Acquiring Visual Servoing Reaching and Grasping Skills using Neural Reinforcement Learning

Thomas Lampe and Martin Riedmiller

***Abstract*— In this work we present a reinforcement learning system for autonomous reaching and grasping using visual servoing with a robotic arm. Control is realized in a visual feedback control loop, making it both reactive and robust to noise. The controller is learned from scratch by success or failure without adding information about the task's solution. All of the system's major components are implemented as neural networks.**

The system is applied to solving a combined reaching and grasping task involving uncertainty directly on a real robotic platform. Its main parts and the conditions for their successful interoperation are described. It will be shown that even with minimal prior knowledge, the system can learn in a short amount of time to reliably perform its task. Furthermore, we describe the control system's ability to react to changes and errors.

I. Introduction

The ability to react quickly to changes is essential for any robotic application that hopes to be useful in natural all-day environments. This includes the area of autonomous control of actuators for reaching tasks, which is not only important in traditional robotic scenarios, but will also play an important role in future human-robot interactions.

Classical planning approaches are limited in this respect. Even using modern hardware, planning a complete reaching and grasping movement requires a significant amount of time, and is usually performed in an open-loop manner [1]. Closed-loop control is frequently limited to grasp adjustments using force [2], [3], [4] or proximity sensors [5] at or just before contact with the target, but does not incorporate visual information during the reaching stage. The time requirements make fully reactive control difficult, even while more recent works are nearly capable of planning in real-time [6] and may re-plan their target when failure seems likely [7].

In contrast, visual servoing methods [8] inherently adhere to the reactive paradigm [9]. Here, a control signal is generated directly from the sensory input, without the need for high-level reasoning. However, even approaches capable of truly reactive operation [10], [11], [12] typically incorporate a significant amount of prior knowledge about the system and the task's solution. Common requirements range from camera calibration over the use of complete system models for generating the visual-motor Jacobian to hand-crafted control strategies [13].

The authors are with the Machine Learning Lab, Department of Computer Science, Faculty of Engineering, Albert-Ludwigs-University of Freiburg, D-79110, Germany (email: {tlampe,riedmiller}@informatik.uni-freiburg.de).

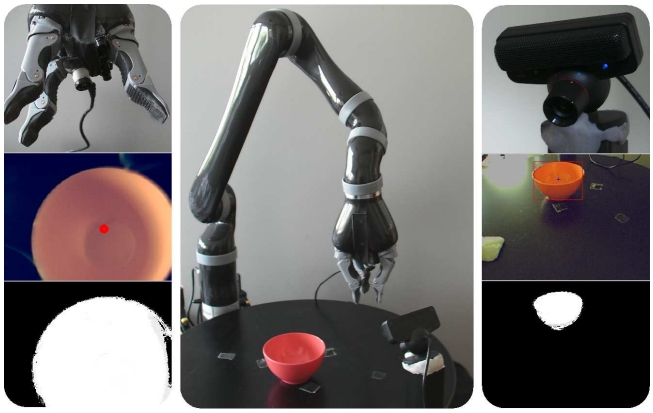

Fig. 1. Setup of the system with the cameras, their visual input and the segmented object. Note that the markings visible on the table do not directly correspond to the training positions, but were used for guidance when placing the object.

Machine learning techniques constitute a useful tool for reducing the amount of expert knowledge required for autonomous control. Ideally a system would combine the reactiveness of visual servoing with the knowledge-free policy acquisition of reinforcement learning. One type of learning paradigm that has proven well-suited to generating a behavior even in the complete absence of domain knowledge is value-function-based learning, particularly the variant of Q-learning. In recent years such approaches have been largely overshadowed by policy-based methods, which have produced impressive successes in various applications of robotics, albeit with the use of system models and imitation learning [14], both of which constitute forms of prior knowledge and may not always be available. For instance, models may become skewed over time in real applications and demonstrations may be difficult to perform even for a human, like in the classical inverted pendulum problem. One major reason for the comparatively rare usage of knowledge-free systems is the frequently high system interaction time required, which prevents the application on robotic systems. However, advanced value-based methods such as Neural Fitted Q Iteration [15] work in a much more data-efficient manner and have proven to be usable in real-world systems rather than only simulated problems [16], [17].

It is important to note that we do not aim to pit the system against state-of-the-art visual servoing methods; a system created using prior knowledge will obviously outperform one learned from scratch. Instead, we intend to illustrate that value-based methods can still be applied in the domain

of autonomous manipulation, which may be advantageous in cases where knowledge about the process model or the problem solution is difficult to generate. To do so, we consider a combined reaching and grasping task in which a robotic arm is required to pick up an object. The setup illustrated in Fig. 1 notably includes the use of a hand-mounted camera. Such monocular eye-in-hand setups are generally desirable in manipulation tasks, as they allow for more precise control and additional robustness against errors due to occlusion. However, they also make extracting an accurate 3D pose of an object, which is used in most manipulation systems [1], difficult, and generally require a known system model as well as a calibrated camera for use in planning. Hand-mounted cameras can eliminate the need for camera calibration [18], though at the cost of the requirement that the final camera input at the end of the movement be known in advance. The desire to avoid even such information motivates our use of a value-based method capable of learning only from experience.

As per our main goal, knowledge about the problem provided to the learner will be limited to a bare minimum. The only assumption will be to split the reaching task into three parts, including coarse and fine control as well as vision-based success prediction.

II. Related Work

Among recent successful visual servoing approaches to reaching and grasping using eye-in-hand setups, there are several attempts to reduce the required amount of information. Like our system, Piepmeier et al. [19] have used an uncalibrated camera, while Shademan et al. [20] estimated the visual-motor Jacobian without knowledge of the system, though hand-crafted control laws were used.

Reaching and grasping tasks have been extensively covered using combinations of policy iteration techniques and imitation learning [1], [7], [21], [22]. There has been a trend to reduce the amount of knowledge needed, allowing the generation of trajectories using only a generic initialization [6], [23].

Similar to the split of the system into controllers of different scopes, Kim et al. [24] divide the motion into gross and fine components, solved by table- and hand-mounted cameras, respectively. An early application of such a scope split in neural reinforcement learning can be found in [25], used for thermostat control.

Along the same lines of the implicit grasp success prediction used here, Detry et al. [26] learned affordance densities of objects, which were explored through pick-and-drop movements. However, this was done in task space, while our approach works directly on sensory inputs. Grasping points in task space have also been extracted explicitly, with no knowledge of or assumptions about the target's shape [27], [28].

While not related to autonomous reaching, the setup used in [29] to achieve real-time reactive control in a high-speed ball-bouncing task was quite similar in its nature. In contrast, however, its solution by means of Model-Predictive Control [30] required the use of a model of the process. Learning was addressed in a subsequent work [31], though it took place only in a simulated environment, still utilizing domain insight.

To summarize, all previous approaches to visual servoing for reaching and grasping rely on calibration or the presence of domain knowledge. In contrast, the system we present in the following requires no information about the solution of the task, but instead acquires it completely from scratch.

III. Description of the Physical System

To illustrate our approach, we consider a combined reaching and grasping task in which a Kinova Jaco robotic arm is required to pick up a bowl located on a planar surface. The arm is expected to cover a working area of 35cm $\times$ 20cm $\times$ 20cm and reach the target from arbitrary positions within. The target itself is movable as well, spanning a range of 30cm along the area's long side, while being roughly centered along the short one.

The system includes two PlayStation Eye USB cameras arranged in the manner shown in Fig. 1, with one mounted on the table and one in the robot's gripper. Neither camera is calibrated, and the hand camera in particular does not point into the exact same direction that the actuator is oriented in, but is skewed to one side by more than 15°. Information about the object, is only available in the cameras' frame of reference. The object, which is distinctively colored from its background, is detected through simple color segmentation, with the centers of gravity of the resulting color regions being made available to the system. It is worth noting that the despite the high visual salience of the object against its background, which can be seen in Fig. 1, the position is still only approximate, as no features such as curvature are used to find the true center and the distance of the object, and changes in lighting condition may introduce errors. Furthermore, the object cannot simply be grasped centrally, but possesses two admissible zones along the rim that the agent will be required to find. This introduces a degree of insecurity with which the learner will be forced to cope.

Both the kinematic pose of the actuator and the robot's joint angles are also available to the system.

The robot can be controlled directly in Cartesian coordinates. Only its spatial position is to be adjusted here, with the actuator's orientation remaining static relative to the base joint's rotation. Finger movement is limited to the closing of the gripper at the system's signal. Consequently, the controller will be required to learn to position the actuator in such a way that this automated action results in a successful grasp closure.

While the task may appear trivial to a human observer, it is considerably less simple given the amount of information – or rather lack thereof – available to the system and the peculiarities of the setup. Without complete 3D information of the environment it is not possible to intuitively determine whether a sensor state would result in a successful grasp, and neither is the sensor configuration at admissible grasp

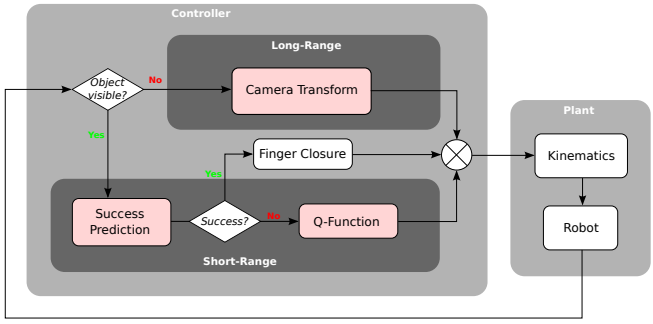

Fig. 2. Architecture of the combined system during control. Elements shaded light red represent modules generated by means of machine learning, i.e. neural controllers.

poses provided in advance. Such information can only be acquired through a lengthy test of moving the arm to a different location and checking whether the object moved along. In addition, no transition model, i.e. knowledge how choosing a certain action would affect the system, is given. As the actuator's orientation in task space and thus the relation between hand camera and object changes as the arm rotates, a simple directional proportional control strategy cannot be used either. The inability to either control the system by hand or design a simple controller prevents the use of imitation learning, and the system will have to acquire a strategy purely through exploration. This in turn increases the required amount of system interaction, thus requiring the use of a data-efficient design.

IV. Control Architecture

To solve the task in the face of these issues, we propose the following system design, illustrated in its entirety in Fig. 2. The system most notably includes two features: firstly, a split into two separate controllers that operate at different task scopes, and secondly the use of a mechanism to predict grasp success.

A. Controller Scope

Intuitively, to allow precise reaching and grasping across the robot's entire working area, both cameras would need to be used in conjunction. The hand camera has a limited field of view and may easily lose the target from sight during long movements. Meanwhile, the table camera, just like any single static camera, possesses a poor spatial discrimination along one direction of the working plane. Furthermore, it is subject to occlusion of the object by the arm itself.

Based on the general assumption that many tasks, with grasping being one of them, require particular precision in state regions close to the goal state, short reaching movements and the grasp itself will be performed by the *short-range* controller (SRC) using the hand-mounted camera, which activates once the object is visible in the latter. Guiding the actuator to such a state for any starting pose and for any target object position is the responsibility of the *long-range* controller (LRC), which utilizes the table-mounted camera for a wider view.

The SRC is to activate only if the object is visible in the hand-mounted camera. In addition, it should not be too close to the image border, as that region can be expected to be underrepresented in the training data and thus be difficult to acquire a stable policy for. Formally we define the SRC's *activation area* as any state that satisfies the condition $-0.9 \leq \iota_i^H \leq 0.9$, with (ι_x^H, ι_y^H) being the object position in the camera scaled to the interval $[-1; 1]$. The LRC will naturally be required to be capable of reaching such a state.

Due to the distinct scopes and requirements both controllers will be designed using different learning techniques and modeling. The focus of this work will lie largely on the SRC, which implements more precise control and will be acquired through reinforcement learning. In contrast, the LRC needs to perform only comparatively coarse movements, albeit over the entire working area, and will be implemented using supervised learning.

B. Success Prediction

As we do not provide the algorithm with a camera model, the target states for the visual servoing task, i.e. those in which closing the fingers should result in a successful pickup, are not known in advance. To avoid having to perform a pickup test after each training episode, we therefore introduce a further split within the SRC by means of the *success predictor*, which learns, using supervised learning, to anticipate grasp success based on the current visual information. The output of this module can then be used by the reinforcement learner to determine the outcome of a trajectory in lieu of an actual test. The prediction will be represented using a standard feed-forward neural network mapping object locations to success.

This bootstrapping approach offers the advantage of being able to both generalize over both sparse data and represent insecurities. Using a neural network as function approximator does not result in a mere binary prediction of success or failure, but a continuous one as illustrated in Fig. 3. Insecure grasp positions that do not guarantee success, such as the exact center of the image, result in intermediate network activations, which can be interpreted as low prediction confidences. By choosing an appropriate confidence threshold, one can implement a desired amount of "caution", and a high threshold will result in a system that only attempts grasps at points that guarantee success.

Learning this prediction separately from the controller's behavior can be expected to lead to an increased stability of the learning process. The late discovery of insecure locations during the training of the SRC would otherwise necessitate a re-learning of the policy by propagating the new information all the way from the target. In addition, by interpolating the success map from a limited number of observations, the required total system interaction time can potentially be reduced, since most likely one of the two tasks will be learned more quickly than the other, and we therefore avoid having to continue collecting data for a task that has already been learned completely.

V. Learning Methods

As the aforementioned system components will be created using several methods of machine learning, we will briefly introduce the specific techniques used in the following.

Reinforcement learning problems are commonly represented as a Markov Decision Problem (MDP), a 4-tuple $\{S, A, p, c\}$ consisting of a set of system states S, possible actions A, a state transition function $p(s, a) \rightarrow s'$ and a transition cost function $c(s, a, s') \rightarrow \mathbb{R}$. Notably, p is unknown in our case, forcing us to perform model-free reinforcement learning, and even c is not completely known; as mentioned in the previous section, the goal region S^+ is not explicitly specified.

Once we have formulated our task as an MDP, we can turn towards learning of the Q-function, which maps pairs of states and actions to an expected remaining trajectory cost. For discrete action sets, policies can be easily retrieved online from this function $Q(s, a) \rightarrow \mathbb{R}$ for a given state s by comparing the function values across possible actions and choosing the one minimizing it, i.e. $argmin_a Q(s, a)$. The task of learning the Q-function will be solved using Neural Fitted Q Iteration (NFQ) [15], a variant of Fitted Value Iteration [32] tailored specifically to using a multi-layer perceptron to approximate the Q-function. Since past experiences are explicitly stored and reused throughout the training process, the method performs in a particular stable and data-efficient manner. While NFQ is a well-established algorithm for reinforcement learning, the approach has thus far not been applied successfully for grasp control due to the difficulty of dealing with higher-dimensional actions coupled with strong interaction time constraints.

Both the Q-function used in NFQ and the ones in the two other constituents of our system are represented by feed-forward multi-layer perceptrons, which are trained in a supervised manner by providing pairs of input and desired output patterns. The output error is minimized through the application of backpropagation [33], which passes the error from the output layer toward the input layer. To update the local parameters of the network, the Rprop heuristic [34] is used, a variation of momentum-based gradient descent. Here, the momentum term is independent of the value of the gradient itself, and adjusted only on the basis of changes in its sign. This approach is known to lead to particularly fast convergence of the learning process.

VI. Short-Range Controller Training

The sub-task handled by the SRC most closely resembles a classical combined reaching and preshaping task. Being the focus of this work, its design will be described in particular detail.

A. Problem Modeling

The problem can be stated as a Markov Decision Process, which is then to be solved through reinforcement learning. Sensory information about the object is incorporated in the system state $s = \{\iota_x^H, \iota_y^H, \theta_0, \tau_z, v_x, v_y\}$.

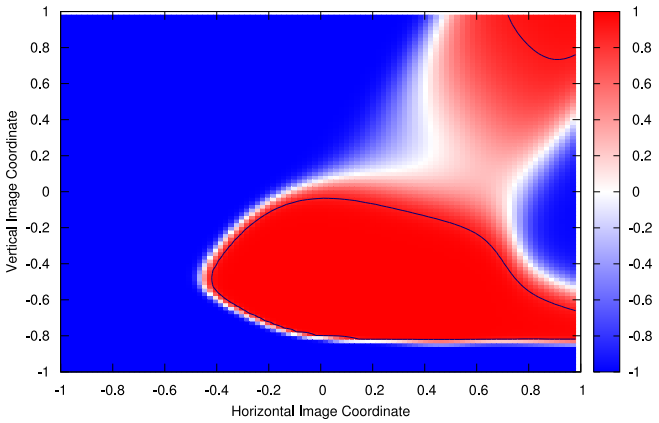

Fig. 3. Activation of the success prediction network given normalized camera coordinates as input. The contour line marks values above threshold, which would cause a training episode to be considered as successful during learning and trigger finger closure during execution. The two red regions correspond to the edges of the bowl; their shift from the center and asymmetry result from the highly skewed alignment of the camera within the robot's actuator, which is one of the challenges the system has to cope with.

The object is represented by the center of gravity of its projection in the camera, designated as (ι_x^H, ι_y^H), which is normalized in the interval $[-1; +1]$.

Information about the robot is provided in the form of the base joint's current rotation θ_0, the end-effector's height τ_z and the Cartesian velocities of the actuator, v_x and v_y. At an interval of $100ms$, the system can take a three-dimensional action $a \in \{-1, 0, 1\}^3$, which translates to the direction of movement in Cartesian space, performed by using the robot's inverse kinematics.

One integral part of the MDP, the cost function, is only partially specified, since it is based on the success prediction which is in turn implicitly represented in a neural network. The net is trained through supervised learning using patterns consisting of object positions (ι_x^H, ι_y^H) in the camera as input, and corresponding observed grasp test success as target output. It thus realizes a mapping $\sigma(\iota_x^H, \iota_y^H) \rightarrow [0; 1]$ which can be used together with a prediction threshold θ to define the cost function as:

$$c(s') = \begin{cases} 0 & \text{if } \sigma(\iota_x^H, \iota_y^H) \geq \theta \\ 0.01 & \text{otherwise} \end{cases}$$

θ is chosen to be 0.95, with the intent that insecurities should be treated as failures to ensure maximum robustness.

It is worth noting that task failure does not result in high costs, as one would usually assume. Instead, any observed transition to the failure state is replaced by a virtual one from the predecessor state to itself. By doing so, we avoid regions of extremely high costs in the Q-function, which due to their proximity to the target state could lower the perceived value of the goal region, and thereby hinder learning.

B. Success Prediction

Since the cost function, and thus the training of the reinforcement learner, requires the presence of the success

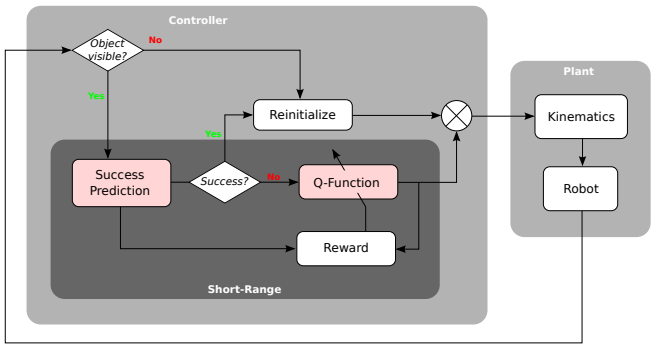

Fig. 4. System architecture during the SRC reinforcement learning stage. Neither LRC nor grasp modules are being used while the Q-function is being updated, though the success prediction module is employed for calculating the costs.

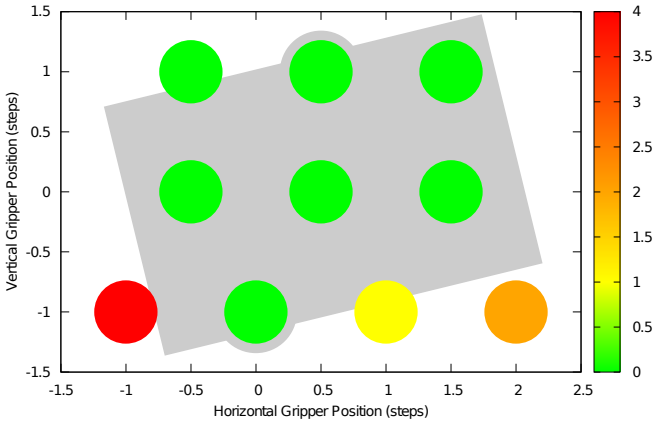

Fig. 5. Number of failed grasps for the chosen policy from different initial actuator positions relative to the target object. Positions within the shaded region lie in the SRC's activation area; again, asymmetry results from the camera's skewed alignment.

prediction module, the latter is generated first. It is represented by a feed-forward network, this time with one hidden layer of size 30. Prior to the reinforcement learning phase, it is trained on 189 pairs of visual information and success, which were acquired during earlier experiments. This training lasts for 1000 Rprop epochs, after which the test error converges. These samples, spanning most of the camera state space, yield the prediction mapping presented in Fig. 3.

C. Policy Acquisition

With the success prediction fixed, the resulting learning system is as depicted in Fig. 4, where the prediction is used to determine the costs.

The multi-layer perceptron used to approximate the system's Q-function is comprised of two hidden layers of size 20 each. During each episode, it is updated using Rprop for 300 epochs. These values were chosen based on past experiences, and have been proven to be well-suited for many different tasks [35]. While the number of epochs may not suffice for the network to converge to the true output values given the current observations, NFQ merely requires their magnitude to be correct relatively to each other, rather than absolutely.

During training, the starting position of the actuator is being varied randomly over an area of 35×20cm, while its height is fixed at 10cm above the table. The target itself is also placed at three different positions spaced 15cm apart, with each being used with equal frequency.

The system alternates between greedy episodes, where the best action returned by the current policy is used, and ones during which standard ϵ-greedy exploration is performed. We choose a high probability of $\epsilon = 0.2$ in order to ensure a sufficient number of exploratory actions even during the short total training duration that we expect. 300 such episodes are performed, each followed by an NFQ update.

While the algorithm used offers no theoretical guarantees that the algorithm will converge to an optimal solution, one can, given sufficient data, expect policies to fluctuate around one. This neccessitates an evaluation of the final policies regarding their performance to locate the best possible result. To allow the algorithm to stabilize as far as possible, an additional 1000 updates are performed without interacting with the robot, using only the transition data acquired thus far. Not all of these are tested, but only the last 10 resulting from the offline training stage. For each, grasp success is determined for 10 initial arm poses, with the object located centrally in the working area. Testing for a given policy is aborted once it failed to pick the object up from any starting pose.

D. Results

In total, training took up to 16 hours using a commodity single-core Pentium 4 setup, though only 35 minutes were needed for actual system interaction, and the remainder could be reduced simply by using a more powerful computer and employing parallelization. All policies were at least capable of grasping at all but one target position, suggesting that performance was fluctuating around the optimum rather than diverging.

Three of the policies were able to perform a successful grasp in every case, and only these policies were used in a second, extended testing phase. Here, the position of the target object was displaced to each side in by increasing amounts of up to 15cm, and each of the resulting targets was again tested for success from 10 relative robot poses.

Through this procedure, the policy with the highest reliability over both target and starting pose variation was chosen. As can be seen in Fig. 5, this policy managed to successfully pick the bowl up at all positions lying in its activation area. In fact, even those poses located further at the image border resulted in a successful grasp in most cases, though this was not required for the combined system.

VII. Long-Range Controller Training

To allow the SRC to be usable for every possible target object position, we must design the LRC in such a way as to reliably provide visibility of the object in the hand camera within its activation area. This can be achieved by

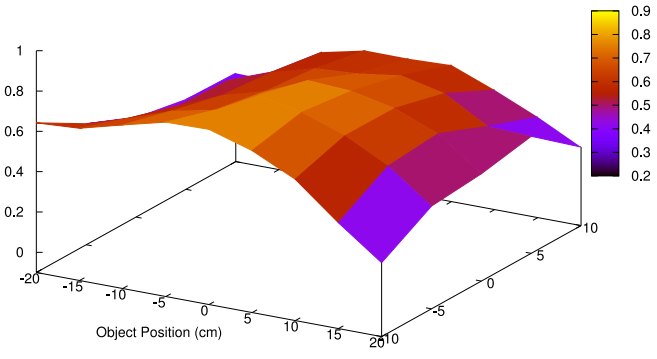

Fig. 6. Distance of the object in the hand camera from the image center as a function of object position in task space. The peak of the distribution lies at 0.8006, well within the range required for the SRC.

generating a camera transformation that generates rough desired tool center point positions (τ_x, τ_y) from the object's image coordinates (ι_x^T, ι_y^T). We use a multi-layer perceptron to approximate this mapping, as its generalization ability enables learning the relation from a limited number of samples. Since the generation of a target position requires only a forward pass of the current visual input through the network, it can easily be re-evaluated for every new frame provided by the camera, providing real-time control. The third degree of freedom, the height τ_z of the gripper, is fixed at a value of 10cm, which was the same height from which the SRC started during training. This pose can then be moved towards by using the robot's inverse kinematics.

A set of 758 pairs of camera positions and corresponding robot poses are generated by placing the object in the arm's gripper and moving it along the working area. We again choose a network with two hidden layers of 20 units, which is trained on this set for 300 iterations of Rprop gradient descent, after which the training error converges. This training stage takes less than a minute.

After training, the LRC is evaluated by placing the target at different positions in the robot's working area, waiting for the system to move to the pose generated by the network, and measuring the distance between the object position in the camera and the image center. To reiterate, for a position to lie within the SRC's activation area, it would have to fulfill the constraint that $|\iota_i^H| \leq 0.9$ for $i \in \{x, y\}$. Therefore we merely need to compute $\max_i |\iota_i^H|$ for each point and check if it lies below 0.9, i.e. in the inner 90% of the visual area.

As can be seen in the evaluation results illustrated in Fig. 6, this is clearly the case, with the distance never even exceeding 0.8. Consequently, the LRC is capable of generating admissible starting positions for the SRC for the system's entire working area.

VIII. Combined Performance

Our main goal, the system's ability to grasp a target in its working area, follows naturally from the combination of its constituents. The SRC was capable of performing a successful grasp from any initial position at which the object was well visible in the hand camera, while the LRC has been shown to ensure exactly this visibility. In this way grasping can be achieved from arbitrary starting positions in the robot's working range, regardless of the object's position and visibility in the hand camera. On average, performing one entire grasp sequence takes about 4.06 seconds from starting to move until closing the fingers around the object. This constitutes no significant difference to the 4.35 seconds needed by a human operator using a three-axis joystick and the same speed settings. In addition, the architecture gives rise to further traits, allowing the grasps to be performed adaptively and reliably.

A. Object Pursuit

Thus far all evaluations have been performed in a completely static environment. But since we chose to design our system specifically in a reactive feed-back manner to deal with dynamic settings, it is also capable of adapting to changes such as a moving object. To illustrate this ability, we used the setup of a sub-task from the ICRA 2012 Mobile Manipulation Challenge [36], in which the bowl to be grasped was placed on a turntable. By varying the distance of the object to the table's center we could effectively adjust the speed and movement range of the object.

The maximum radius we examined was 10cm, as it was the largest one possible to describe a path that lay within the system's 35cm$\times$20cm training area, outside of which its behavior could not be predicted. This amount, which corresponded to an object speed of 2.1cm/sec, still allowed reliable grasping by pursuing the object as illustrated in Fig. 7.

B. Robustness against Vision Errors

In addition, our system also attains robustness against various types of sensor errors.

The use of a short-range eye-in-hand controller enables it to cope with errors relating to the localization of the target object by the table-mounted camera. Despite the fairly robust color segmentation used here, large errors can be expected to result temporarily from occlusion of the object by the gripper itself, as well as smaller ones for longer times due to changes in lighting conditions. We therefore tested the LRC's ability to still reach the SRC's activation area even with an error applied to the camera projection (ι_x^T, ι_y^T). SRC activation and thus grasp success could still be maintained up to an error of 3.5° in any direction, which, given the size and distance of the bowl, corresponds to an occlusion of roughly one third of the object.

A second type of error is constituted by general camera noise, which leads to small fluctuations in the detected object projections. Such noise appears and disappears rapidly and is thus naturally compensated for by the reactive architecture, which prevents short-term errors from having a lasting effect.

IX. Discussion & Outlook

Our system has been shown to perform its intended task reliably, being able to pick up an object in its working range for arbitrary initial positions of arm or target. This

Fig. 7. An example grasp sequence performed by the final system on a moving object.

was possible even in the presence of vision errors or if the object was moving.

Given our goal of including only minimal amounts of task-related prior knowledge in our system, a brief summary of the information used is in order. For one, the split of the system into several components and the assignment of specific pieces of sensor information can be considered to require a degree of understanding of the problem solution. However, such an amount of modeling is ultimately unavoidable in any robotic system that does not use forgo representations entirely [37], and does not constitute any information about *how* the problem is to be solved.

While we do not use a model of the process, we do employ an inverse kinematics model to move the robot in Cartesian space, particularly in the case of the LRC. The reasons for this were primarily of technical nature, as controlling the robot in joint space would have disabled security features that avoid collisions with itself and the environment, which is obviously undesirable in a system intended to learn purely by exploring its action space. But even beyond that, we argue that since the reinforcement learning methods used do not make any assumption about how the actions generated by the controllers are interpreted, the approach should work just as well if controlling the robot in joint space. Ultimately, the overall amount of expert knowledge is still lower than in other contemporary systems, as it avoids aids such as a process model, object models, calibration, or demonstrations of the solution, and in the case of the SRC even of the desired goal.

As the LRC was secondary to the SRC, the implementation we used was comparatively simple, though sufficient to solve the task at hand. Future improvements could include training it in the same manner as the SRC through reinforcement learning with only a different goal state. By learning to adapt to factors such as the skew of the camera, it might contribute further to the robustness of the combined system.

Currently, the system's ability to pursue the object results purely from its reactive design. By including object movement in the training process, an implicit movement prediction could be acquired and even more robust strategies learned.

The limited depth discrimination ability of the LRC could be solved by replacing the single table-mounted camera with a stereo setup, which would increase the system's working area. While preliminary attempts show promise, the difficulty of processing three camera streams in real time precludes the use of such an arrangement at present.

References

[1] O. Krömer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1105–1116, 2010.

[2] M. Kazemi, J.-S. Valois, J. A. D. Bagnell, and N. Pollard, "Robust object grasping using force compliant motion primitives," in *Robotics: Science and Systems*, July 2012.

[3] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[4] J. Steffen, R. Haschke, and H. Ritter, "Experience-based and tactile-driven dynamic grasp control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2007, pp. 2938–2943.

[5] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng, "Reactive grasping using optical proximity sensors," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009, pp. 2098–2105.

[6] N. Ratliff, M. Zucker, J. A. D. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[7] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[8] K. Hashimoto, Ed., *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific, October 1993.

[9] J. Piater, S. Jodogne, R. Detry, D. Kraft, N. Krüger, O. Krömer, and J. Peters, "Learning visual representations for perception-action systems," *Int. J. Rob. Res.*, vol. 30, no. 3, pp. 294–307, March 2011.

[10] P. K. Allen, B. Yoshimi, and A. Timcenko, "Real-time visual servoing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1991, pp. 851–856.

[11] A. Namiki and M. Ishikawa, "Optimal grasping using visual and tactile feedback," in *IEEE/SICE/RSJ Inernational Conference on Multisensor Fusion and Integration for Intelligent Systems*, Whashington, DC, 1996, pp. 589–596.

[12] W. Hong and J.-J. E. Slotine, "Experiments in hand-eye coordination using active vision," in *4th International Symposium on Experimental Robotics*. London, UK: Springer-Verlag, 1997, pp. 130–139.

[13] F. Chaumette and S. Hutchinson, "Visual servo control, part I: Basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. 4, pp. 82–90, December 2006.

[14] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[15] M. Riedmiller, "Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method," in *16th European Conference on Machine Learning*. Springer, 2005, pp. 317–328.

[16] R. Hafner and M. Riedmiller, "Neural reinforcement learning controllers for a real robot application," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 2098–2103.

[17] T. C. Kietzmann and M. Riedmiller, "The neuro slot car racer: Reinforcement learning in a real world setting," in *IEEE International Conference on Machine Learning and Applications (ICMLA)*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 311–316.

[18] B. E. Inria and B. Espiau, "Effect of camera calibration errors on visual servoing in robotics," in *3rd International Symposium on Experimental Robotics*, 1993, pp. 187–193.

[19] J. A. Piepmeier, G. V. McMurray, and H. Lipkin, "Uncalibrated dynamic visual servoing," *IEEE Transactions on Robotics*, vol. 20, no. 1, pp. 143–147, 2004.

[20] A. Shademan, A.-m. Farahmand, and M. Jäagersand, "Robust Jacobian estimation for uncalibrated visual servoing," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 5564–5569.

[21] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *International Conference on Robotics and Automation (ICRA)*, 2011.

[22] N. Ratliff, J. A. D. Bagnell, and S. Srinivasa, "Imitation learning for locomotion and manipulation," in *IEEE-RAS International Conference on Humanoid Robots*, November 2007.

[23] F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning motion primitive goals for robust manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2011, pp. 325–331.

[24] D.-J. Kim, R. Lovelett, and A. Behal, "Eye-in-hand stereo visual servoing of an assistive robot arm in unstructured environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009, pp. 2326–2331.

[25] M. Riedmiller, "High quality thermostat control by reinforcement learning – a case study," in *Conald Workshop*, Carnegie-Mellon-University, 1998.

[26] R. Detry, D. Kraft, O. Krömer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater, "Learning grasp affordance densities," *Paladyn Journal of Behavioral Robotics*, no. 2(1), pp. 1–17, 2011, intelligent Autonomous Systems.

[27] A. Morales, P. J. Sanz, A. P. del Pobil, and A. H. Fagg, "Vision-based three-finger grasp synthesis constrained by hand geometry," *Robotics and Autonomous Systems*, vol. 54, no. 6, pp. 496–512, March 2006.

[28] D. Aarno, J. Sommerfeld, D. Kragic, S. Kalkan, F. Wörgötter, N. Pugeault, D. Kraft, and N. Krüger, "Early reactive grasping with second order 3D feature relations," in *IEEE International Conference on Advanced Robotics*, 2007.

[29] P. Kulchenko and E. Todorov, "First-exit model predictive control of fast discontinuous dynamics: Application to ball bouncing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[30] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Computers and Chemical Engineering*, vol. 23, pp. 667–682, 1997.

[31] P. Kulchenko and E. Todorov, "Policy gradient methods with model predictive control applied to ball bouncing," in *IEEE Adaptive Dynamic Programming and Reinforcement Learning*, 2011.

[32] G. Gordon, "Stable function approximation in dynamic programming," in *Proceedings of the 12th International Conference on Machine Learning (ICML)*, A. Prieditis and S. Russell, Eds., San Francisco, CA, 1995.

[33] D. E. Rumelhart and J. McClelland, *Parallel Distributed Processing*, 1986.

[34] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *IEEE International Conference on Neural Networks*, H. Ruspini, Ed., vol. 1, San Francisco, CA, 1993, pp. 586–591.

[35] M. Riedmiller, "10 steps and some tricks to set up neural reinforcement controllers," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Montavon, G. Orr, and K.-R. Müller, Eds., 2012, vol. 7700.

[36] The ICRA 2012 mobile manipulation challenge. [Online]. Available: http://mobilemanipulationchallenge.org/ [Accessed: 09/11/2012].

[37] R. A. Brooks, "Intelligence without reason," in *12th International Joint Conference on Artificial Intelligence*, R. Myopoulos, John; Reiter, Ed. Sydney, Australia: Morgan Kaufmann, August 1991, pp. 569–595.