Auton Robot (2018) 42:601–614
https://doi.org/10.1007/s10514-017-9653-x

A representation method based on the probability of collision for safe robot navigation in domestic environments

Janno Lunenburg[1] · **René van de Molengraft**[1] · **Maarten Steinbuch**[1]

Received: 5 January 2015 / Accepted: 24 June 2017 / Published online: 12 August 2017

Abstract This paper introduces a three-dimensional volumetric representation for safe navigation. It is based on the OctoMap representation framework that probabilistically fuses sensor measurements to represent the occupancy probability of volumes. To achieve safe navigation in a domestic environment this representation is extended with a model of the occupancy probability if no sensor measurements are received, and a proactive approach to deal with unpredictably moving obstacles that can arise from behind occlusions by always expecting obstacles to appear on the robot's path. By combining the occupancy probability of volumes with the position uncertainty of the robot, a probability of collision is obtained. It is shown that by relating this probability to a safe velocity limit a robot in a real domestic environment can move close to a certain maximum velocity but decides to attain a slower safe velocity limit when it must, analogous to slowing down in traffic when approaching an occluded intersection.

Keywords Navigation · Representation · Collision avoidance · Mobile robots

✉ Janno Lunenburg
jannolunenburg@gmail.com

René van de Molengraft
m.j.g.v.d.molengraft@tue.nl

Maarten Steinbuch
m.steinbuch@tue.nl

[1] Department of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

1 Introduction

Robots that navigate in indoor, domestic environments face an environment that encompasses obstacles and uncertainties. Obstacles generally vary in their size and shape and can be static as well as dynamic. Uncertainty typically arises from three sources (Berg et al. 2011): (i) sensing uncertainty due to noisy and false sensor measurements, (ii) uncertainty about the environment due to (partially) unknown parts of that environment and (iii) robot position uncertainty due to localization errors and external disturbances acting on the robot. In the presence of these characteristics a robot always needs to guarantee that it is safe, i.e., that it can ensure to come to a timely stop when a collision is imminent. Safe navigation can be achieved by moving a robot at very low velocities, typically below 0.1 m/s. However, by incorporating knowledge on the environment a robot can move with higher velocities without becoming unsafe.

A navigation system generally consists of a representation of the environment, one or more algorithms that search for a path or trajectory through this environment and generate motor commands and a policy that coordinates these algorithms. A three-dimensional representation of the environment is a prerequisite to deal with the challenges that a typical domestic environment poses (Marder-Eppstein et al. 2010). An approximate cell decomposition is a common and popular approach to represent the environment (Goerzen et al. 2010; Hornung et al. 2013). Searching and executing a path in an environment with obstacles and uncertainties is typically achieved using a planner that finds a path to the goal that is executed by a reactive algorithm (Goerzen et al. 2010). Such a reactive algorithm controls the robot in real-time to avoid imminent collisions by stopping or swerving the robot when an obstacle is known to be on the robot's path.

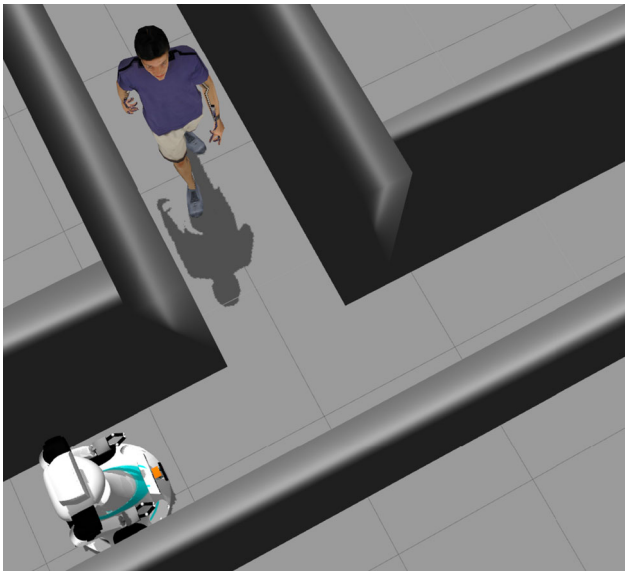


Fig. 1 A corridor with a doorway that is occluded. In a domestic environment this increases the probability of collision for a robot as moving obstacles such as people can emerge from behind such occlusions

Space that is unknown, due to occlusions and a limited sensory range, poses a threat to a robot that navigates in a domestic environment as moving obstacles might emerge from behind occlusions onto the robot's path. An exemplary situation occurs when a robot traverses a corridor as depicted in Fig. 1. Near unknown space, such as a passage or a doorway that is occluded to the sensors, an obstacle can suddenly emerge. This requires the robot to lower its velocity such that it can guarantee that it will not collide with a possible incoming obstacle. The robot can increase its velocity when a confined part of the environment, like the part of the corridor past the doorway, is known to be free as there is no uncertainty arising from any unknown space. A reactive algorithm that has no information about this threat can approach at a velocity that does not allow the robot to detect the imminent collision and react to it to avoid collision. Hence, analogous to slowing down in hazardous situations in traffic, a representation is necessary that allows the robot to decide when it is able to move at a certain maximum velocity and when it must maintain a slower pace to ensure safety.

2 Related work and contribution

To achieve safe behavior some approaches modify the robot's planned path or motion using the obstacle representation. They enlarge obstacles by inflating their representation with a 'safe' distance that encompasses the robot position uncertainty as well as the robot sensing uncertainty (Hsu et al. 2002; Chung et al. 2004). While this may ensure that a robot keeps a greater distance from obstacles, it may also prohibit a robot to traverse tight spaces. This is undesirable as tight spaces such as doorways are typically present in a domestic environment. Other approaches limit the robot's velocity based on distance to obstacles on the robot's path (Lingemann et al. 2005) or the amount of clearance on both sides of the robot (Fox et al. 1997). The former approach will fail when a moving obstacle emerges from occluded regions, e.g., a doorway in a corridor as shown in Fig. 1. The latter approach is conservative in that it will scale down its velocity in a narrow corridor, while in such a confined part of the environment the robot could safely drive at higher velocities.

Uncertainties can explicitly be taken into account to prevent collisions. A range of approaches estimate the probability of collision of the robot along its path (Missiuro and Roy 2006; Burns and Brock 2007; Guibas et al. 2009; Berg et al. 2011; Patil et al. 2012). These approaches plan a path by sampling the environment for feasible robot configurations that reduce the probability of collision during the execution of a path. These approaches result in robot motions with a lower probability of collision by explicitly considering the robot position and sensing uncertainty. However, these approaches do not take into account the uncertainty that arises due to unknown parts of the environment. In Marder-Eppstein et al. (2010), a method is introduced that does track the unknown space in three dimensions. To guarantee safe behavior it is ensured that the robot never traverses this unknown space. However, it is assumed that the environment is mostly static, i.e., it can not ensure safe behavior if an obstacle emerges from an occluded part of the environment. Furthermore, once any unknown space is marked as free it will remain free. This makes the representation over-confident in its assumption that space is free as this space can become occupied again in a changing environment. Hence, a time-dependent occupancy probability model is necessary, instead of merely tracking unknown space.

A more formal approach to assess safety can be found in literature discussing inevitable collision states, e.g., in Fraichard and Asama (2003), Bautin et al. (2010), Althoff et al. (2010), Bouraine et al. (2012) and Althoff and Dolan (2014). However, these methods assume that a model of the future is available but do not consider unmodeled obstacles occurring from behind occlusions. Other methods to assess safety and threats can be found in, e.g., Eidehall and Petersson (2008) and Althoff and Mergel (2011), but these require position information of other objects as well.

Finally, some approaches explicitly model the uncertainty that arises due to unpredictable moving obstacles, e.g., humans (Philippsen et al. 2006, 2008; Rohrmüller et al. 2008). Moving obstacles are extracted from subsequent sensor readings and their position and velocity is estimated to obtain a probabilistic model that resembles the risk of collision. A moving obstacle can, however, be occluded up to an imminent collision. Furthermore, these methods limit

themselves to a two-dimensional representation of the environment and are therefore not suitable for application in a domestic environment.

Hence, current approaches that deal with uncertainties to allow safe navigation are not fully suited for a domestic environment. Not all present uncertainties are considered in an integrated approach and they lack an explicit model to deal with the uncertainty that arises due to occlusions in an environment with unpredictably moving obstacles.

This paper contributes a three-dimensional representation that allows a motion planner to decide when it can move at a certain maximum velocity and when it must maintain a slower pace to ensure safety based on the probability of collision. This is achieved by explicitly representing the multiple uncertainties that are present in a domestic environment and using the probability of collision to determine a safe velocity limit, analogous to slowing down in hazardous situations in traffic.

3 Environment representation

In this section the proposed environment representation for safe navigation is elaborated. First, it will be described how this representation, based on the *OctoMap* framework (Hornung et al. 2013) and first introduced in Coenen et al. (2014), uses probabilistic fusion of sensor measurements to be robust against uncertainty in sensing. Secondly, it is described how uncertainty due to unknown space is represented if no measurements are received and how the probability of obstacles appearing on the robot's path from behind occlusions is represented using a proactive approach (Alami et al. 2002). These first two parts results in the representation of the probability of volumes being occupied by an obstacle. Next, the probability of the robot occupying a volume is represented using a model of the robot position uncertainty. Finally, the probability occupancy of obstacles and the probability occupancy of volumes by the robot are combined to represent a probability of collision. This probability is then related to a safe velocity limit that the robot must attain in order to guarantee safety.

3.1 Robot sensing uncertainty representation

The *OctoMap* framework provides a volumetric octree-based representation of the environment (Hornung et al. 2013). It models the environment as free, occupied and unknown cubic volumes or so-called voxels. Sensor measurements are integrated probabilistically using occupancy grid mapping (Moravec 1988). This technique allows a probabilistic fusion of multiple sensor measurements making it robust to noisy and false sensor measurements. Also, it allows the integration of measurements from multiple, different sensors.

The occupancy of a voxel is updated as measurements are received. Each voxel n, with a resolution r, has a probability $P(n)$ of being occupied. The occupancy probability of all volumes is typically initialized to unknown, i.e., the uniform prior $P(n) = 0.5$. Then, $P(n)$ is updated based on a sensor specific model as measurements $z_{1:k}$ up to time step k are received. The update rule for the estimated voxel occupancy, using the log-odds (L) notation (Moravec 1988), is

$$L(n \mid z_{1:k}) = L(n \mid z_{1:k-1}) + L(n \mid z_k), \tag{1}$$

with

$$L(n) = \log\left[\frac{P(n)}{1 - P(n)}\right], \tag{2}$$

where $L(n \mid z_{1:k})$ is the estimated log-odds probability of a voxel given measurements $z_{1:k}$, $L(n \mid z_{1:k-1})$ is the previously estimated log-odds probability and $L(n \mid z_k)$ denotes the log-odds probability of voxel n being occupied given the measurement z_k. The update of occupancy probability is typically performed in log-odds as using additions is faster than the multiplications that are necessary when (1) is expressed in $P(n)$.

$L(n \mid z_k)$ relies on a sensor model that relates the sensor measurements to the occupancy probability of a voxel:

$$L(n \mid z_k) = \begin{cases} l_{\text{free}}, & \text{if } n \text{ is marked as free} \\ l_{\text{occ}}, & \text{if } n \text{ is marked as occupied} \end{cases} \tag{3}$$

Given equally likely measurements ($l_{\text{free}} = l_{\text{occ}}$), a voxel that is marked as free k times needs to be marked as occupied equally many k times before its occupancy probability is equal again. This makes the representation unable to change as quickly as the environment. As discussed in Hornung et al. (2013), the representation can be made adaptable to a changing environment by limiting the probability in the update rule in (1) to a lower and upper bound on the log-odds value, respectively $l_{\min}$ and $l_{\max}$ or $p_{\min}$ and $p_{\max}$ on the probability. For more details on the update formula and its background the reader is referred to Moravec (1988) and Hornung et al. (2013).

3.2 Environment uncertainty representation

The update rule introduced in (1) models the occupancy probability of voxels under the assumption that they receive measurements. However, large parts of the environment typically yield no measurements as they are occluded to the robot's sensors. It is important that the update rule in (1) also describes the occupancy probability of voxels if no measurements are received. For example, consider a robot that

moves in an environment such that all voxels n in the representation are receiving measurements, either marking voxels as free or occupied. Then, given a static environment, the occupancy of the voxels will approach the threshold, i.e., either $P(n) = p_{\min}$ or $P(n) = p_{\max}$. However, it is incorrect to assume that the occupancy of those voxels that do not receive measurements does not change if the environment is dynamic. In other words, such a representation is over-confident in its assumption that space is either free or occupied. Hence, the occupancy of a voxel is more realistically modeled to become unknown again as no measurements are received for some time.

A time-dependent occupancy probability model is added to the representation to deal with this environment characteristic. Instead of only updating a voxel n if a measurement is received, it is updated at every time step k that the environment is updated. Thereto, the sensor model, introduced in (3), can be extended with the occupancy probability update rule

$$l_{\text{dec}} = (k - k^n_{z,\text{last}})\Delta_{\text{dec}} \quad \text{if } n \text{ is } \textbf{not} \text{ marked}, \tag{4}$$

where $k^n_{z,\text{last}}$ is the time step at which a voxel n received its last measurement update and Δ_{dec} indicates the rate of decay of the probability of a voxel n in l_{dec}/k. The value of the log-odds value l_{dec} depends on the occupancy of a voxel according to:

$$l_{\text{dec}} = \begin{cases} +l_{\text{dec}}, & \text{if } l < 0, \text{ i.e., } P(n) < 0.5 \\ 0, & \text{if } l > 0, \text{ i.e., } P(n) > 0.5 \end{cases} \tag{5}$$

Hence, if a voxel receives no measurements and $P(n) > 0.5$ it gradually turns to unknown again, i.e., $P(n) = 0.5$. This is visualized for a voxel that is at the lower probability threshold $p_{\min}$ in Fig. 2. The update rule in (5) does not let the occupancy probability of an occupied voxel decrease from $P(n) > 0.5$ to unknown as no measurements are received,

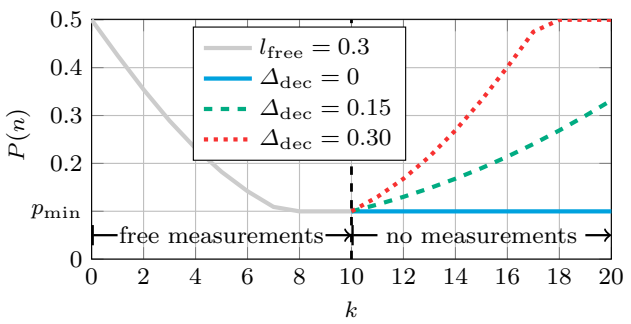


Fig. 2 The occupancy probability of a voxel $P(n)$ decreasing with $l_{\text{free}} = -0.3$ according to (3) as it is marked as free at the first ten time steps and increasing with l_{dec} according to (5) for various Δ_{dec} as it is not marked for ten following time steps

since this would require more knowledge of obstacles to discriminate between them. For example, a voxel that belongs to a static obstacle, e.g., a wall, is more likely to remain occupied than a voxel that belongs to a moving obstacle, e.g., a human. The update rule in (5) is based on a model that is linear in log-odds, similar to the sensor measurement update model in (1). Therefore, the rate of decay Δ_{dec} can intuitively be chosen as a rate at which measurements are discarded again, such that the probability returns to unknown. For example, a measurement that decreases the probability of occupancy is discarded at the next step the environment is updated and no measurement is received by choosing $\Delta_{\text{dec}} = l_{\text{free}}$. Hence, it takes an equal amount of time steps to increase from $P(n) = p_{\min}$ to $P(n) = 0.5$ if no measurements are received as it took to get from $P(n) = 0.5$ to $P(n) = p_{\min}$ when this voxel was marked as free.

The representation of occupancy probability is three-dimensional. However, to keep the computational complexity of the representation tractable the three-dimensional grid of voxels is projected down to a two-dimensional representation. Therefore, each column of voxels in the occupancy map is projected down to a grid cell with occupancy probability p according to

$$P_{\mathbf{c}} = \max_i P(n_i), \tag{6}$$

where $\mathbf{c}$ is a grid cell with resolution r. Taking the maximum occupancy probability is a conservative strategy that is necessary for safe navigation as it ensures that the robot never underestimates the possibility that a voxel is occupied at any height in a column. Other possibilities would be summing the probabilities or taking the average. However, summing the probabilities of each voxel would make the robot too conservative: if $P_i = p_{\min} \forall i$, the cell would probably be free but the robot would still consider it unsafe. The cell probability $p_{\mathbf{c}}$ could even exceed 1. Averaging the probabilities of all voxels in a column, on the other hand, would lead to an overconfident robot: if only a single voxel would be occupied with probability $P = p_{\max}$ and the remaining voxels of the column would be free, i.e., $P = p_{\min}$, taking the average would indicate that the cell is more likely to be free than to be occupied, while the occupied voxel actually indicates that it is probably not safe to go there.

As mentioned, in a domestic environment an obstacle can move on the robot's path from behind an occlusion, as illustrated in Fig. 1. The occupancy probability model that has been introduced so far must be extended with a velocity model to deal with this. This model is based on a proactive approach that is introduced in Alami et al. (2002). By proactive it is meant that the robot is always expecting that a moving obstacle can appear on the robot's path from an occluded region. This is achieved by inflating the occupancy

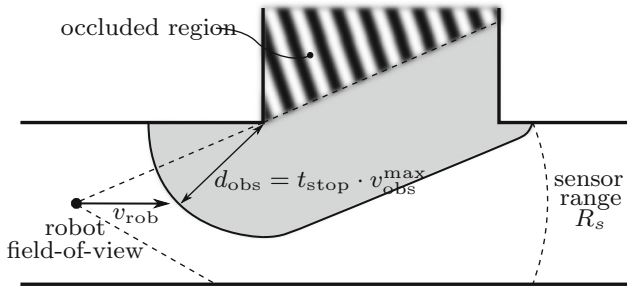


Fig. 3 An obstacle can appear on the robot's path out of a region that is occluded to the robot. To represent this possibility the uncertain region is inflated with the maximum distance the obstacle can travel within the time the robot needs to come to a stop

probability of cells in the two-dimensional projected map that are not marked as an obstacle, as depicted in Fig. 3. The inflation distance d_{obs} depends on the distance that an obstacle can travel within the time the robot needs to come to a stop, $t_{\text{stop}} = v_{\text{rob}}/a_m + t_d$, where v_{rob} is the robot's velocity, $-a_m$ is the maximum deceleration and t_d is the maximum update delay that is present before an incoming obstacle is actually detected. The distance an obstacle can travel with a maximum velocity $v_{\text{obs}}^{\max}$ is now $d_{\text{obs}} = t_{\text{stop}} v_{\text{obs}}^{\max}$, which is conservative as it assumes that an obstacle will maintain its maximum velocity. However, this assumption is necessary as any obstacle trajectory is assumed to be unknown.

The cells of the grid map with an occupancy probability are inflated with the obstacle inflation distance d_{obs}. A grid map cell can be affected by the inflation of multiple cells. Hence, these probabilities must be merged and this is done by taking the maximum probability of all inflated probabilities. In practice this means that all inflated cells are regarded as one potential source of collision risk. By guaranteeing that the robot is safe in this situation it can also be guaranteed that it is so in the case of multiple sources of collision risk.

3.3 Robot position uncertainty representation

The uncertainty that is present in the robot's position is due to errors in the estimation of the robot's position relative to a map of its environment. As is mentioned in, e.g., Choset et al. (2005), if the initial pose of the robot is (approximately) known and a localization algorithm solely needs to track the position of the robot while moving through the environment, the position uncertainty can be modeled as a bivariate normal (Gaussian) distribution

$$\mathbf{x} \sim \mathcal{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{7}$$

with probability density function $f(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the mean $\boldsymbol{\mu}$ is the robot position at $\mathbf{x} = (x; y)$ and $\boldsymbol{\Sigma}$ is the two-dimensional covariance matrix. For simplicity, it is hereby assumed that robot has a circularly shaped platform. As a result, obstacles can be inflated with the robot radius and the robot can be considered as a point in the *configuration space*. The distribution of the robot's position is considered within a prediction interval to limit computation. The occupancy probability of a cell $\mathbf{c}$ by the robot $P(\mathbf{c}, robot)$ is approximated by multiplying the value of the probability density function at the center of this cell $f(\mathbf{x_c}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ by r^2/s. Here, r^2 denotes the surface of cell $\mathbf{c}$ and s is the factor that normalizes the probabilities within the prediction interval $s = \Sigma_{\mathbf{c}} P(\mathbf{c}, robot)$. The resulting discrete probability distribution is denoted by $\mathcal{N}_d(\mathbf{c}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Given a probability of $1 - \alpha$ the robot's position is guaranteed to be in a certain region $\mathcal{A}$ centered around the mean $\boldsymbol{\mu}$. The region $\mathcal{A}$ is an ellipsoid with a 'radius' k, given by the relation $(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = k^2$ that can be deduced from (7). Given α, the ellipsoid shaped region $\mathcal{A}$ follows from $k = \sqrt{\chi_2^2(\alpha)}$, where $\chi_2^2 \alpha$ is the upper (100α) percentile from the two-dimensional chi-squared distribution. For example, for $\boldsymbol{\Sigma} = [0.1\,0; 0\,0.1]$ and $\alpha = 0.05$, the robot's position is located with a probability of $p = 0.95$ in a circle with a radius $k = \sqrt{0.1 \cdot 5.99}$.

3.4 Combined representation

The environment uncertainty model in Sect. 3.2 and the robot position model in Sect. 3.3 are combined to obtain a probability of collision. The probability of a cell being occupied by an obstacle and the probability of that same cell being occupied by the robot are assumed to be independent. Hence, the probability of collision, i.e., a cell is occupied by both the robot and an obstacle, is equal to the product of both probabilities. Now, the probability of collision at a specific cell can be determined according to

$$P(\text{collision}) = \sum_{\mathbf{c} \in \mathcal{A}} P_{\mathbf{c}} \cdot \mathcal{N}_d(\mathbf{c}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{8}$$

This probability of collision must be related to a velocity limit v_{safe} that ensures safe navigation. Thereto, the maximum velocity must be lowered as the probability of collision increases, analogous to driving in traffic where the road becomes dangerous. The maximum velocity can be derived given that $d_{\text{rob}} + d_{\text{obs}} < R_s$ must always hold to avoid collision. Here, R_s is the limited sensory range and $d_{\text{rob}} = \frac{1}{2} a_m (v_{\text{rob}}/a_m)^2 + v_{\text{rob}} t_d$ is the maximum distance that the robot travels to come to a stop. Given $d_{\text{obs}} = t_{\text{stop}} v_{\text{obs}}^{\max}$, the maximum robot velocity is

$$v_{\text{rob}}^{\max} = -v_{\text{obs}}^{\max} - a_m t_d + \sqrt{a_m^2 t_d^2 + (v_{\text{obs}}^{\max})^2 + 2 a_m R_s}. \tag{9}$$

One could argue that to be strictly safe the robot must have a zero velocity in the presence of a nonzero probability of collision. However, the robot must accept some risk of collision

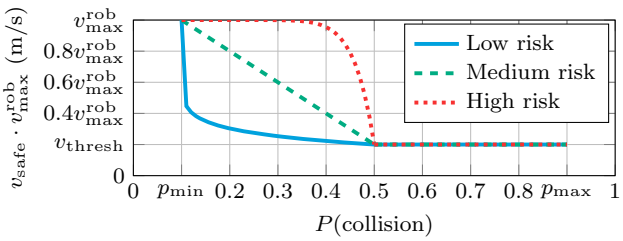


Fig. 4 The safe velocity limit as a n-degree polynomial function of the robot's maximum velocity with $n = 0.1$ (low risk), $n = 1.0$ (medium risk) and $n = 10.0$ (high risk). A threshold velocity, $v_{\text{thresh}} = 0.2v_{\max}^{\text{rob}}$, allows the robot to move at a limited velocity in the presence of any risk of collision. The robot can be allowed to move faster by taking more risk

during navigation as it is generally not absolutely certain that space is free. This is an inherent consequence of the update rule in (1) and the clamping threshold $p_{\min}$ on the occupancy probability. Hence, a threshold velocity v_{thresh} is used, that allows to robot to drive at a velocity close zero in the presence of any uncertainty. The choice for v_{safe} is a trade-off between moving safely at v_{thresh} or moving faster with more risk of collision. This trade-off is visualized in Fig. 4 for a polynomial function with different degrees. Of course, the exact function to relate the probability of collision to a safe velocity limit is a design choice that depends on the consequences of collision in a particular scenario.

4 Implementation

4.1 Global and local planner

The environment representation, introduced in Sect. 3, is implemented in our existing motion planning approach that has been successfully used in the RoboCup@Home league over the past years (Lunenburg et al. 2013, 2014). This approach consists of a global planner that searches for a plan to the goal and a local planner that computes a velocity such that this global path is followed.

The global planner uses an A* algorithm that searches for a cost-optimal path. The costs are encoded in a costmap that represents the distance to obstacles. A plan is computed at a fixed frequency of 2 Hz. If the current plan is free, a re-plan is executed if the estimated time for the new plan is significantly less. This way the robot does not hold on to an old plan when a shorter path has become available but switching between two paths of approximately similar costs is avoided. If the current plan is blocked and an alternative plan is significantly longer, the robot will wait before executing this. This ensures that if a path is blocked for only a short time (e.g., by a moving obstacle), an unnecessarily long re-plan is not executed. If the new plan is of approximately equal length to the original plan it will be executed immediately to avoid unnecessarily long waiting.

The local planner computes an omni-directional velocity (v_x, v_y, v_θ) in the direction of the next pose of the global path that maximizes velocity while obeying acceleration limits and the velocity limit that the representation imposes, i.e., v_{safe} as introduced in Sect. 3.4. If the error e_θ between the path and the robot orientation exceeds a certain threshold, an in-place rotation is performed to keep the robot facing the driving direction. To avoid discrepancies between the local and global planner, both use the same global representation.

4.2 Task integration

With the representation, the global and the local planner the robot can move from a start pose to a target pose. However, navigation systems typically also address issues such as recovery behaviors and replanning to make the system more robust. Nevertheless, navigation is part of a larger task. Hence, it should not be considered in itself and integration with other modules is essential to optimize performance.

Therefore, in order to enhance this integration, the navigation system is coordinated by a task executer, i.e., the software component that activates the various subsystems of the robot to achieve a certain task. This executer has additional knowledge of both the environment and the task so that it can make more informed decisions about the desired behavior. This manifests itself in the definition of goals, actively directing sensors and in recovery behaviors:

Goals Goals can be defined as a pose x, y, θ in the global coordinate frame or as a semantic query to the robots knowledge base. This query will return a list of possible poses that meet the requirement. The best target pose is selected based on distance to travel and proximity to obstacles. Not only do these queries provide a more intuitive interface (you can ask the robot to drive to, e.g., 'the table' or 'a shelf that has not yet been visited'), but they also make navigation more robust by trying the next pose in the list if a target pose turns out to be unreachable.

Directing sensors Actively directing sensors, e.g., gaze direction, greatly enhances the environment representation. Since the task of the robot may put additional requirements on the gaze direction, the gaze direction itself is also controlled by the task executer. Under normal operation, the robot looks to the current path at a fixed distance in front of him. If an obstacle is encountered on the path, the robot will turn its attention to this obstacle.

Recovery behaviors As is also mentioned in, e.g., Marder-Eppstein et al. (2010), even with a good navigation system the robot can still get stuck in some situations, exposing the need for recovery behaviors. Especially in cluttered, dynamic environments, the robot may not be able to clear all obstacles that are no longer there. By explicitly looking at obstacles,

the probability of moving obstacles and sensor noise being cleared increases significantly. The currently used recovery behaviors include clearing the space around the robot if the local planner is stuck and resetting the representation to its default state, consisting only of obstacles and unknown space, if no valid global plan is possible. The task executer decides when these behaviors are executed, easing integration of possible future behaviors such as asking people to move out of the way or removing obstacles by itself.

5 Experimental results

The navigation approach in this paper is verified using the AMIGO robot, a domestic service robot developed by Eindhoven University of Technology. AMIGO competes in the RoboCup@Home League. This annual competition, where domestic service robots compete in performing household tasks, is part of the international RoboCup project (Kitano et al. 1997). AMIGO has a four-wheeled omni-directional base that is capable of navigating through wheelchair-accessible areas. Its torso is equipped with two anthropomorphic arms to perform manipulation tasks. To extend the reach of the arms the torso is connected to the circular base with a lifting mechanism. In this study the arms are not used. A Hokuyo UTM-30LX Laser Scanner, positioned at the front-side of its base, provides a 220° view at 30 cm above the ground. AMIGO uses adaptive Monte Carlo localization (AMCL) (Fox 2001) to localize itself on a static, a priori map. A Microsoft Kinect mounted with a pan-tilt unit on top is used to provide three-dimensional pointcloud data.

Simulations have been performed in a scenario based on the 2013 RoboCup@Home League set-up and experiments have been performed in a domestic environment in the Robotics Lab of the Eindhoven University of Technology as well as in the university library. The parameters are set according to Table 1. The computation time to update the representation is approximately 0.35 s. As it can take up to two updates before an obstacle is detected, the maximum delay is set to 0.7 s and as a result, the maximum robot velocity, computed according to (9) is 0.7 m/s. The voxel resolution of the representation is $r = 0.05$ m, such that AMIGO can still fit through the narrowest doorway with a width of 80 cm. The covariance terms of the position uncertainty model are based on the covariance matrix provided by the AMCL module. For simplicity, the maximum variance obtained from previous tests with AMIGO is used.

Table 1 Different model parameters used in experiments

Sensors		Environment		Robot	
l_{free}	-1.10 ($p = 0.25$)	$v_{\text{obs}}^{\max}$	1.0 m/s	$v_{\text{rob}}^{\max}$	0.7 m/s
l_{occ}	$+0.85$ ($p = 0.70$)	R_s	3.2 m	v_{thresh}	$0.2 v_{\text{rob}}^{\max}$
$l_{\min}$	-1.40 ($p = 0.20$)	t_d	0.7 s	a_m	0.5 m/s^2
$l_{\max}$	$+2.20$ ($p = 0.90$)	r	0.05 m	σ_x^2, σ_y^2	0.1 m
				α	0.05

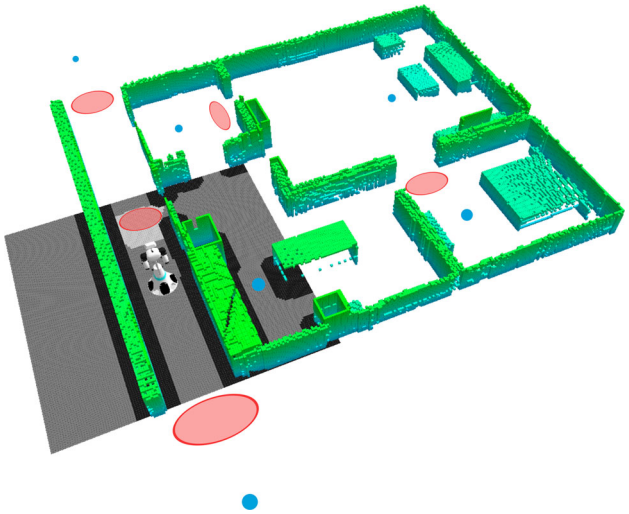


Fig. 5 A model of the 2013 RoboCup@Home League set-up used for the simulation experiments. The *blue dots* indicate the predefined goal locations that are randomly visited by AMIGO. The areas that are marked *red* indicate regions where an obstacle can cross with AMIGO

5.1 Results of simulation experiment

Simulations are performed with AMIGO in a model of the RoboCup@Home League 2013 set-up, as shown in Fig. 5. The representation is tested for the velocity risk profiles $n = \{0.1, 1, 10\}$, probability decay rates $\Delta_{\text{dec}} = \{0.0, 0.15, 0.30\}$ and with and without taking into account moving obstacles, $v_{\text{obs}}^{\max} = \{0.0, 1.0\}$ m/s. The position uncertainty is modeled according to the parameters in Table 1 for all tests.

In a first test the robot drives through the corridor, as shown in Fig. 5, while an obstacle appears from the occluded doorway on its right, thereby blocking the robot's path. This simulation demonstrates the effect of taking obstacle velocities into account as well as the difference between the low, medium and high risk velocity functions. To focus on these two effects, the rate of probability decay is not considered in this test. The velocity profile along the path is presented in Fig. 6. At the beginning of the test the part in front of the robot is unknown and hence it moves at the speed defined by the velocity threshold. When the obstacle velocity is taken into account, i.e., $v_{\text{obs}}^{\max} = 1.0$ m/s, the unknown area is inflated. Since this unknown area behind the robot is inflated past the robot footprint, the threshold velocity for $v_{\text{obs}}^{\max} = 1.0$ m/s must be attained longer than for $v_{\text{obs}}^{\max} = 0.0$ m/s. Near the doorway the test with $v_{\text{obs}}^{\max} = 0.0$ m/s disregards the increased probability of collision due to moving obstacles resulting in a collision for a medium ($n = 1$) and high

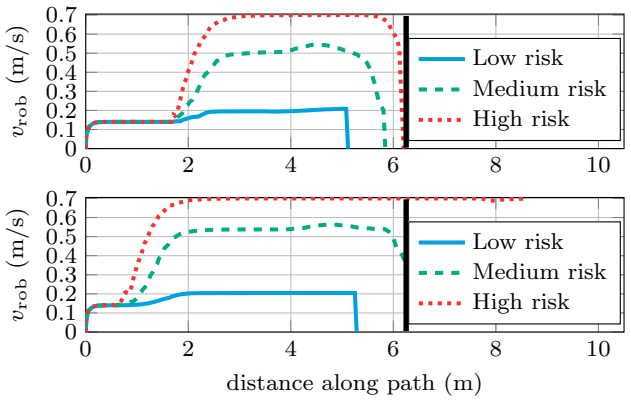


Fig. 6 The velocity of AMIGO with (*upper figure*) and without (*lower figure*) taking the obstacle velocity into account, respectively $v_{\text{obs}}^{\max} = 1.0$ m/s and $v_{\text{obs}}^{\max} = 0.0$ m/s, along the same path through the corridor of the simulation environment. Halfway the corridor an obstacle entered on AMIGO's path from an occluded doorway, indicated by the *black line* at 6.25 m. In the *upper plot*, the robot stops before colliding with the obstacle, regardless of the velocity function. In the *lower plot*, the robot collides with the obstacle if a medium or high risk velocity profile is used

($n = 10$) risk velocity profile. For $n = 10$ the robot even failed to recognize the obstacle and hit it with maximum velocity. Due to the position uncertainty the robot typically moves at a velocity near its threshold for a low risk velocity profile, thereby giving it enough time to detect the obstacle and to stop in time. The test with $v_{\text{obs}}^{\max} = 1.0$ m/s shows that a velocity near the maximum velocity, i.e., $v_{\text{rob}}^{\max} = 0.7$ m/s, is possible without collision.

In a second simulation AMIGO drives to random locations, within a set of predefined locations as depicted in Fig. 5, for 30 min. Obstacles are modeled to move at random time intervals over predefined paths that will intersect with AMIGO, resulting in areas with an increased risk of collision similar to the test in the corridor. By navigating over a longer timespan the influence of the rate of probability decay (Δ_{dec}) can be determined on the performance. As a measurement of performance the number of collisions and the average velocity during the test is reported in Table 2.

The results show that the rate of probability decay has a noticeable influence on the number of collisions. For $\Delta_{\text{dec}} = 0.15$ the number of collisions drops significantly: compared to $\Delta_{\text{dec}} = 0.0$, the number of collisions using $v_{\text{obs}}^{\max} = 1.0$ m/s, the number of collisions decreases from 1 to 0 for a low risk velocity profile, from 5 to 0 for a medium risk velocity profile and from 9 to 3 for a high risk velocity profile. The difference with tests with $\Delta_{\text{dec}} = 0.3$ is not significant: the only difference occurs with $v_{\text{obs}}^{\max} = 1.0$ m/s and a high risk velocity profile, where the number of collisions decreases from 3 to 1. Furthermore, the tests with a low risk velocity function, i.e., $n = 0.1$, resulted in a robot that appeared too conservative as it was not able to achieve its goal position in most situations. Hence, accepting some risk results in a robot

Table 2 Simulation test results

Rate of change	v_{safe} profile		
	Low risk ($n = 0.1$)	Medium risk ($n = 1$)	High risk ($n = 10$)
$v_{\text{obs}}^{\max} = 1.0$ m/s			
$\Delta_{\text{dec}} = 0.3$	$n_{\text{coll}} = 0$	$n_{\text{coll}} = 0$	$n_{\text{coll}} = 1$
	$\bar{v} = 0.11$	$\bar{v} = 0.24$	$\bar{v} = 0.30$
$\Delta_{\text{dec}} = 0.15$	$n_{\text{coll}} = 0$	$\mathbf{n_{coll} = 0}$	$n_{\text{coll}} = 3$
	$\bar{v} = 0.14$	$\mathbf{\bar{v} = 0.27}$	$\bar{v} = 0.32$
$\Delta_{\text{dec}} = 0.0$	$n_{\text{coll}} = 1$	$n_{\text{coll}} = 5$	$n_{\text{coll}} = 9$
	$\bar{v} = 0.19$	$\bar{v} = 0.30$	$\bar{v} = 0.35$
$v_{\text{obs}}^{\max} = 0.0$ m/s			
$\Delta_{\text{dec}} = 0.3$	$n_{\text{coll}} = 0$	$n_{\text{coll}} = 3$	$n_{\text{coll}} = 7$
	$\bar{v} = 0.13$	$\bar{v} = 0.26$	$\bar{v} = 0.31$
$\Delta_{\text{dec}} = 0.15$	$n_{\text{coll}} = 0$	$n_{\text{coll}} = 3$	$n_{\text{coll}} = 7$
	$\bar{v} = 0.14$	$\bar{v} = 0.29$	$\bar{v} = 0.35$
$\Delta_{\text{dec}} = 0.0$	$n_{\text{coll}} = 4$	$n_{\text{coll}} = 12$	$n_{\text{coll}} = 14$
	$\bar{v} = 0.20$	$\bar{v} = 0.33$	$\bar{v} = 0.43$

Bold values indicate the best simulation results in this table: no collisions and an average velocity of 0.27 m/s. Therefore, the parameters used to obtain these results (n = 1, $v_{\text{obs}}^{\max} = 1.0$ m/s, $\Delta_{\text{dec}} = 0.15$) are used in the experiments described in Sects. 5.2 and 5.3

that navigates with a higher average velocity without necessarily increasing the number of collisions. In a few occasions, a moving obstacle was not completely cleared before it was out of the sensor view and therefore cluttered the environment, preventing the robot from reaching its goal. A model for the decrease in occupancy probability for occupied volumes, as discussed in Sect. 3.2, would alleviate this problem.

It can be concluded that the increased probability of collision due to moving obstacles must be taken into account and the probability of occupancy must be time-dependent in order to navigate without collision. Furthermore, the trade-off between velocity and the probability of collision is clear. Accepting some risk of collision during navigation, i.e., choosing a medium risk velocity limit (see Fig. 4), results in a robot that moves at its maximum velocity if it can and at its velocity threshold when it must to ensure safety.

5.2 Validation in a laboratory experiment

The proposed environment representation is validated by an experiment on the robot in a domestic environment. This environment, shown in Fig. 7a, is a partial replica of the 2103 RoboCup@Home League set-up on a slightly smaller scale (0.9:1). In the experiment the robot visits a set of waypoints numbered w_1–w_7. The robot is initialized with a three-dimensional map, as depicted in Fig. 7b, and a static map for localization, as depicted in Fig. 7c, that both are generated off-line. A variety of challenging obstacles are

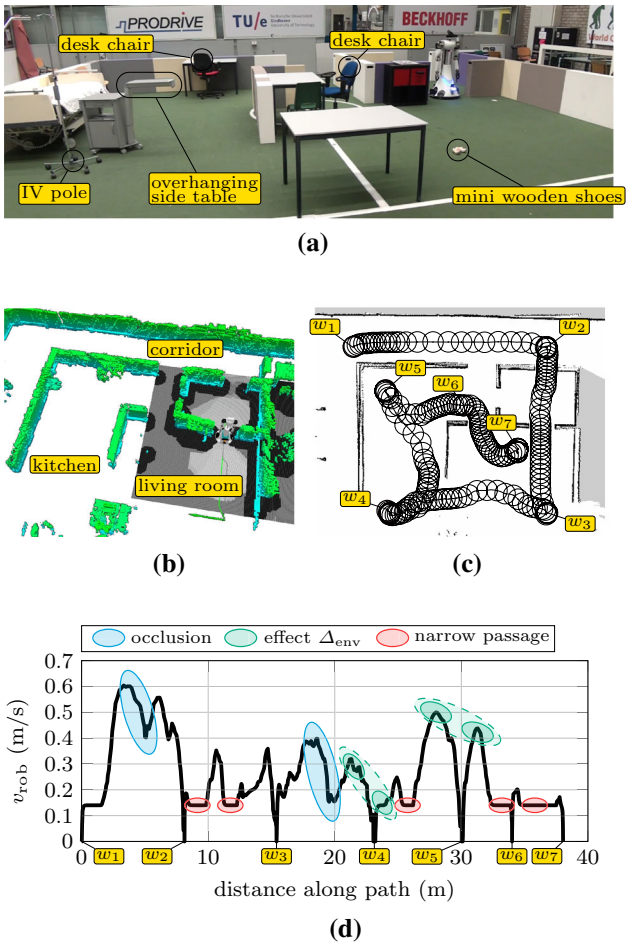


Fig. 7 An experiment with AMIGO in a real domestic environment. The robot drives to a number of predefined waypoints (see Fig. 7c) in the environment depicted in Fig. 7a, b. The resulting velocity profile can be seen in Fig. 7d where a decrease in velocity due to occlusions is marked *cyan*, a velocity difference due to a changing environment is marked *green* and a decreasing velocity due to a narrow passage is marked *red*. **a** Test set-up at the Robotics Lab of Eindhoven University of Technology. **b** Robot visualization. **c** Footprints on the localization map. **d** The robot velocity along the executed path during the experiment

present, that are not in the a priori map, such as a pair of mini wooden shoes between w_3 and w_4, the IV pole and overhanging sidetable near the hospital bed at w_4 and the desk chairs near w_5 and w_6. The parameter set as used in the simulation experiments is used for the experiment and, based on the simulation test results, the variable parameters are set to $v_{obs}^{max} = 1.0$ m/s, $\Delta_{dec} = 0.15$ and $n = 1$.

AMIGO navigated without collision in the domestic environment. In Fig. 7c the robot's footprint on the localization map and its executed path are shown at every second. The velocity along the executed path is displayed in Fig. 7d. The three-dimensional representation correctly represented the obstacles encountered during the run. At parts with an increased probability of collision due to occlusions, e.g., near the doorway in the corridor, the robot lowered its velocity,

Fig. 8 AMIGO driving through the university library

allowing it to stop in time if an obstacle suddenly appears on its path. In narrow passages the robot lowered its velocity to its threshold to be robust against position uncertainty. The effect of the probability decay rate Δ_{dec} is noticeable near w_4 and w_5. The space near these waypoints is marked as free as the robot approaches. However, as the robot continues to its next waypoint the space becomes unknown again as it is out of sensor range. Hence, a lower velocity limit is present. In open and known space, such as at the beginning of the corridor and near w_5, the robot achieved velocities near its maximum velocity. The sudden decreases in velocity between waypoints are the results of in-place rotations to keep the robot facing its driving direction.

5.3 Experiences in a real world experiment

Although the experiment in the previous section clearly demonstrates the caution that the robot takes in case of risk of collisions due to occlusions, to a dynamic environment and narrow passages, the environment itself actually is static without any people moving around. Since the robot is supposed to navigate through a domestic environment, a second experiment has been conducted. This has taken place in the library of the Eindhoven University of Technology (see Fig. 8). The library is much larger (approximately 70 m × 40 m) than the environment in Sect. 5.2 and much more challenging because of the amount op people walking around: the floor where the experiment was performed can host up to 300 people. Furhtermore, paths being blocked by, e.g., chairs and bags located at random locations pose additional challenges.

The first step to obtain a clean default Octomap of the environment was to make a localization map (see Fig. 9). Subsequently, this map was post-processed to remove people, chairs, bags and other non-stationary objects. Furthermore, overhanging desks were included. The result can be seen in Fig. 10a. Finally, this was extruded to obtain a default Octomap, see Fig. 10b.

In this environment, a number of interesting locations was defined as if the robot was showing people the way to, e.g., the elevator, the copier or the location of certain books. Using a

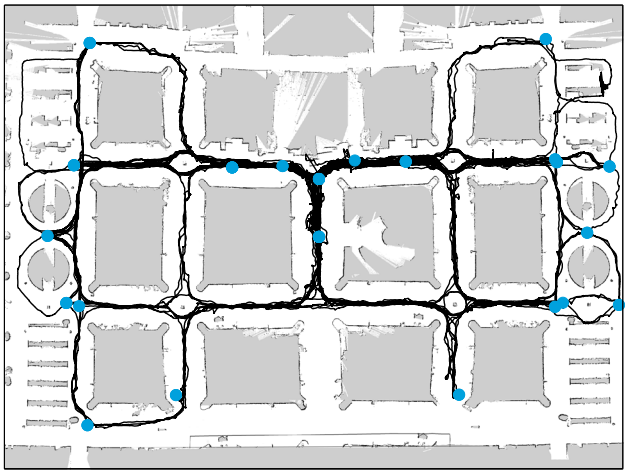
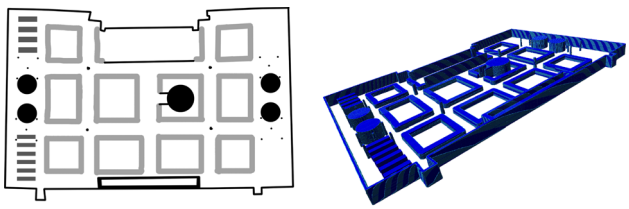
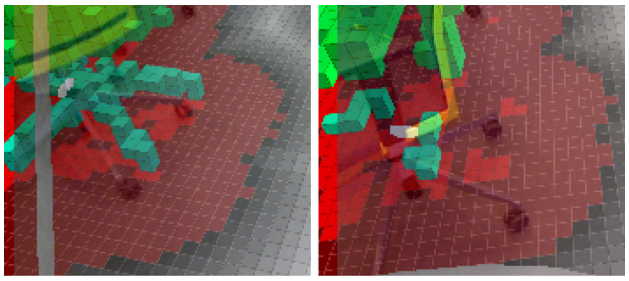


Fig. 9 The localization map of the university library and the paths AMIGO took during the experiments. The goal locations are denoted in *blue*



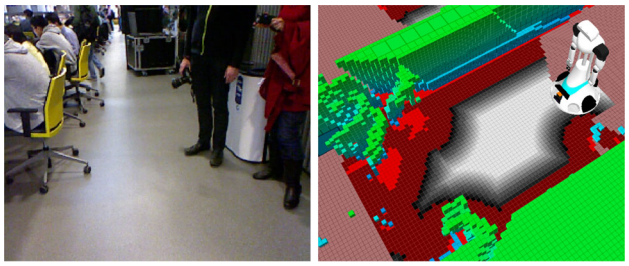
(a) **(b)**

Fig. 10 The localization map was post-processed as an intermediate step. People, chairs and bags were removed and overhanging desks were included. Subsequently, this map was extruded to produce the default Octomap. **a** The processed map. **b** The default Octomap



(a) **(b)**

Fig. 11 Overlay of the camera image and the Octomap and costmap. Usually, chairs and their legs were mapped correctly but occasionally the legs did not appear in the Octomap. **a** The chair has been mapped correctly. **b** The chair has not been mapped correctly



(a) **(b)**

Fig. 12 An example where a moving person has *left* some 'clutter' in the Octomap. **a** View from the robot: it can easily pass through between the people and the chairs. **b** However, this is prevented by some occupied voxels that are the remainder of a person who has just walked by

smartphone, a user could select a location and the robot would show him the way. In case there was no pending goal request, the robot would select one at random. This way, the robot covered a total of 2.9 km in this environment. Although this is by far not as much as in, e.g., Marder-Eppstein et al. (2010), a number of conclusions can be drawn from this experiment.

The representation approach presented in Sect. 3 works. Even difficult objects such as the legs of the deskchairs were generally perceived well, as can be seen in Fig. 11a. However, slight variations in the robot localization could have as a result that multiple voxels in the vicinity of the legs received an 'occupied' measurement but not enough to reach the upper threshold. In that case, the chair would not be mapped correctly (see, Fig. 11b) but usually the robot would still drive around the chair. If the available space was very tight, nevertheless, the robot would slightly touch the chairs. The bags of students that were on the floor were detected correctly every time.

On a few occasions, however, this was not the case (see Fig. 11b).

During the experiments, many people were walking around in the library. Especially around lunch time, there were many people taking a close look at the robot and thereby obstructing its path. In many situations, the robot was still able to continue after a replan. On a number of occasions, however, the robot was stuck and a recovery behavior, either clearing the Octomap in the vicinity of the robot (nineteen times) or resetting the entire costmap (eighteen times) had to be invoked, which solved the problem in all but four cases. Moving people demonstrated a shortcoming of this representation: since the voxels are all independent, the representation might get cluttered if not all voxels of a moving obstacle are cleared. An example of this can be seen in Fig. 12: a couple of floating occupied voxels just outside the sensor range R_s that are the remainder of a person passing by prevent the robot from planning a path forward (Fig. 12b), although there is clearly enough space to pass through the chairs on the left and the people on the right, as can be seen in the robot's view (Fig. 12a). Associating this data with people tracked using, e.g., a laser rangefinder as is done in Rohrmüller et al. (2008) could prevent these issues.

The behavior demonstrated in the lab experiment was also visible in this experiment, particularly the lower velocities due to narrow passages and occluded areas. The former proved to be effective to avoid collisions, however, the latter proved to be too conservative: due to the cluttered environment, there were always (small) unknown areas in the vicinity

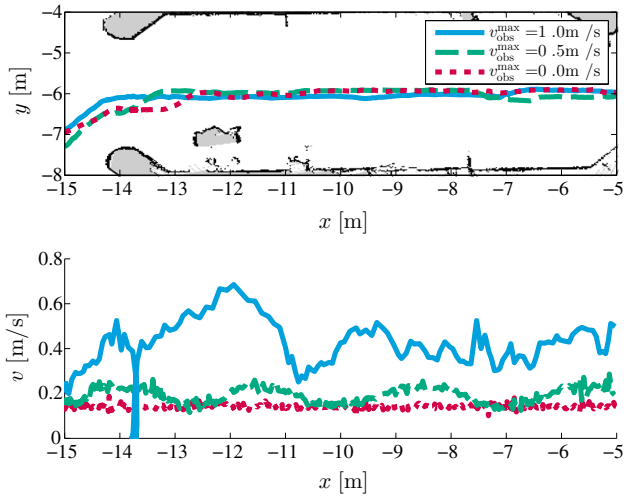


Fig. 13 Three approximately similar paths (*upper plot*) result in significantly varying velocities (*lower plot*) for $v_{\text{obs}}^{\max} = 1.0$ m/s, $v_{\text{obs}}^{\max} = 0.5$ m/s and $v_{\text{obs}}^{\max} = 0.0$ m/s

Table 3 The average robot velocity $\bar{v}$ for varying $v_{\text{obs}}^{\max}$. If $v_{\text{obs}}^{\max} = 1.0$ m/s, $\bar{v} \approx v_{\text{thresh}}$ which is considered too conservative

$v_{\text{obs}}^{\max}$ (m/s)	$\bar{v}$ (m/s)
1.0	0.13
0.5	0.17
0.0	0.28

of the robot because the sensors could not see beyond bags or the legs of people or chairs. After inflation, these caused the robot to drive at the minimum velocity v_{thresh} most of the time. To illustrate this effect, the experiment was conducted with three different values for $v_{\text{obs}}^{\max}$. In Fig. 13, the robot passes through the same part of the environment with $v_{\text{obs}}^{\max} = 1.0$ m/s, $v_{\text{obs}}^{\max} = 0.5$ m/s and $v_{\text{obs}}^{\max} = 0.0$ m/s. In case $v_{\text{obs}}^{\max} = 1.0$ m/s, the robot velocity hardly exceeds the lower bound $v_{\text{thresh}} = 0.14$ m/s, which is even in this environment not sufficient. This is confirmed by the average velocities for the different $v_{\text{obs}}^{\max}$, as can be seen in Table 3.

A major difference with the lab experiment was the library floor, which provided less grip than the carpet in the lab experiment. The local planner proved not sufficiently robust for the resulting slip and therefore caused oscillations. This manifested itself especially in the orientation with an amplitude around 0.2 rad/s. However, a larger problem that was caused by both slip and some inaccuracies in the localization map was a large localization error. This error occasionally exceeded 0.8 m and 0.16 rad, which is much more than originally modeled with σ_x^2, $\sigma_y^2 = 0.1$ m. Since this method only has a global representation method and does not rely on a local collision map, this error can cause the robot either to get stuck (see Fig. 14) or to cause collisions. In this experiment, the localization error

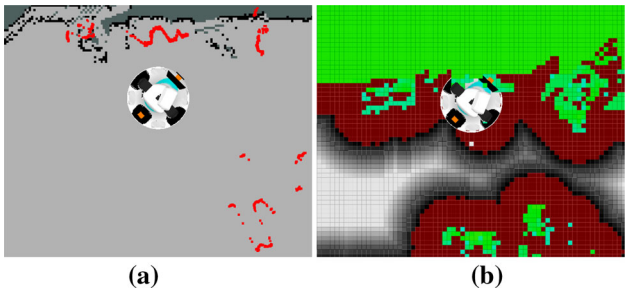


Fig. 14 An example where the robot gets stuck due to a localization error. **a** A localization error occurs. **b** As a result, the robot gets stuck

resulted in imminent collisions on two occasions which is not acceptable in the application. This illustrates the need for a local representation as well, as was already argued in Moore et al. (2009).

6 Discussion

6.1 Parameters

Inflating the obstacle representations often implies that a more or less arbitrary (exponentional) decay function is used. One of the motivations of this research was to eliminate these arbitrary functions and tuning parameters by modeling the various sources of uncertainty of an environment separately. Although many parameters such as the maximum obstacle velocity, the maximum sensor range, maximum update delay and robot velocity and acceleration are well-defined, there are still parameters that are not directly related to a measurable physical quantity. Most notably, the probability decay rate Δ_{dec} and the function relating the probability of collisions to a safe velocity (see Fig. 4) are selected empirically by doing the extensive simulations presented in Sect. 5.1. Future research should provide the necessary insights to relate these model parameters to measurable quantities as well.

6.2 Independency of measurements

As a result of the time dependency of the environment representation in this work, the free space becomes unknown over time but occupied space remains occupied. Hence, if certain voxels of dynamic obstacles are not cleared correctly, these will impede motion planner and might cause the robot to take unnecessary detours or even prevent the robot from reaching its goal. Simply forgetting obstacles over time, on the other hand, might also lead to unsafe situations. This problem is inherent to the independency of the voxels. Therefore, a representation significantly benefits if measurements are associated with the objects in the environment: if the robot

detects that an obstacle has moved away from its path, it can immediately clear all associated voxels. Furthermore, this enables to explicitly account for additional properties such as movement of obstacles.

7 Conclusions and future work

The proposed volumetric representation allows a robot to safely navigate in a domestic environment. It probabilistically models the occupancy of volumes as sensor measurements are received and, as opposed to typical representations, also if no measurements are received. Furthermore, the probability of moving obstacles appearing on the robot's path from behind occlusions is taken into account. These probabilities are combined with a model of the robot position uncertainty to form a probability of collision. Based on this probability a safe velocity limit is defined.

Extensive simulations have demonstrated that this approach results in the desired behaviour, i.e., the robot moves with velocities up to the maximum of 0.7 m/s if this is safe but slows down in case of narrow passages or uncertain areas of the environment. This has been confirmed by laboratory experiments, where the same behavior was demonstrated and challenging obstacles such as small objects on the floor or overhanging tables were successfully avoided. This approach also worked in a real-world experiment, performed in the university library. However, it was found that the inflation of the uncertain areas was still too conservative. Furthermore, the approach was not sufficiently robust against localization errors.

Several directions can be indicated for improvement of the current approach in future work. A number of conservative assumptions were necessary to ensure safe navigation, in particular (i) obstacles may maintain their maximum velocity if they occur on the robot's path, (ii) obstacles *never* disappear, hence the probability of occupied obstacles $P > 0.5$ does not decrease over time and (iii) dynamic obstacles can emerge at any height from every voxel, thus the maximum occupancy probability of a column is inflated . By including more information, e.g., (i) obstacles are not adversary and will try to avoid collisions, (ii) certain obstacles such as humans may move away over time and (iii) there are no flying obstacles, these assumptions can be relaxed to result in more efficient robot navigation. The representation can be improved by adding explicit obstacle models of, e.g., humans as proposed in Philippsen et al. (2006), Philippsen et al. (2008) and Rohrmüller et al. (2008), but also of static parts of the environment, e.g., walls. By discriminating in obstacle representations, the probability of their presence can be modeled separately and thus more accurately. In turn, this will allow a less conservative velocity limit. Furthermore, an actuated sensor can be controlled to actively reduce uncertainty in the vicinity of the robot instead of always looking forward on the robot's path. This will decrease the collision probability and thereby increase the safe velocity limit. For example, at start-up the robot is then able to directly face the uncertain space in front of its base, while during navigation it can look further ahead. The position uncertainty can also be modeled more accurately. It is now represented with a normal distribution based on an a priori determined maximum variance, while directly using the covariance matrix from the AMCL module is more accurate because this is updated based on the sensor measurements. Additional robustness against localization errors can be added by a local representation, i.e., directly reacting to measurements. Better performance of the total system can be achieved by an improved local planner. Finally, deeper insight in the choice of model parameters is desirable. It would be particularly useful to investigate how to measure the probability decay rate Δ_{dec} of a certain environment and how to relate the safe velocity v_{safe} to the probability of collision $P(n)$. Although those parameters depend on the environment as well as the specific application at hand, a theoretical and extended simulative analysis in different environment set-ups can reduce heuristic tuning of parameters and make the presented method better generalizable.

Acknowledgements The research leading to these results is part of the R5-COP project, funded by ARTEMIS and NL Agency (Dutch Ministry of Economic Affairs).

References

Alami, R., Siméon, T., & Madhava Krishna, K. (2002). On the influence of sensor capacities and environment dynamics onto collision-free motion plans. In *IEEE/RSJ international conference intelligent robots and systems* (Vol. 3, pp. 2395–2400).

Althoff, D., Althoff, M., Wollherr, D., & Buss, M. (2010). Probabilistic collision state checker for crowded environments. In *2010 IEEE international conference on robotics and automation* (pp 1492–1498).

Althoff, M., & Dolan, J. (2014). Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, *30*(4), 903–918.

Althoff, M., & Mergel, A. (2011). Comparison of markov chain abstraction and monte carlo simulation for the safety assessment of autonomous cars. *IEEE Transactions on Intelligent Transportation Systems*, *12*(4), 1237–1247.

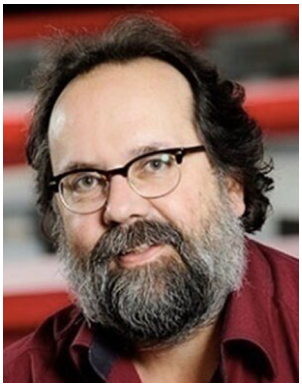
Bautin, A., Martinez-Gomez, L., & Fraichard, T. (2010). Inevitable collision states: A probabilistic perspective. In *2010 IEEE international conference on robotics and automation* (pp. 4022–4027).

Bouraine, S., Fraichard, T.,& Salhi, H. (2012). Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic

environments. In *2012 IEEE international conference on robotics and automation* (pp. 174–179).
Burns, B., & Brock, O. (2007). Sampling-based motion planning with sensing uncertainty. In *IEEE international conference on robotics and automation* (pp. 3313–3318).
Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., et al. (2005). *Principles of robot motion: Theory, algorithms, and implementations*. Cambridge: MIT Press.
Chung, W., Kim, G., Kim, M., & Lee, C. (2004). Integrated navigation system for indoor service robots in large-scale environments. In *Proceedings of the IEEE international conference on robotics and automation* (Vol. 5, pp. 5099–5104).
Coenen, S., Lunenburg, J., van de Molengraft, M., & Steinbuch, M. (2014). A representation method based on the probability of collision for safe robot navigation in domestic environments. In *Proceedings of the 2014 IEEE/RSJ international conference on intelligent robots and systems*
Eidehall, A., & Petersson, L. (2008). Statistical threat assessment for general road scenes using monte carlo sampling. *IEEE Transactions on Intelligent Transportation Systems*, *9*(1), 137–147.
Fox, D. (2001). KLD-sampling: Adaptive particle filters and mobile robot localization. *Advances in Neural Information Processing Systems*, *14*(1), 26–32.
Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Magazine on Robotics and Automation*, *4*(1), 23–33.
Fraichard, T., & Asama, H. (2003). Inevitable collision states. A step towards safer robots? In *Proceedings of the 2003 IEEE/RSJ international conference on intelligent robots and systems* (Vol. 1 pp. 388–393).
Goerzen, C., Kong, Z., & Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems, Theory and Applications*, *57*(1–4), 65–100.
Guibas, L. J., Hsu, D., Kurniawati, H., & Rehman, E. (2009). Bounded uncertainty roadmaps for path planning. In *Algorithmic foundation of robotics VIII* (pp. 199–215). Springer.
Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, *34*, 1–18.
Hsu, D., Kindel, R., Latombe, J. C., & Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, *21*(3), 233–255.
Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., & Osawa, E. (1997). RoboCup: The robot world cup initiative. In W. L. Johnson (Ed.), *Proceedings of the 1997 1st international conference on autonomous agents* (pp. 340–347), ACM, Marina del Rey, CA, USA.
Lingemann, K., Nüchter, A., Hertzberg, J., & Surmann, H. (2005). About the control of high speed mobile indoor robots. In *Proceedings of the second european conference on mobile robots* (pp. 218–223).
Lunenburg, J., Coenen, S., van den Dries, S., Elfring, J., Janssen, R., Sandee, J., & van de Molengraft, M. (2013). Tech united eindhoven team description 2013.
Lunenburg, J., Coenen, S., Derksen, T., van den Dries, S., Elfring, J., & van de Molengraft, M. (2014).Tech united eindhoven @home team description 2014.
Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., & Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *Proceedings of the 2010 IEEE international conference on robotics and automation (ICRA)* (pp. 300–307).
Missiuro, P., & Roy, N. (2006). Adapting probabilistic roadmaps to handle uncertain maps. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1261–1267).
Moore, D., Huang, A., Walter, M., Olson, E., Fletcher, L., Leonard, J., & Teller, S.(2009). Simultaneous local and global state estimation for robotic navigation. In *2009 IEEE international conference on robotics and automation* (pp. 3794–3799).
Moravec, H. P. (1988). Sensor fusion in certainty grids for mobile robots. *AI Mag*, *9*(2), 61.
Patil, S., van den Berg, J., & Alterovitz, R. (2012). Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In *IEEE international conference on robotics and automation* (pp. 3238–3244).
Philippsen, R., Jensen, B., & Siegwart, R. (2006). Toward online probabilistic path replanning in dynamic environments. In *2006 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2876–2881).
Philippsen, R., Kolski, S., Macek, K., & Jensen, B. (2008). Mobile robot planning in dynamic environments and on growable costmaps. In *Workshop on planning with cost maps at the IEEE international conference on robotics and automation*.
Rohrmüller, F., Althoff, M., Wollherr, D., & Buss, M. (2008). Probabilistic mapping of dynamic obstacles using markov chains for replanning in dynamic environments. In *2008 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2504–2510).
van den Berg, J., Abbeel, P., & Goldberg, K. (2011). LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *International Journal on Robotics Research*, *30*(7), 895–913.

Janno Lunenburg received his M.Sc. degree in Mechanical Engineering from Eindhoven University of Technology (TU/e) in 2010. In June 2015 he received his Ph.D., also at TU/e. His research topics included motion planning for domestic service robots and the development of a modular service robot. Currently, he is with Smart Robotics, working on flexible automation. Furthermore, he participates in RoboCup@Home with Tech United Eindhoven.

René van de Molengraft (1963) received the M.Sc. (cum laude) and Ph.D. degrees in Mechanical Engineering from TU/e in 1986 and 1990. In 1991 he fulfilled his military service. Since 1992 he is a staff member of the CST group. Since 2005, he is project leader of the Tech United RoboCup team, which placed second in the 2008–2011 and 2013 RoboCup Middle Size League world championships and first in 2012 and 2014. He is coauthor of more than 50 papers in refereed scientific journals and more than 95 papers in refereed scientific proceedings.



Maarten Steinbuch (1960) is Distinguished University Professor at Eindhoven University of Technology (TU/e), The Netherlands, and head of the Control Systems Technology group. He received the M.Sc. and Ph.D. degrees in 1984 and 1989 resp. From 1987–1999 he was with Philips Electronics. He is Editor-in-Chief of IFAC Mechatronics. He is Scientific Director of the TU/e High Tech Systems Center, and he is director of the TU/e Graduate Program Automotive Systems. His research interests are in the field of mechatronics, robotics, automotive power trains and control of fusion plasmas.