Relational Reinforcement Learning Applied to Appearance-Based Object Recognition

Klaus Häming and Gabriele Peters

University of Applied Sciences and Arts,
Computer Science, Visual Computing,
Emil-Figge-Str. 42, D-44221 Dortmund, Germany
gabriele.peters@fh-dortmund.de
http://www.inf.fh-dortmund.de/personen/professoren/peters/

Abstract. In this paper we propose an adaptive, self-learning system, which utilizes relational reinforcement learning (RRL), and apply it to a computer vision problem. A common problem in computer vision consists in the discrimination between similar objects which differ in salient features visible from distinct views only. Usually existing object recognition systems have to scan an object from a large number of views for a reliable discrimination. Optimization is achieved at most with heuristics to reduce the amount of computing time or to save storage space. We apply RRL in an appearance-based approach to the problem of discriminating similar objects, which are presented from arbitray views. We are able to rapidly learn scan paths for the objects and to reliably distinguish them from only a few recorded views. The appearance-based approach and the possibility to define states and actions of the RRL system with logical descriptions allow for a large reduction of the dimensionality of the state space and thus save storage and computing time.

Keywords: Relational reinforcement learning, computer vision, appearance-based object recognition, object discrimination.

1 Introduction

Relational reinforcement learning (RRL) is an attempt to extend the applicability of reinforcement learning (RL) by combining it with the relational description of states and actions. Disadvantages of RL are firstly its inability of handling large state spaces unless a regression technique is applied to approximate the Q-function. Secondly, the learned Q-function is neither easily interpretable nor easily extendable by a human, making it difficult to introduce a priori knowledge to support the learning process. And last, a generalization from problems the system was trained on to different but similar problems can hardly be done using classical RL. In [1], the relational representation of the Q-function was proposed and studied to overcome these issues. In this paper, we examine the applicability of the relational approach to a computer vision problem. The problem we consider is common in many computer vision applications (e.g., in manufacturing) and consists in the discrimination between similar objects which differ

D. Palmer-Brown et al. (Eds.): EANN 2009, CCIS 43, pp. 301–312, 2009.

slightly, e.g., in salient features visible from distinct views only. In our system a camera is rotated around an object (i.e., it is moved on an imaginary sphere around the object) until the system is able to reliably distinguish two similar objects. We want an agent to learn autonomously how to scan an object in the most efficient way, i.e., to find the shortest scan path which is sufficient to decide which object is presented. We want our approach to be general enough to extend from our simulated problem to a real world problem. Therefore we avoid using information that is uncertain or completely lacking in a real world application. This includes the camera parameters. Thus, in the state representation neither the camera's position nor its viewing direction are encoded, rather the design is purely appearance-based. The agent receives no information on the current 3D position of the camera. The only utilizable information is the appearance of the current view of the object in form of features visible in the view at hand. This will lead to learned rules such as: "seeing those features, an advantageous next action is to move the camera in that direction". This direction is encoded relatively to the current view of the agent.

2 Reinforcement Learning

RL [2] is a computational technique that allows an autonomous agent to learn a behavior via trial and error. A RL-problem is modeled by the following components: a set of states S, a set of actions A, a transition function $\delta : S \times A \to S$, and a reward function $r : S \times A \to \mathbb{R}$. The reward function is unknown to the agent, whose goal is to maximze the cumulated reward. In Q-learning, the agent attaches a Q(uality)-value to encountered state-action-pairs. After each transition, this Q-value of the state-action-pair is updated according to the update equation:

$$Q^{t+1}(s,a) = r(s,a) + \gamma \max_{a'} Q^t(s',a').$$

In this equation, $s' = \delta(s,a)$, while γ is a discount factor ensuring that the Q-values always stay finite even if the number of states is unbounded. As explained later, in our application it will also be helpful in search of a short scan path. During exploration, in each state the next action has to be chosen. This is done by a policy π. In Q-learning, its decision is based on the Q-values. We use the ϵ-greedy policy, which chooses with equal probability one of those actions that share the highest Q-value, except for a fraction of choices controlled by the parameter ϵ, in which the next action is drawn randomly from all actions.

3 Relational Reinforcement Learning

RLL uses propositional logic to encode states. Since this leads to a sparse representation of the Q-function, the number of state attributes for a given problem is smaller in the RRL approach than in the RL approach. Another advantage of RRL consists in the fact that slight changes in the state or action spaces do not necessarily lead to the mandatory relearning-from-scratch give in RL. In our

application of learning short, discriminative scan paths we build a sparse representation of the Q-function by using a selection of encoutered sample views of the objects and apply a regression technique. There are different approaches to this, such as decision trees [1], kernel-based approaches [3], or k-nearest-neighbour approaches [4]. In this paper we adopt the k-nearest-neighbour approach of [4], which is now briefly reviewed. The basic idea is to approximate the Q-value q_i of state-action-pair i by comparing the logical description of that pair to the logical descriptions of the examples of state-action-pairs found so far. The n closest of these examples are used to derive the desired approximation $\hat{q}_i$. The formula is given by

$$\hat{q}_i = (\sum_{j=1}^{n} \frac{1}{dist_{i,j}} q_j)/(\sum_{j=1}^{n} \frac{1}{dist_{i,j}})$$

which is simply the weighted arithmetical mean of the nearest Q-values. The weight is given by a reciprocal distance measure $dist_{i,j}$, which depends on the logical modeling of the state-action-pairs and is defined in Sec. 4. We use a value of $n = 30$. The strategy to decide whether an example gets included into the Q-function is basically to test if this example contributes a significant amount of new information, the Q-function needs denser sampling, or none of these. The measure of information novelty is based on the local standard deviation σ_l of the Q-values in the vicinity of the current state-action-pair. If the difference of the Q-value q_i of the current example to its predicted value $\hat{q}_i$ exceeds σ_l, it is added to the database, which means that a function `quite_new` takes the value `true`:

$$\texttt{quite\_new}(q_i, c_1) = \begin{cases} \texttt{true}, & |\hat{q}_i - q_i| > c_1\,\sigma_l \\ \texttt{false}, & else \end{cases}$$

for a constant c_1. To decide whether a denser sampling is needed, we relate σ_l to the global standard deviation σ_g of the Q-values of all examples constituting the Q-function, which means that we consider a function `quite_sparse`:

$$\texttt{quite\_sparse}(c_2) = \begin{cases} \texttt{true}, & \sigma_l > c_2\sigma_g \\ \texttt{false}, & else \end{cases}$$

for a constant c_2. Both criteria are taken from [4].

4 Appearance-Based Modeling

In this section we first describe how we represent the states and actions in our application. Afterwards we explain the measure of the distance $dist_{i,j}$ between two state-action-pairs i and j.

Definition of States. In a RL-environment the design of the state representation can be difficult, if the states have to represent continuous information. As stated above, it is important to keep the state space small, because the number of possible paths from one state to the final state increases exponentially. The usual solution in an application with moving cameras is to limit the camera to certain

positions on a sphere [5]. In contrast, our approach will allow arbitrary camera positions, because the positions will be encoded only implicitly by the perceived visual appearance and not by camera parameters. This appearance is captured in a set of features (e.g. interest points, but not necessarily). Each feature f_i is attached to the view in which it is detected. For each view we describe the *visibility* of a feature by the following expression: `visibility`(feature f_i) A *state* can now be defined by a list of visible features. The notation f_i stands for a feature and the index identifies this feature globally, i.e., across views. Since it is only necessary to identify whether or not two image-features represent the same global feature when computing the similarity between states, a pair-wise feature matching can be used to implement this global assignment. An encoding of a state can exemplarily look like this: `visibility`(f_1) $\wedge$ `visibility`(f_7) $\wedge$ `visibility`(f_{14}) This means that in this state the globally numbered features f_1, f_7, and f_{14} are visible. Thus, the current state of the RL system is defined by the features which are visible in the current view.

Definition of Actions. Actions of the agent are possible camera movements around an object. They are defined by their direction. As we want to proceed purely appearance-based we cannot utilize 3D information, thus these directions can be defined only in the 2D image plane of the current view. This is illustrated in the left diagram of Fig. 1. The expression which describes an action has the form `to`(α) where the parameter is simply an angle $\alpha \in [0; 2\pi]$ taken around the image center of the current view. Since the camera moves around the sphere of the object in 3D space, we have to derive from α a direction in 3D space. For this purpose we project the in-plane direction onto the object's sphere and choose a fixed step length as shown in the left diagram of Fig. 1. The choice of the next direction of movement is motivated by the idea that advantageous directions are those which promise a change in the density of the features. The left and right images of Fig. 2 show directions the agent is allowed to choose from for two example states. These directions are calculated by forming a linearly interpolated histogram of the feature point density. Each of 36 bins represents a range of 10 degrees, in which the number of features is counted. If bin_i is the number of features in bin i, the value of each bin is then replaced by $bin_i := 2 \cdot bin_i - bin_{i-1} - bin_{i+1}, i = 0, \ldots, 35$. Finally, the maxima of the resulting histogram define the valid directions for the next camera movement. The left part of Fig. 2 explains the derivation of actual camera movements in 3-space from their directions in 2-space of image planes. The centers C_1 and C_2 of two cameras are shown. After applying the movements determined by the actions `to`(α_1) and `to`(α_2), respectively, the cameras take their new positions C_1' and C_2'. These new positions are unambiguous, because the distance of the image plane to the center of the object remains constant. The distance d between the new camera positions determines the similarity of the actions `to`(α_1) and `to`(α_2). The right part of Fig. 2 illustrates the behavior of the state similarity for two examples. On the abscissa the angles in the range of $[-\pi; \pi]$ of a camera are logged, which is rotated around an object with a fixed axis through the center of the object and a step size of 0.1 radians. The ordinate represents the values

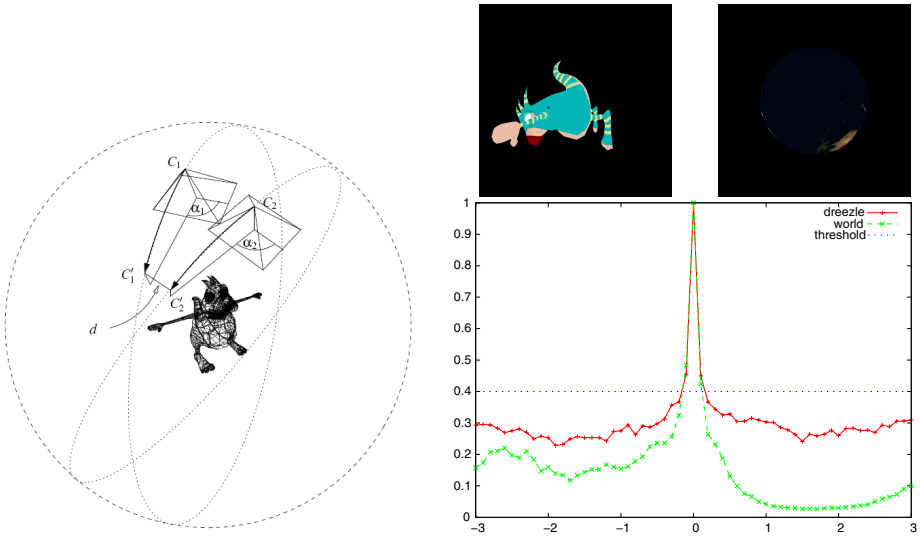

Fig. 1. Left: Derivation of actual camera movements in 3-space from their directions in 2-space of image planes. Right: Behavior of the state similarity for two examples.

of the state similarity function $s_{i,j,\texttt{visibility}}$. For each of the recorded views the state similarity to a reference view is calculated. The red curve represents the similarity values for the object "dreezle", the reference view of which is depicted in the upper left, the green curve does the same for the reference view of the object "world", shown in the upper right. Both reference views have been recorded at the position of zero degrees. The similarity function provides reliable values, even for the object "world", where the reference view contains very few features only. In addition, the inflection point p we use for the threshold function is drawn as dotted line at the similarity value of 0.4.

Distance Measure for State-Action-Pairs. We now examine the distance function $dist_{i,j}$ mentioned in Sec. 3. It is based on a similarity measure $sim_{i,j} : (i,j) \rightarrow [0;1]$:

$$dist_{i,j} = 1 - sim_{i,j}.$$

This similarity measure is defined as a product of the state similarity and the action similarity. $sim_{i,j} = sim^S_{i,j} \cdot sim^A_{i,j}$. Both, sim^S and sim^A, are functions with range $[0;1]$. They are defined as $sim^S_{i,j} = t(s_{i,j,\texttt{visibility}})$ and $sim^A_{i,j} = t(s_{i,j,\texttt{to}})$, where t is a sigmoidal function that maps its argument to $[0;1]$. It imposes a soft threshold to punish poor values quickly while at the same time allowing small deviations from the optimum: $t(x) = \frac{1}{1+\mathrm{e}^{-\mu(x-p)}}$ where μ is the steepness of the slope (set to 20 in our tests) and p is the inflection point (set to 0.4 for $sim^S_{i,j}$ and to 0.8 for $sim^A_{i,j}$). The unthresholded state similarity $s_{i,j,\texttt{visibility}}$ is based on the amount of overlapping between two views. A larger overlap leads to a larger similarity. The overlapping is measured by counting the features that reference the same global feature. This is measured

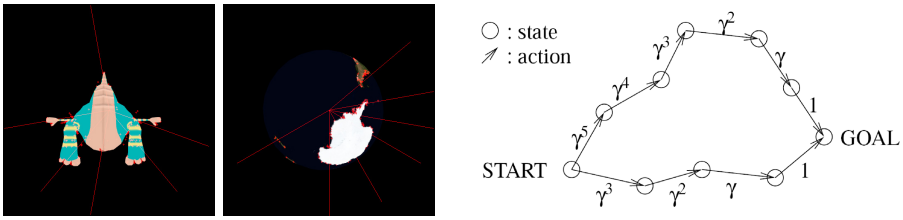

Fig. 2. Left: Appearance-based choice of actions for two objects. Red lines indicate possible directions for the next camera movement. Features are indicated by red dots on the object (close to the direction lines). The directions are computed as local maxima of the variation of the feature density. Right: Short scan paths are enforced by using a discount factor $\gamma \in]0;1[$ and giving zero rewards in all states except for the goal state. A shorter path leads to a less discounted reward. The exponents of γ are derived from the iterated application of the update equation.

by actually carrying out a feature matching between both views. The relation of the amount of same features to the amount of all features is used as the similarity measure. Let $\texttt{visible}(x)$ be the set of all features visible in state x. Then

$$s_{i,j,\texttt{visibility}} = \frac{\texttt{visible}(i) \cap \texttt{visible}(j)}{\texttt{visible}(i) \cup \texttt{visible}(j)}. \tag{1}$$

The right diagram of Fig. 1 illustrates the property of this similarity measure by means of two examples. The term $s_{i,j,\texttt{to}}$ expresses the unthresholded similarity between two actions. It depends on the `to`-expressions introduced in Sec. 4. Two of these expressions point to the same state if the cameras capture the same features after moving in those directions. We use the distance of the camera centers after moving them along the directions the actions point to. The approach is also depicted in the left diagram of Fig. 1.

5 Application

The previously described framework has been used in an application that learns scan paths which are short but nevertheless allow the discrimination of two very similar objects. These objects look essentially the same except for a minor difference, for example in texture as seen in the left column of Fig. 3. This figure illustrates the following. Left: example objects and their differences. Upper row: The two versions of object "dreezle" can be distinguished by the starlike figure on its belly. Lower row: The two versions of object "world" can be distinguished by the yellow scribbling. Right: Phase 1 of the identification of the goal state (determining the most similar view for each object). Each row shows one example of the image retrieval process. The left column shows the current view of the agent. For *each* object in the database we determine the most similar view to the one in the left column. The right column shows the determined most similar view of only one special object, namely of that object which together with the object

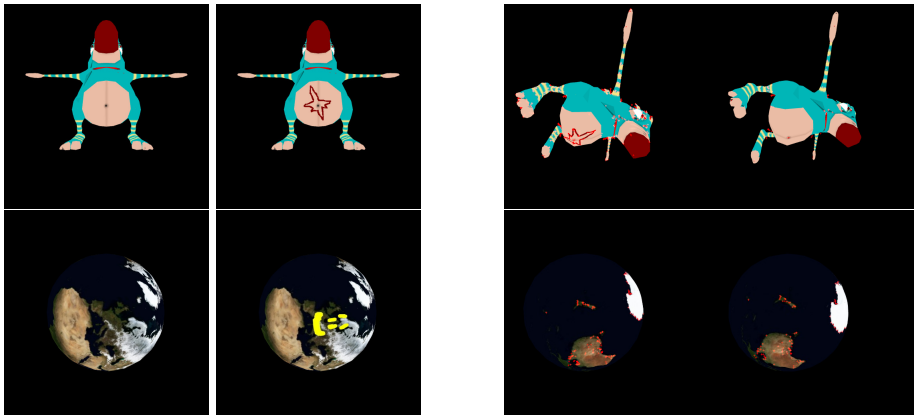

Fig. 3. Left: Example objects and their differences. Right: Phase 1 of the identification of the goal state (determining the most similar view for each object).

of the left column makes up the pair of similar objects. To set up the learning task for each pair of objects, both objects are scanned and their features are stored in a database. This database is used to generate the reward for the agent. One of these objects is then presented to the agent. The agent moves around the object, records images, and stops when she receives a reward of 1. Then the next episode starts. As a result, the agent learns to scan an object in such a way that she is able to distinguish it from a very similar object stored in the database.

Feature Detection and Calculation of Descriptors. To recognize and match feature points between images of an object, which have been taken from different camera positions, they are required to be reasonably stable. To comply with this requirement, we use a scale space Harris detector, combining the real time scale space of Lindeberg [6] with the Harris detector [7], while modifying the latter to make it work in a scale space pyramid [8]. To attach descriptors to these feature points we use Lowe's SIFT descriptors [9]. While he only uses those feature coordinates that pose maxima in scale space, we take all feature points into account as long as they are a local maximum in their own scale. This overcomes some stability deficiencies with the original approach that we experienced and that have been reported in [10] as well. As a result we get up to 500 features in each image. To reduce this amount, each camera takes a second image from a slightly different view point. Then, we apply a feature matching using a kd-tree [11]. The resulting correspondences are filtered by applying the epipolar constraint [12]. Only those feature points that survive this procedure are stored in our database. We aim at about 150 features for each view.

Rewards. We aim at learning the shortest scan path around the sphere of an object to a view that allows for the reliable discrimination from another similar object. As the agent aims to maximize the sum of all rewards, positive rewards in each step will keep the agent away from the goal state as long as possible. For this reason we do not reward a simple movement at all. The only action that

receives a reward is a movement to the goal state. A shortest path can be made attractive to the agent by (ab)using the discount factor γ. If we set $\gamma := 1$, all Q-values will approach 1, since r is always 0. But with $\gamma \in]0;1[$ we will find a greater Q-value attached to states closer to the goal than to those farther away. This is illustrated in the right of Fig. 2 and leads to the desired preference of short paths towards the goal state.

Identification of the Goal State. As noted in the last subsection the reward is zero for all actions except those reaching the goal state. In fact, in our setup the goal state is *defined* as the state where the reward is one. In each step an image I is captured. After computing features and their descriptors of this image, they are used to identify the goal state. This identification proceeds in two phases. Phase 1 iterates through *all* objects inside the database (which consists of several pairs of pairwise similar objects) and identifies for each object the most similar view. This is basically an image retrieval task. Given one object O, our approach begins with building a kd-tree K_O with all descriptors belonging to O. For each descriptor $D_{I,i}$ of I, the most similar descriptor $D_{O,j}$ in K_O is identified and its Euclidean distance $d(D_{I,i}, D_{O,j})$ is computed. This distance d is used to vote for all views of object O, $D_{O,j}$ belongs to: $\texttt{score}(d) = \frac{1}{d}$ (taking the prevention of zero division into account). These scores are accumulated over all descriptors of I for each view separately. The one which receives the largest sum is taken as the most similar view of object O to I. This is done for all objects O in the database. The right column of Fig. 3 shows resulting pairs of images. Phase 2 aims at the decision whether the most similar images, e.g., I_{O1} and I_{O2}, of two candidate objects $O1$ and $O2$ show a significant difference. If so, we have reached our goal of finding the most discriminative views of two similar objects. Then we can mark the current state as a goal state by giving a reward of one. Finally we can find out which of the similar objects we have currently at hand. To do this, we reuse the similarity measure $s_{i,j,\texttt{visibility}}$ of Sec. 4. We compute the similarity of image I (corresponding to state i in (1)) to both candidates and take a normalized difference:

$$g = \frac{|s_{I,I_{O1},\texttt{visibility}} - s_{I,I_{O2},\texttt{visibility}}|}{\max(s_{I,I_{O1},\texttt{visibility}}, s_{I,I_{O2},\texttt{visibility}})} \tag{2}$$

If this value g exceeds a certain threshold, the most discriminative view between the two similar objects is considered identified and the current episode ends. (Once this discriminative view is identified it is simple to determine which of both objects has been scanned.) For our learning scheme a threshold of 0.15 suffices. Fig. 4 shows results of this approach. Phase 2 of the identification of the goal state (finding the discriminative view) is illustrated here. On the abscissa the angles in the range of $[0; 2\pi]$ of a camera are logged, which is rotated around an object with a fixed axis through the center of the object. The ordinate represents the values of the normalized difference g (cf. (2)). Each value of the red curve is computed from three images: the test image I and the two best candidate images I_{O1} and I_{O2}. For the red curve image I shows the "dreezle" figure at a zero rotation angle, depicted on the left in the upper row of Fig. 3. One example

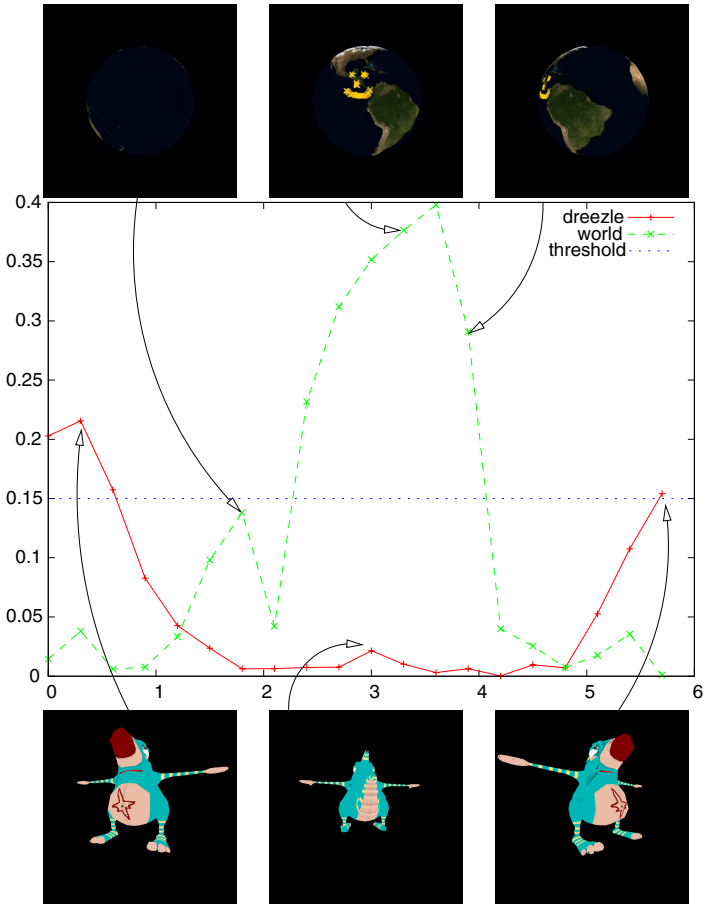

Fig. 4. Phase 2 of the identification of the goal state (finding the discriminative view)

for I_{O1} is the left image in the bottom row, which shows one view of the "dreezle" with the star on it belly, I_{O2} (which is not shown) is the same view of the object without the star. The resulting difference value g for this example is larger than the threshold of 0.15 marked by the dotted line. Thus this view is discriminative enough to tell both objects apart. In contrast, the view in the middle of the bottom row is not discriminative enough, as it does not reveal if the object has a star on its belly. This fact is represented well by the low value of the normalized difference function g. The upper row and green curve show more examples for the object "world". Here the reference view I has also been recorded at the position of zero degrees. This diagram illustrates that the objects can be distinguished more reliably with a larger visibility of their discriminative part.

Results. The left column of Fig. 5 illustrates the development of the set of samples of state-action-pairs constituting the Q-function. After the first episode only two samples are found inside the sample set. We will briefly examine these two

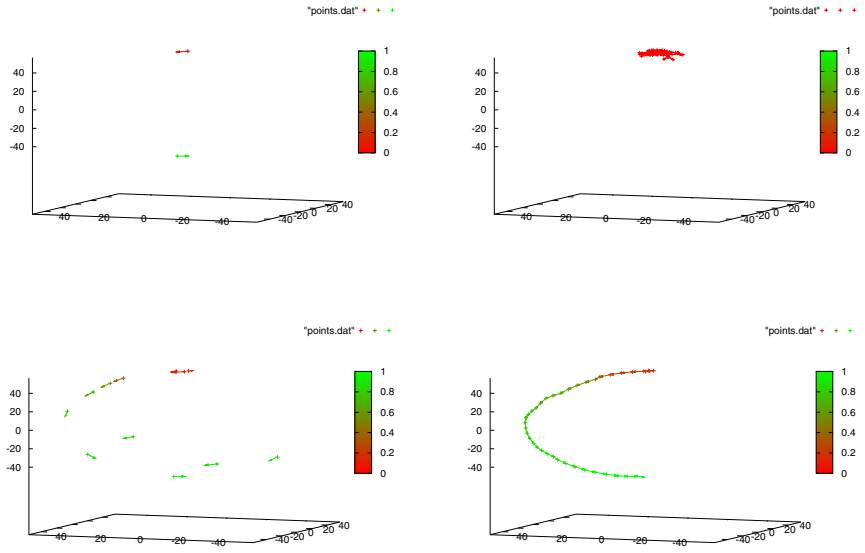

Fig. 5. Left: Learned rules in the form of state-action-pairs used to approximate the Q-function. Right: Application of the learned rules in a discrimination task.

samples to comprehend the constitution of the Q-function. The first reward the agent gets after the first episode is zero. Because nothing is known about the Q-function, this state-action-pair is immediately added to the set of samples. This sample predicts the reward of all following state-action-pairs also as zero. This holds true until the first step encounters a reward of 1. Then one additional sample is added to the Q-function: the state-action-pair that led to the goal state. The state-action-pair with the goal state is not inserted because it does not meet the conditions presented in Sec. 3. This is the end of the first episode. Further episodes insert samples according to the rules presented in the end of Sec. 3. Basically, it is tested if the Q-function needs denser sampling or not. The right column of Fig. 5 shows the paths an agent takes when she uses the Q-functions depicted in the left column of Fig. 5. It is obvious that the paths get shorter the more episodes have been completed. The examples clearly indicate the applicability of the RRL approach to computer vision learning tasks, especially because of its capability of handling continuous domains. Additionally, the Q-function consists of comprehensive state-action-pairs, where each pair encodes a rule that indicates the agent the merit of moving the camera in a certain direction with a given view at hand. This way, a human trainer can easily add information to the Q-function, e.g., by simply presenting a view of the object and a corresponding preferable direction. In addition, the relational encoding removes dependencies on coordinate systems, which may arise when using traditional RL approaches that use camera positions as their basis of a state's encoding. Fig. 5 in detail:

Left column: Learned rules in the form of state-action-pairs used to approximate the Q-function. The axes encode the 3D space with the object in the center of the coordinate system (not shown in these diagrams). A state is given as the origin of a vector and encodes a current view of the object, an action is given by the direction of a vector and encodes the next direction of the movement of the camera. (Top diagram: after 1 episode, bottom diagram: after 3 episodes.) The color encodes the assigned Q-value. Red stands for an expected accumulated reward of zero, while green indicates a reward of one. Right column: Application of the learned rules in a discrimination task. Here the paths are depicted an agent chooses based on the learned Q-function until a successful discrimination took place or a maximum number of steps has been exceeded. (Top diagram: using the Q-function learned after 1 episode (the agent has not found a path within 40 steps), bottom diagram: using the Q-function learned after 10 episodes.) Again, the color encodes the predicted Q-value of the path taken.

6 Conclusion

We proposed an adaptive, self-learning system which utilizes RRL and applied it to the problem of the discrimination between similar objects which differ, e.g., in salient features visible from distinct views only. Usually existing object recognition systems have to scan an object from a large number of views for a reliable recognition. In our RRL approach we introduced a representation of states and actions that are entirely based on the perceived appearance of an object. This enables the system to rapidly learn scan paths for objects and to reliably distinguish similar objects from only a few recorded views. The appearance-based approach and the possibility to define states and actions of the RRL-system with logical descriptions allow for a large reduction of the state space and thus save storage and computing time.

7 Future Research

This work leads to a number of possible future extensions. The image retrieval using a simple kd-tree is quite slow and can be accelerated. Using the best-bin-first-technique of [11] accelerates the process slightly, but the matching reliability deteriorates quickly and values below 20.000 comparisons are strongly discouraged. An integration of the generation of the object database into the learning algorithm would result in a system enabling the agent to explore her environment and constantly add features into the object database. Similar objects can share some of the feature sets. However, to generate representations of distinct objects, a criterion will have to be developed that groups feature sets according to their belonging to the same unique object type.

Acknowledgments. This research was funded by the German Research Association (DFG) under Grant PE 887/3-3.

References

1. Dzeroski, S., De Raedt, L., Driessens, K.: Relational reinforcement learning. In: Machine Learning, vol. 43, pp. 7–52 (2001)
2. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
3. Gartner, T., Driessens, K., Ramon, J.: Graph kernels and gaussian processes for relational reinforcement learning. In: Inductive Logic Programming, 13th International Conference, ILP (2003)
4. Driessens, K., Ramon, J.: Relational instance based regression for relational reinforcement learning. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 123–130 (2003)
5. Peters, G.: A Vision System for Interactive Object Learning. In: IEEE International Conference on Computer Vision Systems (ICVS 2006), New York, USA, January 5-7 (2006)
6. Lindeberg, T., Bretzner, L.: Real-time scale selection in hybrid multi-scale representations. In: Griffin, L.D., Lillholm, M. (eds.) Scale-Space 2003. LNCS, vol. 2695, pp. 148–163. Springer, Heidelberg (2003)
7. Harris, C., Stephens, M.: A Combined Corner and Edge Detector. In: 4th ALVEY Vision Conference, pp. 147–151 (1988)
8. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. International Journal of Computer Vision 60(1), 63–86 (2004)
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (2004)
10. Baumberg, A.: Reliable feature matching across widely separated views. In: CVPR 2001, p. 1774 (2000)
11. Beis, J., Lowe, D.: Shape indexing using approximate nearest-neighbor search in highdimensional spaces (1997)
12. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004)