Context-Based State Estimation for Hybrid Systems with Intermittent Dynamics

Sarjoun Skaff

CMU-RI-TR-07-06

Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

2007

Thesis Committee:
Howie Choset, Chair
Alfred A. Rizzi, Chair
Jeff Schneider
William 'Red' Whittaker
Henrik Christensen, Georgia Institute of Technology

(Defended 14 December 2006)

Contents

Acknowledgments

Through my journey at Carnegie Mellon, I have been accompanied by incredible people to whom I owe intellectual stimulation and strength to complete the journey. They are the face of Carnegie Mellon that will remain in my memory, and my gratitude goes to them. My office mates Chris and Pete from the Skyworker days and David and Jonathan form the RHex days have been there with me all along, always ready to share a big laugh and dive into deep discussions. I am particularly grateful for my friends and colleagues at CMU who have made this journey worth the trouble, particularly Bernardine, Chris, Pete, Paul, Vandi, Ben, Ashley, Sarah, Elie, Dave, Ariadna, Brett, Caroline and Keivan.

My great appreciation goes to my advisers Al Rizzi and Howie Choset, first and foremost for shaping and refining my critical thinking, and for helping me develop my writing and public speaking skills, thank you. My thanks also go to Red Whittaker, who co-advised me as a Masters student. Howie and Red have accompanied my journey from my very first day at Carnegie Mellon. They have provided me unwavering support all along, and I am indebted to them for that. I also thank my thesis committee members Jeff Schneider and Henrik Christensen for their guidance and expert help throughout the research.

I would like to express my deep appreciation for Pei-Chun Lin, for having so generously provided RHex's inertial data used in the experimental sections of this thesis. Many thanks to Hal Komsuoğlu for helping Pei-Chun generate the data, and to Hyungpil Moon for conducting the SLAM experiment in Chapter 5.

Conducting this research would not have been possible without the expert and gen-

erous help of faculty and students at CMU and elsewhere, including George Kantor, Chris Atkeson, Geof Gordon, Fernando de la Torre, Drew Bangell, Uluç Saranli, Dan Koditschek, Martin Buehler, Flavio Lerda, Goran Frehse, Scott Lenser, Brett Browning, and Hakan Younes. Thanks to all.

I have thoroughly enjoyed working with my colleagues at SBP, FRC, MSL, Biorobotics and CFR, specially Ralph Hollis, Suzanne Lyons Muth, Clark Haynes, David Maiwand, Hanns Tappeiner, Al Costa, Amir Degani, Dottie Marsh, Stephanie Matvey, Peggy Martin, Michele Gittleman and Chris Zeis. I am grateful for my colleagues on the Skyworker, Deminer, RHex, RiSE and medical snake teams for teaching me much of the knowledge and many of the skills I have acquired here.

Last but not least, my eternal gratitude goes to my family and friends at large, without whom none of this would have been worth the while. In particular, my mom and dad Elie and Josyan, to whom this thesis is dedicated.

Abstract

Robust state estimation is a key enabling technology for reactive control of hybrid systems, such as high performance mobile robots. Advanced mobile robots often exhibit intermittent interactions with their environment, resulting in high-order, high-dimensional dynamics that undergo high-frequency discrete changes. Estimating the state of such systems is complicated by the difficulty of modeling these rapidly evolving hybrid dynamics. This thesis makes state estimation more effective by recognizing that the hybrid dynamics can be approximated by a collection of simple models and identifying situations when specific models are applicable. Conventionally, state estimation for such systems is performed with multiple-model filters, but these filtering systems do not scale well as the number of models grows, and perform poorly in the face of high-frequency discrete dynamics. To overcome these limitations, this thesis develops an estimation framework that leverages available sensor information to improve the accuracy and scalability of multiple-model systems. The framework represents the dynamics with a hierarchy of contexts, and uses discrete-state estimation techniques to robustly identify the current context. The contextual information is then incorporated into multiple model filters to improve their accuracy and scalability. This thesis shows both experimentally and through simulation that integrating discrete and continuous estimation techniques enables accurate and scalable estimation for hybrid systems such as mobile robots with rapidly switching intermittent dynamics.

Chapter 1

Introduction

State estimation enables mobile robots to gain awareness of their situation within the environment and close the loop on motion controllers to correctly execute navigation plans. With scientific and technological advances, robot interactions with the environment result in increasingly complex dynamics that require high-bandwidth control. For such systems, estimation speed and robustness become paramount, as the successful execution of control strategies depends on the availability of accurate and timely information about the state. This thesis develops a framework for estimating the state of robots with complex dynamics by improving the performance of conventional estimation techniques and adapting them to the constraints of mobile robot estimation.

The presentation of this framework and its results are the principal subject of this thesis. To better frame the type of problem addressed, it helps to first position the problem in the larger field of state estimation, particularly as it applies to robotics.

State estimation has long been integral to signal processing, with notable contributions by Wiener, Kalman and Luenberger providing filters and observers to extract the signal from a data stream [86, 31, 32, 50, 51]. The common characteristic of these estimation tools is that they rely on models of the process generating the signal to filter the noise embedded in the data and recover the original signal.

In robotics, the same estimation principles are applied to localization and map-

ping, sensor fusion, and feedback control, among other applications. Instead of using models of the signal, filters and observers use motion models of the robot to interpret sensor data and recover the robot's state.

The description of some representative applications helps clarify points of difference between the problem addressed in this thesis and related research. For instance, the task of localization and mapping centers on the correct tracking of environment landmarks in the face of uncertainty in robot motion and sensor measurements (e.g. [41, 45, 33, 15]). Little attention is paid to situations where the motion dynamics are complex and difficult to model accurately. In contrast, this thesis focuses on estimating the state of the robot itself rather than that of landmarks, and explicitly addresses systems with complex dynamics whose state is difficult to estimate with conventional techniques.

Another application is sensor fusion, where the task is to integrate sensor information generated by different types of sensors and at different rates (e.g. [63, 77, 48]). The emphasis of sensor fusion research is on correctly incorporating different observations into an estimation filter to improve the accuracy of the state estimates. Similarly, this thesis seeks to make efficient use of available information to generate accurate state estimates. The difference is that the emphasis is not on fusing sensor data, but rather on developing a general framework to efficiently use the information, particularly in situations where some sensor data cannot be directly incorporated into the estimation filter.

In summary, the estimation problem addressed in this thesis consists of generating accurate state estimates for mobile robots with complex dynamics, through the efficient use of information available to the system. This problem definition is shared with other research projects, such as the design of an observer system to track a bouncing ball [62] or an observer to estimate the state of robots with non-holonomic constraints [26]. Such research typically involve the development of specialized estimation tools that handle a specific problem very well. This is a departure from the approach adopted in this thesis, which advocates an estimation

framework designed to improve the performance of conventional estimation tools that are widely used in most estimation fields. Thus, this research is not specific to a particular system, but is intended to provide a new estimation approach that can be applied to a variety of systems.

It is also important to place this work in the context of the hybrid control research field, which addresses similar classes of systems with intermittent dynamics [36, 1]. The hybrid control community approaches such systems in two ways [36]; the first is to devise control policies that ensure convergence and stable operation despite switching dynamics, such as deploying multiple Lyapunov functions to avoid chattering when dynamical modes switch [7]; the second is to model the discrete/continuous dynamics in a unified framework, an approach that often seeks to solve computer science problems such as model checking and verification [14, 75]. The research most closely related to this thesis is the development of unified modeling frameworks that explicitly express a system's hybrid properties. For example, Lemmon [38] develops a supervisory hybrid system consisting of a hybrid automaton whose discrete states represent an abstraction of continuous dynamics, and where transitions among the discrete states are triggered by the continuous state. It would seem that this modeling approach could be used to represent intermittent dynamics of mobile robots, but hybrid automata typically only handle simplified continuous dynamics (such as an integrator that tracks time [36]), and are brittle in the presence of noise. Therefore, alternative techniques are needed to account for complex dynamics and imperfect sensors, a problem addressed in this thesis.

This brief analysis shows that even though the systems of interest to this research are mobile robots with intermittent dynamics, the estimation framework applies to hybrid systems in general. Together with multiple-model filtering, the context-based estimation approach complements the existing body of research in control and modeling for hybrid systems.

The assumptions underlying the estimation framework are as follows. The systems of interest are rigid bodies, possibly equipped with inertia-less appendages.

The robots are expected to have intermittent interactions with the environment, and these interactions result in high-order hybrid dynamics along many degrees of freedom. Throughout the document, the term 'complex' dynamics refers to high-order, high-dimension and intermittent dynamics. The experiments are designed to validate the context-based estimation framework, and do not account for sensor failure.

The following sections describe the philosophical approach to state estimation advocated by this research and present an outline of the thesis.

1.1 Philosophical Approach to State Estimation

A first step in addressing the problem of state estimation for systems with complex dynamics is to recognize that the dynamics can be approximated by a collection of simple models, provided these models are used only when they are accurate. This approach reduces the need for complete motion models that capture the high-order components of the dynamics over six dimensions, and that may be intractable to design.

This multiple-model representation is appropriate for hybrid systems whose dynamics undergo discrete changes, as different dynamics can naturally be described with different models. The representation is also appropriate for non-hybrid but complex systems, in situations where different components of the dynamics can be approximated by simple models, even if the components do not switch discretely among each other.

Conventionally, state estimation for such systems is performed with multiple-model (MM) filtering systems, but these systems do not scale well as the number of models grows. An MM system associates multiple Kalman filters to each dynamical context and runs all filters simultaneously. It averages the individual filters' output weighted by their relative likelihood and generates a consolidated state estimate. This approach requires the activation of the entire set of filters, which can become

computationally intractable.

Multiple model systems can also output low-accuracy state estimates when the dynamics undergo sharp and recurrent transitions. To maintain the accuracy of the consolidated state estimate, individual filters should rapidly converge to correct likelihood values after a sharp transition. However, the computation of filter likelihoods can be incorrect if individual filters do not have enough time to converge before the dynamics change again. Likelihood estimates can also be inaccurate if the filters' observations have a low sensitivity to changes in the dynamics, as they would further delay filter convergence.

This thesis improves the scalability and performance limitations of MM systems by leveraging sensor information to reduce the systems' computational complexity and improve their output accuracy. To this end, the notion of context is defined and the dynamics are represented with a hierarchy of contexts as follows: dynamical contexts represent the dynamics that can be described by one model, and behavioral contexts represent specific sequences and frequencies of transition among dynamical contexts. In other words, contexts are discrete states that associate dynamics to the models that describe them. The approach seeks to exploit the context's hierarchical representation to efficiently and correctly determine which model is accurate.

Since contexts correspond to dynamics and the models that describe them, determining which model is accurate reduces to identifying the robot's current context. This can be done efficiently by using conventional discrete-state estimation techniques such as classification to determine the identity of the context at a bandwidth close to the update rate of onboard sensors. Knowing the current context determines which model is accurate enough to be used by MM filters when estimating the robot's state.

This approach can be better understood with the help of an example. Consider a legged robot that executes a jogging gait and alternates flight and stance contexts. The flight dynamical context is identified when the legs are extended (no contact with the ground) and when the body acceleration is close to gravity. Similarly,

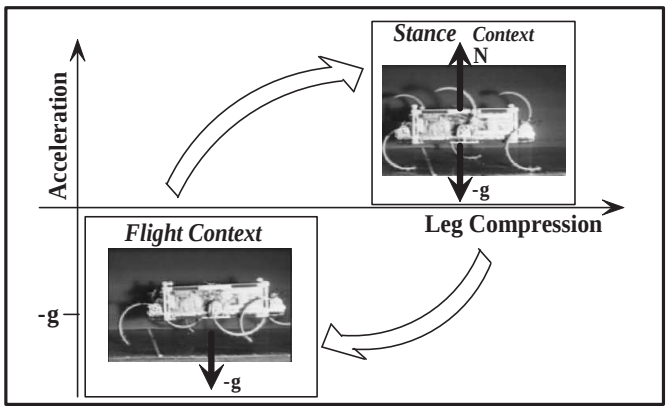


Figure 1.1: A jogging hexapod robot alternates flight and stance dynamical contexts. The contexts are identified by classification of leg compression and body acceleration measurements.

the stance context is identified when the legs are compressed and the acceleration measurements are positive (see Figure 1.1). Thus, the identity of the robot's dynamical context can be determined by classifying measurements of leg compression and body acceleration into the flight or dynamical contexts, at the rate at which measurements are made available to the system.

The jogging robot's behavioral context is defined by the frequency of transition between its flight and stance contexts. By alternating contexts at a specific frequency, the jogging dynamics causes acceleration measurements to follow specific patterns that can be efficiently detected by discrete-state estimation tools. The tool adopted is the combined use of hidden Markov models to recognize the spatial structure of a pattern and of timed automata its recognize temporal structure. Therefore, the jogging behavior is identified when the discrete-state estimators recognize acceleration patterns that correspond to jogging, as illustrated in Figure 1.2.

The ability to efficiently identify the dynamical and behavioral contexts improves the scalability and accuracy of conventional state estimation. The scalability of MM systems is significantly enhanced by first identifying the abstract context and then only activating the subset of filters that correspond to that context. This

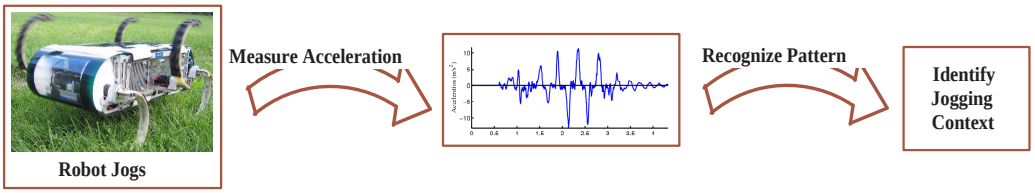


Figure 1.2: The robot's jogging behavioral context is identified by recognizing patterns in acceleration data that are specific to the jogging behavior.

reduces that computational complexity of the MM system, as it no longer needs to average the output of all filters in order to produce accurate estimates.

This approach also improves the output accuracy of multiple-model systems when sharp and recurrent context transitions take place. Sharp changes in the dynamics lead to distinct dynamical contexts that can be classified rapidly and unambiguously. This information directs MM systems to assign correct weights to individual filters, even if the filters' own likelihood estimates are incorrect. As a result, the contribution of accurate filters is emphasized and the contribution of inaccurate filters de-emphasized, which improves the timeliness and accuracy of the consolidated state estimates.

Context-based estimation applies to hybrid systems subject to recurrent and rapidly switching dynamics. Figure 1.3 depicts a family of existing mobile robots that share these dynamic characteristics and therefore benefit from the approach. The figure also provides a graphical summary of the approach. Conventional state estimation for any such mobile robot requires the deployment of a large-scale multiple-model filtering system with significant computational overhead; the context-based framework reduces computational cost by deploying multiple small-scale MM systems, and further reduces computational cost and increases accuracy by determining which MM system and which individual filter are appropriate and any point in time. As depicted in the figure, behavioral context identification determines which small-scale MM system is appropriate, and dynamical context identification determines which filter within the small-scale MM system is appropriate.

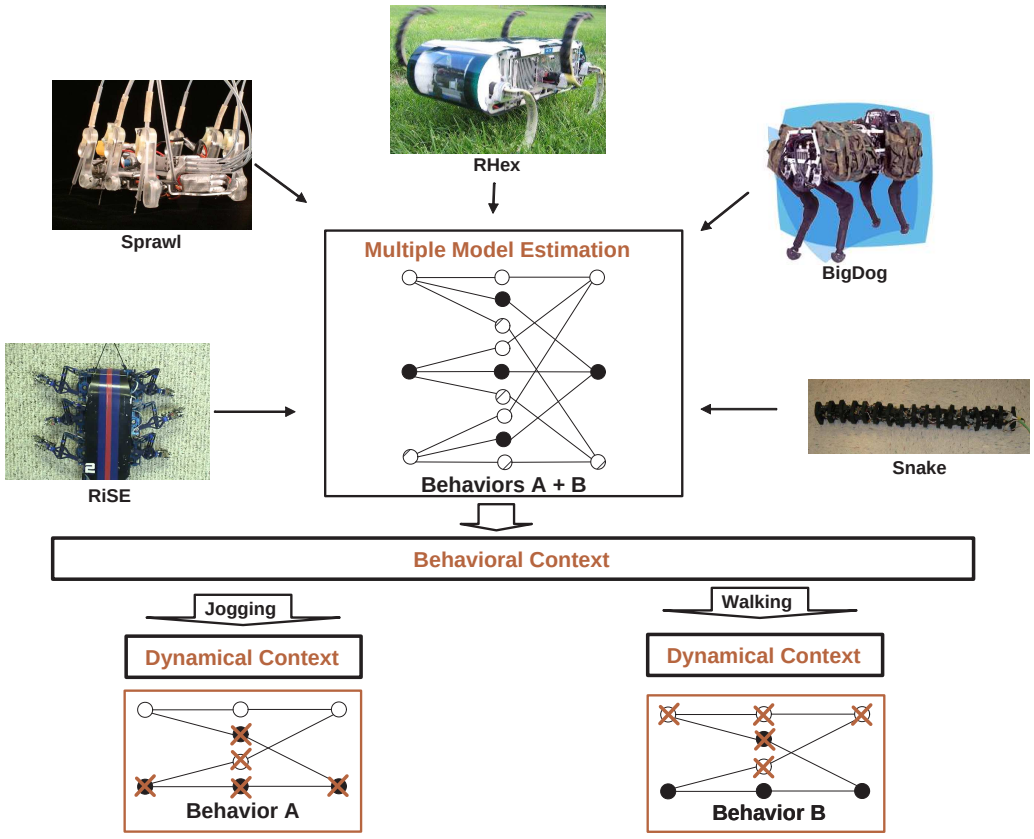


Figure 1.3: Context-based estimation applies to systems such as mobile robots that have recurrent and rapidly switching dynamics. Examples of recurrent dynamics are hyper-redundant robots, also known as snake robots [25, 12, 69]. Examples of rapidly switching dynamics span a spectrum of legged robots, such as Sprawl [34], a small-scale robot for horizontal running; RHex [65], a medium-scale robot for horizontal locomotion; Big Dog, a large scale robot also for horizontal locomotion; and RiSE [2], a medium-scale robot for vertical locomotion. For all these systems, identifying the behavioral context specifies which small-scale MM system to activate, and identifying the dynamical context determines which filter within the small-scale MM to activate.

It is worth noting that the combination of discrete and continuous estimation increases the efficiency of information utilization, as context identification can capture sensor information discarded by simple models and use that information to improve the performance of continuous filters. Additionally, the context identification approach provides the advantage of formalizing abstract models of the dynamics (such as representing jogging dynamics with a behavioral context), which is necessary to close the robot's behavioral control loop and improve its locomotion ability.

This thesis shows that separating the recognition of context from classical state estimation constitutes a rather intuitive framework for efficiently and accurately estimating the state of systems with complex dynamics.

1.2 Thesis Statement

State estimation for mobile robots with complex dynamics can be made more efficient and accurate by representing the dynamics with a hierarchy of contexts and combining discrete and continuous state estimation techniques. High-order, high-dimensional and intermittent locomotion dynamics are approximated by a collection of simple models and represented by a hierarchy of dynamical and behavioral contexts to ensure that models are only used when accurate. Discrete state estimation enables the efficient identification of the contexts, and the context information improves the accuracy and scalability of multiple-model continuous state estimation filters.

1.3 Contributions

This thesis makes two main contributions to state estimation. First, it introduces the notion of dynamical and behavioral contexts to explicitly associate locomotion dynamics to the motion models that describe them. This notion changes the way motion models are selected for state estimation; before, state estimation used motion models that correspond to the current *control policy*; now, selected models

have to correspond to the current *locomotion dynamics*. The new selection criteria is more accurate and can be exploited to improve the accuracy of state estimation.

The second contribution is the context-based estimation framework that leverages contextual information to improve the accuracy and scalability of multiple model filters. This framework consists of discrete state estimation techniques for identifying a robot's context, and of continuous estimation techniques that allow scalable and accurate estimation for systems with hybrid dynamics. More precisely, the framework uses classification, hidden Markov models and timed automata to identify dynamical and behavioral contexts, and a modified generalized pseudo-Bayesian 2 algorithm to generate estimates of the continuous state.

1.4 Thesis outline

The thesis starts with a detailed analysis of the state estimation problem as it applies to systems with complex dynamics (Chapter 2). A close examination of estimation mechanisms for hybrid systems reveals the circumstances under which the performance of models and model-based filters degrades. In particular, the analysis shows that multiple-model filters scale poorly as the number of models increases, and generate low accuracy state estimates when locomotion dynamics are of high order and change abruptly.

To overcome these limitations, Chapter 3 presents a new estimation approach that relies on a contextual representation of the dynamics to improve estimation performance. The chapter gives precise definitions of dynamical and behavioral contexts, describes means of identifying them with discrete-state estimation tools, and shows how context information improves the accuracy and scalability of conventional filters.

This framework is put to practice in experiment and simulation. First, the presentation examines the proposed discrete-state estimation methods for context identification; classification is used to identify dynamical contexts, and a combination of

hidden Markov models and timed automata is used for identifying behavioral contexts (Chapter 4). The techniques and their implementation are described in detail and their performance demonstrated experimentally on a hexapod robot. Second, context identification is shown to improve the accuracy and scalability of multiple model filters (Chapter 5). The thesis presents algorithms that incorporate context information into MM systems, and demonstrates the approach's effectiveness experimentally: the identification of the dynamical context is shown to significantly improve the accuracy of multiple model filters by rapidly determining which models are accurate; and the identification of behavioral contexts is shown to improve the scalability of MM systems, by enabling the deployment of multiple small-scale MM systems instead of a single large-scale system. An extension of the framework is also presented, whereby the identity of a robot's context is used to regulate the parameters of individual estimation filters and improve their output accuracy. Additional development of algorithms for simple context-based filter adaptation is provided in the Appendix.

The thesis concludes in Chapter 6 with a discussion of the research's results, potential applications, and extensions to the current framework that would further increase estimation accuracy.

Chapter 2

Problem Description

Estimating the state of physical systems is a longstanding necessity for aircraft fault detection and target tracking [9, 55, 61, 6], and for automotive control and diagnostics [80, 28, 16], to name a few applications. These industries invest large efforts developing complex models and deploying onboard sensors to obtain accurate state estimates from model-based estimation filters. Robust state estimation is also necessary for mobile robot control, particularly as increasingly dynamic robot motions require high-bandwidth feedback control. Unfortunately, the tools developed for aircraft and automotive estimation cannot be directly used in robotics for two main reasons. First, mobile robots are expected to interact with more challenging environments than cars, whose computer-controlled subsystems are generally designed to operate on flat roads. In contrast, robot-environment interactions often result in high-order, time varying dynamics that are difficult to model accurately. The second reason is that cost, power and size constraints of most mobile robots prohibit the use of specialized sensing and powerful computing as practiced in the aircraft industry.

These considerations motivate the development of an estimation approach that enhances the ability of existing techniques to use simple motion models and low-cost sensors and accurately estimate the state of mobile robots. The approach consists in part of approximating the high-order hybrid dynamics of mobile robots

with a collection of lower-order models to make the design of motion models more tractable. The accuracy of individual models is expected to vary with time, as the dynamics may fit the description of one model at one time and of another model at another time.

The typical estimation tools designed to take advantage of such a model setup are the multiple model (MM) filters [55, 53, 88, 23]. They were initially developed for aircraft fault detection, where a small number of models is used to describe possible faults and detect their occurrence. Applied to mobile robots, MM filters are limited by poor scalability characteristics. The filtering approach associates multiple Kalman filters to each model and runs all filters simultaneously. It averages the individual filters' output weighted by their relative likelihood to generate a consolidated state estimate. This approach requires the activation of the entire set of filters, which can become computationally intractable for robots that require a large number of models to accurately describe their different locomotion capabilities.

Multiple model filters can also produce inaccurate state estimates when sudden changes in the dynamics lead to sharp and recurrent variations in model accuracy. To maintain the accuracy of the consolidated state estimate, individual filters should rapidly converge to the correct likelihood after a sharp change of dynamics. However, three factors reduce the convergence rate of likelihood estimates, the first and second relating to the filtering mechanism itself and the third relating to the type of sensors available to the system.

First, the filter's ability to rapidly converge to the correct likelihood depends on the filter's confidence in its motion model's accuracy relative to the accuracy of the sensor measurements. In situations of low confidence in the motion model, individual filters assign relatively more weight to observations than to model predictions when updating the state. As a result, model predictions are close to observations, filter innovations[1] are small, and output likelihoods, which are a function of inno-

[1]Innovations are defined as the difference between model prediction and sensor observation.

vations, have a low sensitivity to variations in the observations. In other words, the lower the confidence the filter has in its motion model, the lesser the filter's ability to rapidly change its likelihood when the dynamics change abruptly, even if these changes are observed by onboard sensors. This limitation is likely to occur when using simple models, as filters often have relatively confidence in such models' predictions.

The second factor for slow likelihood convergence has to do with the multiple model system's mechanism for assigning weights to individual filters. The weights are a function of the current individual likelihood and of the weights computed at the previous estimation step. By tying current likelihoods to previous weights, this algorithm introduces some delay in redistributing the weights after an abrupt change in the dynamics.

Third, the convergence of likelihood estimates can be slow if the filters' observations are slow at detecting variations in the dynamics. For example, consider the task of estimating a body's position over time by using a filter that relies on position measurements. If the dynamics governing the body's motion change abruptly, they would induce immediate change in the body's acceleration, but only induce small change in its instantaneous position which could be hard to measure. This means that the position sensor has a low sensitivity to the dynamics, thus limiting the filter's ability to rapidly converge to the correct likelihood and detect when its model looses accuracy.

Slow convergence can lead to the failure of multiple model state estimation if the abrupt changes in the dynamics take place at a frequency higher than the convergence rate of individual filters. Such conditions drive the MM system to assign incorrect weights to individual filters, thereby causing the divergence of the consolidated state estimate.

This chapter describes the underlying causes of the scalability and accuracy limitations of MM filters in a bottom-up approach. Since MM filters are composed of individual filters based on motion models, it is safe to assume that the properties

of the motion models affect the performance of individual filters which, in turn, affect the performance of multiple model filters. Therefore, the following sections first analyze the impact of a model's design choices on that model's expressiveness and accuracy; second, a review of Kalman filters highlights their performance limitations when using simple models; and third, a description of multiple model filters shows their inherent scalability limitations as well as their accuracy limitations when individual filters suffer from performance problems.

2.1 Accuracy of Motion Model

Estimation filters use a combination of motion and sensor models to generate state estimates that are robust to noise [86, 31, 50]. Motion models describe the evolution of a system over time by capturing a representation of its dynamics. Observation models map sensor data into measurements of the state being estimated by the filters. The following brief description of the filtering mechanism explains how both types of models are used to generate state estimates.

At each sampling step, a filter compares a motion model's prediction of the state to observations of that state. The idea behind comparing predictions to observations is that the greater the agreement between the two values, the greater the confidence in the state estimate. In practice, model inaccuracies and sensor noise cause discrepancies between the two values, so the filter computes a best estimate of the state as the weighted sum of predictions and observations. Here, the weights are proportional to the relative accuracies of the model and of the sensors. Thus, in the presence of significant sensor noise, the filter assigns higher weights to the predictions, which limits the influence of inaccurate measurements on the state estimates.

Since state estimates combine predictions and observations, their accuracy depends in part on the motion model's accuracy. Therefore, careful examination of the assumptions and trade-offs underlying the model design is necessary to determine

if and when predictions are accurate. In this respect, an important measure of accuracy is the motion model's expressiveness, i.e., the extent to which the model captures the dynamics of the robot. Fully expressive motion models represent the entire dynamics and generate accurate predictions. Realistically, the representation of complex dynamics is likely to make simplifying assumptions that reduce the model's expressiveness and therefore limit its accuracy. Model expressiveness also dictates the extent to which information can be extracted from sensors to form the observations that are combined to predictions. Low-expressiveness models often cause filters to discard some sensor measurements that cannot be combined to predictions, which reduces the estimation accuracy.

To help understand which situations can lead to low model expressiveness, it is useful to first introduce some definitions. Robot dynamics govern the evolution of the complete state of the robot, called *robot state*. Accurate motion models seek to capture as complete a representation of these dynamics as possible. However, practical considerations often lead to limiting the representation to the dynamics that govern only a subset of the robot state, called *model state*. The space of robot states is typically multi-dimensional, so the model state is a *reduced-dimensional* representation of the robot state, and the motion model describes dynamics along some of the state dimensions and ignores the other dimensions. For example, a robot's motion on flat ground can be approximated with a planar model in $\mathbb{SE}(2)$[2] (translation and rotation in the plane), discarding dynamics along the remaining three dimensions even though they may be significant. For instance, a robot collision with an obstacle would generate significant vertical motions that modify the robot's position in the plane, but a purely planar model would be unable capture such a event. This dimension reduction simplifies the design of the model but reduces its expressiveness, as dynamics along unmodeled dimensions are treated as noise.

Model expressiveness can be further reduced when the robot dynamics are com-

[2] $\mathbb{SE}(2)$ is the special Euclidean group homeomorphic to $\mathbb{R}^2 \times \mathbb{S}^1$

plex. In this case, the represented dynamics can be of lower-order than the actual dynamics, discarding higher-order components that may be too difficult to model. A common example is modeling animal running as a spring loaded inverted pendulum (SLIP) [68], which is a largely accurate description of a running body's motion. However, the SLIP model does not capture the dynamics of the inelastic collisions of the foot with the ground, so these dynamics are treated as noise. Thus, *order reduction* is often a necessary design choice to make the modeling of complex dynamics tractable, but it clearly limits the model's expressiveness.

The following subsections describe the effect of low expressiveness on model accuracy and, by extension, on estimation performance.

2.1.1 Accuracy loss through order reduction

Reduced-order models typically capture an aggregate representation of the dynamics, i.e. they describe the low-frequency components of the dynamics and ignore higher frequency components. This means that low-order models are useful at describing the evolution of a system over an aggregate amount of time, but are inaccurate over short periods. As such, these models describe the average effect of the dynamics on the system, and are useful because they enable the filters to accurately estimate the state over time. However, if the filters update their state at high frequency, then the filters have to assign a low confidence value to the model's predictions relative to the sensors' observations. As discussed in Section 2.2.2, this setup reduces a filter's ability to rapidly adjust its likelihood estimates in response to abrupt variations in the dynamics, which reduces the accuracy of multiple model estimation.

Despite the limitations of reduced-order models, they remain a tractable tool useful for estimating the state of systems with complex dynamics. This thesis addresses a class of systems whose state can be estimated with such models, because their high-order dynamics cannot be easily modeled with a single, complete model. Rather, the dynamics are approximated with a collection of lower-order models, which,

when considered as a set, provide a comprehensive description of the the dynamics. Being of low-order, these models are only accurate in the presence of the dynamics which they are designed to describe, and lose accuracy when unmodeled dynamics dominate the motion. Therefore, the challenge is to build an estimation system that is able to incorporate a large number of models but rely only on the appropriate ones at the right time.

2.1.2 Information loss through dimension reduction

Reduced-dimension models limit the ability of state estimation filters to take full advantage of available sensor information. Sensors typically measure elements of the robot state, but some of these elements may not be part of the lower-dimension model state. Measurements of state elements that are common to the robot and to the motion model are said to be *compatible* with the model state. As the sensor model maps available measurements into measurements of the model state, measurements that are incompatible with the model state are discarded. In general, the greater the dimension reduction, the greater the likelihood of some sensor information being discarded. As an example, consider a robot equipped with an onboard six degree-of-freedom (DOF) inertial measurement unit (IMU), comprising three-axis accelerometers and three-axis gyroscopes. If the motion of the robot is represented with a collection of decoupled models, say one model per degree of freedom, then individual models would only be compatible with measurements along their own corresponding dimension, and measurements along the other five dimensions would be discarded.

Discarded measurements could contain information about a model's accuracy. The loss of this information hinders the ability of the estimation system to decide whether to rely on that model or discard its predictions, which could lead to the failure of state estimation. Consider the robot equipped with the 6-DOF IMU mentioned above, and assume that it locomotes on flat ground. A simple motion model (e.g. unicycle model) can describe the robot's translation and rotation in

the plane with sufficient accuracy to predict the robot's planar state. However, only three dimensions of the IMU output are compatible with the model, namely the acceleration along the x and y coordinates, and the rate of rotation around the yaw axis (see Table 2.1 for a summary of the dimensions of the different spaces). Measurements along the other three dimensions, acceleration along z and rate of rotation around the roll and pitch axes, are discarded. Therefore, a state estimator would be unable to use this additional information to compute estimates or to evaluate the model's accuracy. To see the impact of this limitation, imagine that the robot suddenly flips on its back with its wheels still touching the ground. The unicycle-based filter being "blind" to gyroscope roll information, doesn't accommodate the new robot configuration. The model assumes that control input still drives the robot forward, whereas the upside-down robot is moving backward. The erroneous predictions soon cause the divergence of the filter. If the system were able to rationalize the gyroscope roll information, it would infer that the model assumptions have been violated and that the model predictions have lost accuracy. In this case, a different model could be used, or in its absence, the filter would significantly de-emphasize the contribution of model predictions to state updates to reduce the risk of divergence.

	Dim.		Dim.
Robot State Space	6	Model State Space	3
Available Measurements	5	Compatible Measurements	3

Table 2.1: Dimensions of robot state, model state, measurement vector and compatible measurement vector.

Discarding sensor measurements reduces the system's efficiency at information extraction. In general, the more information is made available to the estimation system, the faster the estimation converges. Therefore, the advantages of reduced-dimensional models in terms of simplicity and tractability are offset by lower estimation performance. This motivates building an information processing system

that simultaneously uses simple models and seeks to improve the efficiency of information utilization.

In practical terms, this thesis considers that a major reason for low estimation performance is the filter's reduced ability to assess the accuracy of its model. Therefore, the information processing system it advocates seeks to use information discarded by reduced-order models to assess the accuracy of the entire model set and direct filters to only use appropriate models.

2.2 Accuracy of Single-Model Filters

The accuracy limitations of the motion models naturally induce performance limitations for the filters that use them. As already mentioned, state estimation filters (e.g. Wiener filters, Kalman filters, Luenberger observers) combine model predictions and sensor observations to generate state estimates [86, 31, 50, 51]. All types of filters make assumptions about the relative accuracies of model predictions and sensor observations. The following sections discuss the validity of these assumptions in realistic situations and the impact of violating them on the estimation performance. In addition, adaptation techniques are briefly reviewed to assess their ability to mitigate the loss of estimation performance. The discussion is grounded in the Kalman filter framework, a framework extensively used by the estimation community.

2.2.1 A brief description of the Kalman filters

Kalman filters enable the optimal estimation of the state of a system subject to noise under specific linearity and statistical assumptions. Optimality is guaranteed if the dynamical system is linear, and if the process and sensor noise have a known, white Gaussian distribution. Process and sensor noise expresses uncertainty in the model and in the sensor, respectively. For reference, a generic discrete linear system expressed in the Kalman framework is reproduced in Algorithm 1, with the

Algorithm 1 Kalman Filter

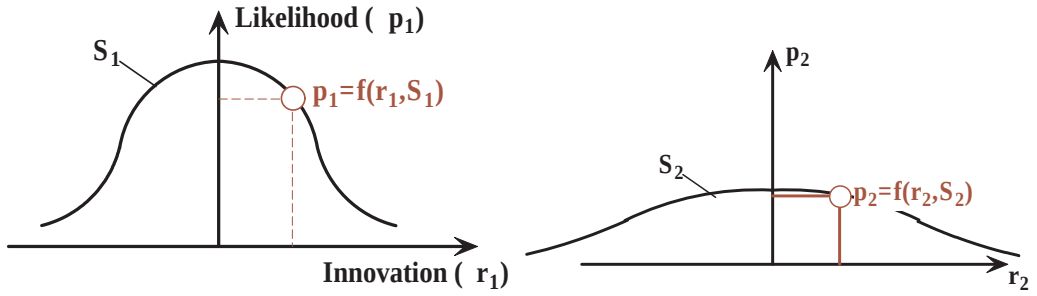
x_{k+1}	$=$	$Fx_k + Bu + \nu$	System Dynamics
y	$=$	$Hx_{k+1} + \omega$	Sensor Measurement
$x^p_{m,k+1}$	$=$	$Fx^u_{m,k} + Bu$	State Prediction
P^p_{k+1}	$=$	$FP^u_k F^T + Q$	Covariance Propagation
r	$=$	$y - Hx^p_{k+1}$	Innovation
S	$=$	$HP^p_{k+1}H^T + R$	Innovation Covariance
K	$=$	$P^p_{k+1}H^T S^{-1}$	Feedback Gain
$x^u_{m,k+1}$	$=$	$x^p_{m,k+1} + Kr$	State Update
P^u_{k+1}	$=$	$P^p_{k+1} - KSK^T$	Covariance Update
p_{k+1}	$=$	$\frac{1}{\sqrt{2\pi\|S\|}} \exp\left(-\frac{1}{2} r S^{-1} r^T\right)$	Innovation Likelihood

following convention used throughout the document: x stands for the system state; F for transition matrix; Bu for input; y for observation; H for observation matrix; x_m for estimated state; P for state covariance; k for sampling time; the indexes p and u for predicted and updated values, respectively; and finally ν and ω stand for model and measurement noise with covariances Q and R, respectively.

2.2.2 Loss of estimation accuracy when using simple models

Given the impact of the innovation likelihood p on the performance of multiple model filters, a close look at the likelihood's computation is warranted. The innovation likelihood equation shows that p is a function of the innovation r and its covariance S. The innovation is defined as the distance between the observed and the predicted state; the more accurate the model prediction, the shorter the distance between predictions and observations.

Incorporating the innovation into the computation of the likelihood serves the logical argument that the filter's output should be trusted if its predictions and



(a) Low innovation covariance (b) High innovation covariance

Figure 2.1: The likelihood is a a function of the innovation (r) weighted by the innovation covariance (S). Figure (a) shows a system where S_1 is low, resulting in a narrow bell curve. Variations in r_1 results in large variations of the likelihood p_1. In contrast, Figure (b) shows a flat bell curve system representing a large S_2. The shape of the curve explains that variations in r_2 lead to small variations of p_2, meaning that the likelihood has a low sensitivity to variations in the innovation.

observations agree. In addition, the likelihood incorporates innovation covariance S which expresses the filter's confidence in the innovation figure itself. A high confidence in the innovation is expressed with a small S, which translates into a 'bell-curve' distribution of p as a function of r (see Figure 2.1(a)). This distribution means that changes in the value of the innovation r induce pronounced changes in the likelihood p. In other words, the likelihood has a high sensitivity to the innovation. Conversely, a low confidence in the innovation translate into a 'flat' distribution of p, and the likelihood has a low sensitivity to the innovation (subfigure (b)).

Thus, if a filter is confident that its motion model and sensors are accurate, then it will confidently interpret a large innovation as an indication that the filter's output cannot be trusted. Conversely, if the filter is not confident in the model's or the sensors' accuracy, then variations in the innovation can be due to either process or sensor noise, and the likelihood becomes a more uniform and low-valued function of the innovation.

It follows from this analysis that when filters have a low confidence in their models, their likelihood has a low sensitivity to changes in the innovation. This means that if the individual filters of an MM system have a low confidence in their motion models, then they will be slow at computing likelihood estimates that correctly reflect the relative accuracy of each filter. This reduces the multiple filter's ability to rapidly adjust the weights associated with the individual outputs after an abrupt variation in the dynamics, leading to a loss of estimation accuracy.

2.2.3 Loss of estimation accuracy when violating filter assumptions

Apart from the likelihood's impact on multiple model filters, practical considerations limit the accuracy of individual Kalman filters by violating the assumptions under which optimality is guaranteed. For instance, few systems are actually linear, so their motion and sensor models have to be linearized in order to be used in a Kalman filter. This leads to the extended Kalman filter [27, 46], which replaces the process and sensor models F and H with their first-order Taylor series expansion $\partial F/\partial x$ and $\partial H/\partial x$, respectively. An alternative technique is the unscented Kalman filter [30], which accommodates a more accurate second-order linearization of the model. Still, both methods use reduced-order representations of the dynamics that treat higher-order components as process noise. Since these higher-order components are unlikely to have a Gaussian distribution, both the linearity and the statistical assumptions for optimality are violated. As a result, the performance of the Kalman filter is difficult to quantify and can be far from optimal. This is particularly true for the class of systems of interest to this thesis, where the dynamics are of high-order and therefore far from linearity. Nevertheless, this thesis purports that such dynamics can be approximated by a collection of linear models with reasonable accuracy, so the Kalman filter remain an attractive framework for multiple model state estimation.

Another problem common to all types of filters is their inability to adapt the

feedback gain K to variations of the model's accuracy. When robots operate in the field, unanticipated transient dynamics often contradict the predictions of motion models, which decreases the models' accuracy. In such situations, the feedback gain should increase in order to increase the weight of observations, decrease the weight of predictions, and reduce the risk of filter divergence. Unfortunately, estimation filters do not have a gain-regulation mechanism. At first glance, the Kalman filter algorithm may seem to regulate the gain, but a close inspection of Algorithm 1 shows that the feedback gain does not depend on the innovation, which is the filter's measure of model accuracy, and therefore does not adapt to variations in the operating conditions.

A possible remedy to this problem is filter adaptation, discussed next.

2.2.4 Improving estimation accuracy through filter adaptation

Adaptation consists of improving the accuracy of a system's parameters by incorporating new information as it becomes available. Adaptation could provide a solution to the problem of variable model accuracy, by either learning the model's parameters online to maintain its accuracy, or by modifying the feedback gain to de-emphasize the model's contribution when it becomes inaccurate. The following survey of available adaptation techniques shows that they are impractical for the class of systems addressed here, because of considerations of implementation complexity and slow convergence.

Learning the parameters of a model is called system identification [47, 74], whereby the model's structure is known *a priori*, but its parameters are identified through experimentation (e.g. [58, 85]). This is done by collecting sensor measurements and running optimization routines on the model's parameters to minimize discrepancies between measurements and model predictions. Unfortunately, this technique cannot be used to adapt the parameters online, because it requires the decoupling of the estimation and learning tasks, so that the model is identified first and then used for estimation [57].

Available extensions to system identification seek to simultaneously estimate the state and identify model parameters, but they are generally difficult to design and are only stable under restrictive assumptions [57, ch.4][78, ch.6]. Notable applications are the estimation of slow-moving system biases through state augmentation. The parameters being estimated are incorporated into the model, where they are usually represented with a stationary model corrupted by zero-mean noise [3]. An example is Strelow and Singh's work [77] to compensate for IMU bias. They incorporate the bias into the motion model, estimate its value at every filter update, and use the estimate to correct the biased measurements.

A related technique is adaptive control, where the system's input is designed to drive the model's parameters towards their true value [60, 8, 5]. Both adaptive control and simultaneous estimation and model adaptation offer asymptotic stability, but are not guaranteed to converge rapidly. This limits their applicability to dynamic robots, as their convergence may be slower than the rate of change of the dynamics. Another disadvantage is that specific adaptive systems have to be developed for each model, so they do not scale well as the number of models increases.

An alternative approach to model adaptation is to adapt the parameters of the filter to variations in model accuracy. For Kalman filters, Maybeck [53] augments the state of the model with the process and sensor noise covariances and provides an algorithm to continuously adapt their value. The drawback of this approach is its significant implementation complexity that requires many restrictive assumptions to ensure stability. A more tractable technique proposed by Escamilla-Embrosio and Mort [21] adapts the value of sensor covariance using fuzzy-logic iteration. They first compare the covariance S of the filter's innovation to the statistical covariance computed from a history of innovation values. Then they minimize discrepancies between the two values by modifying R using fuzzy logic, effectively tuning the filter's confidence in the sensor. Along similar lines but with a different implementation, this thesis proposes in the Appendix a technique to continuously regulate the process covariance of one-dimensional systems with minimal compu-

tational overhead.

A simpler approach to continuous covariance adaptation is to vary the covariances discretely, in incremental steps. For example, Bourassa and Kamoun [6] use two filters to track a maneuvering plane with a radar. Both filters use the same model assuming the plane flies in a straight line but have different process covariances. The filter with the lower covariance is used when the model can accurately describe aircraft in steady flight, and the filter with the higher covariance is used to track aircraft executing maneuvers, which makes the model inaccurate. In the same spirit, the first implementation of the regulation framework proposed in this thesis effects discrete change to the process covariance when a decrease in model accuracy is detected from the analysis of sensor data. The framework also enables continuous covariance regulation, but this development is beyond the current scope.

In summary, adaptive techniques are powerful tools for adapting slowly varying parameters. In contrast, this thesis seeks to develop an estimation framework that is explicitly designed to accommodate rapid variations in operating conditions.

2.3 Limitation of Multiple-Model Filters

The analysis so far has focused on the accuracy limitation of motion models and how these limitations reduce the performance of individual filters. This section extends this analysis to multiple model filters, whose performance depends directly on the performance of individual filters and the models they use. The discussion of MM filters' performance starts with a description of leading MM algorithms that demonstrates their scalability limitations, and ends with a discussion of accuracy problems specific to MM algorithms and those which are inherited from filter and model limitations.

Multiple-model filtering techniques were initially developed for aircraft fault detection and radar target tracking (e.g. [55, 4, 88, 54, 23]). These applications involve systems that operate in different modes, such as flying with various system faults

or executing distinct maneuvers, so different models are designed to represent the dynamics of the different modes. The assumption is that exactly one of these models will correspond to the current mode of operation, so MM filters identify the most accurate model at every sampling step and use it to output accurate state estimates.

Some research in the robotics community is adopting MM approaches for fault detection in mobile robots (e.g. [83, 79]). However, little attention has been paid to the specifics of estimating the state of robots with complex dynamics. Multiple-model filters could be used to accurately estimate the state of such robots, provided that their scalability and performance limitations are adequately addressed. The following sections detail the multiple model filtering process and discusses its limitations when applied to mobile robots.

2.3.1 A brief introduction to multiple model algorithms

Estimating the state of systems with multiple modes of operation typically involves associating hypotheses to mode transitions and evaluating the most likely hypothesis at every sampling step. Each hypothesis consists of a transition from a particular mode at the previous sampling step to a particular mode at the current step. A filter based on the model of the current mode is associated to each hypothesis, and its prior is the previous output of the filter based on the model of the preceding mode. The relative likelihoods of all filter outputs are computed at each sampling step, and the highest-likelihood output is adopted as the system's state estimate. For fault detection, the hypothesis corresponding to the highest-likelihood filter is believed to be correct, thus identifying the current mode and fault.

Clearly, the disadvantage of this approach is that the number of hypotheses increases exponentially with the number of steps. For example, consider a simple two-mode system. Between two consecutive steps, each mode can make two possible transitions to new modes, so the number of hypotheses after N steps is 2^N. The computational cost of running that number of filters rapidly becomes pro-

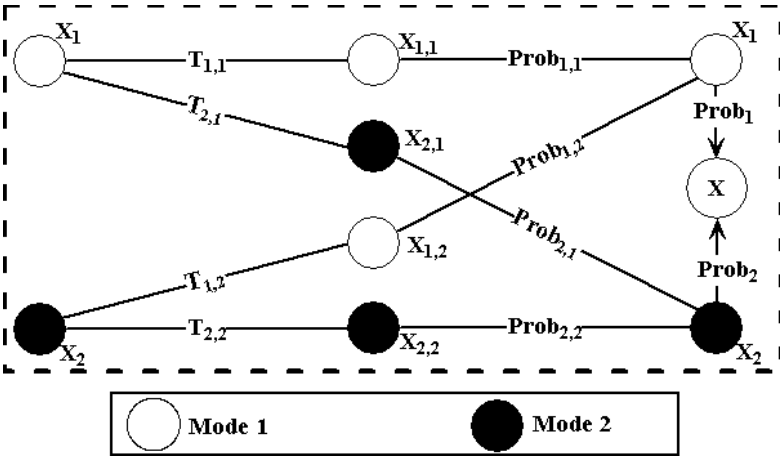


Figure 2.2: A two-mode GPB2 spawns $N^2 = 4$ filters, one for each mode sequence. Their output is consolidated to form the state estimate.

hibitive, so this problem has motivated extensive research into means of reducing the number of hypotheses. The most accurate of these techniques is the generalized pseudo-Bayesian 2 (GPB2) algorithm [11, 52, 42] expressed in the Kalman filter framework, which collapses the hypotheses at each sampling step into a set whose cardinality is equal to the number of modes. This way, the number of hypotheses is maintained constant from one sample step to the next, which improves the estimation tractability.

The GPB2 cycle is shown schematically in Figure 2.2 for a two-mode system over one sampling step. The initial two hypotheses (Mode 1 and Mode 2) spawn four hypotheses as expected, but those are collapsed back to two before the next cycle starts. The collapse is made possible by the consolidation of the outputs of all filters based on the same model. The individual outputs of those filters (e.g. $X_{1,1}$ and $X_{1,2}$) are weighted by their relative likelihoods ($Prob_{1,1}$ and $Prob_{1,2}$) and summed to form the mode-specific estimate (X_1). At every iteration, a best estimate (X) can be extracted by consolidating the mode-specific estimates.

Another interpretation of this approach is that the estimates of the individual filters are weighted according to their relative likelihood, and all weighted outputs are then consolidated into the system's state estimate. This way, the greater the

Algorithm 2 Steps of a GPB2 update [11]

$$
\begin{aligned}
[x_{i,j},\, P_{i,j}] &= filter_i\,(x_j) && (2.1)\\
p_{i,j} &= \frac{1}{\sqrt{2\pi S_{i,j}}} \exp\left(-\frac{1}{2} r_{i,j} S_{i,j}^{-1} r_{i,j}^T\right) \text{ (Likelihood)} \\
T_{i,j} &= \text{Prob}\,(i_t | j_{t-1}) \text{ (Transition Probability)} && (2.2)\\
Prob_{i,j} &= \frac{p_{i,j} T_{i,j} Prob_j}{\sum_j p_{i,j} T_{i,j} Prob_j} \text{ (Probability}\,(j_{t-1}|i_t)\,) && (2.3)\\
x_i &= \sum_j x_{i,j} Prob_{i,j} \text{ (Mode State)} && (2.4)\\
P_i &= \sum_j Prob_{ij} \left(P_{ij} + (x_{ij} - x_i)\,(x_{ij} - x_i)^T\right) \\
Prob_i &= \frac{\sum_j p_{i,j} T_{i,j} Prob_j}{\sum_i \sum_j p_{ij} T_{ij} Prob_j} \text{ (Probability}\,(i_t)\,) && (2.5)\\
x &= \sum_i x_i Prob_i \text{ (Consolidated State)} && (2.6)\\
P &= \sum_i Prob_i \left(P_i + (x_i - x)\,(x_i - x)^T\right)
\end{aligned}
$$

likelihood a filter's output, the greater its contribution to the overall estimate. If the filters are able to accurately estimate their likelihood, then the GPB2 would deliver estimates as accurately and rapidly as the most accurate individual filter.

Described more formally, each GPB2 iteration starts with the assumption that any mode could have been in effect at time $t-1$, and any mode could be in effect at time t. For a system with N modes, a bank of N^2 filters is updated and the output of all filters is consolidated. Algorithm 2 details the steps involved in a cycle. Here, $filter_i$ is based on the model of mode i; i_t represents the hypothesis that mode i is in effect at time t; $x_{i,j}$ is the state estimated by $filter_i$ and whose prior is the output of $filter_j$; r is the residual (innovation); and S and P are the innovation and process covariances, respectively.

2.3.2 Scalability limitation of MM filters

Even though the development of algorithms such as the GPB2 has given multiple model filtering practical value, all such algorithms still require the simultaneous activation of the entire set of filters in order to compare and consolidate their output. This limits the applicability of these algorithms to systems with a small number of models, as the computational cost of individual Kalman filters is often significant. In contrast to conventional systems using MM filters, robotic platforms are generally constrained by size, energy and computational power. In addition, advances in science and technology are enhancing the capability of mobile robots, and they can require large numbers of models to capture increasingly complex dynamics. Therefore, the scalability of estimation algorithms is important to their applicability to robotics.

2.3.3 Accuracy limitation of MM filters

As can be inferred from Algorithm 2, the accuracy of the MM state estimate is less than or equal to that of individual filters. If the filters incorrectly estimate the likelihood of their output ($p_{i,j}$), then the overall accuracy will be lower than that of the most accurate filter, as large weights ($Prob_{i,j}$) would be assigned to low-accuracy outputs.

The risk of incorrectly estimating the likelihood is high for mobile robots that interact aggressively with the environment. These interactions result in dynamics that undergo sharp and recurrent changes which abruptly modify the accuracy of the different models. In response, the weights assigned to filter outputs should be modified rapidly and appropriately in order to maintain the accuracy of the consolidated estimates. To evaluate the speed at which the weights can be modified, it is necessary to examine the mechanism that computes them. Equation 2.3 shows that the weights are a function of the likelihood $p_{i,j}$ of individual filters and of the previous weights $Prob_j$. As discussed in Section 2.2.2, $p_{i,j}$ can be slow to converge when using simple models, which slows the correct assignment of MM weights.

Another source of latency in updating MM weights relates to incorporating previous weights in the computation of the current weights. This dependency introduces a lag between $Prob_{i,j}$ and $p_{i,j}$, so that when $p_{i,j}$ varies sharply in response to an abrupt change in the dynamics, $Prob_{i,j}$ can only updated gradually to a new value, since it is weighted by its previous value.

MM systems can also be unable to rapidly update the weights of individual filters if the sensor information made available to the filters is slow at detecting changes in the dynamics. Such delays can be due to sensors that have low sensitivity to the dynamics, or to simple models that force filters to discard high-sensitivity sensor information. The following sections detail the sources of these delays and their impact on estimation accuracy.

2.3.3.1 Low-sensitivity sensors

As discussed previously, constraints imposed on robot platforms often limit the number and type of available sensors. If onboard sensors generate a signal that is not directly correlated to the dynamics, then they would have a low sensitivity to changes in the dynamics and would therefore be slow at detecting them. Examples include position sensors such as cameras and laser range finders and, to a lesser extent, velocity sensors such as gyroscopes. To understand why such sensors may be slow at detecting changes in the dynamics, recall from Newton's second law that interaction forces are proportional to body accelerations. It follows that acceleration sensors would be most sensitive to the dynamics. However, position is the double integral of acceleration, so large variations in acceleration induce small variations in position. Therefore, position sensors have a lower sensitivity; even if a sensor measures the body's position along a dimension that is aligned with the interaction force, it would provide attenuated information about the dynamics. This slows the MM system's convergence on the correct likelihood distribution, and thus leads to incorrect weight assignments to the filter outputs and low estimation accuracy. The same analysis applies to velocity sensors which are one integral away

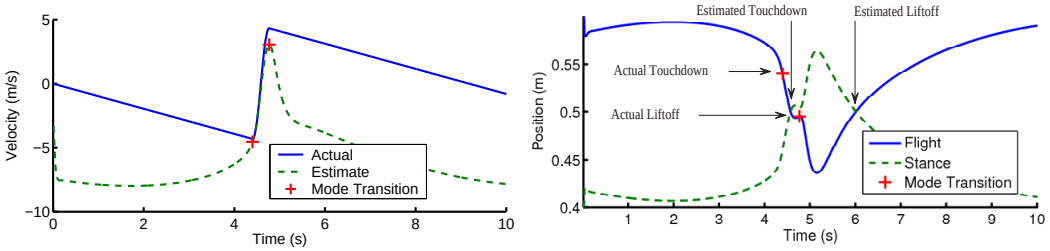
from acceleration.

The following simple simulation highlights the drawback of low-sensitivity sensing. The setup consists of an elastic bouncing ball that alternates flight and stance dynamics:

$$\ddot{z} = \begin{cases} -g, & \text{ballistic projectile,} \\ -K_z\,(z - z_0) - g, & \text{loss-less mass-spring system,} \end{cases}$$

where the state z is height, g is the acceleration due to gravity and K and z_0 are the ball's effective spring constant and rest length, respectively. The task is to estimate the ball's height and velocity using the flight and stance models and a noisy height sensor such as a laser range finder. Since the dynamics are described with two models, a GPB2 algorithm is used for estimation. For lack of better information, the probabilities of transitioning among the modes are set to be uniform, i.e. $T_{i,j} = \frac{1}{2}, \forall i, j \in \{\text{F, S}\}$, where F and S are the flight and stance phases, respectively. Four filters are updated for this two-mode system, one per mode sequence (i_t, j_{t-1}). Equation 2.1 outputs four states $x_{i,j}$, which are consolidated through (2.4) and (2.6) into x, the ball state estimate. Figure 2.3(a) shows that this setup generates inaccurate velocity estimates. The reason for the lack of accuracy is that observations have a low sensitivity to changes in the dynamics, so the GPB2 is slow at recognizing that the ball touched down or lifted off. This is evidenced in subfigure (b), where the likelihoods of the flight and stance modes suffer from a delay compared to the true simulated system. As a result of these incorrect likelihood estimates, inaccurate stance estimates are emphasized during flight and flight estimates during stance.

This analysis suggests that higher-sensitivity sensors such as accelerometers would improve the accuracy of the estimates. In practice, there is an accuracy trade-off between the different types of sensors. Position sensors have the advantage of providing absolute information about the body with respect to its environment, so they maintain their accuracy over time. This property is valuable for localization, which explains why position sensors are widely available on mobile platforms. However, position sensors' low sensitivity to dynamics induces inaccurate velocity



(a) Velocity estimates

(b) GPB2 outputs wrong estimates of mode probability

Figure 2.3: State estimates obtained with a conventional GPB2 lead to RMS errors of 6.34 cm and 15.5058 m/s for position and velocity estimates, respectively. The low sensitivity of the position sensor to changes in the dynamics does not allow the GPB2 to compute the correct probabilities for the flight and stance contexts. These plots indicate the instants when the ball touches down and lifts off with a '+'. Subfigure b shows that the probabilities of flight and stance erroneously cross each other far from the '+' points. Correctly estimated probabilities would cross close to the actual point of touchdown and liftoff.

and acceleration estimates, as illustrated in the previous simulation. In contrast, accelerometers are sensitive to changes in the dynamics, so they enable fast convergence to the correct filter likelihoods. Yet, position estimates derived from accelerometer data suffer from error accumulation and drift over time. Therefore, the challenge is to devise a framework to overcome the drawback of estimation based on position and acceleration sensors, so that state estimation can be accurate even if only one type of sensor is available onboard.

2.3.3.2 Information-Loss from low-dimension models

The accuracy of MM estimation can also degrade if the filters discard sensor data that provides relevant information about the dynamics. As argued in Section 2.1.2, filters based on low-dimensional models discard sensor measurements that are incompatible with the model state. Thus, dynamic forces that modify the accuracy of the estimation models and are measured by onboard sensors go unnoticed if they are not aligned with these models' dimensions. As a consequence, MM filters are unable to rapidly adjust their likelihood distribution as they cannot incorporate information about their models' accuracy. This leads to potentially assigning large weights to inaccurate individual estimates and producing inaccurate consolidated estimates.

Such situations commonly arise when, for simplicity, the robot's motion is represented with decoupled models. The underlying assumption is that such a representation reflects the true nature of the dynamics. In reality, dynamics are rarely decoupled and forces acting along one dimension usually modify the state along other dimensions, even if indirectly. These state modifications do not match the predictions of the decoupled models and are treated as noise; the larger the modifications, the higher the noise and the lower the models' accuracy. Filters based on these models can only measure the indirect effects of the dynamics on the model accuracy, and are therefore slow at converging to correct likelihood estimates. It follows that MM filters will be slow at assigning weights that reflect the accuracy

of the models. This analysis highlights the need for communicating sensor information to filters even if it is incompatible with their model state. Doing so would improve the overall efficiency of information extraction as well as the accuracy of the consolidated state estimates.

2.4 Summary

This chapter analyzes practical conditions under which conventional estimation strategies are likely to fail. The analysis addresses the accuracy limitations of simple motion models, which are often a low-order and low-dimensional description of the true dynamics. Accuracy limitations of motion models cause performance degradation for estimation filters that use such models. In turn, the performance limitation of filters lead to a loss of accuracy of multiple model systems that use these filters. These considerations call for the development of an estimation strategy that addresses the limitations of motion models, individual filters, and multiple model systems. Such a strategy is presented in detail in Chapter 3, and its results are demonstrated in Chapters 4 and 5.

This chapter also notes that multiple model filtering systems suffer from poor scalability characteristics. The estimation strategy presented in Chapter 3 addresses this issue in an effort to make MM systems more tractable for robotic systems with multiple modes of operation.

On occasion, attempts to improve the computational tractability of algorithms such as MM systems are countered with the argument that Moore's law about the rapid increase in processor performance solves this problem independently. For robotic applications, however, there is a strong motivation to minimize the computational load of low level tasks such as estimation and control to free onboard resources for higher level tasks such as planning. Robotic mobility has already seen a paradigm shift from sense-think-act to performing all three processes simultaneously. Increasingly, robotic applications include even higher tasks such as planning for the

deployment of scientific instruments [73], coordination with other robots [18], and reasoning about energy consumption [81]. The steady increase in task requirement and robot capability comes at the cost of greater demand for computing resources. Therefore, enabling estimation for advanced-mobility robots while reducing the computational overhead is justified practically as well as philosophically.

Chapter 3

Technical Approach

As stated earlier, the dynamics of robot locomotion are difficult to model accurately, which complicates the task of estimating the state of mobile robots. This problem can be addressed by approximating the dynamics with a collection of models that provide a comprehensive description of the dynamics when considered as a set, and are simple to design when considered individually. The accuracy of individual models varies with the dynamic situation, as locomotion dynamics may fit one model's description at one time and another model's at some other time.

Multiple-model (MM) filters are typically used for estimating the state of such systems, but they suffer from accuracy and scalability limitations which can make them impractical for mobile robot applications. MM systems activate their entire set of filters to evaluate the accuracy of individual filters, weight their output according to their accuracy, and consolidate the weighted outputs into an overall state estimate [11, 55]. The computational cost of this approach increases as the number of filters grows, and can become prohibitive for mobile robots because of limited computational budget. In addition, using simple models reduces the ability of the MM system to correctly evaluate the filters' accuracy at high bandwidth, which reduces the accuracy of the consolidated estimates when the dynamics change rapidly. These limitations are described in detail in Chapter 2.

This thesis improves the accuracy and scalability of state estimation for hybrid

systems by representing intermittent dynamics with a collection of motion models and combining discrete and continuous estimation techniques to efficiently identify which model is appropriate for estimation. The approach introduces the notion of *context* and represents the dynamics with a hierarchy of contexts as follows; *dynamical* contexts represent the dynamics that are described by one model; and *behavioral* contexts represent specific sequences and frequencies of transition among dynamical contexts. Current dynamical and behavioral contexts are identified through the classification of data generated by onboard sensors. The identification of the dynamical context determines which model accurately represents the dynamics. Therefore, by using classification, this approach enables the identification of accurate models at a bandwidth comparable to that of the fastest onboard sensors. This information helps MM systems evaluate their filters' accuracy rapidly and correctly, thereby increasing the accuracy of the consolidated state estimates. On the other hand, the identification of the behavioral context enables the deployment of multiple small scale MM systems to replace the conventional monolithic systems. This reduces the computational requirements of multiple model filters and substantially improves their scalability.

This chapter describes the proposed approach in detail, including the definition of dynamical and behavioral contexts; the identification of contexts through classification; and the combined framework of context identification and MM filtering that improves the accuracy and scalability of MM filters, and adapts them to mobile robot applications.

3.1 Dynamical Contexts

The concept of dynamical contexts is to associate models to the dynamics they represent in order to facilitate the identification of appropriate models. This is formalized by defining *dynamical contexts* as the dynamics that can be described by a single model. For example, consider the toy problem of the bouncing ball from Chapter 2, where the ball falls and bounces repeatedly on the ground. The

ball's dynamics can be represented by flight and stance models, so the ball can be expected to be in the flight or stance contexts at any given time. Thus, the problem of determining which model is most accurate reduces to identifying the current dynamical context, as this information simultaneously recognizes the dynamics and the model that best represents them.

This formulation of the problem emphasizes the importance of explicitly recognizing the dynamics as a means to rapidly identifying appropriate filters in a multiple model system and thus generating timely and accurate state estimates. Conventionally, multiple model estimators indirectly infer the accuracy of the filters from the likelihood of their output. As shown in Chapter 2, this leads to delayed and inaccurate assessments of filter accuracy, particularly when using simple models and when the dynamics vary abruptly. In contrast, inferring model accuracy directly from observations of the dynamics can be fast and accurate. This is because the classification tools used to identify the contexts can have a high convergence rate. On one hand, statistical classification is independent of the models used by the MM filters, so they are not constrained by considerations of design tractability and can incorporate sensor information discarded by MM filters. On the other hand, the design flexibility of classification techniques facilitates the extraction of sensor information that the designer deems most relevant to the identification of the context. This means that classification tools can be adapted to mobile robots to maximize the bandwidth and accuracy of context identification.

Related context-based approaches can be found in the visual classification community, where *a priori* knowledge of the surroundings (context) is exploited to facilitate the recognition of objects in a scene [82]. Similarly, dynamical contexts encode *a priori* knowledge of the dynamics that is exploited to facilitate the identification of appropriate filters. The proposed tool for encoding that prior knowledge is classification, which builds statistical models of onboard sensor measurements and thereby enables rapid identification of dynamical contexts, as described in the next section.

3.1.1 Identification of dynamical contexts

The approach to context identification is based on the observation that since locomotion dynamics strongly affect the signal of onboard sensors, analyzing that signal would provide information about the dynamics themselves. From there, the approach calls for constructing statistical models that map sensor measurements to the dynamics that induced these measurements and hence to their corresponding model. Such statistical models consist of sets in the sensor space[1], each formed by clustering measurements generated while the system operates in one of its dynamical contexts. For real-world systems that are more complex than the bouncing ball, the clustering is performed by operating that system in a controlled environment, where the dynamics are steady and can be appropriately represented by their corresponding motion models. The measurements generated by onboard sensors are then clustered into sets labeled after the different dynamical contexts. This way, each set of measurements corresponds to a unique dynamical context whose dynamics are known to be accurately represented by an available model. With this setup, dynamical contexts are identified whenever measurements can be classified in one of the sets. Thus constructing the classes is done offline and may require some effort, but the classification itself is computationally inexpensive and can be performed in real time while the system operates in the field.

This concept can be clarified with the help of the same bouncing ball problem, where it is assumed that a noisy sensor onboard the ball can measure its height at all times. During the stance phase of the bounce, the ball's elastic body compresses and the height sensor outputs values that are lower than the ball's rest height. During flight, the sensor reports altitudes that are greater than the rest height. Thus, height measurements can be clustered into two groups, one with values lower and the other with values greater than the ball's height at rest. With this setup, a single measurement of the height enables the immediate identification of the flight and stance contexts through its classification in one of the two sets, as shown in

[1] A sensor space is the space of all possible measurements.

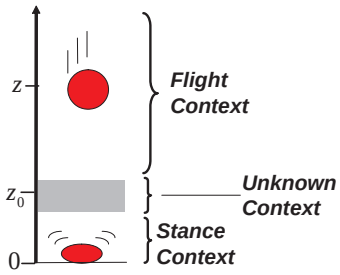


Figure 3.1: Representation of the bouncing ball's dynamical contexts. When the height measurements z indicate that the ball is higher than its rest height z_0, the ball is classified in the flight context. Conversely, height measurements that are lower than the rest length classify the ball in the stance contexts.

Figure 3.1. In addition to the two contexts, the figure illustrates an intermediary unknown context, that expresses the inability to classify the ball with certainty around the rest height because of sensor noise. A practical illustration of this approach is provided in Chapter 4.

Context identification could be implemented using different statistical classification methods from clustering. For instance, Lenser proposes a non-parametric time-series approach to recognize discrete elements of the robot's surroundings, such as the type of terrain traversed, from onboard sensors [39, 40]. The technique consists of learning statistical correlations between pairs of consecutive data points and a given environmental feature (e.g. type of terrain), and using such statistical models to classify current pairs of data points [39]. This approach could be used to model locomotion dynamics, but since these dynamics vary with time, it would be difficult to obtain training sets large enough to construct accurate statistical models. In contrast, the approach proposed in this thesis enables the explicit encoding of designer knowledge of the dynamics into the classification (such as recognizing that measurements of elevated altitude are indicative of flight), which reduces the need for large training sets. In other words, this approach is simpler to implement while providing all necessary functionality for proving the concept of context identification. Research into adapting leading classification techniques to

dynamical systems such as mobile robots is left for future work.

3.1.2 Impact on MM state estimation

MM systems estimate the accuracy of each filter from the likelihood of its output, and weigh all individual outputs according to their accuracy to generate consolidated state estimates. As stated in Section 2.3.3, the accuracy of these consolidated estimates decreases when the individual filters' likelihoods are estimated incorrectly. Fortunately, identifying the current dynamical context provides information about model accuracy independently from filter likelihoods. This means that context identification can help MM filters assign appropriate weights to the individual estimates, even if their likelihood estimates are incorrect.

Assigning appropriate weights to individual filters can be done by modifying the parameters of the MM algorithm online. For the GPB2 described in Algorithm 2 on page 34, (2.5) shows that the weights $Prob_{i,j}$ are a function of the transition probabilities $T_{i,j}$, the output likelihoods $p_{i,j}$ and the previous weights $Prob_j$. Of these terms, only $T_{i,j}$ is a parameter set by the designer; it can therefore be modified to affect the desired changes in $Prob_{i,j}$. If the system operation is classified as being in context i, then model i and its corresponding filter are known to be appropriate and the other filters inappropriate, as different models are assumed to describe distinct dynamics. Thus, assigning non-zero weights to the other filters incorporates knowingly inaccurate estimates to the consolidated output, which decreases its accuracy. This thesis avoids this problem by setting the transition probabilities from any filter to filter i equal to one, and to all other filters equal to zero. This leads to the maximal weight of one being assigned to the output of filter i, and exactly zero to all other filters. Thus, by modifying $T_{i,j}$ based on which dynamical context is identified, the GPB2 is essentially reduced to a single-filter system, where the filter corresponds to the current dynamical context.

Before examining the technical details of this approach in Section 3.1.3, it helps to see how it would apply to the bouncing ball problem. The task is to estimate

the ball's height by using flight and stance models, and noisy data provided by the height sensor. This is a two-context system, so a GPB2 would spawn four filters, along the lines of Figure 2.2, and weigh the output of these filters. At each iteration, the weighted outputs can be consolidated into a best estimate of the state. If the height measurements are higher than the ball's rest height, they would be classified into the flight context, and the context-based GPB2 transition probabilities modified so that transitioning into the flight filter has a probability of one. This way, the overall state estimate is that of the flight filter, whose model correctly represents the dynamics of the ball while airborne. If the GPB2 was left to its own device and computed non-zero weights to the outputs of the other filters, then the contribution of these outputs to the consolidated estimate would necessarily reduce its accuracy. The results of this approach are provided in Chapter 4.

It is worth noting at this point that since the context-based GPB2 relies on a single filter when the context is identified, all other filters can be deactivated without loss of accuracy. Without context identification, MM filters need to run the entire bank of filters in order to generate combined estimates. Here, when the dynamics change, the contexts switch accordingly, which triggers the activation of a new filter and deactivation of an old one. This reduces the computational overhead of multiple model filters and enhances their scalability.

Naturally, this approach would lead to erroneous results if the wrong dynamical context is identified. This risk is mitigated in three ways. First, during the construction of context-specific sets of measurements in sensor space, care should be taken to ensure that sets which correspond to different contexts are distinct from one another. To that effect, the designer would construct the sets from sensor data that are as relevant to the dynamics as possible, so that measurements affected by different dynamics would naturally cluster in distinct sets. Having distinct sets reduces the risk of misclassifying measurements that are corrupted by sensor noise. Second, robustness to sensor noise is further enhanced by constructing the sets from data generated by multiple sensors, each based on different physical principles (fol-

lowing the guidelines of Scheding [66]). Such redundancy prevents accidental faults in one sensor from leading to context misidentification. Last, the risk of misidentification is further reduced by identifying the system's behavior and verifying that the identified dynamical context belongs to that behavior. This can be done through the identification of the behavioral context, which is detailed in Section 3.2. This verification step enables estimation systems to reject false positives generated by sensor data classification.

3.1.3 Technical Approach

Context information is incorporated into the GPB2 by changing the transition probabilities ($T_{i,j}$) in Algorithm 2 as a function of the mode. If mode i corresponds to the identified context and mode j represents all other modes, then set $T_{i,j} = T_{i,i} = 1$ and $T_{j,i} = T_{j,j} = 0$. This means that transitioning into the identified mode and staying in it has a probability of one, and transitioning into a wrong mode and staying in it has a probability of zero. As expected, this produces $Prob_i = 1$ and $Prob_j = 0$.

As a consequence of this strategy, the best state estimate generated by the GPB2 is equal to the state estimate of the accurate individual filter; i.e. $z = z_i = z_{i,i}$. This formalizes the observation that once the system is in the identified mode, it is expected to remain in it until a change of context is detected. In other words, the only valid hypothesis is (i_t, i_{t-1}), and the output of the filter corresponding to mode i constitutes the sole output of the GPB2. By ignoring the contribution of inaccurate mode states, the accuracy of the consolidated state is not decreased unnecessarily. Significantly, ignored filters do not have to be activated as they no longer impact state estimates. This can be done simply by setting $z = z_i = z_{i,i}$, thereby explicitly ignoring inaccurate filters. Thus, when the dynamical context is identified, multiple model filters are effectively reduced to a single-model filter, where the identity of the model is specified by the dynamical context.

In situations where the dynamical context cannot be identified, such as during

transitions between dynamical contexts within a single behavioral context, the transition probabilities are restored to their nominal value, all filters are reactivated, and the GPB2 resumes normal operation.

Algorithm 3 gives a step-by-step description of the procedure for the two-mode problem, with F and S standing for flight and stance modes, respectively. Line 2 of the algorithm corresponds to situations where the dynamical context, and hence the mode, is identified. Transitions to that mode are set to one, and transitions out of the mode to zero. The GPB2 output is now strictly the output z_{ii} of the individual $filter_i\,(z_i)$ corresponding to the identified mode.

Line 7 corresponds to the situation where the dynamical mode cannot be identified. The individual filter variables (states and covariances) are reset to the last estimates to properly initialize the nominal operation of the GPB2. As for the transition probabilities, they are now a function of the state of the system, an example of which is provided in Chapter 5. Setting transition probabilities as a function of state improves the accuracy of the GPB2, as it encodes information about the dynamics that the GPB2 algorithm would be unable to capture from its model set.

Lines 17 through 25 are the statements that perform the selective activation of the appropriate filters.

3.1.4 Consequence of inability to identify dynamical contexts

So far, this approach ensures that, whenever the system's dynamical context is identified, the context-based MM estimator only trusts the model that corresponds to the context. What happens, then, if the classification fails to identify any context? The answer depends on the cause of the failure, which can be one of two possibilities. The first is that the system is transitioning from one context to another (e.g. touching down or lifting off, for the jogging ball), and in the process exhibiting transient dynamics that are not appropriately represented by any of the available models. As the system transitions between contexts, one filter gradually loses accuracy while another gains accuracy. Therefore, a reasonably accurate state

Algorithm 3 GPB2 Modifications

```
1.  for i ∈{F,S}, j = ī {
2.    if (identified mode = i) {
3.      T_{i,j} = T_{i,i} = 1
4.      T_{j,i} = T_{j,j} = 0
5.      z = z_i = z_{ii}
6.      flag_i = T, flag_j = F
      }
    }
7.  if(no mode identified){
8.    for i ∈{F,S}, j = ī{
9.      if{flag_i = F}{
10.       z_i = z_{jj}
11.       P_i = P_{jj}
        }
      }
12.   flag_F = flag_S = T
13.   if(ż < 0){i = S; j = F}
14.   else{i = F; j = S}
15.   T_{i,j} = 0.7; T_{j,j} = 0.3
16.   T_{i,i} = 1; T_{j,i} = 0
    }
17. for i ∈{F,S}, j = ī{
18.   if(flag_i){
19.     update filter_i(z_i)
20.     if(flag_j){
21.       update filter_i(z_j)
        }
      }
22.   if(flag_j){
23.     update filter_j(z_j)
24.     if(flag_i){
25.       update filter_j(z_i)
        }
      }
    }

    Execute Algorithm 2
    Iterate
```

estimate would be formed by consolidating estimates of these filters weighted by their relative likelihood, which can be done with the GPB2. Thus, during context transitions, the strategy is to restore the transition probabilities to their *a priori* values, activate all filters and allow the GPB2 to recover its full functionality.

The second possibility is that the dynamics are not transitioning between two contexts, but rather have changed entirely and are no longer described by the models at hand. An example would be a mobile robot that executes a specific gait and suddenly collides with an obstacle. The large dynamics that result from the collision are likely to cause the robot to exit all its dynamical contexts. Therefore, the estimation strategy here would be to avoid using any of the available models, since relying on knowingly inaccurate models can lead to the failure of state estimation. The challenge raised here is how to disambiguate between the two possibilities, namely whether the departure from dynamical contexts is merely due to a transition among contexts or rather to a fundamental change of the dynamics. This challenge can be addressed by identifying the behavioral context, as discussed next.

3.2 Behavioral Contexts

Beyond detecting whether a system is in a particular dynamical context, it is important to understand whether the dynamics are transitioning the robot from one context to another, in order to decide which MM system to use. This can be done by constructing *behavioral contexts* to represent specific sequences and frequencies of transition among specific dynamical contexts. Different behavioral contexts may represent different dynamical contexts, or the same contexts but with different transition sequences or rates. Similarly to the dynamical context case, behavioral contexts can be identified from classification of sensor data. If a system's behavioral context is identified, then the dynamics are expected to induce transitions among specific dynamical contexts, and a multiple model estimator can be used to estimate the state during context transitions. Conversely, if neither the dynamical nor the behavioral context can be identified, then the robot is operating

in an unknown (and hence unmodeled) mode and estimation should not rely on any of the available models.

To illustrate this concept, consider a hybrid system with two dynamical contexts, D1 and D2. If the system alternates these dynamical contexts at a specific rates, then a behavioral context B represents D1 and D2 and the rate of transition between them. This way, whenever the context B is identified, a multiple model system can accurately rely on the combination of models for the D1 and D2 contexts, even during transitions when the dynamical context cannot be identified. If the system were to be subjected to significant disturbances, the behavioral context B would no longer be identified, and the estimation system would appropriately avoid using the D1-D2 filters. Thus, behavioral contexts enable communication of high-level information to the estimation system. Without this tool, the D1-D2 filtering system would be used all the time, even when the models do not represent the dynamics, with the consequence of generating inaccurate estimates.

This approach applies principally to hybrid systems with cyclical dynamics. The repetitive nature of the dynamics enables behavioral contexts to identify the specific dynamical contexts associated to each behavior. Cyclical dynamics also induce structure in the sensor signal, and this structure can be exploited to identify the behavioral context itself.

More precisely, hybrid locomotion dynamics induce spatial and temporal structure in the signal generated by onboard sensors. In a sense, the structure of the signal can be considered as a signature of the dynamics, with different structures assumed to correspond to different dynamic behaviors. This suggests that a reasonable approach to identifying behavioral contexts is to recognize specific structures in the signal and relate them to the dynamics that induced them and to their associated behavioral context. This approach is similar to the identification of dynamical contexts in that in both cases, information about the dynamics is extracted from sensor data. However, the approaches differ in the type of information required. Dynamical contexts can be identified through the classification of instantaneous

measurements; i.e. they do not require the accumulation of data over time. In contrast, behavioral contexts represent a sequence of dynamical contexts and therefore require the observation of sensor data over time in order to detect these sequences.

To better understand this concept, consider again the hybrid system with the dynamical contexts D1 and D2. Following the approach described earlier, both contexts can be identified by classifying data generated by onboard sensors into sets of measurements that correspond to D1 and D2. Now in order to identify the behavioral context B, the system should transition from D1 to D2 and back. Thus, context B would be identified if the measurements follow a pattern that oscillates between the D1 and D2 measurement sets. This pattern constitutes the spatial structure of the signal.

To increase the accuracy of the identification, the temporal component can be taken into consideration, by measuring the amount of time the system dwells in each of the dynamical contexts. The accuracy of the identification can be increased by verifying that the dwelling time falls within a specific range determined in a laboratory environment. Temporal analysis can lead to another level of precision; it could differentiate between behavioral contexts that share the same spatial structure but can be disambiguated through their temporal structure. For instance, if the system of interest is a legged robot, the jogging and running contexts generate the same spatial structure as both alternate flight and stance contexts, but they do so at different rates,. This leads to distinct periods of oscillation that can be detected through temporal analysis. Thus, the amount of information required for identifying a behavioral context is larger than what is necessary for dynamical contexts, and the classification technique is correspondingly more elaborate.

The details of this technique are presented in the following three subsections, which describe the conceptual and technical approaches and provide a brief overview of related work.

3.2.1 Conceptual approach

For robotics systems, behavioral contexts correspond to locomotion dynamics that induce switching among multiple dynamical contexts. In other words, behavioral contexts represent dynamics that are described by more than one model, with individual models gaining and losing accuracy in a specific sequence and at specific rates. This definition means that behavioral contexts should not be equated with the behaviors themselves; rather, they identify the specific dynamics of a behavior that can be represented with the available models. The same behavior may exhibit other dynamics that are not modeled, and these dynamics would not be part of the behavioral context.

To recognize the dynamics that are modeled and thereby identify a system's behavioral context, this thesis combines probabilistic and deterministic discrete-state estimation methods. The probabilistic component, based on hidden Markov models, seeks to recognize *spatial* structure in a stream of sensor measurements; i.e. it extracts symbols from the data stream, some of which corresponding to dynamical contexts, and tracks these symbols over time to identify the sequence in which they occur. The deterministic component, based on timed automata, recognizes the data stream's *temporal* structure by capturing the rate of transition among the symbols. Thus, by identifying the sequence and rate of transition among symbols in a data stream, this approach determines the sequence and rate of transition among dynamical contexts, which enables the identification of the behavioral context.

More precisely, the *spatial* structure of a stream of data points is defined as the sequence of transition among salient points, or symbols, in that stream, and the *temporal* structure is defined as the rate of transition among those symbols. The assumption here is that different dynamics induce distinct spatial and/or temporal structure in sensor data, so that recognizing the structure identifies the dynamics. This discrete, sequential definition of structure (rather than, say, a spectral definition), enables the designer to choose symbols that correspond to dynamical contexts, which enables the simultaneous identification of the behavioral and dy-

namical contexts. Naturally, since this approach requires the processing of sensor data over time, it does not enable the identification of dynamical contexts at a bandwidth comparable to sensors update rate. However, processing more sensor data means that more information is used for the identification, which increases its robustness to sensor noise. Therefore, this method is a compromise between bandwidth and accuracy, and can be used in conjunction with straight-forward classification to reduce the risk of false positives and negatives.

The other advantage of defining structure as a sequence of symbols is that it simplifies the modeling and identification of the structure. For instance, the sequential nature of the symbols can be captured with a Markov chain discrete-state model that represents an underlying process which outputs these symbols. Such models are simple means of representing specific sequences of symbols, and form the basis for discrete-state estimation techniques to match modeled sequences with observed sequences. If the matching is positive, then the observed structure is recognized and labeled after the model. With this setup, each behavior would be represented with one discrete-state model, so that recognizing the structure in a data stream enables the identification of the behavioral context.

The discrete-state estimation technique chosen to identify the structure is the hidden Markov model (HMM). This choice is motivated by the effectiveness of HMMs at estimating discrete states while maintaining a low computational overhead. This advantage makes HMMs the estimation tool of choice for many applications, such as text parsing and speech recognition. The particular implementation of HMMs presented here is designed to detect whether the observed symbols occur in the sequence predicted by the discrete-state models.

However, the Markov assumption underlying the estimation algorithm prevents the tracking of symbols over time, so the context-based framework complements HMMs with timed automata to overcome this restriction. As their name suggests, timed automata capture the temporal component of a system, so the combination of HMMs and timed automata enable the identification of the spatial and temporal

structure in a signal, and therefore of the behavioral context.

The technical implementation is described next, starting with a discussion of related work.

3.2.2 A brief overview of related work

As noted above, the proposed approach to identifying spatial and temporal structure of sensor data uses a combination of HMMs and timed automata. This section describes these techniques and draws parallels between the problem of identifying behavioral contexts and the problem of recognizing speech, since both problems are addressed with similar approaches. It is worth noting that hidden Markov models are generative processes, as the models describe a process that generates discrete observations which are matched to measured observations, in the same way Kalman filters match predicted states to observed states. It follows that the spatial structure of a signal is not recognized by identifying a specific sequence of symbols per say, but by identifying a specific sequence of HMM states that would generate the expected symbol sequence. This subtle but important distinction enables the differentiation between two identical symbols that are generated by different states; therefore, this property enables the identification of dynamical contexts even when classes overlap.

3.2.2.1 Hidden Markov models

An HMM is a probabilistic graphical model that undergoes transitions among its N states and generates discrete observations. A graphical depiction of a generic HMM is provided in Figure 3.2. State estimation for an HMM is preformed by computing a probability distribution $\alpha_k(i)$ over its states i, which expresses the probability of the HMM process being in each of the states (with $\sum_{i=1}^{N} \alpha_k(i) = 1$) at step k. In broad terms, the probability distribution is computed by first setting a transition probability among different states; second setting the probability of

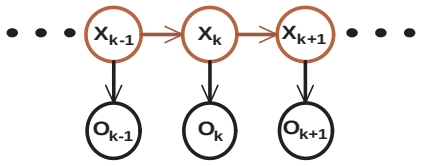


Figure 3.2: Hidden Markov models transition among their discrete states and generate discrete observations.

generating observation symbols when the system is in each state; and last matching the generated (predicted) symbols with the measured symbols to infer $\alpha_k(i)$. More precisely, the transition probabilities are described by a_{ij}, expressing the probability of transitioning from state i at step $k-1$ to state j at step k. These probabilities are set *a priori* by the designer and are the discrete-state equivalent of the prediction step in a Kalman filter (the State Prediction equation in Algorithm 1 on page 26). The observation probabilities are described by $b_j(o_k)$, the probability of generating the observation symbol o_k if the process is in state j at step k. Observation probabilities are also set by the designer and are somewhat similar to Kalman's observation matrix. Note that transition and observation probabilities can be tuned manually, as in this thesis, or learned form labeled data [59]. In a way, the combination of transition and observation probabilities enables the HMM to predict the probability distribution over the states at every step and the likelihood of generating each observation symbol. These predictions are then compared to symbols extracted from sensor data and the result is used to update the distribution over the states, thus achieving a filtering process akin to the update stage of the Kalman filter. Much like continuous filters and observers, this prediction/observation/update sequence provides HMMs with robustness to sensor noise.

The implementation of the approach described above takes the form of the *forward* algorithm. At first, the designer parametrizes an initial probability distribution over the HMM states $\pi_i = \alpha_1(i)$, where i is the state index, and specifies the state transition and the observation probabilities, a_{ij} and $b_j(o_k)$, respectively. Then the

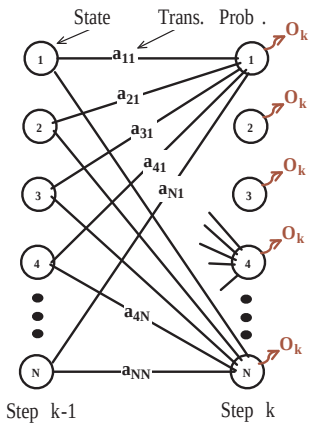


Figure 3.3: Graphical representation of transitions among N HMM states.

state probability distribution α_k is estimated in a process called inference, which consist of the following recursive steps [59]:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1), \tag{3.1}$$

2. Induction:

$$\alpha_{k+1}(j) = \frac{\left[\sum_{j=1}^{N} \alpha_k(i) \cdot a_{ij}\right] b_j\left(o_{k+1}\right)}{\sum_{i=1}^{N} \left\{\left[\sum_{j=1}^{N} \alpha_k(i) \cdot a_{ij}\right] b_j\left(o_{k+1}\right)\right\}}. \tag{3.2}$$

As expected, the probabilistic nature of HMMs provides robustness to sensor noise, as current probability distributions (Equation 3.2) mix observation probabilities $b_j(o_t)$ with model predictions $\sum_{j=1}^{N} \alpha_k(i) \cdot a_{ij}$.

Equation 3.2 also illustrates the Markov assumption of conditional independence, whereby current distributions depend only on the distribution at the previous step and on the current observation, as illustrated in Figure 3.3. This leads to a light computational cost of $O\left(N^2\right)$, but prevents the explicit modeling of the system's duration in a state or rate of transition among states.

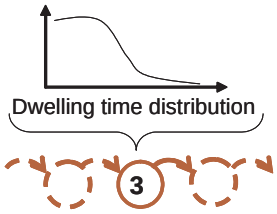


Figure 3.4: Semi-Markov processes explicitly model in-state dwelling time.

Since the goal is to capture the temporal component of the structure in addition to its spatial component, it is worth examining extensions to the HMM that accommodate time. A well known extension are semi-Markov processes (SMPs), which capture time by incorporating a duration model to the computation of transition probabilities ($a_{ij} = f(duration)$) [56]. Unfortunately, SMPs have a high computational cost so they are rarely used for estimation [37], and attempts at lowering the cost come at the expense of significantly increased model complexity as compared to HMMs [10, 87]. Therefore, HMMs are used instead of SMPs, even though they only enable the extraction of the spatial structure of a signal.

Since the spatial structure consists of a sequence of HMM state estimates, it should be possible to analyze that sequence and infer the temporal component of the structure. Such analysis can be performed with timed automata, presented next.

3.2.2.2 Finite state automata and timed automata

A finite state automaton is a deterministic graphical model that undergoes input-triggered transitions among its states and generates discrete observations [35]. The non-probabilistic nature of FSA makes them brittle in the presence of input noise, and therefore are inappropriate tools for estimating the state of systems using unfiltered sensor data. However, FSA enable low-cost state estimation if the input is not corrupted by noise. In addition, they can easily incorporate mechanisms to measure the time separating consecutive inputs and in-state dwelling time. This configuration is referred to as timed automata.

The task at hand is to analyze the sequence of HMM state estimates to recover temporal information; these states can be considered as noise-free input to a timed automaton, which can then extract the temporal structure of the original signal. Here, it is important to stress that timed automata and HMMs have different models and states. HMMs model a process that can generate symbols extracted from a stream of sensor data, whereas timed automata model the sequence and dwelling time of the HMM output. This distinction will be made clear when the technical approach is described in Section 3.2.3 and even more so when the framework is implemented in Chapter 4.

Thus far, it seems reasonable to expect that this two-step estimation process would enable the identification of the spatial and temporal components of a signal, and experimental results demonstrate that this is indeed the case. The merits of this approach are underlined by the fact that similar two-step, HMM-based processes are used in the speech recognition community [37].

3.2.2.3 Speech Recognition

Much like the problem of context identification, speech recognition seeks to identify the source of a sound from the structure of the signal. A typical technique called postprocessor duration modeling treats the spectral and temporal modeling as two separate, loosely connected problems [29]. The duration in each state is modeled with, say, a Gaussian distribution, and constrained by an upper bound. Spectral estimation is performed with HMMs and the duration is computed in a second step from the spectral estimates. The probability distribution of these estimates is then re-computed, this time taking into account the duration probability [37].

In a similar way, the approach proposed in this thesis proceeds in two steps, inferring duration from spatial structure and imposing upper bounds on it, as will be shown in the next section. However, it does not explicitly model the duration to simplify this initial implementation. Nevertheless, the timed automata approach is amenable to duration modeling, along the line advocated by the speech recog-

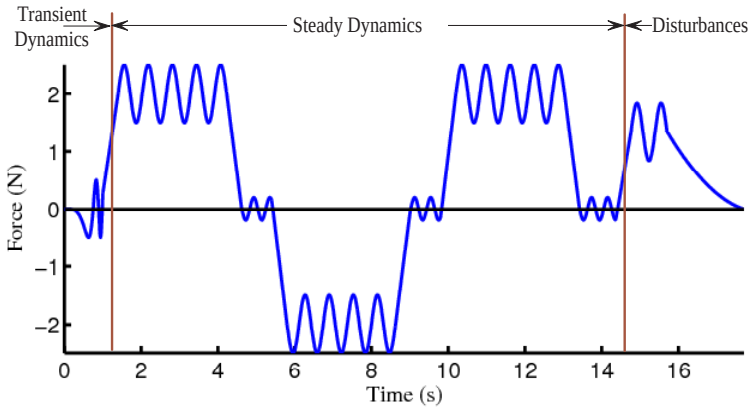
nition community. Naturally, HMMs are also amenable to the learning techniques used in speech training. Therefore, there seems to be potential for adapting speech technologies to the problem of context identification.

3.2.3 Technical approach

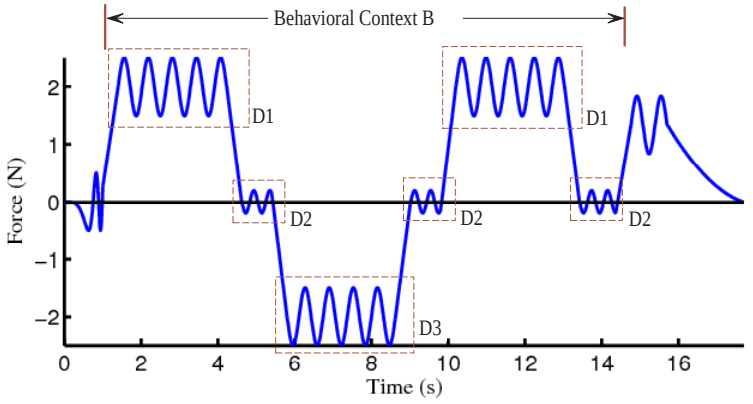
This section clarifies the approach to behavioral context identification, introduced so far at a high level, by describing its implementation with the help of the conceptual hybrid system introduced earlier. Here, the system is made somewhat more complex by assuming that it has three, not two, dynamical contexts, D1, D2 and D3. Suppose that the system's interactions with the environment result in the force profile of Figure 3.5(a), which could be measured via onboard force sensors. At the beginning, startup transients dominate the dynamics; then steady-state operation produces distinctive patterns in the dynamics; and the operation ends when dynamic disturbances break the steady-state pattern and the system comes to a halt.

An examination of the force profile suggests that the steady-state dynamics could be described with a collection of three simple sinusoidal models, one for the top, one for the middle, and one for the bottom regions of the dynamics. Assuming the existence of such models, the dynamics can be described with the three dynamical contexts D1, D2 and D3, as in subfigure (b). In addition, a behavioral context B can be defined to represent a sequence of transition among the three dynamical contexts that is specific to the stead-state region. Thus, as long as D1, D2 and D3 follow each other in the expected sequence, context B is identified, and when disturbances modify that sequence, the system is no longer in context B.

The description of the identification approach is carried step-by-step in the following subsections.



(a) The system exhibits a sequence of transient, steady-state, and disturbance dynamics.



(b) The behavioral context B correpsonds to a specific sequence of dynamical contexts D1, D2 and D3.

Figure 3.5: Force profile of a conceptual hybrid system.

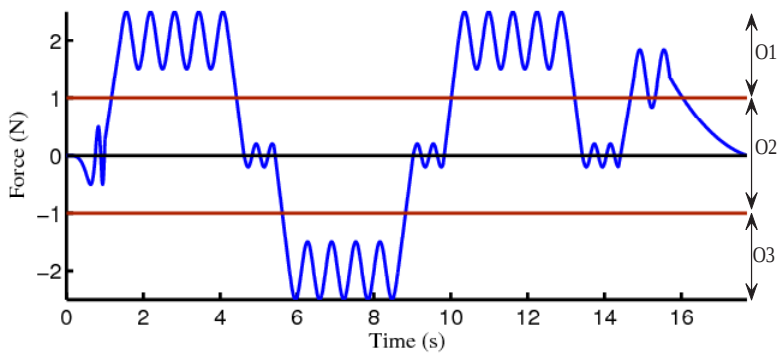


Figure 3.6: Continuous measurements of force are discretized into observation symbols O1, O2 and O3.

3.2.3.1 Hidden Markov Models

The first step in constructing the framework is to discretize the continuous force measurements into symbols that will appear recurrently when the system is in context B. Figure 3.6 shows that such sensor symbols can be O1, O2 and O3, roughly corresponding to the top, middle and bottom regions of the force profile, respectively.

The second step is to build a Markov chain model of a process that could generate these sensor symbols. A first model may contain the states H, M and L, representing high, medium and low forces and generating the symbols O1, O2 and O3, respectively (see Figure 3.7(a)). However, a single M state does not allow the differentiation between situations where the system transitions from high to low and low to high forces.

This limitation is addressed by replacing the single state M with two states M1 and M2, where M1 corresponds to transitions from high to low forces, and M2 corresponds to transitions from low to high (Figure 3.7(b)). With this setup, the same symbol O2 is generated by two different states M1 and M2. Disambiguating these two states is made possible by tracking the sequence of sensor symbols over time. If O2 appears after O1, then the system is in M1, and if it appears after O3, then it is in M2. This is an example of how processing information over time,

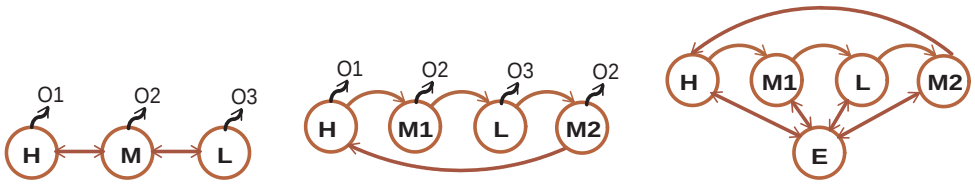
as opposed to instantaneously (as with dynamical context identification), helps distinguish between two HMM states. If the HMM states are chosen to correspond to dynamical contexts (i.e. H corresponds to D1, M1 and M2 to D2 and L to D3), then this system would identify these contexts even if their corresponding classes overlap.

The third step is to specify HMM parameters as per Section 3.2.2.1, with the transition and observation probabilities a_{ij} and $b_j(o)$ designed as to predict observation symbols in the expected sequence. For example, $a_{M2|L} = 1$, and $a_{M2|j} = 0, \forall j \neq L$. This enables the HMM to run the forward algorithm (Equation 3.2), process the symbols o extracted from data discretization and infer the probability distribution α over the states.

When the system is operating in steady state, the observed symbol sequence is expected to match the sequence predicted by the model. Likewise, the sequence of HMM states[2] is expected to match the state transitions described by the model. When the dynamics vary from steady state, the sequence of symbols also varies from the predictions and leads to out-of-order state sequences. Therefore, verifying the order of the state sequence helps recognize the behavioral context.

In order to enable the explicit detection of out-of-order transitions, the HMM model is augmented with an error state E, as in Figure 3.7(c). The error state has low-probability, two-way transitions to all states and all observations have a uniform distribution over it. In other words, the error state is equally likely to generate all sensor symbols. The observation probabilities are designed such that the probability of observing a symbol conditioned on a state that should not generate that symbol is lower than the symbol's probability conditioned on the error state. This means that the likelihood of generating a specific symbol by the error state is greater than the probability of generating that same symbol by a state that should not generate it. For example, $b_E(O3) > b_H(O3)$, where E and H are

[2]In this thesis, the sequence of HMM states is defined as the sequence of most likely states estimates by (3.2). At each step, the most likely state is $\text{argmax}_{1<i<N}(\alpha_i)$.



(a) (b) (c)

Figure 3.7: HMM models state transitions and output generation for a simple system.

the error and high-force states, respectively. More generally, $b_E(o_{k+1}) > b_j(o_{k+1})$ if o_{t+1} is different from the symbol generated by state j, where j is an index among the states.

This property ensures that if the wrong sequence of sensor symbols are observed, then the HMM would transition to the error state. For example, assume that the model predicted that the system would transition to state j, but the observed symbol cannot be generated by j. This is an indication that the state sequence is out of order, so the observation probabilities ensure that $\alpha_E > \alpha_j$. In other words, the HMM assigns the highest likelihood to the error state E when the sequence of states is out of order. Thus, whenever the system is in E, the system is not expected to be in the behavioral context that corresponds to the HMM model (context B, in this case).

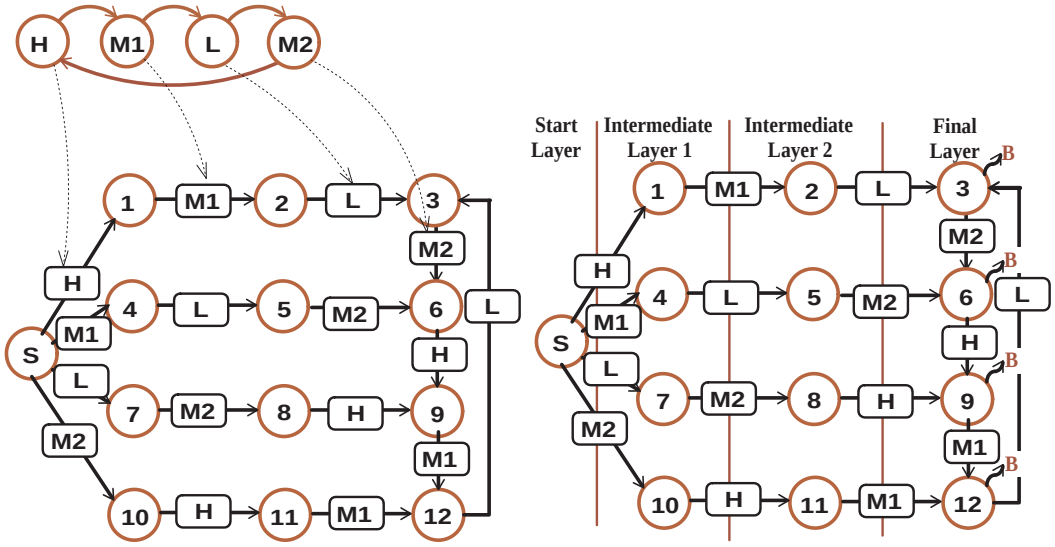
3.2.3.2 Finite State Automata

Detecting a single out-of-order transition, such as the one described in the previous section, is sufficient to recognize that the system is not operating in the expected behavioral context. However, behavioral contexts cannot be positively identified from observing a single transition; instead, the identification requires the observation of a specific sequence of transitions among HMM states. Therefore, the estimation system needs to observe several "good" (i.e. in-order) consecutive transitions to confidently identify the behavioral context.

Unfortunately, the HMM cannot track a sequence of transitions for longer than one time step, because of the Markov assumption. This motivates the need for a mechanism to track transitions over multiple steps and perform bookkeeping. One such a mechanism is a finite state automaton, a tool commonly used to estimate the state of deterministic discrete systems. For the purpose of tracking the HMM states over time, a timed automaton can treat the HMM states as inputs that trigger transitions among FSA states. Figure 3.8 shows an example where the FSA states are represented with circles, and the inputs are represented with rectangles. The FSA state structure is organized in p layers designed to recognize a correct sequence of inputs over p steps. Layers are defined as follows: the first layer contains the starting state; the final layer contains all the state reached after p numbers of correct transitions; and intermediate layers contain states reached after a number n of correct transitions, with $n < p$.

To see how this FSA can identify p successful transitions, assume that the initial input event transitions the automaton from the starting state S to a target state in the first intermediate layer. Consecutive occurrences of the same input cause self transitions, but new inputs that are in order induce transitions from one intermediate layer to the next. After p correct transitions, the automaton reaches the final layer and outputs a success flag identifying the behavioral context ('B' in this case). Once in the final layer, the system can only transition among states that are in the final layer, unless the inputs are out of order. In other words, new in-order inputs maintain the automaton in the final layer and a flag is issued at each state transition. At any time, if an out-of-order input occurs, the automaton is reset, and a new sequence of p correct transitions is needed to recognize the behavioral context. In summary, success flags indicate that the HMM outputs follow the expected sequence, thereby recognizing the spatial structure of the signal and identifying the behavioral context.

For this example, the numerical value of $p = 3$ is a compromise between efficiency and robustness, as increasing p reduces false positives but delays context identification. It is worth noting that no delay is involved in recognizing that the behavioral



(a) HMM output states serve as FSA input

(b) Finite State Automaton

Figure 3.8: HMM output such as H, M1, L, and M2 from Figure 3.7 serve as input (rectangles) that trigger transitions between FSA states (circles), as shown in subfigure (a). The structure of the FSA is organized in p layers in order track input sequences over p steps, as shown in subfigure (b). The output flag 'B' indicates that context B is identified. Reset transitions back to S are omitted for clarity.

context cannot be identified, an important consideration for avoiding the use of inappropriate models.

3.2.3.3 Timed Automata

The FSA model designed to recognize the spatial structure of the signal can also be used to recognize the temporal structure. The temporal analysis mechanism turns the FSA into a timed automaton that can capture some temporal information by way of measuring the delays separating consecutive inputs. These delays are measured by a clock reset each time the automaton enters a new state. Since HMM states correspond to automaton inputs, the automaton simultaneously measures its own and the HMM state duration.

Temporal information can be used as a timeout that triggers a transition out of

a state if the duration exceeds a predefined bound. It also enables time-sensitive transitions, whereby the target state is selected as a function of both the input and the duration in the previous state. This is useful, for example, to differentiate between two data sets that share the same spatial structure but have different temporal structures.

3.2.3.4 Discussion

The choice of these techniques is the result of a compromise between demonstrating the applicability of signal processing techniques to dynamical systems and containing the complexity of the initial implementation. For example, model learning and adaptation techniques are left for future work, as they are not essential to demonstrating the effectiveness of classification at identifying context. Also, the HMM output could be estimated more accurately by using the Viterbi algorithm, but the online version of the algorithm is slow to converge [49], so the forward algorithm is used instead.

The duration bounds of the automaton states are manually computed from frequency of occurrence counts. Using bounds would seem to lead to brittleness, but selecting conservative values improves robustness to limited temporal variations, as evidenced by their successful use in speech recognition and in the experimental results presented in Chapter 4. Hard bounds are also desirable for the application at hand, as they provide appropriate sensitivity to large temporal variations generated when the system leaves a particular behavioral context.

It is worth recalling that identifying the behavioral context is not the same as recognizing the behavior itself, as behaviors generally exhibit a continuum of dynamics, some of which possibly not described by available models. Rather, context identification is a technique to determine that the current dynamics can be approximated with specific models.

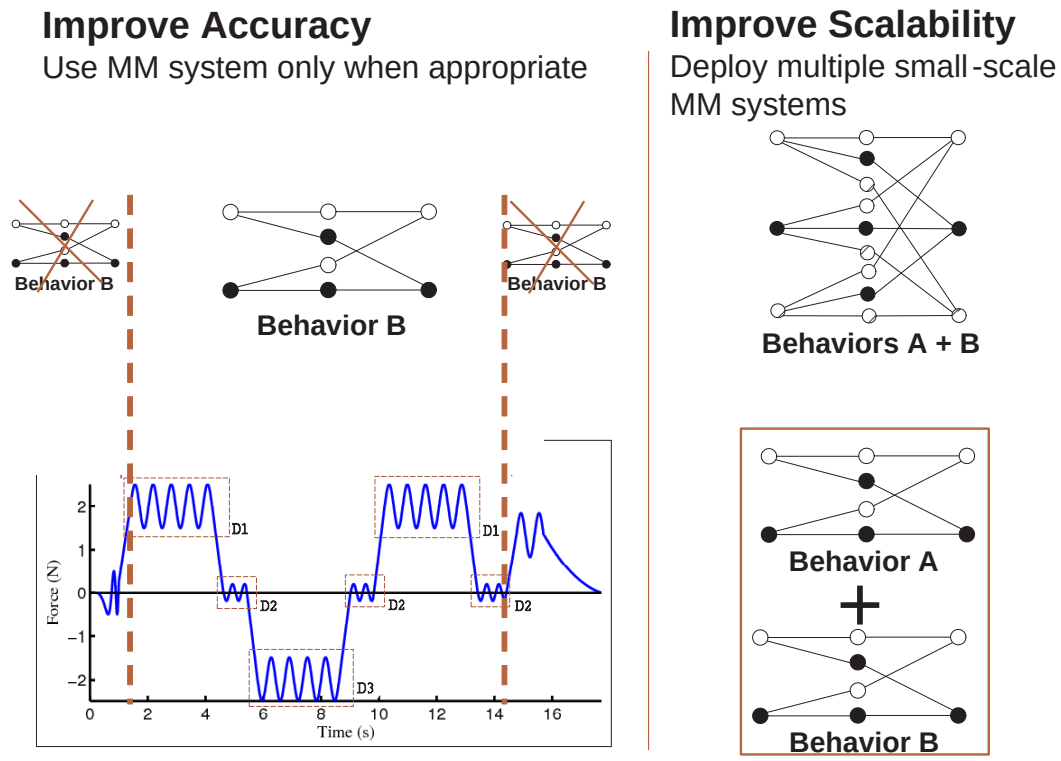


Figure 3.9: The identification of a system's behavioral context enables the deployment of multiple-model filters only when appropriate, which in creases estimation accuracy. Behavioral context identification also increases estimation scalability by replacing large-scale MM systems with a collection of small-scale systems.

3.2.4 Impact on MM estimation

Whereas dynamical contexts determine which *individual* model is accurate enough for state estimation, behavioral contexts identify which model *set* should be used by multiple-model filters. In the example above, an MM system based on the three models D1, D2 and D3 can be used as long as the system is known to be in context B, i.e. in steady-state operation. At the beginning and at the end of the operation, where disturbances dominate the dynamics, the estimation system should not use the three-model set, thereby avoiding knowingly inappropriate models and reducing the risk of estimation divergence. This is explained graphically in Figure 3.9.

A related advantage provided by the identification of behavioral contexts is that it significantly improves the scalability of MM systems, as Figure 3.9 also illustrates. The fact that behavioral contexts correspond to specific dynamical contexts enables the deployment of a collection of limited-scale MM systems, each containing only filters that are specific to one behavioral context. This is in contrast to conventional multiple model estimation that can only be implemented with a single, large-scale MM system formed of all available filters. In other words, if a system has six dynamical contexts, but the current dynamics only induce transitions among two of them, then only a single two-context GPB2 is activated. When the behavioral context changes, indicating that different dynamics affect another set of dynamical contexts, then a new GPB2 specific to these new contexts is deployed, and the old GPB2 is deactivated. If behavioral contexts were not used, then all filters would have to be activated whenever the system preforms a context transition.

To ground this description in an example, suppose that a physical instantiation of the conceptual system described above is a legged robot capable of walking and jogging gaits. The robot would have a number of dynamical contexts; flight and stance contexts for jogging; a walking context; and start-up and stopping contexts that represent transient dynamics at the beginning and end of each gait. Conventional MM filters would require the activation of a number of filters proportional to the total number of dynamical contexts, which is computationally expensive. In contrast, as the robot executes, say, a jogging gait, behavioral context identification would direct the sequential activation of the start-up filter, then of a flight/stance MM filters, and finally of the stopping filter as the robot comes to a stop, and never activate the walking filter. Since the activation is sequential, the computational cost of state estimation does not necessarily increase as the number of contexts increases, which enhances the scalability of MM filtering schemes.

The example of the walking and jogging robot is used in Chapters 4 and 5 to provide further insight and demonstrate the validity of these approaches through implementation details and experimental results.

3.3 Design Considerations

Context-based estimation combines the conventionally separate fields of filter design and pattern recognition, so constructing a context-based estimation system requires some knowledge of both fields. However, a designer can leverage this thesis and literature in each of the two fields to design filters and pattern recognition tools and combine them for accurate estimation.

Continuous filters have long been integral to most communication and control applications, so the designer can take advantage of a well documented body of work in building continuous models of dynamical systems and in tuning the parameters of filters and observers (see for example [52, 53, 4]). Pattern recognition too has been used and documented over the years, and the designer can refer to machine learning literature for insight into the selection of features and the training of classifiers (see for example [17, 24, 19]).

This thesis provides example implementations of pattern recognition and continuous filtering in Chapters 4 and 5. These examples show that simple approaches to pattern recognition (using HMMs) and to filtering (using KFs) lead to accurate estimates, which suggest that the designer does not need to acquire in-depth knowledge of each technique to generate satisfactory results. HMM-based pattern recognition can be further simplified by adopting parameter learning methods such as (e.g. Baum-Welch or Expectation Modification [59]), where designer input would be necessary to label training data, but not to specify HMM parameters. It is anticipated that learning would also improve the accuracy of the pattern recognition, making it a desirable future extension to the current implementation.

In short, the designer skills needed to construct a context-based estimation system are the same skills needed for conventional filtering and pattern recognition. This thesis provides a framework to combine techniques from both fields, with significant impact on accuracy and scalability but negligible impact on design complexity.

3.4 Summary

This context-based estimation approach can be viewed as a departure from conventional hybrid estimation. The MM filtering approach is based on estimating the continuous state of a hybrid system first and then deriving from it an estimate of the discrete state. This thesis seeks to directly and explicitly estimate the discrete state and use these estimates to make better-informed decisions about continuous estimation. Since discrete-state estimation can be computationally inexpensive, this approach accommodates systems with a large number of discrete states, in contrast to conventional MM filters. In other words, explicitly estimating the discrete state enables a two-step reduction of the computational cost of MM filtering; first, behavioral contexts restrict the size of MM filters to specific dynamical contexts affected by the current dynamics; second, dynamical contexts reduce the number of activate filters in the already small MM system to just one.

The impact of identifying contexts extends beyond state estimation. For robotic applications, contexts provide information about the dynamics and can therefore be used as feedback for robot controllers. Behavioral contexts, in particular, provide information about the aggregate effect of the dynamics on the robot and can therefore help controllers verify that their input does indeed produce the intended behavior. This feedback information is currently binary (a context is either identified or not), but the proposed identification techniques can be extended to provide a continuous confidence measure in the identification (similarly to Lenser's work [40]). Such extensions would be used to measure the 'quality' of a behavior, by estimating its 'distance' from the expected behavioral context. This would enable reactive control, whereby behavior controllers can continuously adjust their parameters to improve the quality of the behavior. Such extensions are further discussed in Chapter 6.

It is noteworthy at this point to compare the terminology of dynamical and behavioral contexts to the more conventional terms of modes and behaviors. The state estimation community refers to hybrid systems as multimodal systems, with the

idea that different modes generate distinct dynamics. The robot-control community uses the term behavior to refer to the aggregate action of mode dynamics on the robot body. This way, a legged robot's jogging behavior is composed of flight and stance modes. This thesis adopts a different terminology to explicitly tie the dynamics to their model, as a means of avoiding the use of inaccurate models. The following example demonstrates the subtle but real difference. When the legged robot is commanded to initiate a jogging gait, it can be said to be executing the jogging behavior. Therefore, one would expect flight/stance filters to accurately estimate its state. However, as with most real systems, the gait's start-up phase is likely to be dominated by dynamic transients that could not be represented by either flight or stance models. This motivates the need for an information processing system such as the behavioral context framework to determine exactly when available models can be used.

Chapter 4

Context Identification

The identification of a mobile robot's dynamical and behavioral contexts enables the selection of appropriate motion models and multiple model systems for accurate and scalable state estimation. This chapter implements the approach introduced in Chapter 3 in an experimental setting and provides results that demonstrate how discrete-state estimation techniques enable rapid and accurate context identification.

Recall that dynamical contexts correspond to locomotion dynamics that are described by a single model, so identifying the current dynamical context specifies which model accurately describes the dynamics. This enables MM systems to rely only on individual filters using accurate models, thus increasing the accuracy of consolidated state estimates. Dynamical contexts can be identified by classifying the data generated by onboard sensors into different classes, each corresponding to a specific dynamical context. This can be done as sensor information becomes available, so dynamical contexts can be identified at a bandwidth similar to that of onboard sensors.

Behavioral contexts are an abstracted representation of robot behaviors, as they correspond to transitions among dynamical contexts induced by the dynamics of different behaviors. Each behavioral context corresponds to a specific sequence and rate of transition among a given set of dynamical contexts. This abstracted

representation enables the deployment of multiple small-scale MM systems, where each MM system is specific to one behavior. The computational cost of running a collection of such systems is lower than the cost of running a single large-scale MM system that models all behaviors. Knowing the current behavioral context allows the selective activation of the appropriate MM system, thus increasing the scalability of multiple model filtering. To ensure rapid context identification, this thesis develops a discrete-state estimation approach to track transitions among dynamical contexts and recognize behavioral contexts as soon as specific sequences and rates of transition are observed.

In contrast to dynamical contexts, the identification of behavioral contexts requires classification techniques that process information over time. As a consequence, the lion's share of this chapter is reserved to the implementation and results of the discrete-state estimation approach to behavioral context identification. The classification approach to dynamical context identification is described more succinctly, although with sufficient details to form a precise understanding of the approach. An in-depth presentation of the results is left to Chapter 5, where the impact of classification on state estimation performance can clearly demonstrate the benefits of the framework.

The technical description of the context identification approaches is grounded in the real-world example of a hybrid system whose complex dynamics complicate the task of estimating its state with conventional techniques. The system is a highly dynamic mobile robot, RHex [64], that operates in multiple behavioral and dynamical contexts. The challenge is to estimate RHex's pose (position and orientation) in six degrees of freedom[1] using low-cost onboard sensors such as accelerometers while the robot executes different behaviors. It is worth emphasizing that the task consists of estimating the robot's state, not in controlling its behavior.

The following sections detail the experimental setup and describe the proposed techniques for context identification. Dynamical contexts are identified through

[1]Note that in some degrees of freedom RHex has a limited range of motion.

a straight-forward classification technique that operates at high-bandwidth. Behavioral contexts are identified through the deployment of hidden Markov models (HMMs) and timed automata. Automata enable the tracking of HMM states over time, which HMMs are unable to do, and HMMs filter sensor noise, which automata are unable to do. This chapter demonstrates that the combination of both techniques enables the robust identification of behavioral contexts, and that the classification approach enables the identification of dynamical contexts at high bandwidth.

4.1 Simple Models for a Multiple-Behavior, Hybrid Dynamics System: RHex

RHex is a six legged robot (Figure 4.1) that can execute a collection of dynamic behaviors, including walking, jogging, running, stair climbing, flipping, jumping and swimming. RHex's versatility stems from two key design characteristics: the first is that the robot configuration actuates each of the six legs independently, thus enabling different leg synchronization schemes that produce different behaviors; the second is that the legs are designed to be compliant, which allows them to store and restore energy in the form of elastic deformation, thus enabling energy-efficient locomotion. Compliant legs are particularly important for producing jogging and running behaviors which alternate flight and stance phases akin to animal running. As the robot touches down after each flight phase, the impact with the ground is efficiently absorbed by the legs, and this energy can be restored at takeoff to help produce a new flight phase.

To form a qualitative understanding of the dynamics involved in such behaviors, the following sections study in detail the dynamics of the jogging and walking gaits for RHex. This description seeks to highlight the challenges of state estimation in the presence of high-order and high-dimension dynamics, and to define the goals of context identification for this example.

Figure 4.1: RHex's configuration features six C-shaped compliant legs that rotate around the hip. Different leg synchronization schemes produce different behaviors, such as walking, jogging and stair climbing.

4.1.1 Dynamical analysis of the jogging behavior

In general terms, jogging behaviors consist of alternating flight and stance phases at a given rate. RHex produces this behavior by synchronizing its legs three by three, producing an alternating tripod gait. Each tripod consists of the middle leg of one side of the robot and the front and rear legs of the other side. In this configuration, the jogging gait is produced by circulating both tripods around the hips, such that when one is in the air, the other is touching the ground, except for a brief period when both tripods are in the air during the flight phase. The rate of recirculation and the fraction of the cycle during which the tripods are in stance over one stride (also known as the duty cycle) are fine tuned to enable a sustained regime of successive flight and stance phases.

The motion induced by the jogging behavior is complex. At first, it may seem that the flight and stance motions affect the robot only in the sagital plane (along the vertical, forward, and pitch axes), since all legs rotate in parallel planes. In practice, the configuration of the legs in a tripod generates motions along the

remaining three axes as well, thus inducing motion along all six dimensions of free space ($\mathbb{SE}(3)$). Recall that the tripod configuration makes the robot touch the ground with one leg on one side and two legs on the other sides. Since all legs have the same spring constant, the robot has effectively a leg stiffness on one side that is double the stiffness on the other side. With any vertical motion of the body, this stiffness differential creates a net moment around the roll axis that leads to a roll motion. This moment also induces left-right motion along the lateral axis, as body roll creates a horizontal component of the leg reaction forces. In addition, the tripod configuration creates a force differential in the forward direction, as the two legs on one side generate twice as much traction as the single leg on the other side. Since the legs have non-zero torsional compliance, this force differential induces rotation around the yaw axis. The direction of motion along the roll, yaw, and lateral axes depends on which tripod is in stance, so alternating the tripods changes that direction back and forth and creates an oscillating motion.

This brief overview shows that motions along some axes affect motions along other axes. This means that the motion dynamics are coupled, and that an accurate motion model of the system's center of gravity would have to span all six dimensions. Such a high-dimensional model is difficult to design, and efforts to date, particularly in the biomechanics community, have made the simplifying assumption that sagital and lateral dynamics can be modeled separately. Such models are a lower-dimension representation of the dynamics, as sagital models typically represent motions along the forward, vertical and pitch axes, and lateral models along the forward, lateral and yaw axes.

In the sagital plane, research demonstrates that the motion of a jogging animal approximates that of a spring-loaded inverted pendulum (SLIP) [68], which suggests that RHex's jogging motion could be modeled as a SLIP. In reality, this model does not accurately represent RHex's motion because its leg motion profiles, determined through tuning, do not match SLIP predictions[2]. As a result, undocumented

[2]Saranli shows that leg motion profiles could be actively controlled in a way that produces SLIP motion [65], an approach called template anchoring. However, this approach requires in-

empirical attempts at using the SLIP model to estimate RHex's state have been unsuccessful. In the horizontal plane, research studying six-legged sprawled animals shows that their lateral motion can be represented with a lateral leg spring (LLS) model [67]. Unfortunately, it is unclear how RHex can be controlled to exhibit LLS dynamics, so this model cannot be used with available gaits.

In the absence of models spanning multiple dimensions, the modeling effort could be further simplified by assuming that the dynamics are decoupled, and constructing independent models along each dimension. This dimensionality reduction limits the overall model accuracy, as argued in Section 2.1.2. However, when considered locally, such models can accurately represent specific dynamics over a limited period of time. The challenge therefore is to detect when each model is appropriate for state estimation, which is achieved through the identification of the robot's dynamical context.

To see how low-dimensional models can represent high-dimensional dynamics, it is necessary to examine the gait's dynamics along each dimension. For clarity, this presentation focuses on the dynamics along the vertical (z) and lateral (x) axes, to show how context identification enables the use of simple models.

4.1.1.1 Dynamics along the z-axis

The acceleration along the z-axis is measured by onboard accelerometers and plotted in Figure 4.2. This plot can be interpreted through analysis of the jogging dynamics. When the robot is in the flight phase, it is governed by the dynamics of a ballistic projectile, where the only force acting on the body is gravity. These dynamics are well understood and are represented with the straight-forward model $\ddot{z} = -g$, where z and g are the vertical state (height) and the gravity acceleration, respectively. The flight dynamics are responsible for the negative accelerations close to $-g$ ($-9.8m/s^2$) in the plot. The dynamics of stance are more complicated because they involve leg deformation. As can be seen in Figure 4.1, the legs are

strumenting the robot with pose and leg force sensors that are unavailable on RHex.

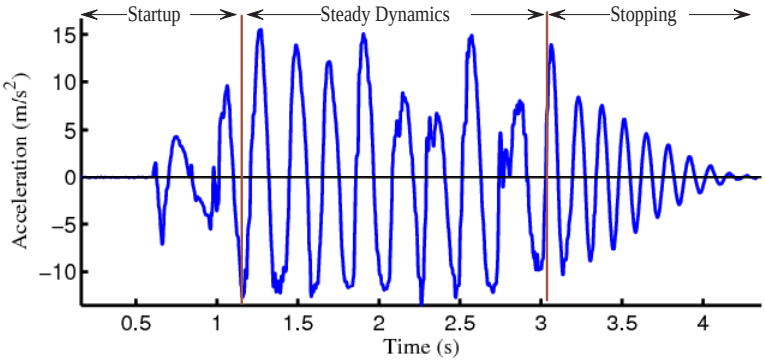


Figure 4.2: Output of vertical accelerometer while RHex executes jogging behavior.

shaped like a "C", so the beam bending theory could be used to model their deformation in the sagital plane. Yet an even simpler model represents the three legs of a tripod with a single linear spring, such that the robot is modeled as a mass-spring system (see Figure 4.3), expressed as $\ddot{z} = -K_z (z - z_0)/M - g$, where K_z is the virtual spring constant, z_0 its rest length and M is the robot mass. The accuracy of both models is demonstrated in Chapter 5, where they are used to generate accurate height estimates.

In summary, the ballistic and spring-mass models describe the two alternating components of the jogging dynamics. To correctly estimate the height of the robot, an MM estimation system needs to determine precisely when each of the models is appropriate, in order to use the ballistic model when the robot is in flight and the mass-spring model when it is in stance. This highlights the importance of rapidly identifying the current dynamical context, as it enables MM systems to select the correct model at high bandwidth.

The flight/stance model pair accurately describes the behavior of the robot as long as it is effectively alternating flight and stance phases. This is the case during steady-state operation, such as between the seconds 1 and 3 in Figure 4.2. However, the jogging behavior, as any other behavior, also exhibits transient dynamics, particularly at the start-up and stopping of the motion, where the robot does not alternate flight and stance phases. During these periods, such as before second

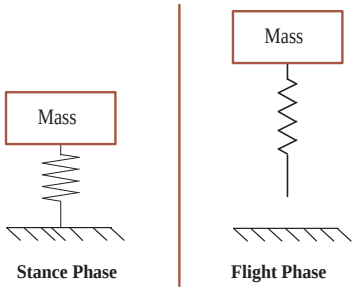


Figure 4.3: RHex's vertical motion can be modeled as a virtual mass-spring system. During stance, the spring compresses and causes the mass to rebound. In flight, the virtual spring detaches from the ground and the mass describes a ballistic trajectory.

1 and after second 3, the ballistic and spring-mass system should not be used, and the corresponding MM system should be replaced with another system using more appropriate models. Therefore, the ability to detect whether the jogging MM system can be used is necessary to enable accurate estimation throughout the operation of the robot. This motivates the development of an information processing system that identifies the robot's behavioral context, thereby determining which MM system should be used.

4.1.1.2 Dynamics along the x-axis

RHex's motion along the x-axis is different from the z-axis motion and should be represented by different motion models. The jogging dynamics along the x-axis are affected by the alternating flight and stance phases, but they are not subject to gravity, as the gravity vector is orthogonal to the lateral axis. The details of the motion can be understood by examining the lateral acceleration profile depicted in Figure 4.4(a). During steady operation, the acceleration profile exhibits large positive and negative amplitudes in succession, separated by periods of near-zero acceleration (between the horizontal lines in the figure). Onboard accelerometers register near-zero acceleration while the robot is airborne, since no force acts on

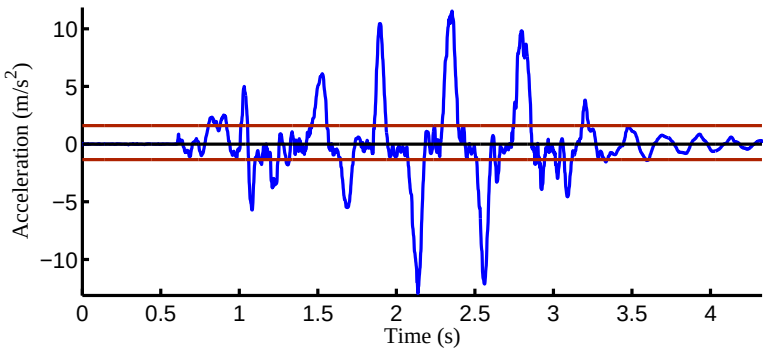
the robot along the x-axis during flight, and the robot simply translates laterally at constant velocity. The positive and negative amplitudes correspond to the left and right tripods being in stance successively, and can be interpreted as follows. After a flight phase, the robot touches the ground with a net lateral speed, which bends the compliant legs laterally along a forward axis passing by the toes. This action can be visualized as a mass (robot) leaning against the top of vertical beams (legs) and bending them, as depicted in Figure 4.5. After the initial deflection, the legs restore the robot body back to the centerline, just in time to undergo a new flight phase. This back-and-forth motion explains the pattern of the acceleration measurements; the increase in amplitude is due to the initial deflection of the legs, up to the extremal acceleration point after which the amplitude decreases as the legs return to their original position.

As with the vertical case, this swaying motion can be represented with a virtual mass-spring system. The parameters of the system can be designed to output acceleration profiles similar to the large-amplitude stance accelerations, as shown in Figure 4.4(b). Here, flight dynamics are simply represented with a constant velocity model, to be used between consecutive stances. Thus, the steady-state dynamics are represented by a combination of mass-spring and a constant velocity models:

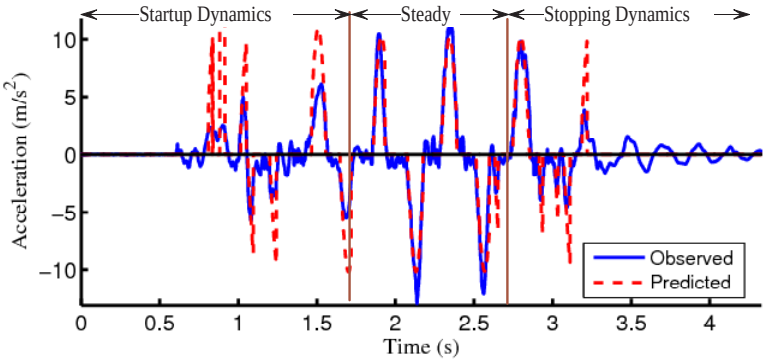
$$\ddot{x} = \begin{cases} 0, & \text{constant velocity,} \\ K_x \left(x - x_0\right)/M, & \text{mass-spring system,} \end{cases}$$

where the state x is lateral position, and K_x and x_0 are the spring constant and rest length of the virtual lateral spring, respectively. Since these models are to be used in successive order, the ability to identify the dynamical context in order to determine when to use each model is paramount.

Figure 4.4(b) also shows that start-up and stopping dynamics are dominated by transients that are difficult to model. This means that the model is a good approximation of the steady dynamics, but becomes inaccurate in the presence of starting and stopping dynamics. Therefore here too, knowing the robot's behavioral context determines when the dynamics can be represented with the mass-spring/constant



(a) The jogging gait induces positive and negative amplitudes duting left and right stances, and periods of near-zero accelerations during flight (region contained between horizontal lines).



(b) Predicted acceleration matches observed acceleration during steady behavior. The model is inaccurate during startup and stopping phases.

Figure 4.4: Output of lateral accelerometer while RHex executes jogging behavior. The jogging gait induces positive and negative amplitudes during left and right tripod stances, and periods of near-zero accelerations during flight.

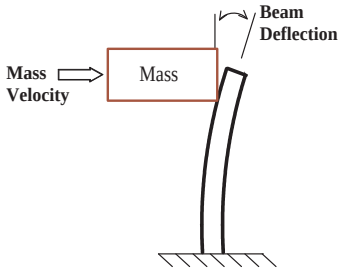


Figure 4.5: RHex's lateral motion can be modeled as a mass leaning against a vertical elastic beam.

velocity models and when these models should not be used for state estimation.

Before leaving this discussion, it is worth noting that the "structure" of the jogging gait (i.e. recirculating tripods, alternating flight and stance, etc.) is the same as the structure of the running gait. The only difference is that the gait parameters are tuned for maximum locomotion efficiency for jogging, and for maximum speed for running, which leads to different leg motion profiles. It follows that the above analysis of jogging dynamics is equally valid for running, and can be used to develop similar low-dimensional models for running.

4.1.2 Dynamical analysis of the walking behavior

Walking dynamics differ from the dynamics of jogging in that successive tripod stances are not separated by a flight phase. Instead, both tripods briefly overlap in stance, with one tripod lifting off a moment after the other tripod touches down. The periodic leg recirculation generates oscillatory motion in the sagital plane, but motions in the lateral plane are less pronounced than with jogging. This is due to a lower rate of leg recirculation that generates smaller forces, and to the fact that having all six legs overlap reduces force differentials, which limits lateral motions. Therefore, the dynamical analysis will focus on the motion along the vertical dimension, as it is more affected by the locomotion forces than the other dimensions.

It is worth recalling that the purpose this analysis is not to control the robot in a way that makes it exhibit dynamics similar to simple models. Instead, the problem is to find simple models that accurately describes the dynamics that result from control laws developed by independent tuning of gait parameters. The tuning seeks to optimize either energy efficiency or speed of locomotion, but does not attempt to generate simple-to-represent dynamics.

The vertical acceleration measured by the onboard accelerometer and displayed in Figure 4.6(a) exhibits a pattern of successive small and large amplitudes. The impulse at the beginning of one cycle of small and large oscillations is generated by a tripod contacting the ground at the end of recirculation, seen in one instance at point 'a' in the plot. Between the moment of contact and when the other tripod lifts off, all six legs touch the ground so the robot can be considered as a mass-spring-damper system attached to the ground with six springs. In this configuration, the impulse generated by the contact translates into a vertical force that acts on the mass, and the six springs dictate the small amplitude and period of the response. At point 'b', the other tripod lifts off, reducing the number of springs from six to three. The result is a mass-spring-damper system with half the spring constant, which explains the larger amplitude of oscillation in the second half of the cycle. At point 'c', the recirculating tripod makes contact with the ground and the cycle repeats.

As the dynamical description suggests, it is possible to represent the vertical walking dynamics with two mass-spring-damper models used in successive order:

$$\ddot{z} = \begin{cases} -K_{z,2}\left(z - z_{0,2}\right)/M - (D_2/M)\dot{z} - g, & \text{two-tripod stance,} \\ -K_{z,1}\left(z - z_{0,1}\right)/M - (D_1/M)\dot{z} - g, & \text{one-tripod stance,} \end{cases}$$

where $K_{z,1}$, D_1, $z_{0,1}$ and $K_{z,2}$, D_2, $z_{0,2}$ are the virtual spring constant, damping coefficients, and rest lengths for one tripod in stance and two tripods in stance, respectively. Generally, $K_{z,2} = K_{z,1}/2$, $D_1 = D_2/2$.

Knowing when to use each model is necessary for accurate state estimation, and the next sections explore a methodology for identifying the dynamical context for

that purpose. In addition, and similarly to jogging, the gait generates transient dynamics at the beginning and at the end of the behavior that are not modeled accurately. Figure 4.6(b) shows that the predictions of these models compare favorably with steady state accelerations, but not during transition phases. Therefore, the walking models should only be used when the robot operates in steady state, which can be detected through the identification of the behavioral context.

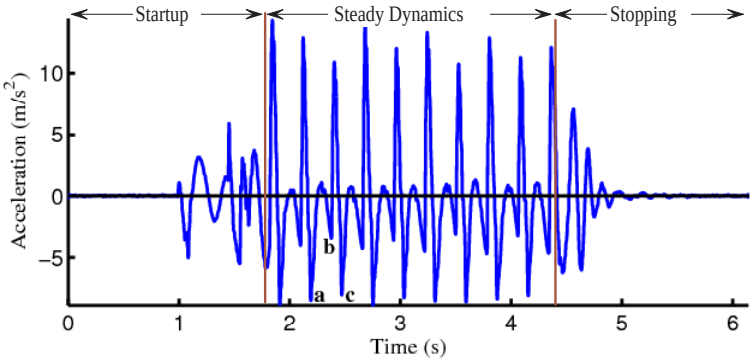
4.2 Identification of Dynamical Contexts

The above dynamical analysis of RHex's behaviors emphasizes the need to recognize dynamics that can be accurately represented with available models. This thesis develops a classification approach to identify the robot's dynamical context.

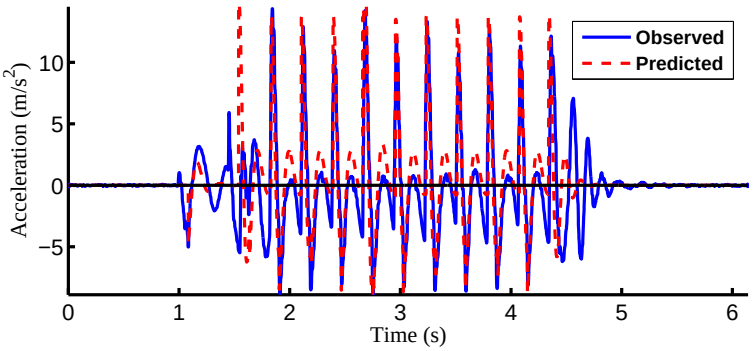
4.2.1 Approach

Knowledge of the dynamical context enables the identification of the model that best represents current dynamics. In the case of RHex, knowing which model accurately represents current dynamics would allow an estimation system to choose, say, between the flight and the stance models as the the robot alternates between the two modes. This information is then used to modify the operation of a multiple model system (such as a GPB2 based on these two models) to only activate the filter based on the appropriate model, thus reducing the system's computational requirements.

Since the two phases alternate rapidly, it is necessary to identify the dynamical context equally rapidly. Recall from Chapter 3 that rapid identification can be achieved by deploying classifiers designed to recognize dynamical contexts at a bandwidth similar to that of onboard sensors. The approach is based on the observation that locomotion dynamics affect the output of onboard sensors, so recognizing that output enables the identification of the dynamics. This can be done by constructing classes of sensor data generated when the robot operates in dif-



(a) Vertical accleration of walking gait. The large and small amplitudes correpsond to stances of three and six legs, respectively.



(b) Model predictions approximate observed accelerations during steady operation.

Figure 4.6: Output of vertical accelerometer while RHex executes walking behavior. Tripod touchdowns generate impulse forces that induce sustained oscillatory motions.

ferent dynamical contexts. Each class is formed of measurements taken while the robot exhibits specific dynamics, so if future measurements are classified in one of these classes, then the corresponding dynamics are identified. The last piece of this setup is to associate a motion model to each class, such that the model is an appropriate representation of the dynamics corresponding to the class. This way, the classification of sensor data simultaneously identifies the dynamics and their corresponding model; in other words, classification identifies the robot's dynamical context[3]. Since classification consists essentially of a straight-forward comparison between current data and previously-constructed classes, the robot's dynamical context can be identified at the rate at which the data is made available by the sensors.

The construction of the classes can be performed with probabilistic methods, as discussed in Section 3.1.1, but such methods are complex to implement and beyond the scope of this thesis. Instead, the classes are constructed by manually clustering labeled data, and extensions to more elaborate methods are left for future work.

To ground this approach in a concrete example, consider the problem of identifying RHex's flight and stance contexts from analysis of its vertical acceleration. The first task is to label classes of accelerometer data points as flight or stance depending on whether the data point where generated while in the flight or stance contexts. This can be done in a laboratory setting where the context can be identified with specialized sensors used to label classes of accelerometer data points. In this case, the specialized sensors consist of strain gauges attached to the legs to measure their deflection; compressed legs indicate stance and extended legs indicate flight. Figure 4.7(a) overlays the output of these strain gauges on the acceleration plot. A visual inspection of the steady-state portion of the plot shows that as expected, and with few exceptions, the flight context is detected when the acceleration approaches gravity ($-9.81 m/s^2$), and the stance context is detected when the acceleration is positive (due to ground reaction). This suggests that the two classes can be formed

[3]As a reminder, the dynamics/model pair is referred to as the dynamical context.

and labeled as follows (see Figure 4.7(b)):

- Flight context: $\ddot{z} < -6m/s^2$,
- Stance context: $\ddot{z} > 0$,

where the vertical acceleration $\ddot{z}$ is measured by onboard accelerometers.

Thus, accelerometer output can be immediately classified into one of the two contexts, and consequently the MM system estimating RHex's height can selectively activate the flight or stance filters. This is in contrast to conventional MM systems that run all filters simultaneously and assign probabilistic weights to their output. In this example, accelerations between $-6m/s^2$ and 0 indicate that the robot is transitioning between flight and stance, so the context cannot be determined with certainty. In these situations, the MM system activates all filters based on both models and resumes conventional operation, as discussed in Section 3.1.2 and demonstrated experimentally in the next chapter.

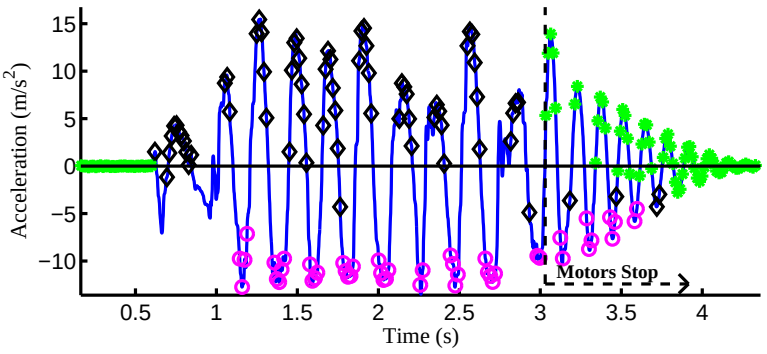
The same analysis applied to the lateral acceleration (Figure 4.4) leads to the following classes:

- Flight context: $-1m/s^2 < \ddot{x} < 1m/s^2$
- Stance context: $\ddot{x} < -1.5m/s^2$ or $\ddot{x} > 1.5m/s^2$

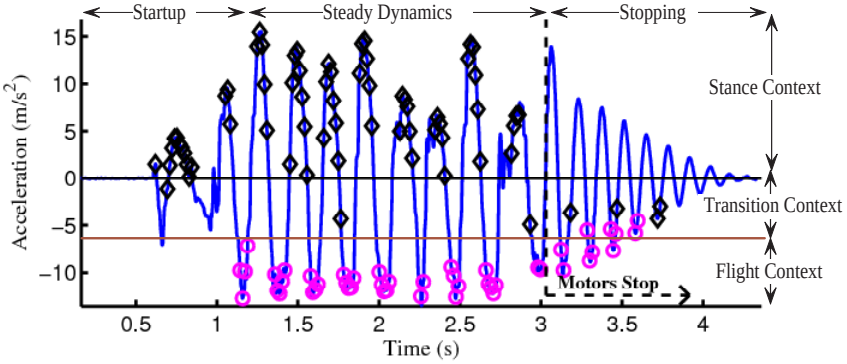
It is worth noting that the strain gauge setup is complex and onerous to install on robots, and its use is generally restricted to the laboratory. Therefore, strain gauges can be used to establish the classes, but not to identify the dynamical context when the robot is deployed in the field.

4.2.2 Limitations

These two examples illustrate the usefulness of a classification approach based on simple comparison between current data and previously-constructed classes.



(a) Flight is represented with circles, stance with diamonds, and stars represent situations where both tripods touch the ground, which happens when the motors stop.



(b) Accelerations smaller than $-6m/s^2$ correspond to the flight context. Positive accelerations correspond to stance.

Figure 4.7: The output of leg strain gauges(circles, diamonds and stars) is overlaid on the vertical oscillatory accelerometer data. The output of strain gauges is used to define classes that correspond to dynamical contexts.

Unfortunately, the approach suffers from two types of shortcomings. First, the simplicity of the classes makes the classification susceptible to sensor noise. For instance, when data points are close to the boundary of a class (say a lateral acceleration $\ddot{x} = 1.6m/s^2$, which is classified in the stance context but is close to the context's boundary $\ddot{x} = 1.5m/s^2$), small sensor disturbances can cause data points to exit that class and lead to a misclassification. This problem is mitigated by designing transition regions among all the classes, such as leaving a gap between the flight and stance contexts. These regions effectively act as buffers that limit the risk of activating the wrong filter. Data points incorrectly exit a context are classified in the transition region. This has the effect of activating all the filters of the MM system, so the loss of accuracy is much smaller than if the MM system trusted a single, wrong filter. Thus, the consequence of misclassifying data points is to revert back to the normal operation of multiple model filters, which is a suboptimal but still acceptable solution over short periods of time. The benefit of this approach is largest when the measurements are classified unambiguously within a context, such that limited noise would not affect the classification.

The second limitation is that this approach requires different dynamics to be represented by distinct classes. This cannot always be guaranteed, as some behaviors produce different dynamics that induce similar measurements. An example is RHex's walking gait, where a portion of the acceleration generated during the two-tripod stance is between -5 and $+1m/s^2$, which is the same acceleration range of the one-tripod stance (see Figure 4.6). Therefore, clustering acceleration data points during the one-tripod stance leads to a class that can correspond to either the one- or the two-tripod dynamics. This ambiguity means that such a class cannot be used to identify the dynamical context.

Thus, the straight-forward classification approach designed to enable high-bandwidth identification cannot be used in some situations. One approach to disambiguating overlapping classes is to perform the classification task over a history of measurements rather than just classifying the current data points. This would require a model that represents the sequence of occurrence of the contexts, which, when

used in conjunction with a history of measurements, enables the identification of correct dynamical contexts. The technique used to address this problem also enables the identification of behavioral context, which is formed from a sequence of dynamical contexts. Therefore, the approach to identifying behavioral contexts can also be used to recognize dynamical contexts that could not be identified with straight-forward classification.

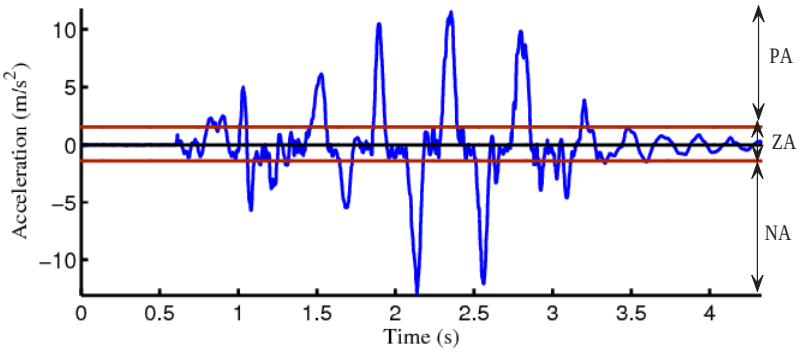
The details of the proposed technique for behavioral context identification are provided in Chapter 3, but the next section completes the explanation by implementing the approach on RHex.

4.3 Identification of Behavioral Contexts

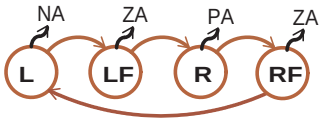
The technique presented in Section 3.2.3 is used here to identify RHex's behavioral contexts. The task is to determine when RHex operates in the jogging and walking contexts from the analysis of its lateral and vertical accelerations. The presentation centers on estimating the jogging context from lateral accelerations, and provides implementation details and experimental results. As for the other behavioral contexts, their identification is performed with a similar implementation, so only results are reported.

4.3.1 Discrete-state model for jogging behavior

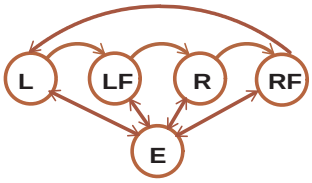
Consider the lateral motion of RHex as it executes the jogging behavior, and recall that sensor readings during the steady portion of the gait correspond to dynamics that are accurately represented by the mass-spring model of Section 4.1.1.2. In order to identify the behavioral context, the acceleration data is first discretized as per Figure 4.9(a). The observation symbols are positive (PA), negative (NA) and zero (ZA) accelerations, where PA corresponds to accelerations larger than $1m/s^2$, NA to accelerations smaller than $-1m/s^2$, and ZA spans the range in between, as shown in the figure.



(a) Lateral acceleration while jogging.



(b) Markov chain generates observation symbols.



(c) Markov chain augmented by error state E.

Figure 4.8: RHex's lateral acceleration is discretized into observation symbols (subfigure (a)), and the Markov chains of subfigures (b) and (c) model a process that generates these symbols.

A Markov chain that could model a process that generates these symbols is presented in subfigures (b) and (c). The model's states L and R stand for left tripod and right tripod touchdowns, LF corresponds to the flight phase during transitions from left to right tripods, and RF corresponds to flight during transitions from right to left. This seems to be a reasonable choice of states, since the left and right tripods are expected to induce positive and negative accelerations, and the flight phase negligible acceleration. The distinction between left and right flight is necessary to anticipate the which tripod lands after each flight phase, so that the appropriate continuous motion model can be used.

In practice, the discretization proves too coarse, as it ignores the acceleration's peak magnitude; it is important to check that the acceleration reaches magnitudes

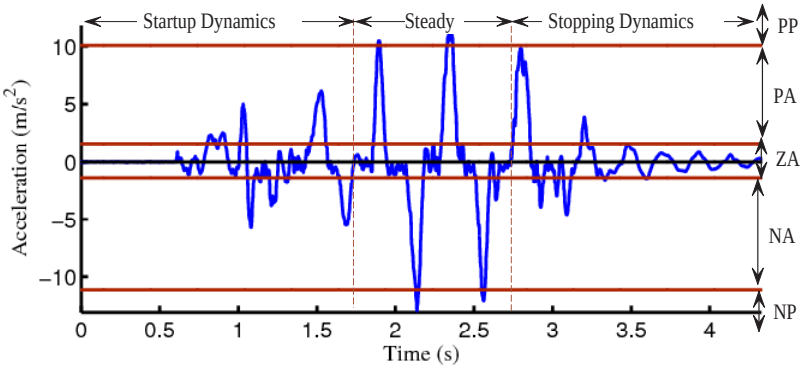


Figure 4.9: Output of lateral accelerometer. The jogging gait induces positive and negative amplitudes during left and right tripod stances, and periods of near-zero accelerations during flight.

that are consistent with the robot's steady-state dynamics, in order to avoid false positives. Therefore, the discretization is refined by adding the symbols PP and NP for positive and negative peaks, respectively, as shown in Figure 4.9.

A Markov chain model of the process generating these symbols is presented in Figure 4.10. Since it is important to verify that jogging accelerations reach the expected amplitude, the right and left tripods are represented by three states: the first state (R1 or L1) corresponds to the ascending acceleration, the second state (R2 or L2) corresponds to the peak acceleration, and the third state (R3 or L3) corresponds to descending acceleration. Thus, R1, R2 and R3 (or L1, L2 and L3) generate the symbols PA, PP and PA (or NA, NP and NA), respectively. As before, RF and LF stand for right and left flight, respectively.

This Markov chain can now be used as a hidden Markov model to compute the probability distribution over the states and enable the identification of the jogging context. If the sequence of observed symbols matches the sequence predicted by the Markov model, then there is a high likelihood that the robot is executing jogging at steady state. Conversely, detecting out-of-order symbols indicates that RHex is no longer in the jogging context.

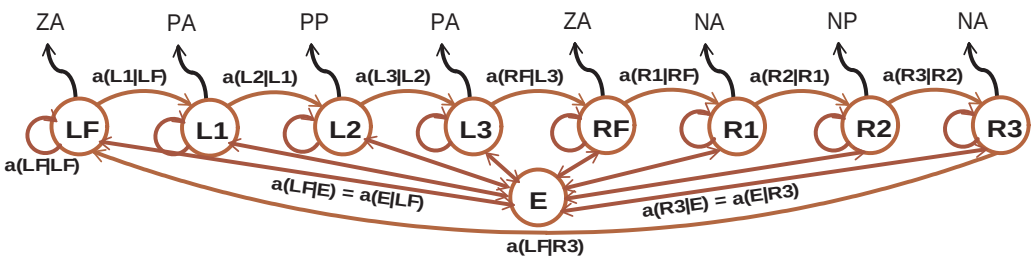


Figure 4.10: Markov-chain model of the process generating jogging symbols. The error state is added to capture out-of-order state transitions.

4.3.2 HMM Results

In order to detect out-of-order transitions, the jogging HMM is augmented with the error state E, and its transition probabilities are determined as follows. Self transitions a_{ii} to state i are computed by counting the number $\overline{s}$ of symbols that occur when the system is in each state. The expected number of observations in a state, conditioned on starting in that state, is $\overline{s} = \sum_{s=1}^{\infty} s \cdot p(s)$, where p is the probability of undergoing s self-transitions to the same state. This equation can be re-written as $\overline{s} = d(a_{ii})^{d-1}(1 - a_{ii}) = \frac{1}{1-a_{ii}}$ [59], so $a_{ii} = 1 - 1/\overline{s}$. For example, R2 generates on average 22 PP symbols, so $a_{R2|R2} = 1 - 1/22 = 0.9432$.

The transition probability from one state to the next is set equal to the transition probability from the same state to E, to ensure that the HMM is not against detecting out-of-order transitions. The probabilities for E are set as $a_{E|E} = 0.999$, and $a_{E|i} = \left(1 - a_{E|E}\right)/N$, where $N = 8$, the number of states excluding E, and $i \in N$. The observation probabilities are set in Table 4.1 and reflect the model of Figure 4.10.

The HMM is initialized in the error state with (3.1) and the probability distribution α is computed recursively with (3.2).

Figure 4.11(a) shows that as expected, the sequence of highest-likelihood estimates follow the prescribed order over the region of steady dynamics. Outside this range, the sequence is no longer expected to be in-order (Figure 4.11(b)). Figure 4.11(c)

b	LF	L1	L2	L3	RF	R1	R2	R3	E
PP	0	0	1	0	0	0	0	0	1/5
PA	0	1	0	1	0	0	0	0	1/5
ZA	1	0	0	0	1	0	0	0	1/5
NA	0	0	0	0	0	1	0	1	1/5
NP	0	0	0	0	0	0	1	0	1/5

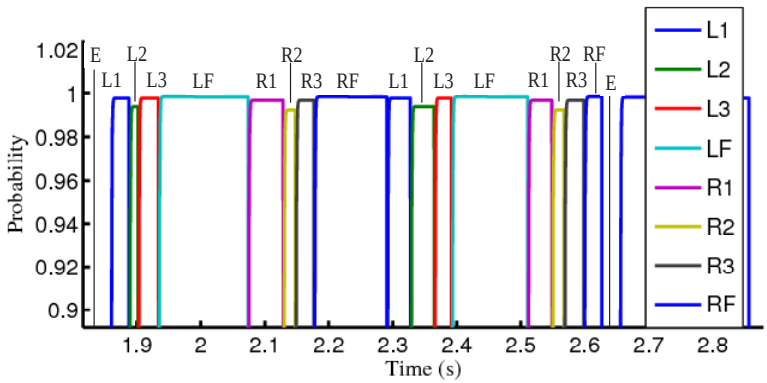
Table 4.1: Observation probabilities for the jogging HMM. The observation probabilities for the state E are uniformly distributed over the observation symbols.

plots the probability of not being in the error state (solid line), an indication that the spatial structure is recognized. The plot shows positive identification over the range of steady dynamics, but false positives are predominant throughout the experiment.

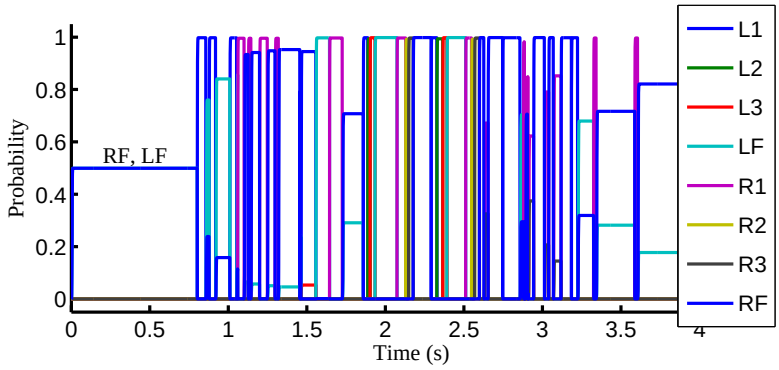
One reason for this false positive problem is the HMM's inability to verify correct sequences over several steps, as mentioned earlier. Another reason is the lack of temporal analysis in the inference algorithm. This is illustrated at the start of the behavior, where the robot is standing still with zero acceleration. The HMM interprets the resulting ZA symbols as indicative of flight, so with the lack of better information, the system thinks that either one of the flight states is in effect ($\alpha_{RF} = \alpha_{LF} = 0.5$, Figure 4.11(b)).

4.3.3 Timed Automata Results

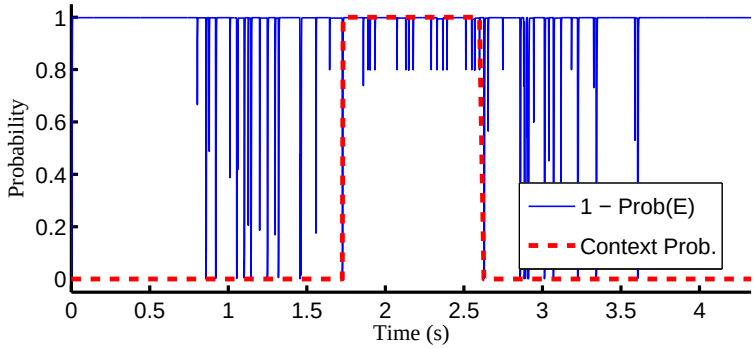
The inability to track transitions over multiple steps and to measure the duration in an HMM state can be remedied through bookkeeping, by using a timed automaton that tracks HMM estimates over time. The HMM output is treated as input to the automaton as per Figure 4.12(b). The automaton's structure has two intermediate layers, so a sequence of three ordered inputs is necessary to reach the final layer and output a flag identifying the jogging context. For example, a sequence of inputs LF→L1→L2 induces transitions S→1→2→3, and output the jogging flag.



(a) Close up shows states estimated in right sequence over region of steady dynamics.



(b) Probability distribution over states for the entire run.



(c) The probability of not being in the error state is a poor indicator of context (solid line). However, the combined output of HMM and timed automaton leads to accurate results (dahsed line).

Figure 4.11: Output of HMM and timed automaton for jogging.

Duration timeouts are also added as in 4.12(c). When a timeout event takes place, the automaton is reset to S, and identification is delayed until a new ordered sequence is observed. The same happens when the input is the HMM error state E. All other out-of-order inputs reset the automaton to the appropriate state in the first layer (see shaded transitions in 4.12(b)).

The output of the timed automaton is plotted in dashed lines in Figure 4.11(c). The jogging context is correctly identified over the region of steady dynamics, and the false positives are eliminated.

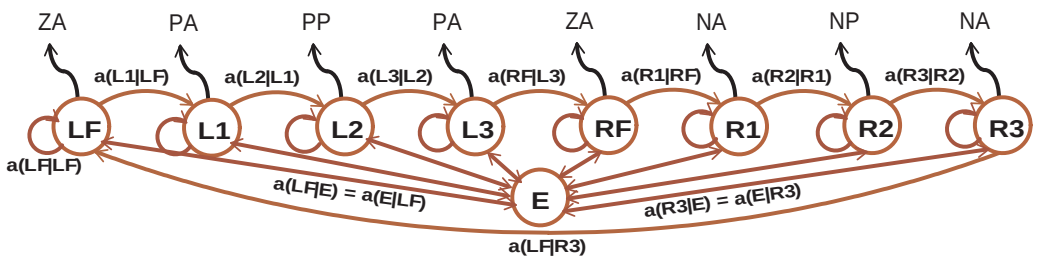
4.3.4 Identification of behavioral contexts along different dimensions

The purpose of this section is to reinforce the understanding of behavioral context. Recall that a behavioral context does not correspond to a behavior as such, but rather to the specific dynamics that can be represented by available models.

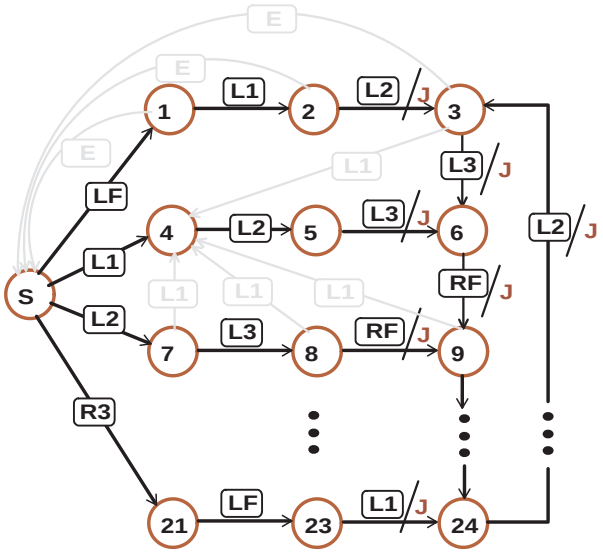
This subtle but important distinction is clarified by comparing the lateral jogging context and the vertical jogging context. Both contexts are identified with data generated in the same experiment, but the lateral context is identified from lateral accelerations, and the vertical context from vertical accelerations. The contexts need not be identical, as the robot dynamics could fit the vertical model at a time where they do not fit the lateral model, and vice versa.

This is indeed the situation with RHex. First, the behavioral context for vertical jogging is identified by constructing an HMM-automaton estimator following the steps outlined earlier. The context information is overlaid on the vertical acceleration in Figure 4.13(a). Now the lateral context identified earlier is overlaid on lateral acceleration in Figure 4.13(b), and as can be seen from the juxtaposition of the two subfigures, the contexts do not coincide.

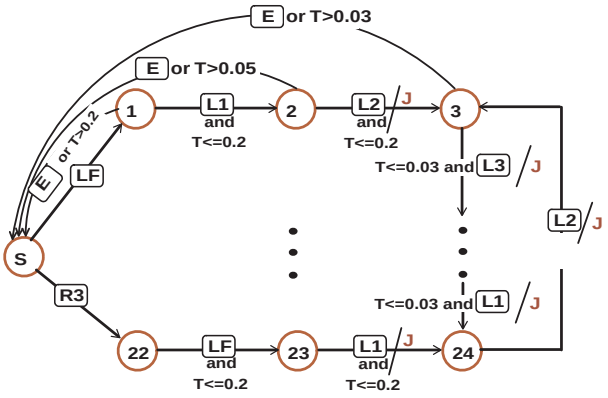
This is explained as follows. As RHex initiates its jogging behavior, the vertical motion rapidly reaches steady state, leading to an early identification of the vertical



(a) The jogging HMM is preproduced here for convenience.



(b) Ordered inputs induce state transitions. Out-of-order inputs reset the automaton.



(c) Timeouts reset autmaton

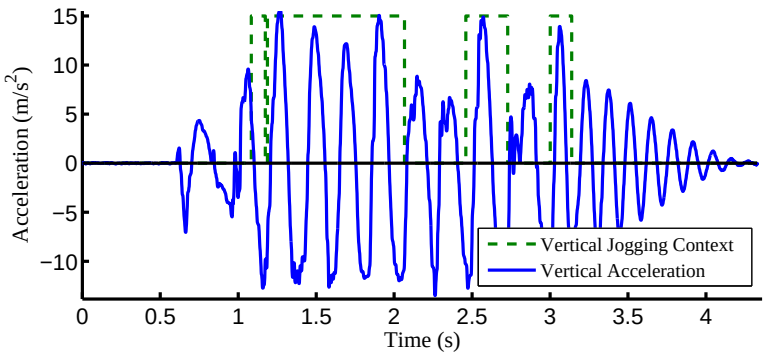
Figure 4.12: Timed automaton for jogging. The graphical representation is abbreviated for clarity.

context. In contrast, the lateral motion takes longer to reach steady state, probably because unlike the vertical case, it is not the direct consequence of motor action, but rather the result of a force differential between the left and right side of the tripods. This force builds up over time, which explains the late identification of the lateral context. Thus, the vertical models can be used soon after the start of the behavior, but the lateral models should wait longer before they can be trusted.

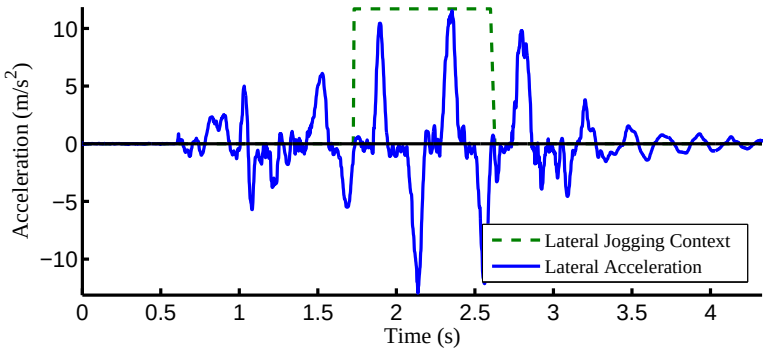
This analysis underlines the distinction between a behavior and a behavioral context. It is possible to anticipate using this distinction to form a new quantifiable definition of a behavior. For instance, a robot could be said to be jogging if and only if all the contexts along different dimensions are concurrently identified. When only a subset of contexts overlap, then there is a low confidence that the jogging behavior is being executed according to expectations. This development could lead to new types of adaptive behavioral control, but it is beyond the scope of this thesis.

4.3.5 Results for the walking behavior

The acceleration profile of the walking behavior dictates a different discretization than for jogging. The data is discretized along the horizontal lines of Figure 4.14(a) to capture the salient points of inflection (marked with circles). A four-state Markov chain models the sequence of symbols that are induced by the steady dynamics and an HMM estimates the sequence of most likely states. A timed automaton shown in Figure 4.15 identifies the behavioral context when a sequence of four ordered inputs is observed. The HMM output is plotted in solid lines in Figure 4.14(b), where the number of false positives is large. Figure 4.14(b) shows that the output of the timed automaton correctly identifies the walking context over the steady region (dashed line) and eliminates the HMM's false positives (solid line).

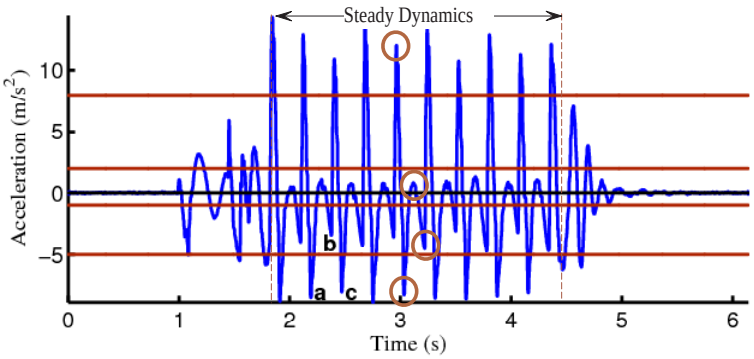


(a) Vertical motion.

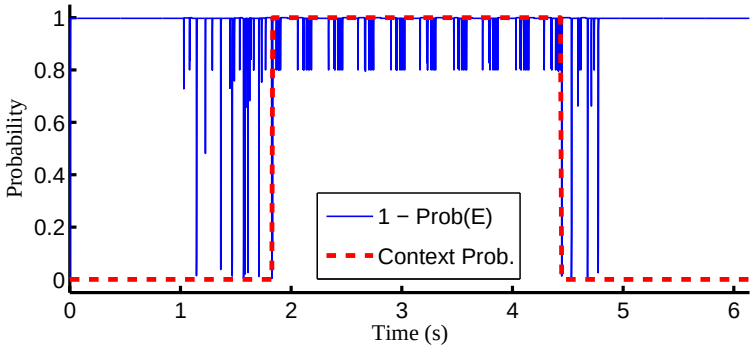


(b) Lateral motion.

Figure 4.13: The behavioral contexts for vertical and lateral jogging do not always overlap.

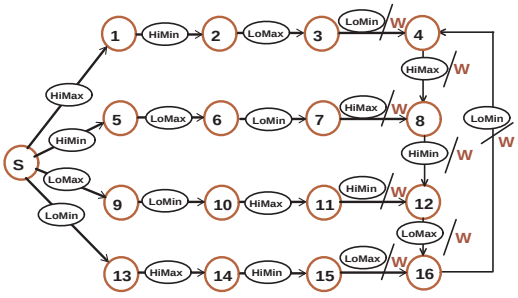


(a) Salient features of the data are represented with circles. They correspond to the symbols generated by the walking Markov model.



(b) Identification of walking context

Figure 4.14: Output of HMM and timed automaton for walking.



(a) Ordered inputs induce state transitions. Out-of-order inputs reset the automaton.

Figure 4.15: Timed automaton for jogging. The graphical representation is abbreviated for clarity.

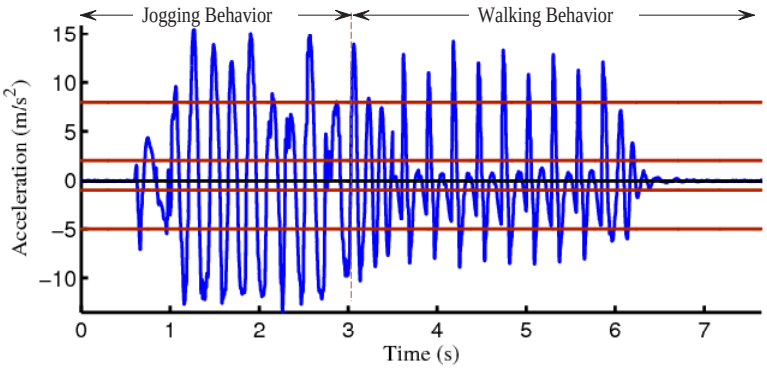
4.3.6 Simultaneous identification of the walking and jogging contexts

The goal of this experiment is to test the robustness of the approach when the system exhibits multiple behaviors. To this end, the vertical and lateral acceleration data of both gaits are consolidated into a single data stream (Figures 4.16(a) and 4.16(b)). The results show that the jogging context is successfully identified from the lateral data (solid line in Figure 4.16(c)). The walking context is also identified successfully from the vertical data, but with a seemingly false positive around second 1. However, a close inspection of the vertical acceleration shows that jogging start-up transients exhibit the same structure as the walking dynamics, which suggests that the jogging gait executes a couple of walking steps before achieving steady-state jogging. This result concurs with the observation of experienced operators of the robot performed independently of this research.

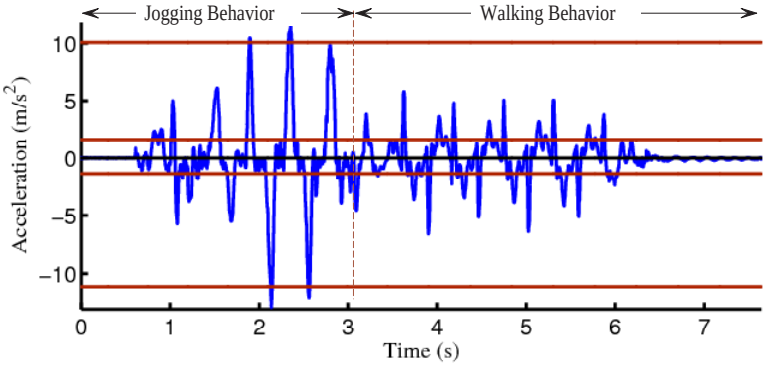
4.4 Summary

The estimation framework developed in this thesis is shown in experiment to enable the identification of a robot's dynamical and behavioral contexts. Dynamical contexts constructed from leg strain gauges can be identified at high bandwidth with straight-forward classification. The identification of behavioral contexts requires a more elaborate information processing strategy to analyze sensor data over time. The strategy relies on the combined use of hidden Markov models and timed automata to recognize patterns in sensor data that correspond to a behavioral context. These developments validate experimentally the framework described in Chapter 3.

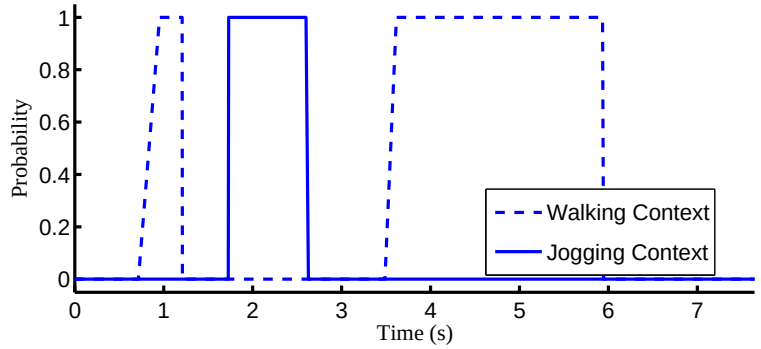
The ability to identify behavioral and dynamical contexts can significantly improve the scalability and accuracy of multiple model filters. This is demonstrated in Chapter 5, where RHex is again used as a real-world example to demonstrate how the results of this chapter improve the performance of state estimation.



(a) Combined vertical accelerations



(b) Combined lateral accelerations



(c) Simultaneous identification of walking and jogging contexts

Figure 4.16: Identifying jogging context in multiple-context data

Chapter 5

Context-Based State Estimation

The purpose of this chapter is to demonstrate through experimental and simulation examples that the context-based estimation framework improves the accuracy and scalability of multiple model (MM) estimation systems. The examples are chosen to illustrate concrete instances where conventional estimation techniques suffer from the problems described in Chapter 2. In light of these examples, proof-of-concept implementations of the context-based framework are presented in detail, so that the framework's mechanism and estimation results are understood in a practical setting.

This chapter is the continuation of Chapter 4, as together they complete the implementation of the framework described in Chapter 3. Chapter 4 shows how a system's context can be identified from analysis of sensor data, and the current chapter ties back context identification to multiple model state estimation. These two parts of the same framework provide a solution to the estimation problems defined in Chapter 2.

As a refresher of the concepts developed in this thesis, recall that the notion of a context is introduced with a double definition: a dynamical context is a discrete state that corresponds to locomotion dynamics represented with one motion model; and a behavioral context is a discrete state that corresponds to dynamics represented by a collection of dynamical contexts that transition among one another.

The purpose of identifying the dynamical context is to direct multiple model (MM) filters to only trust accurate models and improve the accuracy of their state estimates. As for the behavioral context, its identification enables the deployment of multiple small-scale MM systems instead of one large-scale system, which improves the scalability of multiple model filtering. The current chapter puts this strategy to practice in simulation and in experiments conducted on RHex.

The presentation starts with a discussion of simple models, their usefulness at helping filters estimate the state of systems with complex dynamics, and the trade-off between model accuracy and modeling effort. The discussion is grounded in a real-world example that helps understand when and how simple models can enable accurate state estimation. Next, the presentation shows how context identification can improve the scalability of multiple model filtering. A modified MM algorithm is shown in simulation and in experiment to reduce computational requirements by enabling the selective activation of individual filters within the MM system. The presentation also exhibits an instance where low-sensitivity sensing leads to the failure of state estimation; context identification is shown to overcome this problem and enable accurate estimation. Last, an experiment conducted on a wheeled robot illustrates how simple motion models discard sensor information, and shows that the context-based framework provides means for exploiting such information to improve the output accuracy of individual estimation filters.

5.1 State Estimation Based on Simple Motion Models

At first thought, low-order and low dimensional models are inaccurate approximations of high-order and high dimensional dynamics and therefore are not useful for state estimation. Yet, abundant empirical evidence in the aircraft and robotics fields suggests that simple models can be relied upon for accurate control and, as this thesis asserts, for estimation.

For the case of RHex, the robot's complex six-dimensional motion can demonstrably be *controlled* with low-order, three-dimensional models using feedback [71]. In this

example, RHex's lateral, forward and heading motions are modeled as a first order unicycle when jogging, and as a second order unicycle when running. Control policies use these models to regulate the robot's heading and successfully execute line following behaviors at both speeds. The likely reason behind this success is that the low-order models appropriately describe the aggregate motion of the robot, i.e. the average effect of the legs' interactions with the ground on body velocity. As such, the models ignore the fine-scale details of the dynamics, such as traction and inertia of individual legs. Yet, modeling such dynamics is not necessary for navigation tasks as long as the large-scale dynamics are correctly captured by the motion models.

Simple models are also used for estimating the state of robots with complex dynamics. For example, Durrant-Whyte [20] and Verma [83] use models that capture an abstracted representation of rover dynamics to perform successful localization and fault detection. This section studies the usefulness of such simple models at estimating RHex's height during jogging and walking behaviors. The challenge consists of deriving accurate height estimates over time from noisy acceleration measurement performed on board the platform.

It is worth stressing that the filters used in this section are not modified by contextual information. The purpose here is to quantify the accuracy of conventional estimation tools (such as Kalman and multiple model filters) when using simple models.

5.1.1 Estimating RHex's height while jogging

The acceleration for the jogging behavior is shown in Figure 4.3 and reproduced here for convenience in Figure 5.1. A naive approach to height estimation would be to integrate the acceleration data twice to recover height. This causes the estimates to diverge rapidly because the onboard accelerometers have significant noise that leads to error accumulation in the double integral. The result can be seen in Figure 5.2, where the double integration estimate is compared to ground

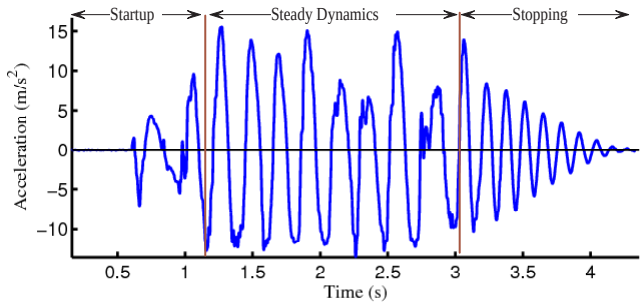
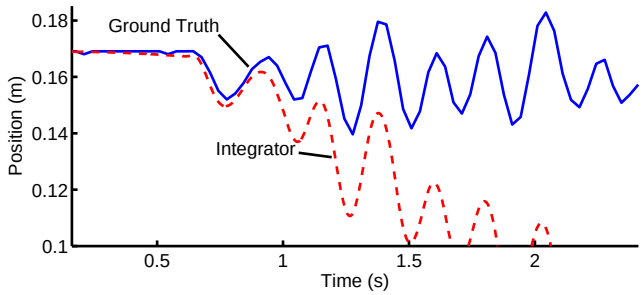


Figure 5.1: Output of vertical accelerometer while RHex executes jogging behavior.



(a)

Figure 5.2: Double integration of accelerometer data leads to diverging estimates of heights.

truth measurements of the robot height[1]. Ground truth measurements show that RHex's height oscillates around the robot's rest height, but the integrator's estimates wrongly oscillate around a downward slope.

An alternative approach is to make appropriate use of motion models to improve the accuracy of height estimates. A close inspection of the dynamics in Figure 5.1 helps understand the locomotion dynamics and design such models. At the beginning of the jogging behavior, the robot is subject to high-order start-up dynamics

[1]The ground truth measurement system consists of high-speed cameras that register the position of LEDs placed on the robot body. The cameras cover a limited surface area, which explains the short experimental runs (about 4 seconds for jogging). Nevertheless, these results are useful because the available space is large enough to allow the robot to achieve steady-state motion before exiting the cameras' field of view.

that induce complex acceleration patterns observed in the first phase of the plot. These start-up transients are gradually replaced by the steady-state dynamics of the second phase, which in turn, are followed by stopping transients that result from energy dissipation once RHex's motors stop. The start-up and stopping dynamics result from complex interactions between the ground and the legs and are difficult to model accurately. In contrast, the steady dynamics are understood to result from a repeating sequence of flight and stance phases, which can be approximate by straight-forward models. These models are described in Section 4.1.1.1 and are formulated as follows:

$$\ddot{z} = \begin{cases} -g, & \text{flight phase,} \\ -K_z\left(z - z_0\right)/M - g, & \text{stance phase,} \end{cases}$$

where z is the state (height), g is gravity, K_z is the virtual spring constant, z_0 its rest length and M is the robot mass.

Given that the steady-state portion of the jogging dynamics can be described by two models, the height can be estimated using the GPB2 algorithm based on the two models. The GPB2 mechanism is described in Algorithm 2 on page 34, but it can be more clearly understood with the graphical representation of Figure 5.3. Consider the setup of the GPB2; since the system consists of two models corresponding to two possible modes of operation, each iteration of the GPB2 starts with the hypothesis of the robot being in the flight or in the stance modes at the end of the previous iteration. These hypotheses are represented with the first column of two circles: F (flight mode, white circle), and S (stance mode, black circle). Each hypothesis has a probability $Prob_i$ and a height estimate z_i, with $i \in$\{F,S\}. In other words, z_i represents the height of the robot if it was in mode i at the end of the previous iteration, and $Prob_i$ represent the probability of the robot being in mode i at the end of the previous iteration, with $\sum_i Prob_i = 1$.

From each of the two initial hypotheses, the robot can transition into one of the two modes, leading to a total of four hypotheses. The transition is governed by the user-defined transition probabilities $T_{i,j}$, with $i, j \in$\{F,S\}. The four hypotheses are represented by the middle column of four circles, where each circle represents

the hypothesis of being in mode i at the current iteration t and having been in mode j at the previous iteration $t-1$. As a shorthand notation, the hypothesis of having mode i at time t is referred to as i_t, and the transition hypothesis is referred to as (i_t,j_{t-1}). To each of the four hypotheses corresponds a Kalman filter (Algorithm 1 on page 26) whose model (F and Bu) represents the dynamics of the current mode, and whose prior ($z^u_{m,k}$) is the output of the corresponding initial mode. For example, the first circle ('Flight' at current iteration, 'Flight' at previous iteration, or ($\mathrm{F_t}$,$\mathrm{F_{t-1}}$)) has a corresponding filter based on flight dynamics ($\ddot{z} = -g$), and its prior z_F computed at the previous iteration. For compactness, the notation refers to $filter_i$ as the filter based on model i, and to $z_{i,j}$ as the output of the filter associated with the hypothesis (i_t,j_{t-1}). With this notation, the output of the first filter is $z_{F,F} = filter_F(z_F)$. The third circle, while having the same flight dynamics, uses the prior z_S per the hypothesis ($\mathrm{F_t}$,$\mathrm{S_{t-1}}$), and outputs $z_{F,S} = filter_F(z_S)$.

Each of the four Kalman filters outputs individual estimates of height $z_{i,j}$ and likelihoods $p_{i,j}$ (Algorithm 1). Next, a probability $Prob_{i,j}$ for each hypothesis is computed by multiplying $p_{i,j}$ by $T_{i,j}$ and $Prob_j$ from the previous iteration (Algorithm 2). The hypothesis probability is then used to scale the individual estimates $z_{i,i}$ and consolidate them into the intermediary mode-specific estimates z_i. This step is commonly referred to as hypothesis collapsing, as it reduces the number of hypotheses back to the original number of two. Thus, the output of the individual filters based on the same model are consolidated together, and serve as the initial estimates for the next iteration of the algorithm.

As the GPB2 is running through the cycle of expanding and collapsing hypotheses, it is possible to extract a unique state estimate at any time. This is simply done by consolidating the intermediate mode-specific estimates z_i weighted by their hypothesis probability $Prob_i$ into the system's best estimate z. In the case of RHex, z is considered the best estimate available for the robot's height.

This approach leads to satisfactory results, with height estimates closely matching

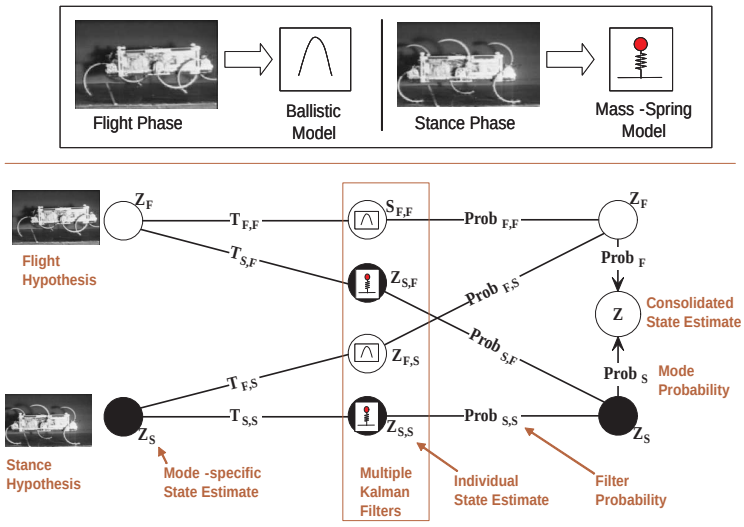


Figure 5.3: A graphical representation of the Flight-Stance GPB2. The GPB2 algorithm spawns four filters whose output is consolidated into the best estimate of robot height.

ground truth measurements as can be visually verified in Figure 5.4. The mean and standard deviation of the root mean squared (RMS) difference between the estimated height and the ground truth measurement over 12 experiments is $1.85cm$ and $0.37cm$, respectively. The mass-spring parameters of the stance model are set to the actual mass of the robot ($8.5Kg$) and the spring constant that yields acceptable state estimates is found to be $6800N.m$, a value close to the physical spring constant estimated at $6600N.m$.

It is worth highlighting that the hypothesis probabilities $Prob_{i,j}$ are the weights with which individual outputs are scaled before consolidation. As emphasized in the previous chapters, it is necessary that the GPB2 algorithm rapidly converges to the correct values of $Prob_{i,j}$ to avoid assigning wrong weights to individual outputs and potentially causing the estimation to fail. Looking forward, Section 5.3 shows that the GPB2 fails when the weights are computed incorrectly, and demonstrates how the context-based estimation framework can overcome this problem.

In the current example, the estimation is successful because the weight assignment

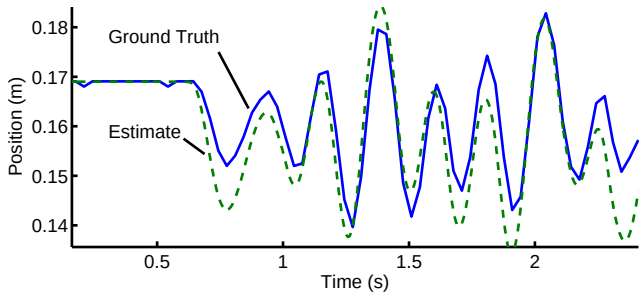


Figure 5.4: A GPB2 based on ballistic flight and mass-spring stance models generates accurate state estimates. Over 12 experiments, RMS errors have a mean $1.85cm$ and a variance of $0.37cm$.

is handled appropriately by the GPB2. This can be seen from the plot of $Prob_i$ in Figure 5.5 that shows that the GPB2 correctly alternates flight and stance modes (even if the probability estimates can on occasion be corrupted by sensor noise, as in the zoomed-in view). The GPB2 is able to correctly compute the hypotheses probabilities because the flight and stance phases are sufficiently long to allow individual filters to converge.

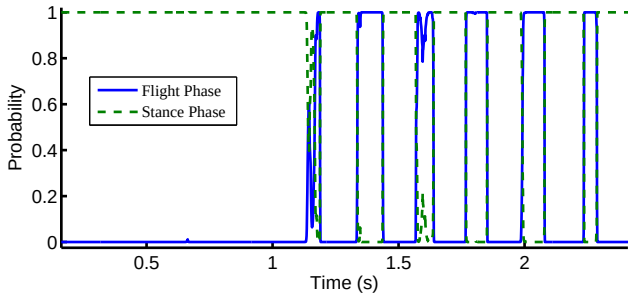
5.1.2 Estimating RHex's height while walking

RHex's acceleration profile while walking is shown in Figure 4.6, and as discussed in Section 4.1.2, the walking dynamics can be modeled as two mass-spring-damper systems:

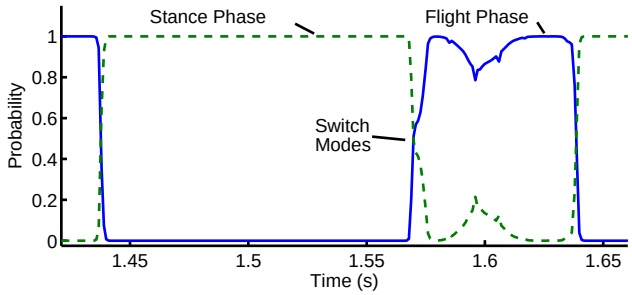
$$\ddot{z} = \begin{cases} -K_{z,2}\left(z - z_{0,2}\right)/M - \left(D_2/M\right)\dot{z} - g, & \text{two-tripod stance,} \\ -K_{z,1}\left(z - z_{0,1}\right)/M - \left(D_1/M\right)\dot{z} - g, & \text{one-tripod stance,} \end{cases}$$

where $K_{z,1}$, D_1, $z_{0,1}$ and $K_{z,2}$, D_2, $z_{0,2}$ are the virtual spring constant, damping coefficients, and rest lengths for one tripod in stance and two tripods in stance, respectively.

Unlike the jogging behaviors where a GPB2 was necessary to handle the two-model estimation, the walking behavior can use a single Kalman filter (KF) whose model is modified by changing its parameters depending on the tripod.



(a) Plot of $Prob_i$ with $i, j \in$ S,F



(b) Zoom-in on plot of $Prob_i$

Figure 5.5: Plots of $Prob_i$ for RHex while jogging. The correct assignment of probabilities enables successful height estimation.

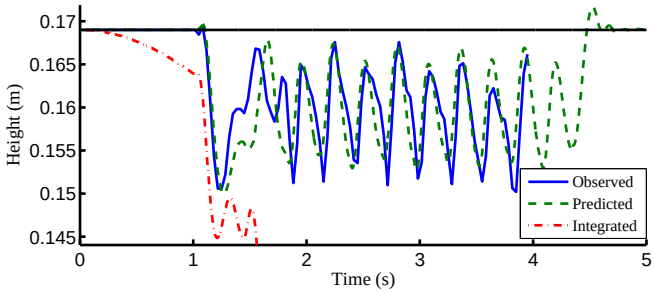


Figure 5.6: Height estimates of RHex while walking. Estimates produced by a model-based KF converge, whereas estimates derived from integrating the acceleration data diverge.

In contrast to the jogging behavior where the precise point of transitions between flight and stance is difficult to determine, transitions between one and two tripods in the walking behavior can be detected directly (and precisely) from motor encoders. This is because the walking motion is quasi-static, so the actual dynamics follow closely the rotation of the legs. It follows that it is not necessary to use a GPB2 for walking; a single KF can be used instead, provided that the two sets of parameters are swapped at each transition.

This straight-forward implementation leads to satisfactory results, as shown in Figure 5.6. The height estimate generated by the filter matches closely ground truth measurements, whereas estimates derived from direct integration of acceleration data diverge.

These two examples demonstrate the usefulness of simple models for state estimation. However, when many such models are necessary to represent complex dynamics, state estimation can only be performed with large multiple-model filters. The next section shows that context identification improves the scalability of MM filters, thus providing practical value to simple models even when used in

large numbers.

5.2 Context Identification Improves Scalability of MM Systems

Chapter 3 presents a double strategy for improving the scalability of MM systems. First, recognizing a robot's dynamical context enables MM filters to identify the accurate model and rely uniquely on information provided by the corresponding filter. This allows the selective activation of individual filters and results in a reduction of the computational cost. Second, the identification of the robot's behavioral context allows the deployment of multiple small-scale MM systems, instead of a conventional monolithic large-scale system. As a consequence, multiple-model filters scale better with systems that exhibit a large number of behaviors (e.g. legged robots). This section gives practical examples of both context-based mechanisms, and provides some implementation details and experimental results.

5.2.1 Dynamical context identification reduces the number of active filters

Improving the GPB2's accuracy and scalability can be achieved by modifying the MM algorithm to account for information about the dynamical context. Recall that the GPB2 algorithm's weight distribution can be understood as a measure of the algorithm's relative confidence in the output of each individual filter; by extension, the weight distribution is the algorithm's estimate of the relative accuracy of each model. Now if the dynamical context is identified, then one model is unambiguously recognized as appropriate for estimation, and the other models as inappropriate. The weight distribution that would reflect this information should assign a value of one to the appropriate filter, and zero to all other filters. More precisely, the weight value of one should be assigned to the filter that corresponds to the hypothesis stating that the system is self-transitioning into the identified mode. The GPB2

algorithm should be modified accordingly to avoid the risk of miscalculating the weights and thereby incorporating knowingly inaccurate individual estimates into the consolidated state estimate.

Section 3.1.3 on page 50 provides the modified algorithm and describes it in detail. For the system at hand, the transition probabilities are set as a function of the state of the robot as follows. If RHex's velocity is negative, the descending robot is expected to touch down, so the transition probabilities are biased towards transitioning into stance[2]. Conversely, positive velocities bias transitions to flight. Setting transition probabilities as a function of state improves the accuracy of the GPB2, as it encodes information about the dynamics that the GPB2 algorithm is unable to capture from its model set.

The identification of the dynamical context can be performed with low-cost contact sensors placed at the end of the legs. In RHex's case, the robot is already equipped with leg-strain gauges that measure leg deformation, so their output is discretized into binary contact/no contact information, and this is the only information used to identify the context. For reference, Figure 5.13(a) shows the discretized output of the strain-gauges overlaid on accelerometer data.

When the strain gauges identify the robot as being in flight or in stance, the modified GPB2 only activates the flight or the stance filter. However, when RHex is about to touch down or to lift off, the strain gauges are unable to identify the mode, so all filters are activated and the GPB2 resumes nominal operation. Since the original GPB2 algorithm is able to accurately estimate the height of RHex while jogging at steady state, the modified algorithm does not noticeably increase its accuracy[3]. However, the modified algorithm does improve the scalability, as evidenced by Figure 5.7. The plot of mode probabilities shows that when the dynamical context is identified, only the accurate filter is activated. Conversely, during transitions from one dynamical context to another, the nominal GPB2 is operated with all filters

[2]The specific values of $T_{i,j} = 0.7$ and $T_{j,j} = 0.3$ are determined empirically.

[3]The next section shows that when the robot operates outside the steady region, the modified algorithm provides appreciable accuracy gains.

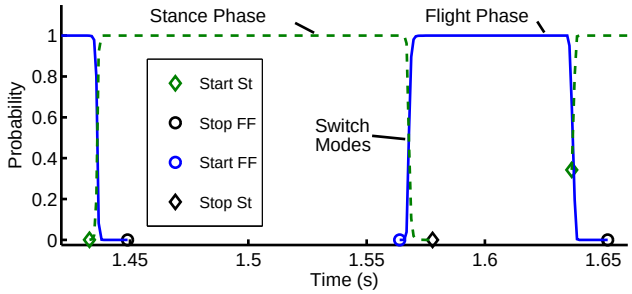


Figure 5.7: Plots of $Prob_i$ for RHex while jogging using the modified GPB2 algorithm. When the dynamical context is identified, only the accurate filter is activated, which reduces the computational cost.

active. It is worth noting that in comparison to Figure 5.5(b), Figure 5.7 shows that the probability estimates are no longer corrupted by accelerometer noise.

An interesting exercise is to try to identify RHex's dynamical context without using contact sensors or strain gauges. As described in Section 4.2, this can be achieved by classifying the accelerometer values of Figure 5.1 into the flight and stance dynamical contexts as follows:

- Flight context: $\ddot{z} < -6m/s^2$
- Stance context: $\ddot{z} > 0m/s^2$
- Unidentified context: $0 > \ddot{z} > -6m/s^2$. Accelerometer output within this context indicates that the robot is transitioning between flight and stance. Context bounds are derived from empirical observation.

State estimates obtained with this contextual classification are virtually indistinguishable from the previous results. The mean and standard deviation of the RMS error over 12 experiments are $2.09cm$ and $0.39cm$, respectively, which compare favorably to the values obtained with the help of strain gauges. This experiment shows by example that the classification approach proposed in Chapter 4 yields satisfactory estimation results.

5.2.2 Behavioral context identification allows reduced-scale MM systems

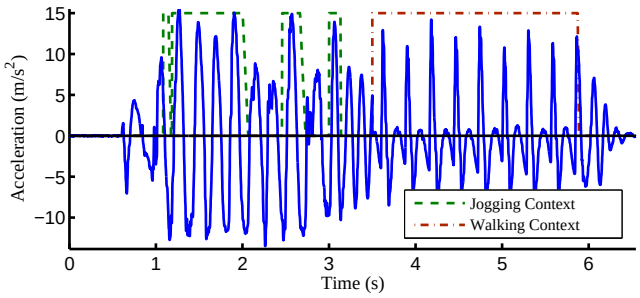
As defined earlier, behavioral contexts correspond to a set of specific dynamical contexts and to specific frequencies and order of transition among these dynamical contexts. It follows that identifying the current behavioral context also determines the set of dynamical contexts that transition among one another. This enables the construction of MM systems that consist only of the individual filters corresponding to the identified set of dynamical contexts. The resulting MM system is likely to be of modest size, and therefore of modest computational cost. Scalability is improved by deploying multiple such low-cost MM systems, instead of a single large-scale system.

This concept is illustrated with the example of identifying RHex's height while the robot jogs *and* walks. Experimental data does not exist for such a run, so the accelerometer output for the jogging and walking behaviors are combined into a single, continuous acceleration profile, where the first half corresponds to jogging and the second half to walking[4] (this setup is first mentioned in Section 4.3.6). Since this is a proof-of-concept exercise, suppose that control information is unavailable, so the estimator has to rely on behavioral context identification to recognize whether the robot is walking or jogging.

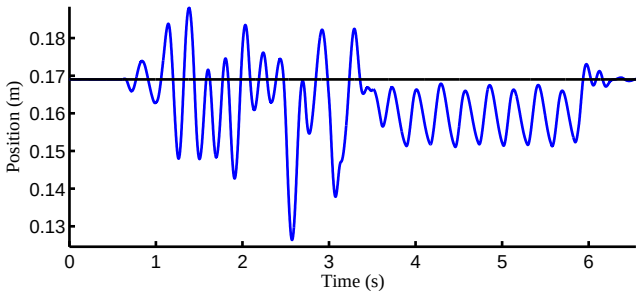
If behavioral information is not available, this two-behavior experiment would require a three-model conventional MM system; two models for jogging and one for walking (refer to Section 5.1 for details about the models). However, information about the behavioral context is available, and the results of Chapter 4 can be directly used here; when the jogging context is identified, then the flight-stance GPB2 of the jogging behavior is used; when walking is identified, the single KF of the walking behavior is used.

Figure 5.8(a) shows the behavioral context overlaid over the acceleration data. The

[4]This setup does not reduce the validity of the experiment. The identification of the behavioral context is performed exactly as it would be if the robot was actually executing both behaviors.



(a) Context information overlayed over acceleration data



(b) Context-based height estimates

Figure 5.8: The identification of the dynamical context enables the selective deployment of a jogging GPB2 and of a walking KF. The resulting height estimates over the combination of jogging and running behaviors have a similar accuracy as when the two behaviors are executed separately.

first half of the acceleration is produced during jogging, and the HMM-automaton mechanism correctly identifies the jogging context. Similarly, the walking context is identified over the second half of the plot.

The result of this context-based state estimation are shown in Figure 5.8(b). As expected, the first half of the plot resembles Figure 5.4, and the second half resembles Figure 5.6. Thus, the accuracy of the estimates and the computational complexity of the filtering system are consistent with the single-behavior results, even though they are conducted on a multiple-behavior system. The maximum number of filters used whenever the behavioral context is identified is four, which is not higher than when the robot executes jogging as a sole behavior.

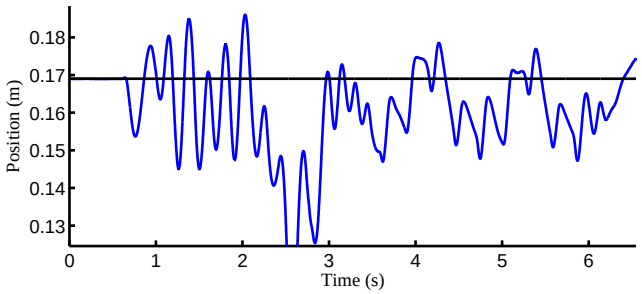
For reference, a conventional three-model GPB2 that does not use contextual information is implemented and its results are reported in Figure 5.9. This MM system activates nine KF at all times, and outputs less accurate height estimates than when using contextual information (subfigure (a)). The lower accuracy is due to the fact that the jogging behavior's stance model and the walking behavior's model have overlapping dynamical contexts. In other words, the GPB2 is unable to correctly disambiguate its models because two of them are based on similar mass-spring systems whose acceleration predictions overlap intermittently. Subfigure (b) shows that probabilities of the walk and stance models oscillate between zero and one throughout the run, indicating that erroneous individual estimates are consolidated into the overall estimate, thus reducing its accuracy.

In summary, Sections 5.2.1 and 5.2.2 demonstrate by example that the identification of a system's dynamical and behavioral contexts improves the scalability of multiple model estimators.

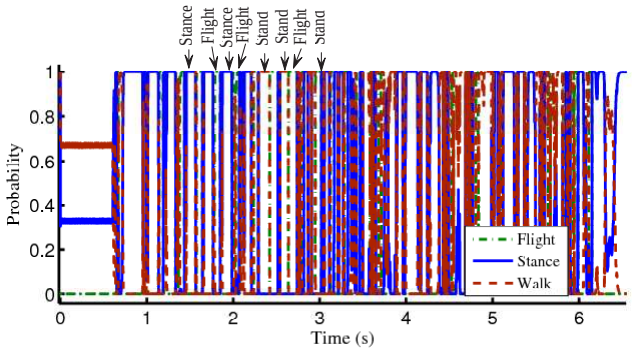
5.3 Context Identification Improves Accuracy with Simple Models

Using simple models to estimate the state of systems with complex and intermittent dynamics can yield accurate results provided the models are only used when they are accurate This condition is fulfilled when estimating the height of RHex during steady state jogging, because the rate at which the dynamics change does not exceed the convergence rate of individual filters. As individual filters converge, GPB2 estimates of mode probabilities also converge, thus enabling accurate estimation.

Unfortunately, these conditions do not hold over the entire experiment. Outside the region of steady-state operation, higher frequency dynamics reduce the time



(a) Height estimates



(b) Mode probabilities

Figure 5.9: The three-model GPB2 activates nine filters at all times and outputs lower-accuracy estimates than obtained with information about the behavioral contexts.

that individual filters have to converge. As the time shortens, individual filters diverge, which causes the consolidated estimate to diverge as well. This is one of the MM estimation problems described by Chapter 2.

This section analyzes in detail the shortcomings of the GPB2 presented in Section 5.1 when it is applied to the entire jogging experiment. Experimental results show that contextual information improves the estimation performance by enabling the rapid identification of accurate models.

5.3.1 Analysis of estimation failure over entire jogging run

The jogging GPB2 algorithm implemented with success in Section 5.1 diverges when the motors are commanded to stop, which subjects the robot to stopping dynamics. The dynamics are reproduced in Figure 5.10(a), and it can be seen in subfigure (b) that the height estimation is accurate over the steady-state region, and rapidly diverges thereafter. To understand the mechanism of this failure, a look at subfigures (c) and (d) shows that the conventional GPB2 erroneously estimates that the flight probability is high (in fact close to one) at the end of the motion. This means that the GPB2 is associating the largest weights to the output of the filters based on the flight model, which explains the estimates' downward slope.

This behavior is counter intuitive as the stopping dynamics exhibit a rapidly disappearing flight phase and would be far better described with the mass-spring model of the stance phase. How did the GPB2 make the wrong probability assessment, particularly that the assessment is based on the relative comparison of each model's accuracy? The most likely answer is found in Figure 5.11, which plots the acceleration around the time the motors stop at second 3. The leg-strain gauges show that the robot is in the tripod stance before the motors stop (diamonds), and in six-legged standing mode afterward (stars). The plot reveals that the acceleration's period of oscillation after stopping is shorter than the period before. This means that the GPB2 has less time to converge before the dynamics switch between stand and flight modes, and indeed this time proves too short for convergence.

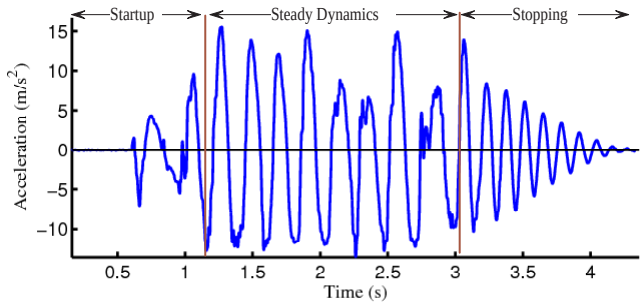
The problem of a short window for convergence is compounded to the inherent brittleness of the stance model whose prediction accuracy is state-dependent. State dependence means that if the previous estimates of the state are wrong, then the current model predictions are also wrong. This is explained as follows. First, note that the stance model $\ddot{z} = -K_z (z - z_0)/M - g$ is responsible for restoring the height back to its rest value z_0 before the next flight, so that height estimates oscillate around z_0 as they do in steady state. After the motors stop, the period of oscillation decreases, which reduces the the time spent in stance before the next flight phase takes place. With less time to complete the stance, the height does not quite reach z_0 before flight, and this problem occurs at each step leading to an accumulation of height loss. This can be seen in Figure 5.10(c), where after second 3 the GPB2 still alternates flight and stance probabilities, but the duration of the stance probability is about twice as short as during steady state.

At the same time, height estimates start to decrease, as seen in subfigure (b). As the estimates decrease, the deflection of the virtual spring $(z - z_0)$ increases, and the stance model outputs increasingly erroneous (and negative) acceleration predictions, until those predictions become consistently lower (and hence less accurate) than flight predictions of $-g$. At that point, the GPB2 assigns the highest probability to the flight mode (subfigure (d)), and the estimation fails.

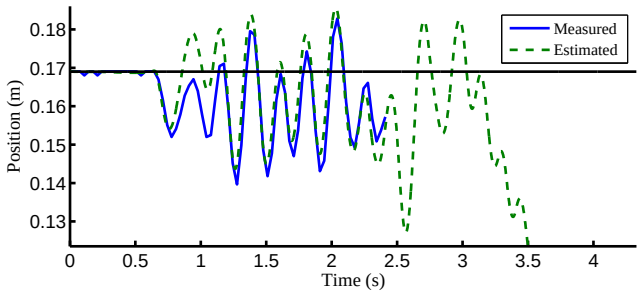
This analysis suggests that the problem can be addressed by making the filters converge faster, and by using a stand model that describes the stopping dynamics more accurately than the stance model.

5.3.2 Adding a mass-spring-damper model

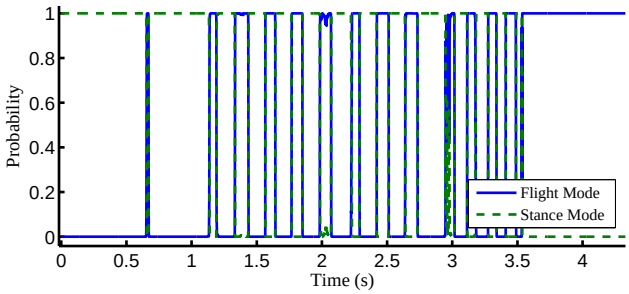
A first step towards implementing the solution consists of developing a stand model to describe the stopping dynamics. As noted earlier, the amplitude of RHex's acceleration decreases gradually after the motors stop, as a result of energy dissipation from the compliant legs. The period of oscillation of the acceleration also decreases, as a six-legged stance has twice the spring stiffness than tripod stances. It follows



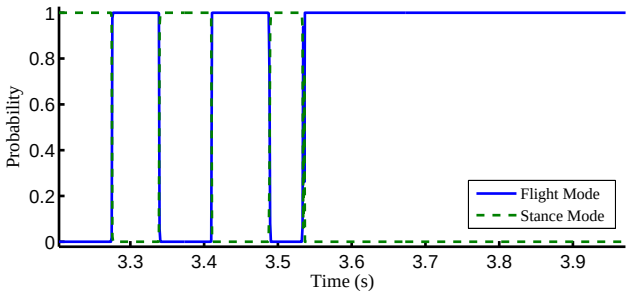
(a) Acceleration plot reproduced for reference



(b) GPB2 height estimates



(c) GPB2 probabilities for the flight and stance modes



(d) Zoom on probabilities shows that GPB2 assumes robot in flight at end of motion

Figure 5.10: Height estimation based on simple flight and stance models are inaccurate outside the range of steady state operation.

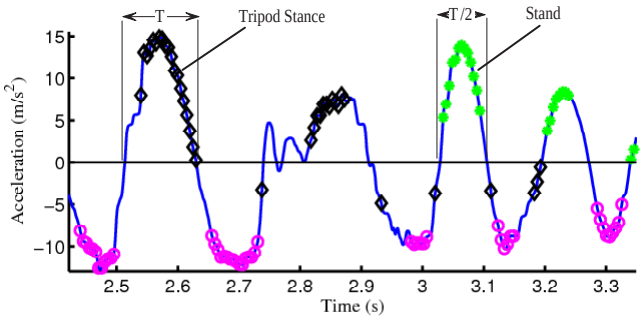


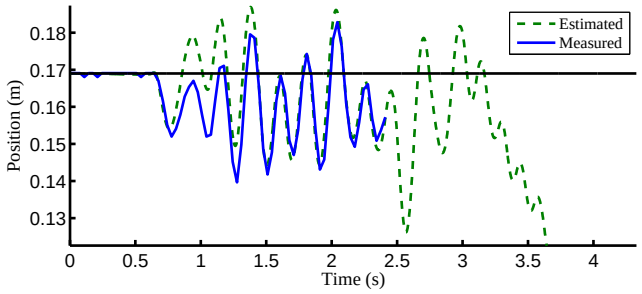
Figure 5.11: The period of the dynamics decreases after the legs stop rotating, as can be seen in the zoomed-in view of the acceleration just as the jogging motion stops. The change in period is explained by the fact that stopping the motors make the robot bounce on six legs as opposed to three legs during the alternating tripod motion. The output of leg strain gauges (circles for flight, diamonds for tripod stance and stars for six-legged stance) is overlaid on the accelerometer data

that RHex's stopping dynamics could be modeled as a dissipative mass-spring system, a model similar to stance but with damping and twice the spring constant. This leads to the following model set:

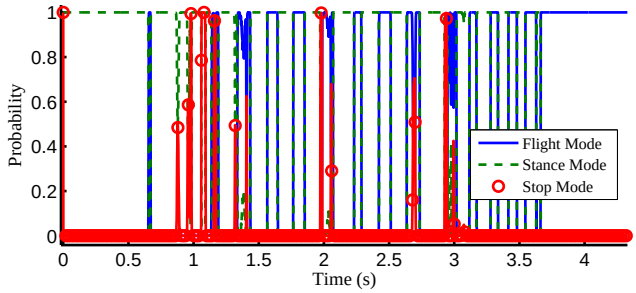
$$\ddot{z} = \begin{cases} -g, & \text{flight phase,} \\ -K_z\left(z - z_0\right)/M - g, & \text{stance phase,} \\ -2K_z\left(z - z_0\right)/M - g - (D/M)\dot{z}, & \text{stand phase,} \end{cases}$$

where D is the viscous damping parameter.

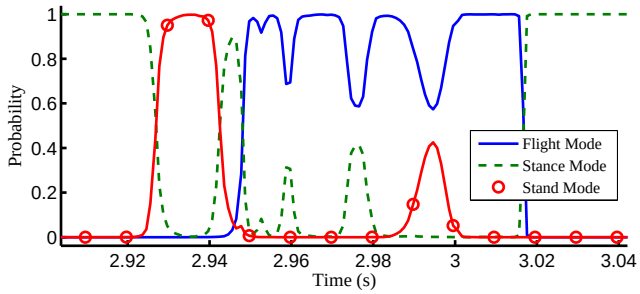
A conventional three-model, nine Kalman-filter GPB2 system is built, and its state estimates are plotted in Figure 5.12. The plots show that this GPB2 still diverges, despite using a more accurate stand model. Efforts to tune GPB2 parameters such as the transition probabilities T_{ij} and model parameters such as D could not improve the results reported. This is due to the fact that the period of oscillation is shorter than the time necessary for the individual filters to converge. As before, and despite using the more accurate stand model, the accumulation of estimation errors forces the conventional GPB2 to wrongly assign high probabilities to flight.



(a) Height estimates using different models for stance and stand modes



(b) Probabilities for the flight, stance and stand modes



(c) Zoom on probabilities shows that GPB2 assumes robot in flight at end of motion

Figure 5.12: Adding a model for the standing motion does not allow the three-model GPB2 to converge.

5.3.3 Identifying dynamical context enables fast convergence

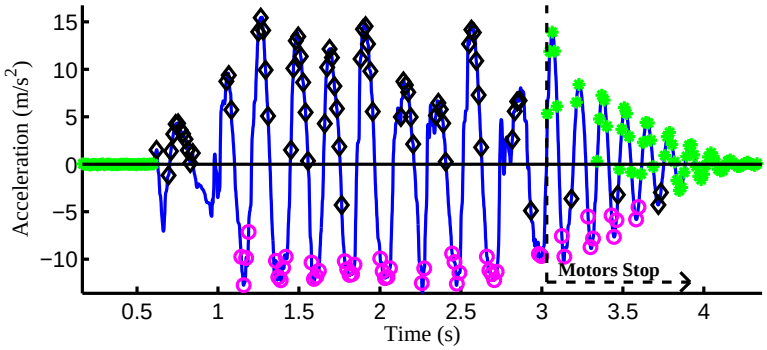
The problem of slow convergence with conventional MM systems can be resolved by identifying the flight-stance-stand contexts from the leg-strain gauges acting as contact sensors (see Figure 5.13(a)). When the strain gauges identify the context, the three-model GPB2 is reduced to a one-model filter (the filter corresponding to the identified context) that outputs accurate estimates. Refer to Section 3.1.3 for details on the implementation of this algorithm. Thus, by incorporating contextual information, the mode probabilities are correctly and rapidly specified, which enables the successful estimation of the height throughout the experiment.

The results are reported in Figure 5.13, where subfigure (b) shows that the height estimates follow an oscillatory dissipative motion ending at the robot's rest height of z_0. The ground truth measurement system's limited workspace does not allow the quantitative validation of the estimate's accuracy, but the fact that the estimates have the same frequency as the measured acceleration and that they converge to the rest height as expected is an indication of validity.

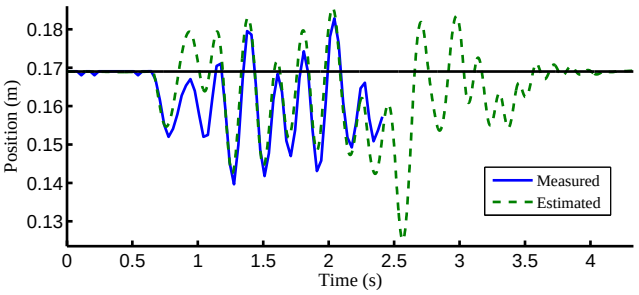
Subfigure (c) shows that towards the end of the motion the GPB2 correctly assigns the highest weights to the stand mode, in contrast to the results obtained without context information.

5.3.4 Identification of behavioral context prevents divergence

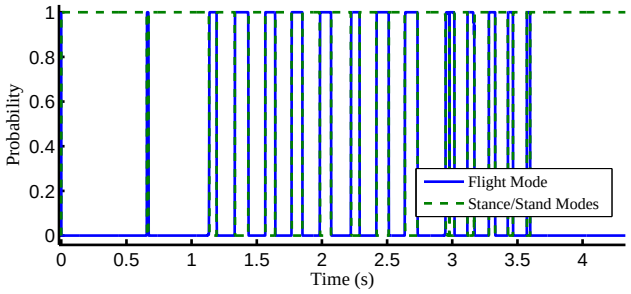
It would be a worthwhile exercise to perform the same estimation task but by extracting context information only from acceleration data, i.e. without using the information provided by the strain gauges. For this, assume that the robot can be in one of two modes: either jogging or coming to rest. Furthermore, since this experiment is intended as a proof of concept, assume that information about the motors being turned off is unavailable. In this setup, the estimation system can use information about the behavioral context to decide whether or not the two-model flight-stance GPB2 can be used. In other words, the behavioral context would



(a) Acceleration plot. Flight is represented with circles, stance with diamonds, and stars represent situations where both tripods touch the ground, which happens when the motors stop.



(b) Height estimate using information about dynamical context



(c) Probabilities for the flight and stance/stand modes

Figure 5.13: Strain gauges identify the dynamical context and enable fast convergence of height estimates.

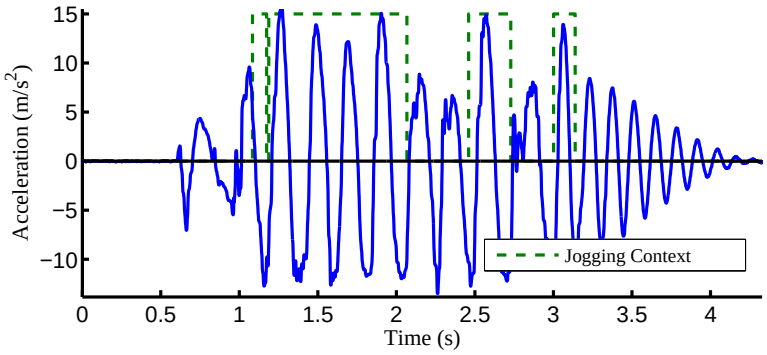
identify the steady state portions of the behavior where the two-model GPB2 can be used with satisfactory accuracy.

By choosing to ignore strain gauge information, the dynamical context cannot be accurately identified in the stand mode. However, information about the behavioral context is available and can identify whether the robot is in steady-state jogging or coming to rest. Thus, the estimation strategy is not to use a combination of flight-stand models to represent the stopping dynamics, but instead to use a single stand model as an approximation. This is an acceptable approximation because the dissipative dynamics rapidly eliminates the flight phase, so the stand can be used alone with limited adverse impact on estimation accuracy.

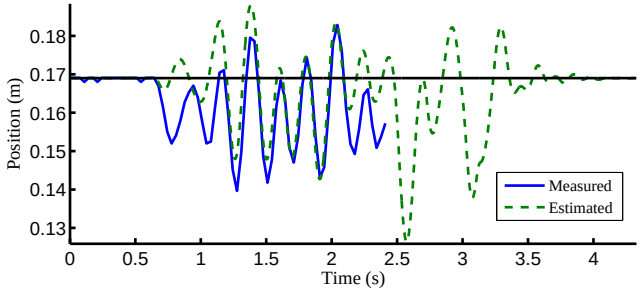
The results are shown in Figure 5.14. Subfigure (a) overlays the jogging context obtained from the HMM-automaton estimation of Chapter 4 over acceleration data. Subfigure (b) shows that even though height estimates are not as accurate as when using the strain gauges, they still converge back to the rest height.

The difference in accuracy is due to the fact that behavioral contexts provide higher-level information about the dynamics than dynamical contexts. Dynamical context provide lower-level, more detailed information about the dynamics, and this enable MM systems to generate accurate estimates, as in the previous example. When only behavioral information is available, as in the current example, the accuracy of the state estimates (Figure 5.14(b)) is lower, but still substantially better than when neither context is used.

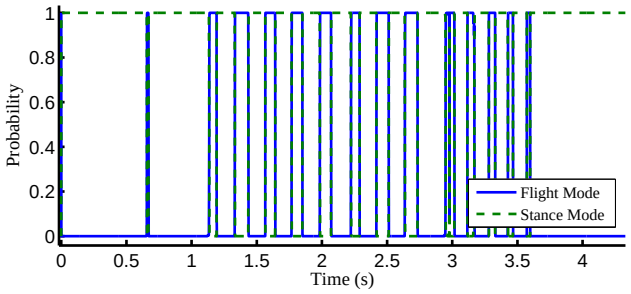
Last, knowing the behavioral context allows the deployment of the two-model instead of the three-model GPB2, thus reducing the number of filters from nine to four and improving the scalability of the estimation system.



(a) Jogging context is overlayed on acceleration plot



(b) Height estimate using information about behavioral context



(c) Probabilities for the flight and stance/stand modes

Figure 5.14: Jogging context obtained through the HMM-Automaton technique of Chapter 4 enables the selective activation of the jogging GPB2. When the robot is not in the jogging context, a spring-damper model describes the robot's stopping motion.

5.4 Dynamical Context Identification Overcomes Limitations of Low-Sensitivity Sensing

The previous section (5.3) studies the problem of loss of estimation accuracy when individual filters do not have sufficient time to converge. This section illustrates another problem presented by Chapter 2, namely the loss of accuracy when available sensors have a low sensitivity to changes in the dynamics.

The problem is demonstrated in a simulated experiment of a bouncing ball (see Section 2.3.3.1 on page 36 and Chapter 3 for an introduction). The setup consists of a unit-mass bouncing ball that alternates flight and stance dynamics as it falls, impacts the ground elastically and rebounds. The ball evolves as a mass-spring system during stance, and as a ballistic projectile during flight. Thus, the dynamics are exactly formulated as follows:

$$\ddot{z} = \begin{cases} -g, & \text{ballistic projectile during flight,} \\ -K\left(z - z_0\right) - g, & \text{loss-less mass-spring system during stance,} \end{cases}$$

where the state z is height, g is the acceleration due to low-magnitude gravity and K and z_0 are the ball's effective spring constant and rest length, respectively. The values of g, K, and z_0 are $\frac{9.81}{10} m/s^2$, $75 N/m$ and $0.5m$, respectively[5].

The task is to estimate the ball's height and vertical velocity using the flight and stance models, and a virtual sensor that generates noisy height measurements. To that effect, a first experiment consists of constructing a conventional, two-model GPB2 that does not use any contextual information. Even though the flight and stance models describe the ball's motion exactly, their associated Kalman filters have non-zero but small process noise covariances Q, and a larger observation noise covariance R, to simulate realistic conditions. The ball is assumed to be equally likely to transition from flight (F) to stance (S) as from stance to flight, so

[5]The gravity vector has a lower magnitude than Earth's gravity to allow for a slow bounce, which is simpler to observe and clearer to present. If the simulation were conducted with the Earth's gravity, it would lead to the same qualitative results, but since the time between touchdown and liftoff is short, the results could not be presented with the same clarity.

$T_{i,j} = \frac{1}{2}, \forall i, j \in \{\text{F, S}\}$.

The simulation is initiated by dropping the ball from a height of ten meters and letting it bounce once (Figure 5.15). The results of this GPB2, reported in Figure 5.16, show that velocity estimates diverge markedly from the true (simulated) velocity (subfigure (b)), and height estimates exhibit small divergence near the point of impact (denoted by the first '+' in subfigure (a)).

The reason for the low accuracy is that filters' observations consists of position measurements which do not provide sufficient information to clearly discriminate between the flight and the stance modes. As a result, the GPB2 wrongly assigns non-zero weights to the stance filters when the ball is in flight, thus including inaccurate estimates to the consolidated state.

This can be seen clearly by analyzing one iteration of the GPB2. Consider the initial state of the ball at the top left of the plot in Figure 5.15, at height $z = 10$ and velocity $\dot{z} = 0$. The flight model predicts an acceleration of $\ddot{z}_S = -g = -0.981 m/s^2$, and the stance model predicts a larger negative acceleration $\ddot{z}_F = \ddot{z}_S - 75(9.5) = -713.5 m/s^2$. If the filters were using acceleration measurements as observations, then this large discrepancy would immediately lead to a low likelihood for the stance filters relative to the flight filters. Indeed, the RHex experiment in Section5.1 shows that when using acceleration measurements, the GPB2 assigns correct mode probabilities and outputs accurate estimates. In contrast, the discrepancy between flight and stance predictions of *position* is small. In fact, this gap is smaller than the acceleration gap by factor equal to the square of the sampling period T, as $z \simeq \ddot{z} \cdot T^2$. The higher the bandwidth of the sensor the smaller the sampling time and the smaller the gap; in this case, $T = 0.01 sec$, so $z \simeq \ddot{z} \cdot 10^{-4}$.

Thus, by relying on position measurements, the innovation[6] of the stance filters would still be larger than the innovation of the flight filters, but only by a small margin. This translates to non-zero relative likelihood for stance filters, which leads

[6] As a reminder, the innovation is the difference between model prediction and sensor observation.

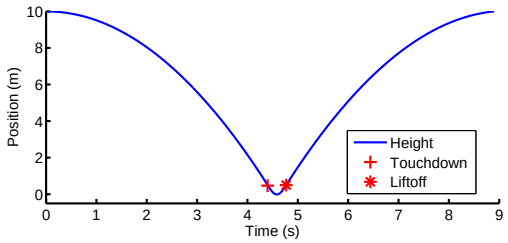
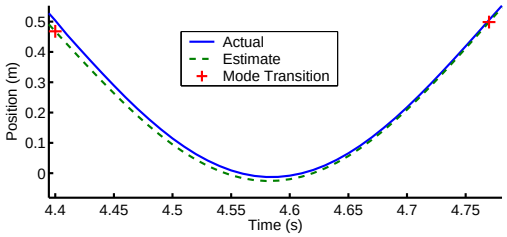


Figure 5.15: The ball falls from a height of ten meters and bounces. The gravity vector has a low magnitude that slows the bounce for presentation clarity.

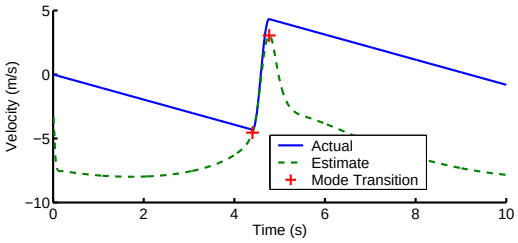
the GPB2 to assigning non-zero probability for the stance mode. Figure 5.16(c) shows that the GPB2 incorrectly assigns non-zero stance probabilities when the ball is in flight (before the first and after the second '+'). This logically decreases the accuracy of the consolidated state because it incorporates stance estimates while in flight. The delays introduced by the low-sensitivity sensing are also visible in the plot, as the GPB2 switches mode probabilities later than when the ball actually switches dynamics.

A remedy to this sensor-induced slow convergence is modify the GPB2 to incorporate contextual information extracted from the sensor. Here, the dynamical contexts are naturally specified as the flight and stance contexts, and their identification is based on the straight-forward observation that when the measured height is lower than the ball's rest height, then the ball is known to be in stance, and otherwise it is in flight. To take into account sensor noise, a transition zone is added in the immediate vicinity of z_0, the touchdown and liftoff points: $z = z_0 \pm \delta$, with $\delta \ll z_0$. In the transition region, the identity of the context cannot be accurately determined from position measurements. This setup is described pictorially in Figure 5.17 and summarized as follows:

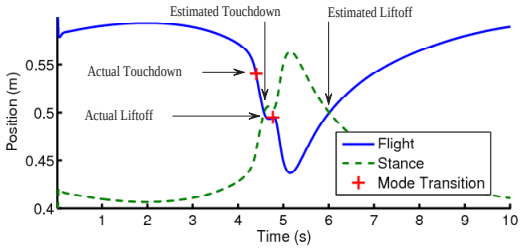
- Flight context: $z > z_0 + \delta$
- Stance context: $z < z_0 - \delta$
- Unidentified context: $z_0 - \delta < z < z_0 + \delta$



(a) Height estimates. The time scale zooms on the bounce for clarity.



(b) Velocity estimates



(c) GPB2 outputs wrong estimates of mode probability

Figure 5.16: State estimates obtained with a conventional GPB2 lead to RMS errors of $6.34cm$ and $15.5058m/s$ for position and velocity estimates, respectively. The low sensitivity of the position sensor to changes in the dynamics does not allow the GPB2 to compute the correct probabilities for the flight and stance contexts. These plots indicate the instants when the ball touches down and lifts off with a '+'. Subfigure (b) shows that the probabilities of flight and stance erroneously cross each other far from the '+' points. Correctly estimated probabilities would cross close to the actual point of touchdown and liftoff.

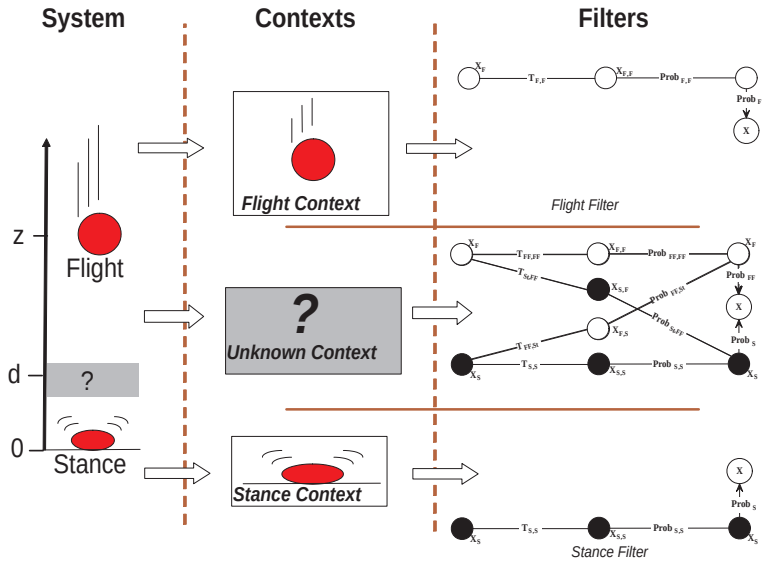


Figure 5.17: Context-based estimation framework for a bouncing ball.

The GPB2 is modified to incorporate the dynamical context information following the description of Section 5.2 and the pseudo code of Algorithm 3. When the ball is recognized to be in flight ($z > z_0 + \delta$) or in stance ($z < z_0 - \delta$), only the corresponding $filter_{\mathrm{F}}(x_{\mathrm{F}})$ or $filter_{\mathrm{S}}(x_{\mathrm{S}})$ is updated, as shown in Figure 5.17. When the context is unidentified ($z_0 - \delta < z < z_0 + \delta$), all four filters are updated. The transition probabilities are adjusted so that when the ball is descending ($\dot{z} < 0$), the probability of transitioning into stance is greater than the probability of staying in flight (e.g. $T_{\mathrm{S,F}} = 0.7;\ T_{\mathrm{F,F}} = 0.3$). The inverse is true when the ball is ascending ($T_{\mathrm{F,S}} = 0.7;\ T_{\mathrm{S,S}} = 0.3$).

Figures 5.18(a) and (b) show that this strategy significantly improves the accuracy of the estimated velocity. RMS errors are reduced from $6.34cm$ to $0.13cm$ and from $15.5m/s$ to $0.0119m/s$ for position and velocity, respectively. Thus, by ignoring the output of inappropriate filters, the consolidated estimate can maintain its accuracy as contexts alternate. The plot of subfigure (a) displays the effect of using a GPB2 during context transitions. Consider the top left portion of the plot; the first star indicates the moment when all four filters of the GPB2 are activated as

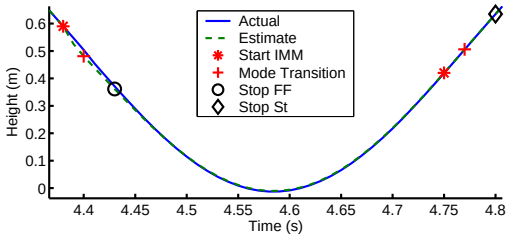
a result of the ball reaching the region of uncertainty; and the first circle indicates the moment where the GPB2 is deactivated as the stance context is successfully identified. As long as the ball is in the transition region, the output of the four filters are consolidated, and the estimate is somewhat less accurate than before the activation of the GPB2 or after it is turned off. This is to be expected, as the GPB2's sub-optimal weight distribution lowers the accuracy of the consolidated state. Subfigure (c) shows that the GPB2 mode probabilities represent the simulated reality more closely, with estimated mode transitions taking place near the points where dynamics switch. The plot also shows the selective activation of accurate filters during flight and stance as well as the transition point from one mode to the other.

5.5 Identification of Dynamical Context Exploits Discarded Sensor Information

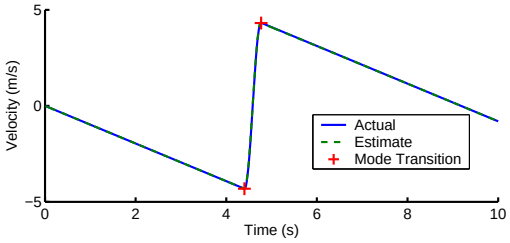
The previous sections describe context-based techniques that overcome the accuracy and scalability limitations of multiple model systems. This section does not provide new insight about the techniques themselves; rather, it presents a simple experimental result that demonstrates how sensor information discarded by simple motion models can be used to improve the output accuracy of estimation filters.

The experiment consists of identifying a mobile robot's dynamical context from sensor information, and using the context's identity to regulate an estimation filter's process covariance. Results show that such regulation improves the output accuracy of the individual estimation filter.

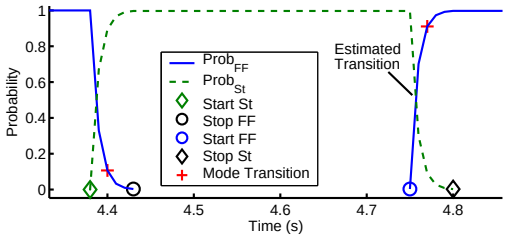
A popular approach for mobile robot localization is the simultaneous localization and mapping (SLAM) technique, which consists of estimating the robot's pose (localization) and the location of feature landmarks in the environment (mapping) at the same time. In its simplest form, SLAM consists of a Kalman filter where the first entries of the state contain the robot's own location variables (such as



(a) Position estimates



(b) Velocity estimates



(c) Mode Probabilities

Figure 5.18: The GPB2 performance increases when knowingly inaccurate estimates are discarded. RMS errors for position and velocity estimates are down to $0.13cm$ and $0.0119m/s$, respectively.

x, y, θ coordinates), and the last entries contain the coordinates of landmarks in the environment. The KF is initialized with a starting pose of the robot and an initial estimate (or even guess) of the landmarks' coordinates. As the robot traverses the environment, its motion model provides the filter with predictions of the robot's own pose *and* of the observations, commonly defined as range and bearing measurements to the landmarks. Next, actual measurements of the landmarks are used to update the filter, thus generating estimates of both the robot's pose and landmarks location [72, 41, 13].

SLAM experiments are almost always conducted on rovers traversing mostly flat terrain and consequently motion models are predominantly planar. Such models would describe the robot's motion in $\mathbb{SE}(2)$, namely the evolution of its $x-y$ position and θ orientation over time. The simplicity of these models makes them and the Kalman filters tractable to design and run. However, such filters do not model motions or events along non-planar dimensions, even though such events may fundamentally change the robot's behavior and render the planar motion model inaccurate. By ignoring these events, SLAM algorithms may inadvertently trust faulty models and diverge.

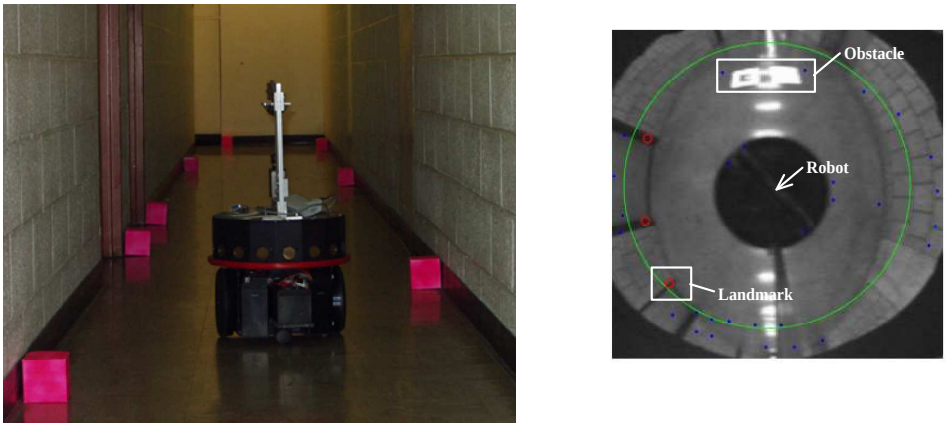
One example of such events is high-centering, a situation where the rover's undercarriage gets stuck on an obstacle, rendering the wheels tractionless. If such an event goes unnoticed, then a SLAM algorithm would maintain an inappropriately high level of confidence in an inappropriate motion model. Evidently, the ramifications of this problem extend to control, as illustrated by a real-world experiment where a rover did not recognize that it had become high-centered and the controller kept spinning the wheels until a tire caught fire [22]. Deciding on appropriate recovery actions is beyond the scope of this thesis, however the SLAM algorithm can be modified to maintain a degree of estimation accuracy. This can be done by first identifying the occurrence of an event that decreases the motion model's accuracy, and second by increasing the covariance of the process noise in the KF so as to reduce the filter's confidence in that model's predictions.

This strategy is implemented in experimentation on a laboratory mobile robot (see Figure 5.19(a)). The setup consists of having the mobile robot navigate a corridor while performing SLAM. Subfigure (b) shows a view from the onboard camera. The squares in the image are the landmarks whose coordinates are registered in the state vector, and the white square in front of the robot is an obstacle that the robot does not recognize as such. When the robot hits the obstacle, it becomes high centered and its wheels loose contact with the ground. However, the SLAM algorithm is based on a planar model and does not recognize the fact that the wheels, though spinning, no longer have traction. Thus, the KF maintains the same confidence in the motion model as when the robot was moving. This means that model predictions indicate that the robot is moving forward, and consequently the filter assumes that it is discovering new landmarks with time. This is a wrong assumption as the new landmarks are in fact the same landmarks observed repeatedly from a fixed location. This is a phenomenon commonly referred to as landmark hallucination, and it leads to the divergence of the location estimation.

The divergence is seen in Figure 5.20(a), where the last two landmarks are thought to be new, whereas they really are the same as the last landmarks seen. The plot shows that the filter mistakenly estimates that the rover kept moving forward and even made a turn, overshooting its real position by $58.5cm$.

To address this problem, the strategy outlined above is implemented. When the robot hits the obstacle, the impact could be detected either by a vertical accelerometer that registers a spike, or by a dedicated contact switch placed beneath the undercarriage that gets triggered upon impact. In either case, the output of the sensor can not readily be integrated into the robot's planar motion model, which motivates the development of an external mechanism that appropriately modifies the filter parameters. In this experiment, the robot lacks such specialized sensing, so the trigger is initiated manually. Nevertheless, this experimental limitation does not undermine the validity of the algorithmic results.

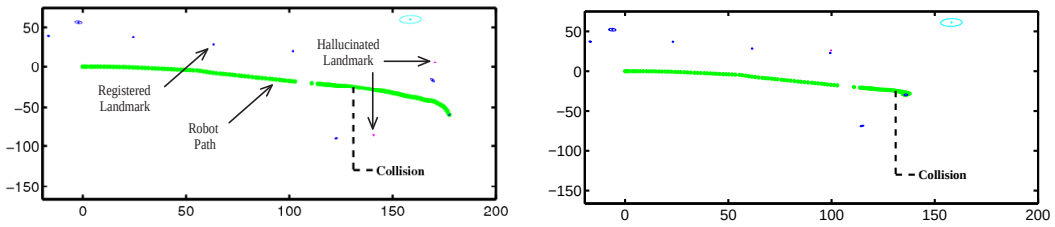
Upon impact, the magnitude of the process covariance Q is increased by three



(a) Laboratory mobile robot

(b) Sample image from onboard panospheric camera. Landmarks are denoted in red circles.

Figure 5.19: Laboratory mobile robot conducts SLAM experiment and encounters obstacle (white square in subfigure (b)), which makes the robot wheels loose traction and reduces the accuracy of the robot's motion model. Images courtesy of Hyungpil Moon.



(a) Robot position estimated with conventional SLAM

(b) Robot position estimated with SLAM whose process covariance is regulated.

Figure 5.20: In this experiment, regulating the covariance of the process noise to account for the loss of motion model accuracy improves the performance of the SLAM position estimates by 88%.

orders of magnitude, from $5times10^{-4}$ to $5times10^{-1}$. This modifies the values of the gain K such that the updated output would emphasize more the sensor measurements than the model predictions. In normal circumstances, such a setup would make the estimation system excessively vulnerable to sensor noise. In this situation, model noise increases dramatically with the loss of prediction accuracy, and sensor noise becomes the lesser evil. Furthermore, sensor noise is reduced because of repeated measurements of the same landmarks. The results, presented in Figure 5.20(b), show that the position estimates are closer to the real value. The overshoot of the real position is reduced to $7cm$, an 88% performance increase.

The estimation strategy used in this experiment is a departure from the direction adopted thus far, in that context information is used to improve the accuracy of a single filter rather than a multiple-model system. As such, this strategy constitutes a preliminary development of context-based model adaptation, where a filter parameter (process covariance in this case) adopts discrete values depending on the robot's dynamical context.

This approach could be further developed to enable, for example, continuous adaptation. A first step in this direction is described in the Appendix, which proposes a regulation technique for one-dimensional systems. Chapter 6 offers other directions for constructing more elaborate adaptation techniques within the context-based framework.

5.6 Summary

The examples in this chapter illustrate specific instances of failure of multiple model state estimation. They are provided to ground the problem description of Chapter 2 in practical terms, and to demonstrate the value of the context-based estimation

strategies laid out in Chapter 3.

These strategies are shown by example to improve the scalability of multiple-model filters by reducing the number of active filters within one MM system, and by allowing the deployment of multiple small-scale MM systems.

State estimation accuracy is also shown to improve with the use of contextual information. The ability to recognize the robot's dynamical context allows an MM system to rapidly converge on the correct weights with which to scale the output of individual filters. This permits accurate height estimation for RHex while jogging, even when the dynamics alternate more rapidly than the convergence rate of the GPB2's individual filters.

Beyond the answers provided by this chapter to some of the problems of multiple-model estimation, open questions remain as to extensions of the approach that may allow it to address a broader range of problems. An in-depth discussion of the merits and limitations of the approach and its potential for future developments is presented in the concluding Chapter 6.

Chapter 6

Conclusion

The context-based estimation framework enables accurate and scalable state estimation for systems with hybrid dynamics. This thesis develops a novel approach of combining discrete and continuous estimation techniques to generate accurate estimates for systems with intermittent dynamics, in situations where conventional continuous-only approaches fail. The thesis also introduces the concept of explicitely tying motion models to the dynamics they represent, formalizes the concept as a hierarchy of dynamical and behavioral contexts, and demonstrates that this formalism achieves significant accuracy and scalability gains for multiple-model filtering systems.

Precise definitions of the dynamical and behavioral contexts are provided in Chapter 3, and the methods by which they are identified are presented in Chapter 4. Chapter 5 shows how identifying a robot's dynamical and behavioral contexts can improve the scalability and the accuracy of multiple-model estimation systems, and provides preliminary results of extensions to the framework that enable single-filter adaptation.

As the presentation comes to an end, it is worth revisiting the original objectives of the work and examining the research results to evaluate the extent to which it meets these objectives. The research is analyzed on the philosophical level to assess the value of the conceptual approach; on the methodology level to study

the degree to which the framework can be generalized; and on the implementation level to understand the framework's practical constraints.

This analysis naturally leads to open questions about ways to expand and improve to the context-based framework, so this chapter also highlights promising directions for future work. The chapter concludes with a brief description of potential applications in the field of mobile robotics.

6.1 Research Objectives

The original motivation behind this research came from the difficulty of estimating the state of RHex. The robot's complex and intermittent dynamics meant that accurate estimation could only be achieved by either developing accurate but complex motion models, or by instrumenting the robot with specialized sensors. Complex models developed in independent work proved unusable (see Section 4.1.1 for details), and specialized sensors in the form of leg-strain gauges proved complex to install and required careful calibration [44, 43]. This motivated the development of a simpler-to-implement estimation framework that relies on models and sensors of moderate complexity and still generates accurate estimates.

This framework is based on the realization that complete and complex motion models can be replaced with a collection of simple models, each with a limited domain of accuracy. Such a strategy can be implemented with available multiple-model estimation systems, which are also well adapted to robots with hybrid dynamics such as RHex. Unfortunately, multiple model systems suffer from scalability limitations and their accuracy generally decreases with the complexity of the dynamics.

Therefore, the objective of this thesis was to develop a state estimation framework that improves the accuracy and scalability of multiple model systems, with the intent to generate accurate estimates with simple models and low-cost sensors.

6.2 Discussion of Philosophical Approach

The adopted solution is to simultaneously estimate discrete and continuous states, whereby discrete states (contexts) are extracted from direct analysis of sensor data, and continuous states are estimated by MM systems. As demonstrated by the examples of Chapter 5, information about the contexts helps MM systems identify accurate models and selectively activate individual filters, thus improving both their accuracy and scalability.

Philosophically, this approach is well suited to hybrid systems whose dynamics make discrete and sudden changes. Discrete-state estimation can be designed to efficiently and reliably detect such discrete changes, and communicate this information to MM systems. The approach also uses discrete states to represents different dynamics that may not be separated by discrete transitions, but are distinct enough to be described by different models. Therefore, the approach is also suited to non-hybrid systems whose dynamics are complex enough that a collection of models is needed to represent them.

Naturally, the performance of discrete state estimation depends on the quality of available sensors and estimator design, but the general simplicity of discrete estimators provides for design flexibility. This flexibility allows designers to encode knowledge and intuition into the construction of the estimators, thereby increasing the likelihood of achieving efficiency and reliability. For example, knowing that accelerations close to $-9.81 m/s^2$ are representative of flight can easily be formalized in the form of a flight dynamical context; likewise, knowing that a ball at high altitude cannot be in stance allows the creation of simple dynamical contexts.

The explicit estimation of discrete states is a departure from the conventional use of MM systems that infer the discrete state from the relative accuracy of the continuous filters. The reasoning developed throughout this thesis and the experimental and simulation results indicate that this approach indeed provides gains of accuracy and scalability at low additional cost.

Discrete-state estimation is also a low-cost tool to handle infrequent special cases that invariably arise in fielded systems. Such cases would be onerous to handle through continuous modeling, but can be simply accounted for by discrete models. For example, consider when a rover collides with an obstacle, hopefully an infrequent event that can be treated as a special case; sensor-based discrete estimation can easily detect the large dynamics produced by the collision, perhaps by observing accelerations that increase beyond a predetermined threshold. In contrast, continuous models would have to describe the collision dynamics to detect them, which is clearly a much harder task.

This analysis suggests that expanding some effort to estimate a system's discrete state reduces the overall effort of estimating its continuous state. Therefore, developing discrete estimators for tasks as a means of improving the performance of continuous estimators is likely to be a worthwhile strategy for most field systems.

Estimating a system's discrete state can also provide valuable information about the dynamics that continuous estimators are not well suited to capture. Such information can take the form of an abstracted description of the dynamics (is the robot running or jogging?) or a general quality assessment of a system's operation (unexpected engine noise indicates a failure). This information would not only help robots monitor their health and performance, but also enable faster and more reactive motion control, as described in Section 6.5.

6.3 Discussion of Estimation Methodology

Using hidden Markov models (HMMs) and timed automata for discrete-state estimation combines the advantages of both techniques; HMMs provide robustness to sensor noise and timed automata capture in-state dwelling time. Recall from Chapter 4 that the HMM estimation approach is a discrete-state analog to the Kalman filter (KF), and as such sensor observations are weighted against (discrete) model predictions, which limits the impact of noise. However, the analogy to KFs also

means that HMMs are recursive processes that cannot measure in-state dwelling time .

Timed automata are able to capture in-state dwelling time because, unlike HMMs, they are not restricted by the Markov assumption of independence among consecutive state estimates. However, automata describe deterministic processes, which makes them vulnerable to input (sensor) noise. The advantages of both techniques are combined through their sequential use, with HMMs filtering sensor noise first and automata computing dwelling time second by relying on robust input. Experimental results indicate that this process extracts necessary information from sensor data to enable successful context identification. The validity of this approach is further underlined by similar and parallel developments in the speech recognition community.

Describing the evolution of discrete states with Markov processes also opens the possibility of leveraging a large body of work in discrete estimation to improve context identification. Examples include, but are not limited to, automatic learning of HMM parameters from labeled data or representing the discrete processes by more sophisticated estimation tools such as Markov decision processes. Looking forward, Section 6.5 proposes expansions of the present work through the deployment of such estimation strategies.

The choice of Kalman filters for continuous estimation is motivated by the ubiquity of the filter in the estimation community. Indeed, the context-based estimation framework builds on extensive research on multiple-model estimation techniques based on the Kalman filter, and in return, provides tools that the community could use to improve the performance of such MM systems.

Different continuous estimation approaches could have been used to implement the concepts developed in this thesis. A notable example is the particle filters (PF) framework, which differs from KFs by the type of function approximation it uses. The philosophical approach of combining discrete and continuous estimation applies equally well to KFs and PFs, so the contexts would be identified as described

by Chapter 4 regardless of the method used. However, the framework's implementation for PFs would differ, so new techniques would be necessary to integrate the contextual information to particle filters. Such development is beyond the scope of the current investigation.

6.4 Discussion of Framework Implementation

The implementation details of the context-based estimation framework merits a close look to understand its practical advantages and limitations. For the demonstrations of Chapter 4, the HMM parameters[1] were tuned manually in an effort to contain design complexity. The disadvantage of this approach is that the HMM produces quasi-binary probability distributions over the states ($\alpha_k(j)$); i.e. in most cases, any given HMM state has either a value close to one or close to zero. This limits the framework's ability to express a continuous measure of confidence in the contexts' identity. Had such continuous metrics been available, they would further improve the accuracy of MM state estimation by enabling the evaluation of "distances" among contexts, and perhaps the prediction a robot's entry to or exit from a context. The current inability to compute continuous confidence measures is due to suboptimal parameters values set by hand; this limitation can be overcome through parameter learning, as proposed in Section 6.5.

Another implication of the framework implementation is the potential brittleness of the automata, due to its sensitivity to sensor noise. However, as noted in Section 6.3, the risk of noise-induced failure is greatly reduced by employing an HMM first to filter sensor noise (thanks to its probabilistic representation), instead of using sensor data directly with a deterministic model. In addition, a potential failure of the discrete estimation system does not necessarily lead to the failure of the continuous state estimation. In the (hopefully rare) cases where sensor noise leads to inaccurate automata estimates, the immediate effect is to acknowledge that the

[1] These parameters are the transition and observation probabilities, a_{ij} and $b_j(o_k)$. See Section 3.2.2.1 on page 58 for details.

context cannot be identified. As a consequence, the context-based MM system is forced to resume nominal operation by activating all available filters, until the discrete estimator succeeds in identifying the context again. During this time, the estimation performance is reduced to that of conventional filtering, with no additional harm caused by automata failure. In other words, the success of context identification improves the accuracy of conventional estimation, but its failure does not reduce it.

An important practical dimension of the framework is the methodology to be followed for knowing when to implement the context-based estimation strategy. This question has been addressed partially even if implicitly throughout the document, particularly in light of the examples of Chapter 5. However, a complete and general answer, if it exists, is beyond the scope of this thesis. Nevertheless, a common-sense analysis of state estimation performance tradeoffs provides some guidelines.

Putting the goal of high accuracy aside, the context-based framework almost invariably improves the scalability of multiple model filters. Therefore, the context-based estimation framework should be deployed if available computational resources are significantly taxed by conventional estimation systems.

In addition, estimation accuracy is rarely consistently acceptable; estimators generally perform the main estimation task well, but do not handle secondary tasks satisfactorily. An example is the estimation of RHex's height, where successful results are obtained with a conventional GPB2 during steady-state operations (primary task), but not during transient operation (secondary task). For these systems, the deployment of the context-based framework provides improvements in accuracy that are likely to justify the associated design overhead.

At times, and in realistic conditions, conventional state estimation systems could also fail completely. For example, collision or accidents can render all available continuous estimators inaccurate, as it is virtually impossible to model such dramatic events. The implementation of the context-based framework can be explicitly designed to account for such situations by detecting abnormal situations when the

contexts can no longer be identified. In this case, the contexts would be constructed to capture values of sensor data in normal situations. When these values are exceeded, the robot departs from the 'normal' contexts and the system recognizes a fault or exception and acts accordingly.

This discussion suggests that the context-based framework should be used in most estimation problems. However, as a design argument, one might ask if it possible to predict when the conventional estimation methods will underperform. It would be difficult to answer this question other than on a case-by-case basis, but some rules of thumb about filter convergence can help frame the answer. For example, conventional wisdom within the estimation community states that for filters to converge, the sensor bandwidth has to be an order of magnitude larger than the system's mechanical frequency. Thus, if available sensors prove too slow for the system at hand, then the speed gains provided by context identification would prove crucial to the success of the estimation task.

Chapter 2 presents other means of predicting estimation failure. If the available sensors have a low sensitivity to changes in the dynamics (such as position sensors), then regardless of their bandwidth, they would not help state estimators converge rapidly.

The philosophical approach of combining discrete and continuous estimation techniques as a means to improve state estimation is demonstrably valid and, to the extent that bibliographical research indicates, has not been proposed elsewhere. On the other hand, the implementation of the new estimation framework is limited to proof-of-concept levels. Nevertheless, the research provides a foundation on which further developments can build to maximize the framework's effectiveness, as discussed in the next section.

6.5 Future Work

The context-based state estimation framework can be used as a starting point for enhancements that improve the accuracy of state estimates and extract additional information from available sensors. Enhancement ideas apply primarily to the discrete part of the framework, but ultimately affect the continuous part as well, as the degree of integration between both parts increases.

6.5.1 Learning HMM parameters

As noted in Section 6.4, a major improvement to the current framework would be the ability to compute more accurate values for the HMM parameters than achieved manually. This problem is well known in the discrete-state estimation community, and a conventional solution is to learn the parameters from labeled data sets. The approach assumes that the HMM states are known *a priori*, and requires the labeling of sensor data, where the labels are the HMM states. For example, labeling RHex's lateral acceleration would be similar to Figure 4.9 on page 97. What is not known *a priori*, however, is the structure of the discrete process, i.e. the transition probabilities among the HMM states and the observation probabilities. Thus, RHex's model before learning would not resemble the Markov chain depicted in Figure 4.10, but rather would have arrows linking each state to all other states, with uniform transition probabilities a_{ij}.

With this setup, the transition and observation probabilities a_{ij} and $b_j(o_k)$ can be learned using algorithms such as the Baum-Welch or the Expectation Modification methods [59]. These are optimization algorithms that refine parameter estimates through the iterative application of cost-minimization functions. The result is typically a set of parameter values that minimizes discrepancies between HMM predictions and observation. Conversations with knowledgeable faculty members at Carnegie Mellon University suggest that the parameters obtained through learning would enable the probability distribution $\alpha_k(j)$ to vary continuously rather than mostly discretely, as it does now.

The learning approach could be expanded and include the discovery of the discrete states themselves [76, 70]. Some HMMs have been successfully designed with this strategy, but it is unclear if systems such as RHex would allow for the automatic generation of accurate HMMs. Nevertheless, this approach is worth investigating, as it could significantly increase the generality of the context-based framework.

6.5.2 Using Markov decision processes

The accuracy of the discrete state estimation can be improved by representing some discrete processes as Markov decision processes (MDPs). MDPs incorporate input information into the computation of the probability distribution, so inputs such as control commands could be used to refine the discrete estimates.

Consider a system that executes different behaviors, such as RHex. The behaviors can be modeled as states in an MDP, where each state represents one behavior, and where the input is the control command sent to the robot (jog, walk, etc.). The MDP's observations are the probability distributions of the same HMM-automaton framework as before. With this setup, when a robot is commanded to jog, the MDP would bias its own distribution towards the jogging state, and information provided by the HMM-automata estimator would reinforce that distribution if the jogging context is identified, or would weaken it otherwise. By incorporating additional information about the behavior, this strategy is likely to improve the accuracy of identifying the behavioral context.

In practice, the MDP states can rarely be observed directly, so this problem is in reality a partially observable MDP (POMDP). Unfortunately, POMDPs are known to have tractability limitations, so further investigation is required to evaluate the applicability of this approach. See Figure 6.1 for a summary of the strategy and a diagram of the complete system.

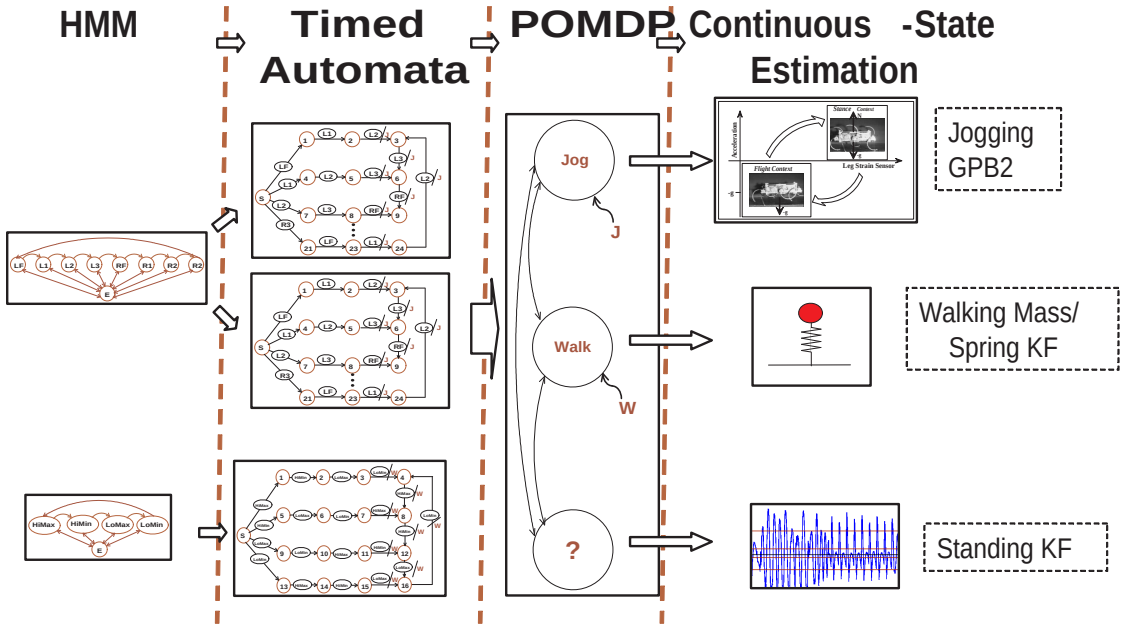


Figure 6.1: POMDP incorporates known control input to refine identification of behavioral controller.

6.5.3 Interpreting HMM distribution as distance metric

As noted earlier, using HMMs that produce continuous distributions $\alpha_k(j)$ provides important advantages to state estimation. The distributions can be interpreted as a measure of the HMM's confidence in each state, so in essence, continuous values of $\alpha_k(j)$ provide a measure of confidence in the contexts themselves. The longer observations match model predictions, the higher the confidence in the probability distribution, and therefore the higher the confidence in the identity of the context.

Such confidence measures are useful when interpreted as a distance metric in the space of contexts. Consider the task of identifying the behavioral context of a robot executing a specific behavior, such as jogging for RHex. When the robot is subjected to transient dynamics, no context is identified. However, as the robot transitions into the the steady state region, the confidence measure of a well-tuned HMM would progressively increase, indicating that the robot is approaching the behavior's context. The confidence would reach its maximal value once the context

is positively identified, the way it is performed in Chapter 4. Thus, confidence becomes a distance metric that measures "how far" a robot's operation is from a specific context.

This distance information can be used to further improve the scalability of multiple-model filters. So far, the context-based framework either activates a single filter within a multiple-model system, or activates the entire set of filters, depending on whether the dynamical context can be identified. This approach can be made more efficient by dynamically varying the number of activated filters. For that, the estimation framework uses the distance metric to estimate the system's proximity to all dynamical contexts, and activates a subset of filters that correspond to the closest contexts. As before, only a single filter is activated if the dynamical context is identified unambiguously, but if no identification is possible, additional filters can be activated progressively as the system moves farther from the context. By avoiding the automatic activation of the entire filter set, this approach further reduces the computational cost of the context-based framework.

The distance information could also be interpreted as a quality measure of the behavior, as deviations from the context suggest that the behavior is not being executed as anticipated. The continuous property of such a quality metric can be used to close the loop on the behavioral controller, where the controller adjusts its parameters in ways that reduce distance to a target context, and therefore increases the gait's quality.

The quality metric could also be used for filter adaptation. For systems where the behavioral context corresponds to a single Kalman filter, as in the case of RHex's walking gait, the parameters of the filter could be regulated continuously (see the Appendix for an implementation of a similar strategy for one-dimensional problems). The farther the robot from the behavior's context, the less accurate the motion model and the greater the process noise covariance. In contrast to the adaptation strategy of the falling ball simulation, the covariance regulation here is a function of sensor information and not of filter parameters. This is an open-loop

strategy which should be able to accommodate higher-dimensional systems than the closed loop approach in the Appendix. It could also be used in conjunction with the closed-loop technique to improve adaptation accuracy.

6.5.4 Incorporating in-state dwelling time into distance metric

The distance metric can be further improved by incorporating the in-state dwelling time to the metric. When the robot operates in steady-state mode, the dwelling time captured by the automata for each state is close to a nominal value. Disturbances change the dwelling time, so detecting that change is evidence of the disturbance.

In order to incorporate this information, the representation of the dwelling time should be compatible with the HMM probability distribution. Therefore, dwelling time information can be expressed in probabilistic terms, a method commonly used in speech recognition [37]. This is done by representing time as a positive distribution that takes the highest value (one) at the nominal dwelling time. As the robot dwells less or more time than nominal, the probability decreases until a cut-off point where it becomes zero.

6.5.5 Expanding HMM structure

So far, the approach to context-based estimation has advocated the design of one HMM per behavioral context. The context of multiple-behavior systems would be identified using multiple *disconnected* HMMs, which provides for small-scale HMMs that are simple to design and inexpensive to run.

A departure from this strategy would be to represent multiple behaviors with a single, fully *connected* HMM, and the added complexity is likely to be offset by improvements in accuracy. The idea is that a fully connected HMM would produce more accurate probability distributions because it would allow the probabilities to shift among states that belong to different behaviors. One way to analyze the

difference between connected and disconnected HMMs is to note that it is mathematically possible for disconnected HMMs to have simultaneously high confidence in each of the behavioral processes they represent. This is clearly a wrong result, since systems are assumed to execute one behavior at a time. On the other hand, the probabilities of a connected HMM have to sum up to one, so at most one of the processes can be accurate. It is worth mentioning that speech recognition research also advocates the construction of a connected network of HMMs, possibly organized in a hierarchical structure.

Practically, the construction of such a connected HMM would start with the development of individual discrete models as before, one model per behavior. The next step is to connect the different HMMs together, with arrows connecting each state to all the other states, whether they belong to the same or to different behavior models. The error states of the individual models are replaced with a single error state for the new connected model. The larger connected model may be too complex to tune by hand, but the tuning can be performed with the same learning techniques mentioned earlier, such as the Baum-Welch method. This way, if the robot is not executing any of the modeled behaviors, the HMM would assign the highest probability to the error state and no context would be identified. Conversely, when the robot executes a modeled behavior, the connected HMM is likely to identify the behavioral context more accurately than before.

In addition to the likely gains in accuracy, this setup provides the advantage of a more expressive distance metric. As before, the probability distribution $\alpha_k(j)$ is interpreted as a distance metric, but now the metric provides *relative* distance information from one context to another. When the robot is transitioning among behavioral contexts, or when it momentarily exits a context, the distribution over the states belonging to different behaviors would reflect the robot's proximity to each behavioral context. As a result, the out-of-context behavior can be quantified as a weighted sum of several modeled behaviors. This would be a new abstracted representation of a robot's complex motions, and it is anticipated that such developments would enable new strategies for reactive control, an example of which is

provided in Section 6.6.

Naturally, the added HMM complexity increases the computational cost of discrete estimation and reduces the overall scalability of context-based estimation. This presents the designer with a cost/performance tradeoff, as HMM complexity can be increased as long as the runtime computational overhead is offset by performance gains. Such design decisions factor available computational resources and accuracy requirements of the task at hand, so they are made on a case-by-case basis.

More generally, it is worth noting that the incremental computational cost of an HMM is lower than that of a multiple-model filter; the cost of running an HMM is that of multiplying matrices together, which increases with the square of the number of discrete states (Equation 3.2); whereas the cost of running an MM system is that of inverting matrices, which increases with the cube of the number of continuous states (Algorithm 1) and with the square of the number of modes (Algorithm 2). Therefore, for a given computational budget, a designer is more likely to improve estimation accuracy by increasing the complexity of discrete-state estimators rather than that of MM systems.

6.5.6 Integrating Kalman filters and HMMs

Looking further in the future, one can envision a new type of multiple-model filtering system, where the individual continuous filters are structurally linked to discrete estimators. A possible architecture would start with HMMs similarly to the examples in this thesis, but the discrete states' observations would include the continuous filters' likelihood in addition to sensor information. Thus, the HMMs would compute the probability distribution based not only on sensor data but also on the performance of the continuous filters. This probability distribution would then be used to replace the current GPB2 functionality. The output of each filter would be weighted by the probability of the state representing its dynamical context, and the weighted outputs would be consolidated as with the GPB2. This way, the weight estimates include both KF likelihoods and sensor information, which

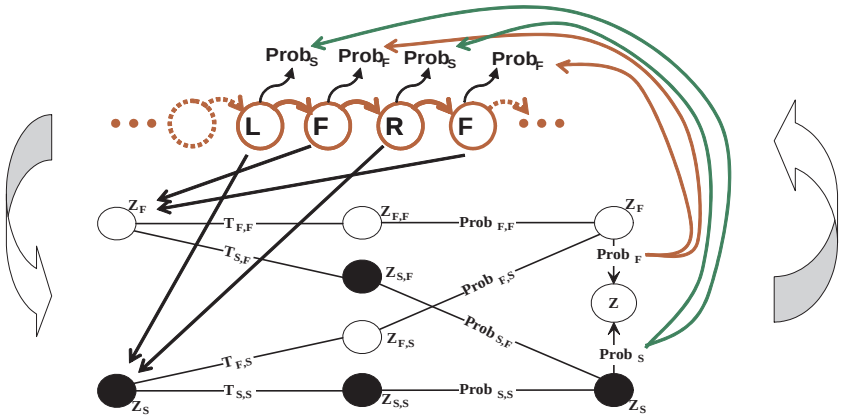


Figure 6.2: Linking discrete-state HMMs and continuous-state MM systems could improve the efficiency of information utilization and further improve estimation accuracy.

should improve the weight's accuracy and therefore the accuracy of the consolidated estimates.

By using filter likelihoods to first compute the discrete probability distribution, then using that probability to scale individual filter outputs, the system closes an estimation loop, as shown in Figure 6.2. This is a different loop than the one developed in Section 5.5 (and also in the Appendix), because the proposed system does not adapt the parameters of individual filters, only mixes their output. It could be worth investigating the possibility of merging the two approaches, whereby multiple-model estimation includes the adaptation of individual filters, but the implications of such an approach on accuracy are conceptually difficult to anticipate.

6.6 Applications

Context-based state estimation applies to a variety of estimation and control tasks, from enabling accurate estimation when conventional methods fail to providing timely information that makes reactive control possible. This diversity of applica-

tions complicates the development of metrics to quantify the framework's accuracy gains, as they vary from one system to another. Throughout the thesis, accuracy measurements were provided where possible for the different systems considered.

More generally, the performance of context based estimation has to meet the task's requirements, and the framework's implementation would be designed accordingly. For example, reactive control requires timely information, and therefore justifies the deployment of classification tools to rapidly identify the dynamical context. Slower vehicles impose less stringent time requirements, so it could suffice to design more conservative but also more accurate discrete estimators such as the HMM-automaton technique to identify the behavioral context.

For the RHex platform, the bandwidth of the estimation is about an order of magnitude higher than the robot's mechanical frequency, so the results of context-based estimation can be used to implement reactive control. An example of such control could involve the regulation of the gait's parameters as a function of state [84].

Perhaps a more interesting application of the framework in the case of RHex is the ability to measure the 'quality' of the behavior being executed, as outlined in Section 6.5. As the robot traverses broken terrain, it may have to change gaits depending on variations in terrain difficulty. On flat, unencumbered terrain, the robot could jog or run while maintain a stable gait. However, the terrain difficulty can increase and lead to gait instability, in which case the robot should slow down and adopt a more conservative gait such as walking. These gait changes could be performed automatically by measuring the quality of the current gait and making control decisions accordingly. When a gait's quality degrades, the robot switches to a more stable gait, thus closing the loop on a behavioral controller.

The ability to measure gait quality can also help in the generation of gaits. Research conducted on Sprawl [34], another hexapod robot, seeks to tune the leg motion parameters to increase gait stability. To date, stability has been difficult to quantify and is currently evaluated through human judgment. Interpreting HMM

probability distributions as a quality metric helps remedy this problem, as the designer could specify a target behavioral context, and tune gait parameters according to a distance-to-context cost function.

To conclude, the estimation approach developed in this thesis improves the accuracy of current state estimators and could enable novel control applications. This work is a first step in a promising investigation of estimation strategies for systems with complex dynamics. Frameworks that combine discrete and continuous estimation are likely to become of greater necessity as robotic systems gain in performance and complexity. Exploring the interaction between the discrete and continuous parts has the potential of leading to new and compelling results for the estimation and control of future robotic systems.

Appendix

Context-Based Filter Adaptation

This appendix presents a preliminary development of a filter adaptation strategy that builds on the context-based estimation approach. The strategy uses context information to regulate the parameters of a single filter in a way that improves its estimation accuracy.

The strategy is implemented in an example that consists of the falling ball simulation, where the task is to regulate the parameters of the flight filter to maintain a level of estimation accuracy even when the ball is no longer in flight.

The approach expands the definition of contexts beyond the description of Chapter 3 to include filter parameters. So far, dynamical contexts have been identified from the classification of sensor data; now, they can also be identified from the classification of filter parameters. For example, a dynamical context can be represented by a range of values of the filter's innovation. When a system's dynamics are accurately represented by the filter's motion model, the innovation is expected to have low values. This allows the identification of the corresponding context whenever innovation values are below a specific threshold.

This is a continuous regulation strategy, although it is restricted to one-dimensional systems. Presenting it here serves two purposes; the first is to demonstrate in simulation that the strategy improves the accuracy of individual filters; and the second is to provide insight about the estimation mechanisms of Kalman filters.

Adaptation of process noise covariance

The approach is presented in the context of the bouncing ball problem, where the task is again to estimate the ball's height, but this time using only a single filter. The challenge is to modify the filter's parameters as the the motion model gains and loses accuracy to generate accurate height estimates.

The technique for continuous parameter adaptation is presented in a step-by-step description of the approach to estimate the ball's height while in free fall. The ball is dropped from high altitude, and its fall is slowed down by air friction, so its equation of motion (plant model) is

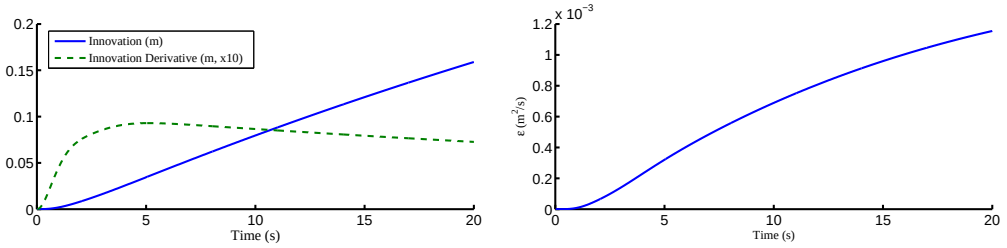
$$z(t) = \frac{\left(e^{-\frac{gt}{V_T}} - 1\right) V_T^2}{g} + V_T t$$

where V_T is the ball's terminal velocity, g is gravity and t is time. Assume that the only available model describes constant-acceleration, friction-free dynamics, $\ddot{z} = -g$. Then the corresponding filter's process and observation equations (estimator model) would be

$$\begin{aligned}
\begin{bmatrix} z \\ v \end{bmatrix}_{k+1} &= \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ v_g \end{bmatrix}_k + \begin{bmatrix} \frac{1}{2}\tau^2 \\ \tau \end{bmatrix} g + \nu \qquad \text{(motion model)} \\
y_{k+1} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} z \\ v \end{bmatrix}_{k+1} + \omega \qquad \text{(observation model)}
\end{aligned}$$

where τ is the discrete sampling period, x and v are position and velocity of the ball, g is gravity, and ν and ω are process and measurement noise of variances Q and R respectively.

At drop-off, the free fall motion model is accurate, as the ball has not yet built up speed and air friction is negligible. Soon thereafter, the velocity increases, and the growing air friction reduces the ball's acceleration and makes the constant-



(a) Innovation r and its derivative $\dot{r}$ ($\dot{r}$ is magnified ten times for display)

(b) Derivative of the squared innovation $\varepsilon = \frac{d}{dt}r^2 = 2r\dot{r}$

Figure 6.3: Innovation characteristics of an increasingly inaccurate motion model

acceleration model progressively less accurate. As a result, the model's predictions also lose accuracy, causing the estimation filter's innovation to grow, as shown in the solid-line plot of Figure 6.3(a).

If the covariance Q of the process noise remains unchanged, then the filter's confidence in the model would not decrease with loss of accuracy. Thus, the weights that the nominal gain K assigns to model predictions would become increasingly incorrect leading to a decrease in the estimates' accuracy. Conversely, if the process covariance increases as the model loses accuracy, then K would progressively de-emphasize predictions. Increasing the value of the process covariance indicates that the model contains more noise, which is the Kalman framework's expression of lower accuracy.

The challenge is to devise a technique for increasing Q proportionally to the loss of accuracy, where proportionality depends on the choice of accuracy metric. That choice is made from analysis of the relation between innovation and model accuracy. Figure 6.3(a) shows that the value of the innovation increases monotonically with the decrease of accuracy. Interestingly, a plot of the accuracy's derivative (in dashed lines) shows a high slope at first, but the slope tapers off. This behavior is due to the fact that the rate of loss of accuracy is large at first when the friction buildup is rapid, but the rate is lower as the ball approaches its terminal velocity. This suggests that a potential candidate for an accuracy metric is the product of the innovation and its derivative, indicating whether the model is losing accuracy and

by how much.

Formally, the metric is $\varepsilon = \frac{d}{dt}r^2 = 2r\dot{r}$, plotted in Figure 6.3(b). This choice reflects the assumption that a previously stable innovation that grows indicates a loss of accuracy in the motion model (sensor faults are not modeled). It is also based on the assumption that relatively large innovations correspond to motion models with poor predictive value. The dynamical context corresponding to the free fall model would consist of values of ε falling below a threshold ε_t. The threshold is manually chosen here for simplicity, but in more realistic situations it would correspond to the maximal value of ε observed when the model is accurate.

This way, the constant-velocity context is identified when $\varepsilon < \varepsilon_t$. To handle cases when $\varepsilon > \varepsilon_t$, a relationship between ε and the model accuracy represented by its covariance Q can be derived as follows. Consider the operation of the covariances in a Kalman filter over two consecutive sampling periods k and $k+1$:

$$
\begin{aligned}
P_k^p &= FP_{k-1}^u F^T + Q_k && (6.1)\\
S &= HP_k^p H^T + R && (6.2)\\
K &= H^T P_k^p S^{-1} &&\\
r_k &= y_k - Hz_k^p &&\\
z_k^u &= z_k^p + Kr_k &&\\
P_k^u &= P_k^p - KSK^T &&\\
z_{k+1}^p &= Fz_k^u + Bu &&\\
r_{k+1} &= y_{k+1} - Hz_{k+1}^p && (6.3)
\end{aligned}
$$

where the superscripts p and u stand for predicted and updated, respectively.

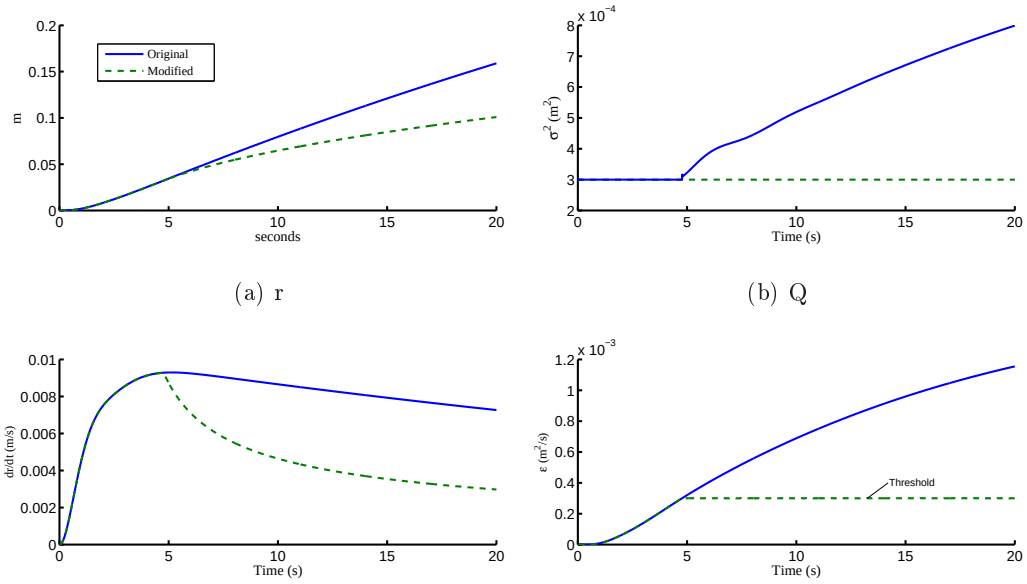
Starting with x_k^p and P_{k-1}^u, it is possible to propagate Q_k from (6.1) through the sequence of equations leading to (6.3). It follows that r_{k+1} and thus $\varepsilon = 2r\dot{r} = 2r_{k+1}\frac{r_{k+1}-r_k}{T}$ (where T stands for sample period) can be rewritten in terms of x_k^p,

P^u_{k-1}, y_k, y_{k+1}, R and Q_k.

Since the innovation in this problem is one dimensional, Q_k can be expressed in terms of ε, x^p_k, P^u_{k-1}, y_k, y_{k+1}. This establishes the relation f between the model's accuracy represented by Q_k and the context predicate ε: $Q_k = f(\varepsilon, x^p_k, P^u_{k-1}, y_k, y_{k+1})$. The expression f is too long to be reproduced here, but, generating it is a straight forward process of term substitution starting from (6.3) and ending in (6.1). The caveat is that this algebraic relation requires the prediction of future measurement y_{k+1}. For simplicity, this simulation projects the actual motion model one step into the future. In more realistic settings, it would be possible to build a statistical model of the sensors that estimates y_{k+1} by extrapolating y_k. It is expected, though not proved, that the inaccuracy of this one-step projection will have little effect on the regulation of ε, since any error will be corrected at the next sample iteration.

This method is applied to the free-fall model, where the one-dimensional Q is regulated so that the value of ε does not exceed the threshold ε_t. As the model's accuracy deteriorates, ε increases beyond ε_t. Q is then increased to reduce ε back to ε_t. Figure 6.4 plots the different parameters of the modified filter in dashed lines. Subfigure (d) shows that ε is maintained at the threshold by modifying Q (these modifications are shown in subfigure (b)) according the algebraic relationship f. As a result, the filter reduces the weight of model predictions, which leads to a state update that matches the measurements more closely. This decreases the rate of the innovation in subfigure (c) and reduces the growth of the innovation in subfigure (a). Thus, the degradation of the motion model's accuracy is captured by the filter and reflected in the state update.

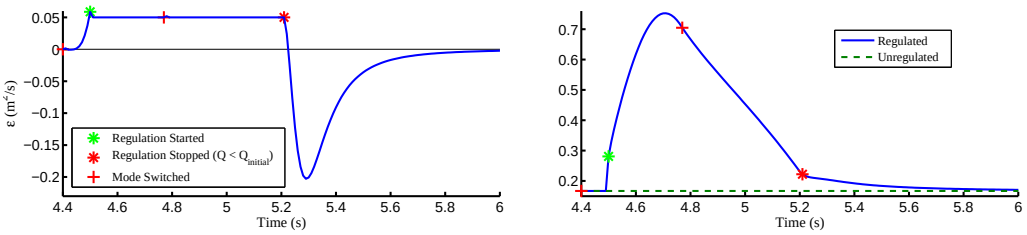
The values of filter parameters throughout a regulated stance are shown in Figure 6.5. At touch-down (marked by the first '+' at the beginning of the plot), the ball starts its stance phase and evolves according to dynamics that are significantly different from flight. The accuracy of the flight model starts to decrease and ε increases. When ε reaches ε_t, the regulation starts (marked by the first '*')



(a) r (b) Q

(c) dr/dt (d) $\varepsilon = 2r\dot{r}$

Figure 6.4: Free fall model. The parameters corresponding to the original and modified filters are plotted in solid and dashed lines, respectively. The innovation of the original and the modified filters are shown in (a), with the corresponding derivatives in (c). The original innovation increases steadily whereas the modified innovation increases at a lower rate. This behavior is obtained by regulating the variance of the process noise (b) in such a way that the value of ε in (d) is maintained equal to the threshold.



(a) ε vs. time

(b) Gain vs. time

Figure 6.5: Nominal Operation

and the covariance of the process noise is increased to maintain $\varepsilon = \varepsilon_t$. When the ball bounces back and resumes free flight (second '+'), the free fall model becomes accurate again and the regulation stops (second '*'). This causes ε to become negative, indicating that the innovation is decreasing, as can be expected when a model's accuracy improves.

At this point, it is worthwhile to briefly compare the proposed adaptation technique to related work. Maybeck [53, ch.10] uses a maximum likelihood approach to adaptively modify the parameters of a filter, and proposes a technique to estimate the covariance of the process noise Q. The technique models the estimated parameter as having a constant value over the last N samples, and assumes that the parameter varies slowly compared to the estimated state. Maybeck derives likelihood functions that measure the consistency of each sample with the constant-value model, and computes the current estimate by finding the parameter value that maximizes the likelihood function. Unfortunately, this technique is significantly more complex that the approach developed here, and Maybeck's efforts at simplifying it to enable online adaptation trade accuracy for tractability.

The approach developed in this thesis is similar to Bar-Shalom and Fortmann's work on "White Noise Model with Adjustable Level" [4]. They measure filter performance by comparing the innovation weighted by its covariance $\varepsilon = r^T S^{-1} r$ to a user defined threshold ε_t. If $\varepsilon > \varepsilon_t$, then Q is scaled up until ε is reduced to ε_t. When applied to the problem of the falling ball, this approach proved impractical

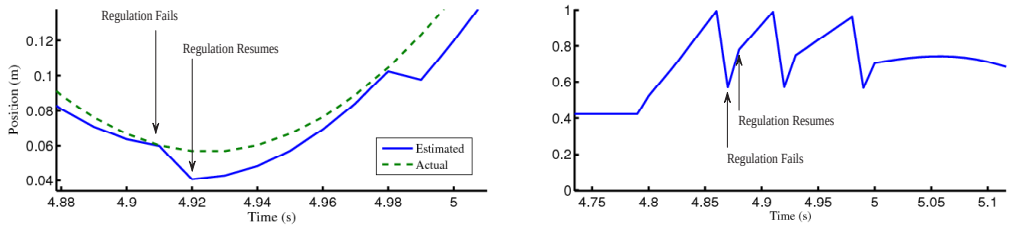
because the sensitivity of S to changes in Q was very low. Indeed, reducing ε by a small amount to maintain its value equal to ε_t requires large changes in Q, and a large Q severely modifies the operation of the filter, producing in particular very low innovations that prevent smooth filter adjustments.

Analysis of the adaptation technique

This section examines the details of the adaptation strategy by lowering the value of ε_t until the covariance regulation fails. Analysis of failures provides an opportunity to gain insight about the mechanisms used by Kalman filters to update the state.

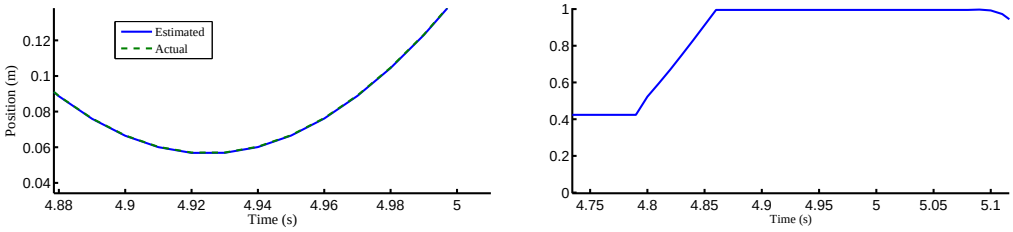
If the simulation is run with a threshold an order of magnitude smaller than in the previous simulation, then filter estimates converge at first but later exhibit the oscillatory behavior of Figure 6.6(a). This behavior is explained as follows. Given that the stance dynamics are different from those of flight, as the free flight model loses accuracy, the filter increasingly shifts emphasis to the observations by pushing the gain K closer to one. However, the free fall model becomes so inaccurate during stance that although $K \to 1$, ε can no longer be maintained at the threshold ε_t. This leads to a failure of regulation and the algorithm briefly restores Q to its original value $Q_{initial}$, as depicted by the lower peaks in subfigure (b). Lower values of Q cause the rapid divergence of the state. The regulation promptly resumes thereafter, again pushing the gain up toward one, and the up-and-down pattern repeats.

Thus, as K approaches unity, the model's accuracy becomes insufficient to maintain ε at the threshold ε_t for any admissible (i.e. non-negative) value of the process covariance. Therefore, the solution is to recognize that the strategy of capping ε no longer applies, so the regulation should be paused. This would set K to the highest value generating an admissible process noise covariance and ε would be allowed to increase beyond ε_t. The regulation is resumed when the model's accuracy increases enough to allow lower gains to lead again to admissible covariance values. The result solves the problem, as shown in Figure 6.7.



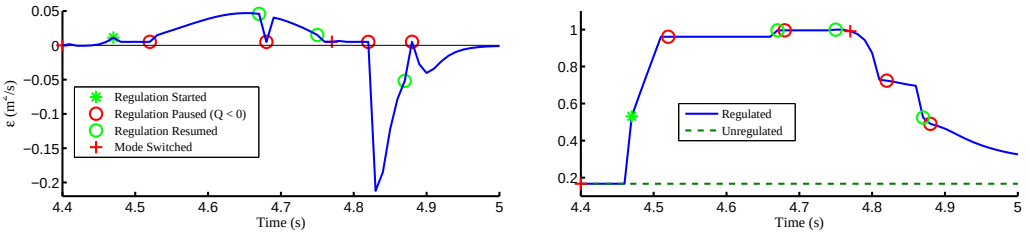
(a) Position vs. time (b) Gain vs. time

Figure 6.6: The adaptation strategy attempts to assign values to Q that are higher than the maximum possible value. This leads to the temporary failure of regulation, causing the Kalman gain oscillates between its initial value ($K_{initial}$) and the value obtained just before regulation fails.



(a) Position vs. time (b) Gain vs. time

Figure 6.7: The Kalman gain K is saturated when the adaptation strategy assigned the highest value possible for Q.



(a) ε vs. time

(b) Gain vs. time

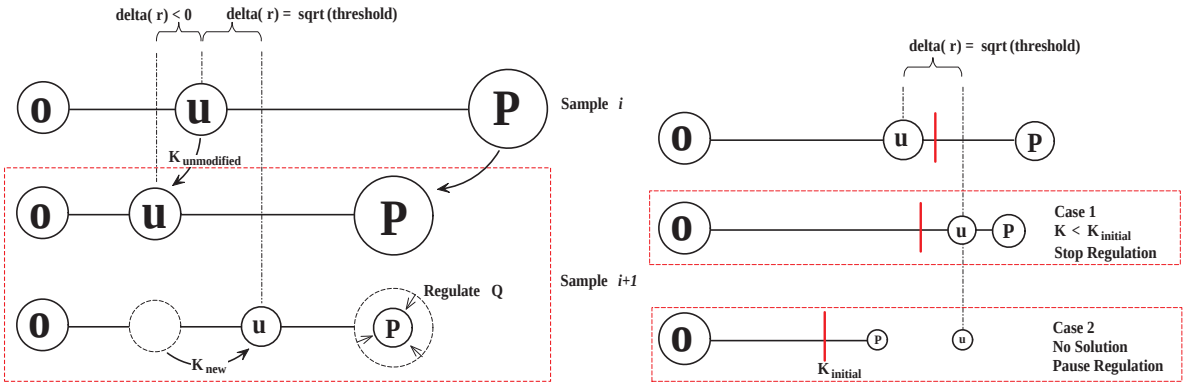
Figure 6.8: Adaptation parameters using a low value of ε_{max}.

The regulated parameters are shown in Figure 6.8. When the the value of K approaches one, the filter pauses the regulation (first 'o' in Figure 6.8). The regulation is resumed when the model's accuracy increases enough to enable admissible values of the covariance to hold $\varepsilon = \varepsilon_t$. This happens briefly at the second 'o' (subfigure (a)) where the model gains accuracy for an instant and the filter gain is increased even closer to unity (subfigure (b)). Almost instantaneously, the model loses accuracy and regulation is paused again (third 'o'). Shortly before liftoff (second '+'), the accuracy is high enough to resume regulation (fourth 'o').

The behavior of ε after liftoff is counter-intuitive. Figure 6.8(a) shows that ε becomes negative as expected after second 4.8, but part way through its evolution, ε is regulated back to ε_t for a brief moment (sixth 'o'). This is counter-productive, as the regulation should not increase the value of ε when $\varepsilon < \varepsilon_t$.

The source of this problem can be identified by analyzing the regulation mechanism in detail. When the model's accuracy increases but still leads to values of ε greater than ε_t, the regulation decreases the process noise covariance in order to hold $\varepsilon = \varepsilon_t$. To see this, Figure 6.9(a) illustrates the operation of a filter with full state access[2] over two iterations of the algorithm. P represents the predicted state, o the observation, and u the state update. The innovation r is represented as the distance between P and o. At iteration i, an inaccurate prediction P leads to a high gain K that pushes the update close to the observation. At iteration $i+1$, an

[2]Full state access means that the state can be directly measured. Translated in Kalman filter terms (Algorithm 1), the observation matrix is unity ($H = I$).



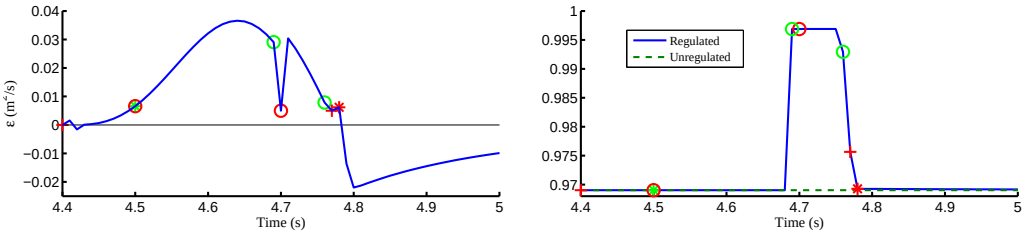
(a) Regulation

(b) Stopping Regulation

Figure 6.9: Regulation

improving prediction pushes P closer to o. If Q were not regulated and left to its previous value, then the same high gain would push the update even closer to the observation. This however, causes r to *decrease* from iteration i to $i+1$, leading to $\varepsilon_{i+1} < \varepsilon_i = \varepsilon_t$. If instead Q was regulated to hold $\varepsilon_{i+1} = \varepsilon_i = \varepsilon_t$, the value of Q would decrease to account for improved prediction accuracy. This reduces K and pushes u closer to P. The new value of Q is chosen so that the change in innovation holds $\varepsilon = \varepsilon_t$.

Unfortunately, it is possible for the model's accuracy to improve too quickly and lead to regulation failure. Consider the two scenarios that lead to terminating the regulation (see Figure 6.9(b)). Case 1 illustrates a successful termination scenario, where Q is reduced to a value below $Q_{initial}$, K is smaller than $K_{initial}$ (represented by a vertical dashed line), the regulation is stopped and the filter parameters are restored to their design values. Case 2 illustrates a failed termination, where the model's accuracy increases rapidly from one sample to the next and P gets so close to o that, in order to hold $\triangle r > 0$ and $\varepsilon = \varepsilon_t$, u would have to be pushed beyond P. This is clearly impossible, and would yield inadmissible values of $K < 0$ and of $Q < 0$. This scenario causes the regulation to pause rather than stop, and to resume when admissible gains can again hold $\varepsilon = \varepsilon_t$. This is what happens at the sixth 'o' of Figure 6.8.



(a) ε vs. time

(b) Gain vs. time

Figure 6.10: Parameters of a successful adaptation strategy

The solution is to adopt a strategy whereby regulation is stopped rather than paused when ε generates inadmissible Q and K even though $\varepsilon < \varepsilon_t$. This solves the problem, as shown in Figure 6.10.

Bibliography

[1] Panos J. Antsaklis and Anil Nerode. Guest editorial hybrid control systems: An introductory discussion to the special issue. *IEEE Transactions on Automatic Control*, 4(43):457–460, April 1998.

[2] Kellar Autumn, Martin Buehler, Mark Cutkosky, Ronald Fearing, Robert J. Full, Daniel Goldman, Richard Groff, William Provancher, Alfred A. Rizzi, Uluc Saranli, Aaron Saunders, and Daniel E. Koditschek. Robotics in scansorial environments. In Charles M. Shoemaker Grant R. Gerhar and and Douglas W. Gage, editors, *Proceedings of SPIE*, volume 5804, pages 291–302, May 2005.

[3] M. Bai, D. H. Zhou, and Helmut Schwarz. Adaptive augmented state feedback control for an experimental planar two-link flexible manipulator. *IEEE Transactions on Robotics and Automation*, 14(6), December 1998.

[4] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Mathematics in science and engineering. Academic Press, 1988.

[5] R. Bodson and J. E. Groszkiewicz. Multivariable adaptive algorithms for reconfigurable flight control. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 4, pages 3330–3335, December 1994.

[6] S. Bourassa and S. Kamoun. A parallel filtering technique for a surveillance radar. In *Canadian Conference on Electrical and Computer Engineering*, volume 1, pages 55–58, September 1993.

[7] Michael S. Brannicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, April 1998.

[8] D. Brown, T. Authenrieth, O. R. Tutty, and E. Rogers. Towards the application of multiple model adaptive control schemes. In *IEEE Colloquium on Adaptive Controllers in Practice*, number 1997/176, April 1997.

[9] M. E. Campbell and S. Brunke. Nonlinear estimation of aircraft models for on-line control customization. In *IEEE Proceedings Aerospace Conference*, volume 2, pages 2/621 – 2/628, March 2001.

[10] L. Campo, P. Mookerjee, and Y. Bar-Shalom. State estimation for systems with sojourn-time-dependent markov model switching. *IEEE Transactions on Automatic Control*, 36:238–243, February 1991.

[11] C. B. Chang and M. Athans. State estimation for discrete systems with switching parameters. *IEEE Transactions on Aerospace and Electronic Systems*, 14(3):418–425, 1978.

[12] Gregory S. Chirikjian and Joel W. Burdick. The kinematics of hyper-redundant robot locomotion. *IEEE Transactions on Robotics and Automation*, 11(6):781–793, December 1995.

[13] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005. ISBN 0-262-03327-5.

[14] Alongkrit Chutinan and Bruce H. Krogh. Verification of infinite-state dynamic systems using approximate quotient transitions. *IEEE Transactions on Automatic Control*, 46(9):1401–1409, September 2001.

[15] A. Costa, G. Kantor, and H. Choset. Bearing-only landmark initialization with

unknown data association. In *IEEE International Conference on Robotics and Automation*, pages 1764 – 1770, New Orleans, LA, April 2004.

[16] J. A. Crossman, Guo Hong, Y. L. Murphey, and J. Cardillo. Automotive signal fault diagnostics. *IEEE Transactions on Vehicular Technology*, 52(4):1063–1075, July 2003.

[17] Pierre A. Devijver and Josef Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, 1982.

[18] M. Bernardine Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Carnegie Mellon University, January 2004.

[19] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2000.

[20] H. Durrant-Whyte, E. Bell, and P. Avery. The design of a radar-based navigation system for large outdoor vehicles. In *International Conference on Robotics and Automation*, pages 764–769. IEEE, 1995.

[21] P. J. Escamilla-Ambrosio and N. Mort. A hybrid kalman filter - fuzzy logic architecture for multisensor data fusion. In *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*, pages 364–369, Septembre 2001.

[22] Christopher Urmson et. al. A robust approach to high-speed navigation for unrehearsed desert terrain. Technical Report 8, Carnegie Mellon University, August 2006.

[23] K. A. Fisher and P. S. Maybeck. Multiple model adaptive estimation with filter spawning. In *Proceedings of the American Control Conference*, pages 2326–2331, Chicago, Illinois, June 2000.

[24] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Elsevier, 2nd edition, 1990.

[25] S. Hirose and A. Morishima. Design and control of a mobile robot with an articulated body. *The International Journal of Robotics Research*, 9(2):99–114, 1990.

[26] Masato Ishikawa and Mitsuji Sampei. State estimation of non-holonomic mobile robots using nonlinear observers. In *IEEE Conference on Robotics and Automation*, pages 1379–1384, 1995.

[27] A. H. Jazwinski. *Stochastic Processes and Fitlering Theory*. Academic Press, 1970.

[28] T. A. Johansen, J. Kalkkuhl, J. Ludemann, and I. Petersen. Hybrid control strategies in abs. In *Proceedings of the 2001 American Control Conference*, volume 2, pages 1704–1705, Arlington, VA, June 2001.

[29] B. H. Juang and L. R. Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, August 1991.

[30] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Proceedings of the Conference on signal processing, sensor fusion, and target recognition VI*, pages 182–193, Orlando, FL, April 1997.

[31] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1(82):35–45, 1960.

[32] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 1:95–108, 1960.

[33] G. Kantor and S. Singh. Preliminary results in range-only localization and mapping. In *IEEE International Conference on Robotics and Automation*, pages 1818 – 1823, Washington, D.C., May 2002.

[34] Sangbae Kim, Jonathan E. Clark, and Mark R. Cutkosky. isprawl: Design and tuning for high-speed autonomous open-loop running. *International Journal of Robotic Research*, 25(9):903–912, 2006.

[35] Z. Kohavi. *Switching and Finite Automata Theory, 2nd ed.* McGraw-Hill, 1978.

[36] Bruce H. Krogh. Hybrid systems: State of the art and perspective. In *18th Brazilian Congress of Automatic Control*, Florianopolis, SC, Brazil, September 2000. I only have a paper copy.

[37] Kai-Fu Lee. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System.* PhD thesis, Carnegie Mellon University, April 1988. Tech Report CMU-CS-88-148.

[38] Michael D. Lemmon, Kevin X. He, and Ivan Markovsky. Supervisory hybrid systems. *IEEE Control Systems Magazine*, 19(4):42–55, 1999.

[39] S. Lenser and M. Veloso. Automatic detection and response to environmental change. In *2003 International Conference on Robotics and Automation*, Taipei, September 2003.

[40] Scott Lenser. *On-line Robot Adaptation to Environmental Change.* PhD thesis, Carnegie Mellon University, August 2005.

[41] John J. Leonard and Hugh F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376 – 382, 1991. Mobile Robot Navigation;Geometric Beacon Tracking;Model-Based Localization;Geometric Parameterization;Beacon Information Extraction;.

[42] Uri Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems.* PhD thesis, Stanford University, October 2002.

[43] P. Lin. *Proprioceptive Sensing for a Legged Robot.* PhD thesis, University of Michigan - Ann Arbor, 2005.

[44] P. Lin, H. Komsuoglu, and D. E. Koditschek. A leg configuration measurement system for full body pose estimates in a hexapod robot. *IEEE Transaction on Robotics*, 21(3):411–422, June 2005.

[45] Brad Lisien, Deryck Morales, David Silver, George A Kantor, Ioannis Rekleitis, and Howie Choset. Hierarchical simultaneous localization and mapping. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, volume 1, pages 448 – 453, October 2003.

[46] Lennart Ljung. Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems. *IEEE Transactions on Automatic Control*, 24(1):36–50, 1979.

[47] Lennart Ljung. *System identification: theory for the user*. Prentice-Hall, 1987.

[48] J. Lobo and J. Dias. Integration of inertial information with vision. In *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, pages 1263–1267, Aachen, Germany, August 1998.

[49] Hui-Ling Lou. Implementing the viterbi algorithm. *IEEE - Signal Processing Magazine*, pages 42–52, September 1995.

[50] D. Luenberger. Observing the state of a linear system. *IEEE Transactions on Military ELectronics*, 1:74–80, 1964.

[51] D. Luenberger. Observers for multivariable systems. *IEEE Transactions on Automatic Control*, 11:190–197, 1966.

[52] P. S. Maybeck. *Stochastic Models, Estimation and Control*, volume 1 of *Mathematics in science and engineering*. Academic Press, 1979.

[53] P. S. Maybeck. *Stochastic Models, Estimation and Control*, volume 2 of *Mathematics in science and engineering*. Academic Press, 1982.

[54] P. S. Maybeck and P. D. Hanlon. Performance enhancement of a multiple model adaptive estimator. *IEEE Transactions on Aerospace ans Electronic Systems*, 31(4):1240–1254, October 1995.

[55] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple

model methods in target tracking: a survey. *IEEE Transactions on Aerospace and Electronic Systems*, 34:103–123, January 1998.

[56] Robert L. Moose and P. P. Wang. An adaptive estimator with learning for a plant containing semi-markov switching parameters. *IEEE transactions on systems, man, and cybernetics*, 3:277–281, 1973.

[57] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Prentice Hall, 1989.

[58] Daniel Nikovski. *State-Aggregation Algorithms for Learning Probabilistic Models for Robot Control*. PhD thesis, Carnegie Mellon University, 2002.

[59] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[60] R. Ravikanth and S. P. Meyn. Performance evaluation in identification and adaptive control of time varying systems. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 1, pages 205–210, December 1994.

[61] H. Rehbinder and B. K. Ghosh. Multi-rate fusion of visual and inertial data. In *Proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 97–102, Baden-Baden, 2001.

[62] Alfred A. Rizzi and Daniel E. Koditschek. An active visual estimator for dexterous manipulation. *IEE transactions on robotics and automation*, 12(5):697–713, 1996.

[63] S. I. Roumeliotis and G. A. Bekey. An extended kalman filter for frequent local and infrequent global sensor data fusion. In *Proceedings of the SPIE Conference on Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, pages 11–22, Pittsburgh, PA, October 1997.

[64] U. Saranli, M. Buehler, and D. E. Koditschek. Rhex: A simple and highly mobile robot. *International Journal of Robotics Research*, 20(7):616–631, July 2001.

[65] Uluç Saranli. *Dynamic Locomotion with a Hexapod Robot*. PhD thesis, University of Michigan, September 2002.

[66] S. Scheding, E. Nebot, and H. Durrant-Whyte. High-integrity navigation: A frequency-domain approach. *IEE Transactions on Control Systems Technology*, 8(4):2217–2222, July 2000.

[67] J. Schmitt and P. Holmes. Mechanical models for insect locomotion: Dynamics and stability in the horizontal plane - I. Theory. *Biological Cybernetics*, 83:501–515, April 2000.

[68] William John Schwind. *Spring Loaded Inverted Pendulum Running: A Plant Model*. PhD thesis, The University of Michigan, 1998.

[69] Elie Shammas, Alon Wolf, and Howie Choset. Three degrees-of-freedom joint for spatial hyper-redundant robots. *Journal of Mechanism and Machine Theory*, 41(2):170 – 190, February 2006.

[70] H. Singer and M. Ostendorf. Maximum likelihood successive state splitting. In *Proc. IEEE Conf. Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, 1996. Thanks Sajid Siddiqi.

[71] S. Skaff, G. Kantor, D. Maiwand, and A. A. Rizzi. Inertial navigation and visual line following for a dynamical hexapod robot. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, October 2003.

[72] R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56 – 68, 1986. spatial uncertainty estimation;pattern recognition;coordinate

frames;locations;spatial relationships;positioning errors;measurement errors;tolerances;robot;orientation;.

[73] Trey Smith. Science autonomy in the atacama. In *Proceedings of ICML 2003 Workshop on Machine Learning Technologies for Autonomous Space Applications*, 2003.

[74] Torsten Soderstrom and Petre Stoica. *System Identification*. Prentice-Hall, 1988.

[75] Jeremy Sproston. Decidable model checking of probabilistic hybrid automata. In M. Joseph, editor, *Lecture Notes in Computer Science*, volume 1926/2000, page 31, Pune, India, September 2000. Springer-Verlag GmbH.

[76] A. Stolcke and S. Omohundro. Best-first model merging for hidden markov model induction. Technical report tr-94-003, Intl. Computer Science Institute, 1994. Thanks Sajid Siddiqi.

[77] D. Strelow and S. Singh. Optimal motion estimation from visual and inertial measurements. In *IEEE Workshop on Applications of Computer Vision*, pages 314–319, December 2002.

[78] K. J. ÅStröm and B. Wittenmark. *Adaptive Control*. Addison Wesley, 1989.

[79] S. Thrun, J. Langford, and V. Verma. Risk sensitive particle filters. In *Proceedings of Neural Information Processing Systems (NIPS)*, December 2001.

[80] M. Tibaldi and E. Zattoni. Robust control of active suspensions for high performance vehicles. In *Proceedings of the IEEE International Symposium on Industrial Electronics*, volume 1, pages 242–247, June 1996.

[81] Paul Tompkins. *Mission-Directed Path Planning for Planetary Rover Exploration*. PhD thesis, Carnegie Mellon University, May 2005.

[82] A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-based vision

system for place and object recognition. In *The 9th International Conference on Computer Vision*, Nice, France, 2003.

[83] Vandi Verma, Joquin Fernandez, and Reid Simmons. Probabilistic models for monitoring and fault diagnosis. In Raja Chatila, editor, *The Second IARP and IEEE/RAS Joint Workshop on Technical Challenges for Dependable Robots in Human Environments.* Toulouse, France, October 2002.

[84] Joel D. Weingarten, Gabriel A. D. Lopes, Martin Buehler, Richard E. Groff, and Daniel E. Koditschek. Automated gait adaptation for legged robots. In *IEEE International Conferance on Robotics and Automation*, 2004.

[85] C. Wellington and A. Stentz. Learning predictions of the load-bearing surface for autonomous rough-terrain navigation in vegetations. In *Proceedings of the International Conference on Field and Service Robotics*, pages 49–54, July 2003.

[86] N. Wiener. *The Extrapolation, Interpolation and Smoothing of Stationary Time Series.* John Wiley & Sons, Inc., New York, N.Y., 1949.

[87] Hakan Lorens Samir Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events.* PhD thesis, Carnegie Mellon University, January 2005.

[88] Y. Zhang and J. Jiang. Integrated active fault-tolerant control using IMM approach. *IEEE Transactions on Aerospace ans Electronic Systems*, 37(4):1221–1235, October 2001.