Learning Nonlinear State-Space Models for Control

Tapani Raiko and Matti Tornio
Neural Networks Research Centre
Helsinki University of Technology
P.O.Box 5400, FI-02015 TKK
Espoo, FINLAND
E-mail: tapani.raiko@hut.fi, matti.tornio@hut.fi

***Abstract*— This paper studies the learning of nonlinear state-space models for a control task. This has some advantages over traditional methods. Variational Bayesian learning provides a framework where uncertainty is explicitly taken into account and system identification can be combined with model-predictive control. Three different control schemes are used. One of them, optimistic inference control, is a novel method based directly on the probabilistic modelling. Simulations with a cart-pole swing-up task confirm that the latent state space provides a representation that is easier to predict and control than the original observation space.**

I. Introduction

Nonlinear control is difficult even in the case that the system dynamics are known. If the dynamics are not known, the traditional approach is to make a model of the dynamics (system identification) and then try to control the simulated model (nonlinear model-predictive control). The model learned from data is of course not perfect, but these imperfections are often ignored. The modern view of control sees feedback as a tool for uncertainty management [11], but managing it already in the modelling might have advantages. For instance, the controller can avoid regions where the confidence in model is not high enough [9].

The idea of studying uncertainty in control is not new. It is known that the magnitude of motor noise in human hand motion is proportional to muscle activation [10]. In control theory, the theoretical foundations are already well covered in [3]. In [14], a nonlinear state-space model is used for control. The nonlinearities are modelled using piecewise affine mappings. Parameters are estimated using the prediction error method, which is equivalent to the maximum likelihood estimate in the Bayesian framework.

Nonlinear dynamical factor analysis (NDFA) [17] is a state-of-the-art tool for finding nonlinear state-space models with variational Bayesian learning. This paper is about using NDFA for control. In NDFA, the parameters, the states, and the observations are real-valued vectors that are modelled with parametrised probability distributions. Uncertainties from noisy observations and model imperfections are thus taken explicitly into account.

Learning is extremely important for control of complex systems [2]. The proposed method involves learning in more than one way. The original NDFA is based on unsupervised learning. That is, it creates a model of the underlying dynamics by passively making observations. When used for control, though, control signals need to be selected either by following an example or by maximising a reward. The model should thus not only learn the dynamics, but also learn to help control.

The rest of the paper is structured as follows: In Section II, a nonlinear state-space model is reviewed and in Section III its use as a controller is presented. After experiments in Section IV matters are discussed and concluded.

II. Nonlinear State-Space Models

Nonlinear dynamical factor analysis (NDFA) [17] is a powerful tool for modelling the dynamics of an unknown noisy system. NDFA scales only quadratically with the dimensionality of the observation space, so it is also suitable for modelling systems with fairly high dimensionality [17].

In NDFA, the observations $\mathbf{x}(t)$ have been generated from the hidden state $\mathbf{s}(t)$ by the following generative model:

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t) \tag{1}$$

$$\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) + \mathbf{m}(t), \tag{2}$$

where $\boldsymbol{\theta}$ is a vector containing the model parameters and time t is discrete. The noise terms $\mathbf{n}(t)$ and $\mathbf{m}(t)$ are assumed to be Gaussian and white. Only the observations $\mathbf{x}$ are known beforehand, and both the states $\mathbf{s}$ and the mappings $\mathbf{f}$ and $\mathbf{g}$ are learned from the data.

Multilayer perceptron (MLP) networks [6] suit well to modelling both strong and mild nonlinearities. The MLP network models for $\mathbf{f}$ and $\mathbf{g}$ are

$$\mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) = \mathbf{B} \tanh\left[\mathbf{A}\mathbf{s}(t) + \mathbf{a}\right] + \mathbf{b} \tag{3}$$

$$\mathbf{g}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{g}) = \mathbf{s}(t) + \mathbf{D} \tanh\left[\mathbf{C}\mathbf{s}(t) + \mathbf{c}\right] + \mathbf{d}, \tag{4}$$

where the sigmoidal tanh nonlinearity is applied componentwise to its argument vector. The parameters $\boldsymbol{\theta}$ include: (1) the weight matrices $\mathbf{A} \dots \mathbf{D}$, the bias vectors $\mathbf{a} \dots \mathbf{d}$; (2) the parameters of the distributions of the noise signals $\mathbf{n}(t)$ and $\mathbf{m}(t)$ and the column vectors of the weight matrices; (3) the hyperparameters describing the distributions of biases and the parameters in group (2).

There are infinitely many models that can explain any given data. In Bayesian learning, all the possible explanations are averaged weighting by their posterior probability. The posterior probability $p(\mathbf{s}, \boldsymbol{\theta} \mid \mathbf{x})$ of the states and the parameters after observing the data, contains all the relevant information about them. Variational Bayesian learning is a way to approximate the posterior density by a parametric distribution $q(\mathbf{s}, \boldsymbol{\theta})$. The misfit is measured by the Kullback-Leibler divergence:

$$C_{\mathrm{KL}} = \int q(\mathbf{s}, \boldsymbol{\theta}) \log \frac{q(\mathbf{s}, \boldsymbol{\theta})}{p(\mathbf{s}, \boldsymbol{\theta} \mid \mathbf{x})} d\boldsymbol{\theta} d\mathbf{s}. \tag{5}$$

The approximation q needs to be simple for mathematical tractability and computational efficiency. Variables are assumed to depend of each other in the following way:

$$q(\mathbf{s}, \boldsymbol{\theta}) = \prod_{t=1}^{T} \prod_{i=1}^{m} q(s_i(t) \mid s_i(t-1)) \prod_{j} q(\theta_j), \tag{6}$$

where m is the dimensionality of the state space $\mathbf{s}$. Furthermore, q is assumed to be Gaussian.

Learning and inference happen by adjusting q such that the cost function C_{KL} is minimised. A good initialisation and other measures are essential because the iterative learning algorithm can easily get stuck into a local minimum of the cost function. The standard initialisation is based on principal component analysis of the data augmented with embedding. Details can be found in [17].

A. Iterated Extended Kalman Smoothing

In a typical NDFA learning phase, both the model parameters $\boldsymbol{\theta}$ and the states $\mathbf{s}$ are updated. The updating of the network weights is computationally the most expensive part of the process, so the speed of the updating of the states is of minor importance [17]. In the control schemes studied in this work, however, the model parameters can be kept fixed and only the states are inferred. As a faster alternative to the update process used in the NDFA Matlab package, extensions of Kalman filtering [7] are explored.

Kalman smoothing estimates the state of a linear Gaussian state-space model in a two-phase forward and backward pass. Extended Kalman smoothing [1] does the same for a nonlinear model by linearising the model based on the current estimate of the states and then applying linear Kalman smoothing. The process iterates between updating the states and the linearisation.

Kalman-based methods are fast because they propagate information through the whole time window in every iteration, whereas the update rules included in NDFA propagate information only one step forward and backward per iteration. Unfortunately, Kalman-based methods have no guarantee of convergence when applied to nonlinear systems. To solve this issue we used iterated extended Kalman smoothing for finding a good initialisation which was then improved by some NDFA updates.

In this work, a non-variational Kalman smoother is used. A variational Kalman smoother does exists [4], but as the Kalman smoother is used only for the initialisation of NDFA, the added complexity was not deemed worthwhile.

B. Task-Oriented Identification

When the dynamic system is controlled by a continuous-valued control signal vector $\mathbf{u}(t)$, it can be taken into account by replacing the equation of dynamics (2) with one of these two options:

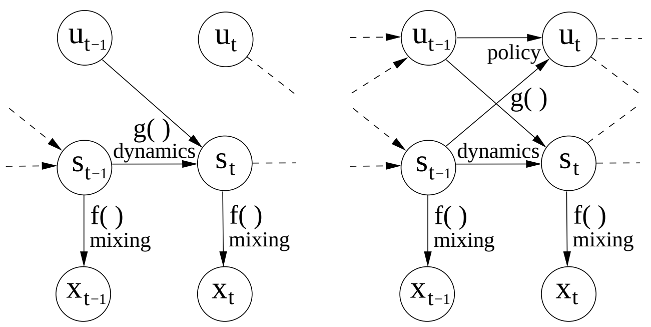

Fig. 1. Traditional model (left) and task-oriented identification (right). Traditionally, the control signals $\mathbf{u}(t)$ are coming from outside the model, but in task-oriented identification they are within the model.

$$\mathbf{s}(t) = \mathbf{g}\left(\begin{bmatrix} \mathbf{u}(t-1) \\ \mathbf{s}(t-1) \end{bmatrix}, \boldsymbol{\theta}_{\mathbf{g}}\right) + \mathbf{m}(t) \tag{7}$$

$$\begin{bmatrix} \mathbf{u}(t) \\ \mathbf{s}(t) \end{bmatrix} = \mathbf{g}\left(\begin{bmatrix} \mathbf{u}(t-1) \\ \mathbf{s}(t-1) \end{bmatrix}, \boldsymbol{\theta}_{\mathbf{g}}\right) + \mathbf{m}(t). \tag{8}$$

The first one (7) assumes that the control signal is coming outside the model. The latter one (8) is called task-oriented identification because it predicts the control signals $\mathbf{u}(t)$ within the model. Figure 1 illustrates these two options.

We choose to use task-oriented identification (Eq. 8) in this paper for the following reasons. Firstly, it allows for three different control schemes described in the next section. Secondly, it creates an opportunity to learn more. The learning algorithm finds such a state space that the prediction of observations and control signals is as accurate as possible. A well-learned state space should thus make control easier. Thirdly, it is biologically motivated. Different parts of the cerebellum can be used for motor control and cognitive processing depending on where their outputs are directed [5].

III. Control Schemes

So far only passive observation and learning has been considered. Now we come to the question how the control signals (or actions) are selected. That is, given the history of observations $\ldots, \mathbf{x}(t_0-2), \mathbf{x}(t_0-1)$ and control signals $\ldots, \mathbf{u}(t_0-2), \mathbf{u}(t_0-1)$, select a good control signal $\mathbf{u}(t_0)$ at the current time t_0. Then, a new observation $\mathbf{x}(t_0)$ is made and time t_0 is increased by one. Three different control schemes and their cooperation are studied below and summarised in Table I.

A. Direct Control (DC)

In direct control schemes, the neural network itself acts as the controller. Many such schemes exists, including direct inverse control, optimal control, and feedforward control [13]. Direct control can only mimic the control done in the data that has been used for learning. It therefore requires examples of correct control aiming at the same goal.

Equation (8) provides a prediction of the control signal $\mathbf{u}(t_0)$ based on the previous control signal $\mathbf{u}(t_0-1)$ and the

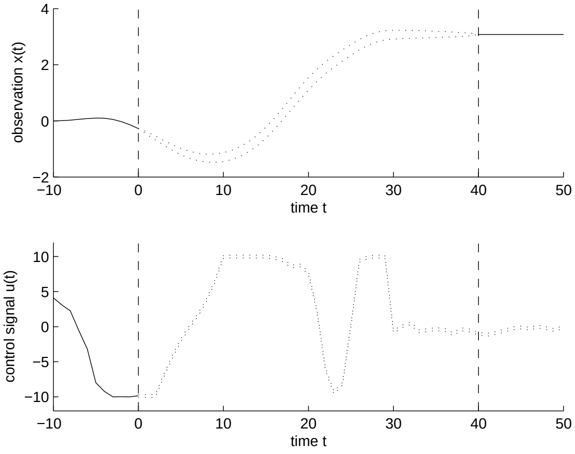

Fig. 2. Optimistic inference control (see Section III-B). The inferred observations and control signals are plotted with confidence intervals. The current time is $t_0 = 0$ and after time $t_0 + T_c = 40$, the observation $\mathbf{x}(t)$ is assumed to be at the desired level $\mathbf{r}$.

previous estimate of the hidden state $\mathbf{s}(t_0 - 1)$. The prediction mapping is called the policy in Figure 1. A control method that we simply call direct control (DC), chooses the control signal by collapsing the inferred probability distribution $q(\mathbf{u}(t_0))$ to its expected value. When the control signal $\mathbf{u}(t_0)$ is selected and the observation $\mathbf{x}(t_0)$ is made, the two probability distribution collapse and these changes affect the estimates of the states $\mathbf{s}(t)$ that are then re-inferred. This works as the error feedback mechanism.

B. Optimistic Inference Control (OIC)

Optimistic inference control (OIC) is a novel method which works as follows. Assume that after a fixed delay T_c, the desired goal is reached. That is, (some components of) the observations $\mathbf{x}$ are at the desired level $\mathbf{r}$. Given this optimistic assumption and the observations and control signals so far, infer what happens in between. Then choose the expectation of $q(\mathbf{u}(t_0))$ as before. An example situation is illustrated in Figure 2.

```
OIC in a nutshell:
Given observations $\ldots, \mathbf{x}(t_0 - 2), \mathbf{x}(t_0 - 1)$ and
control signals $\ldots, \mathbf{u}(t_0 - 2), \mathbf{u}(t_0 - 1)$
1:   Fix future $\mathbf{x}(t_0 + T_c) = \mathbf{x}(t_0 + T_c + 1) = \cdots = \mathbf{r}$
2:   Infer the distribution $q(\mathbf{u}(t), \mathbf{s}(t), \mathbf{x}(t))$ for all $t$
3:   Select the mean of $q(\mathbf{u}(t_0))$ as the control signal
4:   Observe $\mathbf{x}(t_0)$ and release $\mathbf{x}(t_0 + T_c)$
5: Increase $t_0$ and loop from 1
```

OIC propagates the same evidence forwards as the DC and additionally, the evidence from the desired future backwards. The inference is conceptually simple, but algorithmically difficult. The information from the future needs to flow through tens of nonlinear mappings $\mathbf{g}$ before it affects $\mathbf{u}(t_0)$.

In case there are constraints for control signals or observations, they are forced after every inference iteration. If the horizon is set too short or the goal is otherwise overoptimistic, the method becomes unreliable. Even with a realistic goal, it is not in general guaranteed that the iteration will converge to the optimal control signal, as the iteration may get stuck in a local minimum. The inferred control signals can be validated by releasing the optimistic future and re-inferring. If the future changes a lot, the control is unreliable. Note that OIC does not require goal-oriented data, because different goals can be set by changing the desired future.

TABLE I
CONTROL SCHEME SUMMARY

Scheme	Based on	Data	Speed
DC	internal MLP	task-oriented	fast
OIC	probabilistic inference	general	slow
NMPC	cost minimisation	general	slow

C. Nonlinear Model Predictive Control (NMPC)

Nonlinear model predictive control (NMPC) [13] is based on minimising a cost function J defined over a future window of fixed length T_c. For example, the quadratic difference between the predicted future observations $\mathbf{x}$ and a reference signal $\mathbf{r}$ can be used:

$$J(\mathbf{s}(t_0), \mathbf{u}(t_0), \ldots, \mathbf{u}(t_0 + T_c - 1)) = \sum_{\tau=1}^{T_c} |\mathbf{x}(t_0 + \tau) - \mathbf{r}|^2 . \tag{9}$$

Then J is minimised w.r.t. the control signals $\mathbf{u}$ and the first one $\mathbf{u}(t_0)$ is executed.

In this paper, the states and observations (but not control signals) are modelled probabilistically so we actually minimise the expected cost $E_q\{J\}$. The current guess $\mathbf{u}(t_0), \ldots, \mathbf{u}(t_0 + T_c - 1)$ defines a probability distribution over future states and observations. This inference can be done with a single forward pass, when ignoring the policy mapping, that is, the dependency of the state on future control signals. In this case, it makes sense to ignore the policy mapping anyway, since the future control signals do not have to follow the policy.

Minimisation of $E_q\{J\}$ is done with a certain quasi-Newton algorithm [12]. For that, the partial derivatives $\partial \mathbf{x}(t_2)/\partial \mathbf{u}(t_1)$ for all $t_0 \leq t_1 < t_2 \leq t_0 + T_c$ are computed efficiently based on the chain rule and dynamic programming. Details are left for future publications due to lack of space.

The use of a cost function makes NMPC very versatile. Costs for control signals and observations can be set for instance to restrict values within bounds etc. Quadratic costs such as (9) make things easy for the optimisation algorithm.

IV. EXPERIMENTS

Mechanical dynamical systems are easily understandable by people and thus illustrative as examples. We chose a simulated system to ease experimentation. To make the setting more realistic, the controllers do not have access to the simulation equations but have to adapt to control an unknown system instead.

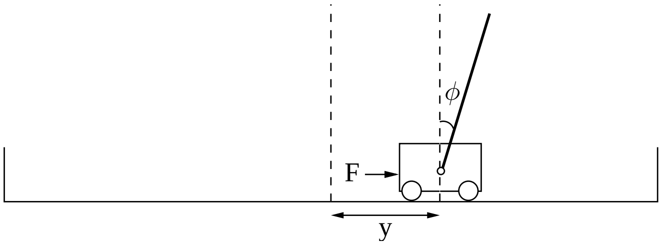

Fig. 3. The cart-pole system

A. Cart-Pole Swing-Up Task

The Cart-Pole system [8] is a classic benchmark for nonlinear control. The system consist of a pole (which acts as an inverted pendulum) attached to a cart (Figure 3). The force applied to the cart can be controlled, and the goal is to swing the pole to an upward position and stabilise it. This must be accomplished without the cart crashing into the walls of the track. Note that a linear controller cannot perform the swing-up.

The observed variables of the system are the position of the cart y, angle of the pole measured from the upward position ϕ, and their first derivatives y' and ϕ'. Control input is the force F applied to the cart. The detailed dynamics and constraints for the simulated cart-pole system can be found in [8].

A discrete system was simulated with a time step of $\Delta t = 0.05$s. The possible force was constrained between -10N and 10N, and the position between -3m and 3m. The system was initialised to a random state around $[y, y', \phi, \phi'] = [0, 0, -\pi, 0]$ with a standard deviation of 0.1 for all the observed variables.

B. Simulation

All simulations were ran with both low ($\sigma = 0.001$) and high ($\sigma = 0.1$) level of Gaussian additive observation noise. Gaussian process noise with $\sigma = 0.001$ was used in all the simulations and the training data set. For the NMPC and OIC methods the length of the control horizon was set to 40 time steps corresponding to 2 seconds of system's real time. The simulations were run for 60 time steps corresponding to 3 seconds of real time to ensure that the controller was able to stabilise the pole.

To study the benefits of using a hidden state-space in modelling the dynamics of an unknown system, a comparison model was built which used identity mapping $\mathbf{I}$ instead of an MLP $\mathbf{f}$ for the observation mapping. In practice this means replacing (1) with

$$\mathbf{x}(t) = \mathbf{s}(t) + \mathbf{n}(t). \tag{10}$$

Also, a modified version of the problem was considered, where only two observations, the location of the cart y and the angle of the pole ϕ, were available.

C. Implementation

The NDFA package version 0.9.5, the scripts for running the experiments, and the used training data are publicly available[1].

During the training phase for indirect methods, training data with 2500 samples was used. In [18], different reinforcement learning algorithms require from 9000 up to 2500000 samples to learn to control the cart. Most of the training data consisted of a sequence generated with semi-random control where the only goal was to ensure that the cart does not crash into the boundaries. Training data also contained some examples of hand-generated sections to better model the whole range of the observation and the dynamic mapping. The model was trained for 500000 iterations, which translates to three days of computation time. Six-dimensional state space $\mathbf{s}(t)$ was used because it resulted in a model with the lowest cost function (Eq. 5).

For the direct control method, training data consisted of 30 examples of successful swing-ups with 100 samples each. They were generated using the NMPC method with a horizon length of 40 time steps. Four-dimensional state space proved to be the best here, and the model was trained for 100000 iterations.

For all the models, the first 1000 iterations of the training were run with the embedded versions of the data to avoid bad local optima. Time-shifted versions of the observed data $\mathbf{x}(t-\tau)$, with $\tau = 1, 2, 4, 8, 16$, were used in addition to the original data.

The state $\mathbf{s}(t)$ was estimated using the iterated extended Kalman smoother. A history of five observations and control signals seemed to suffice to give a reliable estimate. The reference signal $\mathbf{r}$ was $\phi = 0$ and $\phi' = 0$ at the end of the horizon and for five observations beyond that.

To take care of the constraints in the system with NMPC, a slightly modified version of the cost function (9) was used. Out-of-bounds values of the location of the cart and the force incurred a quadratic penalty, and the full cost function is of the form

$$\begin{aligned} J_1(t_0, \mathbf{u}) =& J(t_0, \mathbf{u}) + \\ & \sum_{\tau=1}^{T_c} (\min(10, |u(t_0+\tau)|) - 10)^2 + \\ & \sum_{\tau=1}^{T_c} (\min(3, |x_y(t_0+\tau)|) - 3)^2, \end{aligned} \tag{11}$$

where $x_y(t)$ refers to the location component y of the observation vector $\mathbf{x}(t)$.

D. Simulation Results

For all the control schemes, the cart-pole simulation was run for 100 times and the number of successful swing-ups was collected. As in [8], a swing-up is considered successful if the final angle is between -0.133π and 0.133π, final angular velocity between -2rad/s and 2rad/s, and the cart has not crashed into the boundaries of the area during swing-up.

The results of all the simulations are collected in Table II. For each simulation type, the number of successful swing-ups and the number of partial successes are listed. The partial successes include all the simulation runs that at some point

[1] http://www.cis.hut.fi/projects/bayes/software/

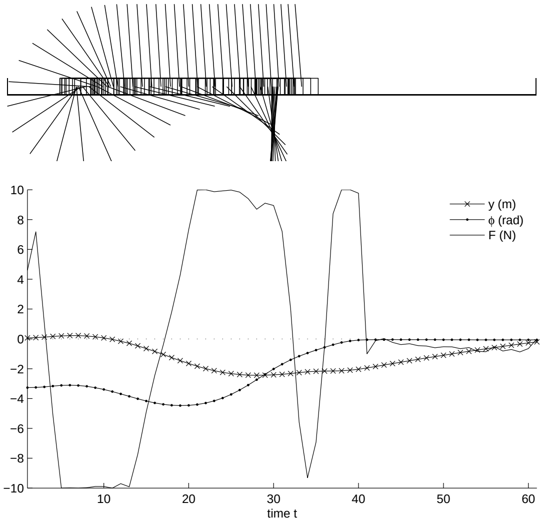

Fig. 4. Example of a successful swing-up with NMPC and low noise. The cart starts from the middle with the pole hanging down, and goes left to swing the pole up.

TABLE II

RESULTS: NUMBER OF SUCCESSFUL AND SEMI-SUCCESSFUL (IN BRACKETS) SWING-UPS WITH LOW AND HIGH NOISE LEVEL σ.

Setting	$\sigma = 0.001$		$\sigma = 0.1$	
Direct Control	14	(48)	4	(31)
Optimistic Inference Control	97	(100)	94	(98)
NMPC	100	(100)	94	(95)
NMPC (only y and ϕ observed)	14	(66)	1	(21)
NMPC ($\mathbf{f} = \mathbf{I}$)	100	(100)	70	(70)
NMPC ($\mathbf{f} = \mathbf{I}$, only y and ϕ)	0	(0)	0	(0)

reached the desired state, but possibly still failed either because the pole was not stabilised or the cart crashed into a wall.

1) Direct Control: The direct control could perform the swing-up part of the task quite well, but there were problems with stabilising the pole. Further testing is still needed to verify if the performance of the method can be improved by extra training with pole stabilising data.

2) Indirect Control: Even though there was some modelling error left in the model used with indirect control schemes, both methods performed extremely well under low noise conditions. Even with added noise, the performance was pretty satisfactory. Examples of successful swing-ups can be found in Figures 4 and 5.

3) Performance: With modern hardware (2.2 GHz AMD Opteron) the direct control typically worked in real-time with the cart-pole simulation. On average, the traditional NMPC method was about 20 times slower than real-time and OIC more than 100 times slower. The bad performance of OIC resulted from the Kalman smoothing (see Section II-A) not converging and having to switch to the slow update mode. Further optimisations to the algorithms or improvements in hardware are clearly required, before systems with fast dynamics can be controlled.

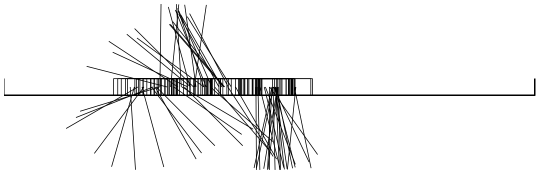

Fig. 5. Example of a successful swing-up with NMPC and high noise. The system is plotted with the observation noise included.

4) Dynamic Model Based Directly on the Observations: With the modified model using the observation space as the state space, the performance was still perfect when the noise level was low. However, with high noise level, the original model performed clearly better than the modified model.

5) Models with Fewer Observations: Even though most of the information on the speed y' and the angular velocity ϕ' can still be inferred taking into account past observations, in practice the problem of learning the dynamics of the system becomes harder and relying on past observations increases the reaction time. The model with hidden state could still perform the swing-up with some success, but a model based directly on observations could not handle the swing-up at all. This result was to be expected, as the dynamic mapping (8) alone cannot adequately describe the modified system.

6) Horizon length: Horizon length was of no great importance to the performance of the NMPC or the OIC. All horizon lengths between 30 and 45 time steps had similar performance. Horizon lengths between 25 and 30 had problems with the cart crashing to the walls. Horizons shorter than 25 time steps could not reliably perform the swing-up task because the reference signal became too unrealistic.

Very long horizons were also problematic. First of all, they increase the computational burden of the algorithm. The increase in the number of the parameters often also leads into increase in the number of local minima, which makes the optimisation problem more involved. In addition, because only an approximative model of the system is available, predictions far to the future become more unreliable. This can lead the algorithm to choose an optimisation strategy which is not feasible in practice.

V. DISCUSSION AND CONCLUSION

Three different control schemes were studied in the framework of nonlinear state-space models. Direct control is fast to use, but requires the learning of a policy mapping, which is hard to do well. Optimistic inference control is a novel method based on Bayesian inference answering the question: “Assuming success in the end, what will happen in near future?” It is based on a single probabilistic inference but unfortunately neither of the two tested inference algorithms work well with it. The third control scheme is a probabilistic version of the standard nonlinear model-predictive control, which is based on optimising control signals based on a cost function. The latter two schemes are both indirect control methods and they performed comparably well in the experiments.

A. Future Work

When learning from data, the model represents well only those phenomena that appear in the data. If the data is too uniform, the model will not become robust. In other words, one should balance between exploration and exploitation. In this paper, the data sets are generated partly by hand and all the control schemes aim at exploitation only. A good starting point for taking exploration into account is in [16].

For direct control, the model was learned using examples of control with a single goal in mind. It is straightforward to generalise this into a situation with a selection of different goals. The dynamics of the system stays the same regardless of the goal and only the policy mapping (see Figure 1) needs to be changed for each goal.

The direct and indirect control methods can be used together. One can use the data produced by indirect control methods for learning the direct controller. This can be done even offline, that is, simulating the estimated model and sampling observations from their predicted distributions. This can be compared to dreaming. The enhancement of the task-oriented identification (policy mapping) in turn helps the indirect methods, too. This idea is comparable to temporal difference learning [15] where the difference of temporally successive predictions is used for adjusting the earlier one. One should be careful, though. If the examples given for learning are fluent all the time, the robustness of the model might start to decrease.

When faced with an unknown state, the best thing to do is often first decrease the uncertainty by for example looking around, and then take action based on what has been revealed. This is called probing. Unfortunately the simple posterior approximation used in this paper does not allow such plans. The future actions (control signals) need to depend on future states but unfortunately they are assumed to be independent here. An interesting continuation is to use another posterior approximation, such as particle filters, for allowing that.

B. Main Results

Selecting actions based on a state-space model instead of based on the observation directly has many benefits: Firstly, it is more resistant to noise because it implicitly involves filtering. Secondly, the observations (without history) do not always carry enough information about the system state. Thirdly, when nonlinear dynamics are modelled by a function approximator such as an multilayer perceptron network, a state-space model can find such a representation of the state that it is more suitable for the approximation and thus more predictable.

When task-oriented identification is used, the state representation becomes such that also the control signals become easier to predict, that is, control becomes easier. The learned policy mapping can also be straightforwardly used for direct control. We think that task-oriented identification should also help indirect control methods but this is yet to be experimentally confirmed.

Nonlinear state-space models seem promising for complex control tasks, where the observations about the system state are incomplete or the dynamics of the system is not well known. The experiments with a simple control task indicated the benefits of the proposed approach. There is still work left in combating high computational complexity and in giving some guarantees or proofs on performance especially in unexpected situations or near boundaries.

Acknowledgement

The authors would like to thank Harri Valpola, Sampsa Laine, Kai Zenger, Heikki Hyötyniemi, and Antti Honkela for fruitful discussions and comments. This research has been funded by the Finnish Centre of Excellence Programme (2000-2005) under the project New Information Processing Principles, and by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

[1] B. Anderson and J. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[2] K.J. Åström, P. Albertos, M. Blamke, A. Isidori, W. Schaufelberger, and R. Sanz. *Control of complex systems*. Springer, 2001.

[3] Y. Bar-Shalom. Stochastic dynamic programming: Caution and probing. *IEEE Transactions on Automatic Control*, 26(5):1184–1195, October 1981.

[4] M. J. Beal and Z. Ghahramani. The variational Kalman smoother. Technical Report 003, Gatsby Computational Neuroscience Unit, 2001.

[5] K. Doya. What are the computations in the cerebellum, the basal ganglia, and the cerebral cortex? *Neural Networks*, 12(7):961–974, 1999.

[6] S. Haykin. *Neural Networks – A Comprehensive Foundation, 2nd ed.* Prentice-Hall, 1999.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[8] H. Kimura and S. Kobayashi. Efficient non-linear control by combining Q-learning with local linear controllers. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 210–219, San Francisco, CA, USA, 1999.

[9] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar. Predictive control with Gaussian process models. In *Proceedings of IEEE Region 8 Eurocon 2003: Computer as a Tool*, pages 352–356, 2003.

[10] G.G. Murray and K. Sykes. The variation of hand tremor with force in healthy subjects. *Journal of Physiology*, 191:699–711, 1967.

[11] R. Murray, K. J. Åström, S. P. Boyd, R. W. Brockett, and G. Stein. Future directions in control in an information-rich world. *IEEE Control Systems Magazine*, 23(2):20–33, April 2003.

[12] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[13] M. Nørgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen. *Neural Networks for Modelling and Control of Dynamic Systems*. Springer-Verlag London Limited, 2001.

[14] F. Rosenqvist and A. Karlström. Realisation and estimation of piecewise-linear output-error models. *Automatica*, 41(3):545–551, March 2005.

[15] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[16] S. B. Thrun. The role of exploration in learning control. In D. A. White and D. A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pages 527–559. Van Nostrand Reinhold, Florence, Kentucky, 1992.

[17] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.

[18] P. Wawrzynski and A. Pacut. Model-free off-policy reinforcement learning in continuous environment. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1091–1096, Budapest, Hungary, July 2004.