2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Off-line controller design for reliable walking of Ranger

Matthew Kelly[1], Matthew Sheen[2], and Andy Ruina[3]

***Abstract*— We present a method for designing a walking controller for the walking robot Cornell Ranger. Our goal is a controller that can be designed using model-based optimization, and then transferred directly to the robot without the need for after-the-fact hand-tuning. The structure of the controller is hierarchical, with a high-level balance controller that plans step-to-step motions, and a lower-level joint controller that coordinates the individual joint motors to achieve the desired limb motions. The balance controller is designed through optimization, with the explicit goals of *a)* achieving a desired walking speed while *b)* minimizing energy use and *c)* avoiding falls due to disturbances. We demonstrate this walking controller on the Cornell Ranger, and find that the resulting gait is comparable to that of a previous (hand-tuned) controller, with regard to energy use, speed regulation, and fall prevention.**

I. INTRODUCTION

Here we present a new control architecture for the Cornell Ranger, a bipedal walking robot shown in Figure 1. While previous controllers for this robot required extensive hand tuning, the controller presented here is designed using off-line optimization and is meant to transfer directly to the robot without modification.

To advance the science of robot control, we desire algorithms that do not have hand-tuning on the robot as a key final step. Queries of robot builders reveal that such hand tuning of hardware is all too common. To avoid controller tuning on mechanical hardware, we turn to off-line model-based optimization for controller design. This process requires an accurate simulation for the robot, as well as a mechanism for making the controller robust to simulation and modeling errors.

Walking is complicated, so the controllers for most walking robots rely on hierarchical control architectures [1]–[10]. These simplify the design process in part by reducing the number of free parameters that describe the controller. The control architecture here incorporates some ideas from the previous controller for the Cornell Ranger [8], SimBiCon [1], and hybrid zero dynamics [6]. The high-level gait control is based on a finite state machine which regulates speed and maintains balance, while the lower-level joint control is simply a proportional-derivative tracking controller on each joint.

It is impractical to automatically design every feature for a walking controller — the state and control space is simply too large for modern techniques to compute a true optimal policy with useful resolution. Instead, we break down the problem. The control architecture *is* manually designed based on our own experience and discussions with experts, as well as the literature. The gains in the low-level joint controllers are experimentally determined, using standard methods [11]. That leaves the parameters of the balance controller, 15 numbers in our implementation, to be automatically selected using optimization.

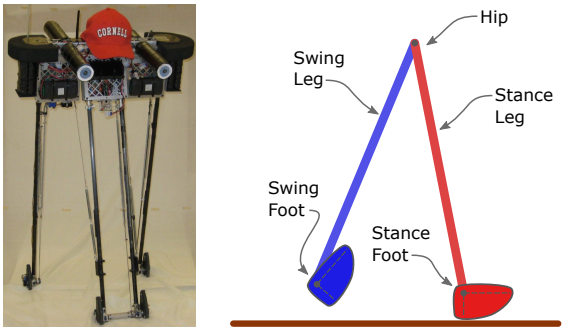

Fig. 1. **Cornell Ranger walking robot:** A photo and a diagram of Cornell Ranger, our experimental test platform for the controller.

II. CORNELL RANGER

Our test robot is the Cornell Ranger, which is described in detail in [8], [12]. Here we will present only a short overview.

Ranger, shown in Figure 1, is at the bottom of the bipedal robot food chain. It was designed only for low-energy walking over flat terrain. It has four legs that are arranged into an inner and outer pair. This arrangement means that the walking control only needs to stabilize front-to-back motions: lateral motions are passively stabilized. The robot is under-actuated by one degree of freedom: there is no motor that can directly control the angle of the stance leg.

A. Hardware

Although simulation is useful for controller design and basic testing, the only way to know if a controller really works is to test it on a real robot. For this purpose we use the Cornell Ranger [8], [12].

Ranger lacks of knees, which forces all changes in effective leg length to come from rotations of the feet. The feet on Ranger are curved with small radius, which means that Ranger cannot statically balance with its feet together. Additionally, the circular curve of ranger's feet is truncated close to the heel. This truncated shape allows for the feet to rapidly clear the ground at the start of swing, but also further limits the effective control authority of the ankle motors.

*This work is supported by the National Robotics Initiative (grant number 1317981)

[1]Matthew Kelly, PhD candidate, Mechanical Engineering, Cornell University, Ithaca NY 14850, USA mpk72@cornell.edu

[2]Matthew Sheen, PhD candidate, Mechanical Engineering, Cornell University, Ithaca NY 14850, USA mws262@cornell.edu

[3]Andy Ruina, Professor, Mechanical Engineering, Cornell University, Ithaca NY 14850, USA ruina@cornell.edu

Ranger was designed for energy effectiveness, which we measure using the total cost of transport (CoT). As described in [13] and [14], CoT is the ratio of total energy consumed to the weight multiplied by distance traveled. The total energy is measured at the batteries, and thus includes the power consumption by the motors, sensors, on-board computers, and communication. The best experimental controller on Ranger had a CoT of 0.19, but Ranger's (more reliable) marathon controller [8] had a CoT of 0.28.

Ranger has a variety of sensors. Foot sensors measure the force between the heel of the foot and the ankle joint, which we then threshold to determine if a foot is in contact with the ground or not. Each joint has absolute angle encoders for both the motor and the end-effector. Finally, there is an IMU (gyro and accelerometers) located on the outer legs that we use for estimating the absolute orientation and angular rate of the outer legs.

B. Model

Our model for the Cornell Ranger is largely based on our previous work [8], [12], [15]. We assume that the robot is a planar biped, with four rigid bodies (outer legs, inner legs, outer feet, inner feet) which are connected by three motors (outer ankles, hip, and inner ankles). We also use a full bench-tested electro-mechanical model of the motors and gear boxes.

There are two notable differences between the model used here and our previous model of Ranger. The first is that here we simplify by assuming that the drive cables connecting the ankle motors to the feet are rigid, where as the previous work [8] treated them as stiff springs. This was done in part because the time-stepping simulation had did not perform well using the stiff spring cable model. Second, we model the shape of the foot as a quintic spline (periodic, with 6 segments), rather than as a complete circle, as illustrated in Figure 2. This allows use to simulate interactions between the heel and the ground.

C. Simulation

The previous simulator for Ranger [8] was designed to study open-loop trajectories with a prescribed sequence of contact configurations. For the research presented in this paper, we need to study the close-loop behavior of the robot for a variety of controllers, some of which will cause the robot to stumble and fall down — a behavior that was not able to be simulated by previous simulations,.

To capture more complex contact sequences, we developed a "time-stepping" simulator for Ranger, which runs a contact-solver on each time-step. This simulation allows us to model the robot walking over any ground profile, using accurate collision shapes for the robot's feet. The simulation is implemented in Matlab, and then compiled to MEX for speed.

D. Control Considerations

While walking, the motors of the robot must add energy to the system to compensate for frictional and collisional losses. Due to the small curved feet, Ranger cannot inject much energy by ankle torques through the step, except by push-off with the back foot at the end of each step. This extension is small (a few centimeters), but enough to propel the robot forward and to adjust walking speed. To get the maximum effect of this push-off it must be timed carefully to occur just before the collision on the front foot [16]–[18].

Ranger does not have knees, and thus as soon as the push-off is complete, the foot needs to rotate up and out of the way so that it doesn't scuff as the swing leg moves forward. Then the foot needs to rotate back down just before heel-strike. Too early and the foot scuffs; too late and the foot strikes down on the back of the heel, which causes a trip, with the foot rotating back up. In each of these cases, the robot falls. See Figure 2 for details regarding foot orientation for push-off and foot-flip.

III. CONTROLLER ARCHITECTURE DETAILS

The control architecture here is divided into four levels, arranged highest-to-lowest:

- The *Balance Controller* runs once per step, at mid-stance, setting the five parameters that describe the gait controller, to achieve balance.
- The *Gait Controller* is a finite-state machine, shown in Figure 3, which sets the target angles and rates for the joint controllers.
- The *Joint Controllers* are proportional-derivative controllers which compute the command torque for each joint.
- The *Motor Controllers* are proportional-integral controllers which compute a low-level PWM commands to achieve a commanded torque in each joint.

We assume that both the robot and controller are left-right symmetric. We will describe the controller for the case when the outer feet are on the ground. At the conclusion of the step, the whole controller is mirrored, with the inner legs becoming the new stance legs.

A. Motor Control

The motor controllers are at the bottom level of the controller. They run a simple proportional-integral control loop at 2 kHz on each of the three joint motors (outer ankle,

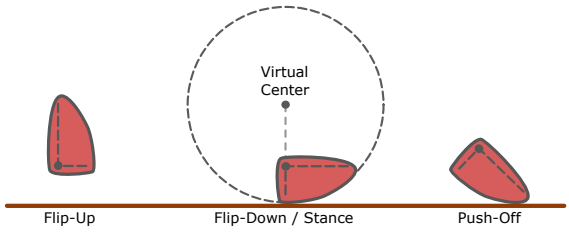

Fig. 2. **Ranger Foot Diagram:** Ranger's feet are small, and their soles are sections of circular arcs. Here we show the three target configurations used by the controller. Flip-up is used for the swing foot, allowing the foot to clear the ground, since the robot has no knees. The flip-down/stance configuration is used by the stance foot for most of the step, providing a steady base for the robot. The final configuration, push-off is used to rapidly extend the foot, propelling the robot forward for the next step.

inner ankle, and hip), tracking a desired joint torque. These motor controllers are coded at a low-level in the robot, and we did not change these.

B. Joint Control

While the robot is walking, the joint controllers (outer ankle, inner ankle, and hip) are continuously running simple proportional-derivative controllers at 500 Hz. Each controller computes a command torque u, which is sent to corresponding motor controller. The reference angle q^*, rate $\dot{q}^*$, and torque u^* are all sent from the gait controller. The measured joint angle and rate are given by q and $\dot{q}$.

$$u = u^* + K_P\left(q^* - q\right) + K_D\left(\dot{q}^* - \dot{q}\right) \tag{1}$$

C. Gait Control

The gait controller coordinates the motion of the three joints on the robot, sending reference commands to the joint controllers at 500 Hz. There are two parts to the gait controller. The first is a finite-state-machine (FSM), which is shown in Figure 3. During a single walking step, this FSM can be in either the *glide* mode, or the *push* mode. In glide mode, the robot is smoothly moving forward, with the gait controller pulling the swing leg through the step. In push mode, the robot extends the rear foot, propelling the robot forward, while simultaneously rotating the swing foot down in preparation for heel-strike.

Glide Mode: The reference commands sent to the swing foot joint controller are simple: the angle is constant, selected such that the foot will not scuff the ground, and the rate and torque references are zero.The reference angle and rate for the stance foot are selected to keep the absolute orientation of the foot constant, while the robot rotates over it. The hip joint in glide is more complicated: the joint angle and rate

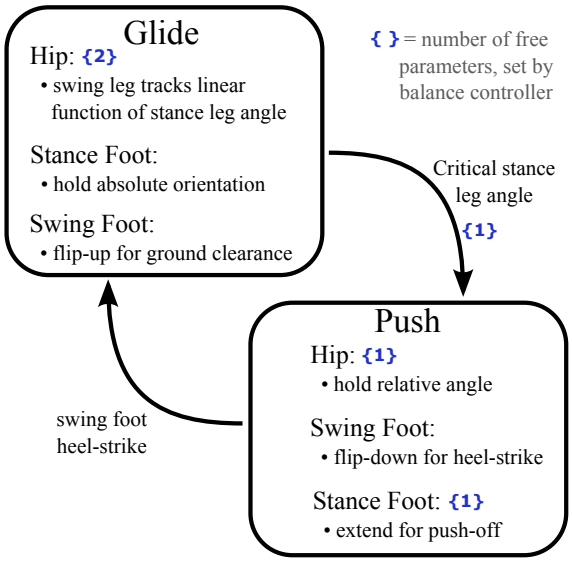

Fig. 3. **Gait Controller:** A finite-state-machine that sets the targets for the joint controllers. It has two states: Glide (or swing) and Push (or step-to-step transition). There are a total of $\{5\}$ parameters, which are set once per step by the balance controller. At each heel-strike transition, the old swing leg becomes new stance leg, and vice versa.

are set based on a linear function of the stance leg angle, and the reference torque is computed to compensate for the torques on the joint due to gravity and the hip spring (which connects the two legs). This phase-based tracking is inspired by [6].

Push Mode: Both ankle joints are set to maintain a constant absolute orientation of the foot, with the stance foot pushing-off the ground and the swing foot flipping-down for heel-strike, as shown in Figure 2. During push, the hip joint holds a constant angle, with the help of feed-forward torque compensation.

Transitions: There are two state transitions in the finite-state-machine. The transition into glide mode is triggered when the contact sensors on the swing foot detect heel-strike. The transition into push mode is triggered as the stance leg passes a critical absolute angle.

Parameters: The gait controller has five free parameters that are set by the balance controller. These parameters are illustrated in Figure 3 and are: 1,2) constant and linear coefficient for the hip joint reference trajectory in glide mode; 3) constant hip angle reference during push mode; 4) critical angle of the stance leg for transition from glide mode to push mode; and 5) absolute angle reference for the stance foot during push mode.

D. Balance Control

The top-level of the control architecture is the balance control, which runs once per step at mid-stance. It changes the five parameters of the gait controller to regulate balance and walking speed. For example, if the robot is walking too slowly, it will increase the reference angle for the push off, adding more energy to the system.

There is a single input the the balance controller: the robot's speed at mid-stance. Thus, the balance controller is simply a function that maps the mid-stance speed of the robot at mid-stance to the set of five parameters that are passed to the gait controller. Here, we implement this function using a look-up table, storing the five parameter values for zero speed, the target speed, and the maximum expected speed. For intermediate speeds we use linear interpolation.

The look-up table for the balance controller has a total of 15 entries (5 parameters at each of 3 speeds), which we compute using off-line optimization, using methods discussed in §IV.

IV. CONTROLLER DESIGN

We claim that the controller is designed "using optimization", but we also acknowledge that there are many decisions that are made by humans as well.

We designed the architecture for the walking controller (§III) through insight and intuition, which we acquired through experiments, discussions at technical conferences, and the literature [1], [6], [8].

There are several constant parameters in the controller, which are also manually set. These include the orientation of the stance foot during glide mode (selected such that the ankle joint lies directly below the virtual center of the foot),

and the relative angle of the ankle joint required during flip up (selected to be close to the joint limit).

The joint controllers (§III-B) in the ankles and hip all have proportional and derivative gains, which are selected by simple experiments on the hardware. These experiments are easily repeatable by any controls engineer using standard methods [11].

The final set of parameters (§III-C, §III-D), are selected entirely by computer optimization (§IV-A, §IV-B). These parameters are copied directly from the output of the optimization to the robot. There is no final hand-tuning step.

A. Objective Function for Optimization

The balance controller (§III-D) is parameterized by 15 numbers, the 3×5 look-up table entries. These are computed by off-line optimization. The objective function in this optimization is chosen to reject disturbances, while minimizing speed and cost of transport.

The objective function evaluates a candidate controller by running several simulations. Each of these simulations starts from the same launch configuration, shown in Figure 4, and then the robot walks over several different ground profiles. The first ground profile is flat and level ground. The remaining ground profiles serve as disturbance tests and are either constant slopes (uphill or downhill) or rolling hills (sine curves).

A candidate controller receives a reward for each successful step that it takes, where the reward is related to both the speed and energy used to complete the step. The reward is maximized by a controller that walks at the desired speed using little energy. Each trial (ground profile) has a fixed number of steps. If the robot falls during a trial, then it receives a reward of zero for that and all future steps in the trial.

The optimization then finds the controller that maximizes the sum of rewards over all trials. The structure of the objective function is such that fall avoidance is more important than speed regulation or energy effectiveness.

B. Optimization Method

Here we used a Covariance Matrix Adaptation Evolutionary Strategy (CMAES) [19], [20] to optimize the balance controller, because it deals well with our non-smooth objective function. We initialize the algorithm by first estimating bounds on the parameters. For example, the push-off target angle must be within the actuator limits, and the hip trajectory coefficients should be roughly consistent with bipedal walking (the swing leg must travel from back to front, etc.).

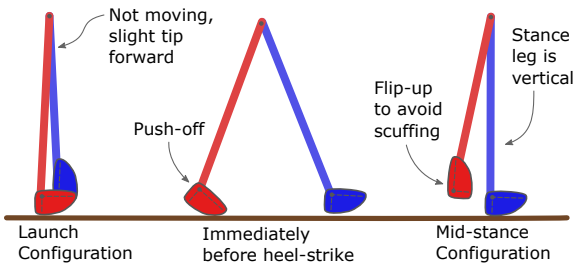

Fig. 4. **Ranger Configurations** The launch configuration (left) shows the static configuration that we use to start Ranger walking, both in simulation and in reality. The middle configuration shows the robot immediately before heel-strike. The final configuration (left) is mid-stance, with the supporting leg vertical. This is the configuration that triggers an update from the balance controller.

V. RESULTS

In addition to the results in this section, we have included a short video showing Ranger walking using the new controller presented here.

A. Off-line Controller Design (Optimization)

We implemented the entire design process for the balance controller in Matlab, with most of the simulation code being compiled to MEX for faster run-time. The code takes about 10 minutes to compute the optimal control parameters running on a laptop (Intel Quad-Core i7 CPU Q720, 1.60 GHz), where each walking step of the robot takes about 0.034 seconds to compute, for a total of about 18,000 steps per optimization.

Here we designed a controller to achieve a single walking speed, although we could repeat the process to compute walking gaits for a whole set of target speeds.

B. Walking: Simulation vs. Experiment

Our controller design process relies on accurate simulation of the robot; in this section we compare experimental data collected during walking to what we expected based on the simulation.

We collected data on the robot over two trials. In each case the robot walked about 80 meters over a stone-tiled floor. The surface of each tile was flat, but there was a change in height of roughly 2 millimeters between the edge of any two tiles. Over a large scale there is no measurable slope to the floor.

The model and simulation match well on power consumption: the simulation predicts 22.2 Watts, and the two trials used 22.1 Watts and 22.9 Watts respectively. The simulation does not do as good of a job at predicting speed, with the simulation walking at 0.66 meters per second, and real robot walking at about 0.58 meters per second. This difference in speed makes the cost of transport a bit higher on the real robot: 0.49 instead of 0.42 in simulation. These results are summarized in Table I.

We also compared the angles of the robot's legs and feet, as functions of time, to those predicted by the simulation. We selected six consecutive walking steps at random, during steady state walking, for both the simulation and the experimental data. We found that the leg angles are a close match

TABLE I
COMPARISON OF SIMULATION AND EXPERIMENTAL DATA.

	Simulation	Trial 1	Trial 2
Duration	119 s	162 s	159 s
Distance	79 m	91 m	91 m
Average Power	22.2 W	22.1 W	22.9 W
Total CoT	0.42	0.48	0.49
Average Speed	0.66 m/s	0.57 m/s	0.58 m/s

throughout the gait. The ankle angles are close for much of the step, but there are significant deviations during push-off, as shown in Figure 5.

C. Robustness Experiments

We evaluated the robustness of the "new" controller by comparing its performance to the previous "old" controller, which was used for Ranger's marathon walk in 2011 [8]. Each controller was first evaluated walking without disturbances to establish a base-line. Then we subjected the robot to disturbances and measured each controller's ability to regulate walking speed and prevent falls. The disturbances are illustrated in Figure 6, and the results are given in Figure 7.

Baseline: We established a baseline performance for each controller by having them walk on a flat stone-tiled floor. The only disturbances were the slight perturbations caused by the height variations ($\approx$ 2 mm) between stone tiles. The old controller walked 183 meters with no sign of falling, while the new controller walked 732 meters, falling 6 times.

Trial 3: Our first disturbance was walking on a more challenging floor, which had sagged over the decades. The resulting ground profile was smooth, but with slight rolling hills: the peaks were 4 meters apart, and the peak-to-trough height was about 2 centimeters. The old controller walked 429 meters, falling 3 times, while the new controller walked 501 meters, falling 6 times.

Trial 4a: Next, we brought the robot back to the original stone-tiled floor and removed the "hip spring" which connects the inner and outer legs, inducing a substantial modeling error. Both controllers walked a distance of 183 meters. The old controller did not fall and the new controller fell 6 times.

Trial 4b: We returned the hip spring to the robot, and then added a small mass (0.35 kg) to the outer legs. The weight was positioned 0.12 meters in front of the hip joint, as shown in Figure 6.The old controller was unstable: it continually sped up until is stumbled and fell, a total of 19 times in 183 meters. The new controller walked well with the added weight on the front. It increased it's speed and only fell 3 times in 183 meters.

Trial 4c: For our final disturbance, we moved the mass (0.35 kg) to the back of the robot, still on the outer legs, as shown in Figure 6. In this case the old controller performed well, only falling once in 183 meters. The new controller did not fare as well, falling 14 times in 183 meters.

VI. DISCUSSION

In this paper we presented a new controller for the Cornell Ranger, a bipedal walking robot. The controller architecture is based on variety of ideas taken from previous walking controllers, including [1], [6], [8]. Although the low-level joint controllers are all hand-tuned proportional-derivative

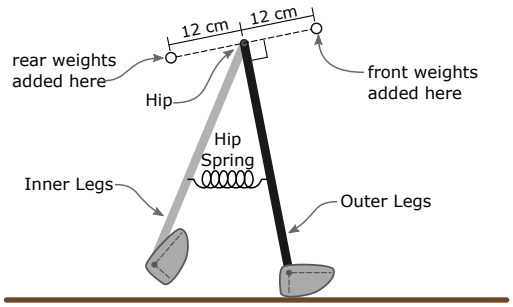

Fig. 6. **Disturbance Diagram** In Trial 4a, we removed the hip spring (k = 7.6 Nm/rad), which coupled the angles of the inner and outer legs. In Trial 4b we added a mass of 0.35 kg in front of the outer legs, in the location shown. In trial 4c, we added the same 0.35 kg mass behind the outer legs.

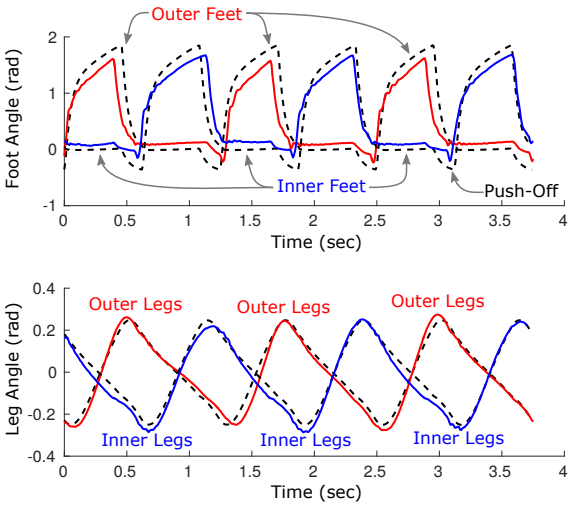

Fig. 5. **Walking: Simulation vs. Experiment** Six steps of periodic walking. The dashed lines show the simulation and the solid lines show experimental data. All angles are absolute, and measured from vertical. Most angles match well, but there is a noticeable difference between simulation and experiment during push-off.

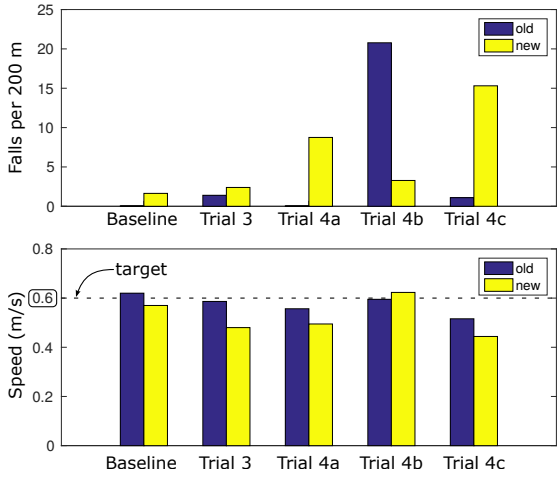

Fig. 7. **Robustness Test** Each of the trials compares the old (2011 Marathon) controller with the new controller presented in this paper. The baseline trial was conducted on a flat stone-tiled floor. Trial 3 was walking on smooth ground with rolling hills: 4 meter peak to peak, 0.02 meters peak to trough. Trials 4a-4c were all conducted on the flat stone-tiled floor to test robustness to modeling errors: missing hip spring (4a), weights added to front (4b), and weights added to back (4c).

tracking controllers, the parameters of the high level balance controller are designed entirely using off-line optimization. In the results section of this paper, we seek to answer two questions, detailed below.

1) ***How well does the simulation match experimental results?*** We found that the simulation is a good match for the experimental data on power use and leg angles. The ankle angles during push-off do not match well. This discrepancy is likely due, in part, to our assumption that the drive cables for the ankle joint are inextensible. As a result, the effect of push-off on the real robot is reduced, which might explain the reduced walking speed seen in the experiments.

2) ***How does this new controller compare to the previous (hand-tuned) controller?*** In general, the previous (hand-tuned) controller out-performs the new controller, although not by a huge margin. This assessment is based on the fact that it does a better job of regulating speed in the presence of disturbances, and has a lower fall rate in all but one trial.

Summary: We were able to design a controller using off-line optimization, without after-the-fact hand-tuning, which walked reasonably well under a variety of disturbances. Despite this, the previous hand-tuned controller out-performed the new controller on most tests. Although the new controller was designed in large part by computer optimization, the architecture of both the controller and optimization were manually selected. It is likely that these manual choices limited the ultimate performance of the new controller.

VII. FUTURE WORK

The work presented here is preliminary: it shows that we can set up a model-based optimization that can design a controller in simulation that we can directly use on a real robot. Although the initial results are passable, there is still much left to do.

In the current implementation, the balance controller only updates the trajectories of the robot once per step. In the future we would like to make these updates continuous, allowing the robot to more quickly react to external disturbances such as a sudden push.

An additional area for improvement is the look-up table for the balance controller, which now uses only a single input: the mid-stance speed of the robot. Ideally, this policy would use at least the state of the stance and swing legs, rather than this simple one-dimensional projection.

Our model of Ranger is reasonably accurate, but there are still some discrepancies between model and reality, especially during push-off. In the future we plan on moving the last few iterations of the optimization process to the robot, using experimental data rather the simulation for automatic fine-tuning of the controller.

Finally, we plan on designing the controller to be capable of walking at several speeds, rather than just the one speed that it is capable of now.

ACKNOWLEDGMENT

This research is supported by the National Robotics Initiative (grant number: 1317981).

REFERENCES

[1] K. Yin, K. Loken, and M. van de Panne, "SIMBICON : Simple Biped Locomotion Control," in *SIGGRAPH*, 2007.

[2] X. B. NPeng, G. Berseth, and M. van de Panne, "Dynamic Terrain Traversal Skills Using Reinforcement Learning," in *SIGGRAPH*, 2015. [Online]. Available: http://www.cs.ubc.ca/ van/papers/2015-TOG-terrainRL/index.html

[3] J. Englsberger, C. Ott, M. A. Roa, A. Albu-sch, and G. Hirzinger, "Bipedal walking control based on Capture Point dynamics," in *International Conference on Intelligent Robots and Systems*, San Francisco, 2011, pp. 4420–4427.

[4] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pp. 200–207, dec 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4115602

[5] T. Koolen, T. de Boer, J. Rebula, a. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, jul 2012. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364912452673

[6] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 42–56, 2003.

[7] M. VUKOBRATOVIĆ and B. BOROVAC, "Zero-Moment Point - Thirty five years of its life," *International Journal of Humanoid Robotics*, 2004.

[8] P. a. Bhounsule, J. Cortell, a. Grewal, B. Hendriksen, J. G. D. Karssen, C. Paul, and a. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1305–1321, jun 2014. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364914527485

[9] P. R. D. Honda, "Asimo Technical Report, Tech. Rep. September, 2007.

[10] S. Kuindersma, F. Permenter, and R. Tedrake, "An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion," in *International Conference on Robotics and Automation*, 2014.

[11] K. Ogata, *Modern Control Engineering*, 5th ed. Prentice Hall, 2010.

[12] P. a. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. G. D. Karssen, C. Paul, and A. Ruina, "MULTIMEDIA EXTENSION # 1 International Journal of Robotics Research Low-bandwidth reflex-based control for lower power walking : 65 km on a single battery charge," *International Journal of Robotics Research*, 2014.

[13] V. a. Tucker, "Energetic cost of locomotion in animals." *Comparative biochemistry and physiology*, vol. 34, no. 4, pp. 841–846, 1970.

[14] ——, "The energetic cost of moving about." *American scientist*, vol. 63, no. 4, pp. 413–9, 1975. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/1137237

[15] P. a. Bhounsule, "A controller design framework for bipedal robots: trajectory optimization and event-based stabilization," Ph.D. dissertation, Cornell Univerisy, 2012. [Online]. Available: http://ruina.tam.cornell.edu/ pab47/Pranav_Bhounsule_Thesis.pdf

[16] A. D. Kuo, "Energetics of Actively Powered Locomotion Using the Simplest Walking Model," *Journal of Biomechanical Engineering*, vol. 124, no. 1, p. 113, 2002.

[17] M. Srinivasan and A. Ruina, "Computer optimization of a minimal biped model discovers walking and running." *Nature*, vol. 439, no. 7072, pp. 72–5, jan 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16155564

[18] S. J. Hasaneini, C. J. B. Macnab, J. E. a. Bertram, and H. Leung, "Optimal relative timing of stance push-off and swing leg retraction," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3616–3623, nov 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6696872

[19] N. Hansen and a. Ostermeier, "Completely derandomized self-adaptation in evolution strategies." *Evolutionary computation*, vol. 9, no. 2, pp. 159–95, jan 2001. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/11382355

[20] N. Hansen, "An analysis of mutative sigma-self-adaptation on linear fitness functions." *Evolutionary computation*, vol. 14, no. 3, pp. 255–75, jan 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16903793