THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

# Learning How to Walk: Warm-starting Optimal Control Solver with Memory of Motion

**Citation for published version:**
Lembono, TS, Mastalli, C, Fernbach, P, Mansard, N & Calinon, S 2020, Learning How to Walk: Warm-starting Optimal Control Solver with Memory of Motion. in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, pp. 1357-1363, 2020 International Conference on Robotics and Automation, Virtual conference, France, 31/05/20.
https://doi.org/10.1109/ICRA40945.2020.9196727

**Digital Object Identifier (DOI):**
10.1109/ICRA40945.2020.9196727

**Link:**
Link to publication record in Edinburgh Research Explorer

**Document Version:**
Peer reviewed version

**Published In:**
2020 IEEE International Conference on Robotics and Automation (ICRA)

**General rights**
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

**Take down policy**
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 12. Sep. 2026

# Learning How to Walk: Warm-starting Optimal Control Solver with Memory of Motion

Teguh Santoso Lembono[1,2] Carlos Mastalli[3] Pierre Fernbach[3] Nicolas Mansard[3] Sylvain Calinon[1,2]

***Abstract*— In this paper, we propose a framework to build a memory of motion for warm-starting an optimal control solver for the locomotion task of a humanoid robot. We use HPP Loco3D, a versatile locomotion planner, to generate offline a set of dynamically consistent whole-body trajectory to be stored as the memory of motion. The learning problem is formulated as a regression problem to predict a single-step motion given the desired contact locations, which is used as a building block for producing multi-step motions. The predicted motion is then used as a warm-start for the fast optimal control solver Crocoddyl. We have shown that the approach manages to reduce the required number of iterations to reach the convergence from ∼9.5 to only ∼3.0 iterations for the single-step motion and from ∼6.2 to ∼4.5 iterations for the multi-step motion, while maintaining the solution's quality.**

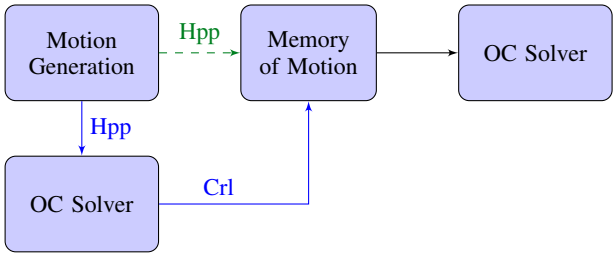



Fig. 1. Two approaches that are tested. In Approach 1 (dashed green), the motion generation using HPP Loco3D produces the dataset HPP, while in Approach 2 (solid blue), the dataset HPP is further optimized by the optimal control solver (Crocoddyl) to produce the dataset Crl. In each approach, the memory of motion is used for warm-starting the optimal control solver.

## I. Introduction

Legged locomotion is often achieved by first computing a sequence of contacts, generating a stable centroidal trajectory, and finally computing a whole-body motion through inverse dynamics [1][2]. However, this approach cannot properly regulate angular momentum because (a) centroidal trajectory optimization [3][4] does not consider the limbs momenta, and (b) whole-body control [5][6] is instantaneous control action that theoretically cannot properly regulate the angular momentum since it represents a nonholonomic constraint on the multibody dynamics [7].

Recently, optimal control is getting more attention in legged robots due to (a) its ability to properly control angular momentum [8], and (b) they can be deployed for real-time control [9][10]. In this vein, we have proposed an efficient multi-contact optimal control framework called Crocoddyl [11]. This framework relies on a novel multiple-shooting optimal control solver called Feasibility-prone Differential Dynamic Programming (FDDP). We have shown that Crocoddyl can generate highly-dynamic manuveurs for various legged robots such as iCub, Talos and ANYMal.

As a locally optimal solver, providing *warm-starts* (i.e., good initial guesses) to Crocoddyl can improve its real-time performance in Model Predictive Control (MPC) setting significantly. In this work, we use the concept of memory of motion to generate the warm-starts for Crocoddyl, to avoid poor local optima while speeding up the convergence. Using HPP Loco3D [12], a versatile locomotion framework for legged robots, we build offline a database of humanoid walking motions. We then train function approximators using the database to generate the warm-starts for Crocoddyl.

The memory of motion concept has been used in other works, e.g. for bicopter and quadcopter [13], serial manipulators [14][15][16], and humanoid manipulation task [17]. However, none of them involves locomotion tasks except in [18], where a trajectory library is constructed for the LittleDog robot. Their work does not involve warm-starting an optimization solver. Instead, the sequence of joint configurations retrieved from the library is used directly, with an integral controller to correct for errors. The library is created by using a joystick to move LittleDog across the terrain. Compared to [18], our approach of using HPP Loco3D to build the library, function approximations to learn the motion, and the optimal control solver Crocoddyl to optimize the motion is more versatile and applicable to higher DoFs robots with more complex dynamics, such as the humanoid robot Talos [19] considered in this work.

### A. Contribution

Our contribution in this paper is as follows. Firstly, we propose a framework for learning a memory of motion to warm-start a multi-contact optimal control solver (Fig. 1). We generate offline a database of dynamically consistent and collision-free motions using HPP Loco3D to build the memory of motion, which is then used to warm-start the optimal control solver Crocoddyl online. We study the effect of having different solvers for the offline and online computations, and describe how we tackle this problem. Finally, we propose a method that learns single-step motions based on function approximations, which are then used to build multi-step motions.

The outline of the paper is as follows. In Section II, the overall framework and the learning method are presented.

[1] Idiap Research Institute, Martigny, Switzerland [2] EPFL, Lausanne, Switzerland [3] Gepetto Team, LAAS-CNRS, Toulouse, France.

This work was supported by the European Union under the EU H2020 collaborative project MEMMO (Memory of Motion, `http://www.memmo-project.eu/`), Grant Agreement No. 780684.

Section III presents the simulation results, both for the single-step and multi-step locomotions. Finally, Section IV concludes the paper and discuss some ideas for the future work.

## II. Method

### A. Problem Definition

We consider the problem of generating whole-body locomotion of a biped robot from one location to another in a known environment and a flat terrain. The desired output consists of the robot joint configuration trajectory $\boldsymbol{q} \in \mathbb{R}^{D_q \times T}$ and the control input trajectory $\boldsymbol{u} \in \mathbb{R}^{D_u \times T}$, where $D_q, D_u, T$ are the joint configuration dimension, the control dimension, and the number of time steps. $\boldsymbol{q}$ includes both the trajectory of the root (i.e., the hip joint) and the joint positions, while $\boldsymbol{u}$ consists of the joint torques at each joint. The joint velocity and acceleration trajectories can also be included, but we only consider the joint configuration and the control trajectory here. To generate these trajectories, we use a fast optimal control solver Crocoddyl as the online solver. Our aim in this work is to provide good warm-starts to Crocoddyl using a *memory of motion* such that the number of solver iterations to reach convergence can be reduced.

### B. Overall Framework

In the proposed memory of motion approach, the choice of the offline and online solver is crucial. The offline solver is to build the database of motions, while the online solver is to control the robot in real time. The online solver has to be fast and efficient, hence it is usually only locally optimal, such as Crocoddyl in this work. The offline solver, on the other hand, is used to generate the database offline, so the speed is less important. There are two alternatives: either the offline solver is the same as the online solver, or a different one. The advantage of choosing the same solver is that the motion in the database would then have the same characteristics (e.g., optimality) as the online solver, which is desirable. However, since the solver would only be locally optimal, it is difficult to use it for generating a good database without providing warm-starts to the solver.

The second alternative uses a different solver for building the database, e.g. a more global planner that can solve various tasks without requiring warm-starts. This is the approach that we take, as can be seen in Fig. 1. HPP Loco3D [20], a locomotion framework for multi-contact locomotion, is used as the offline solver. The framework is versatile and has been applied to both biped and quadruped robots. HPP Loco3D can compute the sequence of stable contacts to achieve the locomotion task, which cannot be done yet in Crocoddyl. We use HPP Loco3D to generate motion samples corresponding to various tasks and store them as the memory of motion.

However, the issue with this approach is that the motion in the database might not be considered *optimal* by Crocoddyl, because HPP Loco3D does not properly regulate angular momentum and uses heuristics for defining the swing-foot trajectories. Indeed, there are qualitatively different motion characteristics between HPP Loco3D and Crocoddyl. The former generates conservative and stable motions, while the latter uses the full dynamics that optimally reduces the joint torques and contact forces. Therefore, warm-starting Crocoddyl using HPP Loco3D output will require additional iterations during the online computation to refine the motion according to its optimality criteria, which is undesirable.

We overcome this problem by leveraging both HPP Loco3D and Crocoddyl for building the memory. That is, we take the motion samples generated by HPP Loco3D and optimize them using Crocoddyl. The resulting output is then saved as the new memory of motion. By doing this, we combine the benefits of both frameworks: we can have a sequence of stable footholds (HPP Loco3D) with optimal whole-body motions (Crocoddyl). We will demonstrate that this will yield improvement in the quality of the warm-starts. In this work, we refer to the database containing the HPP Loco3D motion samples as Database $Hpp$, and the one containing the optimized Crocoddyl motion samples as Database $Crl$.

In what follows, we describe in details HPP Loco3D and Crocoddyl frameworks.

*1) HPP Loco3D:* The planning framework proposed in HPP Loco-3D generates dynamically consistent and collision free multi-contact whole-body motion for legged robot. It takes as input the model of the environment and an initial and goal poses of the robot's root. Optionally, some additional constraints may be specified, such as velocity bounds or a set of initial and final contact positions. This framework decouples the motion planning problem into several sub-problems to be solved sequentially. First, the guide planning produces a rough initial guess of the root path of the robot [12][21]. Then, a contact planner produces a feasible sequence of contacts following the root path [12][22]. After that, an optimal centroidal trajectory satisfying the centroidal dynamic constraints for the given contact points is computed [3]. Finally, a second order inverse dynamic solver[1] generates a whole-body motion, following the references of the contact locations and the centroidal trajectory.

*2) Crocoddyl:* It is a framework for multi-contact optimal control. Given a predefined sequence of contacts, it computes efficiently the state trajectory and control policy by using sparse analytical derivatives and by exploiting the problem structure inherited from the dynamic programming principle. Its optimal control algorithm, called feasibility-prone differential dynamic programming (FDDP), has a great globalization strategy and similar numerical behavior to multiple-shooting methods [11]. During the numerical optimization, FDDP computes the search direction and length through backward and forward passes, respectively. Unlike the classical DDP, the backward pass accepts infeasible state-control trajectories which is a critical aspect to warm-starting the solver from the memory of motion; the forward pass simulates properly infeasible search direction, obtained in the backward pass, which improves the algorithm exploration.

[1] https://github.com/stack-of-tasks/tsid

### C. Learning Strategies

One important question that needs to be considered is, what do we expect the memory to learn? Should it learn directly how to move from one location to another? while this is indeed possible, such method does not generalize very well, even in a given environment. Instead, we decompose the data into single-step motions, and we retrieve a multi-step trajectory as a combination of single-step motions.

*1) Learning Single-step Motion:* Following our previous work [23], we formulate the problem of learning the single-step motion as a regression problem to approximate the mapping $f : \boldsymbol{x} \rightarrow \boldsymbol{y}$, where $\boldsymbol{x}$ is the task and $\boldsymbol{y}$ is the corresponding trajectory output. We separate the database into left-leg and right-leg movements, as this yields better results than combining both and let the memory learns how to choose the leg. Each motion starts with the root position at the origin.[2] The task is defined as the initial and goal foot poses, $\boldsymbol{x} = [\boldsymbol{c}_{l0}, \boldsymbol{c}_{r0}, \boldsymbol{c}_{*T}] \in \mathbb{R}^9$, where $\boldsymbol{c} \in \mathbb{R}^3$ is the foot pose (2D position and orientation), the subscripts $l, r$ correspond to the left and right foot, while $*$ is either $l$ for the left-leg or $r$ for the right-leg database. $T$ is the final time step. $\boldsymbol{y}$ can be either the joint configuration trajectory $\boldsymbol{q} \in \mathbb{R}^{D_q \times T}$ or the control trajectory $\boldsymbol{u} \in \mathbb{R}^{D_u \times T}$. We then apply dimensionality reduction and function approximation techniques to solve the regression problem.

Since the dimension of $\boldsymbol{y} \in \mathbb{R}^{D \times T}$ is high, we need to find a smaller representation of $\boldsymbol{y}$. First, we represent the trajectory of each dimension of $\boldsymbol{y}$ as the weight of radial basis functions (RBFs), as commonly done in probabilistic movement primitives [24][25]. Let $\boldsymbol{y}_i \in \mathbb{R}^T$ be the trajectory of the $i_{\text{th}}$ dimension of $\boldsymbol{y}$. We can define the basis matrix $\boldsymbol{\Phi}$ as $[\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{K-1}] \in \mathbb{R}^{T \times K}$, where $\boldsymbol{\phi}_k \in \mathbb{R}^T$ is the $k_{th}$ basis function, defined as a Gaussian function centered at the $k_{\text{th}}$ mean. The means are distributed equally along the time axis $T$, whereas the variance is chosen to be equal for all the basis functions and to have sufficient overlap. $\boldsymbol{y}_i$ can then be represented as the weights of these basis functions,

$$\boldsymbol{y}_i = \boldsymbol{\Phi} \boldsymbol{w}_i, \tag{1}$$

where $\boldsymbol{w}_i \in \mathbb{R}^K$ can be computed by standard linear least squares algorithm. This reduces the number of variables for each dimension from $T$, which is usually large, to $K$ which can be much smaller. We can then stack the weights corresponding to all dimensions of $\boldsymbol{y}$ and obtain $\boldsymbol{w} = [\boldsymbol{w}_0, \ldots, \boldsymbol{w}_i, \ldots, \boldsymbol{w}_{D-1}] \in \mathbb{R}^{DK}$. Each $\boldsymbol{y}$ has the corresponding weight vector $\boldsymbol{w}$.

Now let's consider all the $N$ samples in the database. We can apply principal component analysis (PCA) to further reduce the dimension of $\boldsymbol{w}$ over this database by keeping only $M$ principal components to obtain $\hat{\boldsymbol{y}} \in \mathbb{R}^M$. We have thus reduced the dimension of $\boldsymbol{y}$ from $DT$ to $M$. Inverse transformation from $\hat{\boldsymbol{y}}$ to $\boldsymbol{y}$ is a matrix multiplication that can be performed quickly. The regression problem then becomes $f : \boldsymbol{x} \rightarrow \hat{\boldsymbol{y}}$. To solve the regression problem, we consider two function approximation techniques: Gaussian process regression (GPR) and Gaussian mixture regression (GMR).

[2]Without any loss of generality, as we can always transform the coordinate system

#### Gaussian Process Regression (GPR)

GPR [26] is a non-parametric method which improves its accuracy as the number of datapoints increases. Given the database $\{\boldsymbol{X}, \boldsymbol{Y}\}$, GPR assigns a Gaussian prior to the joint probability of $\boldsymbol{Y}$, i.e., $p(\boldsymbol{Y}|\boldsymbol{X}) = \mathcal{N}\big(\boldsymbol{\mu}(\boldsymbol{X}), \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})\big)$. $\boldsymbol{\mu}(\boldsymbol{X})$ is the mean function and $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ is the covariance matrix constructed with elements $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the kernel function that measures the similarity between the inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. In this paper we use RBF as the kernel function, and the mean function $\boldsymbol{\mu}(\boldsymbol{X})$ is set to zero, as usually done in GPR.

To predict the output $\boldsymbol{y}^*$ given a new input $\boldsymbol{x}^*$, GPR constructs the joint probability distribution of the training data and the prediction, and then conditions on the training data to obtain the predictive distribution of the output, $p(\boldsymbol{y}^*|\ \boldsymbol{x}^*) \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$, where $\boldsymbol{m}$ is the posterior mean computed as

$$\boldsymbol{m} = \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{X})\, \boldsymbol{K}^{-1}(\boldsymbol{X}, \boldsymbol{X})\, \boldsymbol{Y}(\boldsymbol{X}), \tag{2}$$

and $\boldsymbol{\Sigma}$ is the posterior covariance which provides a measure of uncertainty on the output. In this work we simply use the posterior mean $\boldsymbol{m}$ as the output, i.e., $\boldsymbol{y}^* = \boldsymbol{m}$.

#### Gaussian Mixture Regression (GMR)

Given the training database $\{\boldsymbol{X}, \boldsymbol{Y}\}$ GMR constructs the joint probability of $(\boldsymbol{x}, \boldsymbol{y})$ as a mixture of Gaussians,

$$p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{3}$$

where $\pi_k$, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ are the $k$-th component's mixing coefficient, mean, and covariance, respectively. Let $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, denoting the GMR parameters to be determined from the data.

We can decompose $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ according to $\boldsymbol{x}$ and $\boldsymbol{y}$ as

$$\boldsymbol{\mu}_k = \begin{pmatrix} \boldsymbol{\mu}_{k,x} \\ \boldsymbol{\mu}_{k,y} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_k = \begin{pmatrix} \boldsymbol{\Sigma}_{k,xx} & \boldsymbol{\Sigma}_{k,xy} \\ \boldsymbol{\Sigma}_{k,yx} & \boldsymbol{\Sigma}_{k,yy} \end{pmatrix}. \tag{4}$$

Given a query $\boldsymbol{x}^*$, the predictive distribution of $\boldsymbol{y}$ can then computed by conditioning on $\boldsymbol{x}^*$,

$$p(\boldsymbol{y}|\ \boldsymbol{x}^*, \boldsymbol{\theta}) = \sum_{k=1}^{K} p(k|\ \boldsymbol{x}^*, \boldsymbol{\theta})\, p(\boldsymbol{y}|\ k, \boldsymbol{x}^*, \boldsymbol{\theta}), \tag{5}$$

where $p(k|\ \boldsymbol{x}^*, \boldsymbol{\theta})$ is the probability of $\boldsymbol{x}^*$ belonging to the $k$-th component,

$$p(k|\ \boldsymbol{x}^*, \boldsymbol{\theta}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}^*|\ \boldsymbol{\mu}_{k,x}, \boldsymbol{\Sigma}_{k,xx})}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\boldsymbol{x}^*|\ \boldsymbol{\mu}_{i,x}, \boldsymbol{\Sigma}_{i,xx})}, \tag{6}$$

and $p(\boldsymbol{y}|\ k, \boldsymbol{x}^*, \boldsymbol{\theta})$ is the predictive distribution of $\boldsymbol{y}$ according to the $k$-th component,

$$p(\boldsymbol{y}|\ k, \boldsymbol{x}^*, \boldsymbol{\theta}) = \mathcal{N}\Big(\boldsymbol{\mu}_{k,y} + \boldsymbol{\Sigma}_{k,yx}\boldsymbol{\Sigma}_{k,xx}^{-1}(\boldsymbol{x}^* - \boldsymbol{\mu}_{k,x}), \\ \boldsymbol{\Sigma}_{k,yy} - \boldsymbol{\Sigma}_{k,yx}\boldsymbol{\Sigma}_{k,xx}^{-1}\boldsymbol{\Sigma}_{k,xy}\Big), \tag{7}$$

which is a Gaussian distribution with the mean being linear in $\boldsymbol{x}^*$. The resulting predictive distribution (5) is then a mixture of Gaussians. The point prediction $\boldsymbol{y}^*$ can be obtained from this distribution by applying moment matching to the distribution in (5) to approximate it by a single Gaussian, and use the mean of the Gaussian as the desired output $\boldsymbol{y}^*$, see [25], [27] for details.

*2) Constructing Multi-step Motion:* To use the single-step motion for generating multi-step motions, we have to transform the coordinate system at each step. Let's first assume that we have the planned sequence of contacts (i.e., foot poses), $\{\boldsymbol{C}_i\}_{i=0}^{P-1}$, where $\boldsymbol{C}_i = (\boldsymbol{c}_{li}, \boldsymbol{c}_{ri})$ is the contacts at $i_{\text{th}}$ footstep and $P$ is the total number of footsteps. Assume, without loss of generality, that the motion starts at zero root position horizontally. The first step can be obtained by querying the single-step memory directly to move from $\boldsymbol{C}_0$ to $\boldsymbol{C}_1$, obtaining $\boldsymbol{y}^0$. To move from $\boldsymbol{C}_1$ to $\boldsymbol{C}_2$, we first need to shift the coordinate system such that the current root (based on the last configuration at $\boldsymbol{y}^0$) is in zero horizontal position. The motion from $\boldsymbol{C}_1$ to $\boldsymbol{C}_2$ can then be queried from the memory to obtain $\boldsymbol{y}^1$, and this is iterated until $\boldsymbol{C}_{P-1}$ to obtain $\{\boldsymbol{y}^i\}_{i=0}^{P-1}$. Finally, each trajectory $\boldsymbol{y}^i$ is transformed back to the original coordinate system and concatenated as a single trajectory $\boldsymbol{y}$.

The contact sequence $\{\boldsymbol{C}_i\}_{i=0}^{P-1}$ can be obtained from another planner, such as RB-PRM [12]. The alternative is to also learn it from the database. In this work we assume that the contact sequence is already given, and the contact sequence learning will be considered in future work.

## III. Experiments

To evaluate the proposed approach, we conduct several experiments with the humanoid robot Talos in simulation. The robot joint configuration consists of 3 DoFs root position, 3 DoFs root orientation (described in quaternion), and 32 joint angles ($D_q = 39$), while the control input consists of 32 joint torques ($D_u = 32$). First, HPP Loco3D is used to generate $N = 1200$ samples of one-step motion (right-leg and left-leg movement in equal proportions), starting from the initial contact $(\boldsymbol{c}_{l0}, \boldsymbol{c}_{r0})$ to the goal contact $(\boldsymbol{c}_{lT}, \boldsymbol{c}_{rT})$. One sample thus consists of $\{(\boldsymbol{c}_{l0}, \boldsymbol{c}_{r0}), (\boldsymbol{c}_{lT}, \boldsymbol{c}_{rT}), \boldsymbol{q}, \boldsymbol{u}\}$. These are stored as Database Hpp.[3] Next, each sample is optimized using Crocoddyl, and the resulting samples are stored as Database Crl. The cost function in Crocoddyl consists of state and control regularization (around a standing pose and zero, respectively), and contact placement. Since we need high-quality database for the memory of motion, we use a small convergence threshold of $10^{-5}$ and the maximum number of iterations is set to be 50. The time interval in HPP Loco3D is 1 ms and 40 ms, respectively, so the HPP Loco3D data is subsampled to Crocoddyl's interval for the optimization. Both databases will be used and the performance will be compared. The databases are split into the training and the test dataset, with $N_{\text{train}} = 1000$ and $N_{\text{test}} = 200$. We applied RBF and PCA to reduce the dimensions of $\boldsymbol{q}$ and $\boldsymbol{u}$ with $K = 60$ and $M = 60$, as determined empirically.

[3]The database is available at `https://github.com/MeMory-of-MOtion/docker-loco3d`.

TABLE I
Comparing The Accuracy of GPR and GMR

| Method | Database Hpp | | Database Crl | |
|---|---|---|---|---|
| | Traj. Err. | Contact Err. | Traj. Err. | Contact Err. |
| GPR | 9.53 ± 4.63 | 0.07 ± 0.03 | 13.04 ± 6.15 | 0.04 ± 0.02 |
| GMR | 12.39 ± 5.00 | 0.13 ± 0.05 | 18.51 ± 6.80 | 0.09 ± 0.04 |
| $k$-NN | 18.78 ± 4.96 | 0.49 ± 0.15 | 29.93 ± 9.02 | 0.51 ± 0.10 |

We proceed as follows. First, we evaluate the accuracy performance of GPR and GMR in approximating the mapping $f$. The warm-starts generated by GPR and GPR are then compared to the cold-start in terms of the Crocoddyl performance, i.e., the number of iterations until convergence and the resulting trajectory cost. Finally, we also compare the result of warm-starts using only $\boldsymbol{q}$ to using both $\boldsymbol{q}$ and $\boldsymbol{u}$. For all comparisons, we use both the databases Hpp and Crl and compare their performance.

### A. Comparing GPR and GMR Accuracy

We train GPR and GMR on the reduced-dimension dataset $\{\boldsymbol{x}, \hat{\boldsymbol{y}}\}$ for both databases Hpp and Crl with $N_{\text{train}}$ samples. The task $\boldsymbol{x}$ is defined as the initial and goal foot poses, $\boldsymbol{x} = [\boldsymbol{c}_{l0}, \boldsymbol{c}_{r0}, \boldsymbol{c}_{*T}]$ where $* = l$ for the left-leg movement and $* = r$ for the right-leg movement. The output $\boldsymbol{y}$ is defined here as the joint configuration trajectory $\boldsymbol{q}$, and $\hat{\boldsymbol{y}}$ is its smaller dimension representation. For each $\boldsymbol{x}$ in the $N_{\text{test}}$ samples, we use GPR and GMR to predict the corresponding trajectory $\boldsymbol{y}$, and the accuracy is evaluted against the true trajectory in the database. The trajectory error $(rad)$ is calculated as the difference between the true and predicted trajectory, i.e. $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} ||\boldsymbol{y}^i - \tilde{\boldsymbol{y}}^i||_2$, whereas the contact error $(m)$ is defined as the difference between the foot poses of the true and predicted trajectory, $\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} ||\boldsymbol{C}^i - \tilde{\boldsymbol{C}}^i||_2$.

The results can be seen in Table I (averaged over the left and right leg movements). We compare against $k$-NN with $k = 1$ as a baseline, to demonstrate that the proposed algorithm indeed generalizes well and the good performance is not due to having a very dense database. GPR overall has the lowest errors in both criteria and both databases. GMR has higher errors than GPR but still outperforms the baseline $k$-NN by a large margin. Some example of the motions can be found in Fig. 2. We can see that motion predicted by GPR fit the desired contact locations very well.

Furthermore, for the subsequent results we will use the subscripts Hpp and Crl for the function approximators trained on the databases Hpp and Crl, respectively.

### B. Single-step Motion: Warm-start vs Cold-start

In this section we compare the results of warm-starting Crocoddyl using the function approximators against the cold-start, i.e., using zero initial guess. For each $\boldsymbol{x}$ in the $N_{\text{test}}$ samples, GPR and GMR are queried to obtain the initial guess $\boldsymbol{y}$, defined here as the joint configuration trajectory

Fig. 2. Examples of predicting single-step motions. Warm-start produced by *Top:* GPR, *Middle:* GMR, and *Bottom:* $k$-NN. *Left:* left foot movement. *Right:* right foot movement. The green, blue and red box correspond to the initial left, the initial right, and the goal contact pose, respectively.

$\boldsymbol{q}$. We do not predict the trajectory of the control input here, but instead calculate it from $\boldsymbol{q}$ assuming quasi-static movement. This computed control input trajectory is denoted as $\boldsymbol{u}_0$. The initial guess is used to warm-start Crocoddyl, and the result is compared against the cold-start. We also compare the effect of using Database Hpp and Crl. We use a large convergence threshold ($10^{-2}$) for Crocoddyl here, based on the assumption that at online computation a very refined optimal motion is not really necessary. The number of iterations is also limited to 20. The query time is $\sim$5ms for GPR and $\sim$10ms for GMR in python implementation.

The results can be seen in Table II. The solver is considered successful if it finds a feasible trajectory within the maximum number of iterations. We see from the table that the warm-starts results consistently outperform the cold-starts; it justifies our assumption that warm-starting Crocoddyl can speed-up the computation time for online MPC.

Although in Table I we see that the accuracy of GPR outperforms GMR quite substantially, the warm-starting performance in Table II turns out to be very similar, with GPR having very slightly better performance. This is due to the fact that the initial guesses produced by GPR and GMR still go through an optimization process in Crocoddyl, and hence the small difference between the predictions does not affect the results much. This means that while in standard regression tasks the accuracy is highly important, it may be less important in tasks such as producing warm-starts, as long as the predicted initial guesses are sufficiently close to the optimal solutions.

Finally, we compared the results between the function approximators trained on Database Hpp ($\text{GPR}_{\text{Hpp}}$, $\text{GMR}_{\text{Hpp}}$) and Crl ($\text{GPR}_{\text{Crl}}$, $\text{GMR}_{\text{Crl}}$). Those trained on the database Crl have lower number of iterations, which justify our step of optimizing the HPP Loco3D dataset by Crocoddyl. The dataset in Crl contain motion samples that are optimal according to Crocoddyl, and therefore they perform better in warm-starting Crocoddyl.

TABLE II

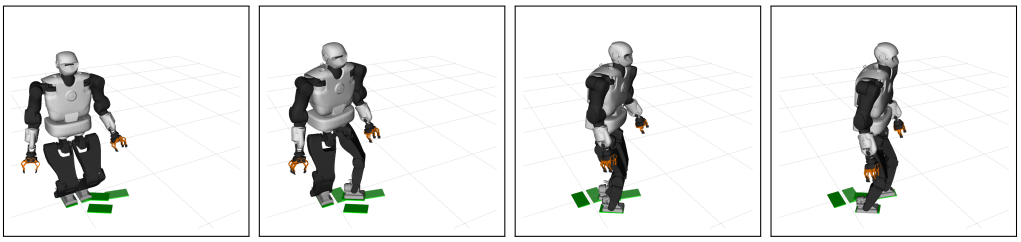
COMPARING WARM-START VS COLD-START: SINGLE-STEP

| Method | Success Rate | Cost | Num. Iteration |
|---|---|---|---|
| Cold-start | 98.50 | 55.56 $\pm$8.32 | 9.52 $\pm$ 3.94 |
| $\text{GPR}_{\text{Hpp}}$ | 99.00 | 55.27 $\pm$8.01 | 5.01 $\pm$ 0.74 |
| $\text{GMR}_{\text{Hpp}}$ | 100.00 | 55.31 $\pm$7.99 | 4.85 $\pm$ 0.66 |
| $\text{GPR}_{\text{Crl}}$ | 100.00 | 55.32 $\pm$7.99 | 3.04 $\pm$ 0.41 |
| $\text{GMR}_{\text{Crl}}$ | 100.00 | 55.32 $\pm$7.99 | 3.06 $\pm$ 0.45 |

### C. Single-step Motion: Evaluating Warm-starts Components

In Section III-B, we only predict the joint configuration trajectory $\boldsymbol{q}$ for warm-starting Crocoddyl while the control trajectory $\boldsymbol{u}$ is computed based on the predicted $\boldsymbol{q}$. In this section we evaluate the different performances if we also predict $\boldsymbol{u}$, or use zero control trajectory as warm-start. We train two different GPRs for the prediction, one for predicting $\boldsymbol{q}$ and the other one for predicting $\boldsymbol{u}$.

Table III shows the comparison results. $(\boldsymbol{q}, \boldsymbol{u}_0)$ denotes the warm-starts using the predicted $\boldsymbol{q}$ while the control trajectory is computed as $\boldsymbol{u}_0$, the same as in the previous subsection. $(\boldsymbol{q})$ denotes the warm-starts using only the joint configuration trajectory while the control trajectory is set to be zero. While the cost remains the same, the number of iterations increases when the control trajectory is omitted from the warm-starts. $(\boldsymbol{q}, \boldsymbol{u})$ denotes the warm-starts using both predicted joint configuration and control trajectory, which has similar results



(a)

Fig. 3. Examples of predicting a multi-step motion by GPR.

TABLE III

COMPARING WARM-START COMPONENTS: SINGLE-STEP

| Method | Success Rate | Cost | Num. Iteration |
|---|---|---|---|
| $(\boldsymbol{q})$ | 96.50 | 55.06 $\pm$8.19 | 5.30 $\pm$ 2.19 |
| $(\boldsymbol{q}, \boldsymbol{u}_0)$ | 100.00 | 55.32 $\pm$7.99 | 3.04 $\pm$ 0.41 |
| $(\boldsymbol{q}, \boldsymbol{u})$ | 100.00 | 55.32 $\pm$7.99 | 2.93 $\pm$ 0.45 |
| $(\boldsymbol{u})$ | 97.50 | 55.36 $\pm$7.95 | 6.83 $\pm$ 2.46 |
| Cold-start | 98.50 | 55.56 $\pm$8.32 | 9.52 $\pm$ 3.94 |

TABLE IV

COMPARING WARM-START VS COLD-START: MULTI-STEP

| Method | Success Rate | Cost | Num. Iteration |
|---|---|---|---|
| Cold-start | 100.00 | 86.36 $\pm$23.16 | 6.17 $\pm$ 1.52 |
| $\text{GPR}_{\text{Hpp}}$ | 100.00 | 86.39 $\pm$23.07 | 6.40 $\pm$ 0.73 |
| $\text{GMR}_{\text{Hpp}}$ | 100.00 | 86.38 $\pm$23.12 | 7.29 $\pm$ 1.32 |
| $\text{GPR}_{\text{Crl}}$ | 100.00 | 86.47 $\pm$23.16 | 4.54 $\pm$ 0.55 |
| $\text{GMR}_{\text{Crl}}$ | 100.00 | 86.51 $\pm$23.22 | 4.71 $\pm$ 0.70 |

to $(\boldsymbol{q}, \boldsymbol{u}_0)$ except for the slightly lower number of iterations. Finally, $(\boldsymbol{u})$ denotes the warm-starts using only the control trajectory while the joint configuration trajectory is set to zero, and cold-start means warm-starting using zero joint configuration and control trajectory.

Comparing $(\boldsymbol{q}, \boldsymbol{u})$ and $(\boldsymbol{q}, \boldsymbol{u}_0)$, we conclude that predicting the control trajectory from the memory does not give significant benefit as compared to computing it based on $\boldsymbol{q}$. However, control trajectory is still important to be included in the warm-starts, as omitting them in $(\boldsymbol{q})$ increases the number of iterations. Finally, comparing $(\boldsymbol{q})$ and $(\boldsymbol{u})$, we conclude that warm-starting using only the joint configuration trajectory performs better than using only control trajectory, which has higher cost and number of iterations.

### *D. Multi-step Motion: Warm-start vs Cold-start*

Finally, we use the single-step motions as a building block for multi-step locomotion, as described in Section II-C. We use HPP Loco3D to generate 50 contact sequences, each consists of $P = 3$ footsteps. $\text{GPR}_{\text{Hpp}}$, $\text{GMR}_{\text{Hpp}}$, $\text{GPR}_{\text{Crl}}$ and $\text{GMR}_{\text{Crl}}$ generate the initial guesses of the joint configuration trajectory $\boldsymbol{q}$ while the control trajectory is computed based on $\boldsymbol{q}$, as in Section III-B. The warm-starts are then given to Crocoddyl, and the results are presented in Table IV. Interestingly the Hpp database does not perform better than the cold-start (indeed, it is the opposite). This could be due to the fact that we build the motion using a concatenation of three single-step motions, each step starts and ends with zero velocity. The nonoptimality of the Hpp database, added with the nonoptimality of the multi-step motion strategy, seem to render the warm-start to be not useful at all. On the contrary, warm-starting using the Crl database still speeds-up the convergence, although not as much as in the single-step case (Section III-B). An example of multi-step locomotion warm-start produced by GPR is given in Fig. 3.

## IV. CONCLUSION AND FUTURE WORK

We have presented a framework for learning a memory of motion to warm-start an optimal control solver. The proposed approach manages to reduce the average number of solver iterations from $\sim$9.5 to only $\sim$3.0 iterations for the single-step motion and from $\sim$6.2 to $\sim$4.5 iterations for the multi-step motion, while maintaining the solution's quality.

This paper shows a preliminary result of warm-starting an optimal control solver. In the current formulation, Crocoddyl does not include constraints such as torque limit or obstacle avoidance, and we are working towards this direction. When the optimal control formulation becomes more realistic and complex, the solver would need even higher computational time, and the proposed warm-starting approach would potentially be even more useful. The final goal is to use the whole framework to control the real robot using MPC.

In this work, we decompose the multistep locomotion into single-step motions, with the initial and goal contact locations as the task that needs to be provided by some other methods. Another potential strategy is to define the task to be the final root pose instead of the contact location. The memory will then determine the contact location to reach the desired root pose. This may allow more flexibility in generating the multi-step motions, as only the root trajectory needs to be provided instead of the contact sequence. Another issue with the single-step decomposition strategy is that the predicted motion has zero velocity at the beginning and the end of each step. We can include the initial and goal root velocity in the task definition, but the task space and hence the required number of samples will increase. While random sampling is used in this work to generate the tasks, active learning [28] can reduce the required number of samples.

Finally, we plan to extend the approach to include multi-contact locomotion with both hands and legs as potential contacts, and locomotion with varying contacts' heights (e.g. climbing stairs or uneven terrains).

## REFERENCES


[1] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2015.

[2] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, 2018.

[3] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018.

[4] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernandez-Lopez, and C. Semini, "Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization," *IEEE Robotics and Automation Letters (RA-L)*, 2017.

[5] A. Del Prete and N. Mansard, "Robustness to joint-torque-tracking errors in task-space inverse dynamics," *IEEE Transactions on Robotics*, vol. 32, 2016.

[6] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive whole-body control for quadruped robots: Experimental validation over challenging terrain," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, 2019.

[7] P.-B. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," in *Fast motions in biomechanics and robotics*. Springer, 2006.

[8] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2018.

[9] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 3346–3351.

[10] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, 2017.

[11] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2020.

[12] S. Tonneau, A. Del Prete, J. Pettr, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, 2018.

[13] N. Mansard, A. Del Prete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018.

[14] C. Liu and C. G. Atkeson, "Standing balance control using a trajectory library," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[15] N. Jetchev and M. Toussaint, "Trajectory prediction: learning to map situations to robot trajectories," in *Proc. Intl Conf. on Machine Learning (ICML)*, 2009.

[16] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2012.

[17] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging precomputation with problem encoding for warm-starting trajectory optimization in complex environments," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[18] M. Stolle, H. Tappeiner, J. Chestnutt, and C. G. Atkeson, "Transfer of policies based on trajectory libraries," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[19] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro, "Talos: A new humanoid research platform targeted for industrial applications," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2017.

[20] J. Carpentier, A. Del Prete, S. Tonneau, T. Flayols, F. Forget, A. Mifsud, K. Giraud, D. Atchuthan, P. Fernbach, R. Budhiraja, M. Geisert, J. Sola, O. Stasse, and N. Mansard, "Multi-contact locomotion of legged robots in complex environments–the loco3d project," in *Workshop on Challenges in Dynamic Legged Locomotion*, 2017.

[21] P. Fernbach, S. Tonneau, A. D. Prete, and M. Taïx, "A kinodynamic steering-method for legged multi-contact locomotion," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[22] P. Fernbach, S. Tonneau, and M. Taïx, "Croc: Convex resolution of centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[23] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 2594–2601, April 2020.

[24] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[25] S. Calinon, "Mixture models for the analysis, edition, and synthesis of continuous time series," in *Mixture Models and Applications*, N. Bouguila and W. Fan, Eds. Springer, 2019, pp. 39–57.

[26] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[27] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach," in *Advances in Neural Information Processing Systems (NIPS)*, 1994.

[28] B. Settles, "Active learning literature survey," University of Wisconsin-Madison, Department of Computer Sciences, Tech. Rep. TR 1648, 2009.