practical, as the chances of stumbling upon successful grasps would be unacceptably low. Instead, we draw grasps from a highly biased *grasp hypothesis density* and use importance sampling techniques to properly weight the grasp outcomes. The result we call a *grasp empirical density*, a term that also communicates the fact that the density is generally only an approximation to the true grasp affordance: The number of particles derived from actual grasp experiences is severely limited – to a few hundred at best – by the fact that each particle is derived from a grasp executed by a real robot. This is not generally sufficient for importance sampling to undo the substantial bias exerted by the available hypothesis densities, which may even be zero in regions that afford actual grasps.

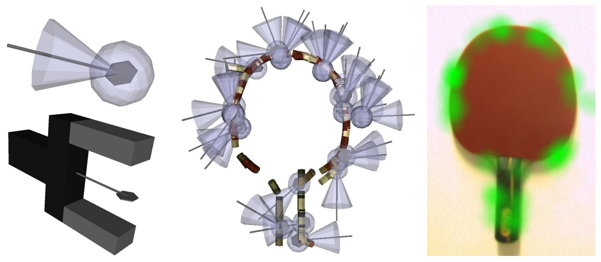

(a) (b) (c)

Fig. 16 Grasp density representation. The top image of Fig. (a) illustrates a particle from a non-parametric grasp density and its associated kernel widths: the translucent sphere shows one position standard deviation, the cone shows the variance in orientation. The bottom image illustrates how the schematic rendering used in the top image relates to a physical gripper. Figure (b) shows a 3D rendering of the kernels supporting a grasp density for a table-tennis paddle (for clarity, only 30 kernels are rendered). Figure (c) indicates with a green mask of varying opacity the values of the location component of the same grasp density along the plane of the paddle.

Grasp hypothesis densities can be derived from various sources. We have experimented with *feature-induced* grasp hypotheses derived from ECV observations. For example, a pair of similarly-colored, coplanar patches suggests the presence of a planar surface between them. In turn, this plane suggests various possible grasps [14]. Depending on the level of sophistication of such heuristics, feature-induced grasp hypotheses can yield success rates of up to 30% [18]. This is more than sufficient for effective learning of grasp hypothesis densities.

Another source of grasp hypothesis densities is human demonstration. In a pilot study, we tracked human grasps with a Vicon^{TM} motion capture system [5]. This can be understood as a way of teaching by demonstration: The robot is shown how to grasp an object, and this information is used as a starting point for autonomous exploration.

Illustrations of some experimental results are shown in Figs. 17 and 18. A large-scale series of experiments for quantitative evaluation is currently underway.

4 Discussion

We described two complementary methods for associating actions to perceptions via autonomous, exploratory learning. RLVC is a reinforcement-learning method that operates on a perceptual space defined by low-level visual features. Remarkably, its adaptive, task-driven discretization of the perceptual space allows it to learn policies with a number of interactions similar to problems with much smaller perceptual spaces. This number is nevertheless still greater than what a physical robot can realistically perform. In many practical applications, interactions will have to be generated in simulation. Among the extensions to RLVC, one particularly inter-

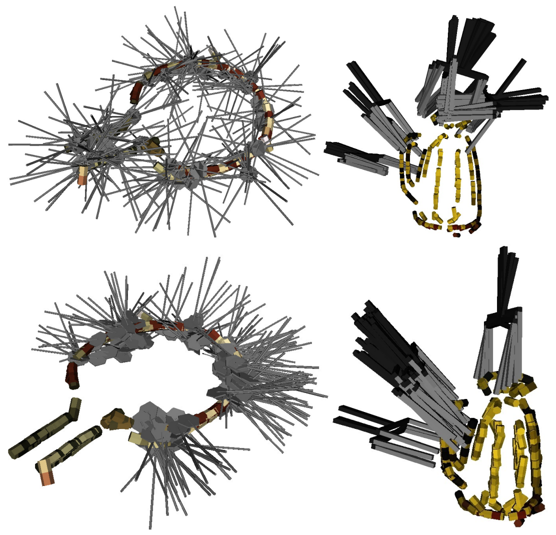


Fig. 17 Particles supporting grasp hypothesis (top) and empirical (bottom) densities. Hypothesis densities were derived from constellations of ECV observations (left) or from human demonstrations (right).

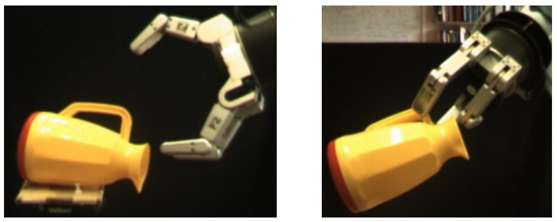


Fig. 18 Barrett hand grasping the toy jug.

esting avenue for further research is RLJC with its uniform treatment of continuous perceptual and action spaces.

RLVC does not learn anything about the world besides task-relevant state distinctions and the values of actions taken in these states. In contrast, the grasp-densities framework involves learning object models that allow the explicit computation of object pose. Object pose is precisely the determining factor for grasping; it is therefore natural to associate grasp parameters to these models. Beyond this association, however, no attempt is made to learn any specifics about the objects or about the world. For example, to grasp an object using grasp densities, the visibility of the contact surfaces in the scene is irrelevant, as the grasp parameters are associated to the object as a whole.

Notably, both learning systems operate without supervision. RL methods require no external feedback besides a scalar reward function. Learning proceeds by trial and error, which can be guided by suitably biasing the exploration strategy. Learning grasp-densities involves learning object models and trying out grasps. Again, the autonomous exploration can be – and normally will need to be – biased via the specification of a suitable grasp hypothesis density, by human demonstration or other means. The Cognitive Vision Group at the University of Southern Denmark, headed by N. Krüger, has put in place a robotic environment that is capable of learning grasp densities with a very high degree of autonomy, requiring human intervention only in exceptional situations. Like a human infant, the robot reaches for scene features and “plays” with objects by attempting to grasp them in various ways and moving them around.

Learning object-relative gripper poses is only the opening chapter of the grasp-density story. The principle can be extended to learning multiple grasp types such as palmar grasps and various multi-fingered pinches, associating hand pre-shapes and approach directions by learning parameters for motor programs, etc. These and other avenues will be pursued in future work.

Acknowledgements This work was supported by the Belgian National Fund for Scientific Research (FNRS) and the EU Cognitive Systems project PACO-PLUS (IST-FP6-IP-027657). We thank Volker Krüger and Dennis Herzog for their support during the recording of the human demonstration data.

References

1. Bellman, R.: Dynamic programming. Princeton University Press (1957)
2. Bertsekas, D., Tsitsiklis, J.: Neuro-Dynamic Programming. Athena Scientific (1996)
3. Breiman, L., Friedman, J., Stone, C.: Classification and Regression Trees. Wadsworth International Group (1984)
4. Bryant, R.: Symbolic Boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys **24**(3), 293–318 (1992)
5. Detry, R., Başeski, E., Popović, M., Touati, Y., Krüger, N., Kroemer, O., Peters, J., Piater, J.: Learning Object-specific Grasp Affordance Densities. In: International Conference on Development and Learning (2009)
6. Detry, R., Pugeault, N., Piater, J.: A Probabilistic Framework for 3D Visual Object Representation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2009). To appear
7. Ernst, D., Geurts, P., Wehenkel, L.: Iteratively extending time horizon reinforcement learning. In: 14th European Conference on Machine Learning, pp. 96–107 (2003). Dubrovnik, Croatia
8. Gouet, V., Boujemaa, N.: Object-based queries using color points of interest. In: IEEE Workshop on Content-Based Access of Image and Video Libraries, pp. 30–36 (2001). Kauai, HI, USA
9. Jodogne, S., Piater, J.: Interactive Learning of Mappings from Visual Percepts to Actions. In: 22nd International Conference on Machine Learning, pp. 393–400 (2005)
10. Jodogne, S., Piater, J.: Learning, then Compacting Visual Policies. In: 7th European Workshop on Reinforcement Learning, pp. 8–10 (2005). Naples, Italy
11. Jodogne, S., Piater, J.: Task-Driven Discretization of the Joint Space of Visual Percepts and Continuous Actions. In: European Conference on Machine Learning, *LNCS*, vol. 4212, pp. 222–233. Springer (2006)
12. Jodogne, S., Piater, J.: Closed-Loop Learning of Visual Control Policies. Journal of Artificial Intelligence Research **28**, 349–391 (2007)
13. Jodogne, S., Scalzo, F., Piater, J.: Task-Driven Learning of Spatial Combinations of Visual Features. In: Proc. of the IEEE Workshop on Learning in Computer Vision and Pattern Recognition (2005). Workshop at CVPR, San Diego, CA, USA
14. Kraft, D., Pugeault, N., Başeski, E., Popović, M., Kragić, D., Kalkan, S., Wörgötter, F., Krüger, N.: Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes. International Journal of Humanoid Robotics **5**, 247–265 (2008)

15. Krüger, N., Lappe, M., Wörgötter, F.: Biologically Motivated Multimodal Processing of Visual Primitives. Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour **1**(5), 417–428 (2004)
16. Moore, A., Atkeson, C.: The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. Machine Learning **21**(3), 199–233 (1995)
17. Nene, S., Nayar, S., Murase, H.: Columbia Object Image Library (COIL-100). Tech. Rep. CUCS-006-96, Columbia University, New York (1996)
18. Popović, M., Kraft, D., Bodenhagen, L., Başeski, E., Pugeault, N., Kragić, D., Krüger, N.: An Adaptive Strategy for Grasping Unknown Objects Based on Co-planarity and Colour Information. Submitted
19. Pugeault, N.: Early Cognitive Vision: Feedback Mechanisms for the Disambiguation of Early Visual Representation. Vdm Verlag Dr. Müller (2008)
20. Samuel, A.: Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development **3**(3), 210–229 (1959)
21. Sudderth, E., Ihler, A., Freeman, W., Willsky, A.: Nonparametric Belief Propagation. In: Computer Vision and Pattern Recognition, vol. I, pp. 605–612 (2003)
22. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press (1998)
23. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
24. Watkins, C.: Learning From Delayed Rewards. Ph.D. thesis, King's College, Cambridge, UK (1989)