Motion Interference Detection in Mobile Robots

Juan Pablo Mendoza
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213–3890
jpmendoza@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213–3890
mmv@cs.cmu.edu

Reid Simmons
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213–3890
reids@cs.cmu.edu

***Abstract*—As mobile robots become better equipped to autonomously navigate in human-populated environments, they need to become able to recognize internal and external factors that may interfere with successful motion execution. Even when these robots are equipped with appropriate obstacle avoidance algorithms, collisions and other forms of motion interference might be inevitable: there may be obstacles in the environment that are invisible to the robot's sensors, or there may be people who could interfere with the robot's motion. We present a Hidden Markov Model-based model for detecting such events in mobile robots that do not include special sensors for specific motion interference. We identify the robot observable sensory data and model the states of the robot. Our algorithm is motivated and implemented on an omnidirectional mobile service robot equipped with a depth-camera. Our experiments show that our algorithm can detect over 90% of motion interference events while avoiding false positive detections.**

I. Introduction

Autonomous mobile robots have reached the point where they can start to perform useful tasks in unconstrained human-populated environments. For robots to become an efficient means for performing such tasks, they need to be able to navigate safely and effectively without supervision. Therefore, robustness during navigation (and, more generally, during task execution) is an area of increasing importance in robotics. To perform tasks robustly in unconstrained and uncertain environments, robots need to reason about their own state and execution, which is why *execution monitoring* –the problem of recognizing and indicating anomalies in behavior– has gained importance in the robotics community [1]. This paper presents a *Hidden Markov Model* (HMM)-based model for detection of *Motion Interference* (*MI*) events–events that disrupt the normal motion of the robot–with the purpose of increasing the robustness, and thus the autonomy, of robots in human-populated environments.

Previous work on execution monitoring for mobile robots has mostly focused on *model-based* monitors, such as the hierarchical monitor of Xavier [2], the knowledge-based SKEMon monitor [3], and the Unit Circle qualitative representation-based monitor [4], in all of which properties such as the dynamics of the mobile robot are explicitly modeled. More recently, there has also been robotics work in *model-free* monitors [5], in which fault detection arises solely from observing the robot's behavior, rather than from a predictive model. The model presented in this paper more closely resembles some estimation-based methods (e.g., using HMMs or Kalman Filters) that have been used for execution monitoring of outdoor robots [6], [7] and planetary rovers [8], [9]. In contrast to these methods, however, where the dynamics or the parameters of the robot are explicitly given, our model is built as an observer outside of the robot's existing architecture: the monitor builds a model from navigation data, such as driving commands and measurable velocity, as opposed to using pre-existing knowledge of the robot's properties. In this respect, our model takes some ideas from activity recognition work [10], [11], where the inner workings of the system to be described are unknown, yet behaviors are successfully classified. In some ways, then, our approach combines strengths from both model-based and model-free monitors: while an explicit model of the robot's behavior is built, this model is learned from observed data, which potentially allows for modeling of complex behaviors whose properties might be too difficult to define analytically.

Fig. 1: The CoBot mobile service robots. CoBot robots autonomously perform tasks in human-populated environments, often without any supervision. This makes robustness during navigation a particularly relevant problem.

II. The Motion Interference detection problem

The robotic platform used to test the algorithm presented in this paper is the CoBot service robot (Figure 1). CoBots are


(a)


(b)


(c)

Fig. 2: Types of Motion Interference considered in this paper: (a) Collision against a partially detectable obstacle, (b) Being held by a person, and (c) Having one or more wheels stuck

equipped with an omnidirectional wheeled base for motion, laser range-finders and/or Microsoft Kinects for obstacle avoidance and localization in the environment and cameras. The level of autonomy of the CoBots is very high, having navigated more than 8.7 km while autonomously completing service tasks requested by inhabitants of the Gates-Hillman Center at Carnegie Mellon University [12]. However, the CoBots' ability to reason about their own state and performance during execution is still limited. The goal of this paper is to provide a model to increase the robustness of CoBots' (and, more generally, mobile robots') autonomous navigation by detecting when some environmental or internal event is interfering with its regular motion. For the remainder of this paper, we focus on the following types of motion interference, as they are the most relevant to the CoBot's execution, though we believe our algorithm is more generally applicable to motion interference.

Collision with partially detectable objects (*collision*)
: While the fairly reliable perception mechanisms of the CoBots, combined with obstacle avoidance algorithms, can provide successful local navigation the vast majority of the time, there are some obstacles that are either undetectable or only partially detectable to the sensors. Transparent obstacles are particularly challenging for such light-based sensors, but opaque obstacles can also be only partially detectable. For example, the top of the table shown in Figure 2a is too tall for either the Kinect or the laser range-finder to perceive it. While the robot avoids the leg of the table successfully, it cannot detect and avoid the much larger full width of the table, leading to potential collisions.

Being held by a person (*hold*)
: In some situations, a person may want to stop the CoBot's motion, and although the CoBots have emergency stop buttons, a person's first reaction may reasonably be to directly stop it by holding it, as shown in Figure 2b. Furthermore, the effect of being suddenly grabbed and stopped seems similar to the effect of a head-on collision between the robot and a transparent or otherwise undetectable wall. In either case, it is important for the robot to be able to detect the situation and react accordingly.

Having one or more wheels stuck (*stuck*)
: Several events in the environment might cause one or more of the robot's wheels to get stuck. For example, a person might accidentally place their foot in front of the robot's wheels during navigation (see Figure 2c), or the robot could have trouble passing over gaps or level changes, such as the entrance into an elevator. A similar type of motion interference might occur independently of the environment due to malfunctioning wheel motors.

Note that all of the described forms of interference will have a similar effect on the robot: the robot's motion in the direction of travel will be impeded, and thus its velocity is going to be diminished to a value smaller than the given velocity command. Because of this similarity, all of these events are grouped for this paper under the category of Motion Interference, and are treated as equivalent events. Experimental results in Section IV support this abstraction. However, in the case of non-equivalent events (e.g., a person pushing the robot forward, or a defective motor given only a fraction of the expected current), the model can readily support detection of different types of events by adding the appropriate states to the model of Section III.

III. A HMM for Motion Interference detection

Hidden Markov Models are particularly well-suited for modeling an *MI* detector: even though the CoBots don't have sensors to directly detect *MI* events, the occurrence of these events can be inferred from the observations that are available to the robot. HMMs provide an appropriate framework to perform these inferences.

A Hidden Markov Model M can be defined as a 5-tuple $M = \{S, \Pi, A, O, B\}$, where

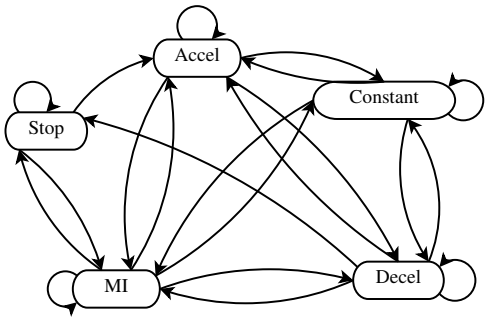

Fig. 3: Diagram of the HMM modeling the Motion Interference Monitor. Ellipses represent hidden states, while arrows represent transitions between states. (While no arrows are shown in transitions which the data determined had probability 0, no transitions were explicitly forbidden)

$S = \{s_i\}$	the set of hidden states in M
$\Pi = \{\pi_i\}$	the initial distribution of M such that $P(S_1 = s_i) = \pi_i$
$A = \{a_{ij}\}$	the transition probabilities such that $P(S_{t+1} = s_j \mid S_t = s_i) = a_{ij}$
$O = \{\vec{o}_i\}$	the space of possible observations
$B = \{b_i\}$	the emission probabilities such that $P(\vec{o}_j \mid S_t = s_i) = b_i(\vec{o_j})$

For the detector described in this paper, each possible simple behavior of the robot is represented by a state in S. The possible behaviors assigned to the CoBot for purposes of *MI* detection are $S = \{stop, accel, constant, decel, mi\}$ representing the states where the robot is stopped, accelerating, at constant positive speed, decelerating, and having its motion interfered, respectively. A diagram for the resulting HMM is shown in Figure 3. Since the robot always starts from the $stop$ state, the respective values for Π are $\Pi = \{1, 0, 0, 0, 0\}$.

True transition probabilities between pairs of states were approximated from a large amount of hand-labeled gathered training data. In general, given a set of observations accompanied by state labels S_t for all times $0 \leq t \leq T$,

$$a_{ij} = \frac{\sum_{t=0}^{T-1} \left(\delta_{S_t,s_i} \cdot \delta_{S_{t+1},s_j}\right)}{\sum_{t=0}^{T-1} \delta_{S_t,s_i}}, \tag{1}$$

where $\delta_{i,j}$ is the Kronecker delta function. That is, the transition probability from state s_i to s_j is given by the total number of labeled observations in which the robot transitioned from state s_i to s_j divided by the total number of labeled observations in which the robot transitioned from s_i to anywhere. The only transition probabilities not calculated using Equation 1 were transitions into *MI* $a_{i,mi} \forall i \neq mi$. Given the rarity of *MI* events in normal CoBot runs, *MI* events had to be artificially created to gather significant data for experimenting, and thus the real probabilities to transition from a different state to the *MI* state are extremely difficult to gather. These values were thus set to $a_{i,mi} = p_{mi}$, where p_{mi} is the only parameter for the *MI* detector. Tuning p_{mi} has the effect of varying the total number of *MI* events detected, and therefore it serves as a parameter to find the desired trade-off value between precision and recall of *MI* events (see Section IV for more details).

Observation space O may vary greatly depending on the sensors of the specific robot and the type of event that needs to be detected. For the task of detecting *MI* events in the CoBot, observations were of the form

$$\vec{o}_t = (u_t - v_t, a_t, j_t, u_t), \tag{2}$$

where u_t is the commanded forward velocity of the robot, and v_t, a_t and j_t are the forward velocity, acceleration and jerk of the robot as measured by its wheel encoders. The reasoning for each observation and the method for obtaining it are the following:

Velocity difference $\mathbf{u_t - v_t}$.
One can expect that when the robot's movement is being interfered, its measured velocity would be significantly lower than its commanded velocity. While u_t is directly obtained as a command, v_t needs to be calculated from the individual encoder velocities. Velocity v_t was obtained from the least squares solution transforming the encoder values for each of the four wheels to forward, sideways and rotational components of the robot's velocity.

Acceleration $\mathbf{a_t}$.
There are times during normal (i.e., not *MI*) robot motion when the velocity difference $u_t - v_t$ is significantly positive. For example, when the robot accelerates from a stopped position to full speed, the change in u_t is discrete, while v_t smoothly changes from 0 to u_t over time. Thus, $u_t - v_t$ alone is not a sufficient observation to detect *MI* events. Therefore, acceleration a_t is added as an additional layer to distinguish between these events: while an accelerating robot has a positive acceleration, a robot during a *MI* event usually has either negative (at the moment the *MI* event begins) or near-zero (once velocity has stabilized) acceleration. a_t can be obtained by applying linear regression to the last N_a measures of velocity v_t as a function of t, and then getting the slope of the resulting line. Even though a_t could be obtained from only the last 2 values of v_t and t, a larger number N_a is used to reduce the effects of noise in the data. N_a must be large enough to negate the effects of noise, but small enough to provide a meaningfully recent value for a_t. For this paper, $N_a = 4$.

Jerk $\mathbf{j_t}$.
In the event that the robot's motion is disrupted while the robot is accelerating, it might be the case that the positive acceleration continues for a while until the final velocity under the disturbance is reached. In these cases, a_t might be within the normal parameters of an accelerating motion at any instant, but the change in acceleration in time may provide valuable information to distinguish *MI* acceleration from normal acceleration. For this

reason, the second time derivative of the velocity is also used as part of the observations. Jerk j_t is obtained by dividing the difference between the two last acceleration measurements by the difference in time between measurements. However, since jerk is significantly more sensitive to noise in the data than acceleration, a larger number of measurements $N_j \geq N_a$ is used to calculate the accelerations to be used for j_t, for purposes of further smoothing of noise at the cost of a more outdated jerk measurement. For this paper, $N_j = 8$. Various values of N_a and N_j were tested on small portions of the data, and the final values were those that maximized performance in these portions.

Velocity command $\mathbf{u_t}$.
The final attribute of the observation is simply the commanded velocity u_t. The purpose of this attribute is to distinguish between *stop* , where $u_t = 0$, and *constant*, where $u_t \neq 0$. Otherwise these states would have identical properties.

There are some design decisions behind the use of the attributes of Equation 2 as observations. First, notice that only the forward velocity, acceleration and jerk of the robot are used. While the CoBot's base is omnidirectional, most of its movements are restricted to the forward direction (plus rotations) because its sensors are pointing forward. Therefore, using only the forward direction has the benefit of requiring estimation of fewer parameters at little cost. Furthermore while the CoBot has other sensors (e.g., Kinect) that could provide estimates of velocity apart from the encoders, for this paper only encoder estimates were used. While encoders provide the simplest method for velocity estimation, the biggest concern with using them as the only estimator is that, on extremely slippery surfaces, the robot's wheels could keep spinning at the commanded velocity even during an *MI* event. We determined, however, that even in the most slippery surfaces in the CoBots' environment (i.e., hardwood floors), slipping was not enough to make *MI* events undetectable using only encoder-based velocities. If this were not the case, however, additional sources of velocity information could be added as observations at the cost of additional parameter estimation.

Finally, emission probabilities B are calculated from hand-labeled training data in a similar manner to transition probabilities A. For the specific monitor described in this paper, the data is treated as being generated by conditionally independent Gaussian variables. The Gaussian assumption fits the data relatively well, but the independence assumption does not hold for the data in question: velocity, acceleration and jerk are strongly correlated. While the general model readily supports treating data as conditionally dependent, there is a trade-off between higher model accuracy at the cost of more parameter estimation and processing (conditionally dependent) and lower model accuracy with a simpler and more efficient model (conditionally independent). For the purposes

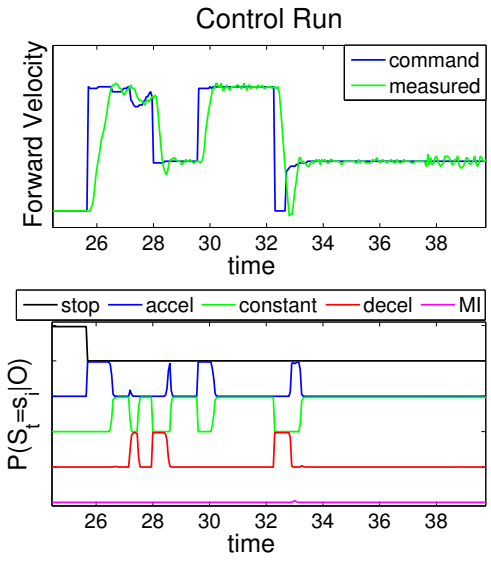

Fig. 4: Sample of data gathered from a normal (control) run of the CoBot navigating in its environment. The top figure shows the velocity command and the measured velocity (meters per second) over time (seconds)), while the bottom stacked probability figure shows the respective assigned probabilities $P(S_t = s_i|O)$ (each between 0 and 1) for each state given the sequence of observations.

of this paper, the simpler model provided satisfactory results (see Section IV), and therefore the it was preferred over the more complex one. Rewriting the observation attributes as $(o_1 \equiv u_t - v_t, o_2 \equiv a_t, o_3 \equiv j_t, o_4 \equiv u_t)$ for brevity, the emission probabilities can then be written as:

$$\begin{aligned} b_i(o_1, o_2, o_3, o_4) &= P(o_1, o_2, o_3, o_4 | S_t = s_i) \\ &= \prod_{j=1}^{4} P(o_j | S_t = s_i) \\ &= \left(\prod_{j=1}^{3} f(o_j; \mu_{i,j}, \sigma^2_{i,j}) \right) P_i(o_4), \end{aligned} \quad (3)$$

where the individual probabilities are defined as:

$$f(o_j; \mu_{i,j}, \sigma^2_{i,j}) = \frac{1}{\sigma_{i,j}\sqrt{2\pi}} e^{-\frac{(o_j - \mu_{i,j})^2}{2\sigma^2_{i,j}}} \quad (4)$$

$$P_i(o_4) = \begin{cases} \delta_{0,o_4} & \text{if } i = 1 \\ 1 & \text{if } i = 2, 4, 5 \\ 1 - \delta_{0,o_4} & \text{if } i = 3 \end{cases} \quad (5)$$

Equation 4 simply describes a Gaussian probability distribution whose parameters were obtained from training labeled data, while Equation 5 describes the probability of observing a certain velocity command $u_t = o_4$ depending on the current state: in state *stop*, $o_4 = 0$ always; in state *constant*, $o_4 \neq 0$, and in any other state o_4 could be anything.

Having defined all S, Π, A, O, B, the HMM-based *MI* detector is fully defined. Now, given any sequence of observa-

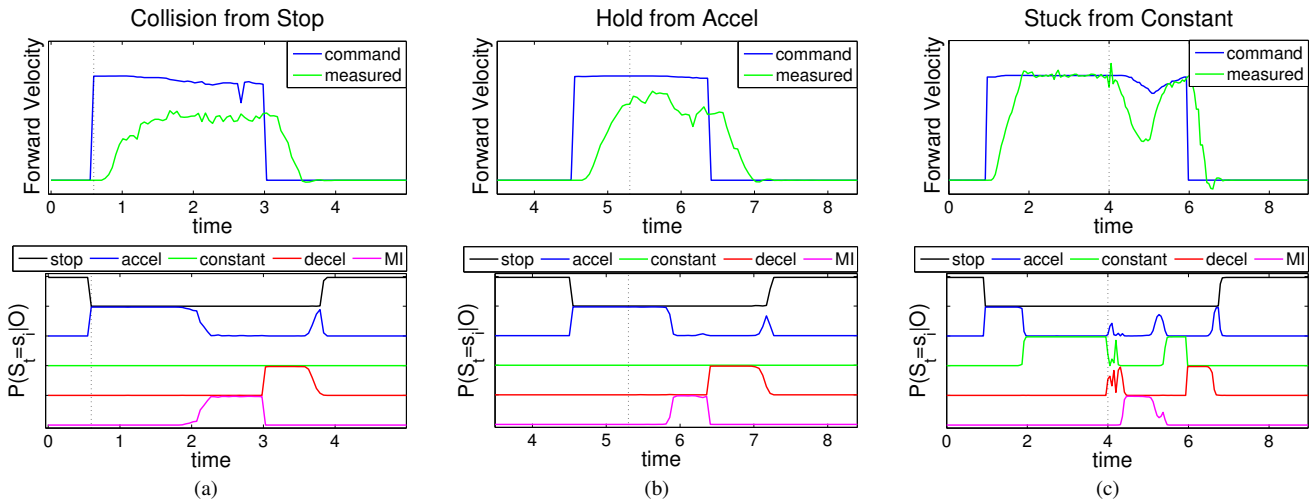

Fig. 5: Examples of data gathered from *MI* runs. As in Figure 4, top figures show velocities while bottom figures show probabilities for each state, as predicted by the monitor. Figures show (a) a collision against a partially detectable obstacle right as the robot starts to move, (b) somebody holding the robot as it is accelerating, and (c) something interfering with a wheel's rotation when the robot is traveling at constant speed. Vertical dotted lines indicate the beginning of an *MI* event, while rise in $P(S_t = s_{mi}|O)$ above 0.5 indicates detection of the event.

tions, the probability of being in state *MI* at each time can be calculated using an algorithm such as the forward algorithm described in [13]. Then, if $P(S_t = s_{mi}|O) > thresh$ (for this paper, $thresh = 0.5$), an *MI* event has been detected at time t. Changing the value of $thresh$ varies the sensitivity of the detector, but for this paper it was kept fixed, as parameter p_{mi} already served this purpose.

IV. Detector Performance Results

A. Methods

To gather the necessary data for training and testing of the detector, two long control runs (no *MI* events) and 29 short test runs (with *MI* events) were conducted on the CoBot robot. For each run, the robot was instructed to autonomously move from its current location to a different location in the building; the driving commands, encoder-based velocity and times of *MI* events (perceived by a human observer) were then recorded. The control runs, during which a human supervisor made sure no *MI* events happened, lasted about 3 minutes each, and gave a total of 7145 observations. A subset of the data gathered during a control run is shown in Figure 4. The test runs were each much shorter, focusing on the *MI* event, and giving a total of 8101 observations. Of the test runs, 15 contained $collision$ events, 6 contained $hold$ events, and 8 contained $stuck$ events. Figure 5 shows the data gathered from three of these experiments.

To train and test the detector, each observation was manually labeled as $stop$, $accel$, $constant$, $decel$ or *MI*. Since *MI* times were previously recorded, each observation during those periods was labeled as *MI*. When the robot travels at constant speed, the standard deviation of the encoder-measured velocity is about $\sigma = 0.028$m/s. From this, each observation where the velocity command u_t was 0 and the measured velocity v_t was within σ of 0 was labeled as $stop$. Similarly, each observation where $|v_t - u_t| \leq \sigma, u_t \neq 0$ was labeled as $constant$. To label $accel$ (or $decel$), every observation after an increase (or decrease) of u_t, and while $|v_t - u_t| > \sigma$ was labeled as $accel$ (or $decel$, respectively). Other observations (i.e., noisy observations where $|v_t - u_t| > \sigma$ but u_t had been constant) were labeled as $constant$.

The detector's performance was then tested using leave-one-out cross-validation: for a given parameter set, 31 tests were conducted (one for each labeled run of the robot). For test i, all runs except for run i were used for training the detector (i.e., finding transition and emission probabilities), which was then tested on run i. A *true positive* detection happened when at least one frame within a *MI*-labeled interval was classified as *MI*. A *false positive* detection was defined as each group of consecutive frames outside of the *MI*-labeled intervals that were classified as *MI*, given that such group was not a continuation of a true positive detection (the probability of being in state *MI* could take a few frames to decay after an *MI* event; this was not considered a false positive). A *false negative* was defined as each *MI*-labeled time interval where no *MI* event was detected.

B. Results

The goal of the detector presented in Section III is to detect *MI* events reliably and within a useful time from initial interference. The proposed measures for judging the model are therefore precision (the fraction of detected *MI* events that were true *MI* events) and recall (the fraction of true *MI* events detected) rates of detection, as well as the average and median time to detection from when interference starts.

The only parameter of the model, transition probability p_{mi}, was varied to find the trade-off between precision and

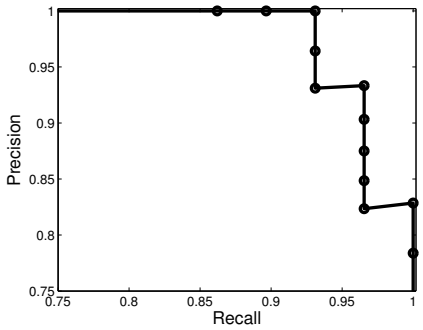

Fig. 6: Trade-off between precision and recall rates for motion interference detection, as parameter p_{mi} is varied.

recall rates. For each tested parameter value, a *full test* –i.e., a set of 31 cross-validation tests as described in Section IV-A– was conducted, with the results of Figure 6. For the purposes of the CoBot project, optimization for high precision was prioritized, given that the project focuses on giving a high degree of autonomy to the robot, and stopping execution for false positive detections would hinder this autonomy.

An optimal parameter value for our purposes was $p_{mi} = 5 \times 10^{-8}$, since it had the highest recall rate for which 0 false positives were detected. For this value, the precision rate was 100% while the recall rate was 93.1%; the average time to detection from initial motion interference was $\bar{t} = 0.647$ seconds. The median time to detection was significantly lower than this, at $t_m = 0.36$ seconds, reflecting the fact that a few outlier detections took significantly longer than the average detection time. These outliers were mostly *MI* events that started from the $stop$ state (e.g., Figure 5a); this can be explained by the fact that the wheel's accelerations looked normal in the beginning, even if they were mostly slipping while spinning, before their behavior was abnormal enough to be detected as an *MI* event. This suggests that perhaps adding an estimate of velocity from a different sensor as an observation could help diminish the average time of detection. Overall, however, the detection time was usually well under a second, which is a useful time-frame for many applications, such as stopping when being held from a dangerous situation, or reacting to an inescapable collision.

V. Conclusion

We presented an HMM-based model for Motion Interference (*MI*) detection for a mobile robot. We identified the sensory observables of the robot and three types of motion interference. Through experiments conducted on the CoBot service robot, we have shown that such a model can successfully detect events that are not directly perceivable by the robot. Our work in general contributes an approach in which robots can reason about specific events by looking at their internal and external sensed state with input from their commanded controls. While this work focuses on the detection of *MI* events rather than actions to recover from them, we considered a base stop command to the robot when the event is detected, and will pursue research on other possible actions.

The *MI* events we considered limit the forward velocity of the robot, but other types of *MI* events could be detected using a similar approach (e.g., pushing the robot so that its measured velocity is above its velocity command). It is in principle feasible to detect anomalies in the behavior of the robot even if these anomalies have not been explicitly modeled: the formulation of HMMs allows us not only to calculate the probability of being in a particular state given a series of observations, but also the probability that a model describes the observable of a robot given a particular series of observations [13]; one could thus expect that a robot that has fallen in an unmodeled state would yield a significantly different model probability distribution than a robot running normally (i.e., within the model). In this way, HMM-based models for execution monitoring could provide a natural model for implementation of hybrid model-based and model-free monitoring. Finding whether this is a practical method to detect anomalies in our robots is a topic of future research.

VI. Acknowledgments

This work was partially supported by the National Science Foundation award number NSF IIS-1012733 and the AFOSR grant number FA2386-10-14138. The views and conclusions contained in this document are those of the authors only.

References

[1] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[2] J. Fernandez and R. Simmons, "Robust execution monitoring for navigation plans," in *Proceedings of IEEE Int. Conf. on Intelligent Robots and Systems*, 1998.

[3] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Monitoring the execution of robot plans using semantic knowledge," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 942–954, 2008.

[4] H. Liu and G. M. Coghill, "A model-based approach to robot fault diagnosis," *Knowledge-Based Systems*, vol. 18, no. 4-5, pp. 225–233, 2005.

[5] O. Pettersson, L. Karlsson, and A. Saffiotti, "Model-free execution monitoring in behavior-based robotics," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 37, no. 4, pp. 890–901, 2007.

[6] S. Scheding, E. Nebot, and H. Durrant-Whyte, "The detection of faults in navigation systems: A frequency domain approach," in *Proceedings of the Int. Conf. on Robotics and Automation*, 1998, pp. 2217–2222.

[7] E. M. Nebot, H. F. Durrant-Whyte, and S. Scheding, "Frequency domain modeling of aided GPS for vehicle navigation systems," *Robotics and Autonomous Systems*, vol. 25, no. 1-2, pp. 73–82, 1998.

[8] R. Washington, "On-board real-time state and fault identification for rovers," in *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2000, pp. 1175–1181.

[9] V. Verma, G. Gordon, R. Simmons, and S. Thrun, "Particle filters for rover fault diagnosis," in *IEEE Robotics & Automation Magazine special issue on human centered robotics and dependability*, 2004.

[10] D. Govindaraju and M. Veloso, "Learning and recognizing activities in streams of video," in *Proceedings of the AAAI Workshop on Learning in Computer Vision*, 2005.

[11] K. Han and M. Veloso, "Automated robot behavior recognition," in *Proceedings of IJCAI Workshop on Team Behaviors and Plan Recognition*, 1999.

[12] M. Veloso et al., "Symbiotic-autonomous service robots for user-requested tasks in a multi-floor building," 2012.

[13] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.