Online Learning of Vision-Based Robot Control during Autonomous Operation

Kristoffer Öfjäll and Michael Felsberg

Linköping University Post Print

N.B.: When citing this work, cite the original article.

Original Publication:

Kristoffer Öfjäll and Michael Felsberg, Online Learning of Vision-Based Robot Control during Autonomous Operation, 2015, New Development in Robot Vision, [ed] Yu Sun, Aman Behal and Chi-Kit Ronald Chung, Springer Berlin/Heidelberg, 137-156.

http://dx.doi.org/10.1007/978-3-662-43859-6_8

Copyright: Springer-Verlag Berlin Heidelberg

<http://link.springer.com/>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-110891>

Online Learning of Vision-Based Robot Control during Autonomous Operation

Kristoffer Öfjäll and Michael Felsberg

Abstract Online learning of vision-based robot control requires appropriate activation strategies during operation. In this chapter we present such a learning approach with applications to two areas of vision-based robot control. In the first setting, self-evaluation is possible for the learning system and the system autonomously switches to learning mode for producing the necessary training data by exploration. The other application is in a setting where external information is required for determining the correctness of an action. Therefore, an operator provides training data when required, leading to an automatic mode switch to online learning from demonstration. In experiments for the first setting, the system is able to autonomously learn the inverse kinematics of a robotic arm. We propose improvements producing more informative training data compared to random exploration. This reduces training time and limits learning to regions where the learnt mapping is used. The learnt region is extended autonomously on demand. In experiments for the second setting, we present an autonomous driving system learning a mapping from visual input to control signals, which is trained by manually steering the robot. After the initial training period, the system seamlessly continues autonomously. Manual control can be taken back at any time for providing additional training.

1 Introduction

Perception-action learning for robotic systems is a challenging problem. Preferably, both learning and autonomous operation as well as switching of operational mode should be online such that new training data can be provided if and when previous training data is insufficient. Switching to training mode, obtaining training data and

Kristoffer Öfjäll

Computer Vision Laboratory, Linköping University, e-mail: kristoffer.ofjall@liu.se

Michael Felsberg

Computer Vision Laboratory, Linköping University e-mail: michael.felsberg@liu.se

finally switching back to autonomous mode should be possible to perform without taking the system out of operation.

There are two primary classes of problems, one class where sufficient information for self-evaluation is present in the learning scenario and one complementary class where it is not, that is, additional information is needed to determine the correctness of an action. In the first class, the system is able to observe and assess the success of its actions itself, without external intervention; in the second class, an external supervisor (typically human) provide the system with feedback on its performance. Online learning in the two classes are similar with respect to mode switching and request of new training data, the difference lies in what entity initiates the mode switch and from where training data is provided.

One typical problem from the first class is *reaching*, where the system has received a stimuli somewhere in the visual field and tries to position a manipulator at the position of the stimuli. An example from the second class is *autonomous driving*, where typical driving behaviors such as staying on the roads are not inherently available in the learning situation but have to be acquired from external sources. In the application presented here, not even the notion of *a road* is present in the system initially. The two learning concepts are related to two major modes of learning in biological systems: random exploration and mirroring. Problems in the first class are typically solved by random exploration such as small children randomly moving their arms, first for identification of the visual stimuli generated by arm movements and later for learning of the *inverse kinematics*: how desired positions in the visual field can be reached.

This chapter presents a perception-action learning system for robotics where the training data is either generated by modulated motor babbling (random exploration) or by demonstration. The proposed learning system is incremental and real-time capable during both learning and autonomous operation.

In experiments, the proposed method shows its capabilities of learning from exploration (inverse kinematics) and learning from demonstration (autonomous driving). For learning of the inverse kinematics of a robotic arm, training data may be self-generated by random exploration. However, as is shown in the experiments, an exploration scheme biased towards minimizing the pose error provides for faster convergence to previously unknown poses and produces training data more relevant for the problem at hand. For the autonomous driving experiment, the system is not given a-priori knowledge on the type of visual event it should expect or the driving rules it should follow, but rather learns them from a human. After manually piloting the robot 1.5 laps around a track, the controls are released and the robot successfully continues around the track. The system demonstrates ability to generalize to changes of the track. Correction of unwanted behavior is demonstrated in the experiments by reclaiming manual control during a fraction of a lap. Switching between autonomous driving and demonstration is performed without stopping the robot. A video demonstrating the applications in this chapter is available.¹

¹ <http://users.isy.liu.se/cvl/ofjall/onlinePerceptionActionLearningSmaller.mp4>

2 Previous Work

The general Perception-Action learning field is far too wide for a comprehensive presentation in the current scope. Here we concentrate on the two application examples presented in the introduction and provide some references to learning approaches previously applied to problems regarding inverse kinematics and autonomous navigation.

With online learning methods, the robot model can adapt to changes while the system is in operation. This has shown to be useful, especially in cases where the controlled system tends to change properties within short time frames such as in [10]. Online learning enables immediate switching between learning mode and autonomous mode. All experience gathered during a training session is directly available for use when switching back to autonomous mode. In contrast, for offline learning systems, the system has to be taken out of service while processing training data.

First, the inverse kinematics problem is briefly presented (Sect. 2.1), which is used as an application for demonstration of learning from exploration (Sect. 2.2). Learning from demonstration is applied on a visual autonomous navigation (Sect. 2.3) task. Finally, two components used in the proposed method are presented, locally weighted projection regression (Sect. 2.4) and Levenberg-Marquardt minimization (Sect. 2.5).

2.1 Inverse Kinematics

The inverse kinematics of a robotic arm is a mapping from the desired *pose* (position and orientation) of the end effector to a set of angles for the joints of the robot (the *joint configuration*) which makes the end effector attain the desired pose. The classical approach uses handcrafted geometric models of the robotic arms [32]. The accuracy depends on the complexity of the model and any changes of the robot behavior due to wear are disregarded. Control models based on learning systems have the possibility to adapt to individual differences between robots of the same manufacturer and type as well as being able to learn different robot setups. By retraining the system, also changes in the robot can be handled. Different variations of neural networks have been popular approaches to this problem [21]. The results have been improved by using modular neural networks [17, 18], where several different neural networks are trained and the output from the locally most suitable network is used.

The neural network approaches mostly use offline training and require training points in the order of millions. Depending on the physical layout of the robotic arm, the mapping can be decomposed into one mapping from orientation of the end effector to a subset of the joints and another mapping from position to the remaining joints [1]. Any changes of the robot in these examples still require offline retraining.

2.2 Active Learning and Exploration

Active learning generally means that the learning system can affect the generation or selection of training data, the term is not clearly defined but [29] provides a survey. Active learning is mostly interesting in the case where training data is generated by the system itself, not by an external teacher. Here, this applies to the inverse kinematics learning application where the system has the possibility to actively control the robotic arm. This can be used to generate training data that maximizes information gain given the current state of the learnt model.

One fundamental requirement of online self-learning is the availability to the system of an estimate of its error immediately after an action. In the case of vision-based robotic manipulation, the actual pose achieved by the system can be inferred from the visual sensor data and compared with the desired pose to estimate the error in the last action of the system. This visual feedback is fundamental in visual servoing and in combination with online kinematics learning this enables continuous improvement of the kinematics model.

In cases where the current learnt model does not provide any clue regarding reducing the pose error, an exploration strategy is required for training data generation. Motor babbling is a popular approach where random motions around the current configuration are carried out. In some cases this would provide information on how to proceed towards the desired pose, but as described in [27], *falling down might not tell us much about the forces needed in walking*. That is, motor babbling may generate training data that does not affect the predictions of the learnt model.

Approaches which address this issue to some extent are proposed by [25, 3], where goals are used to direct the babbling and exploration of the available motion space. The work by [4] use a statistical foundation for presenting a more general and theoretical framework for this type of active learning.

When the space of possible movements is large, exploring the whole space can be very time consuming and depending on the application, only a small subset of these movements may be used. The Shifting Setpoint Algorithm [26] is a method where models are built along tubes in the motion space between desired points. As the name suggests, a setpoint is shifted towards the desired point and motor babbling is carried out around it. When the model is good enough locally (as determined by the algorithm), the setpoint is shifted again. This solves a similar problem but robot movement iterations and time are unnecessarily spent generating accurate models in between desired poses where a coarser model would suffice.

2.3 Visual Autonomous Navigation

One of the earliest successful learning approaches to vision based autonomous navigation systems was ALVINN (Autonomous Land Vehicle In a Neural Network) [19]. Like the work presented in this chapter, ALVINN learns how to control a vehicle by observing a human driving. The learning is based on a single hidden layer

back-propagation network. A more recent work from LeCun et al. focuses on learning vision based obstacle avoidance for off-road robots [11]. The learning algorithm in [11] requires very large collections of data and is based on a large 6-layer convolutional network. The system learns features that predict traversable areas. The convolutional network approach has also been used to create a track-following robot [28]. All these examples use offline learning, where training data has to be processed for hours or days before the system can operate autonomously.

Here an autonomous driving challenge is used as an example application where correct behavior is not apparent from the problem itself. The problem of autonomous navigation has been approached in a number of different ways in the literature, which can be roughly divided between the classical control-based approaches [6, 35, 34] and the learning-based approaches [19, 20, 23] (we refer to [12] for an in-depth review). The approach used in this chapter belongs to the second category, where perception is visual and the correct action is demonstrated by manual control of the robot, *learning from demonstration*.

The visual perception consists of a generic, holistic representation of the whole visual field, using so-called Visual Gist [16]. Visual Gist provides a compact and generic approach to image description, and has been used successfully for scene identification [24], image search [7], indoor vs. outdoor detection [30, 31], road type detection [9] and driver action prediction [22].

2.4 Locally Weighted Projection Regression

Locally Weighted Projection Regression [36], LWPR, has successfully been applied to learning problems related to robotics applications [2, 26, 8, 15]. The general idea is to use the output from several local linear models, distributed in the input space, weighted together to form the output.

The output y_{dk} for each local model k for dimension d consists of r_k linear regressors

$$y_{dk} = \beta_{dk}^0 + \sum_{i=1}^{r_k} \beta_{dki} \mathbf{u}_{dki}^T (\mathbf{x}_{dki} - \mathbf{x}_{dk}^0) \quad (1)$$

along different directions $\mathbf{u}_{dki}$ in the input space. Each projection direction and corresponding regression parameter β_{dki} and bias β_{dk}^0 are adjusted online to fit the training examples. Variations in the input explained by each regression i is removed from the input $\mathbf{x}$ generating the input to the next regressor $\mathbf{x}_{dk(i+1)}$.

The total prediction $\hat{y}_d$ in one output dimension d

$$\hat{y}_d = \frac{\sum_{k=1}^K w_{dk} y_{dk}}{\sum_{k=1}^K w_{dk}} \quad (2)$$

depends on the distance from the center $\mathbf{c}_{dk}$ of each of the local models. Normally a Gaussian kernel is used, generating the weights

$$w_{dk} = \exp \left(-\frac{1}{2} (\mathbf{x} - \mathbf{c}_{dk})^T \mathbf{D}_{dk} (\mathbf{x} - \mathbf{c}_{dk}) \right) \quad (3)$$

where the metric $\mathbf{D}_{dk}$ is updated while the model centers $\mathbf{c}_{dk}$ remain constant. Even though the regression is performed in a low dimensional projected space, the local models still live in the full input space. An advantage of LWPR which is useful for self-learning, is the possibility to derive analytic Jacobians of the learnt model [13].

2.5 Numerical Optimization

For inverse kinematics, minimizing a suitable distance between the current and desired poses will generate a solution to the given problem. There are many different algorithms available from the field of optimization [33]. Especially numerical optimization methods geared towards reaching a minimum using as few iterations as possible will avoid unnecessary movements of the robot.

If the kinematics of the robotic arm is not explicitly known, a numerical method has to be used. This implies iterative or grid based methods, where the former is expected to require fewer movements of the arm. One popular numerical algorithm is the Levenberg-Marquardt method [14]. For fast convergence, the algorithm requires a good initial solution and the derivative of the distance function at the iteration points. One possibility is to use the current pose of the robot as initial solution and estimating the derivatives using finite differences. As will be shown (Sect. 3.1), there are more efficient (in terms of time to find a solution) ways of choosing the initial solution as well as obtaining derivatives.

In Gauss-Newton optimization, the update $\mathbf{q}_\Delta$ in each step is obtained by solving a linearized problem

$$\mathbf{J}^T \mathbf{J} \mathbf{q}_\Delta = -\mathbf{J}^T \mathbf{e} \quad (4)$$

where $\mathbf{J}$ is the Jacobian at the current point and $\mathbf{e}$ is the current residual vector. In LM, the joint space update $\mathbf{q}_\Delta$ in each step is obtained by solving

$$(\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J})) \mathbf{q}_\Delta = \mathbf{J}^T (\mathbf{x}_d - \mathbf{x}(\mathbf{q})) \quad (5)$$

where $\mathbf{J}$ is the Jacobian at the current configuration $\mathbf{q}$, $\mathbf{x}(\mathbf{q})$ is the pose of the current configuration and $\mathbf{x}_d$ is the desired pose. The method is a weighted (by λ) combination of Gauss-Newton and weighted gradient descent.

3 Proposed Method

We provide an online learning system with applications to inverse kinematics (Sect. 3.1) and autonomous navigation (Sect. 3.2). These examples represent the self learning case and the case where external information is required. In the first

case, the mode switch is initiated by a pose error larger than a fixed accuracy threshold and training data is provided by numerically minimizing the pose error. In the second case, a mode switch is initiated manually by activating manual control of the robot. While under manual control, the learning system is trained.

3.1 Learning Inverse Kinematics by Exploration

The inverse kinematics represent the system’s knowledge of which control signals are required to reach any desired state. In the case of a robot reaching for an object, this means predicting joint angles for attaining a specified *desired pose* of the end effector.

If the system can obtain an estimate of the current pose of the end effector, it can also estimate performance (pose error) autonomously by comparing actual and desired poses—assuming a distance measure in pose space. This provides for self-learning by exploration.

We propose that by combining an online learning approach with a numerical optimization method, the desired pose is reached quicker on average. The current learnt inverse kinematics model can be used to provide an initial solution as well as estimated Jacobians as required by numerical optimization methods. In return, the iterations of the numerical method serves as excellent training data along the lines of active learning.

In this chapter, a combination of Locally weighted projection regression, LWPR, and Levenberg-Marquardt, LM, is evaluated. The proposed integration of both methods is compared with a naïve combination of the two algorithms. In the baseline implementation, only outputs from the two algorithms are used. For the integrated method, internal data is shared.

Initial Solution

In the LWPR algorithm, the weight of each local model (3), is compared to a threshold w_{cut} to determine which local models should be used when predicting the output of a given input. This can also be used to determine where reasonable predictions can be expected. We propose using this information for determining an initial solution. Given the weight of each local model (3), using the notation of (3), reasonable predictions can be expected within the set

$$X_p = \bigcap_d \left(\bigcup_k \{ \mathbf{x} : (\mathbf{x} - \mathbf{c}_{dk})^T \mathbf{D}_{dk} (\mathbf{x} - \mathbf{c}_{dk}) \leq -\ln w_{cut} \} \right) \quad (6)$$

as at least one local model k in each output dimension d should provide a useful prediction. Given a desired pose $\mathbf{x}_d$ it is thus possible to find an initial pose $\mathbf{x}_i \in X_p$ such that no other pose $\mathbf{x} \in X_p$ is closer to the desired pose.

An approximation of the optimal initial pose $\mathbf{x}_i$ still within X_p can easily be found by starting in the desired pose $\mathbf{x}_d$ and using gradient ascent on

$$\min_d \left(\max_k \exp \left(-(\mathbf{x} - \mathbf{c}_{dk})^T \mathbf{D}_{dk} (\mathbf{x} - \mathbf{c}_{dk}) \right) \right) \quad (7)$$

to improve the dimension with the worst response of the best fitting model. The ascent is continued until the value of the expression reaches w_{cut} where the learnt inverse kinematics model is expected to be accurate enough. The model can then be used to move the arm directly close to this pose. As the initial pose may be on the border of X_p (the set of poses where inverse kinematic predictions can be made) the accuracy of the inverse kinematics model can not be expected to be perfect.

If the robotic arm has previously moved close to the desired pose, the initial pose will be very close to the desired pose. In areas where the inverse kinematics model is accurate, the desired pose will be reached directly.

3.2 Learning Autonomous Driving from Demonstration

For the autonomous navigation application, a mobile robot is supposed to navigate along a track. However, the problem specification does not specify what kind of track or what kind of visual features that defines the track. This is to be learnt by the system along with appropriate steering actions for staying on the track. Since the track appearance or layout is not known beforehand, the learning system can not evaluate its own performance.

We propose an online learning system which is trained by demonstrating correct behavior (manually driving the robot) and which is supposed to be able to follow the track immediately and autonomously after the initial training sequence. Specifically, assuming consistency in track markings, we want to learn a mapping $f : \Phi \rightarrow \Theta_{\text{turn}}$ (where Φ denotes the visual features extracted from images P and Θ_{turn} the steering angles) that will keep the vehicle driving on the track indefinitely, for any track layout. Our aim is then to learn f from N example pairs $(\phi^i, \theta_{\text{turn}}^i)_{i=1}^N$, sampled from a demonstration by a human driving around the track.

The chosen approach has the following characteristics: (i) the steering is estimated directly from the visual input, frame by frame and is therefore not affected by previous errors; (ii) the system does not have or build a model of the track, allowing for navigation on potentially infinite paths; (iii) the system's visual input encompasses the whole visual information, and it is the learning that specifies which aspects of the visual scene are relevant for steering control; (iv) it learns and runs real-time; and (v) any misbehavior or lack of training data can be corrected by additional manual control without stopping the system.

The system's visual perception consists of a generic, holistic representation of the whole visual field, using so-called visual Gist [16]. This encodes the whole visual field into one feature vector, using multiscale filtering to encode the visual

scene. There exists different versions of the Gist features. In this work, we compute the image Gist by filtering a downscaled (128×128 pixels) version of the image P with Gabor filters $G_{\lambda, \theta}(P)$ tuned at different scales (λ_i) and orientations (θ_i), and averaging the filters' responses over Gaussian channels $C_{i,j}$ regularly spaced over the image. We use 4 scales, 8 orientations and 8×8 channels, resulting in a feature vector of dimension 2048.

In this case, LWPR is unable to handle an input space of dimension 2048 as every local model contains the center of the local model and a matrix representing the size of the local in the input space. The input space dimension is reduced to 256 by means of PCA where the principal components are obtained from the Gist feature vectors from manually controlling the robot one lap clockwise and one lap counter clockwise on the track in Fig. 7. Note that the same principal components calculated from the indoor track also is used for the outdoor forest road in Fig. 11.

4 Evaluation

The proposed systems, learning from exploration or from demonstration are evaluated. Evaluation is performed both in controlled environments and under realistic conditions.

4.1 Learning from Exploration

The explorative approach of obtaining training data is evaluated in a reaching scenario with a robotic manipulator. The evaluated system is sequentially presented with desired poses of the end effector and the task is to set the joint angles of the arm such that the desired pose is obtained. The evaluated system obtains measurements of the current pose of the end effector. For better accuracy in performance comparisons and as common in literature [5, 18, 13, 33], the presented methods are primarily evaluated on a simulated, planar robotic arm. The learning system has also been used to control a 7-DoF KUKA LWR, however it was not possible to obtain objective performance data from the latter experiment.

The simulated arm has three joints and the desired pose is the two dimensional position of the tip of the last link. The evaluated method controls the arm by specifying the three joint angles and the simulation returns the position of the final link with added Gaussian noise. The pose error $\mathbf{p}_e = \mathbf{p}_d - \mathbf{p}_c$ is used in the quadratic error function of the Levenberg-Marquardt algorithm (lsqnonlin implementation in Matlab). For LWPR, the implementation by Klanke, Vijayakumar and Schaal is used [36].

An initial training set with 15 points distributed in a small part of the reachable space of the robot is generated. Evaluation points are generated in four clusters with 50 points in each cluster. The training and evaluation points are shown in Fig. 1.

Fig. 1 (Left) The simulated robotic arm reaching for the red diamond marker. The blue circle indicates the initial position, the blue crosses are the iterations and the green crosses are the initial training data. (Right) The evaluation points marked with red crosses.

For evaluation, each LWPR model is trained using the training points. Each method is then used to move to the first evaluation point in the first cluster and the number of iterations required to reach the point is counted. This is repeated for the first point in the second cluster and so on until all evaluation points are reached.

When the first evaluation point in a cluster far from the training data is to be reached, the system is expected to require some iterations to move from the closest previously visited area to the desired position. When the arm is supposed to return to the second point within the same cluster, the number of iterations required is expected to be lower. After visiting a few points within the cluster, the system is expected to be able to move directly to the desired pose.

This behavior is expected both for systems using numerically estimated Jacobians (naïve) and for systems using Jacobians from the learnt model (integrated). For the Jacobians from the learnt model, the accuracy is expected to be low during the first runs but increasing as more points are visited within each cluster. For the numerically estimated Jacobians, the arm has to be moved for Jacobian estimation. Thus, if the system does not move the arm to the correct position in the first iteration, at least four additional movements of the arm are required.

In contrast to the learning approaches, using a numerical optimization method alone, always starting at the current arm pose, the number of iterations required to reach each point could be expected to stay constant as no information regarding the behavior of the robot is kept.

Fig. 2 Quartiles of the number of iterations required to reach each of the last 100 evaluation points for each evaluation session.

Fig. 3 Mean number of iterations required to reach each evaluation point. (Left) Integrated method; (Right) Naïve combination.

Fig. 4 Mean number of iterations required to reach each evaluation point with reduced measurement noise. (Left) Integrated method; (Right) Naïve combination.

Fig. 5 Mean number of iterations required to reach each evaluation point with increased measurement noise. (Left) Integrated method; (Right) Naïve combination.

4.1.1 Results

The evaluation results are shown in Fig. 3. The graphs show the number of iterations required to reach each test point for the naïve combination and the proposed integrated method respectively. We expect real measurements to be noisy so independent zero mean Gaussian noise with standard deviation 0.03 was added to the position estimates. This corresponds to 0.2% of the diameter of the space reachable by the robot. The shown results are the average of ten runs. As expected, the number of iterations required to reach a point decays with increasing number of visited points in the same cluster.

If the Jacobians from the learnt model were perfect, the method using these should require about one fourth of the iterations required by the naïve method. Using Jacobians from the learnt model requires significantly fewer iterations than using numerically estimated Jacobians, but the theoretically possible reduction is not achieved. This is due to errors in the learnt model and to the implementation of the numerical solver not needing to estimate the Jacobian at every step.

Additionally, to assess the effect of the noise, simulations were run with reduced and increased noise variance. Reducing the standard deviation to one sixth of the original noise, the results in Fig. 4 are obtained. In Fig. 5 the standard deviation of the noise is doubled.

In the case of reduced noise, both methods perform better. The integrated method is better than using numerically estimated Jacobians. Increasing the noise, the numerical method performs significantly worse. Here, the averaging introduced by the learning method is a great advantage. In Fig. 2 evaluation results for the last 100 points in each session are presented more compactly. For each noise level, t-tests on this data indicate significant improvements ($p_{H_0} < 0.01$) where H_0 : Equal mean error for naïve and integrated methods. One example of iterations for reaching a desired pose is shown in Fig. 1.

4.1.2 Learning Inverse Kinematics on the KUKA Robot

The proposed learning system is implemented on a 7 DoF KUKA LWR robotic arm. With this system, the 6 DoF pose of the cameras and tools can be controlled directly instead of controlling the joint angles of the robot. The online self learning algorithm enables increasing pose precision over time and self-recalibration after hardware changes or maintenance. The inverse kinematics solver shipped with the arm is based on a geometric model and has issues with singularities. This poses a problem in the intended active vision applications where all desired poses are not known in advance and thus can not be checked for singularities before deploying the system.

In Fig. 6, a desired pose for a general view is selected. The desired pose is outside the region of known inverse kinematics solutions and the pose is reached by numerical minimization of the pose error. Intermediate poses provide training data. The figure shows time-equidistant frames from a video. Note that the desired pose is close to a singularity, as the desired pose is approached the rotational axes of the fifth and the seventh joints (counted from the base) are close to parallel. This configuration would not be well handled by the inverse kinematics solver shipped with the arm.

4.2 Learning from Demonstration

For evaluation of the learning from demonstration capabilities, a mobile robot is used based on a radio controlled car fitted with a laptop and a single grayscale camera. The laptop interfaces to the servo control on the car, so that controls generated by either the teacher (during training) or the software are actuated. The autonomous navigation module generates control signals from visual input from the camera. There is also a low-resolution color camera on the platform, however it is not used in the experiments.

Fig. 6 Explorative learning of inverse kinematics for reaching on a 7 DoF robotic arm.

This work uses teleoperation to gather the examples, i.e. the robot is operated by a teacher while recording both the control signals and the sensor readings. The teacher controls the car with a standard remote control transmitter. Training examples are processed online onboard the mobile robot. When the robot is able to generate control signal predictions from the visual input, the teacher is notified via an indication on the robot display and the remote controller can be released. The robot control can be manually overridden at any time for providing additional training examples. In the experiments, steering is predicted by the system while throttle is kept at a constant level.

The evaluation environment is a reconfigurable circuit made of tiles of carpet with sections of road markings. Different circuits can be made by placing tiles in the desired configuration, see fig. 9 for example circuit configurations. To evaluate the quality of a trajectory of the robot around the circuit, the robot is tracked. Very accurate tracking trajectories were obtained by tracking a red marker attached to the robot. Fig. 9(e) show results of this red marker tracking (the dashed lines). An homography is computed (by manually marking corresponding positions) from the position of the red marker to the ground plane, the result being the solid lines. Finally a second homography is computed to project both the images and the trajectories into a planar view, see fig. 9(d) for the projection of Fig. 9(e). Since the red marker is not visible while behind the red beam, trajectories are linearly interpolated in that region.

4.2.1 Autonomous Navigation Performance

Fig. 7 shows the trajectory for the first five laps of a previously untrained system. The robot is manually controlled through six corners (dashed blue line) whereafter the controls are released and the robot continues autonomously around the track (solid magenta line). As the trajectory in the lower part of the track was not considered accurate enough, manual control was used to override steering predictions through two corners (dashed black line). The new training data corrects the undesired behavior and during the following laps, the robot stays on the road (solid cyan line).

After the five first clockwise laps, the robot is manually turned around for counter clockwise laps. These are shown in Fig. 8. The robot is manually controlled through three corners whereafter the robot is able to generalize and make the forth turn autonomously. Steering control signals are plotted in Fig. 10. The switches to manual control for initial training, path correction and training for counter clockwise driving are clearly visible.

The robot continues to run autonomously while the track is changed, Fig 9. Road side markers are changed (Figs. 9(a), 9(b)) and the robot generalizes, that is it successfully continues to navigate the track without additional training. Similarly, when restoring the initial road sides and changing the road itself, the robot is still able to follow the track with a tendency to cut the upper left corner compared to the autonomous trajectories from the unaltered track (Figs. 9(c), 9(d)). This behavior of

Fig. 7 Trace of first five clock-wise laps for a previously untrained system overlaid on an image of the track reprojected onto the ground coordinate system, with coordinates in meters.

cutting corners was introduced in the training data for counter clockwise driving (dashed blue line in Fig. 8).

Additionally, the robot is evaluated in an outdoor environment, Fig. 11. In this environment, tracking is unavailable however the robot successfully follows the road. Note that no changes has been made to the robot for this evaluation, neither in software nor hardware. The robot is manually driven for approximately two minutes whereafter it drives autonomously.

5 Conclusions

We have presented an online perception-action learning system capable of immediate and online switching between autonomous and learning mode. Two applications have been presented, one where the learning system is provided with enough information for self-evaluation and self-learning (inverse kinematics using explorative

Fig. 8 Trace of the four laps following Fig. 7 where the robot runs counter clockwise.

learning) and one where external information is required for determining the correct action (autonomous driving using learning from demonstration).

For learning of inverse kinematics, experiments have shown (i) that self-learning of inverse kinematics is possible using minimization of pose error as the explorative component, and (ii), that learning improves if utilizing information available from the already learnt model when performing exploration. The main experiments have been performed on a simulated robotic arm, however the proposed system has been able to perform successfully on a real 7 DoF robotic arm and has been implemented and used in the GARNICS demonstrator system.

The system has also shown its capability of learning a mapping from visual percepts to steering actions on a mobile robot platform. In this case, the correct action (depending on factors such as road type and desired driving style) is not available from the problem specification but is provided by means of demonstration. The same system has shown ability to learn to drive on different types of roads in different environments (indoor/outdoor).

For the indoor track, 1.5 laps of demonstration is sufficient for autonomous operation. After turning the robot around, additional demonstration during three quarters

(e) Part 10, original view with red marker trajectory (dashed) and the corresponding ground projection (solid).

Fig. 9 Autonomous trajectories by the robot while reconfiguring the track (cyan), with last laps on unmodified track for comparison (magenta).

Fig. 10 Steering control during initial sequence, -1 is max right, 1 is max left.

Fig. 11 Robot driving autonomously on a forest road.

of one lap is enough for successful autonomous driving counter clockwise around the track. The robot generalizes to the last part of the track as well as to changes of the track. The outdoor experiment provides further examples of generalization capabilities, the robot is manually controlled for two minutes whereafter the robot immediately and autonomously continues along a previously unseen part of the road.

References

1. Angulo, V.R.d., Torras, C.: Learning inverse kinematics via cross-point function decomposition. In: Proceedings of the International Conference on Artificial Neural Net-

- works, ICANN '02, pp. 856–864. Springer-Verlag, London, UK, UK (2002). URL <http://dl.acm.org/citation.cfm?id=646259.684479>
2. Atkeson, C., Moore, A., Schaal, S.: Locally weighted learning for control. *Artificial Intelligence Review* **11**(1-5), 75–113 (1997). DOI 10.1023/A:1006511328852
 3. Baranes, A., Oudeyer, P.Y.: Intrinsically motivated goal exploration for active motor learning in robots: A case study. In: *Intelligent Robots and Systems (IROS)*, 2010 IEEE/RSJ International Conference on, pp. 1766–1773 (2010). DOI 10.1109/IROS.2010.5651385
 4. Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. *CoRR* **cs.AI/9603104** (1996)
 5. de la Cruz, J.S., Kulić, D., Owen, W.: Online incremental learning of inverse dynamics incorporating prior knowledge. In: *Proceedings of the Second international conference on Autonomous and intelligent systems, AIS'11*, pp. 167–176. Springer-Verlag, Berlin, Heidelberg (2011). URL <http://dl.acm.org/citation.cfm?id=2026956.2026975>
 6. Dickmanns, E., Graefe, V.: Dynamic monocular vision. *Machine Vision and Applications* **1**, 223–240 (1988)
 7. Douze, M., Jégou, H., Sandhwalia, H., Amsaleg, L., Schmid, C.: Evaluation of gist descriptors for web-scale image search. In: *CIVR'09: Proceedings of the ACM International Conference on Image and Video Retrieval* (2009)
 8. D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, pp. 298–303 vol.1 (2001). DOI 10.1109/IROS.2001.973374
 9. Kastner, R., Schneider, F., Michalke, T., Fritsch, J., Goerick, C.: Image-based classification of driving scenes by a hierarchical principal component classification (HPCC). In: *IEEE Intelligent Vehicles Symposium*, pp. 341–346 (2009)
 10. Larsson, F., Jonsson, E., Felsberg, M.: Simultaneously learning to recognize and control a low-cost robotic arm. *Image Vision Computing* **27**(11), 1729–1739 (2009). DOI 10.1016/j.imavis.2009.04.003. URL <http://dx.doi.org/10.1016/j.imavis.2009.04.003>
 11. Lecun, Y., Muller, U., Ben, J., Cosatto, E., Flepp, B.: Off-Road Obstacle Avoidance through End-to-End Learning. In: Y. Weiss, B. Schölkopf, J. Platt (eds.) *Advances in Neural Information Processing Systems* **18**, pp. 739–746. MIT Press, Cambridge, MA (2006)
 12. Markelic, I.: Teaching a robot to drive: A skill-learning inspired approach. Ph.D. thesis, University of Göttingen (2010)
 13. Mitrovic, D., Klanke, S., Vijayakumar, S.: Adaptive optimal feedback control with learned internal dynamics models. In: O. Sigaud, J. Peters (eds.) *From Motor Learning to Interaction Learning in Robots, Studies in Computational Intelligence*, vol. 264, pp. 65–84. Springer Berlin Heidelberg (2010)
 14. Moré, J.J.: The Levenberg-Marquardt algorithm: Implementation and theory. In: G.A. Watson (ed.) *Numerical Analysis*, pp. 105–116. Springer, Berlin (1977)
 15. Öfjäll, K.: Leap, a platform for evaluation of control algorithms. Master's thesis, Linköping University (2010)
 16. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision* **42**(3), 145–175 (2001)
 17. Oyama, E., Agah, A., MacDorman, K.F., Maeda, T., Tachi, S.: A modular neural network architecture for inverse kinematics model learning. *Neuro-computing* **38**, 797–805 (2001)
 18. Oyama, E., Maeda, T., Gan, J., Rosales, E., MacDorman, K., Tachi, S., Agah, A.: Inverse kinematics learning for robotic arms with fewer degrees of freedom by modular neural network systems. In: *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 1791–1798 (2005). DOI 10.1109/IROS.2005.1545084
 19. Pomerleau, D.: *Alvin*: An autonomous land vehicle in a neural network. In: *Proc. of NIPS* (1989)
 20. Pomerleau, D.: Efficient training of artificial neural networks for autonomous navigation. *Neural Computation* **3**(1), 88–97 (1991)
 21. Pourboghrat, F., Shiao, J.C.: Neural networks for learning inverse kinematics of redundant manipulators. In: *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. ii, p. 1004 vol.2 (1991). DOI 10.1109/IJCNN.1991.155683

22. Pugeault, N., Bowden, R.: Learning pre-attentive driving behaviour from holistic visual features. In: ECCV 2010, Part VI, LNCS 6316, pp. 154–167 (2010)
23. Pugeault, N., Bowden, R.: Driving me around the bend: Learning to drive from visual gist. In: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pp. 1022–1029 (2011). DOI 10.1109/ICCVW.2011.6130363
24. Renninger, L., Malik, J.: When is scene identification just texture recognition? *Vision Research* **44**, 2301–2311 (2004)
25. Saegusa, R., Metta, G., Sandini, G., Sakka, S.: Active motor babbling for sensorimotor learning. In: Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on, pp. 794–799 (2009). DOI 10.1109/ROBIO.2009.4913101
26. Schaal, S., Atkeson, C.: Robot juggling: implementation of memory-based learning. *Control Systems, IEEE* **14**(1), 57–71 (1994). DOI 10.1109/37.257895
27. Schaal, S., Atkeson, C.: Learning control in robotics. *Robotics Automation Magazine, IEEE* **17**(2), 20–29 (2010). DOI 10.1109/MRA.2010.936957
28. Schmiterlöw, M.: Autonomous path following using convolutional networks. Master’s thesis, Linköping University (2012)
29. Settles, B.: Active learning literature survey. Tech. rep. (2009)
30. Siagian, C., Itti, L.: Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29**(2), 300–312 (2007)
31. Siagian, C., Itti, L.: Biologically inspired mobile robot vision localization. *IEEE Transactions on Robotics* **25**(4), 861–873 (2009)
32. Siciliano, B., Sciacicco, L., Villani, L., Oriolo, G.: *Robotics, Modelling, Planning and Control* (2009). ISBN 978-1-84628-642-1
33. Sugihara, T.: Solvability-unconcerned inverse kinematics based on levenberg-marquardt method with robust damping. In: Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on, pp. 555–560 (2009). DOI 10.1109/ICHR.2009.5379515
34. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.E., Koelen, C., Markey, C., Rummel, C., van Niekirk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P.: Stanley: The robot that won the DARPA Grand Challenge. *Journal of Robotic Systems* **23**(9), 661–692 (2006)
35. Turk, M., Morgenthaler, D., Gremban, K., Marra, M.: VITS—a vision system for autonomous land vehicle navigation. *IEEE Trans. in Pattern Analysis and Machine Intelligence* **10**(3), 342–361 (1988)
36. Vijayakumar, S., D’souza, A., Schaal, S.: Incremental online learning in high dimensions. *Neural Comput.* **17**, 2602–2634 (2005). DOI 10.1162/089976605774320557. URL <http://dl.acm.org/citation.cfm?id=1119418.1119423>