The 35th Israeli Conference on Mechanical Engineering – ICME 2018
Beer-Sheva, 9-10 October 2018

A Simple State Sensing Method for Dynamic Running

Omer Nir[1], Amir Degani[1]

1 Technion's Autonomous Systems Program, Technion-Israel Institute of Technology, Haifa, Israel 32000 (e-mail: OmerNir@campus.technion.ac.il)
2 Civil and Environmental Engineering Department and Technion's Autonomous Systems Program, Technion-Israel Institute of Technology, Haifa, Israel 32000 (e-mail: adegani@technion.ac.il)

1. Abstract

State sensing for dynamic running typically requires a combination of accurate and high-bandwidth sensors. In this paper we introduce a model-based estimation method that only requires a contact sensor to implement. We utilize relation between the stride phases durations and the state of the robot. We evaluate our estimator in simulation with typical modeling and measuring errors. Finally, we perform a test-case analysis with a classic control scheme to evaluate how the proposed estimator handles an uneven terrain. The results indicate our proposed estimator can be used as a practical tool for state estimation for SLIP-like running robots.

Keywords: Underactuated Robotics, Legged robots, Dynamic locomotion, State estimation

2. Introduction

Legged locomotion has unmatched abilities to traverse unstructured terrain compared with wheeled systems [1]. A popular model for dynamic legged locomotion is the Spring Loaded Inverted Pendulum (SLIP) model. Sensing state for SLIP-like robots requires using a combination of sensors such as accelerometers and encoders. Sensing either of the state variables pose significant technical difficulties, especially with running gaits, where the stance phase is typically short and inertial measurements are required. Most work related to control strategies for the SLIP model for running do not explicitly discuss the state estimation problem. Often, researchers assume full state knowledge in analysis and simulation, and run into significant difficulties in experiments.

3. Modeling and methodology

The SLIP model, first introduced by Blickhan in 1989 [2], describes the dynamics of legged running. The model consists of a point mass attached to a springy massless leg, as depicted in **שגיאה! מקור ההפניה לא נמצא.** In the model a stride has two phases: flight phase, in which the leg is not in contact with ground and the body follows a ballistic trajectory, and a stance phase in which the leg is in contact with the ground and the system behaves as a springy inverted pendulum To generalize our discussion, we introduce the non-dimensional SLIP model, similar to [3], [4]. For the non-dimensional SLIP, we define the non-dimensional state variables: $\tilde{\zeta} = \zeta/\zeta_f$, $\tilde{\varphi} = \varphi$, $\tilde{x} = x/\zeta_f$, $\tilde{y} = y/\zeta_f$, and $\tilde{t} = t\sqrt{\zeta_f/g}$. Substituting the non-dimensional state variables into the stance equation, we arrive at the non-dimensional state equation

$$\begin{bmatrix} \ddot{\tilde{\zeta}} \\ \ddot{\tilde{\varphi}} \end{bmatrix} = \begin{bmatrix} -\beta\left(\tilde{\zeta} - 1\right) - sin(\tilde{\varphi}) + \tilde{\zeta}\dot{\tilde{\varphi}}^2 - c_\zeta \dot{\tilde{\zeta}} \\ -2\dot{\tilde{\zeta}}\dot{\tilde{\varphi}} - cos(\tilde{\varphi}) - c_\varphi \dot{\tilde{\varphi}} \end{bmatrix}, \qquad (1)$$

where $\beta = k\zeta_f/mg$. Similarly, the non-dimensional flight dynamics simplify to

$$\begin{bmatrix}\ddot{\tilde{x}}\\ \ddot{\tilde{y}}\end{bmatrix}=\begin{bmatrix}0\\-1\end{bmatrix}. \quad (2)$$

The model parameters and state variables are defined in **שגיאה! מקור ההפניה לא נמצא.**. Other than the two damping coefficients, equation (1) has only one parameter, β, defining the stance dynamics, and the flight phase (2) has no free parameters. The damped SLIP model describes a damped, non-conservative, system. The damping occurs with each stance phase, it is modeled as viscous damping. To modulate the system's energy, we compensate for dissipation of energy by preloading the springy leg during each flight phase which is instantaneously released at leg touch-down.

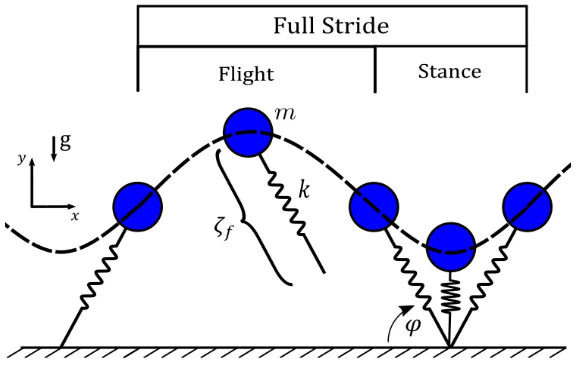
Similar to Altendorfer, Koditschek, and Holmes, we can differentiate between two state sensing types: 1) Body frame sensing, i.e., sensing the internal state of the robot. 2) Inertial state sensing, that is, measuring the state of the robot's inertial frame. Depending on sensors quality and performance, they are often expensive, and usually one type of sensor is not enough for full state estimation. In general, body frame sensing is easier to accomplish than inertial sensing, but the latter is necessary for running and hopping gaits [5].

To estimate the state of the system we numerically evaluated the durations of flight and stance phases for many initial states and different control inputs, $C(\varphi,\zeta)$, leg angle and length. For a cross-section of the parameter space where $\varphi\ and\ \zeta$ are set, $C(\varphi,\zeta)=const$, the relation between $\bar{\chi}=\{\dot{x},y\}$ space and $\bar{T}=\{T_f,T_s\}$ space is injective, meaning that by knowing the mapping function and $\bar{T}$ we can calculate the state $\bar{\chi}$. It is easy to measure $\bar{T}$ using a clock and a contact sensor at the tip of the leg. When the system is at stance phase the leg touches the ground and the contact sensor is engaged. When the leg lifts off from the ground, and the system enters flight phase, the sensor is disengaged.

We can utilize the injective relation between $\bar{\chi}$ and $\bar{T}$ to find the state of the system by numerically solving the inverse dynamics for $\bar{T}$. However, to use the estimator in real-time we need to accelerate the estimation process. For real-time performance we favor mapping the relation between $\bar{T}$ and $\bar{\chi}$. We based our mapping on a neural network function (f_{NN}) trained with approximately 20,000 randomly generated initial states. Once trained, the estimator's inputs are T_f, T_s, φ, ζ, and the corresponding output is the system's state, $\bar{\chi}=\{\dot{x},y,\dot{y}\}$, at the apex of the stride ($\dot{y}=0$).

4. Simulation results

To evaluate the estimation error, we first discretized the state space. For each discrete state we found the dead-beat control action, $C_{DB}(\varphi,\zeta)$, that will bring the robot to the starting state after the stride, $\bar{\chi}_i=f(\bar{\chi}_{i-1},C_{DB})$. We evaluated the non-dimensional estimation errors for state variables $\tilde{\dot{x}}$ and $\tilde{y}$ at each discrete state, as depicted in Figure 2-a**שגיאה! מקור ההפניה לא נמצא.**. Finally, we calculated the mean estimation error and the standard deviation over the entire state space. The estimation error for both $\dot{x}$ and y is below 0.1 for most of the state space, and the error is



mostly*Figure 1. The Spring Loaded Inverted Pendulum model with state variables and model parameters.*

Modeling is never perfect, an estimator sensitive to modeling errors will perform poorly. In our case, a modeling error is the difference, Δβ, between the estimator's model and the exact model. The value of Δβ represents our ability to accurately measure the system's parameters.

To evaluate the estimator's sensitivity to modeling errors, we chose an arbitrary point in state space, $\tilde{\dot{x}} = 1.6$ $\tilde{y}$=1.3. We evaluated the estimation error for the system with different β values ranging from 10 to 50, (typical values for a human runner) and for different Δβ values ranging from 0 to about 6. The results of the sensitivity analysis are plotted in Figure 2-b **שגיאה! מקור ההפניה לא נמצא.**. Note that the estimation of $\tilde{\dot{x}}$ is more sensitive than that of $\tilde{y}$. We hypothesize that the difference in sensitivities is due to the indifference of flight phase dynamics to modeling errors.

As for modeling errors, measuring errors are also unavoidable. In our case, a measurement error is a deviation of the measured duration of a phase (T_f and/or T_s), from the actual duration. A phase duration measurement error, $\Delta\tilde{t}$, is comprised of a random noise and a constant bias. The random noise is caused by the mechanical and electrical properties of the sensor, while the bias originates from installation and tuning of the sensor. Typically, for decent quality sensors the error due to bias is dominant over the random noise.

To evaluate the estimator sensitivity to measuring errors we introduced a bias to the time measurement inputs of the estimator. As before, we chose the arbitrary test point, $\tilde{\dot{x}} = 1.6$ $\tilde{y}$=1.3 as a test case for analysis. The bias is in the non-dimensional time $\tilde{t}$, and ranges from 0 to 0.06. Results of the measuring sensitivity are plotted in Figure 2-c. Again, the sensitivity of $\tilde{\dot{x}}$ and $\tilde{y}$ estimates act differently, for large values of β, $\tilde{\dot{x}}$ estimation is quite sensitive, while $\tilde{y}$ estimate in the range tested, is almost invariant to measuring errors.

To test and evaluate the usefulness of the proposed estimator, we simulated running using the non-conservative model. We designed a simple feedback "test controller". This test controller is a variation of the well-known Raibert's controller [6]. Our controller decouples regulation of energy and velocities ratio, $E_i = 1/2m\dot{x}_i^{\,2} + mgy_i$ and $R_i = y_i/\dot{x}_i$, respectively. Both the desired energy E_d, and the desired velocities ratio R_d, are calculated from the desired set-point $\dot{x}_d$ and y_d at apex. This test controller is used to validate and demonstrate the usefulness of our estimator, it is therefore purposely standard and not optimized in anyway.

To test the steady-state performance, we first calculated the BoA of the test controller with full state knowledge to serve as a benchmark. The results, depicted in Figure 3-a, show a 9.4% reduction in BoA area when using our proposed estimator. The reduction is restricted to the circumference of the BoA, which is both expected and desirable.

To examine the transient performance, we calculated the progression of an arbitrary trajectory, shown in Figure 3-b, once with full state knowledge and again with our proposed state estimator. The trajectory of the system with the estimator roughly followed the trajectory of the system with full state knowledge, with no significant advantage to the full state knowledge. The error remained bounded and eventually constant as the system converged.

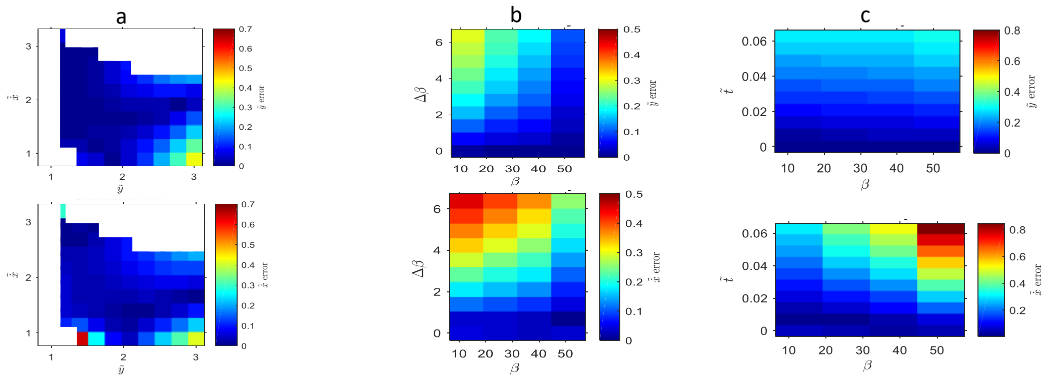


Figure 2. a) Non-dimensional estimation error. b) Modeling sensitivity c) Time measuring sensitivity.

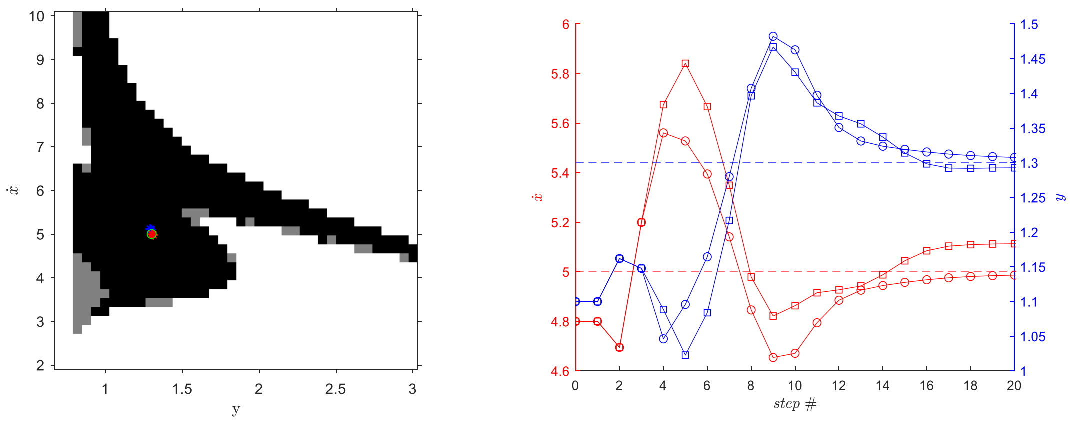


Figure 3. a) Basin of Attraction of the chosen controller. The black region depicts the BoA with our proposed state estimator. The enlarged region that also includes the gray BoA is with full state knowledge. The green circle represents the desired set point. The blue '' represents the fixed-point of the system with our state estimator, and the red 'x' represents the fixed-point of the system with full state knowledge. B) Convergence of apex state with full state knowledge (o) and estimated state (□). The red, $\dot{x}$, represents the horizontal velocity at the apex, and the blue, y, represents the apex height. The dashed lines indicate desired values.*

I. Conclusions

In this paper we presented a simple inertial state estimation method requiring only a single contact sensor to implement. We evaluated the performance of our estimator with simulated modeling and measuring errors. We gave a generalization enabling robot designers to evaluate the necessary modeling and sensing accuracy for their design if they wish to deploy the proposed state estimator to their robot. The results indicate adequate and practical performance of the estimation method, even under significant terrain height perturbations.

In this work, we did not examine the influence of stance control on the estimation. It is worth perusing in future work since it is a common control strategy. Of course, testing on real robots is also beneficial. We also believe that replacing the Neural-Network estimator with a simplified model can also improve the performance. Finally, we note that researches that already work with their own robot and sensors can, with relative ease, add our estimator to their existing system to improve performance.

II. References

[1] M. H. Raibert and E. R. Tello, "Legged Robots That Balance," *IEEE Expert*, vol. 1, no. 4, pp. 89–89, 1986.

[2] R. Blickhan, "The Spring-Mass Model for Running and Hopping," *J. Biomech.*, vol. 22, no. 11–12, pp. 1217–1227, 1989.

[3] R. M. Ghigliazza, R. Altendorfer, P. Holmes, and D. Koditschek, "A Simply Stabilized Running Model," *SIAM J. Appl. Dyn. Syst.*, vol. 2, no. 2, pp. 187–218, 2003.

[4] N. Shemer and A. Degani, "A Flight-Phase Terrain Following Control Strategy for Stable and Robust Hopping of a One-Legged Robot Under Large Terrain Variations," *Bioinspiration and Biomimetics*, vol. 12, no. 4, 2017.

[5] R. Altendorfer, D. E. Koditschek, and P. Holmes, "Stability Analysis of Legged Locomotion Models by Symmetry-Factored Return Maps," *Int. J. Rob. Res.*, vol. 23, no. 10–11, pp. 979–999, 2004.

[6] M. Raibert, "Legged Robots," *Commun. ACM*, vol. 29, no. 6, pp. 499–514, 1986.