Proprioceptive Motion Modeling for Monte Carlo Localization

Jan Hoffmann

Institut für Informatik
Humboldt-Universität zu Berlin, Germany
jan@aiboteamhumboldt.com

Abstract. This paper explores how robot localization can be improved and made more reactive by using an adaptive motion model based on proprioception. The motion model of mobile robots is commonly assumed to be constant or a function of the robot speed. We extend this model by explicitly modeling possible states of locomotion caused by interactions of the robot with its environment, such as collisions. The motion model thus behaves according to which state the robot is in. State transitions are based on proprioception, which in our case describes how well the robot's limbs are able to follow their respective motor commands. The extended, adaptive motion model yields a better, more reactive model of the current robot belief, which is shown in experiments. The improvement is due to the fact that the motion noise no longer has to subsume any possible outcome of actions including failure. In contrast, a clear distinction between failure and normal, desired operation is possible, which is reflected in the motion model.

1 Introduction

Due to the dynamic nature of the RoboCup environment, collisions with other robots occur frequently. Collisions are particularly bad for the Sony Aibo (or for that matter: any legged robot), as two robots easily get one another's legs entangled, making directed locomotion impossible. On the one hand, this leads to impaired mobility, which the robot needs to recover from as quickly as possible. On the other hand, collisions leave the robot badly localized since its actions did not have the expected result.

Detecting collisions is difficult in the Sony Aibo as it does not have any dedicated touch sensors on its shell. Even with the Aibo's touch sensors in its paws it is difficult to detect if the robot's leg touches the ground. To cope with this lack of dedicated sensory input and to provide a degree of proprioception, the movement of the robot's legs are examined [10,5]. The robot's legs (and to some extent its head) are the parts of the robot that come into contact with other robots or the environment when collisions occur. It was found that not only the overall movement of the robot but the movement of individual limbs is impaired during a collision. Monitoring the deviation of intended motions (control) to actual motion of limbs (servo readings), collisions can be detected [5].

G. Lakemeyer et al. (Eds.): RoboCup 2006, LNAI 4434, pp. 258–269, 2007.

To understand the effect of collisions on localization, one has to consider the motion update of the Bayes filter employed [12]:

$$\text{bel}^{-}(x_t) \longleftarrow \int p(x_t|x_{t-1}, u_{t-1})\text{bel}(x_{t-1})dx_{t-1} \quad (1)$$

$$\text{bel}(x_t) \longleftarrow \eta p(z_t|x_t)\text{bel}^{-}(x_t) \quad (2)$$

In the motion update (Eqn. 2), the (prior) localization belief bel^{-} is updated using knowledge about the action currently performed by the robot, i.e., $p(z_t|x_t)$. For a moving robot, this intended action is given by the motor commands and the expected outcome. This *probabilistic motion model* is commonly obtained by having the robot move about in its environment and measuring the deviation between desired and actual motion. Collisions are usually not included explicitly in the motion model as they are difficult to detect and the effect on the outcome of motions is hard to model. Particle filters offer a certain robustness with respect to errors caused by such un-modeled phenomena by means of random search. In these cases, the motion error is generally assumed to be larger than the actual error observed during calibration to account for unforeseen events. This generally requires a large particle count and reduces the accuracy of the localization: neither does the particle distribution reflect the uncertainty brought about by collisions, nor does it properly reflect the relative certainty when the robot is moving freely.

In contrast, we will show how collision detection can be used to create an adaptive motion model by explicitly modeling different states of robot mobility. This establishes a dynamic level of trust in odometry. The amount and shape of noise associated with locomotion is modeled according to the state of mobility of the robot. The resulting belief representation better resembles the current state of the robot which allows it to recover more quickly from collisions as particles are more spread out. It also allows the robot to monitor the belief entropy and thus determine if it can go on with its task or if it is better to stop and re-localize before proceeding.

The importance of better modeling arises in part from the low particle count used due to limited computational resources. With high particle counts, areas of low probability are still represented by a small number of particles. If something unexpected happens, such particle filters can recover quickly as unlikely areas are not completely deserted. As the particle count is lowered, however, particle filters become more susceptible to errors caused by un-modeled events as unlikely regions of the state space may not be represented at all. Spreading out the particle distribution when disturbances occur helps cope with such events and yield quicker convergence when sensor data is acquired.

Related Work. Legged robots are generally believed to be able to deal with a wider range of environments and surfaces than wheeled robots. The many designs of legged robots vary mainly in the number of legs, ranging from insectoid or arachnoid with 6, 8 or more legs[2],4-legged, such as the Sony Aibo used in our research [4,7], to humanoid with 2 legs (biped) [1,9].

Apart from the current action failing, collisions (and subsequently being stuck) have severe impact on the robot's localization as odometry is used in the localization process [3,11]. As stated before, the Sony Aibo lacks dedicated sensors that would allow the robot to detect touching objects and collisions. This leaves two sensors to achieve proprioception and thus collision detection, the accelerometer and the directional sensors of the robot's servos:

We found the accelerometer data of the Aibo to be very noisy and used it only in situations where low pass filtering could be applied. One such application is to have the robot's head look parallel to the ground. The most important application, though, is to figure out if the robot has fallen over and then trigger a recovery action. In very static, well controlled environment, the accelerometer signal can be also be classified to differentiate between surfaces that the robot is walking on [13].

The Aibo uses servo motors in its leg joints. A servo is an electrical motor with an integrated position feedback device and controller. The position/direction measurements of these servos can be used to achieve proprioception. Quinlan et al. [10] show how this can be used to effectively monitor the traction of the robot, namely by comparing the current servo direction measurements of the leg joints to reference values gathered in a prior calibration run. It relies solely on the direction measurements and does not take into account the control commands. In contrast, our work uses the approach presented by [5], which is based on the assumption that there is, in fact, a similarity between intended and actual motion and the variance of the sensor signal is bounded for the entire period of the motion. Under these assumptions, training can be greatly simplified as the number of parameters required to describe unhindered motion can be reduced by orders of magnitude. The reference value is proportional to the control signal, which is calculated by the walking engine as the inverse kinematic calculation is performed [4].

While improving the sensing model has been the focus of many papers (see bibliographical remarks in [12]), modeling the motion has received little attention and quite crude models prevail in the context of robot localization [11]. In [8], the motion of the ball is modeled in detail, taking into account interactions with its environment using a Bayes net. Our approach takes this idea and applies is to model robot motion, using proprioception as evidence for what state the robot might find itself in.

Outline. We will first summarize the concept of probabilistic motion modeling and describe the motion model used for the Sony Aibo and how this model can be enhanced. We then illustrate the benefits of adaptive motion modeling in several experiments.

2 Probabilistic Motion Model

The motion model is a probabilistic model of the outcome of control action u_t performed by the robot. It is given by the conditional probability density

$$p(s_t|u_t, s_{t-1}). \tag{3}$$

It describes the motion update of the Bayes filter in Equations 1 and 2. There are two types of approaches to modeling the motion of the robot: one is called velocity motion modeling whereas the other is based on odometry [12]. In wheeled robots, an odometer is used to count the number of turns of the wheels as the robot moves. Given the diameter of the wheels, the distance traveled can be calculated. In contrast, the velocity motion model is based on the predicted outcome of control actions. Both velocity- and odometry-based motion modeling suffer from drift and slippage. However, the odometer constantly *measures* the turns of the wheels whereas discrepancies of actual motion and model are not compensated in the velocity motion model. As the Aibo has no wheels and deducing the distance traveled from the leg's movement is very difficult, we use a velocity-based motion model.

Motion Model of the Sony Aibo. Although the Aibo uses legs for locomotion and does not have an odometer, the term odometry is commonly used to describe how far the robot has traveled given a sequence of control command. A control command is also called a *motion request* and consists of desired robot speeds $\boldsymbol{m} = (\dot{x}, \dot{y}, \dot{\theta} = \omega)^{\mathrm{T}}$ with angular velocity ω. Given the motion request, the distance traveled of the robot can be calculated [11,12]. The true distance traveled, however, is never exactly the same in two subsequent runs. Odometry is subject to cumulative errors (drift) and by itself does not account for slipping, sliding, or skidding. We model the odometry error as a normally distributed random variable of finite variance, resulting in the effective speed $\boldsymbol{m}_{\mathrm{eff}}$ and effective locomotion (distance traveled) $\Delta\boldsymbol{r}$:

$$\boldsymbol{m}_{\mathrm{eff}} = \boldsymbol{m} + \boldsymbol{\epsilon}\,(\dot{x}, \dot{y}, \omega) = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \omega \end{pmatrix} + \begin{pmatrix} \epsilon_{\dot{x}}(\dot{x}, \dot{y}, \omega) \\ \epsilon_{\dot{y}}(\dot{x}, \dot{y}, \omega) \\ \epsilon_{\omega}(\dot{x}, \dot{y}, \omega) \end{pmatrix} \tag{4}$$

$$\Delta\boldsymbol{r} = \Delta t \cdot \boldsymbol{m} \tag{5}$$

$$\Delta\boldsymbol{r}_{\mathrm{eff}} = \Delta t \cdot \boldsymbol{m}_{\mathrm{eff}} \tag{6}$$

The true velocity of the robot equals the command velocity plus an additive error ϵ of zero mean. Note that ϵ is a function of all three control inputs, i.e., drift may cause the robot to turn although $\omega = 0$. In a simple model, the noise is assumed gaussian and the standard deviation σ_i of the probability density function (PDF) of ϵ_i is assumed linearly proportional to the weighted sum of the control inputs, requiring a total of 9 parameters to describe the odometry error:

$$\sigma_i(\dot{x}, \dot{y}, \omega) = \alpha_{i,1}\ \dot{x} + \alpha_{i,2}\ \dot{y} + \alpha_{i,3}\ \omega \tag{7}$$

When a particle filter is employed, this means for each particle, an error is sampled from the error PDF associated with ϵ. This error is added to the motion request $\boldsymbol{m} + \boldsymbol{\epsilon} = \boldsymbol{m}'$, which is in turn used to calculate the new robot pose at time t.

In the actual Monte Carlo Localization (MCL) implementation used by the GermanTeam, each particle's pose is updated using the motion request $\boldsymbol{m}$. The odometry error $\Delta t \cdot \boldsymbol{\epsilon}$ is only added after this update. Furthermore, the standard deviation in Equation 7 is approximated in the following fashion:

$$\sigma_i(\dot{x}, \dot{y}, \omega) \approx \sigma_i(|v|, \omega) \tag{8}$$

with

$$|v| = \sqrt{\dot{x}^2 + \dot{y}^2}, \tag{9}$$

assuming that the effect of the translational error is the same in dimensions x and y. The robot pose of a sample is thus updated by:

$$\boldsymbol{r}_t \leftarrow \boldsymbol{r}_{t-1} + \Delta\boldsymbol{r} + \Delta t \begin{pmatrix} \beta_1|v| \ \mathrm{rand}(-1,1) \\ \beta_2|v| \ \mathrm{rand}(-1,1) \\ (\beta_3|v| + \beta_4\omega) \ \mathrm{rand}(-1,1) \end{pmatrix} \tag{10}$$

Where $\beta_1, ..., \beta_4$ are parameters describing the error model and $\mathrm{rand}(-1,1)$ is a function that returns a random number in the range $[-1,1]$ (uniform distribution is used for computational speed). The values of parameters $\beta'_i = \Delta t \cdot \beta_i = 8\mathrm{ms} \cdot \beta_i$ used in the implementation are: $\beta'_1 = 0.1\mathrm{ms}$, $\beta'_2 = 0.02\mathrm{ms}$, $\beta'_3 = 0.002(\mathrm{rad/m})\mathrm{ms}$, $\beta'_4 = 0.2\mathrm{ms}$.

3 Adaptive Motion Model

Some preliminary work on integrating information about collisions into the motion model was presented in [6]. We extend this naive approach by more accurately modeling collisions and slip. The underlying idea is to separate the components of the error brought about by the inherent odometry error $\boldsymbol{\epsilon}_{\mathrm{odo}}$ and the error caused by slippage and collisions $\boldsymbol{\epsilon}_{\mathrm{col}}$. This allows to lower the noise added per step when the robot is thought to move freely (accurate odometry) and to only add large amounts of noise (uncertainty) if collisions are detected. The effective speed $\boldsymbol{m}_{\mathrm{eff}}$ of the robot is thus modeled as:

$$\boldsymbol{m}_{\mathrm{eff}} = \boldsymbol{m} + \boldsymbol{\epsilon}_{\mathrm{odo}} + \boldsymbol{\epsilon}_{\mathrm{col}} \tag{11}$$

The robot pose is thus given as (cf. Equation 10):

$$\boldsymbol{r}_t = \boldsymbol{r}_{t-1} + \Delta\boldsymbol{r} + \boldsymbol{\epsilon}_{\mathrm{odo}}\Delta t + \boldsymbol{\epsilon}_{\mathrm{col}}\Delta t \tag{12}$$

$$= \boldsymbol{r}_{t-1} + \Delta\boldsymbol{r} + \boldsymbol{\epsilon'}_{\mathrm{odo}} + \boldsymbol{\epsilon'}_{\mathrm{col}} \tag{13}$$

This separation improves localization accuracy when the robot moves about freely while at the same time enabling it to more quickly recognize collision events. The development was helped by the advent of means to better calibrate the walking engine [4], making odometry much more precise as long as movement is unhindered.

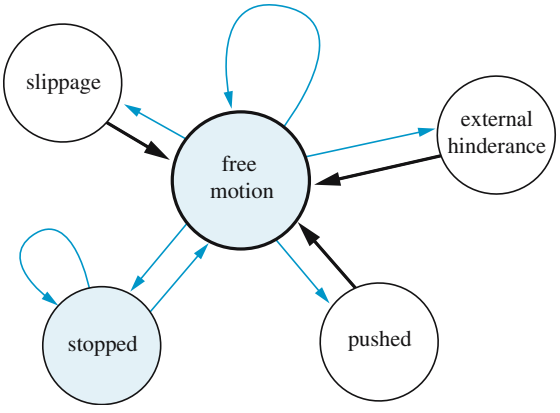

Fig. 1. Bayes net describing the mobility of the robot motion. State transitions happen when collisions occur; transition probabilities not shown.

3.1 Modeling States of Mobility

We will show how the state of mobility of the robot can be modeled. This is done much like the state of the ball is modeled in [8]. The Bayes net in Fig. 1 models the things that can happen to the robot with regards to its ability to move. State transitions away from the "free" state are triggered by collision events. The possible states and their effect on the robot's position and orientation are:

Free motion. The robot moves freely, no internal or external disturbances occur, locomotion error is determined by the odometry error only.

Slippage. This state subsumes motion disturbances like slipping and skidding that occur without external disturbance and that are often caused by abrupt changes of the motion request.

The effective translational speed of the robot is reduced and the robot orientation is subject to error:

$$v' = v \ \mathrm{rand}(\beta_1, 1.0) \tag{14}$$

$$\omega' = \omega + \beta_2 \ \mathrm{rand}(-1.0, +1.0) \tag{15}$$

with $0 \leq \beta_1 < 1$ and angle β_2.

External hindrance. The robot's motion cannot be executed as intended, e.g., because it is running into a wall. The effective motion is smaller than the motion request.

Effective translational speed and angular velocity are reduced:

$$v' = v \ \mathrm{rand}(\beta_3, 1.0) \tag{16}$$

$$\omega' = \omega \ \mathrm{rand}(\beta_4, 1.0) \tag{17}$$

with $0 \leq \beta_3 < 1$ and $0 \leq \beta_4 < 1$.

Pushed. The robot is being pushed by another robot; a force acts upon it resulting in the robot being turned and displaced.

$$\alpha = \text{rand}(-\pi, +\pi) \tag{18}$$
$$d = \beta_5 \ \text{rand}(0, 1) \tag{19}$$
$$x' = x + d \ \cos(\alpha) \tag{20}$$
$$y' = y + d \ \sin(\alpha) \tag{21}$$
$$\theta' = \theta + \beta_6 \ \text{rand}(-1.0, +1.0) \tag{22}$$

with displacement error β_5 and angle β_6.

Stopped. The robot runs into an obstacle and is stopped dead in its track. The translational speed of the robot becomes zero. However, it is often observed that robots turns when being stuck:

$$v' = 0 \tag{23}$$
$$\omega' = \beta_7 \ \text{rand}(-1.0, +1.0) \tag{24}$$

with angle β_7.

Variables used: translational speed of the robot $|v| = \sqrt{\dot{x}^2 + \dot{y}^2}$, robot orientation θ, angular velocity $\omega = \dot{\theta}$, model parameters β_i, and $\text{rand}(a, b)$ is a function that returns a random value in the interval $[a, b]$.

When the robot is in the "stopped" state, it remains in this state for some time until it has freed itself by a recovery action or the entanglement with another robot has somehow been resolved. Characteristically, when the robot is stopped, it will continuously bump into the obstacle, unable to free itself for a few moments. Only when it has freed itself will it no longer detect (frequent) collisions. It therefore remains in the stopped state as long as collisions are detected at high frequency.

When collisions occur, we observed that the robot tends to turn towards the cause of the collision, usually making matters worse. The motion of the robot's legs on the side where the collision occurs is hindered, slowing them down; this results in a difference in the forward component of the motion of the left and right legs, causing the robot to turn like a differential drive vehicle would. The maximum turning speed caused by the difference in speeds is given by $\omega = v_{\text{left}}/w$ (w = lateral distance between the robot's feet). The collision percept z_c contains information about the location of a collision. The angular velocity of the robot thus is changed in Equation 24 in the following fashion:

$$\omega' = \frac{v}{w} \ c(z_c) \ \beta_i \ \text{rand}(0, +1.0) \tag{25}$$

with:

$$c(z_c) = \begin{cases} +1 & \text{if collision left} \\ -1 & \text{if collision right} \\ 0 & \text{if collision left and right} \end{cases} \tag{26}$$

This enables us to model a robot running into a wall, turning towards it until finally facing it. When facing it, collisions will be detected both left and right, marking the end of the turning motion.

4 Experimental Results

In the following experiments, a robot moves forward at constant speed and experiences one or more collisions. The belief represented by the particle distribution is generated without external perception, i.e., it is solely based on odometry, odometry error, and proprioception-based collision error. This is done to emphasize the effect of the proposed motion model. In an actual application where vision percepts are constantly integrated into the robot's belief, the adaptive motion model makes sure that the particle distribution is spread out enough to model the belief's uncertainty and to allow quicker re-localization after collision events.

The parameters used in our experiments were hand-tuned; automating the process of tuning poses a challenging machine learning task and remains future work.

4.1 No Collision

Fig. 2 a) shows the particle distributions of a robot moving forward without experiencing any collisions. Snapshots of the particle distribution are shown in two second intervals. The robot moves at a speed of 200mm/s and it thus takes the robot ten seconds to cross the distance of two meters as indicated. In this illustration and in the following ones, the particle distributions are made up of 100 particles. The initial particle distribution has a standard deviation of zero, i.e., all particles represent the same robot pose $\boldsymbol{r}_0 = (x_0, y_0, \theta_0)$.

The figure contrasts the particle distribution using the standard GermanTeam motion model and the distribution using the adaptive motion model. The latter adapts to the situation of moving freely by using a lower odometry error resulting in a more confined, low entropy distribution. Since the standard motion model is a function only of the robot speed, a trade off between accuracy and robustness is made: the particle distribution needs to spread out as collisions and hindrances may occur but at the same time the noise added needs to be limited for the distribution to remain stable and to not diverge too much. The standard model therefore has a higher entropy than the adaptive motion model.

4.2 Single Brief Collision

In the first collision experiment, illustrated in Fig. 2 b), a robot moves forward at $|v| = 200$mm/s, runs into another robot for about 2s and then continues to walk forward. The particle distribution of the robot integrating collision information is of typical sickle shape caused by angular disturbances; it is also more spread out after the collision and accounts for the various potential turning motions

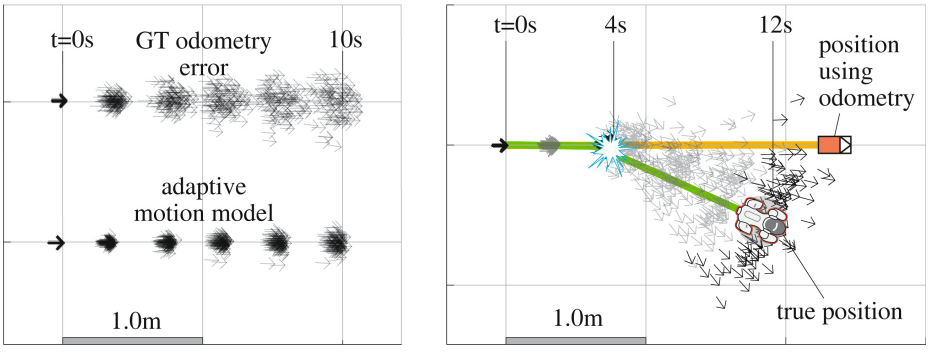

Fig. 2. a) Particle distribution representing a robot moving forward at constant speed experiencing no collisions. The *top* distribution uses the basic motion model used by the GermanTeam, the *bottom* distribution uses the adaptive motion model. Since the adaptive model "knows" that the robot is moving freely, the error of dead reckoning is smaller.
b) Robot running into an obstacle and turning in the process; it then continues to move freely. Note how the particle distribution has become much more spread out compared to the unhindered movement.

experienced by the robot. Information about the side on which the collision occurred is integrated and results in a non-symmetric PDF. Not taking into account collisions and only using odometry, the robot believes to have traveled in a straight line and also to have traveled further than it actually has.

As intended, the uncertainty associated with the belief increases after the collision. Fig. 3 (bottom) shows the relative "sum of squared deviations"[5] during the run, indicating when collisions are detected. Entropy is used to describe the uncertainty associated with the particle distribution [12]:

$$H_p(s) = -\sum p(s_i) \log_2 p(s_i) \tag{27}$$

The entropy is divided into orientation and position entropy. As an artifact of the way that entropy is calculated (grid based), the values of $H_p(s)$ start at zero and remain zero in early stages of each run. This is because at the very beginning, all particles fall within the center cell, resulting in $H = 1 \log_2 1 = 0$.

Before the collision, the entropy of the adaptive motion model is lower than that of the standard motion model. When the collision occurs, the amount of motion noise is increased, which can be noticed in both the position and the orientation entropy. After the collision, the position entropy of the adaptive model continues to rise caused by particles diverging due to the angular uncertainty in conjunction with the robot moving forward.

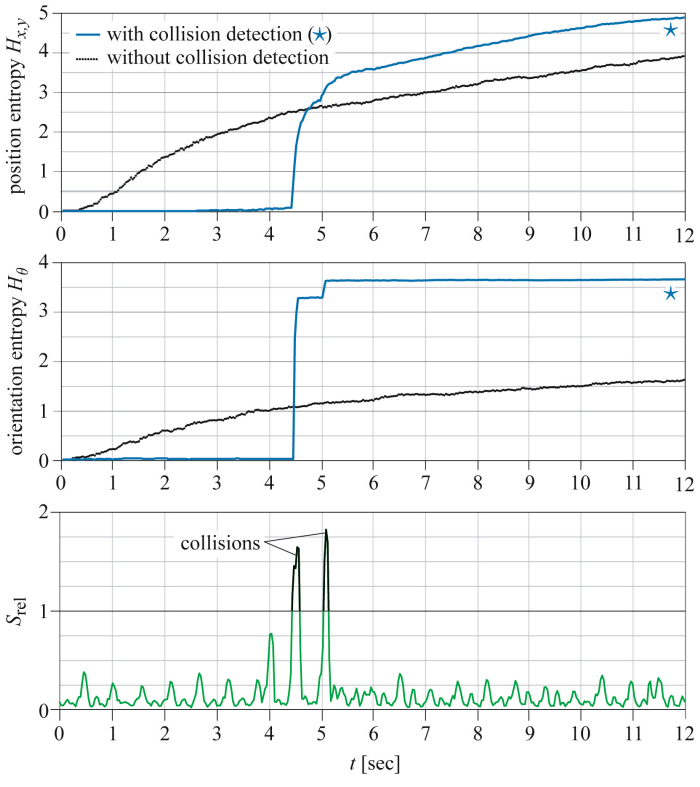

Fig. 3. Impact of collision on the entropy of the particle distribution. Values of S_{rel} greater than 1 are interpreted as collisions. The marked curves (*) represent the entropy of the particle distribution using the adaptive motion model. Without using collision information, the entropy continuously rises over time. Incorporating collision information, the entropy is lower *before* the collision and increases drastically as collisions are detected.

4.3 Two Subsequent Collisions

In this experiment, the robot moves forward but experiences two collisions in brief succession, one on its left and one on its right (Fig. 4). The two collisions somewhat compensate each other in terms of resulting robot orientation. However, the robot is slowed down by the collisions and the total distance traveled is reduced compared to the robot moving freely.

The particle distribution is spread out quite a bit, accounting for the two collisions and their probable outcomes. The impact on the particle distribution of the second collision is not as prominent as the distribution is already quite disturbed. Note that the distribution already is a little patchy in some areas, i.e., not all areas of the PDF are equally well described by the particle distribution due to the small number of particles employed.

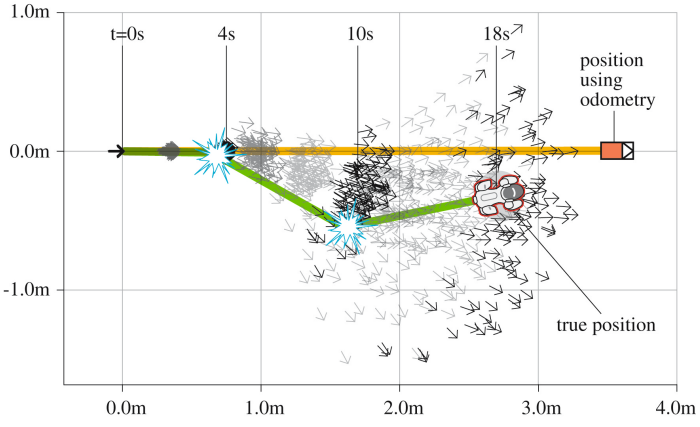

Fig. 4. Robot experiencing two collisions in brief succession. Note that the distance the robot has actually traveled is much shorter than the distance calculated using only odometry.

5 Conclusion

We were able to show how motion of a mobile robot can be modeled more closely to improve Monte Carlo Localization (MCL). The basic velocity-based motion model of a legged robot is enhanced by modeling various states describing the robot's ability to move and perform actions. As evidence for state transitions, proprioception is used, e.g., to detect if the robot has run into an obstacle. Whereas the simple velocity-based motion model tries to model all these states in a single PDF, the adaptive model offers a much more accurate description of robot motion and thus offers several advantages over the simple model: a) the particle count used in MCL can be lowered as the particle distribution better describes the belief PDF; b) when the robot is moving freely, the particle distribution remains relatively confined as it does not need to spread out to accommodate for potential action failures; c) when the robot does run into something, the error caused by this is more quickly reflected in the particle distribution.

The resulting particle distribution describing the robot's belief is more reactive and more accurately describes the situation that the robot is in. This has interesting applications to robot control and active vision when the current belief uncertainty is taken into account for control.

Acknowledgments

Work was funded by the German Research Foundation, Priority Project 1225. The software framework was developed by the GermanTeam; source code and documentation available for download at http://www.germanteam.org.

References

1. Behnke, S., Müller, J., Schreiber, M.: Toni: A soccer playing humanoid robot. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, Springer, Heidelberg (2006)
2. Clark, J.E., Cham, J.G., Bailey, S.A., Froehlich, E.M., Nahata, P.K., Full, R.J., Cutkosky, M.R.: Biomimetic design and fabrication of a hexapedal running robot. In: Proc. of the 2001 Intl. Conference Robotics and Automation (ICRA), IEEE Computer Society Press, Los Alamitos (2001)
3. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: Proc. of the 1999 IEEE Intl. Conference on Robotics and Automation (ICRA), vol. 2, IEEE Computer Society Press, Los Alamitos (1999)
4. Düffert, U., Hoffmann, J.: Reliable and precise gait modeling for a quadruped robot. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, Springer, Heidelberg (2006)
5. Hoffmann, J., Göhring, D.: Sensor-actuator-comparison as a basis for collision detection for a quadruped robot. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, Springer, Heidelberg (2005)
6. Hoffmann, J., Spranger, M., Göhring, D., Jüngel, M.: Exploiting the unexpected: Negative evidence modeling and proprioceptive motion modeling for improved markov localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, Springer, Heidelberg (2006)
7. Hornby, G., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., Fujita, M.: Evolving robust gaits with Aibo. In: Proc. of the 2000 Intl. Conference on Robotics and Automation (ICRA), IEEE Computer Society Press, Los Alamitos (2000)
8. Kwok, C., Fox, D.: Map-based multiple model tracking of a moving object. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, Springer, Heidelberg (2005)
9. Dario, C.L.P., Guglielmelli, E.: Humanoids and personal robots: design and experiments. Journal of Robotic Systems 18(2) (2001)
10. Quinlan, M.J., Murch, C.L., Middleton, R.H., Chalup, S.K.: Traction monitoring for collision detection with legged robots. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, Springer, Heidelberg (2004)
11. Röfer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Jüngel, M., Laue, T., Lötzsch, M., Nistico, W., Spranger, M.: GermanTeam 2004: The German national RoboCup team. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, Springer, Heidelberg (2005)
12. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge, MA, USA (2005)
13. Vail, D., Veloso, M.: Learning from accelerometer data on a legged robot. In: Proc. of the 5th IFAC Symp. on Intelligent Autonomous Vehicles (IAV-2004) (2004)