Quadratic Encoding of Optimized Humanoid Walking

Junggon Kim, Nancy S. Pollard and Christopher G. Atkeson

Abstract— **In this paper we show that optimal stepping trajectories and trajectory cost for a walking biped robot on rough terrain can be encoded as simple quadratic functions of initial state and footstep sequence. In order to find this encoding, we build a database of optimal walking trajectories for a 3D humanoid model by sampling the input space (initial state and footstep sequence) and solving a physically-based trajectory optimization problem for each sample. Then, the function coefficients are obtained by fitting the data using least squares. The performance of the proposed method is evaluated by comparing the function values with other optimal walking motion data generated with different footstep samples. As an application, we use a quadratic function to calculate the effort cost used in finding an optimal footstep sequence with an A* algorithm. Our study shows that a simple function can encode optimal walking effectively, which provides a fast alternative to online optimization of walking with full body dynamics.**

I. Introduction

Despite its computational cost, a full body dynamics model is useful for improved understanding, prediction, and control of dynamic behaviors such as locomotion for humanoid robots. However, it has been rarely used in planning problems which involve some form of optimization and thus require a large number of evaluations of the dynamics, especially when the planning must be done fast or online. In this paper we focus on humanoid walking and explore ways of considering the full body dynamics in online locomotion planning problems such as footstep planning.

We argue that many features of optimized walking can be encoded with simple functions, and that using these functions can be effectively equivalent to considering the full body model dynamics. For example, we find a simple model that relates a walking input, such as the initial state and a sequence of footsteps, to the torque expenditure of an optimized walking motion by learning the relationship with massive amounts of training data. One feature of the walking dynamics (i.e., the effort cost) is encoded in a simple function, and this can be used in searching for an optimal footstep sequence that minimizes the effort cost. At every iteration in planning, we directly evaluate the quality score of a current step sequence from the learned function instead of running a trajectory optimization (for finding an optimized walking motion) and calculating the torque expenditure with inverse dynamics.

Two major questions arise here: 1) Does there exist a simple function that can effectively encode a dynamic feature of optimized walking (e.g., the effort cost)? 2) How can we obtain enough data for training the function to encode the feature?

All the authors are with The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213. {junggon, nsp, cga}@cs.cmu.edu

In this paper we explore these questions particularly in the domain of humanoid walking. In order to generate a walking motion from a given input, we use a physically-based trajectory optimization. Since we need a large amount of motion data, we run the optimization for many possible inputs in parallel on a cluster supercomputer. After building a motion database, we encode the walking motions with a simple function. According to our study, we have found that a quadratic function performs well in encoding humanoid walking motions.

Our approach shares similar philosophies and ideas with existing approaches in robotics and biomechanics. Pursuing simplicity or a simple representation is inspired by the idea that the fundamentals of walking should be simple, which can be found in much previous research, including passive walking and proposals for a canonical walking function [1], [2], [3].

In Section II we describe our method to build a large walking motion database. Then, in Section III, we encode the walking dynamics underlying the motions with a quadratic function using least squares and evaluate the performance of the function in predicting the dynamics of a new walking motion. As an application, we apply the encoded dynamics (the quadratic function) to planning an optimal footstep sequence for a humanoid using A* search in Section IV. We discuss some interesting issues related to our approach in Section V.

II. Trajectory Database

In this section we explain how to build a database of optimal walking trajectories for a 3D humanoid model. The trajectory data in the database will be used to encode the walking motion with a simple function.

A. Overview

A footstep is defined as (p_x, p_y, p_z, θ) where (p_x, p_y, p_z) is the foot position on the ground in 3-dimensional space and θ denotes the foot angle in the yaw direction (or the rotation around z-axis). A relative footstep system is used – a footstep represents the position and orientation of the touchdown foot with respect to the stance foot.

Our walking motion database contains not only single step motions but also multiple step motions. Each walking motion is obtained from a given initial state and a step sequence by solving a trajectory optimization problem (II-B). Considering multi-step walking motions instead of single steps in footstep planning allows the planner to look further ahead, which can

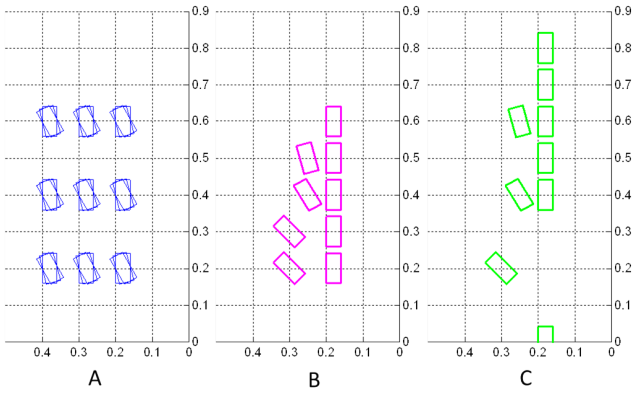

Fig. 1. Three footstep primitive sets used in building our motion databases (for left foot). For the right foot, the symmetric footsteps are used. Only the footsteps with zero height ($p_z = 0$) are depicted here. The unseen footsteps have the same configuration but with different height values ($p_z = \pm 0.1$ m). The number of total footsteps are 81, 27 and 27 respectively. The stance foot is at the origin and not shown.

improve the quality of the planned step trajectory (Section IV).

A set of footstep primitives is used to sample step sequences for building a walking motion database. For each sampled step sequence, we find an optimal walking motion that starts from a given initial state and follows the given steps, and we save it to the database. We first find optimal motions where the robot takes only a single step. Once every footstep primitive has been processed in this manner, we find two-step walking motions by adding an additional step to the existing single step motions and solving the corresponding optimization problems. We repeat the step extension process up to a target number of steps. In our implementation, exhaustive step extension, which produces $2m^n$ n-step sequences with m primitives, is used up to a certain number of footsteps and then the step sequences are extended randomly to the target number of steps. For example, with a set of 81 primitives (footstep set $\mathbb{A}$ in Fig. 1), we do the exhaustive extension up to two steps – this results in 162 single steps and 13,122 ($= 2 \cdot 81^2$) double-step sequences – and then add steps randomly up to five steps. Finally, the whole procedure is repeated several times with different initial states. We solve the large number of optimization problems in parallel on a cluster with several hundred CPU cores.

We have built three databases ($\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$) with the three sets of footstep primitives shown in Fig. 1. Eight different initial states were used to initiate the walking motions. The database $\mathbb{A}$ generated from the 81 footstep primitives (footstep set $\mathbb{A}$) has 202,579 motions with up to five steps and was used for encoding the features of optimal motions with quadratic functions. The other two databases $\mathbb{B}$ and $\mathbb{C}$ have 297,299 and 291,648 motions with up to three steps respectively, and were used for evaluation purposes. Note that all walking motions in the databases are non-periodic and have either different footstep sequences, different initial states, or both.

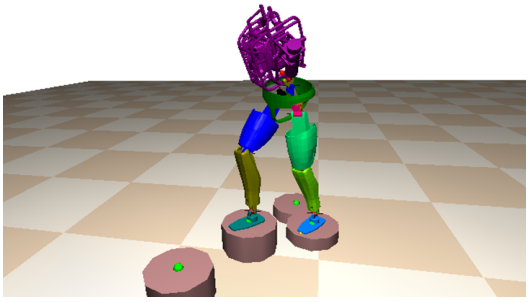

Fig. 2. Our robot model (15 DoFs) and a step sequence including turning and height change. From the given step sequence, we find an optimal walking motion using trajectory optimization. We repeat this for many possible step sequences and build a walking motion database which will be used to train a simple function encoding the dynamics of the optimized motions.

B. Trajectory Optimization

We find a walking motion using a trajectory optimization technique that has been widely used in robotics and computer graphics [4], [5], [6], [8]. When we find an optimal walking motion from a given initial state and an n-step sequence, we internally set up a bigger (n+m)-step problem by adding additional m steps to the end. Then, we impose a periodic walking state at the last step so that the robot reaches the periodic walking state in m steps. In our implementation we set m = 1 to make the optimization problem size as small as possible, and this moves the robot back to a predefined periodic mode in a single step after finishing the given n steps.

Our robot model (Fig. 2) has 15 revolute joints – each leg has 6 DoFs (3 at the hip, 1 at the knee, and 2 at the ankle) and the waist connecting the pelvis and the torso has 3 DoFs. The arms and head are assumed to remain at the same configuration during walking and their mass and inertia are added to the torso. The total mass of the model is about 97 kg.

The trajectory of each joint is represented with a series of quintic polynomials whose coefficients are the optimization variables. The total time of the walking motion and the timing of each step are determined by the optimization. An optimal set of the curve parameters that minimizes a physically-based objective function while satisfying the footstep and other constraints is found by using a numerical optimization technique. Our objective function

$$J = \int \left(w_1 \tau^T W_j \tau + w_2 \delta^2 + w_3 \kappa^T \kappa \right) dt + w_4 I_z^2 + \int \left(w_5 q_u^T q_u + w_6 \dot{q}_u^T \dot{q}_u + w_7 \ddot{q}_u^T \ddot{q}_u \right) dt \quad (1)$$

penalizes the joint torques (τ) and the vertical component of the impulse (I_z) to the swing foot at the moment of collision with the ground. We calculate the joint torques using the inverse dynamics of the robot model, and the swing foot impulse is obtained by assuming an inelastic

collision with the ground[1]. We also penalize the upper body movement ($q_u, \dot{q}_u, \ddot{q}_u$), the lateral deviation (δ) of the swing foot trajectory from the straight line connecting the previous and target foot positions to avoid self collision between the feet, and the reaction moments in roll and pitch directions at the swing leg knee (κ) to reduce undesirable swing leg wobbling. The given initial state, footstep sequences and final periodic walking state are handled as equality constraints in the optimization problem. The joint angle and torque limits, the height clearance for the swing foot, and the Coulomb friction cone condition for the contact forces (at the stance foot) and the swing foot impulse are considered as inequality constraints. The feet are constrained to be parallel to the flat ground during walking, and the double support phase was ignored for problem simplification. We used SNOPT [9], a software package for large-scale nonlinear programming using a sequential quadratic programming algorithm, in the optimization.

C. Implementation Detail

When we find an n-step walking motion using optimization, we use the optimized (n-1)-step walking motions which have already been computed to initialize the n-step optimization process. This improves the speed and success rate of the optimization process. Nevertheless, there are cases where the last foot position of the initial trajectory is far from the target position and we experienced a large number of failures in optimization. In addition, due to the large number of constraints such as the joint torque limits and the friction cone constraints, the feasible region is too small for the optimizer to find a solution from an infeasible initial trajectory.

In such circumstances, a continuation method, which gradually changes the constraints [7], [10], can alleviate the narrow feasible region problem and improve the success rate of the optimization process. In our implementation, when the initial trajectory has a different footstep sequence from a target, we gradually change the constraint toward the target – we move footstep positions and angles at most 5 mm and 1 deg at a time. Also, at each trial with a certain footstep constraint setting, we subdivide the optimization problem into multiple steps and increase the level of the other constraints. For example, we start the optimization by ignoring many physical constraints such as the torque limits and the friction cone constraints, and then restart the optimization with added constraints in the next step. Thus, for certain cases, finding an optimal walking motion can take more than a few hours. Most cases, however, required less than five minutes (in the case of three step motions).

After the optimization is finished, the computed walking trajectory is examined using a quality metric. We calculate the average deviation of the center of pressure from the stance foot center in time, $\frac{1}{T}\int_0^T ||p_{cp} - p_f||^2 dt$ where T is the total walking time and p_{cp} and p_f denote the center of pressure and the stance foot position in x-y plane respectively, and add the trajectory to the database if the value is less than 0.005.

[1]The collision impulse to the swing foot is obtained as $I = -\left(JM^{-1}J^T\right)^{-1} J\dot{q}_f$ where $M = M(q_f)$, $J = J(q_f)$ denote the system mass matrix and the swing foot Jacobian matrix at the moment of swing foot landing ($t = t_f$), and $(q_f, \dot{q}_f)$ is the system state right before the collision. The impulse changes the joint velocity instantly as $\dot{q}_{f+} = \dot{q}_f + M^{-1}J^T I$, and the new state $(q_f, \dot{q}_{f+})$ initiates the walking motion for the next step.

III. Quadratic Encoding of Walking

We encode the optimal walking motions with a simple function. An encoding function maps the initial state and step sequence to a specific feature of the optimal walking motion, and the function coefficients are fit to the precomputed data set in the database using least squares. As the dimension of the full state of the robot model is prohibitively large, we use the position and velocity of the center of mass (relative to the stance foot coordinate frame) to represent the initial state. Thus, for n-step walking motions, the input of the function becomes a vector with length $6 + 4n$.

Although we reduced the dimension of the initial state by abstracting it to the center of mass position and velocity, the initial state space is still vast and many walking motion samples with different initial states are necessary to train an encoding function. As mentioned earlier, however, we have used a small number of states to initiate the multi-step walking motions in the database. In order to create enough n-step motions having different initial states, we extract the last length n gaits from (n+k)-step motions in the database ($k > 0$). For example, from a five-step walking motion, we extract the $\{3, 4, 5\}$-th steps to obtain a three-step walking motion which has a different initial state. Note that the new walking motion meets the same final condition we used in trajectory optimization — the robot walking is required to reach a predefined periodic mode in an additional step (Section II-B). In this way we obtain more n-step motions starting from diverse initial states.

A quadratic function is used to encode a specific feature of the optimal walking motions:

$$f(x) = \frac{1}{2}x^T Ax + b^T x + c \tag{2}$$

where $x \in \Re^s$ is the input vector ($s = 6 + 4n$) and $A \in \Re^{s\times s}$, $b \in \Re^s$ and $c \in \Re$ are the coefficients that must be determined by least squares, and A is set to be symmetric. The number of the coefficient elements for n-step walking motions is $\frac{1}{2}s(s+1) + s + 1$, and in the case of 3-step ($s = 18$), there are 190 coefficients.

We encode several features of the non-periodic multi-step optimal walking motions such as the effort cost ($\int \tau^T \tau dt$, the sum of squared joint torques) and the position and velocity of the center of mass at the end of the motion. These features will be used in footstep planning using A* search in Section IV. We will also discuss the possibility of encoding more features such as the full trajectory of the center of mass position and its timing.

The walking motion database $\mathbb{A}$, generated from the 81 footstep primitives, is used to find the function coefficients using least squares. In the case of 3-step walking, about

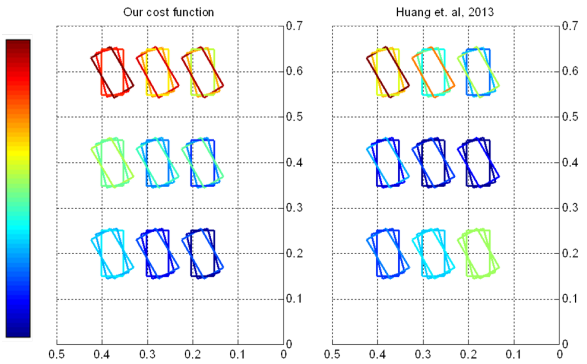

Fig. 3. Comparison of our step cost function with a previous cost function. The stance foot is located at the origin. (Left) step cost from our quadratic function. (Right) step cost from Huang et. al, 2013.

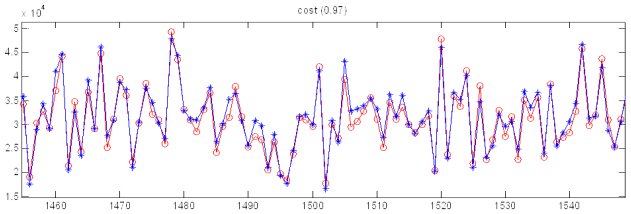

Fig. 4. Comparison of the estimated effort costs and the optimization results for 3-step walking motions randomly chosen from an untrained database. Correlation coefficient of the two data = 0.97. (8,000 samples tested but only part of them is shown here)

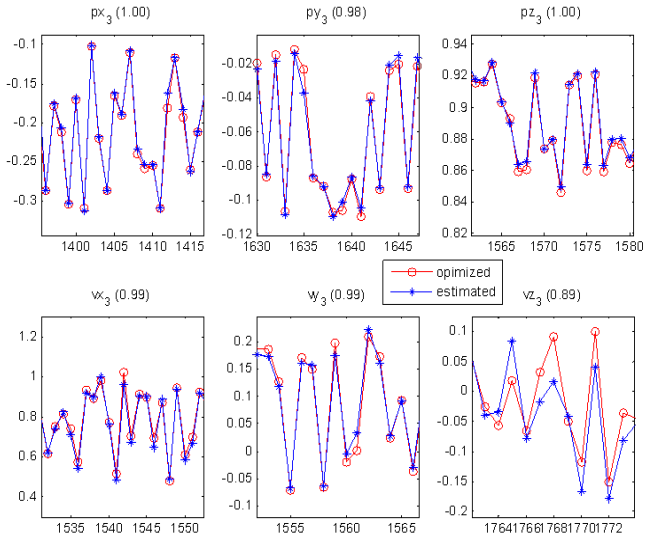

Fig. 5. Comparison of the estimated final state (center of mass position and velocity) and the optimization results for 3-step walking motions randomly chosen from an untrained database. The correlation coefficient of the two data is shown in the sub-figure title.

20,000 of the randomly selected motions from the database were used for data fitting.

Fig. 3 shows an overall shape of the effort cost function and compares it with a heuristically designed energy cost function by Huang et. al, 2013 [11]. One noticeable difference is that, in our function, the cost decreases as the step length decreases while the previous function returns higher energy cost when the step length becomes smaller than a certain distance. This is because the previous energy function assumes a constant walking speed so a smaller step has a higher step frequency, which results in a higher energy cost. Another interesting point is that our cost function shows a rapid increase in the cost as the foot is turned especially with large but narrow steps (the top/right area in the figure). Although the focus of this work is not on human walking, we believe this reflects a real difficulty that humans also experience when performing that action. The previous energy function does not capture this feature.

We evaluated the performance of the encoded functions by comparing their values with different optimal motion data. For comparison we randomly picked 8,000 3-step walking motion samples from database $\mathbb{B}$, which was generated with a different set of footstep primitives and not used in the data fitting. For each motion sample, we evaluate the function values from the initial state and footstep sequence of the motion and compare the values with the original motion data which was obtained from the optimization.

Fig. 4 shows some of the results for the effort cost function. The red circles represent the original cost data obtained using inverse dynamics for the optimized motion, and the blue stars are the estimated values using the quadratic function. The encoded value and the optimized result show a high correlation, which means that the quadratic function can reliably predict the effort cost of an optimized motion without having to use any trajectory optimization and inverse dynamics.

We also investigated the encoding performance for the center of mass position and velocity at the end of the motion, and the results are shown in Fig. 5. Interestingly, the performance was even better for estimating the center of mass position than the effort cost. This is likely because the position is a direct output of the trajectory optimization, while the effort cost requires further computation using inverse dynamics. The center of mass velocity can also be estimated with a high accuracy except for the vertical component (v_z), which has a relatively low correlation compared to the others. We believe this was caused by the discontinuous velocity change by the impulse to the swing foot from the ground and the vertical velocity was the most affected by this.

It would be interesting to explore what other features of the optimal walking motion can be accurately approximated by simple quadratic functions. We examine how well they can approximate the entire center of mass trajectory of an optimal walking motion from a given initial state and a sequence of footsteps. For this, we sample K points from each gait and encode the center of mass position at each sampling time $t^k = t_0 + (t_f - t_0)k/K$ where t_0 and t_f is the start and end time of a gait and $k = \{1, \cdots, K\}$. Since we need footstep timing information to reconstruct the center

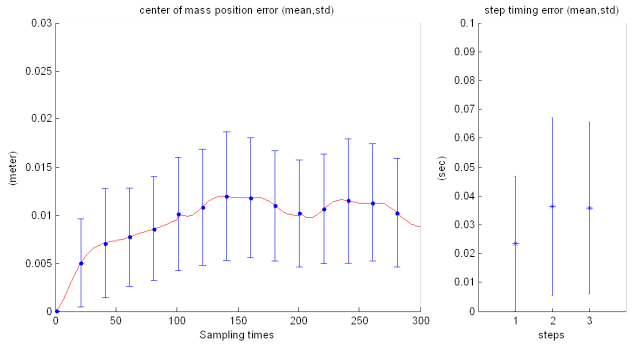

Fig. 6. The mean and standard deviation of the error in the estimated center of mass trajectory of 3-step walking motion. (Left) Position estimation error at each sampling time. (Right) Footstep timing estimation error at each step.

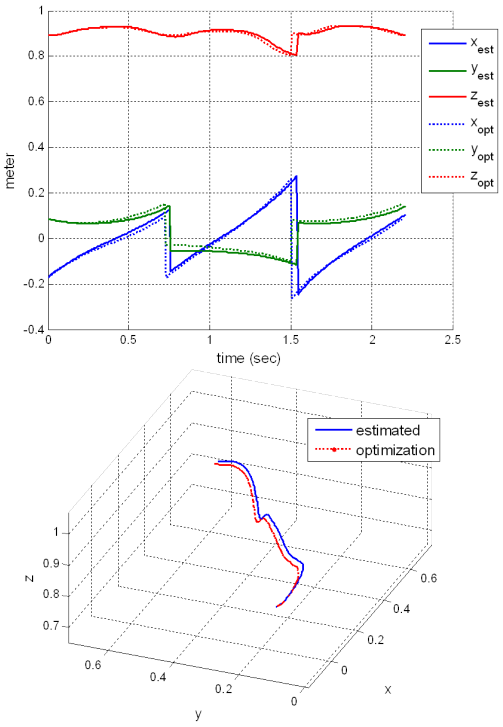

Fig. 7. Comparison of an estimated center of mass trajectory with the optimized motion data (line: estimated, dotted: optimized). The input footsteps include turning and stepping down. (Upper) Center of mass position relative to current stance foot. (Lower) The same trajectory in 3D space seen from the world coordinate frame.

of mass trajectory in time, the time length of each gait is also encoded. When we encode 3-step walking motions with 100 sample points per gait, we have 903($= 3{\cdot}3{\cdot}100{+}3$) functions to be trained using least squares. Note that data fitting of the functions can be run in parallel, and also the center of mass position at each sampling time can be decoded from the quadratic functions in parallel.

For every sampling time t^k, we computed the average deviation of the estimated center of mass position from the original motion data (Fig. 6). The average position error was less than 1.5 cm at every sampled point. For the footstep timings, the quadratic functions estimated the time length of each gait with less than 40 msec error on average. In Fig. 7 we showed an estimated center of mass trajectory of a 3-step walking motion and compared it with the original trajectory. Note that, in the upper figure, the position is relative to the stance foot. Although the original motion is complex and contains turning and stepping down, our quadratic functions estimate the entire trajectory reliably.

IV. Application: Step Planning with A* Search

In this section we apply the encoded walking data to a humanoid step planning problem – finding an optimal step sequence that minimizes the robot effort (or the sum of squared joint torques) using an A* search algorithm. A* search with inflated heuristics has been widely used in many applications due to its fast search speed [12], [13]. Its solution is sub-optimal and the inflation factor sets a bound on the sub-optimality [14]. Decreasing the factor results in more optimal solution but generally requires a longer search time.

For footstep planning, the Euclidean distance between the current position to the target is often used as the *cost-to-go* (h: a heuristic or the estimated cost to the target). We use the quadratic effort-cost function to evaluate the *cost-to-here* (g: the effort cost of the currently found step sequence from the start). The effort cost is normalized by dividing with a reference cost per distance which has been obtained from optimal periodic straight walking with a step length of 0.4 m. This reinterprets the effort cost value as the distance that can be travelled with straight periodic walking. Given an initial state and a step sequence, the quadratic function provides an accurate effort cost estimation without having to run the time-consuming optimization for finding a walking motion and inverse dynamics for calculating the torque expenditure of the motion. This makes it possible to consider the full humanoid dynamics in on-line step planning.

We discretize the footstep space with a small action set which is a common approach in step planning [15], [16]. We use the 9 planar footsteps shown in Fig. 1 ($\mathbb{B}$) as the action set, and obtain the step height value from a terrain height map. Note that the action set is different from the 81 footstep primitives that has been used for encoding the walking motions with quadratic functions. Since the footsteps were constrained to remain horizontal in our trajectory optimization setting, the quadratic function does not consider the effect of terrain slopes at foot placements on the effort cost. Thus, in the case of uneven terrain we consider a terrain cost to penalize placing feet on a slope in addition to the effort cost. We assumed a foot placement on a 10 deg slope imposes an additional cost equivalent to the effort cost of the straight optimal walking for 1 m and increase the terrain cost quadratically to the slope angle. Note that, since the quadratic encoding technique is not necessarily subject to a particular optimization method, it should be possible to remove the ad hoc terrain cost by training the effort cost function with a

TABLE I

FOOTSTEP PLANNING RESULTS (FIG. 8)

	multi-step reasoning (n=3)			single step reasoning		
	steps	effort	terrain	steps	effort	terrain
Path 1 (top)	30	17.6	3.8	34	21.3	3.8
Path 2 (middle)	64	40.3	7.5	68	44.7	11.3
Path 3 (bottom)	34	20.6	15.0	34	23.4	15.0

new set of optimized walking motions considering slope at foot placement.

As our cost function can handle multiple steps, we can make the search algorithm look further ahead to find a better path. In our A* search implementation, each A* node keeps the two cost values (g and h), the footstep information (position and orientation) and the robot state (center of mass position and velocity). The algorithm expands the search tree by creating child nodes with predefined footstep actions, and for each child, evaluates the walking cost to there and the heuristic cost from there to the goal. If we set the algorithm to look n-steps ahead, it traces back to its n-th ancestor (every node knows its parent) and then evaluates the effort cost for the future n steps toward the child node using the quadratic function. The g value of the child node becomes the sum of the ancestor's g and the n-step cost. At the same time, the state of the child node is set by calling the other six quadratic functions which have been trained for estimating the center of mass position and velocity at the end of walking. The total cost of the node is obtained as

$$f = g + t + \epsilon h \tag{3}$$

where t is the accumulated terrain cost and ϵ is the inflation factor of the heuristic, and this is compared with other node costs for choosing a branch for the next search.

In Fig. 8 and Table I we compare the step planning results of multi-step reasoning (n = 3) and single step search (n = 1). It is not currently feasible to compare our results to using trajectory optimization instead of the approximate functions. Three different start positions were tested here. As expected, looking further ahead helps the planner find a better step sequence. In all three cases, the multi-step planning found lower effort cost paths. Interestingly, the two bottom paths are identical but the multi-step planner reasoned the robot can walk along the path using less effort with more efficient multi-step walking motions. Another interesting point is that the multi-step planner can turn toward the target earlier if it is better because it can see further ahead while the short-sighted single step planner hesitates because of the current high cost in turning. Increasing the heuristic inflation factor could guide the single step planner to make an early turn, but this often results in an aggressive path requiring higher cost overall. The elapsed time for the A* search is affected by many factors including the locations of the start and target points, the terrain shape and the heuristic inflation factor. In this test, the single-threaded multi-step planning for the longest path (the middle one, 64 steps) took 260 msec in our implementation on a laptop equipped with an Intel i7 CPU.

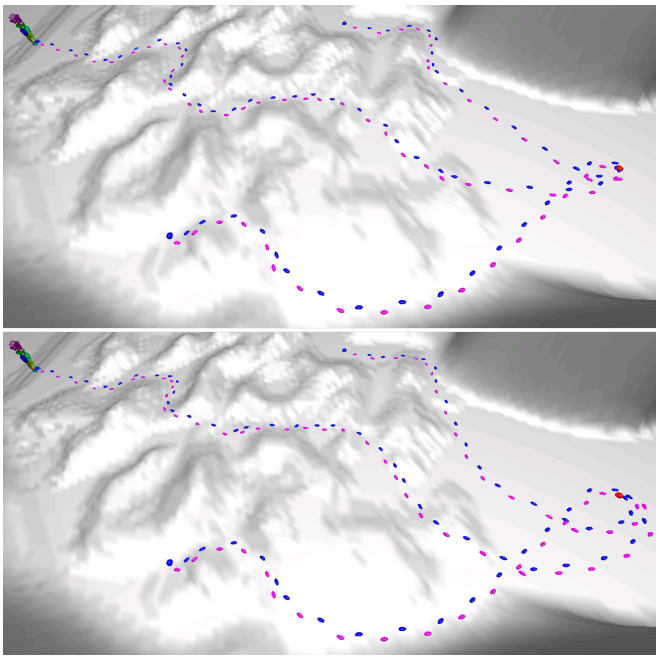

Fig. 8. Multi-step reasoning (upper, n=3) vs. Single step reasoning (lower) : Step sequences, starting from three different start locations to a single goal location (red ball in the right), planned on a rough terrain with an A* search using the footstep primitive set $\mathbb{B}$ and the quadratic cost function obtained from database $\mathbb{A}$. The same heuristic inflation factor ($\epsilon = 4$) and terrain cost function were used.

V. DISCUSSION

A. Additional Comparisons

1) Linear vs. quadratic: We have used quadratic functions to encode optimized walking. Here we provide information on encoding with a linear function $f(x) = b^T x + c$. Fig. 9 compares the performance of the linear encoding with the quadratic one. The same 8,000 3-step walking motion samples were used in the evaluation. As expected, the linear encoding shows degraded performance in estimating the features of walking motion. However, as the linear function has a much smaller number of coefficients than the quadratic function (19 vs. 190 in the case of 3-step), it could be a practical choice for small resource systems.

2) Database $\mathbb{B}$ vs. $\mathbb{C}$: We have trained our quadratic functions using database $\mathbb{A}$ (generated with footstep set $\mathbb{A}$), and evaluated the performance of the functions using database $\mathbb{B}$. Here, we make another test with database $\mathbb{C}$ and compare the result with the previous one. Note that some of the steps in $\mathbb{C}$ are outside of the scope of the trained footstep set $\mathbb{A}$ (Fig. 1). Fig. 10 shows the two test results and there was no significant change in the performance in estimating the features of walking motions except for the vertical velocity component (v_z). We think this was caused by the increased swing foot impulse by the larger steps in footstep set $\mathbb{C}$.

B. Data-driven approach with local interpolation

Since our method uses a massive amount of walking data to encode the underlying walking dynamics, one possible

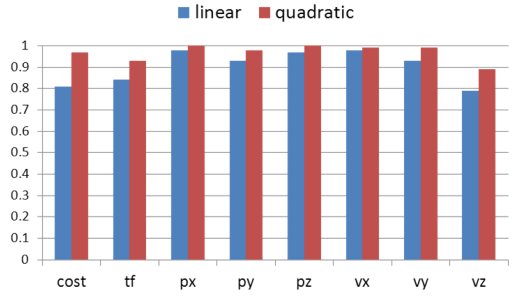

Fig. 9. Performance comparison between linear encoding (blue) and quadratic encoding (red). Correlation coefficients of the estimated quantities and the optimization results are shown. (From left to right: walking effort cost, footstep timing (last gait), position (p_x, p_y, p_z) and velocity (v_x, v_y, v_z) of the center of mass at the end of 3-step walking motion.)

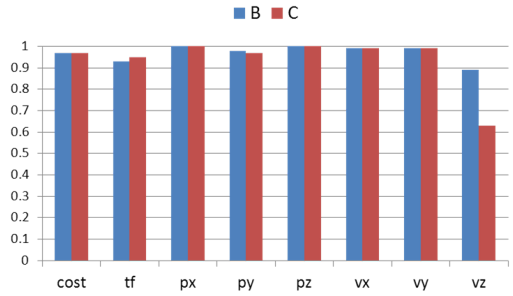

Fig. 10. Performance of the quadratic encoding in estimating walking motions in different databases ($\mathbb{B}$ and $\mathbb{C}$).

alternative for fast evaluation of dynamics is to save the data and do an interpolation to get an approximate value corresponding to a particular input. Many researchers have explored this area. Perhaps the work most relevant to humanoid walking would be motion graphs, which have been successfully used to synthesize new behaviors for animated characters based on existing data [17], [18]. However, while the data for building motion graphs is usually obtained by capturing human motions, which is expected to be physically correct at least for the human subjects (after spending a large amount of manual work for data correction), it is difficult to obtain the motion data for a humanoid robot with the same level of quality. We obtain walking motions of a humanoid model using trajectory optimization and this is done automatically and in parallel to achieve enough data for training. The numerical optimization process does not always work well and this may result in low quality motions in some input regions. As local interpolation is sensitive to a local defect in the data, data-driven methods may result in a wrong answer to an input query in that region. A short-term treatment of this problem would be to remove low quality data and prevent choosing the related actions in planning, but this would limit the search space of the planner.

VI. Conclusion

In this paper we showed that many features of optimized humanoid walking can be encoded as global quadratic functions of initial state and a footstep sequence. Our quadratic functions showed remarkable performance in estimating features of optimized walking such as the state of the optimal walking motion in time and the torque expenditure during walking. In order to obtain enough motion data for training the encoding function, we sampled initial states and footstep sequences, and solved the trajectory optimization problem for each sample in parallel on a cluster supercomputer. As an application, we used the encoding function to calculate the walking cost used in finding an optimal footstep sequence with an A* algorithm. Our approach provides a fast alternative to considering walking dynamics of a full body model in online locomotion planning problems.

It would be interesting to investigate the performance of encoding the entire walking motion, i.e., not only the walking cost and the center of mass trajectory, but also the trajectories of all joints or bodies. Other interesting future work is to test the universality of the encoding technique by applying it to different styles of optimal walking.

VII. Acknowledgments

This material is based upon work supported in part by the US National Science Foundation (ECCS-0824077, and IIS-0964581) and the DARPA M3 and Robotics Challenge programs. The authors would like to thank Prof. G. Gibson for allowing them to use a cluster supercomputer for building the walking motion database.

References

[1] T. McGeer, "Passive dynamic walking," Intl. J. of Robotics Research, vol. 9, no. 2, pp. 6282, Apr. 1990.
[2] S. Collins and A. Ruina, "A bipedal walking robot with efficient and human-like gait," ICRA 2005.
[3] A. D. Ames, "First Steps Toward Automatically Generating Bipedal Robotic Walking from Human Data," In 8th International Workshop on Robotic Motion and Control, RoMoCo'11, 2012.
[4] M. W. Hardt, "Multibody Dynamical Algorithms, Numerical Optimal Control, with Detailed Studies in the Control of Jet Engine Compressors and Biped Walking," Ph.D. Thesis, U.C. San Diego, 1999.
[5] A. C. Fang and N. S. Pollard, "Efficient synthesis of physically valid human motion," SIGGRAPH 2003.
[6] J. Kim, J. Baek and F. C. Park, "Newton-type algorithms for robot motion optimization," IROS 1999.
[7] K. Yin, S. Coros, P. Beaudoin and M. van de Panne, "Continuation methods for adapting simulated skills," SIGGRAPH 2008.
[8] A. Witkin and M. Kass, "Spacetime constraints," SIGGRAPH 1988.
[9] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," SIAM J. Optim., vol. 12, pp,. 979-1006, 2002.
[10] A. Seeger, *Recent Advances in Optimization,* Springer 2006.
[11] W. Huang, J. Kim and C. G. Atkeson, "Energy-based optimal step planning for humanoids," ICRA 2013.
[12] A. Bagchi and P. K. Srimani, "Weighted heuristic search in networks," Journal of Algorithms, 6:550-576, 1985.
[13] B. Bonet and H. Geffner, "Planning as heuristic search. Artificial Intelligence," 129(l-2):5-33, 2001.
[14] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving,* Addison-Wesley, 1984.
[15] J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots," IROS 2001.
[16] J. Chestnutt, M. Lau, G. Cheung, J.J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," ICRA 2005.
[17] L. Kovar, M. Gleicher and F. Pighin, "Motion graphs," SIGGRAPH 2002.
[18] A. Safonova and J. K. Hodgins, "Construction and optimal search of interpolated motion graphs," SIGGRAPH 2007.