Proceedings of Machine Learning Research vol 211:1–14, 2023 5th Annual Conference on Learning for Dynamics and Control

Filter-Aware Model-Predictive Control

Barış Kayalıbay BKAYALIBAY@ARGMAX.AI
Atanas Mirchev ATANAS.MIRCHEV@ARGMAX.AI
Ahmed Agha AHMED.AGHA@ARGMAX.AI
Patrick van der Smagt
Justin Bayer BAYERJ@ARGMAX.AI
Machine Learning Research Lab, Volkswagen Group, Munich

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Partially-observable problems pose a trade-off between reducing costs and gathering information. They can be solved optimally by planning in belief space, but that is often prohibitively expensive. Model-predictive control (MPC) takes the alternative approach of using a state estimator to form a belief over the state, and then plan in state space. This ignores potential future observations during planning and, as a result, cannot actively increase or preserve the certainty of its own state estimate. We find a middle-ground between planning in belief space and completely ignoring its dynamics by only reasoning about its *future accuracy*. Our approach, filter-aware MPC, penalises the loss of information by what we call "trackability", the expected error of the state estimator. We show that model-based simulation allows condensing trackability into a neural network, which allows fast planning. In experiments involving visual navigation, realistic every-day environments and a two-link robot arm, we show that filter-aware MPC vastly improves regular MPC.

Keywords: model-predictive control, partially-observable, dynamic programming

1. Introduction

In partially observable Markov decision problems (POMDPs) we can only estimate the system state from a history of observations. This makes planning in POMDPs complicated, as the planner must find an optimal balance between reducing costs according to the current state estimate, and gathering new observations which will refine that state estimate, allowing for even better cost minimisation.

Optimally trading-off cost reduction and information gathering is only possible by planning in the space of beliefs, but that is often too expensive. Another approach, taken by model-predictive control (MPC),[1] is to decouple state estimation and planning completely. The planner only focuses on costs, finding the best sequence of controls starting from the current belief. Planning in this way is as cheap as planning in a fully observable problem, but allows no way of actively acquiring new information or preserving the information we already have.

We try to find a middle ground between these two approaches. In our approach, we do not plan in belief space, but we also do not ignore future beliefs completely. Instead, we settle for reasoning about the future accuracy of the state estimate. This results in a controller that can actively maintain a desired amount of accuracy in its belief.

1. Related terms are receding horizon control, limited lookahead control, and rollout. We use the term MPC and clarify what we mean in section 3.

As a motivating example, we can look at visual navigation, where a visual tracker uses RGB-D images to track the location of a robot which is trying to reach some target coordinates. Following the shortest path might force the robot to pass through a long corridor that does not contain enough information for the visual tracker to work with. The state estimate might degrade so much that it is of no help to the planner. If we have a way of predicting the tracker's accuracy around this corridor, we can inform the planner about it, which can then decide to follow a slightly longer path, but one where the state estimate will remain accurate.

We propose a straightforward method for blending this aspect into MPC. We introduce the idea of trackability, which maps a system's state to the expected error of a state estimator. By putting a threshold on trackability, we add an additional constraint to the MPC problem. Solving this augmented problem minimises the total cost of the original control task, while maintaining a desired degree of accuracy in the state estimator. We show that trackability can be seen as a value function of a new MDP where the state contains both the original system state and the state estimator's belief. This allows us to learn a neural network that predicts trackability using standard approximate dynamic programming techniques. We verify these ideas in experiments designed to illustrate the problem, including visual navigation, realistic every-day environments and a two-link robot arm. In all cases, we obtain substantial improvements over baselines that ignore how the belief will evolve over time.

Our exact contributions are:

- We formulate a measure of how accurate state estimation will be under a plan and a starting state. We learn this measure with a neural network for fast planning.
- We present filter-aware MPC, a version of MPC that counteracts state estimation errors by putting a threshold on trackability.
- We empirically show that filter-aware MPC improves regular MPC and that it performs competitively against other baselines using observations.

2. Related work

Åström (1965) was among the first to publish theoretical results on decision-making with missing information. Sondik (1971) developed a method for finite-horizon optimal control, showing that the value function is piecewise-linear in this setting. Later works dealt with the infinite horizon case (Sawaki and Ichikawa, 1978; Sondik, 1978). A large body of work focused on performance improvements for exact inference (Littman, 1996; Kaelbling et al., 1998; Cassandra et al., 1998). Others sought to find efficient approximations (Parr and Russell, 1995; Thrun, 1999; Zhang and Zhang, 2001; Pineau et al., 2003). Recent works presented tree search algorithms using particle filters (Sunberg and Kochenderfer, 2017; Sunberg et al., 2017), allowing to solve POMDPs with continuous states, controls and observations.

Many works focused on the case with Gaussian beliefs. These include (Platt et al., 2010), working with extended Kalman filter updates and (Todorov, 2005), which allows belief planning in an LQG setting with multiplicative noise. van den Berg et al. (2021) presented a value learning algorithm with the EKF. Rafieisakhaei et al. (2017) improved LQG planning by optimising for the lowest perception uncertainty. Rahman and Waslander (2020) introduced a vision-oriented approach with Gaussian beliefs where the uncertainty of the belief is constrained. Similar measures were taken in (Hovd and Bitmead, 2005) and (Böhm, 2008).

Our approach in this paper is also related to *coastal navigation* (Roy and Thrun, 1999; Roy et al., 1999), which focuses on preserving the accuracy of the state estimate, rather than full planning in belief space. Similarly, *belief roadmaps* (Prentice and Roy, 2009) try to find a path from a start location to a target while minimising the uncertainty of an EKF. This was later extended to work with unscented Kalman filters (He et al., 2008). Belief roadmaps were built upon by Zheng et al. (2021) by constraining the covariance of the belief.

Our method differs from these works in a number of ways. First, we look at the expected error of a state estimator. The expected error will be essentially the same as the entropy if the state estimate is the true posterior (or an accurate approximation), but is also applicable to state estimators that are not accurate approximations of the true posterior in all cases and ones that produce a point estimate. Second, we limit expected tracking errors into the future (beyond the planning horizon) and rely on approximate dynamic programming techniques for that, which is an aspect we have not found in the literature. Finally, since our approach only operates in state space, we are able to work on high-dimensional problems where (approximate) belief planning is typically infeasible, e.g. problems with image observations.

Several recent papers showed great results on partially observable problems using model-based methods (Karl et al., 2017; Hafner et al., 2019; Becker-Ehmck et al., 2020; Hafner et al., 2020). These works learn a policy that works with full state information by simulating the problem with a generative model. At test time, an inference network maps the observation stream to a latent state, which is passed to the policy. This strategy is similar to MPC with a state estimator: the policy takes the current state estimate and tries to find an optimal strategy. The work of Igl et al. (2018) and Han et al. (2020) are notable exceptions, as they train policies in belief space. Similarly, Lee et al. (2019) have presented a latent space algorithm which approximates belief planning. Our work mainly differs from these in that we focus on MPC, which has desirable properties such as state-constraints, stability and performance guarantees.

Finally, we would like to point the reader to the field of dual control (Mesbah, 2018), which contains methods building on MPC that deal with the trade-off between cost reduction and state estimation (Köhler et al., 2021).

3. Background

3.1. Systems with Imperfect State Information

We work with problems where a hidden state $\mathbf{z}_t \in \mathbb{R}^{N_\mathbf{z}}$ changes over time according to some transition function f given controls $\mathbf{u}_t \in \mathbb{R}^{N_\mathbf{u}}$ and process noise $\mathbf{w}_t$. The hidden state can only be estimated through observations $\mathbf{x}_t \in \mathbb{R}^{N_\mathbf{x}}$ that are produced by an emission function g subject to measurement noise variables $\mathbf{v}_t$. The process and the observation noise might be state and control dependent with known distributions and dependency. Using some control $\mathbf{u}_t$ in some state $\mathbf{z}_t$ leads to a cost $\mathbf{c}_t \in \mathbb{R}^+$ which is determined by the cost function c. This setup can be summarised, for $t \geq 1$:

$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \mathbf{u}_t, \mathbf{w}_t) \quad \mathbf{c}_t = c(\mathbf{z}_t, \mathbf{u}_t) \quad \mathbf{x}_t = g(\mathbf{z}_t, \mathbf{v}_t), \tag{1}$$

with initial state distribution $p(\mathbf{z}_1)$. The controls are given by a stationary policy $\mathbf{u}_t = \pi(\mathbf{x}_{1:t}, \mathbf{u}_{1:t-1})$ that is conditioned on all previous observations and controls. The expected total cost of a policy π is:

$$J^\pi(\mathbf{z}) = \mathbb{E}\Big[\sum_{t=1}^{\infty} \beta^{t-1} c(\mathbf{z}_t, \mathbf{u}_t) \mid \mathbf{z}_1 = \mathbf{z}\Big],$$

where $\beta \in (0, 1)$. An optimal policy attains minimal expected cost, i.e. $\pi^\star = \arg\min_\pi \mathbb{E}\left[J^\pi(\mathbf{z})\right]$.

3.2. Model Predictive Control for Imperfect State Information

A variant of MPC related to open-loop feedback control (Dreyfus, 1965) approximates this problem by devising an optimal plan for the first K steps and using its first step:

$$\pi_{\text{MPC}}(\mathbf{x}_{1:t}, \mathbf{u}_{1:t-1}) = \arg\min_{\mathbf{u}_t} \min_{\mathbf{u}_{t:t+K-1}} \mathbb{E}_{\mathbf{z}_t|\mathbf{x}_{1:t},\mathbf{u}_{1:t-1}} \left[\sum_{k=t}^{t+K-1} \beta^{k-t} c(\mathbf{z}_k, \mathbf{u}_k) + \beta^K \hat{J}(\mathbf{z}_{t+K}) \right]. \quad (2)$$

A terminal cost function $\hat{J}$ is used to alleviate some of the bias resulting from omitting the future.

3.3. State Estimation for MPC

Sampling the future states $\mathbf{z}_{t:t+K}$ in eq. (2) necessitates a distribution over the current state, the *belief*. This is done by a state estimator, which reads the observations into an internal state, or *carry*, which we denote with $\mathbf{h}$. The carry is updated by a function $h(\cdot, \cdot)$ which looks at the current observation and the previous carry and control: $\mathbf{h}_{t+1} = h(\mathbf{h}_t, \mathbf{x}_{t+1}, \mathbf{u}_t)$.

The form of the carry depends on the state estimator. For a particle filter, the carry would be the set of particles and weights. For an RNN, it would be the hidden state vector. Using the concept of the carry, we define the belief as $q(\mathbf{z}_t \mid \mathbf{h}_t)$, which acts as the distribution to use in eq. (2). It is often reasonable to assume that the initial state of the system is known (e.g. when we are only interested in relative motion). We formalise this as setting $\mathbf{h}_1 := \mathbf{h}^*_{\mathbf{z}}$, where $\mathbf{h}_1$ is the initial carry and $\mathbf{h}^*_{\mathbf{z}}$ is a carry that deterministically identifies the true state $\mathbf{z}$. We generally use the same notation $\mathbf{z}$ for both true and estimated states to keep notation simple.

3.4. Suboptimality of MPC

Model predictive control is suboptimal because it assumes the agent will collect no future observations. In fact, it can be shown that this control scheme results from a single step of policy improvement over open-loop planning (Bertsekas, 2005) in the rollout framework. While this often results in dramatic improvements, we argue that there remains a blind spot in MPC.

As an extreme case, consider an additional action which grants the agent access to the true state at the *next* step. There is no incentive for the MPC-based policy to take it, as it will not reduce the expected cost of open-loop plans.

The last insight serves as the major motivation of our work: we will explicitly integrate the expected performance of a state estimator in the planning of MPC.

4. Filter-Aware MPC

We aim to improve MPC by letting it distinguish between actions not just in terms of how much they reduce the cost, but also how they influence the belief. We want to find out the future accuracy of the state estimate under different plans and put a constraint on that value such that we only pick plans that guarantee a certain level of accuracy.

At every time step, any state estimator will introduce some inaccuracy, even the optimal Bayes filter. We denote this error with $\epsilon(\mathbf{z}_t, \mathbf{h}_t)$. In the case of a probabilistic state estimator, this might be its negative log-likelihood $-\log q(\mathbf{z}_t \mid \mathbf{h}_t)$, or the sum of squares in case of a deterministic one. If we take the example of a Kalman filter, $\mathbf{h}_t$ would be the mean and covariance matrix of the posterior distribution over the system state at time t, and $\epsilon(\mathbf{z}_t, \mathbf{h}_t)$ would be the negative log-likelihood of $\mathbf{z}_t$ under that distribution.

For any policy, the discounted expected total state estimation error starting from a state $\mathbf{z}$ and a carry $\mathbf{h}$ is:

$$J^{\text{err}}(\mathbf{z}, \mathbf{h}) = \mathbb{E}\Big[\sum_{t=1}^{\infty} \beta^{t-1} \epsilon(\mathbf{z}_t, \mathbf{h}_t) \mid \mathbf{z}_1 = \mathbf{z}, \mathbf{h}_1 = \mathbf{h}\Big], \tag{3}$$

where the expectation is over both the transition and the emission noise and the controls are picked by some policy π. Future tracking errors are discounted by $\beta \in (0, 1)$.

We define $J^{\text{err}}(\mathbf{z}, \mathbf{h})$ as the trackability of $\mathbf{z}$ under the policy π and the carry $\mathbf{h}$. Given a sequence of controls $\mathbf{u}_{t:t+K-1}$ and stochastic dynamics, the trackability of the state at time t' will be stochastic. We extend eq. (2) with a chance constraint (Farina et al., 2016):

$$\begin{aligned} \min_{\mathbf{u}_{t:t+K-1}} \quad & \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}_t | \mathbf{h}_t)} \Big[J^{\text{MPC}}(\mathbf{z})\Big], \\ \text{s.t.} \quad & \Pr\big(J^{\text{err}}(\mathbf{z}_{t+i}, \mathbf{h}_{t+i}) \leq \delta\big) \geq \Delta \quad \text{for} \quad i = 1, 2, \ldots, K. \end{aligned} \tag{4}$$

Here, δ is a threshold which defines how much tracker error we are willing to tolerate and Δ is a minimum probability of satisfying this condition that we wish to guarantee. We refer to this version of MPC as filter-aware MPC. Just like regular MPC, filter-aware MPC minimises the total cost. Unlike regular MPC, it also considers how a sequence of actions will affect the state estimate, only allowing plans where a minimum degree of state estimation accuracy can be achieved with high enough probability.

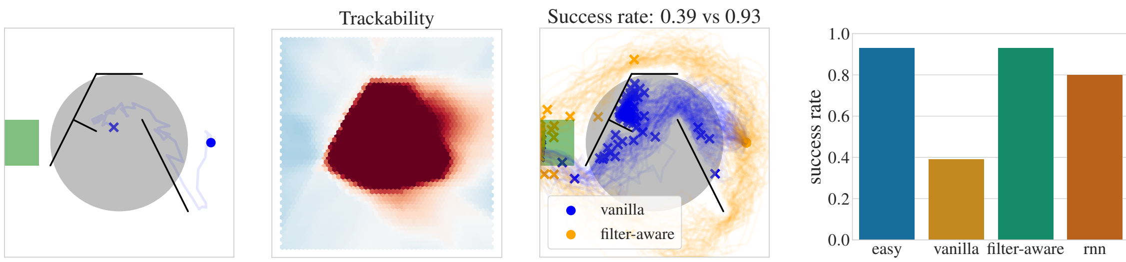
Unfortunately, optimising eq. (4) will not bring us very far in practice. The problem is that checking the constraint for time $t + i$ requires us to predict the future carry $\mathbf{h}_{t+i}$. This is as computationally expensive as planning in belief space. In this version of filter-aware MPC, checking the constraints comes at a greater expense than evaluating the MPC objective.

We make an approximation to arrive at a feasible algorithm. We approximate $J^{\text{err}}(\mathbf{z}_t, \mathbf{h}_t) \approx J^{\text{err}}(\mathbf{z}_t, \mathbf{h}^*_{\mathbf{z}_t})$. In other terms, when we check the trackability of the state visited in time t, we only care about tracking errors that will happen in future time steps $t' > t$ as if the process was started fresh from $\mathbf{z}_t$ with a perfect state estimate. In the remainder, we drop the conditioning on $\mathbf{h}^*_{\mathbf{z}}$ and set $J^{\text{err}}(\mathbf{z}) := J^{\text{err}}(\mathbf{z}, \mathbf{h}^*_{\mathbf{z}})$ for brevity. We are able to do so because $\mathbf{h}^*_{\mathbf{z}}$ is a function of $\mathbf{z}$.

We will use a multi-layer perceptron (MLP) to predict $J^{\text{err}}(\mathbf{z})$. An MLP is a natural choice for $J^{\text{err}}(\mathbf{z})$, since this can be seen as the negative value function (or cost-to-go) of an MDP where the cost is the tracking error and the MDP-state combines the system state $\mathbf{z}$ and the carry $\mathbf{h}$, though the latter does not enter into our computations since we are only interested in the case where $\mathbf{h} = \mathbf{h}^*_{\mathbf{z}}$. Taking this angle, we use TD(λ) (Sutton, 1988) to learn an MLP ϕ that approximates trackability. Next, by evaluating $\phi(\mathbf{z}_t)$, we are able to check the constraints without ever reasoning about future state estimates directly.

4.1. Learning Trackability

Using the state estimator and our models for the transition and emission f and g, we can create rollouts of a regular MPC policy. These rollouts have the form $\mathcal{D} = \{\mathbf{z}^i_{1:T}, \mathbf{h}^i_{1:T}, \mathbf{e}^i_{1:T-1}\}_{i=1}^N$, where we denote the state estimation error at time t by $\mathbf{e}_t := \epsilon(\mathbf{z}_t, \mathbf{h}_t)$.



(a) (b) (c) (d)

Figure 1: Toy scenario of a 2D particle aiming for the green region on the left starting from the right. The grey circle is a "dark zone" with higher observation noise. (a) A random walk. (b) Learned trackability. (c) Filter-aware vs vanilla MPC. (d) Success rates of vanilla MPC on an easy system with no grey zone, vanilla MPC, filter-aware MPC and an RNN policy.

Given $\mathcal{D}$, we can train the neural network by minimising:

$$\min_{\phi} \sum_{i}^{N} (\phi(\mathbf{z}_1^i) - J^{\text{err}}(\mathbf{z}_1^i; \lambda, \phi))^2, \tag{5}$$

where $J^{\text{err}}(\mathbf{z}_1^i; \lambda, \phi)$ is the λ-return of $\mathbf{z}_1^i$ defined as:

$$J^{\text{err}}(\mathbf{z}_1^i; \lambda, \phi) = (1-\lambda) \sum_{k=1}^{T-1} \lambda^{k-1} J_k^{\text{err}}(\mathbf{z}_1^i; \phi), \qquad J_k^{\text{err}}(\mathbf{z}_1^i; \phi) = \sum_{t=1}^{k} \beta^{t-1} \mathbf{e}_t^i + \beta^k \phi(\mathbf{z}_{k+1}^i).$$

4.2. A Practical Implementation of Filter-Aware MPC

Once the trackability net ϕ has been learned, any optimisation algorithm that can handle constraints can be used to optimise eq. (4) by replacing $J^{\text{err}}(\mathbf{z}_{t+i}, \mathbf{h}_{t+i}) \approx \phi(\mathbf{z}_{t+i})$. In our experiments we used simple random search for simplicity and speed. Here we sample K candidate plans $\mathbf{u}_{1:H-1}$ from a proposal distribution and take the one that performs best among those that satisfy constraints. If no plan can satisfy the constraints, we pick the one with the minimal violation. This crude algorithm does not guarantee constraint satisfaction and cannot handle chance constraints, but we have found it to work well for our problems.

5. Experiments

We experiment in settings with different levels of complexity and realism. For a detailed overview of each experiment, please refer to our extended technical report (Kayalibay et al., 2023).

5.1. Toy Scenario

We start with a toy experiment that demonstrates our ideas in a simple setting. We visualise this problem in fig. 1 (a). We have a 2D agent that tries to reach the green box on the western side of the room. The cost is the distance to the goal zone and the agent can observe its location with some additive Gaussian noise that is significantly higher inside a circle in the center (marked by the grey circle in fig. 1 (a)). The agent can control its velocity subject to additive noise.

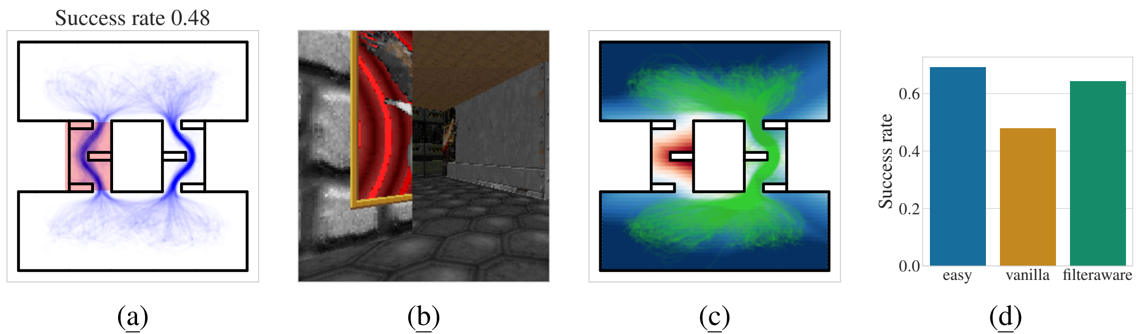


Figure 2: ViZDoom Setup. The agent starts in one room and must reach the other. (a) Vanilla MPC picks the left corridor where observations are corrupted. (b) Sample observation. (c) Trackability function and filter-aware rollouts. (d) Vanilla MPC on an easy problem where both corridors are safe vs vanilla MPC on the harder problem vs filter-aware MPC on the harder problem, where the left corridor corrupts observations.

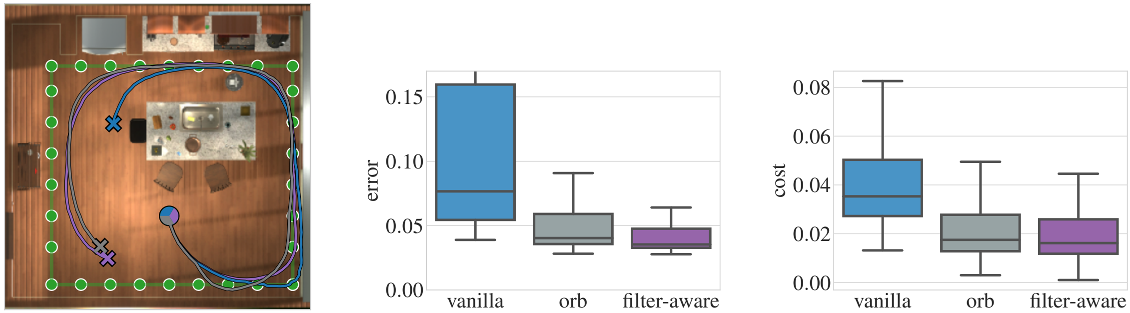
An easier problem, where the observation noise is the same everywhere and has a scale of 0.03, can be solved by a model-predictive controller and a bootstrap particle filter with 93% success, as shown in fig. 1 (d). If we increase the observation noise to 1.0 inside the grey circle, the controller fails, because MPC leads the agent inside the circle, where the particle filter is no longer able to keep track. We can learn the trackability with a neural network, as outlined in section 4. For that, we use 500 rollouts of the MPC policy with 30 time steps each. We plot the network's output in fig. 1 (b), which reveals that the network can separate the problem area from the safe zones. If we use the network to formulate constraints, the agent starts avoiding the grey circle and taking a slightly longer but safer route, as shown in fig. 1 (c).

We compare filter-aware MPC against vanilla MPC and an RNN policy in fig. 1 (d). The RNN policy is trained by gradient descent on a differentiable implementation of this system to minimise the total cost within a horizon of 50. We find that filter-aware MPC alone recovers the original success rate under this harder version of the problem, outperforming both vanilla MPC and the RNN, which can theoretically implement belief planning, as it has access to the full interaction history.

5.2. Navigation in ViZDoom

We now turn to a visual navigation problem based on the ViZDoom simulator (Wydmuch et al., 2018). Here we use coloured point-to-plane ICP (Chen and Medioni, 1992; Steinbrücker et al., 2011; Audras et al., 2011) as our state estimator since it is a well-established method for visual tracking. The ViZDoom environment has two rooms that are connected by two corridors. The agent is placed in a random location in one room and must go to a random location in the other room. It can see the world with an RGB-D camera and moves by picking a turning angle and a speed which is applied along its facing direction. These controls are perturbed by noise, making the dynamics stochastic. We show a sample RGB image in fig. 2 (b).

We make the problem harder by corrupting observations in the left corridor, analogous to the grey circle from our toy experiment. This time, we zero-out the agent's RGB-D observation in the left corridor, meaning the agent can only rely on its transition model. Our goal here is to simulate issues that plague visual tracking in the real world such as poor lighting conditions or reflective surfaces. We show how navigation with vanilla MPC works under these conditions in fig. 2 (a). The



(a) (b) (c)

Figure 3: Orbit task. (a) Top-down view. The agent must follow the green landmarks. Blue is the average vanilla MPC trajectory, purple the filter-aware one, grey the ORB baseline. (b) Box plots for state estimation errors. (c) Box plots for task cost.

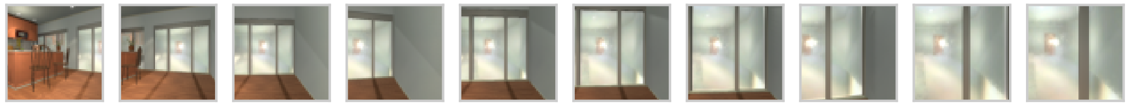
agent selects both corridors equally, even though the left corridor has poor observability, leading to a success rate of 48%.

Filter-aware MPC increases the success rate to 64%. We learn the trackability using 900 rollouts of the MPC policy, having 200 time steps each. We plot the learned trackability function and a set of filter-aware rollouts in fig. 2 (c). The filter-aware controller avoids the left corridor in all cases. We compare filter-aware MPC, vanilla MPC and vanilla-easy, which is vanilla MPC run on a setup where both corridors are safe in fig. 2 (d). Filter-aware MPC outperforms vanilla MPC by 16% when both are run on the same problem and approaches the performance of vanilla-easy, which achieves a success rate of 69%.

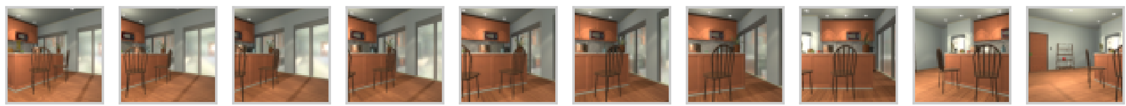
5.3. Orbiting in a Realistic Environment

The next experiment is designed to test filter-aware MPC under realistic conditions with naturally occurring features that are dangerous for the state estimator. We use the AI2-THOR simulator (Kolve et al., 2017) which features models of realistic living spaces. Real-life environments contain areas that are harder for a visual tracker to work with: blank walls, reflective surfaces, areas that cause sensor interference. We experiment with this type of situation in a navigation-adjacent task, where a robot equipped with a visual tracker is trying to follow a fixed trajectory.

We show our test scene in fig. 3 (a). The green markers are landmarks that the agent must follow. The agent receives RGB-D images that are passed to the same visual tracker as in the ViZDoom experiments. The agent can control its turning angle and speed with added noise. Unlike in ViZDoom, the camera's facing angle can be controlled independently of its motion. The cost on the agent's position and moving angle is defined to make it follow the landmarks. As the cost does not depend on the camera's facing angle, the vanilla controller simply tries to keep the camera facing towards the agent's movement direction. The environment contains a large reflective surface in the eastern wall, which can throw off a visual tracker. A filter-aware controller should be able to detect that looking at the reflective surface is dangerous for the tracker and look elsewhere, even while the agent is moving towards the reflective surface. We find this to be the case, as shown in fig. 4 (a–b). Here, the trackability is learned from 4500 rollouts of the MPC policy with length 30. While the naive controller looks where it moves, filter-aware MPC detects that doing so will cause the tracker to fail and instead looks towards the center of the room. In practice, looking where you



(a)



(b)

Figure 4: Orbit task. (a) The naive strategy looks directly at the reflective surface, causing tracking failure. (b) Filter-aware MPC avoids looking at the reflective surface.

move would be necessary to avoid obstacles. That can still be done with a filter-aware controller by using additional sensors, like a depth sensor or an additional camera for detecting obstacles.

We compare filter-aware MPC against two baselines: a) vanilla MPC b) an engineered baseline which uses ORB features (Rublee et al., 2011) as a proxy for trackability. The ORB baseline learns a neural network to predict the number of ORB features that will be visible given a camera pose and uses that value as a constraint during control, analogous to filter-aware MPC. We designed this baseline to resemble methods that focus on visual navigation (Falanga et al., 2018; Rahman and Waslander, 2020). The number of ORB features will be low when the camera is pointed at blank walls or the reflective surface on the eastern wall, and high when it is pointed towards the center of the room, which contains many strong visual landmarks. Following areas densely covered by ORB features should therefore make it easier for the visual tracker to estimate the agent's state.

In fig. 3 we draw a comparison between the three methods in terms of state estimation errors (b) and task cost (c) based on a set of 100 rollouts of length 200. Both the ORB baseline and filter-aware MPC improve significantly over the vanilla strategy, with the filter-aware controller performing the best in terms of state estimation. We show average rollouts for each method in fig. 3 (a). Both the ORB baseline and filter-aware MPC complete the course, with filter-aware having a slight edge.

5.4. Planar Two-Link Arm with Occluded Regions

In our final experiment we use a two-link robot arm that tries to reach random targets, based on the Brax (Freeman et al., 2021) implementation of the classic reacher environment.

The agent can observe its joint angles and a vector pointing from the tip of the arm to the target. To make state estimation more difficult, we set all observations to zero whenever the agent's arm enters the left-hand part of the work space (detected by checking the x-coordinate of a set of points on the arm). The agent is never started inside this region and the targets are always on the right-hand side such that the agent can avoid losing track of its state if it is careful. We add Gaussian noise to the controls before they are applied. We use a bootstrap particle filter for state estimation. We learn the trackability from 5000 rollouts of length 50 using the MPC policy.

We compare filter-aware MPC, vanilla MPC and an RNN policy in fig. 5 (a). The RNN policy follows the implementation of Ni et al. (2021). Both the RNN policy and filter-aware MPC improve over vanilla MPC, though the RNN has an edge here. We also present the performance of vanilla MPC in an easy setup where observations are available everywhere. Note that the RNN policy

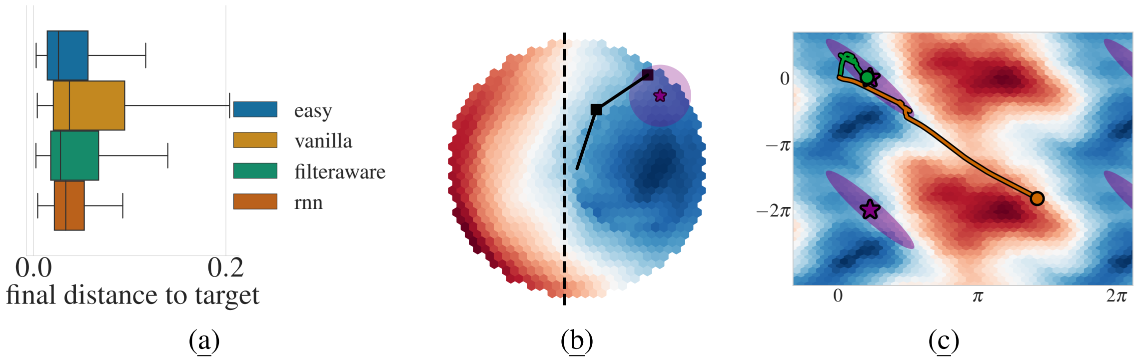


Figure 5: Reacher task. (a) Box plots. "easy" is vanilla MPC used on an easier problem. (b) Top-down view. The circle shows a radius of 0.05 around the target. Observations are missing beyond the dashed line. The colour indicates trackability. (c) A filteraware (green) vs a vanilla rollout (orange) in configuration space. The ellipse shows a radius of 0.05 around the target. Vanilla MPC enters the danger zone and loses track of itself.

performs better than vanilla MPC even when vanilla MPC is used on the easy problem. This suggests that the performance gap between filter-aware MPC and the RNN might be due to the difference between MPC and amortised policy learning. In other words, MPC itself appears to be at a general disadvantage compared to an RNN policy in this problem.

We can inspect the learned trackability function in fig. 5 (b). The network separates the blind zone from the safe area. In fig. 5 (c), we compare a filter-aware rollout and a regular one in the configuration space of the arm. Regular MPC crosses into the blind zone and cannot recover.

6. Limitations of Filter-Aware MPC

Filter-aware MPC reduces state estimation errors by avoiding plans that are difficult to track. We do this by placing constraints on the plan. Any constrained controller can work worse than the unconstrained version if the constraints are too limiting. We picked thresholds for trackability by hand in each experiment, relying on visualisations of trackability over the state space. This is only possible in problems that can be visualised well in at most three dimensions. In higher-dimensional settings the thresholds would have to be picked by hyper-parameter search.

The core assumption of our method is that we can learn trackability from rollouts. We assumed that either the true environment model or a reasonably accurate approximate model is available, and used the true system in each of our experiments.

7. Conclusion

We introduced filter-aware MPC, an improvement over vanilla MPC for partially observable problems. Filter-aware MPC constrains the planner such that it preserves a certain level of state estimation accuracy. This bridges the gap between regular MPC, which only tries to reduce the cost without reasoning about future beliefs, and belief planning, which is often too expensive. Filter-aware MPC depends on a measure of trackability which can be learned by a neural network, enabling filter-aware planning with little additional cost. In experiments on increasingly realistic problems, we have tested filter-aware MPC and found that it consistently improves over vanilla MPC.

References

Karl Johan Åström. Optimal control of Markov processes with incomplete state information. Journal of Mathematical Analysis and Applications, 10(1):174–205, 1965.

Cedric Audras, A Comport, Maxime Meilland, and Patrick Rives. Real-time dense appearance-based SLAM for RGB-D sensors. In Australasian Conf. on Robotics and Automation, volume 2, pages 2–2, 2011.

Philip Becker-Ehmck, Maximilian Karl, Jan Peters, and Patrick van der Smagt. Learning to fly via deep model-based reinforcement learning. CoRR, abs/2003.08876, 2020. URL https://arxiv.org/abs/2003.08876.

Dimitri P. Bertsekas. Dynamic programming and optimal control, 3rd Edition. Athena Scientific, 2005. ISBN 1886529264. URL http://www.worldcat.org/oclc/314894080.

Christoph Böhm. Avoidance of poorly observable trajectories: A predictive control perspective, 07 2008.

A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact algorithm for partially observable Markov decision processes. In Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI), pages 54–61, 1998.

Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. Image and vision computing, 10(3):145–155, 1992.

Stuart E Dreyfus. Dynamic programming and the calculus of variations. Technical report, RAND CORP SANTA MONICA CA, 1965.

Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. PAMPC: perception-aware model predictive control for quadrotors. CoRR, abs/1804.04811, 2018. URL http://arxiv.org/abs/1804.04811.

Marcello Farina, Luca Giulioni, and Riccardo Scattolini. Stochastic linear model predictive control with chance constraints–a review. Journal of Process Control, 44:53–67, 2016.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, pages 2555–2565, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. CoRR, abs/2010.02193, 2020. URL https://arxiv.org/abs/2010.02193.

Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1lL4a4tDB.

Ruijie He, Sam Prentice, and Nicholas Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In 2008 IEEE International Conference on Robotics and Automation, pages 1814–1820, 2008. doi: 10.1109/ROBOT.2008.4543471.

Morten Hovd and Robert Bitmead. Interaction between control and state estimation in nonlinear MPC, 11 2005.

Maximilian Igl, Luisa M. Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In Jennifer G. Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 2122–2131. PMLR, 2018. URL http://proceedings.mlr.press/v80/igl18a.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. Artif. Intell., 101(1-2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL https://doi.org/10.1016/S0004-3702(98)00023-X.

Maximilian Karl, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt, and Justin Bayer. Unsupervised real-time control through variational empowerment, 2017.

Baris Kayalibay, Atanas Mirchev, Ahmed Agha, Patrick van der Smagt, and Justin Bayer. Filter-aware model-predictive control, 2023. URL https://arxiv.org/abs/2304.10246.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. arXiv, 2017.

Johannes Köhler, Matthias A. Müller, and Frank Allgöwer. Robust output feedback model predictive control using online estimation bounds, 2021. URL https://arxiv.org/abs/2105.03427.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. arXiv preprint arXiv:1907.00953, 2019.

Michael Lederman Littman. Algorithms for sequential decision making, 1996.

Ali Mesbah. Stochastic model predictive control with active uncertainty learning: A survey on dual control. Annual Reviews in Control, 45:107–117, 2018. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2017.11.001. URL https://www.sciencedirect.com/science/article/pii/S1367578817301232.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps, 2021. URL https://arxiv.org/abs/2110.05038.

R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03, page 1025–1030, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

Robert Platt, Russell Louis Tedrake, Leslie P. Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In Proceedings of the Robotics: Science and Systems Conference (RSS), 2010.

Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In Proceedings of the 12th International Symposium of Robotics Research (ISRR), 2009.

Mohammadhussein Rafieisakhaei, Suman Chakravorty, and P. R. Kumar. T-lqg: Closed-loop belief space planning via trajectory-optimized lqg. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 649–656, 2017. doi: 10.1109/ICRA.2017.7989080.

Shatil Rahman and Steven L. Waslander. Uncertainty-constrained differential dynamic programming in belief space for vision based robots. CoRR, abs/2012.00218, 2020. URL https://arxiv.org/abs/2012.00218.

Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robots. In S. Solla, T. Leen, and K. Müller, editors, Advances in Neural Information Processing Systems, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/file/df9028fcb6b065e000ffe8a4f03eeb38-Paper.pdf.

Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), 1:35–40 vol.1, 1999.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In 2011 International Conference on Computer Vision, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.

Katsushige Sawaki and Akira Ichikawa. Optimal control for partially observable Markov decision processes over an infinite horizon. Journal of the Operational Research Society, 21, 1978.

E. J. Sondik. The optimal control of partially observable Markov decision processes over the infinite horizon: Discounted costs. Operations Research, 26, 1978.

Edward Sondik. The optimal control of partially observable decision processes, 01 1971.

Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense RGB-D images. In 2011 IEEE international conference on computer vision workshops (ICCV Workshops), pages 719–722. IEEE, 2011.

Zachary Sunberg and Mykel J. Kochenderfer. POMCPOW: an online algorithm for pomdps with continuous state, action, and observation spaces. CoRR, abs/1709.06196, 2017. URL http://arxiv.org/abs/1709.06196.

Zachary N. Sunberg, Christopher J. Ho, and Mykel J. Kochenderfer. The value of inferring the internal state of traffic participants for autonomous freeway driving. In 2017 American Control Conference (ACC), pages 3004–3010, 2017. doi: 10.23919/ACC.2017.7963408.

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

Sebastian Thrun. Monte Carlo POMDPs. In Proceedings of the 12th International Conference on Neural Information Processing Systems, NIPS'99, page 1064–1070, Cambridge, MA, USA, 1999. MIT Press.

Emanuel Todorov. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. Neural Comput., 17(5):1084–1108, may 2005. ISSN 0899-7667. doi: 10.1162/0899766053491887. URL https://doi.org/10.1162/0899766053491887.

Jur van den Berg, Sachin Patil, and Ron Alterovitz. Efficient approximate value iteration for continuous gaussian pomdps. Proceedings of the AAAI Conference on Artificial Intelligence, 26 (1):1832–1838, Sep. 2021. doi: 10.1609/aaai.v26i1.8371. URL https://ojs.aaai.org/index.php/AAAI/article/view/8371.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. ViZDoom competitions: Playing doom from pixels. IEEE Transactions on Games, 2018.

N. L. Zhang and W. Zhang. Speeding up the convergence of value iteration in partially observable Markov decision processes. Journal of Artificial Intelligence Research, 14:29–51, feb 2001. doi: 10.1613/jair.761. URL https://doi.org/10.1613%2Fjair.761.

Dongliang Zheng, Jack Ridderhof, Panagiotis Tsiotras, and Ali-akbar Agha-mohammadi. Belief space planning: A covariance steering approach. CoRR, abs/2105.11092, 2021. URL https://arxiv.org/abs/2105.11092.