Learning a Centroidal Motion Planner for Legged Locomotion

Julian Viereck[1,2], Ludovic Righetti [1,2]

Abstract— Whole-body optimizers have been successful at automatically computing complex dynamic locomotion behaviors. However they are often limited to offline planning as they are computationally too expensive to replan with a high frequency. Simpler models are then typically used for online replanning. In this paper we present a method to generate whole body movements in real-time for locomotion tasks. Our approach consists in learning a centroidal neural network that predicts the desired centroidal motion given the current state of the robot and a desired contact plan. The network is trained using an existing whole body motion optimizer. Our approach enables to learn with few training samples dynamic motions that can be used in a complete whole-body control framework at high frequency, which is usually not attainable with typical full-body optimizers. We demonstrate our method to generate a rich set of walking and jumping motions on a real quadruped robot.

I. INTRODUCTION

Recently quadrupeds like Spot or ANYmal have shown a new level of autonomy by traversing rough terrain. For control, these kind of robots often use a simplified dynamics model of the robot using only the dynamics of the center of mass [1]. In order to carry out more complex movements with a legged robot, it becomes important to take the full body dynamics into account. Recently, there has been progress in providing faster full body optimizers [2], [3], [4]. However, these methods can still require seconds to optimize a full-body movement over a sequence of contacts. This is too long to run the computation online while the robot is executing the motion. Reducing the optimization horizon can help decrease optimization time but to date, full nonlinear optimizer for legged robots are not run in fast control loops at the order of a few milliseconds. Therefore, in many cases, either the full-body motion is computed up front and only replayed on the robot [4], [5], [6] or a simplified model of the dynamics is used for online optimization [1], [7], [8]. This makes it difficult to react to (fast) changes in the environment that require full-body adaptation.

In this work, we present a new machine learning based approach for computing whole body movements for legged robots performing dynamic locomotion tasks. The method is fast and allows running the computation of full-body movements online at 100 Hz. Our approach uses the output of an existing kino-dynamic optimizer to train a planner capable of generating various motion patterns and generalize motions

Fig. 1: The robot Solo12 used for the experiments [9].

outside of the training data, e.g. to adapt the motion to new footstep sequences. Furthermore, the resulting computation time is an order of magnitude faster than typical trajectory optimization methods.

The use of machine learning to cache the results of locomotion plans has been investigated before. In [10] multiple tasks are optimized and a global neural network policy is trained from all the local optimal trajectories. In this work, however, we are not learning a policy but preserve a structured output from the motion optimizer. [11] uses a neural network to predict internal optimization variables to warm start a classical optimization method. While the method shows a speedup and improved initialization, it is not fast enough for real time use. Offline computations of full-body motions are also used to learn feasible contact transitions for efficient dynamically-consistent contact planning [12].

The work closest to ours is the one presented in [13]. The approach also learns a network to predict the centroidal motions of a robot and intertwine the prediction with inverse kinematics. In contrast to our work, however, a full body motion generator is not used to generate the training data. Further, the input to the neural network is not relying on motion patterns as we do in this contribution. Finally, results, while impressive, are limited to simulations while we demonstrate our approach directly on a real quadruped.

Methods based on reinforcement learning have also gained popularity to compute full-body movements. They are model-free methods that typically sample from a simulator to learn policy. While these methods have shown applications on real robots lately [14], [15], [16], their optimization is black-box and tend to require a lot of samples compared to

the approach presented in this work.

The contributions of the paper are as follow: 1) we propose a method for learning and predicting the output of a whole body motion optimizer for a legged robot for dynamic tasks with contacts, 2) we introduce a way to learn motion patterns which allow to generalize the learned motions outside of the trained area, 3) we demonstrate the capabilities of our approach to generate motion plans at 100 Hz, one order of magnitude faster than the original motion planner, and 4) we demonstrate the applicability of our approach on walking and jumping experiments with a quadruped robot.

II. BACKGROUND

In this section, we describe the elements necessary for our approach, including the whole-body controller and inverse kinematics used to generate full movements and the kino-dynamic optimizer that will be replaced by our approach.

A. Kino-dynamics motion optimizer

We are interested in the kino-dynamic motion optimizer proposed in [17] that decomposes the problem into a centroidal dynamic optimization problem and a kinematic problem. Given a motion description (i.e. sequence of contacts and cost function), the solver alternatively optimize the dynamic and kinematic problems until they reach consensus. While this method is very efficient [4] it still can take seconds to find a complete plan.

1) Centroidal dynamic optimization: The centroidal optimization problem, which our approach will aim to learn, is formulated as

$$\min_{\mathbf{h}_t, \mathbf{F}_{e,t}, \mathbf{\jmath}_{e,t}, \tau_{e,t}} \sum_{t=1}^{N} \phi_t^{\text{dyn}} \left(\mathbf{h}_t, \mathbf{\jmath}_{e,t}, \mathbf{F}_{e,t}, \tau_{e,t} \right) \quad (1a)$$

subject to:

$$\mathbf{h}_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{k}_t \\ \mathbf{l}_t \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{t-1} + \frac{1}{m} \mathbf{l}_t \Delta_t \\ \mathbf{k}_{t-1} + \sum_{e \in e_{\text{cnt}}} \kappa_{e,t} \Delta_t \\ \mathbf{l}_{t-1} + m\mathbf{g} \Delta_t + \sum_{e \in e_{\text{cnt}}} \mathbf{F}_{e,t} \Delta_t \end{bmatrix} \quad (2a)$$

$$\kappa_{e,t} = (\mathbf{p}_{e,t} - \mathbf{x}_t) \times \mathbf{F}_{e,t} + \gamma_{e,t} \quad (2b)$$

$$\gamma_{e,t} = (\mathbf{R}_{e,t}^{\text{x,y}} \mathbf{\jmath}_{e,t}) \times \mathbf{F}_{e,t} + \mathbf{R}_{e,t}^{\text{z}} \tau_{e,t} \quad (2c)$$

$$\mathbf{\jmath}_{e,t}^{\text{x,y}} \in [{}^{\min}\mathbf{\jmath}^{\text{x,y}}, {}^{\max}\mathbf{\jmath}^{\text{x,y}}] \quad (2d)$$

$$\left\| \mathbf{F}_{e,t}^{\text{x,y}} \right\|_2 \leq \mu \mathbf{F}_{e,t}^{\text{z}}, \quad \mathbf{F}_{e,t}^{\text{z}} > 0 \quad (2e)$$

$$\left\| \mathbf{p}_{e,t} - \mathbf{x}_t \right\|_2 \leq {}^{\max}\mathcal{L}_e \quad (2f)$$

where the problem finds center of mass $\mathbf{x}_t$, linear $\mathbf{l}_t$ and angular $\mathbf{k}_t$ momentum, contact force $\mathbf{F}_{e,t}$, center of pressure $\mathbf{\jmath}_{e,t}$ and yaw torque $\tau_{e,t}$ at each contact to minimize user and consensus with the kinematics optimization costs ϕ_t^{dyn}. m is the mass of the robot, $\mathbf{p}_{e,t}$ the position of endeffector e at time t, $\mathbf{g}$ is the gravity vector, Δ_t the discretization time, $\mathbf{R}_{e,t}$ are rotation matrices describing the orientation of the endeffectors (z being the axis orthogonal to the surface), μ the friction coefficient, and ${}^{\max}\mathcal{L}_e$ the maximum distance between the CoM and an endeffector. Equation (2a) to Equation (2c) ensure consistency with the centroidal dynamics,

Equation (2d) are center of pressure bounds, Equation (2e) are friction cone constraints and Equation (2f) ensures that the contact surfaces remain reachable. In this paper, we use the solver proposed in [4] to solve the problem.

B. Inverse kinematics

Given centroidal quantities and desired velocity of the endeffectors, we use a standard differential inverse kinematics algorithm for computing the corresponding whole body motion of the robot. We use three tasks: one tracking task on the centroidal motion which additionally stabilizes the base orientation, a task for each leg to track the desired endeffector velocity (zero velocity in case the endeffector should stay in contact with the ground) and a task to regularize the default posture of the robot. We use the pseudo inverse of the problem to compute joint and base velocities. The computation along the trajectory is then as follows: We get a desired centroidal quantities for the current time step, we compute the inverse kinematics for the current time step, we integrate the velocity forward to obtain the new robot posture. Then the procedure repeats until the end of the trajectory.

C. Whole body controller

We use the whole body controller introduced in [9]. The controller computes the desired wrench $\mathbf{W}_{\text{CoM}}$[1] at the center of mass using a reference wrench $\mathbf{W}_{\text{CoM}}^{\text{ref}}$ and a PD controller of the form

$$\mathbf{W}_{\text{CoM}} = \mathbf{W}_{\text{CoM}}^{\text{ref}} + \begin{bmatrix} \mathbf{K}_c(\mathbf{x}^{\text{ref}} - \mathbf{x}) + \mathbf{D}_c(\mathbf{l}^{\text{ref}} - \mathbf{l}) \\ \mathbf{K_b}(\mathbf{q}_b^{\text{ref}} \boxminus \mathbf{q}_b) + \mathbf{D_b}(\mathbf{k}^{\text{ref}} - \mathbf{k}) \end{bmatrix}, \quad (3)$$

where $\mathbf{K}_c, \mathbf{D}_c, \mathbf{K_b}$ and $\mathbf{D_b}$ are gains, $\mathbf{x}^{\text{ref}}$ and $\mathbf{l}^{\text{ref}}$, $\mathbf{k}^{\text{ref}}$ are the reference CoM position, linear and angular momentum, $\mathbf{q}_b^{\text{ref}}$ is a quaternion for the desired base orientation. $\mathbf{x}$ and $\mathbf{l}$, $\mathbf{k}$ and $\mathbf{q}_b$ are the corresponding measured quantities. The $\boxminus$ operator computes the difference between two quaternions as an angular velocity using the logarithmic map of SO(3).

To achieve the desired centroidal wrench, forces at the endeffectors in contact with the ground are allocated as

$$\min_{\mathbf{F}_i, \boldsymbol{\eta}, \zeta_1, \zeta_2} \sum_i \mathbf{F}_i^2 + \alpha(\boldsymbol{\eta} + \zeta_1 + \zeta_2) \quad (4)$$

$$\text{s.t.} \quad \mathbf{W}_{\text{CoM}} = \sum_{i \in C} \begin{pmatrix} \mathbf{F}_i \\ \mathbf{p}_i \times \mathbf{F}_i \end{pmatrix} + \boldsymbol{\eta}$$

$$F_{i,x} < \mu F_{i,z} + \zeta_1, F_{i,y} < \mu F_{i,z} + \zeta_2, 0 \leq F_{i,z} \quad \forall i \in C,$$

where C contains the indices of endeffectors currently in contact with the ground, $\mathbf{F}_i$ is the desired force at the ith endeffector, $\mathbf{p}_i$ is the position of the ith endeffector with respect to the CoM, α is a large weight and $\boldsymbol{\eta}, \zeta_1, \zeta_2$ are slack variables.

The torques for each leg $\boldsymbol{\tau}_i$ are computed using an impedance controller

$$\boldsymbol{\tau}_i = \mathbf{J}_i^T \left(\mathbf{F}_i + \mathbf{K}(\mathbf{I}_i^{\text{ref}} - \mathbf{I}_i) + \mathbf{D}(\dot{\mathbf{I}}_i^{\text{ref}} - \dot{\mathbf{I}}_i) \right), \quad (5)$$

[1]Note that the centroidal wrench is given by the centroidal force $\mathbf{F}$ and centroidal momentum $\mathbf{M}$ as $\mathbf{W}_{\text{CoM}} = \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix}$.

where $\mathbf{I}_i^{\text{ref}}$ and $\mathbf{I}_i$ are the desired and measured endeffector positions.

III. LEARNING A CENTROIDAL MOTION PLANNER

In this section, we describe the propose approach to compute whole-body motions. Figure 2 shows an overview of the approach. Given a motion description (cost function, desired contact sequence and timing, etc), a motion planner computes the resulting whole-body motion plan. The plan contains full-body kinematic as well as dynamic trajectories, i.e. base position, joint positions, linear and angular momentum, contact forces, which is then tracked by a whole-body controller. A typical trajectory optimization approach would compute the whole body motion plan using an optimizer [4]. However, such optimizers are computationally expensive and cannot compute solutions at high frequencies (e.g. 100 Hz).

In this paper, we propose to learn the motion optimizer instead to generate a typical gait, e.g. walking, jumping, etc, such that the computation can be done in real time. One straightforward manner to accomplish this would be to use motion plans computed by kino-dynamic optimizers to directly learn the entire whole-body motion plan at once. However, in practice such approach does not work very well as long-term predictions can be unstable and thereby make it impossible to predict a whole body motion plan over multiple time steps and contacts.

In contrast, we follow the (exact) kino-dynamic decomposition proposed in [17], [4]. To compute the kinematic quantities of the whole body plan an inverse kinematics problem is solved tracking the previously computed centroidal quantities (center of mass, linear and angular momentum). Instead of computing the centroidal quantities, we predict these quantities using a previously trained neural network. We call this neural network the centroidal neural network, see Figure 2. In this setting, the neural network has to predict the centroidal quantities for only a single time step into the future, instead of a full trajectory. In addition, the kinematic quantities like joint positions of the robot can be used as input to the network for prediction.

Using the centroidal network together with the inverse kinematics has also the benefit of using the ground truth dynamics model of the robot during the prediction. This helps to avoid diverging or unrealistic predictions.

A. Motion patterns

The motions we learn are often repetitive. For instance walking in a straight line is made up of the same motions at different offsets. Though these motions look similar, they would require to make different predictions for the centroidal network as the center of mass is moving over time. To avoid this, the centroidal network predictions are done in a local frame. For this, we divide the motion description into motion patterns. Each pattern specifies a local frame. In this local frame, the repeating motions look the same to the centroidal neural network and therefore it can predict the same output again.
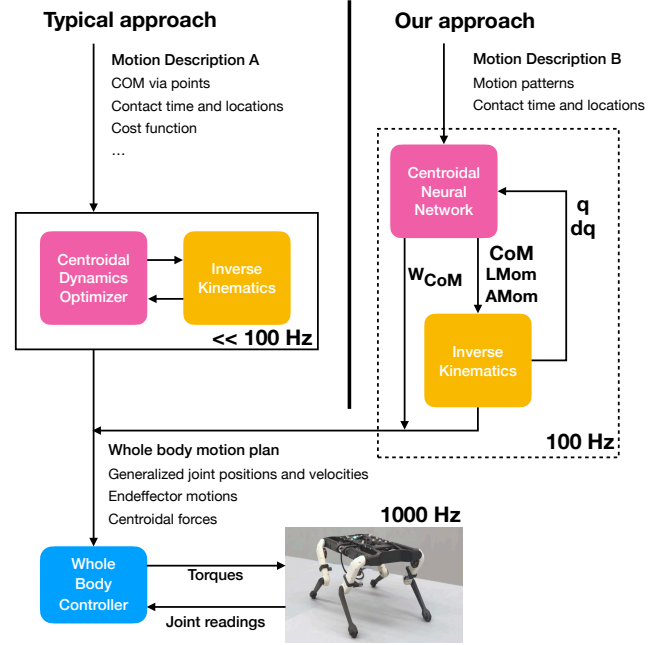

Fig. 2: Pipeline overview: A motion description gets mapped to a whole body motion plan. In typical approaches (left side) this happens using a whole body motion optimizer, e.g. the kino-dynamic optimizer proposed in [17], which cannot be done in real time at 100 Hz. Our approach (right side): a centroidal neural network predicts the next desired centroidal quantities which are mapped to whole-body motions through inverse kinematics. The centroidal neural network gets feedback from robot state (positions and velocities). This method easily runs at 100 Hz. The output of both methods is then sent to a whole-body torque controller (lower part).

B. Centroidal neural network training and design

We use the motion optimizer in [4] to compute training data for the neural network. First, we optimize a motion description using the optimizer. Then, we record the inputs and the computed centroidal motions at every time step. Training the centroidal neural network is then a regression from the centroidal neural network's inputs to the computed centroidal motions. We use the generalized positions and velocities of the robot from the last ten time steps as an input to the network. Here the base orientation is expressed in roll-pitch-yaw angles. We found it beneficial to add Gaussian noise on the generalized position and velocity training data. This helps to make the network prediction more robust to small divergences during the prediction process. Note that our centroidal neural network is not using all the information from the motion description. For instance the information about CoM via points is not used. The network still learns how to predict the motion patterns properly even when using less information.

In addition to the generalized positions and velocities, we

also use contact information from the last 50 until 50 future time steps as input. We found it necessary to have a very large range of contact information provided to the neural network for predicting long ranging effects like swinging the body back after landing. The contact information comes from the motion description (e.g. they can be computed with a contact planner [18]). At every time step each endeffector is either in contact with the ground at a specified position or moving to the next desired contact location. If the endeffector is planned to be in contact with the ground we put a contact duration of zero. Otherwise we encode the time till making the next contact. The contact information for each endeffector at each time step is then the contact location and contact duration. All quantities are expressed in the local frame of the current motion pattern.

In our evaluations, we noticed that the last motion pattern before the robot stops behaved sometimes differently. This is due to the motion optimizer trying to minimize the linear and angular momentum towards the end. In this case, the centroidal network has to change its prediction when the robot needs to stop. To indicate if a motion pattern is the last one to the centroidal network, a binary flag is added to the input. The binary flag is set to ten if the current motion pattern is the last one and otherwise the value is set to zero.

While in principle the centroidal force and centroidal momentum can be computed by taking the derivative of the linear and angular momentum, we found these differentiated quantities to be too noisy to use on the real robot. Therefore, we also added these quantities as outputs of the centroidal neural network. Similarly, the CoM position can be computed by integrating the linear momentum. We found the resulting CoM trajectory to drift away from the desired one. Therefore, we are predicting the CoM position from the centroidal neural network and fuse it with the linear momentum prediction in the inverse kinematics step.

The centroidal neural network is modeled as a feedforward neural network. The first hidden layer is made up of 32 neurons and uses a soft sign activation function. Afterwards there are two more hidden and one output layer using soft sign for the first layer and ReLU activation functions for the second layer. Each layer comes with 128 hidden neurons. During experiments we found it crucial for the prediction performance to have the initial bottleneck with 32 neurons and a saturating output function like soft sign. The output is of size 15: nine outputs represent the three dimensions (x, y, z) each for the center of mass, linear and angular momentum; six outputs represent the centroidal wrench. The network regression is optimized using Adam optimizer using a learning rate of 1e-4 and weight decay of 1e-4. The batch size is 256 samples and we train for 64 epochs (less than one minute on a GPU). The regression is done using a L1 loss between the center of mass, linear, angular momentum and centroidal wrench separately.

IV. EXPERIMENTS

All experiments are performed on a real Solo12 quadruped robot (Figure 1), an open-source [19], [9] torque-controller robot with 12 actuated degrees of freedom (shoulder, hip and knee joints for each leg). In our experiments the absolute base position and orientation of the robot were measured using a motion capture system. These and the experiments in [20] are the first demonstrations of the capabilities of Solo12.

To assess the quality of the generated motions on the robot, we compare the centroidal network predictions compared to the computed motions with the kino-dynamics optimizer. In particular, we look at the tracking performance using the whole-body controller to assess the dynamic feasibility of the generated motions. We compute the tracking error between the planned trajectory and the tracked trajectory when executing the original plan and the plan generated by the centroidal network. The tracking error is computed as the distance error for the center of mass (COM) and the base orientation. We report the mean tracking error as well as the maximum tracking error.

In the following we demonstrate our method on three example motions: first a set of static walks that we compare with the kino-dynamic optimizer, then a longer sequence of static walks that were not part of the training data, and finally a set of jumping motions. We answer the following questions with our experiments: 1) Is our method able to make predictions in real time, 2) is our method able to learn motion patterns that can be executed on a robot and 3) is our method able to generate movements with variables desired contact sequences?

A. Static walk motion

We create a motion generator that allows us to step in any direction. The motion generator performs always the same sequence of leg motions: front left, hinge right, front right, hinge left. For training our centroidal neural network, we generate motions with three consecutive static walks using our motion generator. This results in a walk of 2.9 seconds duration. Each of the walks is in a randomly sampled direction. We assign a different motion pattern to each of these static walks. We generate 60 motion descriptions and use them to train the centroidal neural network described in Section III-B.

To test the quality of the learned centroidal neural network, we generate eight test static walks samples with ten consecutive static walks each with a total duration of 14.1 seconds. As with the 60 training samples the directions of the static walks are sampled randomly.

The tracking results for the static walks are shown in Figure 4a. From the plots we see that the tracking of the center of mass as well as the base orientation is very similar between the original plan and the network generated plan. Overall, the robot is able to track all motions generated by the neural network without falling.

B. Marathon motion

This motion tests the ability of the method to generate long lasting motions. We randomly generate a stepping sequence made up of 50 steps and test the ability of our approach to generate motions for long step sequences. The results of
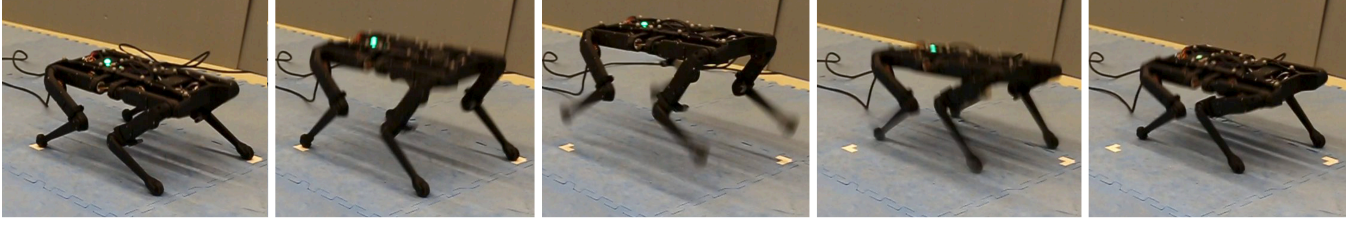
Fig. 3: Screenshots from a sideways jumping motion. The images are 0.1 s apart.

(a) Tracking error during walking
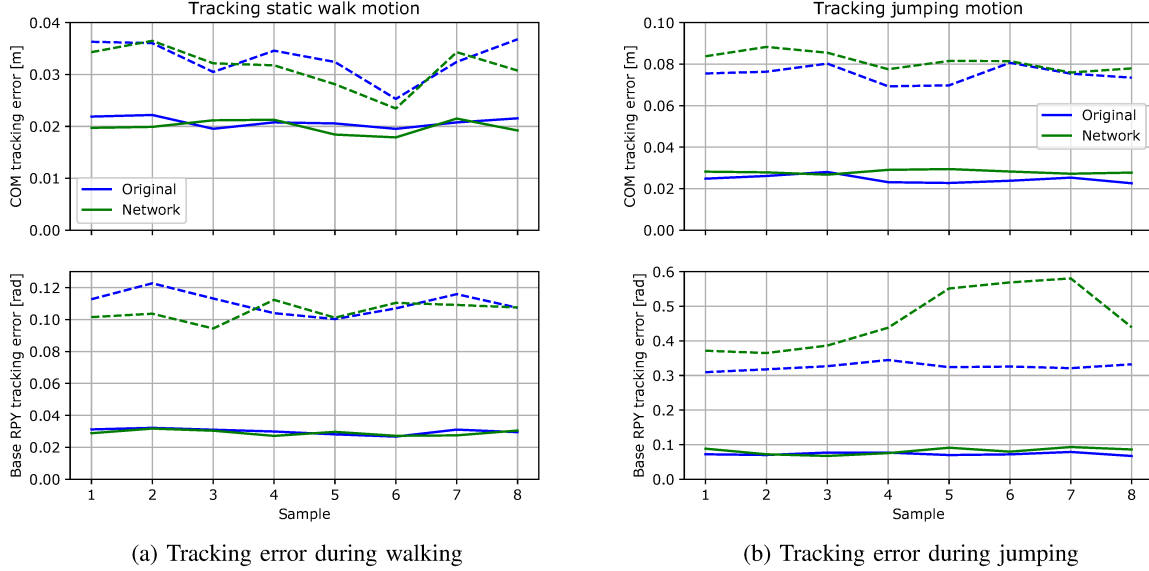
(b) Tracking error during jumping

Fig. 4: (Left) Results from tracking eight random static walks over ten steps on the robot. (Right) Results from tracking eight motions with eight random jumps each on the robot. In blue we show results with the original plan optimized with the kino-dynamic optimizer [4] (labeled "Original") and in green with the plan generated using the centroidal neural network (labeled "Network"). Top plot shows the tracking error for the CoM; lower plot shows the tracking error for the base angular orientation. Solid line is the mean tracking error while dash line shows the maximum tracking error. The centroidal network is capable of producing motions of similar quality as the original planner.

the marathon task are shown in the video provided with this submission. As one can see, the robot is able to execute the 50 static walk steps without any problem, although the sequence of steps and associated motion plans were not part of the training data.

C. Jumping motion

Beside walking, we are interested in more dynamic motions, such as jumping. We create a motion generator that generate jumps in random directions. Similar as before for the walking motions, we generate 60 motion descriptions consisting of three different jumps as training data. Each jump is assigned a different motion pattern. For testing the learned centroidal network, we generate eight jumping motions with 10 jumps each. Each jump goes into a randomly selected direction, to demonstrate the ability of the approach to generate movements from previously unseen step sequences.

The results for the jumping task are shown in Figure 4b; screenshots of one jump are shown in Figure 3. As one can see, the COM tracking is very similar when tracking the

original plan and when tracking the plan generated by the neural network. For the tracking of the base orientation, the mean tracking error is again similar between the two tracking methods. However, the maximum tracking error when using the neural network plan is up to nearly twice as large as the original plan in three out of the eight cases. Again, the robot managed to track all plans generated using the centroidal neural network without falling.

For one of the jumping task, detailed plots of the centroidal neural network prediction are shown in Figure 5. We notice that it can generate predictions very close to what the original planner would compute (i.e. close to the optimal motions). The result after running the inverse kinematics for the same section is shown in Figure 6. We notice that the network generates kinematically feasible motions. Further, the inverse kinematic part plays an important role as it filters the motion before sending it to the whole-body controller.

D. Timing

Timing information about the motions and how long it takes to optimize them using the original method and the
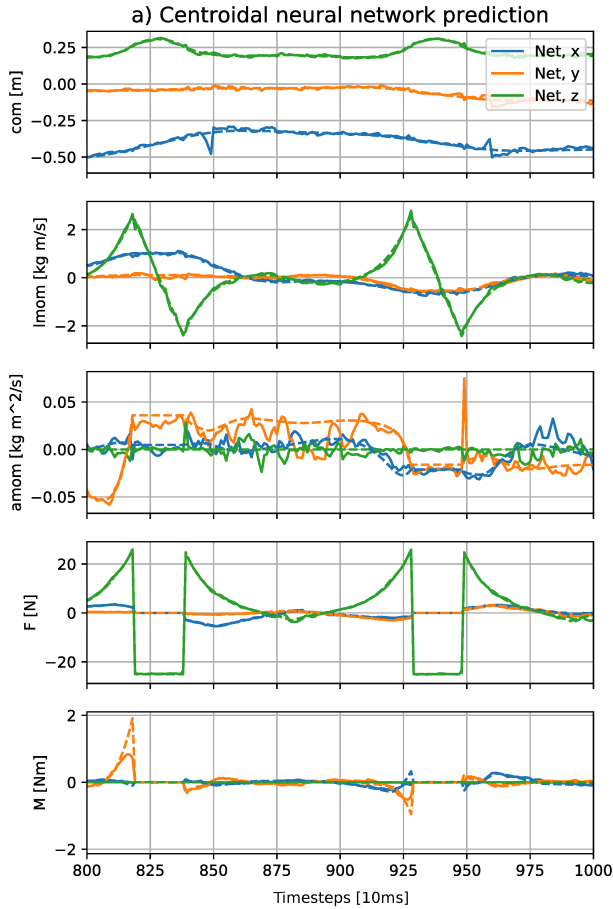
Fig. 5: Comparing the centroidal neural network predictions to the original plan predictions for two seconds along one of the test jump sample trajectories. The output from the centroidal neural network is plotted as solid lines, the original plan as dashed lines.

centroidal neural network is shown in Table I. Here we show the time it takes for the network to generate a complete plan when integrating forward for the whole duration of the motion. As one can see, the neural network method is significant faster: For the walking motion it is 22.1 times faster than the original method and up to 41.3 times faster on the jumping motion. Note in addition, that the prediction time is always lower than the plan duration for the network method. This makes it possible to predict the plan in real time.

E. Discussion

During the motions the robot moves outside of the area it was trained on. This shows that using the motion patterns the robot is able to extrapolate motions to new areas for the same repetitive motions. In addition, as seen in Table I our method is able to make predictions in real time as the compute time is lower than the plan duration for all the motions. Last but not least, the proposed method was able to generate plans for static walks as well as for jumping motions. This demonstrates its ability to generate motions for dynamic tasks involving multiple switching contacts.
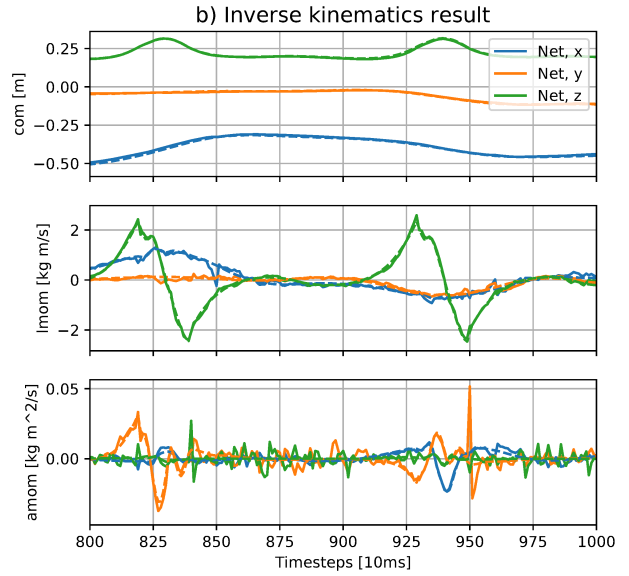

Fig. 6: Comparing result of running inverse kinematics based on the centroidal neural network input from Figure 5 and the original plan for two seconds along one of the test jump sample trajectories. The output from centroidal neural network inverse kinematics is plotted as solid lines, the original plan as dashed lines.

Task	Motion duration	Method computation time		Speedup
		Original	Network	
Walk	14.1 s	35.4 s	1.6 s	22.1×
Marathon	70.1 s	236.6 s	8.3 s	28.5×
Jump	11.0 s	99.1 s	2.4 s	41.3×

TABLE I: Timing information for the different motions. The table shows how long each motion is and how long it takes to compute the plan for the motion using the original kino-dynamics optimizer [4] and the proposed neural network method. The speedup is how much faster the centroidal neural network method is over the original method.

V. CONCLUSION AND FUTURE WORK

In this paper, we considered the problem of speeding up a whole body motion planner to real time. We do this by learning a centroidal neural network. We use the centroidal neural network together with an inverse kinematics solver to solve for a whole body plan, which is then tracked using a whole body controller. In our experiments we can show that we achieve this goal and predict faster than real time. The presented method has been validated for three tasks that are about static walking in different directions as well as jumping in different directions. This demonstrates that the method works for contact rich, dynamic tasks. The use of motion patterns allows us to generalize to new areas outside of the training data. While the current motions have only been evaluated on flat ground thus far, we are planning in future work to extend the motions to more complex non-planar scenarios and to study extensions of the approach to more complex robots such as a humanoid.

References

[1] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[2] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[3] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2536–2542.

[4] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," *arXiv preprint arXiv:2010.01215*, 2020.

[5] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.

[6] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[7] E. Daneshmand, M. Khadiv, F. Grimminger, and L. Righetti, "Bipedal walking control using variable horizon mpc," *arXiv preprint arXiv:2010.08198*, 2020.

[8] G. Mesesan, J. Englsberger, B. Henze, and C. Ott, "Dynamic multi-contact transitions for humanoid robots using divergent component of motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4108–4115.

[9] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[10] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization." in *RSS*, 2014.

[11] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," *arXiv preprint arXiv:2002.06719*, 2020.

[12] Y.-C. Lin, L. Righetti, and D. Berenson, "Robust humanoid contact planning with learned zero-and one-step capturability prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2451–2458, 2020.

[13] T. Kwon, Y. Lee, and M. Van De Panne, "Fast and flexible multilegged locomotion using learned centroidal dynamics," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 46–1, 2020.

[14] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[15] J. Siekmann, S. Valluri, J. Dao, L. Bermillo, H. Duan, A. Fern, and J. Hurst, "Learning memory-based control for human-scale bipedal locomotion," *arXiv preprint arXiv:2006.02402*, 2020.

[16] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6129–6136, 2020.

[17] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2703–2710.

[18] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[19] "Open dynamic robot initiative," https://open-dynamic-robot-initiative.github.io/, accessed: 2020-08-13.

[20] P.-A. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a Reactive Walking Controller for the New Open-Hardware Quadruped Solo-12," in *2021 IEEE International Conference on Robotics and Automation - ICRA*, Xi'an, China, May 2021. [Online]. Available: https://hal.laas.fr/hal-03052451