Efficient Failure Detection on Mobile Robots Using Particle Filters with Gaussian Process Proposals

Christian Plagemann[1] **Dieter Fox**[2] **Wolfram Burgard**[1]

[1] Albert-Ludwigs-University
Department of Computer Science
Freiburg, Germany
{plagem,burgard}@informatik.uni-freiburg.de

[2] University of Washington
Department of Computer Science & Engineering
Seattle, WA, USA
fox@cs.washington.edu

Abstract

The ability to detect failures and to analyze their causes is one of the preconditions of truly autonomous mobile robots. Especially online failure detection is a complex task, since the effects of failures are typically difficult to model and often resemble the noisy system behavior in a fault-free operational mode. The extremely low *a priori* likelihood of failures poses additional challenges for detection algorithms. In this paper, we present an approach that applies Gaussian process classification and regression techniques for learning highly effective proposal distributions of a particle filter that is applied to track the state of the system. As a result, the efficiency and robustness of the state estimation process is substantially improved. In practical experiments carried out with a real robot we demonstrate that our system is capable of detecting collisions with unseen obstacles while at the same time estimating the changing point of contact with the obstacle.

1 Introduction

The detection of and reaction to unforeseen failure events is a fundamental ability for autonomous systems as an improper treatment of such situations may cause loss of control and can possibly harm the system or the environment. Consider, for example, a mobile service robot that collides with a low doorstep or a planetary rover that hits an unforseen small rock. In such situations, the mobile robot should be able to (a) detect the unexpected event quickly and (b) to infer the cause of the event in order to react appropriately. Many successful previous approaches to the online failure detection problem are based on the sequential Monte-Carlo method. They typically track multiple system modes in parallel before they decide on the most probable one. A crucial problem for such sampling-based approaches especially in real-time scenarios, however, lies in the high dimensionality of the state space to be tracked and the low frequency and unpredictability of failure events. To increase the efficiency of sample-based representations, different techniques such as lookahead sampling [de Freitas *et al.*, 2003], risk sensitive sampling [Thrun *et al.*, 2001], and hierarchical sampling [Verma *et al.*, 2003] have been proposed. In this paper, we propose an alternative approach, in which Gaussian processes (GP) are used to learn proposal distributions that represent informed guesses about the failure mode and its parameters. We demonstrate how GP classification can be used to predict discrete failure modes and how GP regression can be used to model continuous failure parameters. Since Gaussian processes provide full predictive distributions including the predictive uncertainties, they can readily be used as proposal distributions in the particle filter implementation. By including the most recent sensor measurements received by the robot, these proposals can be seen as an approximation of the optimal proposals for the respective state variables.

Our system does not require additional hardware such as inertial sensors or bumper rings, which may be expensive or otherwise unfavorable in certain domains, e.g., for aerial blimps. The proposed algorithm has been implemented on a real robot and can be executed in real-time. We discuss results from several test runs in realistic scenarios in which we compared the achievable detection rates and pose tracking quality of our system with an optimized standard particle filter implementation without informed proposals.

This paper is organized as follows. After discussing related work, we describe particle filters for sequential state estimation and failure detection in Section 3. Our Gaussian process proposals are introduced in Section 4, followed by the experimental results.

2 Related Work

The close coupling between a mobile system and its environment makes it hard to detect abnormal behavior using instantaneous statistical tests only without tracking possible failure modes over time [Dearden and Clancy, 2002]. For this reason, probabilistic state tracking techniques have been applied to this problem. Particle filters represent the belief about the state of the system by a set of state samples [Thrun *et al.*, 2005]. In particle filter based approaches to fault diagnosis, the system is typically modeled by a non-linear Markov jump process [Driessen and Boers, 2004] or a dynamic mixture of linear processes [de Freitas *et al.*, 2003]. The look-ahead particle filter introduced in the latter work also approximates the optimal proposal distribution by considering the most recent sensor measurements, but in contrast to our work focuses on the case of discrete failure modes without continuous failure

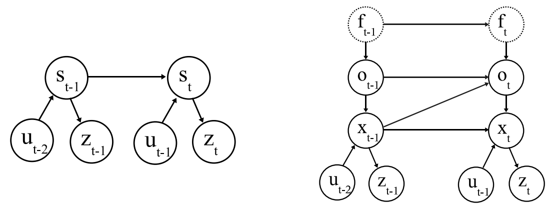

Figure 1: Graphical model for the dynamic system in an abstract view (left) and a more detailed form (right) where the system states s_t are split up to include the discrete failure mode f_t, the failure parameters o_t, and the robot's state x_t.

parameters. Benanzera *et al.* [2004] combine consistency-based approaches, i.e., the Livingstone system, with particle filter based state estimation techniques. Verma *et al.* [2003] introduce the variable resolution particle filter for failure detection. Their approach is to 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. A different kind of model abstraction hierarchy based on explicit model assumptions was developed in [Plagemann *et al.*, 2006] to detect failures online. Other approaches that deal with the time efficiency of particle filters include [Kwok *et al.*, 2002] in which real-time constraints are considered for single system models.

Gaussian processes have been widely studied in the Machine Learning community. Excellent introductions have been given by [MacKay, 1998] and [Rasmussen, 1996]. [Girard *et al.*, 2002] present a Gaussian process based model for multiple step ahead prediction of time series. Classification models are discussed in [Neal, 1997]. [Schwaighofer *et al.*, 2003] present an application of Gaussian processes to localization using signals of a wireless phone network.

3 Sequential State Estimation

The temporal evolution of dynamic systems such as mobile robots can be described using the graphical model depicted in the left diagram of Figure 1. In this figure, s_t denotes the state of the system at time t, z_t stands for received sensor measurements, and u_t is the issued control command. Given the modeled independence assumptions, the state of the system can be evaluated recursively using the Bayes Filter formalism [Thrun *et al.*, 2005]

$$\begin{aligned} &p(s_t \mid z_{0:t}, u_{0:t-1}) \\ &= \eta_t \underbrace{p(z_t \mid s_t)}_{\text{observation model}} \int \underbrace{p(s_t \mid s_{t-1}, u_{t-1})}_{\text{transition model}} \underbrace{p(s_{t-1} \mid z_{0:t-1}, u_{0:t-2})}_{\text{recursive term}} ds_{t-1} \end{aligned} \quad (1)$$

Particle filters are widely used as a sample based implementation of this recursive filter. They represent the state of the system by a set of weighted samples $\mathcal{X} = \{\langle s_t^{[i]}, w_t^{[i]} \rangle\}$. With the sample-based representation, the integral in Equation 1 simplifies to a finite sum over the samples resulting from the previous iteration. The transition model and the observation model can be applied directly to predict and weight the individual samples respectively.

3.1 Modeling Failure Events and Parameters

For dynamic systems under the influence of external or internal failure events, the state s_t of the system can be split up to include the discrete failure mode f_t, the continuous failure parameters o_t, and the remaining part of the robot's state x_t. We assume the commonly used constant failure rate model (see [Ng *et al.*, 2005]), where the failure event itself does not depend on the rest of the system. The right diagram of Figure 1 shows the underlying graphical model. Since the observations are independent of the failure state (f_t, o_t) given the state x_t of the robot, the observation model simplifies to $p(z_t \mid s_t) = p(z_t \mid x_t)$. The transition model can be factorized as

$$p(s_t \mid s_{t-1}, u_{t-1}) = p(x_t, f_t, o_t \mid x_{t-1}, f_{t-1}, o_{t-1}, u_{t-1}) \quad (2)$$

$$= \underbrace{p(f_t \mid f_{t-1})}_{\text{failure event model}} \cdot \underbrace{p(o_t \mid f_t, x_{t-1}, o_{t-1})}_{\text{failure parameter model}} \cdot \underbrace{p(x_t \mid o_t, x_{t-1}, u_{t-1})}_{\text{robot motion model}} . \quad (3)$$

The constant failure rate model states that failure events are distributed exponentially depending on a failure rate parameter λ by

$$p(f_t) = 1 - e^{-\lambda \cdot (t - \tilde{t})} , \quad (4)$$

where $\tilde{t}$ denotes the time of the last failure event. It can easily be shown that for such a model, the mean time between failures becomes $MTBF = \frac{1}{\lambda}$. For realistic failure rates λ, this model results in extremely low failure probabilities per filter iteration. Assume, for example, a mean time of 30 minutes between collisions of a service robot with unseen obstacles. This implies $\lambda = \frac{1}{1800s} = 0.000\overline{5}$ and with a filter frequency of $\delta t = 0.1$ seconds yields a failure probability of $p(f_t \mid \neg f_{t-1}) \approx 0.000056$ within one iteration. For such a small value, just one of $20,000$ particles would jump to a failure mode on average. Thus, one would either need an extremely large particle set or would risk that failures remain undetected. This problem is amplified by the fact that not only the discrete failure mode has to be sampled, but also the unknown continuous failure parameters. Since in general, there is no prior knowledge about the parameters of randomly occurring failures, we assume a uniform distribution

$$p(o_t \mid f_t, x_{t-1}, o_{t-1}) = \mathcal{U}_{[o_{min}, o_{max}]}(o_t) \quad (5)$$

over a certain interval. Note that this model applies only to the case where the system transitions into a failure state. The evolution of failure parameters within a failure state is typically governed by a much more peaked distribution similar to the motion model of the robot. In Section 5, we describe our model for the evolution of collision parameters based on rigid body dynamics. This model is able to track collisions sufficiently accurate, if the initial collision parameters have been estimated well enough. The main focus of this work therefore is to improve the detection of failure events and to efficiently estimate the initial parameters.

To address the problem of low sampling probabilities for important parts of the state space, [Thrun *et al.*, 2001] introduced the risk sensitive particle filter that incorporates a learned risk function to force the filter into less likely but important states. While this approach ensures a reasonable amount of samples in the important failure modes, it cannot adapt to the specific situation the robot is in when the sampling decision is made. We therefore propose to use learned proposal distributions to be able to make informed guesses about the discrete failure mode and the continuous failure parameters online.

3.2 Data-driven Proposal Distributions

In the sequential importance sampling scheme (see [Doucet, 1998]) an arbitrary proposal distribution can be used to directly sample the relevant areas of the state space as long as (a) all possible states have a non-zero possibility of being chosen and (b) the importance weights of the particles are adjusted appropriately. Proposal distributions that depend on the most recent sensor measurements or on features extracted by separate algorithms are typically denoted as *data-driven proposals* or *detector-mediated proposals* (see [Khan *et al.*, 2004]). Such proposals aim at approximating the optimal proposal $p(s_t \mid s_{t-1}, z_t, u_{t-1})$ which fully includes the most current sensor measurement z_t. With the process model defined in Equations 1 and 3, the weight for particle i at time t becomes

$$\begin{aligned} w_t^{[i]} &= \frac{p(s_{1:t}^{[i]} \mid z_{1:t})}{\pi(s_{1:t}^{[i]} \mid z_{1:t})} = \frac{p(z_t \mid s_{1:t}^{[i]}, z_{1:t-1})\, p(s_{1:t}^{[i]} \mid z_{1:t-1})}{\underbrace{p(z_t \mid z_{1:t-1})}_{=:1/\eta}\, \pi(s_{1:t}^{[i]} \mid z_{1:t})} \\ &= \eta \cdot \frac{p(z_t \mid s_t^{[i]})\, p(s_t^{[i]} \mid s_{t-1}^{[i]})}{\pi(s_t^{[i]} \mid s_{1:t-1}^{[i]}, z_{1:t})} \cdot \underbrace{\frac{p(s_{1:t-1}^{[i]} \mid z_{1:t-1})}{\pi(s_{1:t-1}^{[i]} \mid z_{1:t-1})}}_{=w_{t-1}^{[i]}} \\ &= \eta \cdot w_{t-1}^{[i]} \cdot p(z_t \mid s_t^{[i]}) \cdot \frac{p(x_t^{[i]} \mid o_t^{[i]}, x_{t-1}^{[i]})}{\pi(x_t^{[i]} \mid o_t^{[i]}, x_{t-1}^{[i]})} \cdot \\ &\quad \frac{p(f_t^{[i]} \mid f_{t-1}^{[i]})}{\pi_f(f_t^{[i]} \mid F_t)} \cdot \frac{p(o_t^{[i]} \mid f_t^{[i]}, x_{t-1}^{[i]}, o_{t-1}^{[i]})}{\pi_o(o_t^{[i]} \mid f_t^{[i]}, F_t)} \end{aligned} \qquad (6)$$

where π is the proposal distribution forto the transition model defined in Equation 3 with the failure event model exchanged for π_f and the failure parameter model exchanged for π_o. Here, π_f and π_o are the learned proposal distributions for the discrete f_t and the continuous o_t respectively. The control commands u_t have been omitted for clarity, they are straightforward to include. The normalizing factor η is constant for all particles i. Furthermore, F_t denotes an arbitrary feature vector extracted at time t. Any feature vector F_t as well as any functions π_f and π_o can be used for this task as long as the assumptions

- $\pi_f(f \mid F_t) \neq 0$ for all f with $p(f \mid s_{1:t}, z_{1:t-1}) \neq 0$
- $\pi_o(o \mid f_t, F_t) \neq 0$ for all o with $p(o \mid s_{1:t}, z_{1:t-1}) \neq 0$.

hold, which means that all possible failure states have to be assigned a non-zero probability of being chosen. Visually speaking, Equation 6 states that after each filter iteration, the particle weights have to be multiplied with the current observation likelihood $p(z_t \mid s_t^{[i]})$ and with two correction terms for the two learned proposal distributions. To calculate these correction terms for a specific sample $s_t^{[i]}$, we divide the probabilities defined in Equations 4 and 5 by the likelihoods according to which the state variables $f_t^{[i]}$ and $o_t^{[i]}$ have been drawn from π_f and π_o. Another precondition for the learned proposals therefore is the availability of likelihoods for sampled values. As we will see in the next section, Gaussian processes always ensure the non-zero probability assumptions, can easily be sampled from, and provide the transition probabilities needed for the weight correction described above.

4 Gaussian Processes Proposals for Discrete and Continuous State Variables

Gaussian processes are a powerful tool for learning and representing full predictive distributions for non-linear functions [Rasmussen, 1996] and have also proven successful in solving classification problems. Given a training set and priors on the function to be learned, the Gaussian process model allows to predict function values at new input locations and also supplies predictive uncertainties. We first introduce the regression and classification models as well as their role in the general failure detection setting described above. In Section 4.3, we then describe a specific application in detail, the detection and tracking of collision events for mobile robots.

4.1 Non-linear Regression

Assuming a non-linear functional dependency g between m-dimensional input vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and real-valued targets $t_1 \ldots, t_n$ such that $t_i = g(\mathbf{y}_i) + \epsilon_i$ for independent error terms ϵ_i, the task is to predict future targets t_{n+j} at new input locations $\mathbf{y}_{n+j}$. The idea of Gaussian processes is to view all target values $t_1, \ldots, t_{n+j}$ as jointly Gaussian distributed with an $(n+j)$-dimensional mean and a covariance $cov(t, t') = k(\mathbf{y}, \mathbf{y}')$ specified via a kernel distance function k on the input vectors. Predictions for a new query point $\mathbf{y}_{n+1}$ can thus be performed by conditioning on the known target values $t_1, \ldots, t_n$, see [Neal, 1997]. The mean μ and variance σ^2 of the Gaussian predictive distribution for t_{n+1} turn out to be

$$\mu = E(t_{n+1} \mid t_1, \ldots, t_n) = \mathbf{k}^T C^{-1} \mathbf{t} \qquad (7)$$

$$\sigma^2 = V(t_{n+1} \mid t_1, \ldots, t_n) = v - \mathbf{k}^T C^{-1} \mathbf{k} \qquad (8)$$

with C the $n \times n$ covariance matrix for the given targets, $\mathbf{t}$ the n-dimensional vector of given target values, and $\mathbf{k}$ the n-dimensional vector of covariances between the new target value t_{n+1} and the known targets $t_1, \ldots, t_n$.

In our failure detection setting, we can use this regression model to predict the continuous parameters of new failure events. We take extracted features F_t as input vectors $\mathbf{y}$ and failure parameters o_t as targets t. Given a training set of such quantities and a properly chosen covariance function k, we can compute the predictive distribution $\pi_o(o_t \mid f_t, F_t) := \mathcal{N}(\mu, \sigma^2)$ as detailed in Equations 7 and 8. This (normal) distribution naturally meets the requirements for proposal distributions named in the previous section. It can be sampled from directly, it supplies the likelihoods of sampled values, it has an infinite support, and it therefore only assigns non-zero likelihoods.

An important aspect that has been left out so far is the choice of covariance function and how its parameters can be set. The covariance function plays an important role in the Gaussian process framework as it represents the prior knowledge about the underlying function g. By changing its form and parameters, one can control the generalization behavior and smoothness of the predictor. A common choice of covariance function (see [MacKay, 1998]) that is also used in this

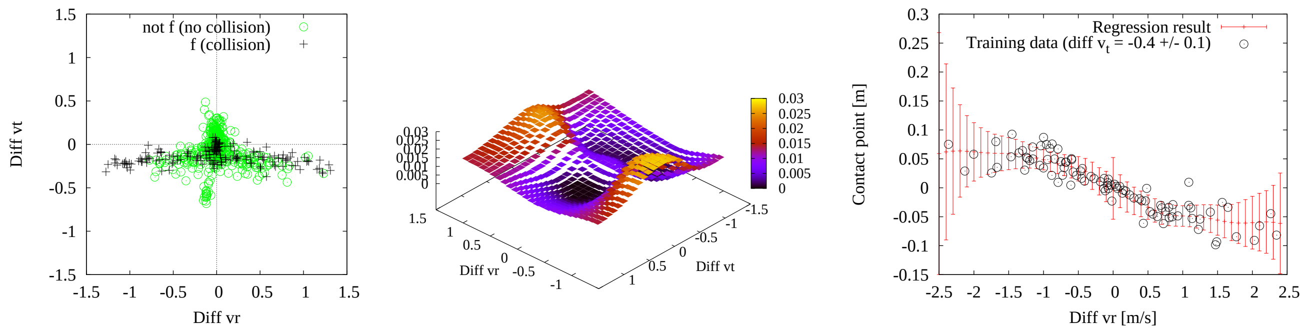

Figure 2: The collision event training set where two feature values are mapped to class instances (left), the learned class probabilities $\pi_f(F_t)$ using Gaussian process classification (middle), and the learned regression model for the collision parameters visualized by a cut through the 2-dimensional pdf that maps velocity deviations to contact points for a collision (right).

work is

$$k(\mathbf{y}, \mathbf{y}') := a_0 + v_0 \cdot \exp\left(-\frac{1}{2} \sum_{d=1}^{m} w_d (y_d - y'_d)^2 \right)$$

with a constant component and a non-linear, stationary term, which depends only on the distance between input points. The parameters of the covariance function are called hyperparameters of the Gaussian process. They can either be fixed by maximizing the likelihood of the given data points or, for fully Bayesian treatment, can be integrated over using parameter-specific prior distributions. For our experiments reported in Section 5, we employed the latter strategy with Gamma priors on the hyperparameters. We set these priors to favor smooth regression functions to avoid overfitting to the training data. The analytically intractable integration over the hyperparameters is approximated using Markov Chain sampling and the prediction results are cached on a fine-grained grid.

4.2 Binary Classification

Binary classification problems can consistently be modeled in this framework by including for every binary target t_i a real-valued latent variable l_i, such that

$$p(t_i = 1) = \frac{1}{1 + e^{-l_i}} \; , \tag{9}$$

which is known as the logistic model that links class probabilities to real values, see [Neal, 1997]. The latent variables can now be given a Gaussian process prior as in the regression setting and predictions of class probabilities can be performed by predicting the corresponding latent variables and evaluating Equation 9. For the failure detection problem, we again use feature vectors F_t as inputs and binary failure labels f_t as targets. The predicted class probabilities for new features then directly define the failure event proposal $\pi_f(f_t \mid F_t)$.

4.3 Learning to Predict Collision Events and Parameters

In this section, we apply the sequential failure detection approach described above to the hard problem of collision detection for mobile robots under noisy sensor measurements and without additional hardware like bumpers or inertial sensors. For learning the collision event proposal and the proposal for the contact point of the robot with the obstacle, we achieve excellent results with simple features based on the rotational and the translational velocity of the robot. Thus, we use as input to the Gaussian process classifier the two-dimensional feature vector $F_t = (\Delta v_t, \Delta v_r)$, where Δv_t is the difference between the translational velocity estimated by the particle filter and the one estimated by local laser scan matching. Furthermore, Δv_r is the difference of the rotational velocities respectively. A training set of 500 automatically labeled trajectories was generated by simulating random collisions with different obstacles using the 3D simulator Gazebo [Koenig and Howard, 2004]. Whereas the left diagram of Figure 2 depicts the gathered data points, the middle diagram in the same figure shows the learned class probabilities depending on the two velocity differences described above. As can be seen from the left diagram, Δv_t is negative for nearly all "collision" data points, which corresponds to the fact that the robot is slowed down when a collision occurs. The data points for "no collision" are spread widely and do not separate well from the "collision" data points due to noisy sensor measurements and imperfect labeling of the collision events. This makes the classification problem a hard one. It should be stressed, that we use this classifier as a proposal distribution for collisions rather than as a collision detector directly, because the features are too ambiguous to allow for perfect instantaneous classification. Experiments with a real robot (see Section 5) showed that this yields high detection rates with a low number of false alarms.

Given a collision event, the continuous collision parameters o have to be estimated to simulate the effects on the system and to continue the tracking process. Since the task is not to fully track the pushed obstacle over time, a simple model that abstracts from the obstacle's geometry and exact pose has proven sufficient to describe the effects on the robot. A collision with an unseen obstacle is represented by the obstacle mass m and the contact point c on the front of the robot. Therefore, the collision parameters are $o = (m, c)$. We learn the proposal distribution $\pi_o(o_t | F_t)$ for the parameters o_t using the same velocity-based features and simulated training set as described above and the Gaussian process re-

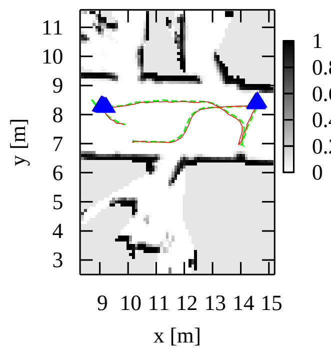

Figure 3 (a): Two correctly detected collisions (triangles), the estimated trajectory (solid line), and the ground truth (dotted).

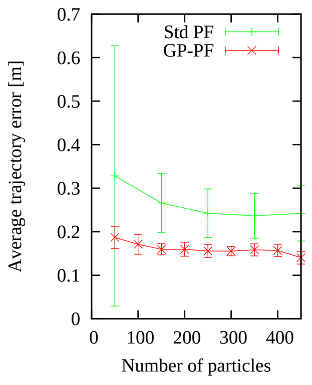

Figure 3 (b): Average deviation of the estimated trajectory from the ground truth in meters for a varying number of particles.

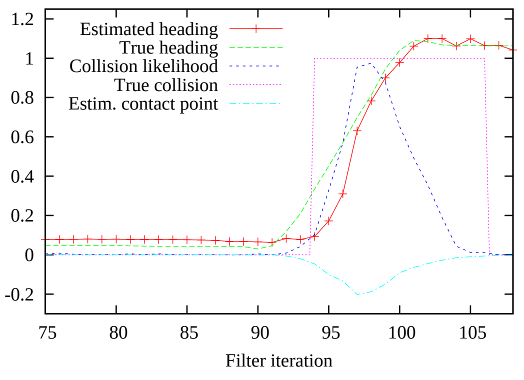

Figure 3 (c): Estimation details around a correctly detected collision including the manually labeled true collision event. One filter iteration corresponds to 0.1 seconds.

gression technique. The right diagram of Figure 2 depicts a cut through the learned 2-dimensional distribution for the collision parameter c, which is the contact point of the obstacle on the front of the robot. The point of contact is measured in meters from the center of the robot's front to the right. It can be seen from the diagram that unexpected clockwise rotations ($\Delta v_r < 0$) of the robot are mapped to positive values for the contact point, which corresponds to a collision on the righthand side of the robot.

5 Experimental Results

Our system has been implemented on a real ActivMedia Pioneer 3DX robot and has been extensively tested in an office environment. Before presenting experimental results, we describe the motion model we implemented for our localization system. Since this model is not the focus of this work, we give only a brief overview here.

The most widely used motion model for mobile robots is based on the wheel encoder measurements (see [Thrun *et al.*, 2005]). This information, rather than the the actual control command, is taken as control input u_{t-1}, which under normal circumstances results in accurate predictions of the performed movement. Under the influence of failures like collisions or wheel slip, however, the motion of the wheels is not consistent with the whole robot's motion any more. A more appropriate model for such situations that still is efficient enough to be evaluated online is based on simple rigid body dynamics (see [Plagemann *et al.*, 2006]). We model the robot as a rigid body in the 2-dimensional plane, represented by a set of constant values and the variable state vector $x_t = (pos_x, pos_y, pos_\theta, vel_t, vel_r)$ which includes the translational velocity vel_t and the rotational velocity vel_r. In each filter iteration, the wheel thrusts are calculated from the actual velocity command that was sent to the motors. From this, the next state vector is computed by numerical simulation using the physical relationships between forces, acceleration, and speed. Due to space limitations, we refer to [Witkin and Baraff, 1997] for details about rigid body physics. With this model, collisions with another rigid object at a given point of contact can be simulated using the same type of physical abstraction, namely computing the impulse, the resulting forces, and ultimately the influence on the robot's state vector. At the same time, this model describes how the point of contact between the robot and the obstacle changes over time and therefore defines the transition model for failure parameters of Equation 3. From our experience, this physical model achieves the best balance between accuracy and efficiency. Simpler models fail to handle important test cases while more complex models have too many free parameters to be evaluated in real time.

5.1 Quantitative Evaluation of Failure Detection and Tracking Performance

To quantitatively evaluate the usefulness of our approach, we compared it to a particle filter that implements the same process model with the standard uninformed proposals described in Section 3.1. The parameters of the standard filter were optimized for best tracking performance and failure detection rates to ensure comparability. We recorded data by manually steering the robot through the environment and arranged for two collisions, one with boxes of milk and the other one with a box of lemonade bottles. Both obstacles were placed at arbitrary positions and the obstacle heights were too low for the laser sensor to detect them. Figure 3(a) depicts a typical test run, where our system successfully tracked the pose of the robot and detected the two collisions. On the recorded data set, we tested our improved particle filter with Gaussian process proposals (GP-PF) as well as the standard particle filter (Std PF) for different parameter settings. Each filter was executed 50 times for each parameter setting.

Table 1 gives the failure detection performance of the different filters. The detection rate is defined as the number of correctly identified failures relative to the full number. The false positives rate is the amount of false alarms relative to the number of detections. The ground truth collision events were manually entered and a collision was counted as correctly detected, when the marginal failure likelihood exceeded a threshold Θ after a maximum of six filter iterations (0.6 seconds) after the true failure event. The threshold Θ was optimized independently for each filter to allow an unbiased comparison.

	Std PF 150 particles	Std PF 950 p.	GP-PF 50 p.	GP-PF 300 p.
Detection Rate	85 %	90 %	87 %	98 %
False Positives	76 %	72 %	38 %	14 %

Table 1: Detection results with the optimized standard particle filter (Std PF) and our approach that uses Gaussian Process Proposals (GP-PF).

The diagram in Figure 3(b) gives the average deviation of the tracked poses of the robot compared to the ground truth trajectory. The ground truth trajectory was computed using a scan matcher. The results visualized in this diagram show that our system stays around ten centimeters closer to the true trajectory and produces less variance in these estimates than the standard approach. This is mainly due to the fact that the failure parameter (here, the point of contact with the obstacle) is estimated more accurately. To give an impression about the accuracy with which our filter estimates the point of contact and thereby the path of the robot, one failure event is depicted in detail in the diagram of Figure 3(c). It can be seen, that the estimated failure likelihood increases shortly after the labeled failure event and that the heading angle of the robot is correctly estimated.

The detection rates as well as the tracking results show that learned Gaussian process proposals can indeed increase the reliability and efficiency of online state estimation approaches. The time requirements for the improved particle filter are around 10% to 15% higher than for the standard implementation without Gaussian process proposals. Nevertheless, the implemented system with 200 particles still processes one minute of recorded data in less than 23 seconds on a PC with a 2800 MHz CPU.

6 Conclusions

In this paper, we showed that efficient proposal distributions for particle filters can be learned using Gaussian process models and that both discrete as well as continuous state variables can be treated in a consistent manner. We applied the approach to the hard problem of online failure detection on mobile robots and presented a system for detecting unforseen collisions. Experiments with a real robot demonstrated, that the developed system is able to track the state of the robot more reliably through collision events than an optimized version of the standard particle filter with uninformed proposals. Our system does not require any additional hardware and can be trained conveniently using a simulator.

While our current system only deals with binary failure variables, we believe that the multi-class case, potentially including different simultaneous failures, can be formulated and solved using similar models. For a limited set of such discrete failure modes, the combination of this system with the look-ahead particle filter [de Freitas *et al.*, 2003] may be beneficial. Moreover, in future work, we would like to extend our approach to the more general problem domain of learning sampling models for dynamic Bayesian networks.

Acknowledgments

This work has been supported by the EC under FP6-004250-CoSy, by the German Ministry for Education and Research (BMBF) through the DESIRE project, and by the National Science Foundation under NSF CAREER grant IIS-0093406.

References

[Benazera *et al.*, 2004] E. Benazera, R. Dearden, and S. Narasimhan. Combining particle filters and consistency-based approaches. In *15th Int. Workshop on Principles of Diagnosis*, Carcassonne, France, 2004.

[de Freitas *et al.*, 2003] N. de Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole. Diagnosis by a waiter and a mars explorer. In *Proc. of the IEEE*, 2004.

[Dearden and Clancy, 2002] R. Dearden and D. Clancy. Particle filters for real-time fault detection in planetary rovers. In *Proc. of the Thirteenth Int. Workshop on Principles of Diagnosis*, 2002.

[Doucet, 1998] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Dept. of Engeneering, University of Cambridge, 1998.

[Driessen and Boers, 2004] J.N. Driessen and Y. Boers. An efficient particle filter for nonlinear jump markov systems. In *IEEE Sem. Target Tracking: Algorithms and Appl., Sussex, UK*, 2004.

[Girard *et al.*, 2002] A. Girard, C.E. Rasmussen, J. Quiñonero Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In *NIPS*, 2002.

[Khan *et al.*, 2004] Z. Khan, T.R. Balch, and F. Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV (4)*, pages 279–290, 2004.

[Koenig and Howard, 2004] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. Tech. rep., USC Center for Robotics and Emb. Systems, 2004.

[Kwok *et al.*, 2002] C. Kwok, D. Fox, and M. Meila. Real-time particle filters. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 1057–1064, 2002.

[MacKay, 1998] D.J.C. MacKay. Introduction to Gaussian processes. In *Neural Networks and Machine Learning*, 1998.

[Neal, 1997] R. Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification. Technical report, Dept. of Computer Science, University of Toronto., 1997.

[Ng *et al.*, 2005] B. Ng, A. Pfeffer, and R. Dearden. Continuous time particle filtering. In *Proceedings of IJCAI*, 2005.

[Plagemann *et al.*, 2006] C. Plagemann, C. Stachniss, and W. Burgard. Efficient failure detection for mobile robots using mixed-abstraction particle filters. In *Europ. Robotics Symposium 2006*.

[Rasmussen, 1996] C.E. Rasmussen. *Evaluation Of Gaussian Processes And Other Methods For Non-Linear Regression*. PhD thesis, Dept. of Computer Science, Univ. of Toronto, 1996.

[Schwaighofer *et al.*, 2003] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann. Gpps: A gaussian process positioning system for cellular networks. In *NIPS*, 2003.

[Thrun *et al.*, 2001] S. Thrun, J. Langford, and V. Verma. Risk sensitive particle filters. In *NIPS*, 2001.

[Thrun *et al.*, 2005] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[Verma *et al.*, 2003] V. Verma, S. Thrun, and R. Simmons. Variable resolution particle filter. In *Proc of IJCAI*, 2003.

[Witkin and Baraff, 1997] A. Witkin and D. Baraff. An introduction to physically based modeling. In *SIGGRAPH'97 Course Notes*, 1997.