Mobile Robot Fault Detection based on Redundant Information Statistics

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

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

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

***Abstract*— Detecting and reacting to faults (i.e., abnormal situations) are essential skills for robots to safely and autonomously perform tasks in human-populated environments. This paper presents a fault detection algorithm that statistically monitors robot motion execution. The algorithm does not model possible motion faults, but it instead uses a model of normal execution to detect anomalies. The model of normal execution is based on comparisons between redundant sources of information; specifically, wheel encoder readings and localization algorithm output are used as the redundant sources of displacement information. The algorithm was implemented on a service robot that often navigates in a human-populated environment without supervision. Experimental results show that the algorithm can detect even minor motion faults and stop execution immediately to guarantee safety to the humans around the robot.**

I. Introduction

As autonomous mobile robots start to populate human environments, safety during execution is becoming a primary concern for researchers and developers. This rising necessity for robustness in unconstrained environments has made *execution monitoring* –the problem of recognizing and indicating anomalies in behavior– increasingly prominent in the robotics community [6]. An important feature of unconstrained, human-populated environments is their richness: in such complex environments, it is not realistic to assume that the robot's designers can foresee every way in which execution could fail. Therefore, execution monitors that explicitly require *faulty execution models* (e.g., motion interference [4] or wheel failures [10]) are helpful in detecting faults quickly, but not sufficient to detect unforeseen faults. This work, on the other hand, does not assume any previous knowledge of the ways in which execution could fail, but instead only requires some *normal execution model*, and uses this model to detect deviations from normal behavior. Once a significant deviation from normal behavior has been detected, the robot can, at the very least, stop immediately until a qualified operator can fix the problem.

For this paper, normal execution models are created using *redundant information*: equivalent information that is provided by multiple sources. For example, robot location can be provided by both Wi-Fi signal and visual landmarks. Information obtained from these multiple sources is approximately equal under normal circumstances, and significant deviations from this are symptoms of failure in execution.

Fig. 1: The CoBot mobile service robots. CoBot robots autonomously perform tasks in human-populated environments, often without any supervision. Autonomous execution monitoring during navigation is thus essential for the safety of people around them.

The robotic platform used to test the algorithm presented in this paper is the CoBot service robot (see Figure 1). The CoBots are exposed to unsupervised physical interaction with humans constantly, having navigated among humans more than 100 km while autonomously completing service tasks requested on demand by inhabitants of the Gates-Hillman Center at Carnegie Mellon University [9]. Since the CoBots often perform these tasks without the supervision of any member of the developing team, safe autonomous interaction with humans is a high research priority. Each CoBot is equipped with an omnidirectional four-wheel base for motion (and encoders for odometry), laser range-finders or Microsoft Kinects for obstacle avoidance and localization in the environment and cameras. Under normal execution, many precautions are taken to ensure safe execution around humans (e.g., if there is no safe path around a human, the CoBots stop and ask the human to move, instead of attempting to steer around him or her). However, these algorithms may fail under abnormal circumstances, such as malfunctioning obstacle-detection sensors, leading to potentially unsafe execution. Prompt detection and recovery from abnormal situations is

thus essential for robustness and safety of the CoBots.

II. Probabilistic failure detection

The monitor described in this paper operates by comparing equivalent observations (at time t) o_{t1} and o_{t2} given from two separate sources. During normal execution, the difference $x_t = o_{t1} - o_{t2}$ between them is expected to be close to 0. However, in real-world applications, with noisy sensors and actuators, this difference at each time-step is usually not exactly 0, and even in the limit as time goes to infinity, the expected difference may be close to but not exactly 0. Because of this, the approach used in this work is probabilistic in nature, incorporating uncertainty naturally into the model.

Formally, the method for detecting faults is the following: Given a set of comparison observations $X = \{x_1, x_2, \ldots, x_T\}$ with sample mean $\bar{X}$, the algorithm estimates the probability that the true mean μ of the underlying model lies within some acceptable interval $[\mu_-, \mu_+]$ around 0. That is, the algorithm calculates $P(\mu_- \leq \mu \leq \mu_+)$. To do this, one can first define a standardized variable Z:

$$Z = \frac{\bar{X} - \mu}{\sqrt{\frac{\sigma^2}{n}}}, \tag{1}$$

where σ^2 is the variance of X and n is the size of X. The standardized problem then becomes that of calculating $P(Z_- \leq Z \leq Z_+)$, where Z_- and Z_+ are calculated analogously to Z from μ_+ and μ_- respectively. Given that these variables are in standard form, the desired probability can be obtained straightforwardly from the standard cumulative normal distribution function $\Phi(z)$:

$$\begin{aligned} P(\mu_- \leq \mu \leq \mu_+) &= P(Z_- \leq Z \leq Z_+) \\ &= P(Z \leq Z_+) - P(Z < Z_-) \\ &= \Phi(Z_+) - \Phi(Z_-) \end{aligned} \tag{2}$$

After calculating this probability, detecting failures in execution is reduced to choosing a threshold probability p_{thresh} below which it is too unlikely that the observed data still fits the expected model.

III. Redundant information in the CoBot

The CoBots have several sensors and algorithms from which redundant information can be obtained to monitor their execution. The algorithm presented here can be used on any element of the following non-exhaustive list of redundant information:

Location and displacement. Several sensors and algorithms can provide information at each timestep about the robot's location or displacement. Currently, the commanded velocity, wheel encoders, laser range-finder and Microsoft Kinect are used to localize the robot using an augmented particle filter [1], [2]. Each one of these, including the output of the localization algorithm itself, can be used as a source of location or displacement information. The CoBots also have infrared detectors for identification of pre-specified landmarks in the environment, which can provide this information as well. Furthermore, streams of images from the RGB cameras could be used to obtain location or displacement information, with algorithms such as visual odometry [5] or visual landmark recognition [8], although these are not currently implemented on the CoBots.

Time to task completion. The knowledge base of the CoBots include an estimate of how long each of its tasks is expected to take [3]. This knowledge can be computed both at the higher levels (e.g., how long it takes to fetch an object from a certain office) as well as the lower levels (e.g., how long it takes to traverse a particular segment in its navigation planning graph). This expected knowledge can be compared to the actual measured time of task completion to look for faults in execution.

Expected object locations. One of the CoBots' algorithms [7] for finding objects in their environment queries the internet to infer the probability that a certain object will be found in a certain space (e.g., the probability that coffee is found in the kitchen). While searching for these items in their environments, the CoBots could compare the inferred probability returned by this algorithm to the actual experienced frequency with which it finds such objects, to find abnormal situations (e.g., if someone is actively hiding all the coffee from the CoBots).

One of the most immediate safety concerns for the CoBots is the need to be confident that the robots are moving as expected while navigating among humans, without running into problems such as wheel motor or encoder malfunctions, collisions against imperceptible obstacles, getting lost, and others. Many of these motion failures can lead to dangerous situations (e.g., robots drifting to unsafe territory or attempting to drive through unperceived objects), and therefore this paper focuses on detection of motion failures. (Once a motion failure has been detected, the simplest safe action is taken: the robot stops immediately.) The monitor in this paper was thus trained to detect failures in displacement data, and the two redundant sources were the wheel encoders and the output of the localization algorithm. These sources of information were chosen because their output can be relatively simply used to obtain displacement at each timestep.

To obtain displacement (in robot x, y, and heading θ coordinates) from the wheel encoders, the displacement of each of the four encoders is mapped to x, y and θ using a least squares solution. To obtain displacement from the localization algorithm, x, y and θ displacement in world coordinates (obtained by comparing locations at consecutive timesteps) are transformed to robot coordinates by applying the appropriate rotation. Figure 2a shows the resulting normal execution displacement data as obtained from these two sources. Notice that the noise in this data is very significant, and therefore a statistical approach to fault detection is necessary.

To fully define a monitor based on these observations, parameters σ^2, μ_-, μ_+ and p_{thresh} must be defined for each of

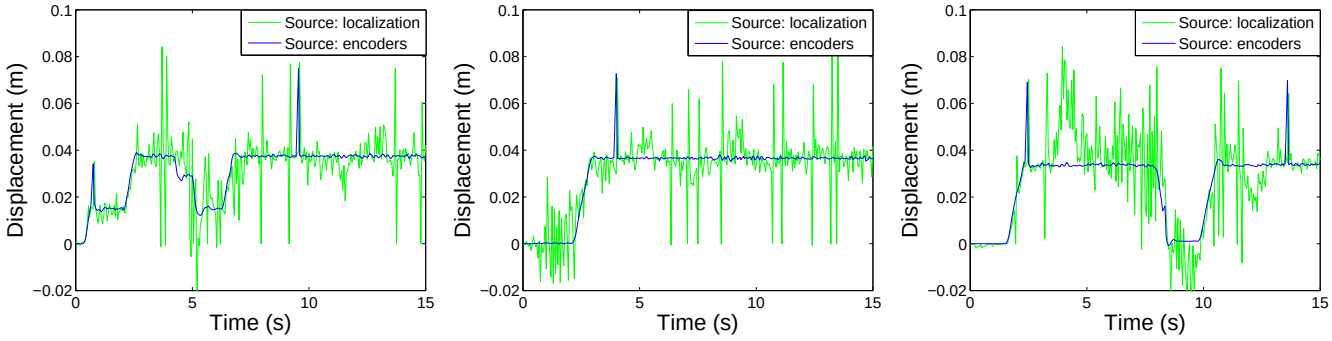

Fig. 2: Encoder and localization-based displacement data. (a) shows the first 15 seconds of data during normal execution, while (b) and (c) show similar periods with $\varepsilon = -0.1$ and $\varepsilon = -0.4$ respectively. Note that there is no obvious difference between $\varepsilon = 0$ and $\varepsilon = -0.1$ from visual inspection, but at $\varepsilon = -0.4$, the abnormal behavior is more clear

the dimensions (x, y, θ). For this paper, these parameters were chosen to be equal for each of the dimensions for simplicity, although this does not need to be the case. Also, parameter values were chosen to be conservative towards autonomy —i.e., they were chosen with the goal of minimizing false positive fault detections. For the experiments in this paper, $\sigma^2 = 0.001$ was chosen as a conservative approximation of the measured $\sigma^2 = (0.00019, 0.00005, 0.00012)$ for (x, y, θ) respectively. Similarly, $\mu_- = -0.001$ and $\mu_+ = 0.001$ were chosen as conservative approximation of the measured long-term error during normal execution (the measured value was $\mu = (-0.0007, -0.0003, -0.0004)$. Finally, $p_{thresh} = 0.01$ was chosen to be low enough so that no false positives were detected during normal execution.

IV. Experiments and Results

To test the execution monitor's performance, a wheel encoder malfunction was artificially induced. During each timestep of the CoBot's navigation, the reported displacement of one of the four encoders (always the same one) deviates from the true encoder displacement reading by a pre-specified fraction ε of the true displacement. For example, $\varepsilon = 0$ represents normal execution, while $\varepsilon = -0.1$ represents a situation where three of the wheel encoders work normally, but the fourth one reports $0.9d$, where d is the true displacement of the fourth wheel. Figures 2b and 2c show the displacement data from wheel encoders as well as localization algorithm output for two values of ε. For the experiments in this paper, all values of ε where less than 0; however, preliminary tests suggest the monitor can detect faults with $\varepsilon > 0$ as well.

Different values of ε were tested to determine how detection times would vary as a function of how subtle the fault to be detected was. For each ε value, 10 tests were run to find the average time to detection. For each test, the robot was instructed, at a high level, to autonomously move to different destinations in the building, avoiding obstacles and interacting with people as usual. During each test, every time a new wheel encoder observation was received, $P(\mu_- \leq \mu \leq \mu_+)$ was calculated as shown in Equation (2), using all the observations received by that time. If $P(\mu_- \leq \mu \leq \mu_+)$ fell below p_{thresh}, a fault was detected. Results are shown in Figure 3. As is shown in the figure, and consistent with intuition, the time required to detect faults increases significantly as the tested fault gets more subtle and thus more observations are necessary to be confident that they were not produced by the expected model. Some faults more minor than $\varepsilon = -0.05$ (e.g., $\varepsilon = -0.01$ was tested) are not detectable by the monitor, since these subtle faults maintain the true deviation μ within the acceptable margin $[\mu_-, \mu_+]$ even as time approaches infinity.

V. Future Work

While the monitor presented in this paper has shown promising detection results for detection of one type of unmodeled failure, two directions for future work seem clear.

A first direction to expand this work is to test the monitor on several different faults. One of the key strengths of the monitor presented here is that it does not explicitly model faults, but instead it uses information redundancy to assess normality. This means that other kinds of motion failures (e.g., collisions, getting lost) should also be detectable by this monitor. Preliminary tests were conducted to test whether this monitor could detect collisions against imperceptible obstacles; on average, collisions were detected 1.98 seconds after the collision started (significantly slower than the 0.65 seconds in [4], but without explicitly modeling the fault). Showing detection of multiple kinds of unmodeled faults is essential to show the flexibility of this monitor.

As a second direction for future work, notice that one of the biggest limitations of the monitor, as presented in this paper, is that all faults are considered to happen globally. At each timestep, all the observations accumulated throughout the current run of the robot are grouped into a single statistic, which is then analyzed to determine its normality. This global assumption works well for faults that do not depend on the robot's state, but other faults (e.g., collisions or getting lost) are very much localized in specific sets of states of the

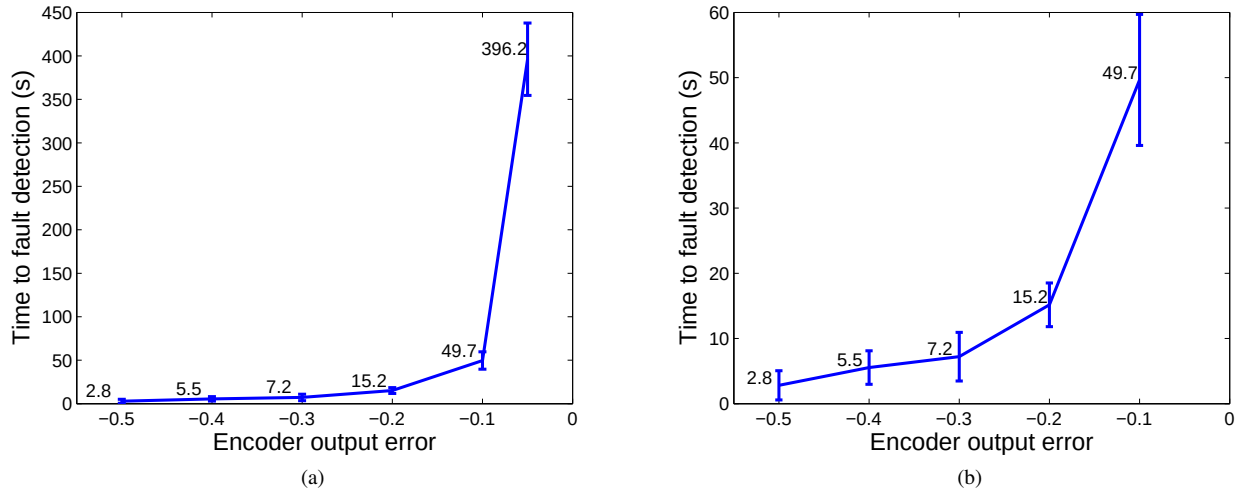

Fig. 3: Time to fault detection as a function of the chosen fractional error ε. (a) shows all the experimental results, while (b) leaves out $\varepsilon = -0.05$ for ease of visualization of the rest of the data. Error bars mark one standard deviation of the data.

robot (e.g., a particular location, velocity or time). This may be part of the reason collision detection times were slower than in previous work. The authors are currently working on an extension of the monitor that will gather observations from particular areas of the robot's state space to find areas in which faults have occurred. Aside from such a model's obvious advantage of being able to detect localized faults more effectively, it will also contribute to the problem of fault isolation: the robot will be able to communicate not only that a fault has happened, but also in what region of the state space of the robot the fault happened (e.g., a fault happened in a certain area of the building, or only when the robot was going at a certain speed). Such a monitor would be significantly more helpful in the areas of fault isolation, and would allow for more useful fault communication to a human operator.

VI. Conclusion

This paper presented a probabilistic algorithm for fault detection during robot motion execution. The algorithm does not need a model of possible faults, but it instead looks for deviations from normal execution to detect failures. The algorithm thus does require a model of normal execution to be provided. For this paper, the model of normal execution was based on comparing equivalent information from redundant sources to find statistically significant differences between them, indicating a fault. Specifically, displacement data derived from wheel encoder output is compared to displacement data obtained from a particle filter-based localization algorithm to find statistically significant discrepancies that indicate a fault in execution.

Experimental results show that the algorithm can consistently detect faults and safely stop when the robot has a malfunctioning wheel encoder, with malfunctions as small as 5% off normal wheel encoder values. Further work will focus on two aspects: testing the monitor for detection of other safety-critical situations, such as collisions and getting lost, and expanding the algorithm to be able to detect and communicate localized faults, and not only global ones. Such further work will demonstrate the general applicability of the algorithm as a fault detection and isolation model.

References

[1] Joydeep Biswas, Brian Coltin, and Manuela Veloso. Corrective gradient renement for mobile robot localization. In *Proceedings of IEEE Int. Conf. on Intelligent Robots and Systems*, pages 73 – 78. IEEE, September 2011.

[2] Joydeep Biswas and Manuela Veloso. Depth camera based indoor mobile robot localization and navigation. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation (to appear)*. IEEE, 2011.

[3] Brian Coltin, Manuela Veloso, and Rodrigo Ventura. Dynamic user task scheduling for mobile robots. In *Proceedings of the the AAAI Workshop on Automated Action Planning for Autonomous Mobile Robots at AAAI 2011*, August 2011.

[4] Juan Pablo Mendoza, Manuela Veloso, and Reid Simmons. Motion interference detection in mobile robots. In *Proceedings of IEEE Int. Conf. on Intelligent Robots and Systems (to appear)*, 2012.

[5] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 1:652–659, 2004.

[6] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88, 2005.

[7] Mehdi Samadi, Thomas Kollar, , and Manuela Veloso. Using the web to interactively learn to find objects. In *Proceedings of the Twenty-Sixth AIII Conference on Artificial Intelligence*, 2012.

[8] Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21:735–758, 2002.

[9] Manuela Veloso et al. Symbiotic-autonomous service robots for user-requested tasks in a multi-floor building. *Under submission*, 2012.

[10] Vandi Verma, Geoff Gordon, Reid Simmons, and Sebastian Thrun. Particle filters for rover fault diagnosis. In *IEEE Robotics & Automation Magazine special issue on human centered robotics and dependability*, 2004.