A Vision System for Interactive Object Learning

Gabriele Peters
Universität Dortmund, Informatik VII,
Otto-Hahn-Str. 16, D-44227 Dortmund, Germany
peters@ls7.cs.uni-dortmund.de

Abstract

We propose an architectural model for a responsive vision system based on techniques of reinforcement learning. It is capable of acquiring object representations based on the intended application. The system can be interpreted as an intelligent scanner that interacts with its environment in a perception-action cycle, choosing the camera parameters for the next view of an object depending on the information it has perceived so far. The main contribution of this paper consists in the presentation of this general architecture which can be used for a variety of applications in computer vision and computer graphics. In addition, the funcionality of the system is demonstrated with the example of learning a sparse, view-based object representation that allows for the reconstruction of non-acquired views. First results suggest the usability of the proposed system.

1 Introduction

Both, computer vision as well as computer graphics are concerned with the visual appearance of objects of the real world. Major problems of computer vision are the recognition or classification of objects from images. An important challenge of computer graphics consists in the generation of internal models of objects from images, e.g., for the purpose of geometric modelling or graphic illustration. For both fields of research the acquisition of an object representation is necessary.

The current situation is for the most part marked by a separation between object acquisition and further processing of the acquired information in a specific application, whether in the field of computer vision or in the field of computer graphics (figure 1). This often leads to the fact that the recorded data do not meet the requirements of the application. A usual way to obtain reasonable results anyhow is the development of heuristics. Another approach, which has been used to different extents in the mentioned fields of application, is the active visual acquisition of objects. This means that the processing of the recorded data gives feedback to the acquistion part of the system. In analogy to human information processing the system should autonomously learn strategies of object acquisition on the basis of application-specific objectives only.

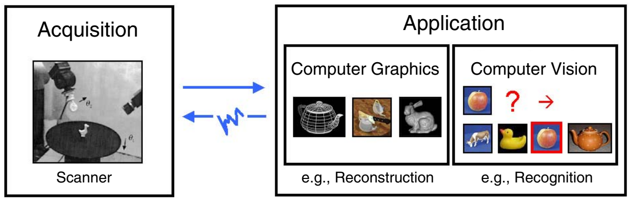

Figure 1. Usually the data acquisition and the further processing of the data occur strictly successively.

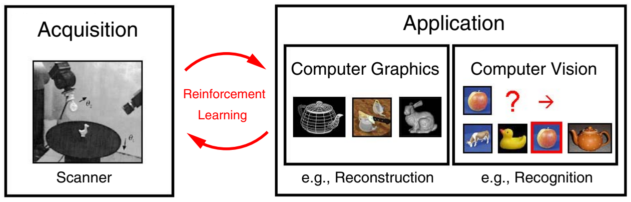

Figure 2. The proposed vision system allows for a feedback from the application to the data acquisition via reinforcement learning.

We propose an architectural model for a responsive vision system based on techniques of reinforcement learning that takes these considerations into account (figure 2). It is capable of acquiring object representations based on the intended application only and thus can be employed for a variety of tasks. The system can be interpreted as an intelligent scanner that interacts with its environment in a perception-action cycle, choosing the camera parameters for the next view depending on the information it has perceived so far. So, the subsequent input depends on the actions taken previously.

Section 2 summarizes related work. In section 3 we propose the responsive vision system. In section 4 this system is applied to a typical problem, the acquisition of a view-based object representation, which can be used for view morphing. In addition, first results for this problem are presented here. Finally, in section 5 some conclusions are drawn.

2 Related work

The term *viewpoint planning* summarizes techniques of deciding the optimal viewpoint distribution which captures all relevant information about an object or a scene for a specific task. Within the last decade a variety of methods for viewpoint planning have been proposed. But in the field of computer vision it is usually not employed until the level of object recognition [1], instead of utilizing it also for object acquisition. This holds true also for [2], where the dynamic aspect plays a role also not until the level of recognition. In [3] an approach to the acquisition of view-based object representations is proposed where key-frames for the representation are chosen from an image sequence. But the strategy for the choice of key-frames as well as the scan path are given. The same holds true for the adaptive tracking approach proposed by [4]. In [5] an approach to 3d model acquisition with an "eye-in-hand" configuration is described but again with a given scan strategy. Also for more adaptive systems, which try to adapt the scan path to the object or the application, holds true that the strategies for scanning an object or a scene are mostly given by the developer [6, 7, 8, 9, 10]. Only recently these strategies are also learned automatically, for example with methods of reinforcement learning. This approach is chosen, e.g., by [11, 12, 13] for the autonomous emergence of strategies for object recognition. However, we do not know any approach to object acquisition by active learning up to now and propose a method which adaptively learns a view-based object representation without a given strategy.

3 A responsive vision system

Figure 3 displays the general architecture of a vision system that learns object representations interactively depending on the intended application. The different components of the system are separated in three modules: *Learning* in the upper right part of the diagram, *Acquisition* in the lower left, and *Application* in the upper left. The resulting object representation is shown in the lower right part. In the diagram examples for concrete design decisions are printed in italics. In the next section the different modules are described, in section 3.2 the interaction between these modules is explained, and in section 3.3 we focus on some design decisions a user has to make when she likes to utilize this system for her own needs.

3.1 Modules

Module *Learning*. In traditional approaches of computer science a problem is solved by an algorithm that has been developed by the programmer who has reflected on the problem. In contrast to this, approaches exist which delegate also the discovery of solution procedures to the machine. Often principles of nature are a role model for such approaches. One example are evolution-based methods such as genetic programming [14]. Another example are behavior-based techniques such as *Reinforcement Learning*, which seem to be an appropriate approach to our problem of object learning. The principles of reinforcement learning are sketched in the following. An *agent* interacts with its *environment* by perception and action. In an interaction step i the agent receives information $s_i \in S$ on the current *state* of the environment as input via perception. Then the agent chooses an action $a_i \in A$ according to its *policy function* $p : S \to A$. The action is carried out and changes the state of the environment (*transition function* $d : S \times A \to S$). The agent is able to adapt its behavior to certain conditions. For this purpose the agent receives direct feedback for its last action by a scalar *reward signal* r. In addition, a valuation of the state transition is conducted by a *value function* $Q : S \times A \to \mathbb{R}$. The behavior of the agent should maximize the long term sum of the reward signals, i.e., the expected *return*. The value function is learned by systematic trial-and-error for which a bunch of techniques has been developed. In [15, 16, 17] overviews are given.

Module *Acquisition*. There are some steps in the acquisition process which are common to all kinds of applications and all kinds of data structures of the object representation. Whether a 3D model or a view-based representation should be learned, in any case there will be some steps of preprocessing of a perceived view of the object, such as image enhancement or segmentation techniques. As the perceived image is the only information about the environment available to the agent, there will also be some feature extraction technique. Examples for features are responses of a Gabor wavelet transform or the scale-invariant feature transform (SIFT) [18]. Depending on the specific design there may also be a calculation of a representation of a single view (e.g. in form of a graph labeled with local object information or in form of the 3d positions of object features) before the new information is incorporated into the object representation. At last, to extract useful information from a sequence of views the

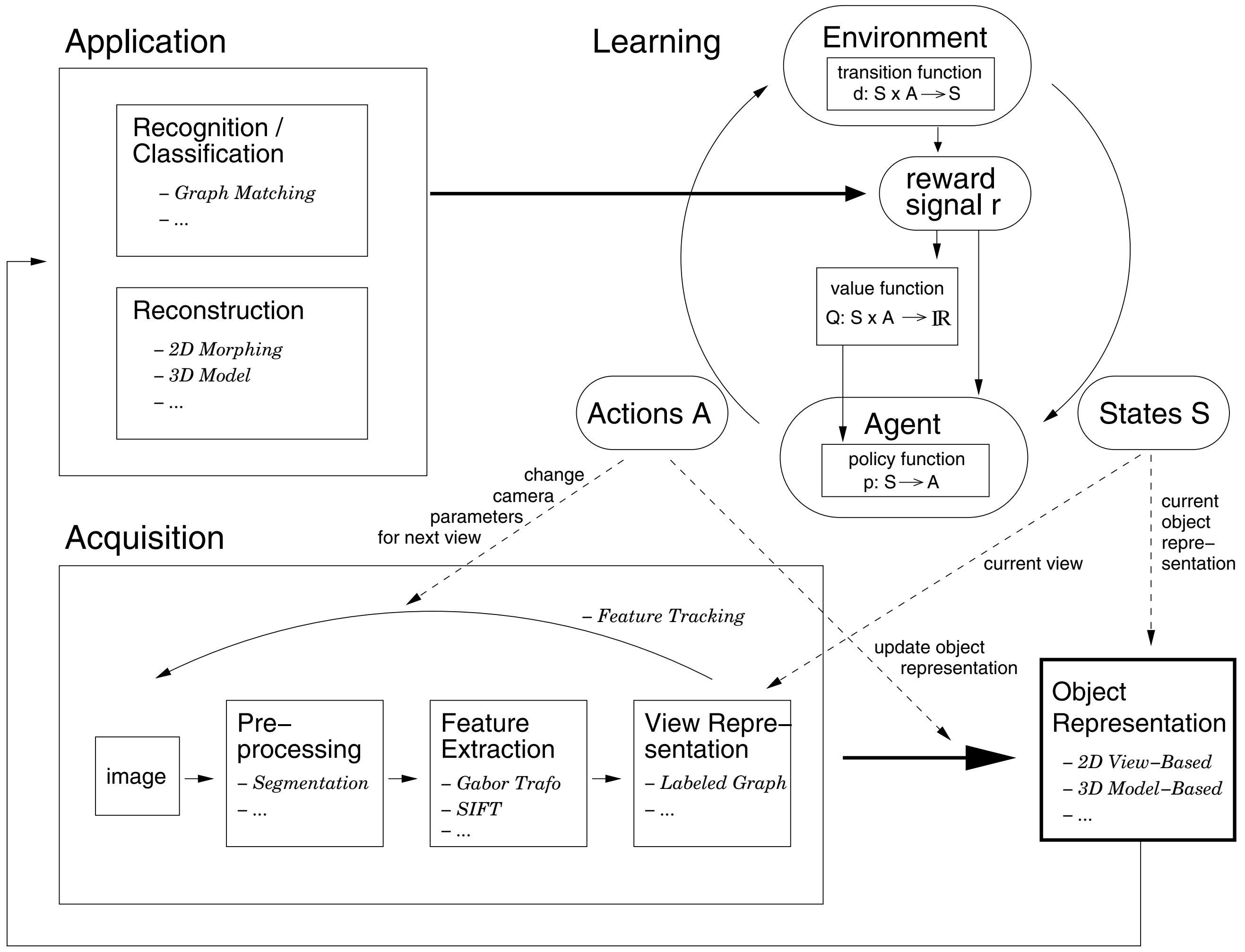

Figure 3. Architecture of a vision system that learns object representations.

common information between successive views has to be exploited. This can be realized for example by 2d feature tracking or by bundle adjustment techniques.

Module *Application*. The two major fields of application for interactive object learning are computer vision with the goal of object recognition and classification and computer graphics with the purpose of object reconstruction. After the learning process and given a test view or a sequence of test views recognition and classification should be possible even if the agent has not experienced those test views. Analogously, the reconstruction of the complete object and the rendering from unfamiliar views are claimed. This module gives the crucial feedback to the *Learning* module utilizing the object representation learned so far.

3.2 Interaction between the modules

The *Learning* module implements the perception-action cycle of the system. It produces actions and states as output and takes feedback from the defined application as input. The feedback from the application modifies the subsequent environment of the agent. The goal of the acquisition process is defined by the reward signal. This reward signal is calculated according to the specific application. Thus, goal-directed behavior emerges, resulting in different object representations for different applications.

3.3 Design decisions

Besides decisions concerning the data structure of the object representation and the way how new information is integrated (*Acquisition* module) the majority of design decisions falls in the *Learning* module. Here we have to define the following parameters:

States S: A state should incorporate the information necessary for the agent to decide for the next action. It seems reasonable to include at least information on the object learned so far and on the current view which is perceived.

Actions A: An action should change the camera parameters, i.e., position and orientation of the camera in relation to the object, but also the focal lenght could be useful to change depending on the application. In addition, an update of the object representation can also be implemented as action of the agent.

Value function Q: The estimation of the value function is the goal of reinforcement learning. It reflects how good it is for the agent to perform a given action in a given state. The definition of the policy function has to be taken into account here.

Policy function p: It determines for each state the next action of the agent. One of the design decisions here consist in the determination to which extend the strategy learned so far is exploited or, on the other hand, is ignored in favor of exploration.

Reward function r: Its definition is the crucial design decision as it determines the goal of the learning process. Here the response of the application must be integrated.

Summarizing, as the chosen application determines the learning process and the resulting object representation this system will be applicable for many purposes, in computer vision as well as computer graphics. Once a basic system has been established the user primarily has to define states, actions, and a reward function to adopt it for a different application.

4 Application to a standard vision problem

We have applied the proposed system to a standard task in computer vision, namley the acquisition of a sparse, view-based object representation. To test whether the relevant information on the object has been captured by the learned scan path we reconstruct non-acquired views from perceived views by 2d view morphing. In section 4.1 we describe briefly the components of this application. A more detailed description can be found in [19]. In section 4.2 we concretize the design decisions of section 3.3, and in section 4.3 first results are summarized, which have also been described in more detail in [19].

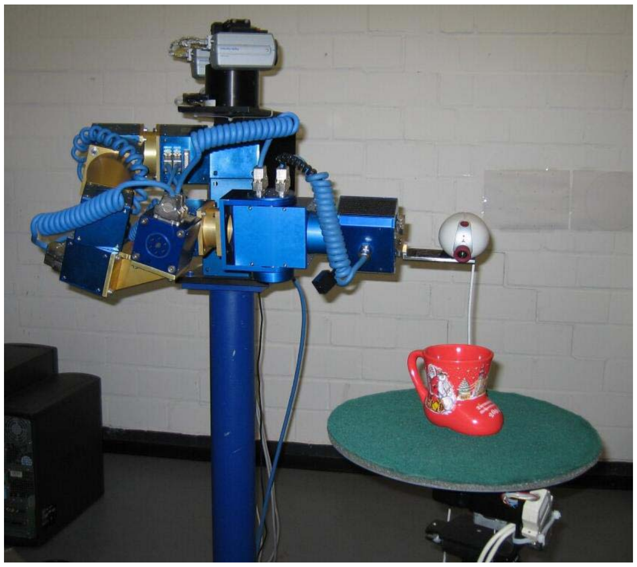

Figure 4. Simulated setup with camera and object.

4.1 Basic components of the system

Data base and view representation. We simulate an eye-in-hand camera setup with the object on a table such as shown in figure 4. The camera rotates around the object at a fixed distance and is oriented to the center of the object base. The observed object views are represented in a data base which contains views for 100 lines of longitude and 25 line of latitude on the upper view hemisphere resulting in 2500 views for one object. Each of the recorded views is preprocessed by a *Gabor wavelet transform* using a filter bank with wavelets of 8 orientations and 4 frequencies followed by a simple *segmentation* utilizing gray level values. A regular grid graph is placed on the object segment and the nodes of the graph are labeled with the corresponding Gabor wavelet responses.

Correspondences by tracking. Correspondences between successive views are obtained by tracking the nodes of a graph from frame to frame within a local area of the view hemisphere. This is realized by utilizing the information obtained from the Gabor transform at each node of the graph [20]. The grid graph shown in the left view of figure 5 is tracked along the sequence to the view shown on the right. The nodes stay on corresponding object points. A similarity function between two graphs based on the Gabor wavelet responses is defined reflecting the similarity between the particular views.

Sparse object representation. A sparse, view-based

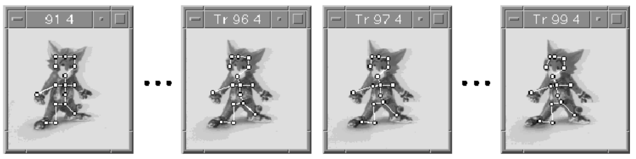

Figure 5. Tracking of object points.

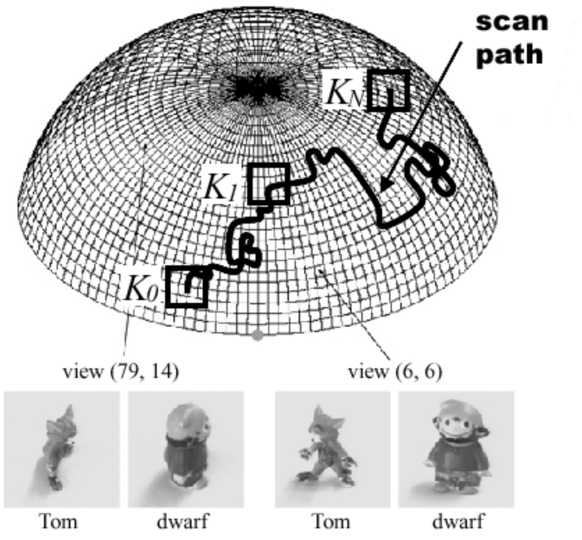

Figure 6. Sample views of two objects and a possible scan path with three key views.

object representation consists of original grid graphs and tracked graphs of only some key views of the scanned path. Given a scan path on the view hemisphere we start with its first view (key view K_0) and incorporate its original grid graph $\mathcal{G}_{orig}^{K_0}$ in the object representation. This graph is tracked along the scan path until the similarity between the tracked graph at the current view of the scan path and $\mathcal{G}_{orig}^{K_0}$ drops below a preset threshold. The tracked graph $\mathcal{G}_{track}^{K_1}$ for this second key view K_1 is also stored in the object representation. For K_1 a new grid graph $\mathcal{G}_{orig}^{K_1}$ is generated and incorporated into the representation as well. It is also tracked until the similarity to $\mathcal{G}_{orig}^{K_1}$ drops again below the threshold, and so on. For the first and the last key view of the scan path only one graph is stored ($\mathcal{G}_{orig}^{K_0}$ and $\mathcal{G}_{track}^{K_N}$, respectively), whereas for each other key view $K_j, j = 1, \ldots, N-1$ of the scan path two graphs $\mathcal{G}_{track}^{K_j}$ and $\mathcal{G}_{orig}^{K_j}$ are stored in the object representation, ensuring piecewise correspondences for local areas of the view hemisphere (figure 6).

Reconstruction of non-acquired views. An unfamiliar view is morphed from those two consecutive key views which are closest to it, using the correspondences provided by the tracking procedure. For view morphing we use a standard technique described in [21]. A morphed view can then be compared to its original version by an error function. This yields an error for a reconstructed view. In the example in figure 7 the non-acquired view $(7, 11)$ is reconstructed from the key views $(3, 7)$ and $(14, 7)$. It can be compared to the original view $(7, 11)$.

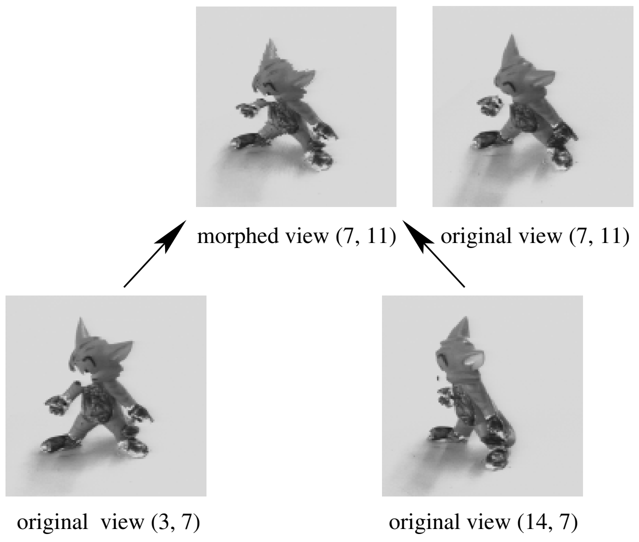

Figure 7. Reconstruction of a non-acquired view.

This technique is used for the calculation of the reward signal after each step of a scan episode as well as for the calculation of the total reconstruction error after a scan path has been learned.

4.2 Design decisions for the application of 2d view morphing

States S: The current position of the camera only is not sufficient to define a state of the environment. But the complete path which has been scanned would yield too many states. Thus, we define a state as a vector which contains the current position of the camera and four values which describe the degree of unfamiliarity of the areas to the north, east, south, and west of the current position on the view hemisphere, respectively. To calculate the degree of unfamiliarity of an area we assign a value to each unfamiliar position of an area. This value is the distance from this unfamiliar position to the next familiar position (i.e. one that has already been scanned). Then the value of an area is the sum of all values of unfamiliar positions in this area. The possible values of an area are quantized into