Towards Combining Robotic Algorithms and Machine Learning: End-To-End Learnable Histogram Filters

Rico Jonschkowski Oliver Brock

***Abstract*— Problem-specific robotic algorithms and generic machine learning approaches to robotics have complementary strengths and weaknesses, trading-off data-efficiency and generality. To find the right balance between these, we propose to use robotics-specific information encoded in robotic algorithms together with the ability to learn task-specific information from data. We demonstrate this approach in a proof of concept: the end-to-end learnable histogram filter. This fully differentiable implementation of a histogram filter encodes the structure of recursive state estimation using prediction and measurement update but allows the specific models to be learned end-to-end, i.e. in such a way that they optimize the performance of the filter, using either supervised or unsupervised learning.**

I. Introduction

The classic paradigm in robotics is to write *problem-specific robotic algorithms*, e.g. for recursive state estimation [1], that capture some essential truth about the problem which makes them very robust and data efficient. The field is currently shifting towards the new paradigm of *machine learning*, in particular deep learning, that allows robots to adapt to their tasks without human intervention and enable solutions even for those tasks which we humans cannot solve algorithmically, e.g. image classification. However, these two paradigms should not be viewed as exclusive but rather as complementary approaches that need to be balanced to trade-off their strengths and weaknesses. We propose to combine these approaches by exploiting robotics-specific information that is encoded in state-of-the-art algorithms and utilizing machine learning to determine the specifics of the given task.

This paper presents a proof of concept for this idea by demonstrating this combination for the *histogram filter*[1] [1], a method for recursive state estimation that represents the belief as a histogram over continuous states. Histogram filters require the specification of a motion model and a measurement model which define how the robot's actions and observations affect its belief. We combine the original algorithm with learning by implementing the histogram filter and its models in a differentiable way. Consequentially, we can learn these models end-to-end using backpropagation, which means that the models can be optimized for the overall performance of the filter rather than being learned individually using a proxy objective.

We tested the end-to-end learnable histogram filter in a partially observable 1-D localization task, in which neither the measurement model nor the motion model is known beforehand. The results show the advantage of the proposed method over both a histogram filter with individually learned models and standard end-to-end learnable recurrent networks.

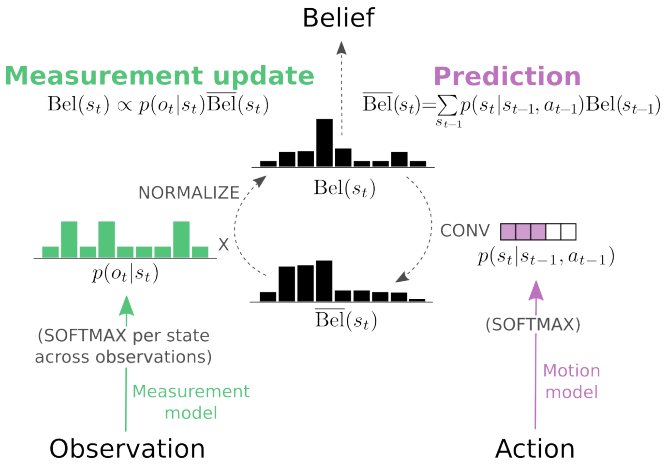

Fig. 1. The end-to-end learnable histogram filter. Motion model (purple) and measurement model (green) are learned; everything else is predefined.

Recent work has applied the same idea to Kalman filters [2]. In contrast to their work, our method learns both models jointly and is able to represent multi-modal beliefs.

II. Combining Robotic Algorithms and Machine Learning

Each information that a robot requires for solving a given task must either be hard-coded in the robot's algorithms or learned from data. Consequently, there is a trade-off between how much information must be put into the algorithms and how much information must be extracted from data. Algorithms form a spectrum from being task-specific, which makes them data-efficient and robust, to being fairly generic, which increases the range of possible tasks at the cost of requiring more data.

Robotic algorithms are at the specific end of this spectrum as they often require detailed models tailored to a specific task, e.g. a histogram filter with a specific motion model and measurement model works for one robot in one environment. Machine learning algorithms are at the other end of the spectrum. They are fairly generic (only assuming properties such as smoothness or hierarchy) and use large amounts of data to determine the specifics, which makes them applicable to a wide range of tasks, provided that there are enough data.

Since the set of robotic tasks is more restricted than the set of tasks usually considered in machine learning, we have

The authors are with the Robotics and Biology Laboratory, Technische Universität Berlin, Germany

[1]If applied to discrete states, the filter is known as *discrete Bayes filter*.

the opportunity to incorporate additional information about robotic tasks into learning. Physics, for example, is a constant that governs the interaction of any robot and its environment and can be used to make learning more efficient [3, 4]. But even beyond physics, robotic tasks include rich structure that can be exploited.

Robotic algorithms (implicitly) encode information about the structure of robotic tasks. We propose to use this *robotics-specific information* from robotic algorithms and combine it with machine learning to fill in the task-specific details based on data. We are convinced that this is a promising approach to strike the right balance between generality and data-efficiency in robotics.

III. Preliminaries

Before we demonstrate the combination of robotic algorithms and machine learning in form of the end-to-end learnable histogram filter, we will cover preliminaries about histogram filters and deep learning in this section.

A. Histogram Filters

A *Bayes filter* is method to recursively estimate the probability distribution over a latent state (e.g. robot pose) based on the history of observations and actions (e.g. camera images and odometry). A *histogram filter* is a type of Bayes filter that represents this belief as a histogram; a discretization of the state space with one probability value per discrete state s.

Histogram filters—and all Bayes filters—repeatedly update their belief by alternating two steps: *prediction* and *measurement update*. These steps are based on the previous belief $\text{Bel}(s_{t-1})$, the previous action a_{t-1} and the current observation o_t. The prediction step for a given state s_t,

$$\overline{\text{Bel}}(s_t) = \sum_{s_{t-1}} p(s_t \mid s_{t-1}, a_{t-1}) \text{Bel}(s_{t-1}), \tag{1}$$

sums over all previous states s_{t-1} and computes how likely the state would change from s_{t-1} to s_t if action a_{t-1} was applied times how likely state s_{t-1} actually was. It thereby sums over all possible ways through which state s_t could have come about. To make the prediction, the histogram filter requires the definition of a motion model $p(s_t \mid s_{t-1}, a_{t-1})$.

The measurement update,

$$\text{Bel}(s_t) \propto p(o_t \mid s_t) \overline{\text{Bel}}(s_t), \tag{2}$$

then multiplies the predicted belief $\overline{\text{Bel}}(s_t)$ by the likelihood of the current observation o_t and normalizes the result, which is an application of Bayes' rule. This step requires the definition of a measurement model $p(o_t \mid s_t)$.

Later in the paper, we will see how it is possible to train the entire histogram filter end-to-end to optimize state estimation performance. Towards this goal, we will formulate the prediction, the measurement update, and the corresponding models in the deep learning framework.

B. Neural Networks / Deep Learning

The most important aspect of artificial neural networks / deep learning [5] is the training of a sequence (or network) of functions by backpropagation—the repeated application of the chain rule to compute the gradient of the learning objective with respect to the network parameters. Any sequence of differentiable functions can in principle be trained with backpropagation. The histogram filter, however, consists of a loop rather than a sequence of functions.

a) Backpropagation Through Time: Loops in the network prohibit the direct application of backpropagation. Still, we can train such *recurrent networks* (e.g. an implementation of a recursive state estimation loop) if we unroll the network for a number of time steps, which creates a sequence of functions (or feedforward network) with shared parameters and inputs from different time steps. Applying the backpropagation algorithm to this unrolled network is called *backpropagation through time* and allows us to train recurrent networks.

b) Probabilities and Activation Functions: Since we want to model probability mass functions with neural networks, we need to constrain their outputs to represent probabilities. Network outputs should be non-negative numbers that sum to one. We can achieve this with the softmax activation function, $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$, which exponentiates and normalizes the network output.

IV. End-To-End Learnable Histogram Filters

The end-to-end learnable histogram filter (E2E-HF) is a differentiable implementation of a histogram filter that allows both motion model and measurement model to be learned end-to-end by backpropagation through time. Alternatively, we can view the E2E-HF as a new recurrent neural network architecture that implements the structure of a histogram filter (Fig. 1).

A. Belief

The histogram over states is implemented as a vector $\mathbf{b}$ of probabilities with one entry per bin,

$$\mathbf{b}_t = [\text{Bel}(S_t = 1), \text{Bel}(S_t = 2), \ldots, \text{Bel}(S_t = |S|)].$$

We can also think of the belief as a layer where the activation of each neuron represents the value of one histogram bin. The belief is the output of the current step and an input to the next step—together with an action a_t and an observation o_{t+1}.

B. Prediction

The most direct implementation of the prediction step would be to define a learnable function f that maps state and action to a histogram over next states,

$$f : s_{t-1}, a_{t-1} \mapsto [p(S_t = 1 \mid s_{t-1}, a_{t-1}), p(S_t = 2 \mid s_{t-1}, a_{t-1}), \ldots, p(S_t = |S| \mid s_{t-1}, a_{t-1})],$$

and to use f in the prediction step (Eq. 1). However, this approach is computationally expensive and allows arbitrary

"motions" of the robot, e.g. that an action has an entirely different effect at every state. Instead, we can often assume robot motion to be consistent across the state space,

$$p(s_t \mid s_{t-1}, a_{t-1}) = p(\Delta s_t \mid a_{t-1}),$$

where $\Delta s_t = s_t - s_{t-1}$. With this assumption, we can compute the prediction $\overline{\mathbf{b}_t}$ as a convolution

$$\overline{\mathbf{b}_t} = p(\Delta s_t \mid a_{t-1}) * \mathbf{b}_{t-1},$$

where the belief $\mathbf{b}_{t-1}$ is convolved with a fixed motion filter $p(\Delta s_t \mid a_{t-1})$ for the given action a_{t-1}.

The convolution is a predefined part of the end-to-end learnable histogram filter, but the mapping from action to motion filters is learned:

$$f : a_{t-1} \mapsto [p(\Delta S_t = -x \mid a_{t-1}), p(\Delta S_t = -x+1 \mid a_{t-1}), \\ \ldots, p(\Delta S_t = x \mid a_{t-1})],$$

where x is the maximum state change that is considered such that $2x+1$ is the size of the motion filter.

The filter is computed by a learnable motion model (see purple part in Fig. 1) which can be any feed forward network (e.g. a linear function with a softmax nonlinearity). Due to the softmax nonlinearity, the convolutional filter is constrained to be a probability distribution (positive numbers that sum to one). Note that this implementation of the prediction assumes that actions cause the same state change regardless of the state they are applied to. Actions such as "move towards the nearest door" cannot be represented in this way because their effects depend the states they are applied to.

C. Measurement Update

For discrete observations, it is straight-forward to implement the measurement model $p(o_t \mid s_t)$ for all states in a function

$$g : o_t \mapsto [p(o_t \mid S_t = 1), p(o_t \mid S_t = 2), \ldots, p(o_t \mid S_t = |S|)].$$

We will refer to the resulting vector of likelihoods as $\mathbf{g}_{o_t}$. It is important that this vector is normalized across observations not across states (which is what a softmax activation function would do). To realize the correct normalization, we need to compute the unnormalized likelihood vector $\tilde{\mathbf{g}}_o$ for every observation o and compute the softmax over the corresponding scalars in different vectors rather than over the scalars of the same vector.

$$\mathbf{g}_o = \frac{e^{\tilde{\mathbf{g}}_o}}{\sum_{o_i} e^{\tilde{\mathbf{g}}_{o_i}}}.$$

With this implementation of the measurement model, the measurement update (Eq. 2) can be implemented directly (see green part in Fig. 1). The likelihood of the current observation for each state is computed by the measurement model, which can again be any feed forward network, provided that the softmax nonlinearity as applied correctly as described above. It should be possible to extend this approach to continuous observations, if we can approximate the normalization using the sampled observations.

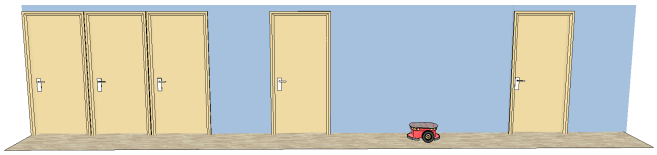

Fig. 2. 1D-hallway task

D. Learning

The filter can be learned end-to-end in a *supervised* way using backpropagation through time based on a sequence of observations, actions, and true (discretized) states, e.g. using the categorical cross-entropy loss between the belief the true state.

The end-to-end histogram filter can also be learned by optimizing any other differentiable loss function, e.g. in an *unsupervised* way by predicting the next observation based on the measurement model and the current belief. $\text{Pred}(o_t) = \sum_{s_t} p(o_t \mid s_t)\overline{\text{Bel}}(s_t)$, where $\text{Pred}(o_t)$ is the probability of that observation given the current belief.

V. Experiments

We compare the **E2E-HF** with both a histogram filter for which the models are learned individually (**HF**) and different variants of vanilla recurrent neural networks (**RNN**s) and long-short-term memory networks (**LSTM**s) [6] with 32 hidden units. We should expect that 1) the added robotics-specific information contained in the E2E-HF increases data-efficiency compared to RNNs and LSTMs and that 2) end-to-end learning improves the performance of the E2E-HF compared to HF with individually trained models. Furthermore, we investigate whether 3) the E2E-HF can be learned without any state labels (**E2E-HF unsupervised**). As this work is still in an early stage, we are using a well known toy task.

A. 1D-Hallway Task

In this task (see Fig. 2), a robot moves in a one-dimensional hallway with indistinguishable doors[2]. It does not know its position and only has access to its actions (LEFT / RIGHT / DO NOTHING) and observations (DOOR / NO DOOR). The task of the robot is to localize in this partially observable environment based on its stream of actions and observations, without knowing its motion model or measurement model beforehand. In our specific setting, the hallway has ten states (0-9), five of which have doors (0, 1, 2, 4, and 8). Training and test data are generated using a random walk. All models are trained on overlapping 32-step segments of a single trajectory of varying length (up to 1600 steps) and then tested on 1000 new trajectories of 32 steps.

B. Results

1) When we compare the different learning curves for supervised learning (see Fig. 3), we see that the histogram filter architecture is much more data efficient than the standard

[2]The 1D-hallway task is based on an illustrative example in [1].

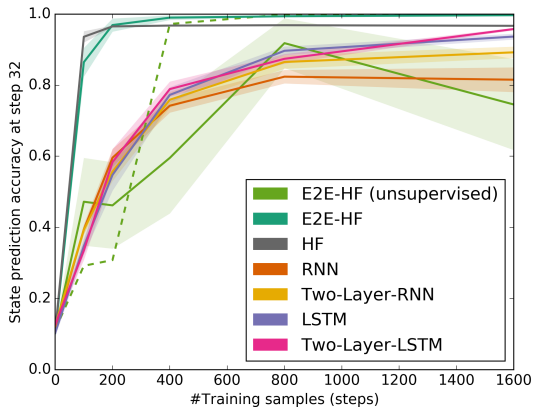

Fig. 3. Learning curves for different methods from 10 trials. Solid lines are means, shaded surfaces are standard errors. For the E2E-HF learned without state-labels/supervision, we also show the median as a dashed line.

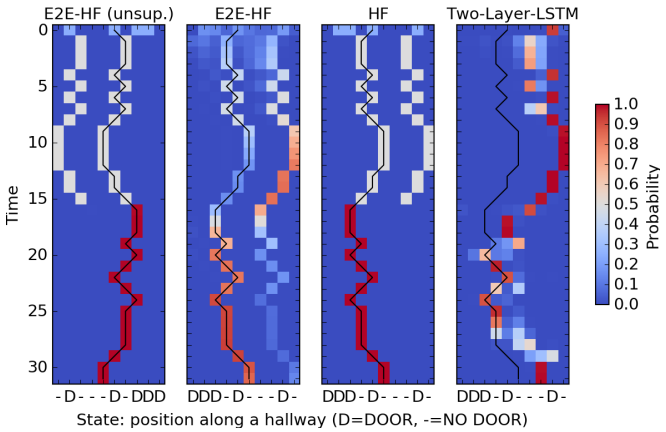

Fig. 4. Belief tracked during one test run (top-to-bottom) by different models, each trained on 400 steps. The black line shows the true trajectory.

recurrent deep learning architectures for this task. It only needs 200 to 400 training samples to achieve high accuracy.

2) End-to-end learning (E2E-HF) surpasses individual learning (HF) because it optimizes the models for the filtering process. This allows the model to accommodate for modeling errors–in this case, the false assumption that every starting state is equally likely, which is violated since the random walk often leads the robot to the edges of the environment. The E2E-HF can compensate for such errors by adapting the measurement model (see also Fig. 4, second from left). In the shown example, the histogram filter first prefers the right hypothesis even though both hypotheses are equally consistent with the evidence. It only switches to the other hypothesis after seeing counter-factual evidence at step 16. In some cases, this will help the filter to prefer outer hypotheses which are indeed more likely in the training and test data.

3) The histogram filter can be trained even without any state labels by predicting future observations using the measurement model (E2E-HF unsupervised). Even though our current approach to unsupervised learning can get stuck in local minima (which leads to the high variance), the high median performance is promising (see dashed line in Fig. 3). Of course, the histogram filter might learn a flipped state representation (see Fig. 4, left) which we counted as correct for the learning curve.

VI. Conclusions

In this paper, we have proposed to use robotic algorithms as the source of robotics-specific information that can be combined with machine learning. We are convinced that this approach is very promising to balance data-efficiency and generality in robotics.

We have presented a proof of concept for this idea in form of the end-to-end learnable histogram filter. By making the histogram filter and its models fully differentiable, we combine the structural assumptions of a Bayes filter with the end-to-end learnability of a recurrent neural network.

The structural assumptions regularize learning to make it more data efficient, while end-to-end learning has advantages over individual learning of the motion and measurement model: 1) the models can learn to accommodate for each others errors to improve the overall performance; 2) we can use any objective function for learning, which allows unsupervised learning of the histogram filter, i.e. it does not require state labels.

In this paper, we have only shown the combination of robotic algorithms and machine learning for one specific algorithm. There are many robotic algorithms that can potentially benefit from being combined with machine learning in a similar fashion.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[2] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop KF: Learning Discriminative Deterministic State Estimators," *arXiv preprint arXiv:1605.07148*, 2016.

[3] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, no. 3, pp. 407–428, Jul. 2015.

[4] J. Scholz, M. Levihn, C. L. Isbell, and D. Wingate, "A Physics-Based Model Prior for Object-Oriented MDPs," in *ICML*, 2014.

[5] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," 2016.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.