Efficient Failure Detection for Mobile Robots Using Mixed-Abstraction Particle Filters

Christian Plagemann, Cyrill Stachniss, and Wolfram Burgard

University of Freiburg
Georges-Koehler-Allee
79110 Freiburg, Germany
{plagem,stachnis,burgard}@informatik.uni-freiburg.de

Summary. In this paper, we consider the problem of online failure detection and isolation for mobile robots. The goal is to enable a mobile robot to determine whether the system is running free of faults or to identify the cause for faulty behavior. In general, failures cannot be detected by solely monitoring the process model for the error free mode because if certain model assumptions are violated the observation likelihood might not indicate a defect. Existing approaches therefore use comparably complex system models to cover all possible system behaviors. In this paper, we propose the mixed-abstraction particle filter as an efficient way of dealing with potential failures of mobile robots. It uses a hierarchy of process models to actively validate the model assumptions and distribute the computational resources between the models adaptively. We present an implementation of our algorithm and discuss results obtained from simulated and real-robot experiments.

1 Introduction

Whenever mobile robots act in the real world, they are affected by faults and abnormal conditions. Detecting such situations and allowing the robot to react appropriately is a major precondition for truly autonomous vehicles. While the applied techniques need to be able to reliably detect rare faults, the overall estimation process under error-free conditions should not be substantially more complex compared to systems that are optimized for the normal operational mode. Separate monitoring processes that use more complex models to cover all possible system behaviors introduce an unnecessary high computational load. In this paper, we introduce mixed-abstraction particle filters as an effective means for adaptively distributing the computational resources between different system models based on the estimated validity of their specific model assumptions.

The term ”fault detection” is commonly referred to as the detection of an abnormal condition that may prevent a functional unit from performing a

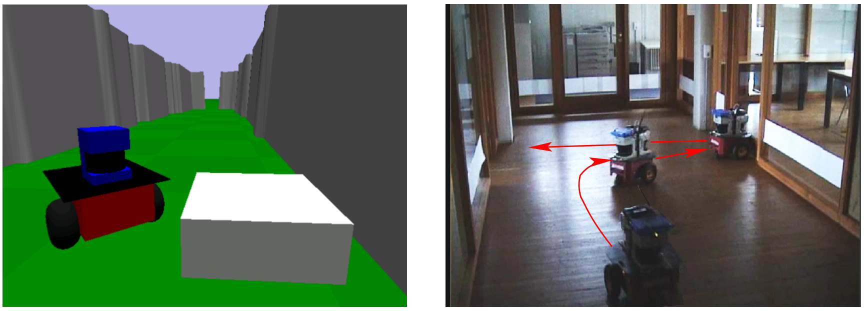

Fig. 1. The left image depicts a simulated robot before colliding with an obstacle which is not detected by its sensors. The right photograph shows a real mobile robot that collides with an undetected glass door while moving on its planned trajectory to the neighboring room.

required function [11]. Most works in the fault detection and isolation literature deal with internal faults such as defects in hardware or software. For the mobile robot domain, we apply the same nomenclature to external influences like collisions or wheel slip since their effects are similar to those of internal defects and the resulting models have the same structure.

As an illustrating scenario, consider a mobile robot equipped with wheel encoders, a laser range finder, and a sufficiently accurate map of the environment. In the fault-free case, the position of the robot can be tracked using a standard tracking algorithm such as a Kalman filter or a particle filter with a simplistic odometry-based motion model, which is formally given in Section 3.3. In odometry-based models, the next system state x_t is directly predicted from the odometry, which is the measurement o_t obtained from the wheel encoders.

Although such models allow us to evaluate different state hypotheses by weighting them using exteroceptive measurements, e.g., using a laser range measurement l_t, they do not directly allow us to detect collisions with obstacles that cannot be perceived by the sensors of the robot. This is due to the fact that when the robot stops moving, its wheel encoders do not record any motion, which is perfectly consistent with the recorded laser measurements. Therefore, no filter degradation occurs and there is no possibility to detect such faults inside the filter. One typical solution to overcome such problems is to compare the estimated trajectory with the planned one on a higher system level. As major drawbacks of such an approach, one cannot infer the actual cause for the deviation from the planned trajectory and the system architecture is complicated by the stronger connection between the planning and tracking module.

An alternative solution is to consider the actual motion commands that have been sent to the motors instead of just using the wheel encoder readings. However, this makes the system model substantially more complex and the predictions, which are now based on motor currents and accelerations, less accurate. In our experiments, we observed that such a model is around 32 times slower to compute than the odometry-based model. It is important to note that the odometry-based model makes the implicit assumption, that the wheel encoder measurements reflect the intended motion. If this assumption is violated, the standard estimation technique cannot be used to estimate the joint probability $p(x, f)$ anymore, where x stands for the pose of the robot and f indicates the failure state.

In this paper, we propose to make the model assumptions explicit and to build a model abstraction hierarchy. We present the mixed-abstraction particle filter algorithm that uses such a hierarchy to direct computational resources to the most efficient model whose assumptions are met. In this way, it minimizes the computational load while maximizing the robustness of the system.

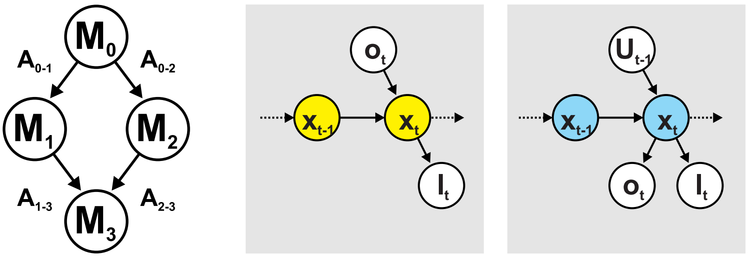

Fig. 2. Arbitrary model hierarchy (left) with an unrestricted model M_0, several restricted models M_1, M_2, M_3 as well as the specific assumptions $A_{i \to j}$ that restrict the state spaces respectively. Two models for the same physical process: the standard odometry-based model (e.g. M_1) that uses the odometry measurements o_t as controls (middle) and the laser measurements l_t as sensor inputs. A less restricted model (e.g. M_0, on the right) that includes the actual motion commands u_t as controls.

The paper is organized as follows. After the discussion of related work, we present our mixed-abstraction particle filter algorithm and its application to monitoring mobile robots in Section 3. Finally, Section 4 presents experimental results.

2 Related Work

Model-based diagnosis has been approached from within the AI community using symbolic reasoning with a focus on large systems with many interacting components and from the control theory community concentrating on fewer components with complex dynamics and higher noise levels [8]. With symbolic approaches, the system is typically assumed to move between discrete steady states [17]. Here, diagnosis is often based on system snapshots without a history. Krysander [8] proposed a hybrid model consisting of discrete fault modes that switch between differential equations to describe the system behavior. The diagnosis system is designed for large systems with low noise levels, where instantaneous statistical tests are sufficient to identify a faulty component.

As Dearden and Clancy [3] pointed out, the close coupling between a mobile system with its environment makes it hard to apply discrete diagnosis models directly, because extremely complex monitoring components would have to be used. A more robust and efficient way is to reason directly on the continuous sensor readings. As a result, probabilistic state tracking techniques have been applied to this problem. Adopted paradigms range from Kalman filters [16] to particle filters in various modelings [1, 3, 12]. Particle filters represent the belief about the state of the system by a set of state samples, which are moved when actions are performed and re-weighted when sensor measurements are integrated (see Dellaert *et al.* [4]). In particle filter based approaches to failure diagnosis, the system is typically modeled by a dynamic mixture of linear processes [2] or a non-linear Markov jump process [5]. Benanzera *et al.* [1] combine consistency-based approaches, i.e. the Livingstone system, with particle filter based state estimation techniques.

Verma *et al.* [15] introduce the variable resolution particle filter for failure detection. Their approach is similar to ours in that they build an abstraction hierarchy of system models. The models of consideration build a partition of the complete state space and the hierarchy is defined in terms of behavioral similarity. Our focus in contrast lies on switching between overlapping system models for certain parts on the state space. Our model hierarchy is therefore based on efficiency differences and explicit model assumptions about the system state. The two approaches should therefore be seen as complementary rather than alternatives.

Other approaches that deal with the time efficiency of particle filters include Kwok *et al.* [10] in which real-time constraints are considered for single system models or techniques in which a Rao-Blackwellized particle filter is used to coordinate multiple models for tracking moving objects [9].

3 Particle Filters for Sequential State Estimation

A mobile robot can be modeled as a dynamic system under the influence of issued control commands u_t and received sensor measurements z_t. The temporal evolution of the system state x_t can be described recursively using the formalism of the so called Bayes filter

Algorithm 1 Particle_filter($\mathcal{X}_{t-1}, u_{t-1}, z_t$)

```
1: $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2: for $m = 1$ to $M$ do
3:     sample $x_t^{[m]} \sim p(x_t \mid u_{t-1}, x_{t-1}^{[m]})$
4:     $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
5:     $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
6: end for
7: for $m = 1$ to $M$ do
8:     draw particle $i$ with probability $\propto w_t^{[i]}$
9:     add $x_t^{[i]}$ to $\mathcal{X}_t$
10: end for
11: return $\mathcal{X}_t$
```

$$p(x_t \mid z_{0:t}, u_{0:t-1})$$

$$= \int p(x_t \mid z_t, u_{t-1}, x_{t-1})\, p(x_{t-1} \mid z_{0:t-1}, u_{0:t-2})\, dx_{t-1} \tag{1}$$

$$= \eta_t \quad \underbrace{p(z_t \mid x_t)}_{\text{observation model}} \quad \int \underbrace{p(x_t \mid u_{t-1}, x_{t-1})}_{\text{motion model}}\, \underbrace{p(x_{t-1} \mid z_{0:t-1}, u_{1:t-2})}_{\text{recursive term}}\, dx_{t-1}. \tag{2}$$

The term η_t is a normalizing constant ensuring that the left-hand side sums up to one over all x_t. With Equation 1, we assume Markovian dependencies, namely that x_t only depends on the most recent measurement z_t and control command u_{t-1} given knowledge about the preceding state of the system x_{t-1}. Particle filters are an implementation of the Bayes filter and can be used to efficiently estimate the posterior $p(x_t)$ in a sequential manner. Here, the posterior is represented by a discrete set of weighted samples $\mathcal{X} = \{\langle x_t^{[m]}, w_t^{[m]} \rangle\}$. With the sampled representation, the integral in Equation 2 simplifies to a finite sum over the samples resulting from the previous iteration. The motion model and the observation model can be applied directly to move and weight the individual samples respectively. Algorithm 1 formulates the standard particle filtering approach [14].

In Algorithm 1, the state of the system at time t is represented by a set $\mathcal{X}_t$ of state samples $x_t^{[m]}$. In Line 3, we perform a state prediction step using the external motion command u_{t-1} and the motion model $p(x_t \mid u_{t-1}, x_{t-1}^{[m]})$. Line 4 incorporates the current sensor measurement z_t by re-weighting the state samples according to the measurement model $p(z_t \mid x_t^{[m]})$. From Line 7 to 10, a resampling step is performed to concentrate the samples on high-probability regions of the state space. We refer to Thrun *et al.* [14] for details about the resampling step and its efficient implementation.

For the odometry-based model that treats the odometry measurements as controls, we have $u_{t-1} = o_t$ and $z_t = l_t$, where o_t is the odometry measurement and l_t is a perception of the environment. For the dynamic model depicted in the right diagram of Figure 2, u_{t-1} is the actual control command and $z_t = \langle o_t, l_t \rangle$. Both models are described in more detail in Section 3.3.

3.1 Process Model Hierarchy

A fundamental problem in science and engineering is choosing the right level of abstraction for a modeled system. While complex and high-dimensional models promise high estimation accuracy, models of less complexity are often significantly more efficient and easier to construct. In the area of mobile robotics, the accuracy-efficiency trade-off is an important issue, since on the one hand, computational resources are strictly limited in online problems and on the other hand, estimation errors have to be avoided to prevent serious malfunctioning. We therefore propose an online model selection algorithm with adaptive resource allocation based on the Bayesian framework.

An abstraction hierarchy for process models is given by the specific assumptions that the different models make about the world (compare Figure 2). We define the *model abstraction hierarchy* as an acyclic directed graph with the different system models M_i as nodes and their model assumptions $A_{i \to j}$ as edges, leading from the more general model i to a more restricted one j. A model assumption $A_{i \to j}$ is defined as a binary function on the state space of the unrestricted model M_i.

As an example, consider the process model for a mobile robot that should be able to continuously localize itself in a given map. A general model M_0 would include the the pose of the robot x and additionally take physical factors like ground friction, tire pressure, load balance, motor characteristics, etc. into account and treat those as additional state variables in a state vector f. In most situations, however, it is quite common and reasonable to assume a simpler model M_1, where these additional variables are constant and do not need to be estimated during operation. Formally, the state space of M_1 is therefore $\{x, f \mid f = \mathit{const}\}$, which is a projection of the more general space $\{x, f\}$ of model M_0. The assumption $A_{0 \to 1}$ would in this case be defined as

$$A_{0 \to 1}(x, f) := \begin{cases} \text{true} & : \quad f = \mathit{const} \\ \text{false} & : \quad f \neq \mathit{const}. \end{cases}$$

It is important to note that the validity of an assumption can only be tested in a less restricted state space, where this assumption is not made. In practice, this means that we have to test for every edge in the model abstraction graph the associated assumption using the more general model. As a measure for the validity of an assumption $\tilde{A}$ at time t, we use the ratio

$$v_t(\tilde{A}) := \frac{\sum_A p(z_t \mid x^{[m]}) \, \frac{1}{|A|}}{\sum p(z_t \mid x^{[m]}) \, \frac{1}{|\mathcal{X}|}} , \tag{3}$$

where A is defined as the subset $\{x^{[m]} \mid \tilde{A}(x^{[m]})\}$ of all particles $\mathcal{X}$ for which $\tilde{A}$ is valid. More informally, we compute the amount of evidence in favor of

Algorithm 2 Mixed-Abstraction_Particle_filter

```
1: Calculate samples for the unrestricted model $M_i$ until the assumption $A_{i\to j}(\tilde{x}_t)$
   is valid for a minimal number of samples $\tilde{x}_t$
2: Build a first estimate of $v_t(A_{i\to j})$ according to Equation 3
3: **repeat**
4:    **if** $v_t(A_{i\to j}) >= \Theta$ **then**
5:       Calculate samples for $M_j$
6:    **else**
7:       Calculate samples for $M_i$
8:    **end if**
9: **until** either $M_i$ received enough samples or ($v_t(A_{i\to j}) >= \Theta$ and $M_j$ received
   enough samples)
```

a restricted state space relative to the unrestricted case. The quantity $v_t(\tilde{A})$ is based on the current approximation of the posterior distribution by the particle filter.

3.2 Adaptive Model Selection

To adaptively switch between alternative system models, the validity of the model assumptions have to be estimated online and computational resources have to be distributed among the appropriate models. The distribution of resources is done by increasing or decreasing the number of particles for the different models. To achieve this, we apply the following algorithm that takes as input the *model abstraction hierarchy* graph defined in the previous section. When a new measurement z_t is obtained, the mixed abstraction particle filter algorithm draws samples from the particle set representing the current posterior $\mathcal{X}^i_{t-1}$ for all system models i, incorporates the measurement, and builds new posterior distributions $\mathcal{X}^i_t$. The key question in this update step is which model posterior $\mathcal{X}^i_t$ should receive how many samples. We base this decision on the estimated validity of the model assumptions $A_{i\to j}$. If the estimated quantity $v_t(A_{i\to j})$ drops below a predefined threshold Θ, we sample into the more complex model M_i and otherwise prefer the more efficient one M_j. This decision is made repeatedly on a per particle basis until a model has received enough samples and all its assumptions are validated. In each iteration, we start with the most unrestricted model M_i and perform the following steps for each of its outgoing edges $A_{i\to j}$.

In the update steps mentioned above, the samples are taken from the previous posterior distributions $\mathcal{X}^j_{t-1}$, if assumption $A_{i\to j}$ was valid in the previous step and from $\mathcal{X}^i_{t-1}$ otherwise. When all outgoing edges $A_{i\to j}$ of model M_i have been processed in the described manner, the same update is applied to models M_j further down in the hierarchy until either the leaf nodes have been processed or one of the assumptions did not receive enough evidence to justify further model simplifications.

Several quantities like the numbers of samples necessary for each model (Line 9) or the validity threshold Θ (Line 4) have to be determined in an offline learning step. For the experimental results reported below, we optimized these values on a set of representative trajectories, recorded from real and simulated robots.

To recapitulate, the mixed abstraction particle filter estimates the system state by running several particle filters in parallel, each using a different system model. Samples are assigned applying the following rule. For each two alternative system models, the simpler one is prefered as long as there is positive evidence for the validity of the corresponding model assumption.

3.3 Motion Models for Mobile Robots

The *standard odometry-based motion model* for a wheeled robot estimates the posterior $p(x_t \mid x_{t-1}, o_t)$ about the current pose x_t based on the previous pose x_{t-1} and the wheel encoder measurement o_t. A popular approach [6] to represent the relative movement is to use three parameters, an initial rotation α, a translation d, and a second rotation β as illustrated in Figure 3. Typically, one uses a Gaussian distribution for each of these parameters to model the noise.

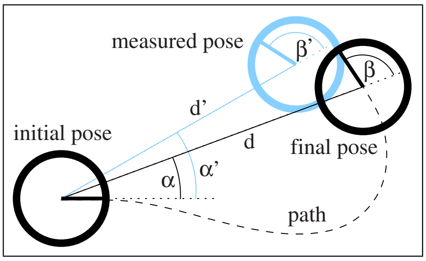

Fig. 3. Parameters of the standard odometry-based motion model.

Under the influence of events like the collision with an obstacle or wheel slippage, the odometry-based model is not applicable anymore since the wheel encoder measurements do not provide useful information about the actual motion. To handle such situations, we construct an alternative model, termed *dynamic motion model*, that depends on the actual motion commands that were sent to the motors of the robot. This model includes the geometry of the robot and its physical attributes like mass and moment of inertia. For each wheel, we compute the influence of the velocity command on the translational and rotational energy of the robot. In this representation, we can directly incorporate the effects of collisions, slippage, and deflating tires. We then convert the energy state of the system to its speed and obtain a state prediction for the next time step.

It is important to note that the dynamic motion model is not designed for the failure states only. Rather it is able to deal with normal conditions as well. It is therefore considered as more general as the standard odometry-based model in our model abstraction hierarchy. The assumption placed on the system state by the odometry-based model is, that there are no external influences like collisions, slippage, etc.

4 Experiments

4.1 Quantitative Evaluation Using a Simulator

To quantitatively evaluate our system, we performed several simulation experiments and compared the results to the known ground truth. We used the high-fidelity simulator Gazebo [7], in which physics and motion dynamics are simulated accurately using the Open Dynamics Engine [13]. In several practical experiments carried out with real robots, we experienced the Gazebo simulator as well suited for studying the motion dynamics of robots even under highly dynamic events like collisions. For example, we did not have to change any system parameters when we ported our system from the simulator to the real robot.

To demonstrate how the proposed algorithm coordinates multiple particle filters that have been designed independently, we confronted the system with two different faults within one scenario. A simulated ActivMedia Pioneer 2DX robot (see the left image of Figure 1) was placed in the corridor of a 3D office environment. We manually steered the robot through this environment. On its path, it encountered a collision with an undetectable object. After that, its left tire started to deflate. Four filters were used independently to track the state of the system. The first filter was based on the standard odometry-based motion model described in Section 3.3. The second filter used the dynamic motion model described in the previous section and also included a model for collision faults. The third model was also based on the dynamic motion model but was capable of dealing with deflating tires. Finally, the forth filter was the proposed mixed-abstraction filter that combined all of the filters described above.

Figure 4 depicts the true trajectory as well as the tracking results obtained with the individual filters overlayed on a map built from laser measurements in a real building. The three arrows in the diagram mark the following three events: a collision with an undetected glass door (1), a point where the robot stopped backing off and turned around (2), and a point where the left tire of the robot started losing air (3).

As can be seen from the figure, the filter that is able to deal with deflating tires diverges immediately after the collision at point 1. Since the filter able to deal with collisions cannot deal with deflating tires, it diverges at point 3. The odometry-based model keeps track of the robot despite the collision at point 1, however it is not aware of any fault at this point. The combined filter in contrast succeeds in tracking the robot despite of both incidents.

The middle and lower image of Figure 4 plot the internal states of the specialized detectors within the mixted-abstraction filter. These values reflect the belief of the system about the presence of certain faults. The middle image plots the relative number of particles in the fault mode of the collision detector over time. As can be seen, this number raises significantly shortly after the collision as well as after the full stop at point 2. After the robot had been intenionally stopped there, the system cannot know whether an obstacle is in its way or not. The lower image of Figure 4 shows the evolution of the relative number of particles in the fault mode of the deflation detector. Since the collision at point 1 has been handled by the collision detection within the mixed-abstraction filter, this filter does not switch to a failure mode until point 3. At that point, the filter switches into its failure mode and in this way enables the mixed-abstraction filter to keep track of the pose of the robot.

4.2 Analyzing the Gain in Efficiency

In this experiment, we quantatively evaluate the gain in efficiency that we achieve by dynamically distributing samples between the individual filters. In the modeled scenario, a simulated robot follows a trajectory and collides twice with an obstacle, which is too small to be detected by its laser range finder. After the collisions, the robot backs off and continues its task. The top diagram of Figure 5 shows the trajectory of the vehicle and the locations where the algorithm correctly detected a collision. The other two diagrams illustrate the failure detection process of the same simulation run. The bar at the bottom indicates the true time stamps of the faults. Whereas the plot in the second row depicts the relative likelihood for a collision as defined in Equation 3, the curve plotted in the third row gives the times needed for the individual iterations of the particle filter.

Table 1 gives the results of a comparison of our adaptive model-switching approach to three other implementations, where only single models were used for state estimation and fault detection. The results are averaged over the full trajectories of 100 runs per implementation. The implementations considered here are realized on the basis of two models. Whereas model M_0 is the complex model that considers the actual motion commands and therefore is able to track the position as well as the failure state, model M_1 is the standard odometry-based system model, which is able to track the position of the robot reliably with low time requirements, but cannot detect the collisions. The first implementation is based on model M_1 with 20 particles, the second is model M_1 with 200 particles. While the third implementation is based on model M_0 with 300 particles, the forth one is the mixed-abstraction particle filter combining implementation one and three. The common task of all implementations was to track the position of the robot along a trajectory on which the robot encountered two undetected collisions after 8.2 and 28.9 seconds. A typical estimate of the trajectory generated by the mixed-abstraction filter including markings for detected collisions is depicted in the left diagram of

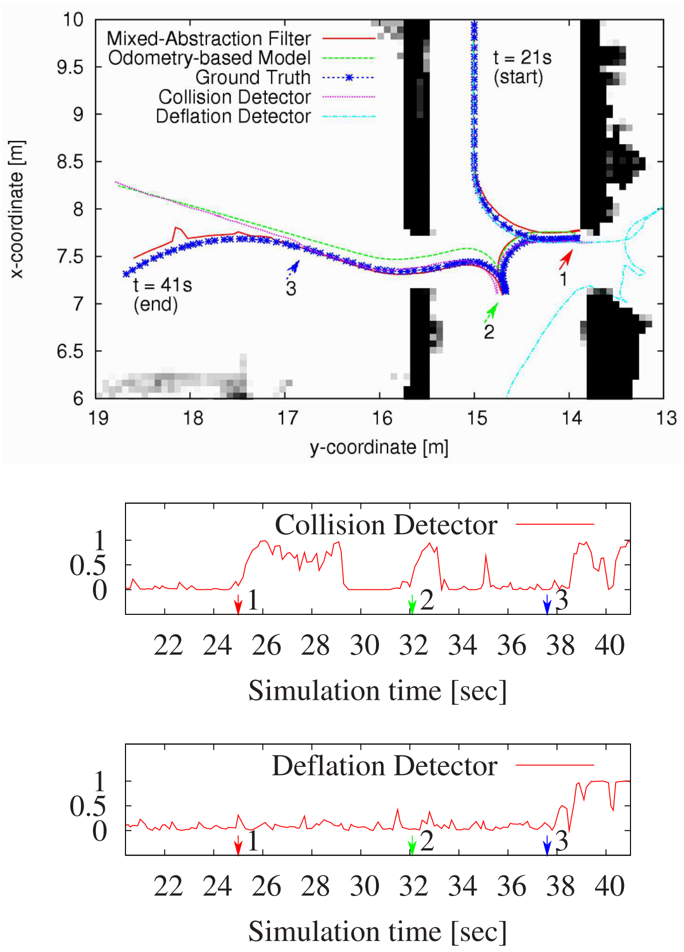

Fig. 4. Results of an experiment with two fault-events (top). The collision of the robot with a glass door is marked with the arrow "1", the point where it stopped backing of and turned is marked with "2", and at point "3" the left tire of the robot started losing air. The diagram in the middle plots the relative number of the particles in the fault mode of the collision detector. The lower diagram shows the corresponding number for the deflation detector.

Figure 5. The lower right image of Figure 5 plots the value of $v_t(A)$ over time. As can be seen from the figure, the evidence for a fault substantially increases at the time of the incidences. The upper right image in the same figure plots the CPU time used by the mixed-abstraction filter. It nicely shows that the computation time is only high when the evidence of a failure has increased.

Table 1 shows average values obtained from the 100 test runs for each implementation. Whereas model M_1 was never able to detect a failure, M_0

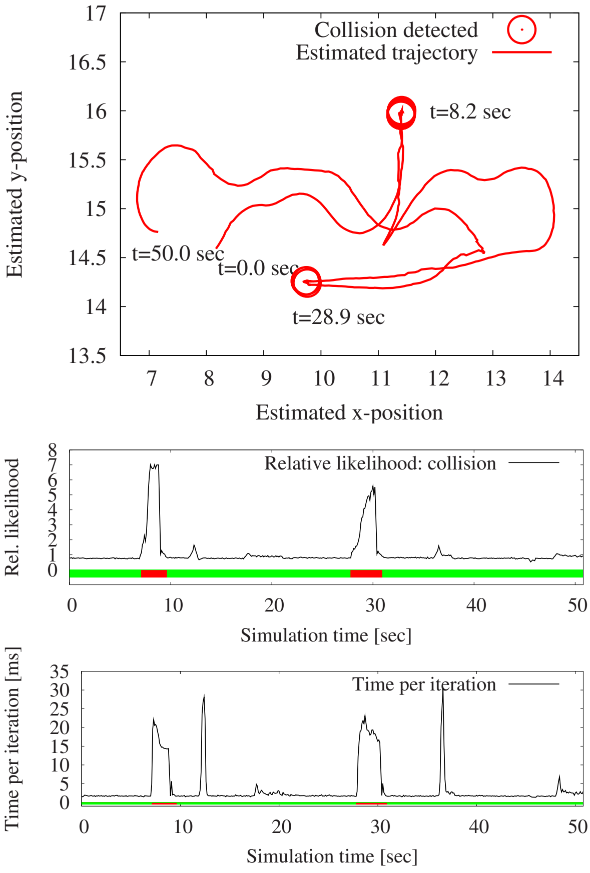

Fig. 5. A trajectory followed by a simulated robot (first row) with marks at positions where the evidence for a collision was high. The plot in the second row depicts the relative likelihood for collisions. The plot also shows the ground truth (the bar at the bottom). The last plot shows the time needed for each iteration (third row).

as well as our adaptive switching algorithm detected all failures equally well. However, our adaptive model required substantially less computation time compared to M_0 alone using 300 particles.

4.3 Evaluation on a Real Robot

We also tested our system on a real ActivMedia Pioneer 2DX robot in an office environment. The right image of Figure 1 depicts the experimental setup. Three positions of the robot were manually cut from a recorded video and overlayed on one image to illustrate the process. The robot planned a path to the neighboring room on the right-hand side of the corridor. While executing the planned trajectory, the robot could not detect the glass door that

Table 1. Results of a series of simulation runs using different system models for state estimation. The results are averaged over the complete trajectories of 100 runs per model.

System model	Failure Detection rate	Average time per iteration	Average estimation error	
			Position	Orientation
M_1: standard odometry 20 particles	0 %	0.67 ms	0.13 m	3.6°
M_1: standard odometry 200 particles	0 %	5.83 ms	0.13 m	3.4°
M_0: dynamic 300 particles	100 %	10.10 ms	0.11 m	5.8°
adaptive-switching: M_1: 20 particles M_0: 300 particles	100 %	3.42 ms	0.12 m	3.9°

blocked its path and thus collided with the wooden part of the door. In this situation, the standard odometry-based system model used for localization does not indicate a defect, because the wheel encoders report no motion, which is perfectly consistent with the laser measurements. The left diagram of Figure 6 gives the evolution of the observation likelihoods for the standard model, which stays nearly constant. In contrast to this, the proposed mixed abstraction particle filter detects that the model assumption of the standard model is not valid anymore after the collision with the door and switches to the more complex system model. The right diagram of Figure 6 visualizes this process. For the sake of clarity, we plotted the estimated validity of the negated model assumption, which can be interpreted as the evidence against the assumption that no collision has occurred. The upper curve corresponds to the time needed per iteration. Note that the required computational resources only slightly exceed those of the standard odometry-based model (see left diagram for a comparison). Only in the failure case, the runtime increases seriously since the more complex model requires substantially more particles. Please also note, that the runtime goes back to the original value after the robot has backed off and the model assumption of the simplified model is valid again.

5 Conclusion

This paper presents an efficient approach to estimate the state of a dynamic system including its failures. Complex models with high computational requirements are needed in order to detect and track unusual behaviors. We therefore proposed a mixed-abstraction particle filter which distributes the computational resources in a way that failure states can be detected and tracked but at the same time allows us an efficient estimation process in case

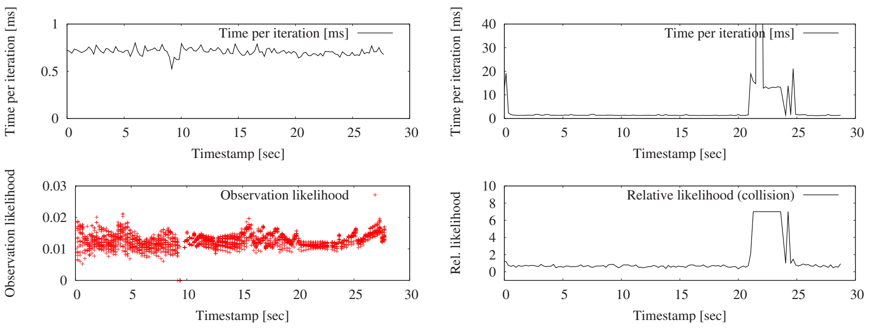

Fig. 6. In an experiment with a real robot, the observation likelihood of the standard system model (lower left) does not indicate the collision with a glass door at time $t = 22$ seconds. The mixed-abstraction particle filter (right) detects the fault without needing substantially more computational resources in the fault-free case (upper diagrams).

the systems runs free of faults. To achieve this, we apply a process model hierarchy which allows us to model assumptions that hold for the fault-free case but not in general.

In several experiments carried out in simulation and with real robots, we demonstrated that our technique is well-suited to track dynamic systems affected by errors. Our approach allows us to accurately track different failure states and at the same time is only marginally slower in case the system is running free of faults. We believe that our approach is not limited to the failure detection problem and can also be advantageous for various state estimation tasks in which different system models have to be used to correctly predict the behavior of the system under varying conditions.

Acknowledgments

This work has partly been supported by the EC under contract number FP6-004250-CoSy, by the German Science Foundation (DFG) under contract number SFB/TR-8 (A3) and the German Federal Ministry of Education and Research (BMBF) under contract number 01IMEO1F.

References

1. E. Benazera, R. Dearden, and S. Narasimhan. Combining particle filters and consistency-based. In *15th International Workshop on Principles of Diagnosis*, Carcassonne, France, 2004.
2. N. de Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole. Diagnosis by a waiter and a mars explorer. In *Invited paper for Proceedings of the IEEE, special issue on sequential state estimation*, 2003.

3. R. Dearden and D. Clancy. Particle filters for real-time fault detection in planetary rovers. In *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis*, pages 1–6, 2002.
4. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.
5. J.N. Driessen and Y. Boers. An efficient particle filter for nonlinear jump markov systems. In *IEEE Sem. Target Tracking: Algorithms and Applications, Sussex, UK*, 2004.
6. J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. of the IEEE/RSJ InternationalConference on Intelligent Robots and Systems*, 1998.
7. N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. technical report. Technical report, USC Center for Robotics and Embedded Systems, CRES-04-002, 2004.
8. M. Krysander. *Design and Analysis of Diagnostic Systems Utilizing Structural Methods*. PhD thesis, Linköping University, Sweden, 2003.
9. C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *RoboCup 2004: Robot Soccer World Cup VIII*, pages 18–33, 2004.
10. C. Kwok, D. Fox, and M. Meila. Real-time particle filters. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 1057–1064. MIT Press, 2002.
11. N. Leveson. *Safeware : System Safety and Computers*. Addison-Wesley Pub Co., Reading, Mass., 1995.
12. B. Ng, A. Pfeffer, and R. Dearden. Continuous time particle filtering. In *Proceedings of the 19th IJCAI, Edinburgh*, 2005.
13. R. Smith. Open dynamics engine. `http://www.q12.org/ode/ode.html`, 2002.
14. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
15. V. Verma, S. Thrun, and R. Simmons. Variable resolution particle filter. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 976–984. Morgan Kaufmann, 2003.
16. R. Washington. On-board real-time state and fault identification for rovers. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1175–1181, 2000.
17. B.C. Williams and P.P. Nayak. A model-based approach to reactive self-configuring systems. In Jack Minker, editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC*, College Park, Maryland, 1999. University of Maryland.