Simultaneous learning of a robot's gait and its body dimensions and compliances

by Alan Asbeck and Nick Choo

Overview

The goal of this project was to have a simulation simultaneously learn a robot's gait as well as body dimensions and compliances—in essence, create a design through machine learning. Can we generate an optimal gait and body design instead of spending much time hand-tuning these parameters?

As a practical motivation, the robot used in the simulation was loosely modeled after an actual robot, iSprawl, created by Sangbae Kim in Prof. Mark Cutkosky's lab in the Mechanical Engineering department here at Stanford, though some details of the simulation differed from the actual robot. Figure 1 shows pictures of the simulated and iSprawl robots. Originally, we simulated a quadruped robot with three degrees of freedom per leg, including a knee joint, for a total of 62 parameters; however, this was much too complicated for the simulation to learn good parameters effectively, so we switched to this simpler robot.

The simulation was done using the Arachi Dynamics Engine (Arachi, Inc.) and RiSE modeling environment used by the Robotics in Scansorial Environments DARPA project.

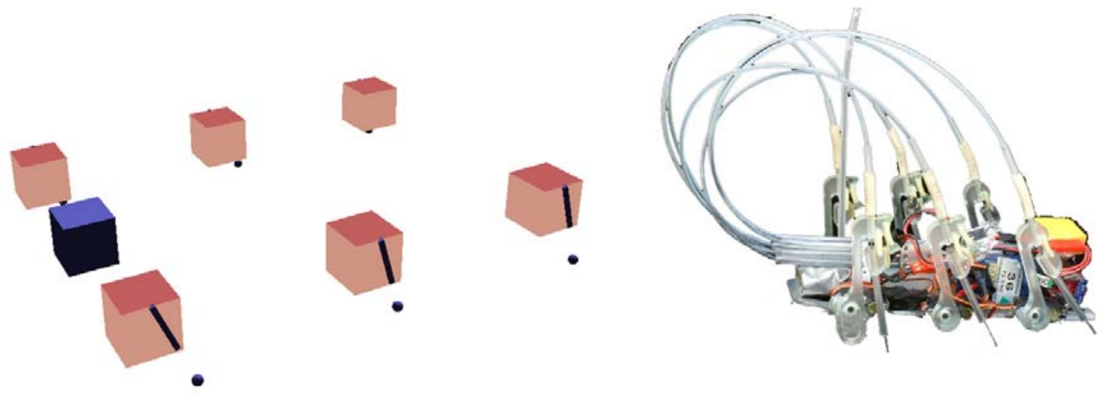


Figure 1: Pictures of the robot used in the simulation (on the left), and iSprawl (on the right). The blue block on the left of the simulated robot denotes the front and was massless.

Simulation

The simulated robot consisted of an actuated piston joint and a passive hip joint in each leg (see figure 2). It moves using an alternating tripod gait: the pistons in each tripod were driven in a fixed sinusoidal pattern, with the two tripods 180˚ degrees out of phase. When each piston extends, the leg rotates backwards about the hip joint, so the robot is driven upwards and forwards. Then, when the piston lifts the leg off the ground, the passive spring in the hip returns the leg to its initial position.

In the simulation, the robot was constructed to be laterally symmetric, as iSprawl is in real life; this enabled the robot to go nearly straight for long distances. The robot had 17 parameters that could be adjusted: the bias angle, position gain k_p, and velocity gain k_v for each of the three pairs of hip joints; the position and velocity gains for each of the pistons, and the body length and width. These are denoted in figure 2. The k_p, k_v terms control the spring constant and damping for the hip angle and piston suspension:

$$T = k_p*(x_{desired} - x) + k_v*(v_{desired}- v)$$

where T is the torque applied to the joint, x is the position of the joint, and v is the velocity of the joint.

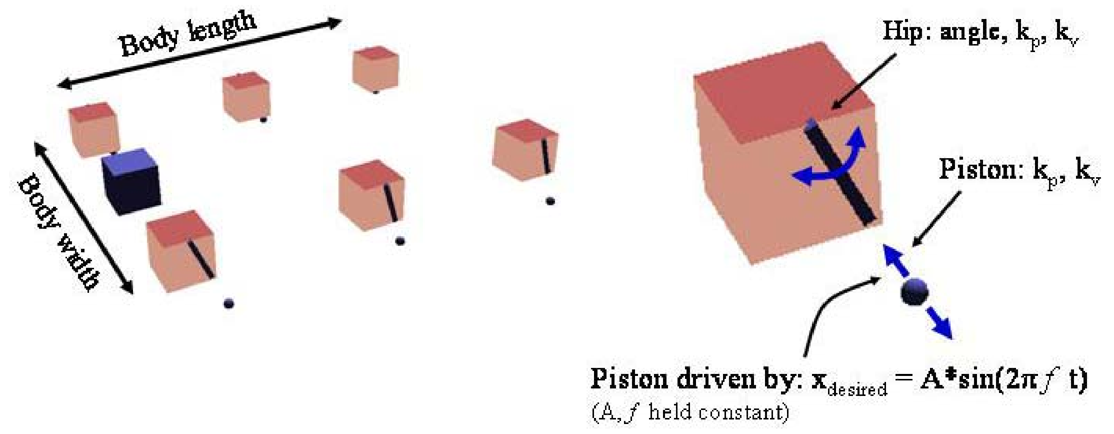


Figure 2: Parameters varied in the simulation.

The robot's gait is controlled by these parameters because the mechanics of the actual gait are tightly coupled to them. In the case of iSprawl, the legs move very quickly and are all angled so as to push backwards, leading the robot to move in a smooth motion. In these simulation results (discussed below), the robot learned to move in more of a bounding motion, due to its low piston frequency; in one gait, the rear and middle hip joints were angled so as to push the robot backwards, functioning quite differently than the legs of iSprawl.

Algorithms

Our learning algorithms are based on the idea that distance traveled is an (unknown) function $f(\theta)$ of a parameter vector θ plus some stochastic noise. Our "reward" function was chosen to be the distance traveled in the forward direction (along the y-axis), minus the lateral deviation from the straight-ahead path (the magnitude of the x-coordinate), and we try to maximize this reward by finding the maxima of $f(\theta)$ over our parameter space. In both our methods of looking for the optimal gait, we essentially iterate over the following operation on θ.

$$\theta^{i+1} = \arg\max_{\theta=\theta^i,\theta^i+\Delta}[f(\theta\ )] \quad \text{subject to} \quad \theta_{\min} \le \theta \le \theta_{\max}$$

The first algorithm is a hill-climbing algorithm, in which we try to have our parameter take a "step" in a random direction, and evaluate its performance at that point.
The coordinate-ascent algorithm instead considers each parameter, taking "steps" until the reward function achieves close to a maximum in the particular parameter, and iterating through the parameters until convergence. We also took smaller and smaller "steps" when the parameters were close to a (possibly local) optimum.
Each algorithm was run several times for random initial parameters θ until convergence within a specified tolerance, although several runs were stopped before convergence due to time constraints.

Gradient (Hill-Climbing) Method

Begin with a random parameter vector
Record the reward
Repeat:

- Add a random Δ to θ , record the reward.
- If the reward of a step taken in the new direction Δ is less than the old reward, take a step in the other direction.
- Update θ only if reward increases. If *maxIter* iterations have passed without improvement, decrease step size; report as convergent solution if step size < *minSize*.

Coordinate-Ascent Method

Begin with a random parameter vector
Record the reward
Repeat:

- Add a fixed step length Δ_j to θ_j , record the reward.
- If the reward of a step taken in the new direction Δ_j is less than the old reward, take a step in the other direction.
- Continue "stepping" until average reward does not improve.
- If all parameters have taken one "forward" and one "backward" step without improvement, decrease step size, report as convergent solution if step size < *minSize.*

Issues

"Noisy" Reward

The first issue that we encountered in the simulation phase of the project was that there was a high degree of noise in the reward obtained per trial. Examining the variance of the reward over different reward magnitudes suggested that the noise level is highly correlated with magnitude of reward. This makes it difficult to determine whether a "step" actually improved the reward especially in the latter part of runs.

To counter the effects of the noisy reward, we ran each trial between 2 to 5, usually 3 times, and averaged the rewards. We also specified a tolerance based on the magnitude of the reward, and if a reward gain was within this tolerance we would still "step" in this direction, but consider it a "no-improvement" step.

Finally, we averaged the rewards over longer simulation trials, and let the robot "run" longer (10 seconds) to reduce variances from initialization (robot legs are moving freely before we "drop" the robot to the ground for each trial).

Computational Bottlenecks

The main problem that we faced in running the simulations was that simulations the simulations took very long to run. In general, 10 seconds of simulation time took 1:10 minutes of real time to run, even with graphics turned off. With averaging, taking only 80 "steps" takes almost 5 hours.

Results

Simulations greatly increased the reward over a naïve implementation with random parameters—they were able to find a good parameter set even if the initial gait was moving in the wrong direction. A plot of the rewards versus time for several runs is shown in figure 3. Both the hill-climbing and coordinate ascent approaches worked well.

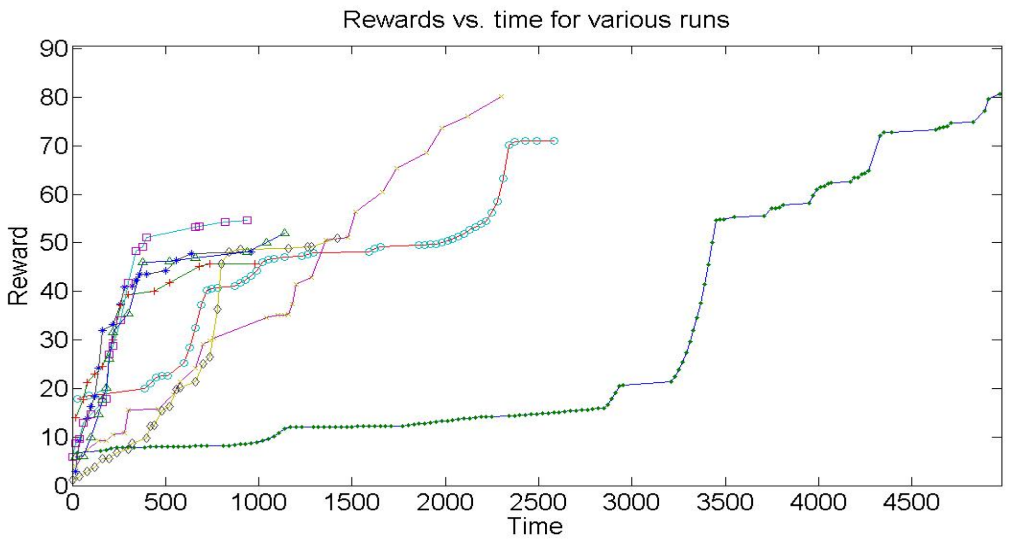


Figure 3: Rewards achieved by the robot as a function of time. This includes two simulations that had not yet converged.

The final joint compliances and gaits found by the simulation varied, indicating that our simulation frequently got stuck in local optima. However, several of the gaits converged to nearly the same final reward, suggesting that this may be close to a global maximum.

Furthermore, some joint compliances converged to very similar results across different simulations (see figure 4). The piston compliances for the front hip converged to very similar results, a very strong spring and low damping. This combination of parameters enables the leg to push against the ground as strongly as possible, providing the greatest upwards/forwards motion. Several other joint compliances also converged to reasonably similar results across simulations, particularly the middle hip compliances and rear piston compliances. It is interesting to note that the simulation that performed the best (the '•' in the plot) actually had parameters that deviated significantly from the others in many cases.

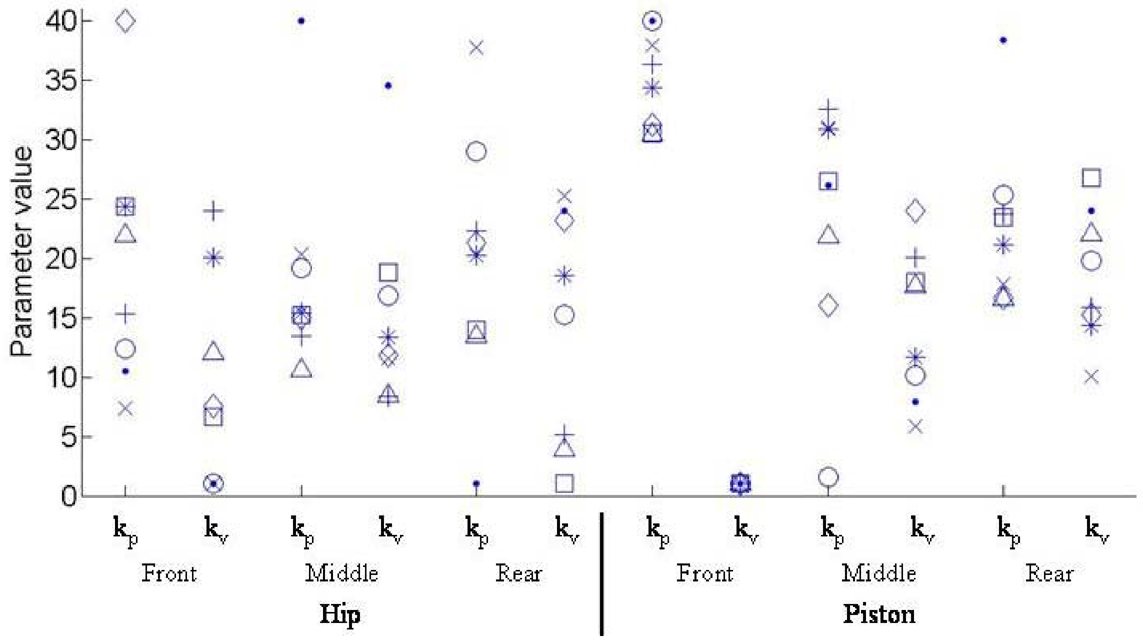
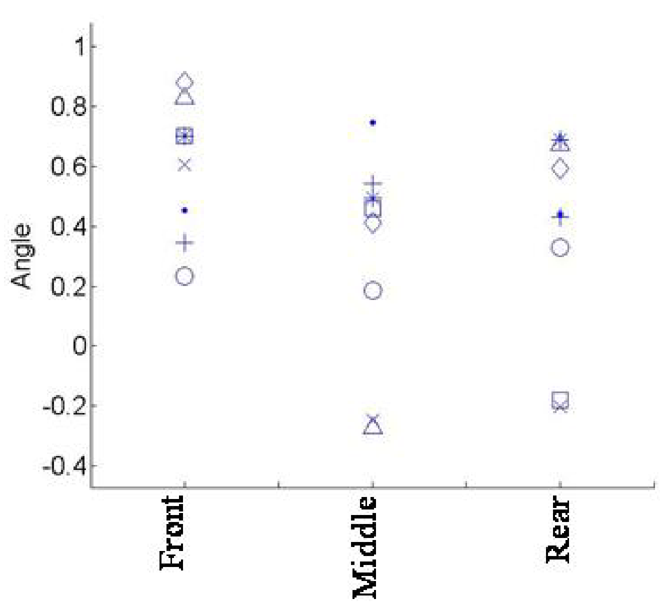


Figure 4: The plot above shows the final k_p and k_v values for various different simulations.

The hip angles found by the simulation varied widely, as shown in figure 5. This was a surprising result since the hip angle was found to control the speed of the robot significantly in laboratory experiments with iSprawl. In the gait the simulated robot is executing, the hip angles may not make much of a difference.

The body dimensions varied, but in almost all cases the body length was larger by a factor of two than the body width (see figure 5). This result is similar to what intuition suggests, that a long narrow body is optimal for facilitating the body to go in the straightest possible direction (e.g., the long narrow car bodies of drag racers).



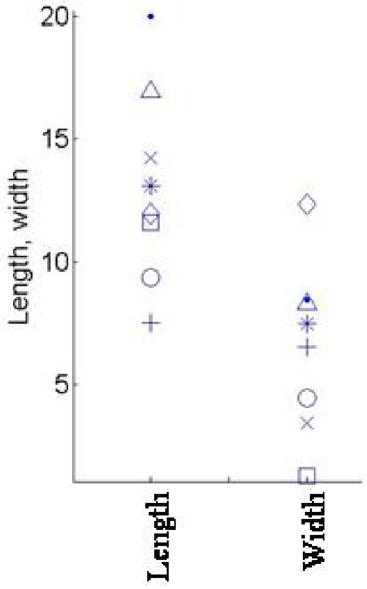


Figure 5: The left plot shows the final hip angles, and the bottom right plot shows the final body dimensions.

Conclusions

Performance Assessment

We found that Hill-Climbing and Coordinate Ascent both do well, converging to similar rewards; final gaits performed significantly better than with the initial parameters. One added benefit of these algorithms (and partially the problem setup) is robustness – all our trials tended to converge to high rewards even with very "bad" initial parameters. A qualitative analysis of the robot gaits revealed that they tended to exhibit the tendency of "bounding" motions. Most parameter values seemed to converge to within a range, although there were several parameters in the obtained gaits that varied greatly between gaits.

There are several caveats to the results obtained. The main issue is that we currently have no theoretical upper bound on speed, and thus we cannot gauge how close the gaits are to the true optimal gait for the specific contact model, although we suspect that the algorithms converged to a close approximation of the true optimal gait. Also we have not yet tested the parameters on the real robot to determine whether our parameters really translate to a good gait in real life.

Future work

One area for further work ahead is obtaining theoretical results on an upper bound for robot speed, to get an idea of how close our solutions are to optimal. A useful experiment would be to match simulation parameters to the actual iSprawl robot, to compare the simulation performance to actual performance on iSprawl. Finally, we plan to expand on this work by exploring other possible algorithms for gait/body dimensions and compliance optimization.

Acknowledgements

We thank Prof. Mark Cutkosky and the Biomimetics lab for use of the lab resources in performing these simulations. We also thank Daniel Santos and Jonathan Karpick for help with the Arachi simulator.