Combining Analytical and Learned Models for Model Predictive Control

Thomas Baumeister
Department of Computer Science
Technical University Munich
80333 Munich, Germany
thomas@baumeister-prv.de

Alina Kloss
Autonomous Motion Department
Max Planck Institue for Intelligent Systems
72076 Tübingen, Germany
alina.kloss@tuebingen.mpg.de

Jeannette Bohg
Department of Computer Science
Stanford University
Stanford, CA 94305
bohg@stanford.edu

Abstract

Robotics aims to deploy systems in increasingly complex and dynamic environments. This leads to a growing demand of robust and safe control algorithms for spatio-temporal domains, as failing potentially results in critical damage to the robot or its surroundings. Control policies can be learned in a purely data-driven manner. However, this approach is typically limited to applications where large datasets can be easily collected. The resulting policies are often challenged when they have to generalize to new situations. In this work, we propose to use Model Predictive Control (MPC) that relies on a dynamics model which integrates physical modeling and learning. We show how the underlying hybrid dynamics model can make more accurate long-term prediction than purely learned dynamics models specifically in new situations. Thereby, it ensures robust control. We demonstrate this in quantitative experiments on the task of planar pushing for which a physics model and a large real-world dataset is available.

1 Introduction

Deploying robots in unstructured and dynamic environments requires robust control laws even when facing large uncertainties. Model predictive control (MPC) incorporates continuous re-planning that addresses this problem. MPC relies on roll-outs of a dynamics model to predict future system states given a sequence of control inputs. One of the main challenges in designing an MPC architecture is to find an appropriate dynamics model that allows fast roll-outs into the future and efficient optimization over a time horizon. Analytical dynamics models are based on the laws of physics and can thus be expected to generalize well. However, these models often require approximations to become tractable and assume precise information about system parameters and state which may be hard to obtain. In contrast, neural networks have the ability to learn even very complex dynamics but are not guaranteed to generalize well to data outside of their training distribution. Lacking the ability to precisely predict future system states in unseen scenarios could lead to potentially catastrophic failure of the controller. We argue that combining analytical and learned models leverages the strength of each approach and thus enables robust control.

We extend a hybrid one-step prediction model [7] to a sequence-to-sequence model and embed this into an MPC approach. Our testbed is planar pushing, for which a well-known physics model [10] and the real-world MIT Push dataset [14] are available. We show how this novel approach increases generalization of the dynamics model and thereby enables robust control in unseen scenarios. We also show how it improves data and training efficiency in comparison to purely learned models.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

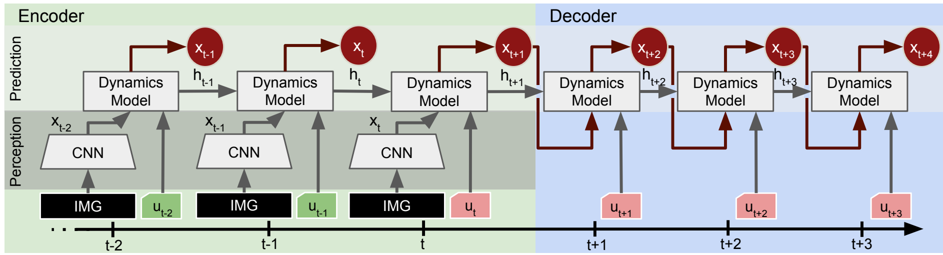

Figure 1: RNN architecture that performs MPC (i) by predicting the state $\boldsymbol{x}$ N timesteps into the future and (ii) by optimizing the future control inputs $\boldsymbol{u}_{t:t+N}$ to minimize a given cost function. The architecture is divided into an encoder and a decoder. Only the encoder has a perception model to estimate the state $\boldsymbol{x}_t$ from an input image while both, the encoder and decoder, have dynamic models to predict the future state $\boldsymbol{x}_{t+1}$.

2 Related Work

Combining Analytical and Learned Dynamics Models While Bauza and Rodriguez [3] and Li et al. [9] developed purely data-driven models for planar pushing, Ajay et al. [1] demonstrated that combining deterministic simulators with learnable, stochastic neural networks outperforms purely analytical and purely learned simulators. Moreover, an extensive evaluation of the advantages and limitations of combining learning and analytical models has been conducted in [7]. In contrast to previous work [3, 9, 1], we do not assume a fully observable state and estimate dynamic model parameters over the course of a sequence. We extend the one-step prediction model [7] to a sequence-to-sequence model that predicts over a longer time horizon.

Model Predictive Control MPC relies on a dynamics model to predict future states of the system $\boldsymbol{x}_{t+1:t+N+1}$ for the prediction horizon N given the future control inputs $\boldsymbol{u}_{t:t+N}$. The control inputs are optimized so that a cost function J is minimized. Variations of MPC have been successfully implemented in a large variety of applications [4, 8], such as aggressive driving [12], laser applications [2] or reactive planar manipulation [5]. Despite the great success, the performance of the controller is limited by the ability of the model to capture the dynamics of the system [11, 13].

3 Method

We propose a dynamics model that takes the form of a recurrent neural network (RNN). Its encoder takes a sequence of N images and control commands $\boldsymbol{u}_{t-N:t}$ as input. Its decoder predicts a sequence of future system states $\boldsymbol{x}_{t+1:t+N+1}$. This model is visualized in Figure 1.
Each encoder cell consists of a perception and a dynamics model. The perception model infers the state $\boldsymbol{x}_t$ at timestep t from the input image using a convolutional neural network (CNN). Given the estimated state $\boldsymbol{x}_t$ and the control inputs $\boldsymbol{u}_t$, the dynamics model predicts the future state $\boldsymbol{x}_{t+1}$ as well as some dynamic model parameters $\boldsymbol{h}_{t+1}$. Within the encoder, only the dynamic model parameters $\boldsymbol{h}$ are forwarded to the next cell, as $\boldsymbol{x}$ is estimated from the input images at each timestep. By forwarding $\boldsymbol{h}$, parameters can be estimated which are not encoded in a single image. The decoder cell takes the predictions from the previous timestep as input. This can be rolled out to predict several steps into the future. In our proposed method, the dynamics model is combining an analytical and a learned model. This is described in detail in Section 4.
During training, the input images as well as the corresponding control commands for the encoder cells $\boldsymbol{u}_{t-N:t}$ and the control commands for the decoder cells $\boldsymbol{u}_{t+1:t+N}$ are given. We train the model parameters $\boldsymbol{\Theta}$ to optimize the prediction accuracy. After training, we apply the model for a control task where the control command to the last encoder cell and to all decoder cells $\boldsymbol{u}_{t:t+N}$ (colored pink in Figure 1) become variables that can be optimized to achieve some desired states $\boldsymbol{x}^*_{t+1:t+N+1}$. To optimize $\boldsymbol{u}$, we use a variation of backpropagation through time (BPTT). The decoder thus performs MPC by first predicting the systems state N timesteps into the future and then optimizing the control commands $\boldsymbol{u}_{t:t+N}$ by: $\underset{\boldsymbol{u}_{t:t+N}}{\arg\min} \sum_{i=t+1}^{t+N+1} J_i = \underset{\boldsymbol{u}_{t:t+N}}{\arg\min} \sum_{i=t+1}^{t+N+1} ||\boldsymbol{x}^*_i - \boldsymbol{x}_i||^2_2$. We use Adam [6] to train the model and optimize the control inputs.

4 Experiments

We compare the proposed hybrid model to an analytical and purely learned model in terms of the following metrics: prediction accuracy, data and training efficiency as well as generalization to new scenarios. For this purpose, we designed two experiments. The first experiment evaluates prediction accuracy over different test sets given either 20k or 50k training samples. The second experiment assesses the performance of MPC when relying on either of the three dynamics models.

Dataset We used the MIT Push Dataset [14] for our experiments where eleven different objects (Figure 3) are pushed on four different surface materials. This data is annotated with object and pusher pose as well as contact points and normals [7]. For the initial experiments presented in this paper, we used the 20k and 50k subsets of the full dataset for training and testing the dynamics model. They contain only data points with push velocities up to 50 mm/s and zero pusher acceleration. One data point in our datasets consists of a sequence of RGB images and the corresponding object positions before and after the pushes are applied as well as the pusher's initial positions and movements.

Dynamic Model Variants Fig. 2 shows the three different dynamics models we compare in our experiments. These models were plugged into the architecture shown in Fig. 1. Each model takes as input the current state of the system $\boldsymbol{x}_t$ estimated by the CNN and the control command $\boldsymbol{u}_t$. It predicts the future state $\boldsymbol{x}_{t+1}$. Given the predicted state and the control command $\boldsymbol{u}_{t+1}$, the dynamics models can be recursively rolled out to predict several steps into the future.

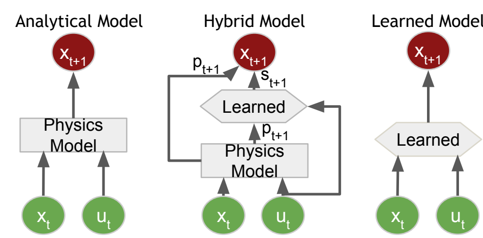

Figure 2: Three dynamics models. $\boldsymbol{x}$ represents the system state, $\boldsymbol{u}$ the control command, $\boldsymbol{p}$ the object pose and $\boldsymbol{s}$ contact dependent parameters. The number of learned parameters is increasing from left to right.

In the context of planar pushing, the system state $\boldsymbol{x}$ consists of the object pose $\boldsymbol{p}$ as well as contact dependent parameters $\boldsymbol{s}$ (α: angle of contact normal, $\boldsymbol{co}$: vector from contact point to object's center, δ: angle between $\boldsymbol{co}$ and x-axis). The dynamic model parameters $\boldsymbol{h}$ correspond to friction parameters.

In Fig. 2, the number of learned parameters is increasing from left to right. The leftmost model is purely analytical. Since we do not assume any prior knowledge about the object shape, we cannot model the transition of the contact dependent parameters $\boldsymbol{s}$ over time. Therefore, we keep these parameters constant during prediction. In contrast, the other prediction models learn the dynamics of these parameters from data. Apart from the multistep prediction, another distinction to [7] is the estimation of $\boldsymbol{h}_t$. The friction parameters $\boldsymbol{h}_t$ cannot be observed from single images and were thus assumed to be given in [7]. Here, we extended a long-short term memory (LSTM) architecture to estimate these dynamic model parameters over the course of a sequence. To increase the comparability among the models, we also let the purely analytical model estimate $\boldsymbol{h}$. This was achieved by adding a small fully connected neural network to the architecture shown in Fig. 1 to infer friction per input frame without transferring this information over time.

Experiment 1: Efficiency and Generalization We evaluate the data and training efficiency of the dynamics models and their ability to generalize over different object shapes. For this purpose, we test each model on shapes which were not included in the training and validation set. As evaluation metric, we use the mean absolute error between the prediction and the ground truth over the entire sequence. The object shapes shown in the bottom row of Fig. 3 are the test shapes and all other shapes in the dataset were used to train the model. In the left column, the test shapes are more similar to the training shapes than in the right column. It should therefore be harder to generalize for the test condition than in the right column. Moreover, we trained each model once with 20k and once with 50k examples to investigate training data efficiency. The results in Fig. 3 show that the mean and standard deviation (std) of the test error for the prediction models which include a physics model are significantly lower than for the purely learned model. Furthermore, the smaller the number of learned parameters, the lower is the initial mean and std. With increasing amount of training examples and epochs the prediction accuracy of the hybrid model is further improving.

These results agree with our hypothesis that hybrid models are able to increase the generalization ability over object shapes and substantially reduce the amount of training time and data needed to achieve high prediction accuracy as compared to a purely learned model.

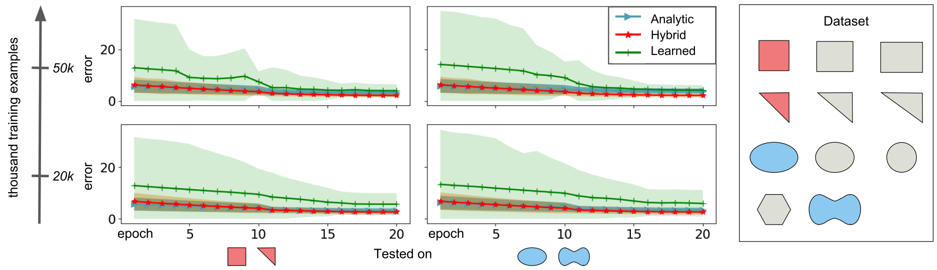

Figure 3: Mean and standard deviation of the absolute difference between prediction and ground truth averaged over a push sequence, plotted over training epochs. The test shapes (red and blue) were not in the training and validation set (gray shapes). This visualizes the generalization ability of the different dynamic models to unseen object shapes as well as their data and training efficiency.

Experiment 2: Performance of the MPC We evaluate the performance of our MPC approach when using different dynamics models. We used the models trained for 20 epochs with 20k training examples. The metric to quantify the control performance is the mean and std of the translational and rotational error between the desired and actual pose of the object during different phases of the push (beginning, middle and end). Since we had no access to the robot that was used to collect the dataset, we used pushes in the dataset to emulate real robot actions: Given an optimized control input $\boldsymbol{u}_t$ from the MPC, we search the dataset for the most similar case and applied it. This ensures that we know the actual effect of the applied push and do not have to resort to simulation. It also introduces some discrepancy between the desired and executed action which is expected behaviour when executing pushes on the real world. We again tested the MPC only on shapes which have not been in the training set to evaluate its generalization ability to novel situations.

Table 1: Quantitative analysis of the control performance showing the mean and stdt of translational (Δx in mm) and rotational error ($\Delta\phi$ in deg). The push sequence was divided into beginning (I), middle (II) and end phase (III).

Model	$\Delta x_I(\sigma_x)$	$\Delta\phi_I(\sigma_\phi)$	$\Delta x_{II}(\sigma_x)$	$\Delta\phi_{II}(\sigma_\phi)$	$\Delta x_{III}(\sigma_x)$	$\Delta\phi_{III}(\sigma_\phi)$
Analytical	**1.04**(0.91)	**0.71**(0.56)	3.31(3.10)	2.71(3.06)	5.66(4.84)	4.74(4.44)
Hybrid	1.66(1.65)	1.14(0.89)	**2.57**(2.59)	**1.88**(1.57)	**3.33**(3.01)	**2.65**(2.75)
Learned	2.96(2.61)	2.53(1.98)	6.15(3.99)	5.51(4.33)	8.97(6.01)	8.02(6.38)

From the results shown in Table 1 we can see that the hybrid MPC approach is in particular advantageous in the later parts of the push sequence. While the MPC architecture using the analytical model has the lowest error in the beginning of the push sequence, the error accumulates considerably faster than the one of the hybrid approach. Moreover, we see that the purely learned model does not perform as well as the other models. We have generated several videos to show qualitative results of the MPC performance. They can be found here.

5 Conclusion

The main contribution of this work is to demonstrate that combining analytical and learned models within an MPC architecture improves the robustness of the controller in the face of uncertainties. For our testbed of planar pushing, we showed that a hybrid dynamics model outperforms a purely learned model in both, long-term predictions and control. In addition, the hybrid approach reduces the dependency on large training sets, as it is significantly more data efficient than a purely learned model. We also showed that it leads to a lower accumulated error than a purely analytical model over a control sequence.

References

[1] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie Pack Kaelbling, Joshua B. Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *International Conference on Intelligent Robots and Systems (IROS)*, 2018. URL `http://lis.csail.mit.edu/pubs/ajay-iros18.pdf`.

[2] Thomas Baumeister, Steven L. Brunton, and J. Nathan Kutz. Deep learning and model predictive control for self-tuning mode-locked lasers. *J. Opt. Soc. Am. B*, 35(3):617–626, 2018. doi: 10.1364/JOSAB.35.000617. URL `http://josab.osa.org/abstract.cfm?URI=josab-35-3-617`.

[3] M. Bauza and A. Rodriguez. A probabilistic data-driven model for planar pushing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3008–3015, 2017. doi: 10.1109/ICRA.2017.7989345.

[4] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice — a survey. *Automatica*, 25:335–348, 1989. ISSN 0005-1098. URL `http://dx.doi.org/10.1016/0005-1098(89)90002-2`.

[5] Francois Robert Hogan, Eudald Romo Grau, and Alberto Rodriguez. Reactive planar manipulation with convex hybrid MPC. *CoRR*, abs/1710.05724, 2017.

[6] Diederik P. Kingma and Jimmy Ba, editors. *Adam: A Method for Stochastic Optimization*, 2015. URL `http://arxiv.org/pdf/1412.6980v9`.

[7] Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects. *arXiv*, 2018. URL `https://arxiv.org/abs/1710.04102`.

[8] Jay H. Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415, 2011. URL `https://doi.org/10.1007/s12555-011-0300-6`.
[9] Jue Kun Li, David Hsu, and Wee Sun Lee. Push-net: Deep planar pushing for objects with unknown physical properties. In *Robotics: Science and System*, 2018.
[10] K. M. Lynch, H. Maekawa, and K. Tanie. Manipulation and active sensing by pushing using tactile feedback. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 416–421, 1992.
[11] Swati Mohanty. Artificial neural network based system identification and model predictive control of a flotation column. *Journal of Process Control*, 19(6): 991–999, 2009. ISSN 09591524. doi: 10.1016/j.jprocont.2009.01.001.
[12] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202.
[13] Yu-Geng Xi, De-Wei Li, and Shu Lin. Model predictive control — status and challenges. *Acta Automatica Sinica*, 39(3):222–236, 2013. ISSN 18741029. doi: 10.1016/S1874-1029(13)60024-5.
[14] Kuan-Ting Yu, Maria Bauzá, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 30–37, 2016.