Informed Information Theoretic Model Predictive Control

Raphael Kusumoto[1] and Luigi Palmieri[2] and Markus Spies[3] and Akos Csiszar[1] and Kai. O. Arras[2]

***Abstract*— The problem of minimizing cost in nonlinear control systems with uncertainties or disturbances remains a major challenge. Model predictive control (MPC), and in particular sampling-based MPC has recently shown great success in complex domains such as aggressive driving with highly nonlinear dynamics. Sampling-based methods rely on a prior distribution to generate samples in the first place. Obviously, the choice of this distribution highly influences efficiency of the controller. Existing approaches such as sampling around the control trajectory of the previous time step perform suboptimally, especially in multi-modal or highly dynamic settings. In this work, we therefore propose to learn models that generate samples in low-cost areas of the state-space, conditioned on the environment and on contextual information of the task to solve. By using generative models as an informed sampling distribution, our approach exploits guidance from the learned models and at the same time maintains robustness properties of the MPC methods. We use Conditional Variational Autoencoders (CVAE) to learn distributions that imitate samples from a training dataset containing optimized controls. An extensive evaluation in the autonomous navigation domain suggests that replacing previous sampling schemes with our learned models considerably improves performance.**

I. Introduction

Fast and reliable local motion planning is a key technique for efficiently accomplishing robot navigation tasks. The challenge is to compute smooth controls to steer autonomous systems to the desired goal, taking into account dynamics and additional constraints such as closeness to the global path, while avoiding un-foreseen obstacles. Model predictive control (MPC) [14], [13] is an efficient technique to perform local motion planning and control by solving a receding-horizon model-based open-loop optimal control problem. Classical MPC techniques work well when the goal is the stabilization of constrained systems around some equilibrium points or trajectories [2], [6], [19]. Modern numerical solvers can solve efficiently (non-)linear constrained optimization problems for convex cost function and accurate approximation of the system dynamics [12]. Lately several algorithms have extended MPC with machine learning [10], [26] to solve more challenging problems coming from the robotics domain with more difficult nonlinear dynamics and interactions with the environment.

Recently a novel information-theoretic approach to MPC (IT-MPC) [25] has been introduced to overcome some of the natural limitations of standard (non-)linear MPC techniques. Contrarily to standard MPC techniques, IT-MPC can work considering arbitrary system dynamics and cost functions. The approach shifts its attention to stochastic domains and extends the path integral control method (MPPI) [22]. IT-MPC generates open-loop sequences of sampled controls by minimizing the Kullback–Leibler (KL) divergence between the current control distribution and the optimal one, derived by the desired cost-function. The control scheme converges to the optimal distribution as the number of samples increases. The authors show in [25] that IT-MPC can achieve good performance in an aggressive-driving scenario. Control samples for IT-MPC are generated in [25] by a normal distribution centered around the previous control sequence. This method performs well as long as the optimal control sequence only changes slightly from one step to the other. If this is not the case, for example due to a new goal location or dynamic obstacle, local sampling around the nominal trajectory leads to poor convergence to the optimal control sequence.

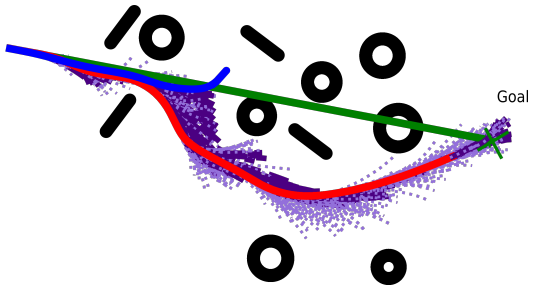


Fig. 1: Example path (in red), controls (in dark purple) and samples in (light purple) generated by our approach for a wheeled mobile robot considering control uncertainties. With our method the robot is capable of successfully navigating among un-foreseen obstacles and following the obstacle-unaware global path (in green) differently from the IT-MPC (its resulting path in blue), which could not find a solution.

To overcome these limitations, in this work we propose to extend IT-MPC by adding an informed sampling process, i.e., a learned distribution, that is aware of the robot dynamics, of the environment and of the global task to accomplish. We use Conditional Variational Autoencoders (CVAE) [21] to learn distributions that imitate samples from a training dataset containing optimized controls. Sampling from this distribution leads to improved performance since less samples fall in costly parts of the state space. Furthermore, to learn task-driven robot behaviors, differently from its original formulation, we propose a novel loss function for the CVAE.

[1]Raphael Kusumoto and Akos Csiszar are with ISW, University of Stuttgart, raphael_kba@hotmail.com, akos.csiszar@isw.uni-stuttgart.de

[2]L. Palmieri and K. O. Arras are with Bosch Corporate Research, Stuttgart, Germany, {luigi.palmieri,kaioliver.arras}@de.bosch.com

[3]M. Spies is with Bosch Center for Artificial Intelligence, Stuttgart, Germany, markus.spies2@de.bosch.com

The latter considers also the environmental conditions in its reconstruction error. In a series of experiments, we show that our approach outperforms a set of baselines in different environments in terms of final trajectory cost, traveled distance and task success.

The paper is structured as follows: in Section II we detail the related work. In Sections III, IV we introduce our approach. Section V describes the setup of experiments and Section VI discusses the obtained results.

II. Related Work

Several approaches adopt MPC-based techniques for solving robotics tasks in combination with machine learning [10], [26], [3]. In [10] the authors combine an MPC controller with a deep architecture for physical prediction of the robot model. Similarly, we learn the model used for the optimization but use the IT-MPC framework for generating motions. In [26] an MPC algorithm is used in combination with guided policy search [11] for the offline learning of control policy. Differently, our approach uses IT-MPC for the on-line system control and offline learning of the environment-aware sampling distribution. Williams et al. [24] propose a robust IT-MPC framework, a Tube-MPC approach for solving general optimal control problems, with a sampling based nominal controller. Its generality allows our approach to be extended in a future work to this framework. Drews et al. [3] combine MPPI with a deep network that predicts future cost maps directly from images taken from a monocular camera. On the contrary, in our approach we learn a context and task aware distribution that generates samples towards promising areas of the state space and use a hand-crafted cost function.

Learning sampling distributions in the context of global motion planning has recently received a lot of interest [7], [16], [18], [8]. Ichter et al. [7] use CVAE for learning sampling distributions for sampling-based motion planning. This work considers as conditional variables of the CVAE obstacles encoded in a grid map, start and goal poses. CVAEs are also adopted in [18], [8] to learn multi-modal probability distributions of human-human interactions. Differently from [7] they use recurrent neural networks (RNN) and long short-term memory (LSTM) to model the encoder and decoder part of the CVAE. Perez et al. [16] use a fully convolutional neural network to enhance sampling-based motion planning capabilities: in particular the sampling distribution is biased towards areas where the network predicts a path. As in [7], [18], [8], we use a CVAE for learning a sampling distribution, that differently from the latter methods, will directly generate controls based on system dynamics, the context and the task to solve. This is achieved by introducing a novel loss function for CVAEs. Moreover, as opposed to [7], [18], [8], we use the learned distribution in an MPC-based local planner.

III. Preliminaries

In this section we introduce preliminaries needed to describe our approach: the Information Theoretic-MPC framework in Sec. III-A and Conditional Variational Autoencoders in Sec. III-B.

A. Information Theoretic-MPC

Here we recap the Information Theoretic-Model Predictive Control (IT-MPC) approach [25] that considers stochastic nonlinear discrete time systems of the form $\boldsymbol{x}_{t+1} = F(\boldsymbol{x}_t, \boldsymbol{v}_t)$, where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the system state at time t of dimension n, $\boldsymbol{v}_t \in \mathbb{R}^m$ is the input control variable of dimension m with white noise applied to nominal control $\boldsymbol{u}_t$, i.e., $\boldsymbol{v}_t = \boldsymbol{u}_t + \boldsymbol{\varepsilon}_t$ with $\varepsilon_t \sim \mathcal{N}(0, \boldsymbol{\Sigma})$. Let us define the input sequence as $(\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t_\mathrm{f}-1}) = \boldsymbol{V} \in \mathbb{R}^{m \times t_\mathrm{f}}$, its mean input as $(\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots \boldsymbol{u}_{t_\mathrm{f}-1}) = \boldsymbol{U} \in \mathbb{R}^{m \times t_\mathrm{f}}$ and the disturbance as $(\varepsilon_0, \varepsilon_1, \ldots \varepsilon_{t_\mathrm{f}-1}) = \mathscr{E}$. We denote the resulting joint probability distribution that generates $\boldsymbol{V}$ as $\mathbb{Q}_{\boldsymbol{U},\boldsymbol{\Sigma}}$.

The goal of IT-MPC is to solve the stochastic optimal control problem of the form

$$\boldsymbol{U}^* = \underset{\boldsymbol{U} \in \mathscr{U}}{\arg\min}\; E_{\mathbb{Q}_{\boldsymbol{U},\boldsymbol{\Sigma}}} \left(\sum_{t=0}^{t_\mathrm{f}-1} c(\boldsymbol{x}_t) + \phi(\boldsymbol{x}_{t_\mathrm{f}}) + \lambda (\boldsymbol{u}_t^\mathrm{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{u}_t) \right), \tag{1}$$

where $\mathscr{U}$ is the set of possible input sequences for the system, $c(\boldsymbol{x}_t)$ is a state based cost, $\phi(\boldsymbol{x}_{t_\mathrm{f}})$ a terminal cost and $\lambda > 0$. Williams et al. [25] derive the optimal control distribution $\mathbb{Q}^*$ as

$$q^*(\boldsymbol{V}) = \frac{1}{\eta} \exp -\frac{1}{\lambda} S(\boldsymbol{V}) p(\boldsymbol{V} | \boldsymbol{0}, \boldsymbol{\Sigma}) \tag{2}$$

where $S(\boldsymbol{V})$ denotes the state cost of an entire trajectory, p is a Gaussian with covariance $\boldsymbol{\Sigma}$ and η is a normalization factor. This optimal control distribution is not necessarily Gaussian, depending on the state cost function. To approximate the original problem (1), the authors propose to minimize the KL divergence between the optimal control distribution $\mathbb{Q}^*$ and $\mathbb{Q}_{\boldsymbol{U},\boldsymbol{\Sigma}}$

$$\boldsymbol{U}^* \approx \underset{\boldsymbol{U} \in \mathscr{U}}{\arg\min}\; \mathbb{D}_{KL}(\mathbb{Q}^*, \mathbb{Q}_{\boldsymbol{U},\boldsymbol{\Sigma}}). \tag{3}$$

In case the optimal control distribution is Gaussian, the approximation is correct and the resulting control trajectory is optimal. Given samples $\boldsymbol{V}_k$ around a nomial control trajectory $\hat{\boldsymbol{U}}$ as described above with disturbances $\mathscr{E}_k$, Eq. 3 can be solved by an importance sampling scheme

$$\boldsymbol{U}^* \approx \hat{\boldsymbol{U}} + \left(\sum_{k=1}^{K} w_k \mathscr{E}_k \right) \tag{4}$$

where w_k are importance sampling weights. Even though in theory (4) approximates the optimal control distribution in one shot, in practice it is more robust to use (4) as an update rule in an iterative manner. The resulting algorithm is detailed in Algorithm 1, without the blue line. For a derivation of the importance sampling weights and more details about the overall algorithm we refer to the original paper [25].

B. Learning sampling distributions with CVAE

In this work we will use Conditional Variational Autoencoders [21] to learn a model that can be used to generate informed sampling distributions for IT-MPC-based controllers, i.e., to provide samples in low-cost areas of the state space to improve efficiency of the controller in the desired task.

Algorithm 1 Informed IT-MPC algorithm. Blue represents the pseudocode to remove to obtain the original IT-MPC.

Input: $\boldsymbol{U}^*$: Initial control sequence, K: Number of samples, t_f: Time horizon, F: Transition model, ϕ, c: State cost Σ, λ: Hyper-parameter, $\boldsymbol{C}$ decoder conditions
while task not completed **do**
  $\boldsymbol{x}_o \leftarrow$ GetStateEstimate()
  $\boldsymbol{U}^* \leftarrow$ InformControls($\boldsymbol{x}_0$, $\boldsymbol{u}^*$, $\boldsymbol{C}$, Σ, ϕ, c, F)
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
    $\boldsymbol{S}_k \leftarrow 0$
    $\mathscr{E}^k \leftarrow (\varepsilon_0^k, \ldots, \varepsilon_{t_\mathrm{f}-1}^k), \varepsilon_t^k \in \mathscr{N}(0, \Sigma)$
    for $t \leftarrow 1$ **to** t_f **do**
      $\boldsymbol{x} \leftarrow F(\boldsymbol{x}, \boldsymbol{u}_{t-1} + \varepsilon_{t-1}^k)$
      $\boldsymbol{S}_k \leftarrow \boldsymbol{S}_k + c(\boldsymbol{x}) + \lambda \boldsymbol{u}_{t-1}^\mathrm{T} \boldsymbol{\Sigma}^{-1} \varepsilon_{t-1}^k$
    end for
    $\boldsymbol{S}_k \leftarrow \boldsymbol{S}_k + \phi(\boldsymbol{x})$
    $\boldsymbol{w}_k \leftarrow$ ImportanceSamplingWeights($\boldsymbol{S}_k$, λ)
    $\boldsymbol{U}^* \leftarrow \boldsymbol{U}^* + \sum_{k \leftarrow 1}^K \boldsymbol{w}_k \mathscr{E}^k$
  end for
  ApplyControl($\boldsymbol{u}_0^*$)
  for $t \leftarrow 1$ **to** $t_\mathrm{f} - 1$ **do**
    $\boldsymbol{u}_{t-1}^* \leftarrow \boldsymbol{u}_t^*$
  end for
  $\boldsymbol{u}_{t_\mathrm{f}-1}^* \leftarrow$ Initialize($\boldsymbol{u}_{t_\mathrm{f}-1}^*$)
end while

Algorithm 2 Generate control trajectory using the learned CVAE model.

Input: $\boldsymbol{x}_0$, $\boldsymbol{U}^*$, $\boldsymbol{C}$, F, ϕ, c, Σ, λ
$\hat{\boldsymbol{U}} \leftarrow$ CVAEDecoder($\boldsymbol{C}$)
$\hat{\boldsymbol{x}}, \boldsymbol{x}^* \leftarrow \boldsymbol{x}_0$
$\hat{\boldsymbol{S}}, \boldsymbol{S}^* \leftarrow 0$
for $t \leftarrow 0$ **to** $t_\mathrm{f} - 1$ **do**
  $\hat{\boldsymbol{x}} \leftarrow F(\boldsymbol{x}, \hat{\boldsymbol{u}}_t), \boldsymbol{x}^* \leftarrow F(\boldsymbol{x}^*, g(\boldsymbol{u}_t^*))$
  $\hat{\boldsymbol{S}} \leftarrow \hat{\boldsymbol{S}} + c(\hat{\boldsymbol{x}}) + \lambda \hat{\boldsymbol{u}}_t^\mathrm{T} \boldsymbol{\Sigma}^{-1} \hat{\boldsymbol{u}}_t$, $\boldsymbol{S}^* \leftarrow \boldsymbol{S}^* + c(\boldsymbol{x}^*) + \lambda \boldsymbol{u}_t^{*\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{u}_t^*$
end for
$\hat{\boldsymbol{S}} \leftarrow \hat{\boldsymbol{S}} + \phi(\hat{\boldsymbol{x}})$, $\boldsymbol{S}^* \leftarrow \boldsymbol{S}^* + \phi(\boldsymbol{x}^*)$
if $\hat{\boldsymbol{S}} < \boldsymbol{S}^*$ **then**
  $\boldsymbol{U}^* \leftarrow \left(1 - \frac{\hat{\boldsymbol{S}}}{\hat{\boldsymbol{S}} + \boldsymbol{S}^*}\right) \hat{\boldsymbol{U}} + \left(1 - \frac{\boldsymbol{S}^*}{\hat{\boldsymbol{S}} + \boldsymbol{S}^*}\right) \boldsymbol{U}^*$
end if
return $\boldsymbol{U}^*$

A Conditional Variational Autoencoder (CVAE) is trained to imitate a distribution of observed data $\boldsymbol{X}^{(i)} \in \mathbb{R}^{N_x}$ conditioned on $\boldsymbol{C} \in \mathbb{R}^{N_b}$ using an unobserved, latent representation $\boldsymbol{Z} \in \mathbb{R}^{N_z}$, i.e., $p(\boldsymbol{X}|\boldsymbol{C}) = \int_{\boldsymbol{z}} p(\boldsymbol{X}|\boldsymbol{z},\boldsymbol{C}) p(\boldsymbol{z}|\boldsymbol{C}) d\boldsymbol{z}$ of the states. Fig. 2 illustrates the two components of a CVAE, an encoder and a decoder. The encoding process finds a parametric function (e.g., a neural network) that maps the input $\boldsymbol{X}$ and conditions $\boldsymbol{C}$ to a normal distribution $q(\boldsymbol{z}|\boldsymbol{X},\boldsymbol{C})$ with a mean μ and variance Σ in the latent space $\boldsymbol{z}$. The decoder can be intuitively interpreted as the reverse process of encoding. It finds a parametric function that given a latent variable and conditions computes samples from the input distribution, that is $p(\boldsymbol{X}|\boldsymbol{z},\boldsymbol{C})$. The main idea of CVAE is not to directly maximize the marginal likelihood $\log p(\boldsymbol{X}^{(i)}|\boldsymbol{C})$, but its Variational Lower Bound of the form

$$\begin{aligned}\log p(\boldsymbol{X}^{(i)}|\boldsymbol{C}) \geq & -\mathbb{D}_{KL}[q(\boldsymbol{z}|\boldsymbol{X}^{(i)},\boldsymbol{C})||p(\boldsymbol{z}|\boldsymbol{C})] \\ & + \mathbb{E}_{q_{(\boldsymbol{z}|\boldsymbol{X}^{(i)},\boldsymbol{C})}}[\log(p(\boldsymbol{X}^{(i)}|\boldsymbol{z},\boldsymbol{C}))].\end{aligned} \tag{5}$$

Here, the first term on the right-hand side is a KL-Divergence between the distribution of $q(\boldsymbol{z}|\boldsymbol{X}^{(i)},\boldsymbol{C})$ and the desired latent space distribution $p(\boldsymbol{z}|\boldsymbol{C})$. The log-likelihood of $p(\boldsymbol{X}^{(i)}|\boldsymbol{z},\boldsymbol{C})$ can be interpreted as the reconstruction loss. After training, by sampling in the latent space, the decoder can be used to generate samples from $p(\boldsymbol{X}|\boldsymbol{z},\boldsymbol{C})$, which is an approximation of the desired distribution $p(\boldsymbol{X}|\boldsymbol{C})$.

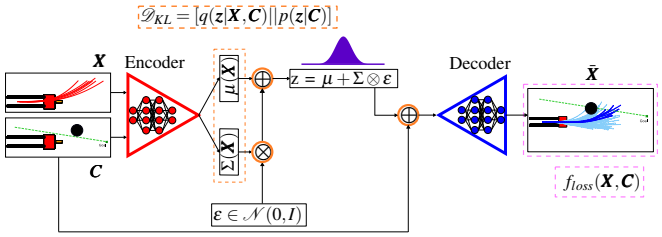


Fig. 2: Scheme of the CVAE, in which the decoder generates an approximation $\tilde{\boldsymbol{X}}$ of the given inputs $\boldsymbol{X}$. The network receives two concatenated vectors with the input values and the conditions $\boldsymbol{C}$. The encoder and decoder are both composed of four fully connected layers with a ReLU as the activation function, each circle in the figure represents 100 nodes. Texts in the dotted boxes describe components of the loss function.

IV. Informed Information Theoretic MPC

The goal of IT-MPC is to find a control distribution that minimizes the expected cost (1). This distribution depends strongly on the the cost function, which in turn depends on contextual information such as the desired global path or obstacles. The sampling-based approach (4) is more accurate if the initial sampling distribution is already close to the optimal one. Instead of naive initialization such as zero- or constant velocity we propose to inform and guide the sampling distribution towards low-cost areas of the state-space by using offline learned context- and task-aware generative models.

In the following, we first describe how to learn informed control distributions from data and then present how we combine the learned generative models with IT-MPC.

A. Learning Informed Control Distributions

The main idea of our approach is to generate control samples that optimize the original objective. Given environmental conditions, we train a CVAE to output a distribution that is close to the training samples, i.e., that minimizes the reconstruction loss in (5). However, it needs to be defined what "close" means in terms of trajectories. Assuming Gaussian control distributions, the log probability in (5) can be computed using Euclidean distances between the input samples and the output [7], [18]. In contrast to this, we use a reconstruction loss that directly captures the original cost function, i.e., low cost corresponds to high probability, similar to [23], [1]. To this end we use a domain-specific loss function $f_{loss}(\boldsymbol{X},\boldsymbol{C})$. The resulting loss we use for training is thus

$$\mathscr{L}_{itmpc} = \mathbb{D}_{KL}[q(\boldsymbol{z}|\boldsymbol{X},\boldsymbol{C})||p(\boldsymbol{z}|\boldsymbol{C})] + f_{loss}(\boldsymbol{X},\boldsymbol{C}), \tag{6}$$

where the first term is the KL-divergence in latent space as in (5). In Sec. V we will provide details on the loss function $f_{loss}(\boldsymbol{X},\boldsymbol{C})$ for specific applications. Since we use

loss functions that depend on the environmental context, the cost and thus on the state $\boldsymbol{x}$, we need to have access to an analytical form of the system dynamics for backpropagating gradients. This is in contrast to using log probability that is computed directly on the controls.

B. Combining IT-MPC with Informed Sampling

The original IT-MPC sampling scheme computes the optimal control in the limit, if the assumptions described in Section III-A hold. This result is independent of the initial distribution. In practice, however, the initial distribution considerably affects the efficiency of the method due to a limited number of samples, cost functions that lead to non-Gaussian optimal distributions and due to model inaccuracies.

In this work, we choose to use the learned models to compute the nominal trajectory, i.e., the mean of the initial distribution in a way that provides samples in low-cost areas of the state space to improve efficiency of the controller in the desired task. Modifying the nominal trajectory does not affect theoretical properties of the original algorithm itself.

Williams et al. [24] propose to use the optimized trajectory $\boldsymbol{U}^*$ as an initialization in the next iteration. We follow the same line but in addition use the learned model to allow for a more efficient initialization. We modify the original algorithm by calling `InformControl` given the current approximation of the optimal trajectory $\boldsymbol{U}^*$ (or an arbitrary initialization in the first loop), as shown in Algorithms 1 and 2. `InformControl` generates a control trajectory $\hat{\boldsymbol{U}}$ using the learned CVAE model in line 2 of Alg. 2. In particular, from the decoder based on the conditions $\boldsymbol{C}$, we draw the N_d likely samples and choose the best one $\hat{\boldsymbol{u}}$. It then computes the state cost of both $\hat{\boldsymbol{U}}$ and $\boldsymbol{U}^*$ by forward propagating system dynamics (lines 3–10). If the generated trajectory has lower cost compared to $\boldsymbol{U}^*$, the algorithm returns a weighted combination of both control sequences, with weights depending on the respective cost (lines 9–14). Otherwise, it returns the current control sequence.

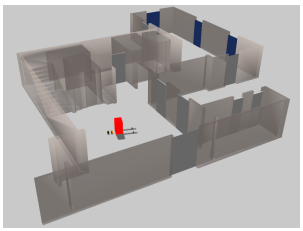
In the next section we present a quantitative comparison between previous methods and the proposed informed sampling approach, which shows the benefits of using learned control distributions.

V. Evaluation

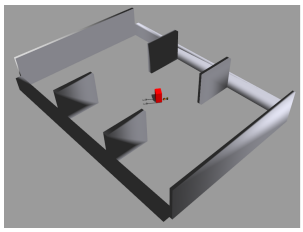
In this section we describe how we evaluate our approach in terms of path quality and planning efficiency. We detail the experiments (i.e., the simulated scenarios and the metrics) in Sec.V-A and describe the loss and cost functions in Sec. V-B.

A. Experiments

We evaluate our approach and compare it to several baselines in two different settings. In *Experiment 1*, we ask the planners to execute a path tracking task, i.e., the robot has to reach the goal and meanwhile stay as close as possible to the global path. In *Experiment 2*, we extend the previous task by also performing obstacle avoidance, while reaching a goal and tracking a global path, which was generated not considering the obstacles. For each setting we define multiple environments of varying difficulty and use



(a) Map 1 side view



(b) Map 2 side view

Fig. 3: Maps for testing our approach and the baselines in *Experiment 1*.

different f_{loss} definitions. For the experiments we consider an autonomous forklift, modeled as a front-wheel drive bicycle kinematic model as in [20], controlled with the longitudinal acceleration u_a and steering velocity u_φ, see Fig. 4-5.

1) Environments: In each experiment we test our approach and the baselines in different environments.

In *Experiment 1*, the main task for the controllers involves tracking a reference path generated by a global planner (i.e., A*). The computed path does not fully respect the dynamics of the vehicle. For the experiments we adopt two offline computed maps (see Fig. 3a, 3b), and randomly drawn initial poses and goals. For each pair of start and goal poses we perform 200 runs and average the obtained metrics. In this experiment, we compare our approach to two baselines: the original IT-MPC, and an IT-MPC based controller informed by a network trained with the standard Euclidean loss function (IT-MPC+CVAE). The latter baseline is used to show the benefit of the modification we applied to the CVAE structure and its loss function as described in Sec. III-B. For each map we use two different *setups*. The first uses 500 control samples and variance of 0.25 and for the second the samples number is reduced to 100 samples. For the inverse temperature and alpha values of 1 and 0.01 are used respectively. If the goal is not reached after 60s the simulation is considered as a failure.

In *Experiment 2*, we test the baselines on 150 environments. For each, obstacles, start and goal poses are drawn randomly. In these scenarios, we compare our method to two baselines using the same cost function (as described in Sec. V-B.2, see Eq. 7): the original IT-MPC, and a Dynamic Window based planner [4]. In these experiments 500 control samples were drawn with a variance of 1.0 for the IT-MPC and 0.25 for the IIT-MPC. The inverse temperature and alpha values remained the same as *Experiment 1*.

2) Metrics: We evaluate our approach and baselines in terms of: average time to the goal (T_g), accumulated distance to the global path (d_{acc}), average cost of the solution (C_s), maximum distance to the global path (d_{max}), success rate. In all the experiments, we consider a tangential velocity v range of $0.6\frac{\mathrm{m}}{\mathrm{s}}$ to $-0.4\frac{\mathrm{m}}{\mathrm{s}}$, a steering velocity range of $1.0\frac{\mathrm{rad}}{\mathrm{s}}$ to $-1.0\frac{\mathrm{rad}}{\mathrm{s}}$ and a control rate of 10Hz.

B. Datasets and Learning Parameters

For each experiment we collect a specific training set using state-of-the-art planning techniques and specify task-dependent loss and cost functions.

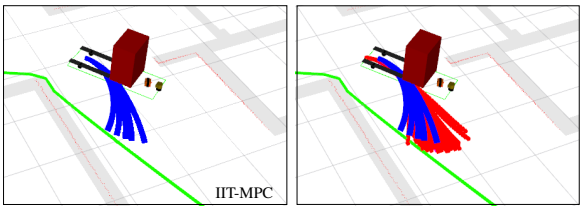


Fig. 4: Comparison between the trajectories sampled by the controls distributions, learned by IIT-MPC (blue) and computed by IT-MPC (red) for a path-tracking task. The green line represents the reference path and the grey areas correspond to obstacles. Our approach is informed about the global path positions and generates samples which are closer to it, leading to a lower cost solution.

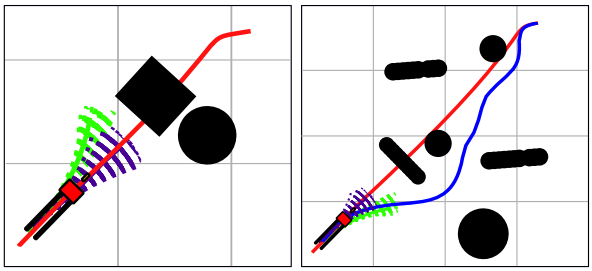


Fig. 5: The IT-MPC unimodal distribution (see **blue** samples) is mainly centered around the previous controls, while our approach generates samples (**in green**) from a context and task-aware multi-modal distribution that helps the robot to better perform obstacle avoidance and achieving the desired task (see **blue path** in the **right** figure, global path in **red**).

1) Training sets: In *Experiment 1*, we define the input $\boldsymbol{X}$ to the CVAE as a sequence of controls to perform path tracking. Conditions $\boldsymbol{C}$ are the reference path, the initial robot forward velocity and initial steering angle. Learning is performed on a dataset composed by two parts: one generated by forward propagating the best of randomly drawn controls (400000 data points) and another obtained by simulated IT-MPC robot operations (250000 samples). In this latter case, firstly an A* global path is computed and the original IT-MPC is used to generate optimal control samples.

In *Experiment 2*, the training dataset contains control samples $\boldsymbol{X}$ computed by a planning pipeline that performs obstacle avoidance and trajectory tracking. Conditions $\boldsymbol{C}$ include obstacles, goal position (taken by the global path), initial forward velocity and steering angle. The training dataset is composed of 350000 entries with different random obstacles positions, start and goal poses and global paths. For each sample, obstacle avoidance is achieved by applying initially the Informed-RRT* algorithm [5] for generating an obstacle-aware path, post-smoothed with cubic splines. Final control samples are computed by performing trajectory tracking with an LQR optimal controller. Global path and obstacles are given in x,y coordinates in the robot frame. The latter are generated by a simulated laser scanner with a field of view of 120 degrees.

2) Loss and Cost Definitions: We now detail the different loss functions used for learning and the cost functions used for motion planning in each experiment setting.

In *Experiment 1*, the planners solve a path-tracking task. The reconstruction term is defined as $f^1_{loss}(\boldsymbol{X},\boldsymbol{C}) = \frac{1}{t_f-1}\sum_{t=0}^{t_f-1}||\boldsymbol{x}_t - \mathbf{P}_t||$, with t_f being the number of path points, $\boldsymbol{x}_t$ being Cartesian point t of the local trajectory (a sample generated by the CVAE) and $\mathbf{P}_t$ Cartesian point of the global path. The cost function used for the controllers minimizes the Euclidean distance between the computed Euclidean coordinates of the local trajectory $\boldsymbol{x}_{(x,y)}$ to the reference path $\mathbf{P}$. An extra obstacle cost O_t is also given in order to avoid collision to obstacles, i.e., their positions and costs are computed from an occupancy grid map. The final cost is given by $c(\boldsymbol{x}) = ||\boldsymbol{x}_{(x,y)} - \mathbf{P}|| + O_t$. In this experiment we use a latent space dimension of 1.

In *Experiment 2*, the robot follows the global path while performing obstacle avoidance. The reconstruction term is defined as $f^2_{loss}(\boldsymbol{X},\boldsymbol{C}) = ||\boldsymbol{x}_{t_f-1} - \mathscr{G}|| + \sum_{i=0}^{N_{obst}} \frac{2}{\sigma\sqrt{2\pi}} e^{-\left(\boldsymbol{x}-\mathbf{p}^i_o\right)^2 / 2\sigma^2}$, with $\boldsymbol{x}_{t_f-1}$ being the last point of a CVAE sample, $\mathbf{p}^i_o$ the position of the obstacle i and $\mathscr{G}$ the goal position. The cost function used for motion planning is defined as follows:

$$c(\boldsymbol{x}) = w_1||\boldsymbol{x}_{t_f-1} - \mathbf{P}_{lh}|| + w_2 \sum_{i=0}^{N_{obst}} \frac{2}{\sigma\sqrt{2\pi}} e^{-\left(\boldsymbol{x}-\mathbf{p}^i_o\right)^2 / 2\sigma^2}, \quad (7)$$

with $w_1, w_2 > 0$. The first term in (7) represents the task of reaching a sub-goal $\mathbf{P}_{lh}$ selected from the global path (based on a defined lookahead distance), while in the loss function we consider a final goal $\mathscr{G}$. The second term for both functions is responsible for obstacle avoidance. Positions are represented in the robot coordinate frame. For the second experiment we have a latent space dimension of 5.

For the training of the CVAE we use the Adam optimizer, with a learning rate and weight decay of 1×10^{-5}, βs = (0.9, 0.999) and $\varepsilon = 1\times10^{-8}$. The network architecture is detailed in Fig. 2 and implemented in Pytorch [15], N_d set to 10. We use Gazebo [9], ROS [17], and implement algorithms in C++ (for *Experiment 1*) and Python (for *Experiment 2*) to simulate robot operations and evaluate metrics.

VI. Results and Discussion

In this section we discuss the results collected for both *Experiment 1* and *Experiment 2*.

A. Results Experiment 1

Firstly we consider the results collected in Tables I, II, III. In all the tables we show the results of the comparison reported as relative improvement of our approach compared to the other baseline per metric. Positive values indicate better performance for our approach. In Tables I, II collect the results concerning the comparison of our approach with IT-MPC. Table III shows the results for the comparison of IT-MPC with a simple loss function (i.e., Euclidean distance) and the classical IT-MPC. Our approach, as documented by the results in Tables I, II, achieves overall better performance than IT-MPC, thanks also to its possibility to learn context

	Map 1	
	Setup 1	Setup 2
T_g	3.76%	0.99%
d_{acc}	9.72%	8.50%
C_s	5.66%	7.36%
d_{max}	6.15%	7.19 %
Success(IIT-MPC/IT-MPC)	89.5/87.5%	92.5/80.5%

TABLE I: Results of the Experiment 1 for map 1. Comparing IT-MPC and IIT-MPC.

	Map 2	
	Setup 1	Setup 2
T_g	1.70%	1.84%
d_{acc}	3.55%	5.40%
C_s	3.16%	5.63%
d_{max}	3.01%	4.75%
Success(IIT-MPC/IT-MPC)	98.0/96.5%	96.5/97.0/%

TABLE II: Results of Experiment 1 for map 2. Comparing IT-MPC and IIT-MPC.

	Map 1	
	Setup 1	Setup 2
T_g	-3.02%	0.58%
d_{acc}	-0.83%	4.41%
C_s	4.76%	9.60%
d_{max}	1.68%	3.94 %
Success(IT-MPC + CVAE /IT-MPC)	89.0/89.0%	93.5/85.5%

TABLE III: Results of Experiment 1 for map 1. Comparing IT-MPC and IT-MPC+CVAE (i.e., Euclidean loss function).

and task-aware multi-modal distributions see also Figures 4, 5. In average, it needs less time to achieve the required task while keeping a lower distance to the global path (in average 5% closer). Same behavior is achieved when reducing the number of samples, see *setup* 2 columns of the tables. The approach also finds in average solutions with a lower cost, i.e., in average 5% better. Success rates of the both are on par. The naive implementation of the combination IT-MPC with CVAE trained using only Euclidean distance, achieves good results in terms of solution costs but other metrics are lower in average to the ones obtained by our algorithm. Time to achieve the goal is also larger than the one obtained by using the standard IT-MPC.

B. Results Experiment 2

In Table IV we report the results obtained for the obstacle avoidance case. Also for this case we report the relative improvement per metric of our approach with respect to each baseline. Our approach achieves better performance in every metric compared to the Dynamic Window Planner, including success rate. IT-MPC shows better results in terms of d_{max}, keeping the vehicle closer to the reference path: this meaning that this baseline performs less obstacle avoidance behaviors although having the same cost function of our approach. Moreover, the original IT-MPC also requires more time to complete the task, leading to a worst d_{acc} rate. The experiments demonstrate how the information provided by the network guides the sampling procedure towards an area

	Dynamic Window	IT-MPC
T_g	3.07%	18.39%
d_{acc}	6.96%	7.00%
C_s	16.04%	22.44%
d_{max}	9.67%	-8.42 %
Success(IIT-MPC/Other)	83.3/76.6%	83.3/57.3%

TABLE IV: Experiment 2 results. Comparing our approach to Dynamic Window Planning and IT-MPC.

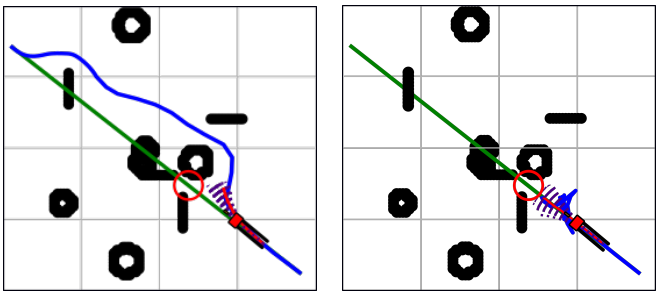


Fig. 6: Obstacle avoidance experiment using our approach **(left)** and IT-MPC **(right)**. The obstacles are represented by the black circles and lines. The green and blue lines are the obstacle-unaware reference and driven path respectively. The indigo dots represent the samples and the red line their expectation. Our approach is able to follow the global path and to successfully avoid obstacles.

of lower cost, in which no obstacles are found, see Fig. 6. For the cost metric (C_s) our approach shows a large improvement of 16% and 22.44% respect to each baseline.

VII. Conclusions

In this work we present an Informed approach to Information Theoretical Model Predictive Control (IIT-MPC). By using Conditional Variational Autoencoders (CVAE) to learn distributions that imitate samples from a training dataset containing optimized controls, we guide the sampling distribution of IIT-MPC towards less costly area of the state space. This allows our approach to achieve better minimization of the designed cost function. An extensive evaluation in the autonomous navigation domain suggests that replacing the previous IT-MPC sampling scheme with our learned models considerably improves performance in terms of path quality and also planning efficiency (i.e., completions of path tracking and avoiding obstacles tasks, reducing the number of necessary samples) compared to a set of baselines. Moreover, we show in the evaluation that learning the CVAE parameters by using task-dependent loss functions (i.e., reconstruction term) results in better planning performance compared to a standard Euclidean distance loss. We plan to extend the approach to dynamic environments by including into the CVAE conditions also human attributes, positions and velocities.

Acknowledgment

This work has been partly funded from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732737 (ILIAD).

REFERENCES

[1] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David A Forsyth. Learning diverse image colorization. In *Proc. of the IEEE Conf. on Comp. Vis. and Pat. Rec. (CVPR)*, 2017.

[2] Moritz Diehl, H Georg Bock, Johannes P Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[3] Paul Drews, Grady Williams, Brian Goldfain, Evangelos A. Theodorou, and James M. Rehg. Aggressive deep driving: Combining convolutional neural networks and model predictive control. In *Proc. of the 1st Annual Conference on Robot Learning*, 2017.

[4] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[5] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[6] Thomas M Howard, Colin J Green, and Alonzo Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In *Results of the Int. Conf. on Field and Service Robotics*, 2010.

[7] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[8] Boris Ivanovic, Edward Schmerling, Karen Leung, and Marco Pavone. Generative modeling of multimodal multi-human behavior. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[9] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

[10] Ian Lenz, Ross A Knepper, and Ashutosh Saxena. DeepMPC: Learning deep latent features for model predictive control. In *Proc. of the Robotics: Science and Systems (RSS)*, 2015.

[11] Sergey Levine and Vladlen Koltun. Guided policy search. In *Int. Conf. on Machine Learning (ICML)*, 2013.

[12] David Q Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[13] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[14] Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

[15] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Prof. of NIPS Autodiff Workshop*, 2017.

[16] Noé Pérez-Higueras, Fernando Caballero, and Luis Merino. Learning human-aware path planning with fully convolutional networks. In *Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[17] Morgan Quigley, Josh Faust, Tully Foote, and Jeremy Leibs. ROS: an open-source robot operating system. In *Proc. of ICRA Workshop on Open Source Software*, 2009.

[18] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[19] David H Shim, H Jin Kim, and Shankar Sastry. Decentralized nonlinear model predictive control of multiple flying robots. In *Proc. of the IEEE Int. Conf. on Decision and Control (CDC)*, 2003.

[20] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[21] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*. 2015.

[22] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11(Nov):3137–3181, 2010.

[23] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[24] Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M Rehg, and Evangelos A Theodorou. Robust sampling based model predictive control with sparse objective information. In *Proc. of the Robotics: Science and Systems (RSS)*, 2018.

[25] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic MPC for model-based reinforcement learning. In *Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[26] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In *Int. Conf. on Robotics and Automation (ICRA)*, 2016.