# A Monte Carlo Approach for Collision Probability Computation

Alain Lambert
IEF, UMR CNRS 8622
Université Paris Sud-XI
Bat. 220, Centre d'Orsay
91405 Orsay cedex - France

Guillaume Saint Pierre, Dominique Gruyer
LIVIC
INRETS/LCPC
Bat 824, 14 route de la minière
78000 Versailles Satory - France

***Abstract*— In order to navigate safely, it is important to detect and to react to a potentially dangerous situation. Such a situation can be underlined by a judicious use of the locations and the uncertainties of both the navigating vehicle and the obstacles. We propose to build an estimation of the collision probability from the environment perception with its probabilistic modelling. The probability of collision is computed from a product of integrals of a product of Gaussians. The integrals take into account the uncertain configurations and the volume of both the vehicle and the obstacles.**

## I. Introduction

The anticipation of a collision is necessary for a safe navigation. The prediction of collisions could be used for obstacle avoidance, speed monitoring or path planning. Such a prediction has been computed in various ways during the last years.

[1] defines a security area modeled by a circle (centered on the robot position) whose radius is proportional to the speed. A collision judgement is based on an intersecting test between this circle and high-confidence position error ellipses. Controlling the speed and steering of the mobile robot along a preplanned path is done by using the collision judgement. [2] uses an interaction component (deformable virtual zone) of the robot with the environment which leads to avoidance-oriented control laws. Furthermore an emergency area around the robot causes an emergency stop if it is broken by an obstacle. [3] performs an on-line speed monitoring by computing a time to collision. Longer times to collision lead the higher speed. In order to detect a collision, the authors grow a mobile robot with its uncertainty ellipse and they do a collision test between the resulting shape and the obstacles. The process is repeated all along the path in order to compute a time to collision. Uncertainty ellipses have also been used in [4] for safe path planning. The safety is realized thanks to a collision test between a robot enlarged with its uncertainty ellipsoid and the obstacles. [5] computes a distance to undecided regions (unknown region) or to nearby obstacles. Next they use this distance information to compute the speed.

We think that only using a measured distance to collision [5][6] is not sufficient as the real distance could be quite different and lead to unexpected collisions. Using a security area [1][2] around the vehicle is a good idea only if this security area represents the uncertainty on the vehicle location. Nevertheless such an area (an ellipse [3][4]) is a discrete and binary representation of a continuous probability of presence. Most authors uses an ellipse which represents the probability of presence of the vehicle at 90%. Unfortunately by defining a threshold they loose information for higher level algorithms.

That's why in this paper we propose to use the entire available information (the pdf of the vehicle and the obstacles) for defining the probability of collision. Such an approach has been followed in [7] for a punctual robot and a geometrical obstacle without considering the uncertainty in orientation. We are going to overcome those restrictions (punctual and no uncertainty in orientation) in order to compute a realistic collision probability for real world application. To the best of our knowledge, it is the first time that the 3D uncertainty and the volume of the objects are used when calculating the probability of collision.

In the next section we introduce the necessary models. In section 3 we propose an analytical formula for computing the probability of collision between two configurations. We have no analytical solution when considering objects instead of configurations. Nevertheless, by considering a circular vehicle with its configuration uncertainties and a perfectly localized obstacle, we can tackle the problem with published algorithms (section 4). The general case (both vehicle and obstacle with any shape and added uncertainties) is studied in section 5. In section 6 we consider the probability of collision between multiple objects.

## II. Problem statement

### A. Vehicle and obstacle models

The vehicle configuration is denoted $\mathbf{x}_v = (x_v, y_v, \theta_v)^T$ where $(x_v, y_v)$ are the coordinates of a characteristic point which is located midway between the two rear wheels of our vehicle and $\theta_v$ is its orientation. All variables are defined with respect to the global frame.

The obstacles configuration is denoted $\mathbf{x}_o = (x_o, y_o, \theta_o)^T$. The shape of both vehicle and obstacles are denoted $\mathcal{V}_v$ and $\mathcal{V}_o$. The geometric center of $\mathcal{V}_o$ is $\mathbf{x}_o$.

### B. Uncertainty modeling

The pdf (probability density function) of a configuration $\mathbf{x} = (x, y, \theta)^{\mathrm{T}}$ having a $\mathbf{\Sigma}$ covariance matrix and an $\hat{\mathbf{x}}$ mean

is:

$$p(\mathbf{x}) = \frac{1}{\left(\sqrt{2\pi}\right)^n \sqrt{\det \mathbf{\Sigma}}} e^{-\frac{1}{2}\left((\mathbf{x}-\hat{\mathbf{x}})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\hat{\mathbf{x}})\right)} \quad (1)$$

where $n$ is the $\mathbf{x}$ dimension. We consider that $\mathbf{x}$ dimension is 3 for the sake of simplicity : (nevertheless the $\mathbf{x}$ dimension can be higher).

$$p(\mathbf{x}) = \frac{1}{\left(\sqrt{2\pi}\right)^3 \sqrt{\det \mathbf{\Sigma}}} e^{-\frac{1}{2}\left((\mathbf{x}-\hat{\mathbf{x}})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\hat{\mathbf{x}})\right)} \quad (2)$$

where $\mathbf{\Sigma}$ is a 3x3 covariance matrix :

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_x^2 & \rho_{xy}\sigma_x\sigma_y & \rho_{x\theta}\sigma_x\sigma_\theta \\ \rho_{xy}\sigma_x\sigma_y & \sigma_y^2 & \rho_{y\theta}\sigma_y\sigma_\theta \\ \rho_{x\theta}\sigma_x\sigma_\theta & \rho_{y\theta}\sigma_y\sigma_\theta & \sigma_\theta^2 \end{bmatrix} \quad (3)$$

Such a matrix could be the result of a filter process like the Extend Kalman Filter or could be directly defined by:

$$\mathbf{\Sigma} = E\left((\mathbf{x}-\widehat{\mathbf{x}})(\mathbf{x}-\widehat{\mathbf{x}})^T\right) \quad (4)$$

The pdf of a $v$ vehicle and an $o$ obstacle are denoted $p_v$ and $p_o$ with their associated $\mathbf{\Sigma}_v$ and $\mathbf{\Sigma}_o$ matrices.

Finally, $v$ and $o$ are defined by : $v = (\mathbf{x_v}, \mathbf{\Sigma}_v, \mathcal{V}_v)$ and $o = (\mathbf{x}_o, \mathbf{\Sigma}_o, \mathcal{V}_o)$

## III. Probability of collision between 2 uncertain configurations

The probability of collision between a $v$ and an $o$ uncertain configurations (assuming that $\mathcal{V}_v = \mathcal{V}_o = \emptyset$) is defined by :

$$\mathbb{P}_{coll}(v,o) = \iiint_{\mathbb{R}^3} p_v(x,y,\theta) \cdot p_o(x,y,\theta) \cdot dxdyd\theta \quad (5)$$

The integral 5 can be analytically computed. Details of the calculations in the multivariate case can be found in appendix A. Let's assume that $p_v(x,y,\theta)$ is the density of a $\mathcal{N}_d(\widehat{\mathbf{x}}_v, \mathbf{\Sigma}_v)$ distribution, and $p_o(x,y,\theta)$ the density of a $\mathcal{N}_d(\widehat{\mathbf{x}}_o, \mathbf{\Sigma}_o)$ distribution. Let us denote $\mathbf{x} = (x,y,\theta)^T$, $\mathbf{A} = (\mathbf{x}-\widehat{\mathbf{x}}_v)^T \mathbf{\Sigma}_v^{-1}(\mathbf{x}-\widehat{\mathbf{x}}_v) + (\mathbf{x}-\widehat{\mathbf{x}}_o)^T \mathbf{\Sigma}_o^{-1}(\mathbf{x}-\widehat{\mathbf{x}}_o)$ and $\mathbf{B} = [\mathbf{x}-\mathbf{m}]^T \mathbf{\Sigma}^{-1}[\mathbf{x}-\mathbf{m}]$ with:

$$\mathbf{m} = \mathbf{\Sigma}^{-1}\left(\mathbf{\Sigma}_v^{-1}\widehat{\mathbf{x}}_v + \mathbf{\Sigma}_o^{-1}\widehat{\mathbf{x}}_o\right) \quad (6)$$

$$\mathbf{\Sigma}^{-1} = \left(\mathbf{\Sigma}_v^{-1} + \mathbf{\Sigma}_o^{-1}\right) \quad (7)$$

We have

$$\mathbb{P}_{coll}(v,o) = \frac{\exp\left[-\frac{1}{2}(\mathbf{A}-\mathbf{B})\right]\sqrt{\det(\mathbf{\Sigma})}}{\sqrt{\det(\mathbf{\Sigma}_v)}\sqrt{\det(\mathbf{\Sigma}_v)}} \quad (8)$$

with

$$\begin{aligned}\mathbf{A}-\mathbf{B} = {} & \widehat{\mathbf{x}}_v^T\mathbf{\Sigma}_v^{-1}\widehat{\mathbf{x}}_v + \widehat{\mathbf{x}}_o^T\mathbf{\Sigma}_o^{-1}\widehat{\mathbf{x}}_o \\ & - \left(\mathbf{\Sigma}_v^{-1}\widehat{\mathbf{x}}_v + \mathbf{\Sigma}_o^{-1}\widehat{\mathbf{x}}_o\right)^T \mathbf{\Sigma}^{-1}\left(\mathbf{\Sigma}_v^{-1}\widehat{\mathbf{x}}_v + \mathbf{\Sigma}_o^{-1}\widehat{\mathbf{x}}_o\right)\end{aligned} \quad (9)$$

Equation (8) has been used to compute the probabilities of collision of Fig. 1. During this experiment corresponding to a real outdoor situation, a car (so called "the vehicle", right part of the figure) was running on its way whereas there was another static car (so called "the obstacle") on the opposite lane (left part of the figure). The vehicle was moving from the bottom to the top of the figure in a straight line using only its proprioceptive sensors. The pdf of the vehicle is shown at 6 different time instants (every 4 meters). The pdf of the collision has been computed at each of those time instants but only 2 pdf are noticeable. The maximum height of the biggest pdf of the collision is tiny (0.00205) compared to the corresponding pdf's height of both the obstacle (0.16) and the vehicle (0.13). Consequently the pdf of the collision has been multiplied by 100 on Fig. 1 for a better visualization. The greatest probability of collision computed using Eq. (8) is equal to 0.007. It allows us to conclude that the situation is safe which is unrealistic considering the real situation with the volume of the cars. We should have thought about taking into account the volumes as they are big regarding the estimated distance between the vehicles. We have not investigated the interest of Eq. (8) when the volumes are not null because the following approaches (see section 4 and 5) provides results that are good enough both in term of computing time and precision.

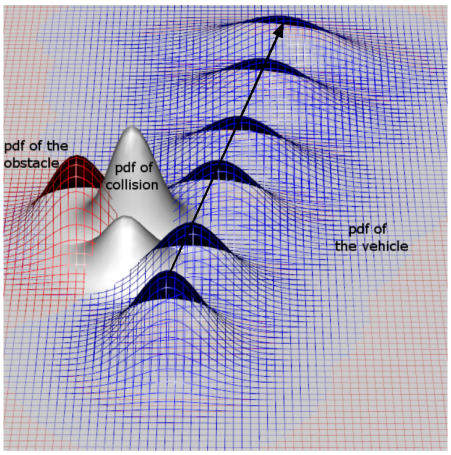



Fig. 1. pdf of the collision between an obstacle and a vehicle moving in straight line

## IV. Probability of collision between a well known circular object and another circular object with Gaussian uncertainties

We consider a basic situation: a circular vehicle is near a fixed obstacle (a column for instance). In this type of situation, there is no uncertainty on both the position and the orientation of the obstacle (the configuration of the obstacle is perfectly known). We want to find the collision probability associated with a given configuration of both the circular vehicle and the column. We consider the circular vehicle as an $\hat{\mathbf{x}}_v = \left(\hat{x}_v, \hat{y}_v, \hat{\theta}_v\right)^T$ configuration with an associated $r_v$ radius and $p_v(\mathbf{x}_v)$ pdf. The circular object has an $\mathbf{x}_o = (x_o, y_o, \theta_o)^T$ configuration with an associated $r_o$ radius.

Thus, the following integral needs to be computed:

$$\int_D p_v(x_v, y_v, \theta_v) dx_v dy_v d\theta_v \quad (10)$$

where

$$D = \{\mathbf{u}_v = (x_v, y_v) \in \mathbb{R}^2 \backslash \quad (11)$$
$$(\mathbf{u}_v - \mathbf{Q}_v)^T \mathbf{E} (\mathbf{u}_v - \mathbf{Q}_v) < (r_o + r_v)^2, \text{ and } \theta_v \in \mathbb{R}\},$$

$\mathbf{Q}_v \in \mathbb{R}^2$ the ellipse center, $E$ the positive semi-definite ellipsoid matrix ($\mathbf{E} = \mathbf{I_d}$ in the case of a circular base of the obstacle), and $r_o + r_v$ the new radius. We compute the integral of the $p_v(\mathbf{x}_v)$ pdf over a circular region whose radius is the radius of the column plus the diameter of the vehicle.

This simple case is not analytically tractable: multivariate normal integrals computation is often a difficult problem. For the bivariate case, there are now many algorithms available for integrals computation over various integration regions (spherical, ellipsoidal, rectangular, circular, etc.). In a comparative study of these algorithms, Terza and Welland [8] show that the quality of these algorithms has a significant variation. Recently, after a study of different algorithms, Patefield and Tandy [9] developed an hybrid double precision algorithm. In the case of dimensions higher than 2, the computation objective for multivariate normal integrals can be reached either by using a numerical approximations involving Monte-Carlo methods [10], or a locally adaptive numerical integration strategy based on Simson's rule [11], or sub-region adaptive multiple integration method [12]. Such algorithms are now well known, and give accurate results for the multivariate normal integrals computation problem.

We chose to use the AS 106 algorithm (which is described in [13]) in order to compute Eq. (10). Given that the n-dimensional vector $\mathbf{x}_v$ has a multivariate normal distribution with expected mean value vector $\widehat{\mathbf{x}}_v$ and non-singular covariance matrix $\mathbf{\Sigma}_v$, this algorithm computes the distribution function of the quadratic form $(\mathbf{x} + \mathbf{a})^T \mathbf{V}^{-1} (\mathbf{x} + \mathbf{a})$ for a fixed vector $\mathbf{a}$ and symmetric positive definite matrix $\mathbf{V}$. The quadratic form is expressed as an infinite series in central $\chi^2$ distribution functions: both the distribution functions and the series coefficients are evaluated recursively. The Matlab used implementation is available at the following location: http://www.math.wsu.edu/faculty/genz/homepage.

The AS106 algorithm can also be used when considering an elliptical object rather than a circular one (in this case $\mathbf{E} \neq \mathbf{I_d}$). This algorithm is very fast: the average computing time is less than 0.1 millisecond on our Pentium IV processor (2Ghz). Unfortunately, the integral computation has no related work if we consider 2 objects with Gaussian uncertainties as described below.

## V. Probability of collision between any 2 objects with Gaussian uncertainties

### A. Analytical description of the problem

The probability of collision between a $v$ and an $o$ polygonal object is the probability that $v$ and $o$ share a same part of the space. Consequently, given an $\hat{\mathbf{x}}_v$ configuration (with a $p_v$ associated pdf and a $\mathcal{V}_v$ volume) and an $\hat{\mathbf{x}}_o$ configuration (with a $p_o$ associated pdf and a $\mathcal{V}_o$ volume) the probability of collision is given by:

**Algorithm 1** Probability of collision between $v$ and $o$

```
1: function ProbabilityOfCollision(v, o)
2:     P_coll(v, o) ← 0
3:     for j ←1 to N do
4:         x_v ← randc(x̂_v, Σ_v)
5:         x_o ← randc(x̂_o, Σ_o)
6:         if V_v(x_v) ∩ V_o(x_o) ≠ ∅ then
7:             P_coll(v, o) ← P_coll(v, o) + 1
8:         end if
9:     end for
10:    P_coll(v, o) ← P_coll(v, o) / N
11:    return(P_coll(v, o))
12: end function
```

$$\mathbb{P}_{coll}(v,o) = \int_D p_v(x_v, y_v, \theta_v) \cdot \quad (12)$$
$$p_o(x_o, y_o, \theta_o) \cdot dx_v dy_v d\theta_v dx_o dy_o d\theta_o$$

with

$$D = \{ \ (x_v, y_v, \theta_v, x_o, y_o, \theta_o) \in \mathbb{R}^6$$
$$\backslash \ \mathcal{V}_v(x_v, y_v, \theta_v) \cap \mathcal{V}_o(x_o, y_o, \theta_o) \neq \emptyset\} \quad (13)$$

If $v$ and $o$ are punctual objects then Eq. (12) turns to Eq. (5). If $v$ and/or $o$ have an infinite volume then $v$ and $o$ always collide and Eq. (12) equals one. If $o$ is a well known circular object and $v$ a circular object with Gaussian uncertainties then (12) turns to Eq. 10.

### B. Monte Carlo solution

As we have no analytical solution to Eq. (12), we propose to use a MC (Monte Carlo) method.

First, we need to rewrite Eq. (12) as :

$$\mathbb{P}_{coll}(v,o) = \int_{\mathbb{R}^6} \Upsilon \cdot p_v(x_v, y_v, \theta_v) \cdot p_o(x_o, y_o, \theta_o)$$
$$\cdot dx_v dy_v d\theta_v dx_o dy_o d\theta_o \quad (14)$$

Where $\Upsilon = \Upsilon(\mathcal{V}_v(x_v, y_v, \theta_v), \mathcal{V}_o(x_o, y_o, \theta_o))$ is a collision test between the volume $\mathcal{V}_v$ of the vehicle and the volume $\mathcal{V}_o$ of the obstacle:

$$\Upsilon(\mathcal{V}_v(x_v, y_v, \theta_v), \mathcal{V}_o(x_o, y_o, \theta_o)) =$$
$$\begin{cases} 1 & \text{if } \mathcal{V}_v(x_v, y_v, \theta_v) \cap \mathcal{V}_o(x_o, y_o, \theta_o) \neq \emptyset \\ 0 & \text{if } \mathcal{V}_v(x_v, y_v, \theta_v) \cap \mathcal{V}_o(x_o, y_o, \theta_o) = \emptyset \end{cases} \quad (15)$$

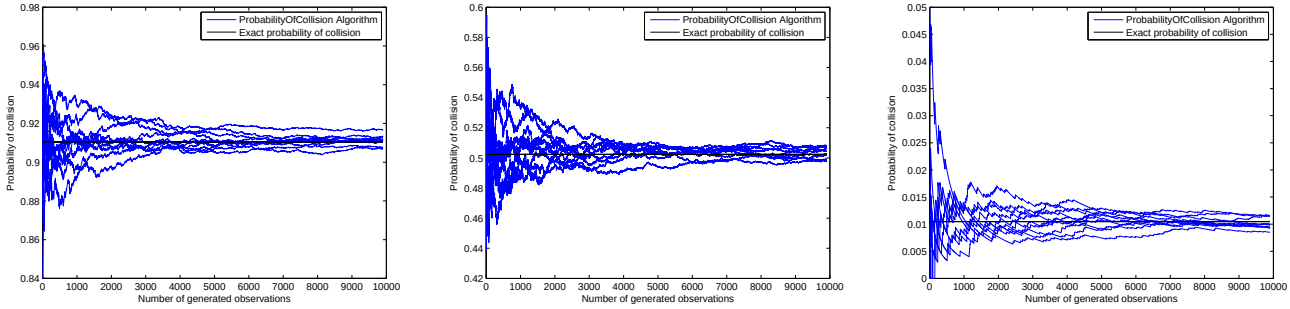
Denoting $z = (x_v, y_v, \theta_v, x_o, y_o, \theta_o)$ (and $Z$ the associated random variable), Eq. (14) can be rewritten as:

$$\mathbb{P}_{coll}(v,o) = \int_{\mathbb{R}^6} \Upsilon(z) f(z) dz, \quad (16)$$

with $f(z) = p_v(x_v, y_v, \theta_v) \cdot p_o(x_o, y_o, \theta_o)$ and $\Upsilon(z) = \Upsilon(\mathcal{V}_v(x_v, y_v, \theta_v), \mathcal{V}_o(x_o, y_o, \theta_o))$.

Secondly we know [14] that for evaluating the integral

$$\int_{\mathbb{R}^6} \Upsilon(z) f(z) dz = \mathbb{E}_f [\Upsilon(Z)] \quad (17)$$




(a) High probability (b) Medium probability (c) Low probability

Fig. 2. Result of the proposed algorithm for an high, a medium and a low probability of collision

we can use a sample $(z_1, ..., z_m)$ generated from the density $f$. Therefore, Eq. (17) can be approximated by the empirical average

$$\bar{\Upsilon}_m = \frac{1}{m}\sum_{j=1}^{m} \Upsilon(z_j), \tag{18}$$

since $\bar{\Upsilon}_m$ converges almost surely to $\mathbb{E}_f[\Upsilon(Z)]$ by the Strong Law of Large Numbers. Variable $z$ is Gaussian as $(x_v, y_v, \theta_v)$ and $(x_o, y_o, \theta_o)$ are both Gaussian and independent. Therefore, the sample $(z_1, ..., z_n)$ can be obtained by generating separately $(x_{v_j}, y_{v_j}, \theta_{v_j}) \sim p_v(x_v, y_v, \theta_v)$ and $(x_{o_j}, y_{o_j}, \theta_{o_j}) \sim p_o(x_o, y_o, \theta_o)$.

Thanks to Eq. (18) we can rewrite Eq. (14) as:

$$\mathbb{P}_{coll}(v, o) = \frac{1}{m}\sum_{j=1}^{m} \Upsilon(\mathcal{V}_v(x_{v_j}, y_{v_j}, \theta_{v_j}), \mathcal{V}_o(x_{o_j}, y_{o_j}, \theta_{o_j})). \tag{19}$$

Drawing samples $(x_v, y_v, \theta_v)$ and $(x_o, y_o, \theta_o)$ is done as explained in appendix B. We assume that such drawing is done by an existing randc() function (many excellent generators exist to do such a job).
The Eq. (19) leads to algorithm 1 with a linear complexity in $O(N)$ where $N = m$ is a number that determines the accuracy of the integral computation.

### C. Experimental results

Results provided by algorithm 1 for different values of probability of collision have been shown on Fig. 2. The algorithm has approximated 3 different values of probability of collision : a high (Fig. 2.a), a medium (Fig. 2.b) and a low value (Fig. 2.c). Both vehicle and obstacles were represented as polygonal lines for the collision test inside the probabilistic collision test.

For each value the algorithm has been ran 10 times (although one run is sufficient to obtain an estimate of the probability) with until $10^4$ samples for each run ($N = 10^4$ in algorithm 1). Consequently the subfigures of Fig. 2 have 10 curves (plus the line of the exact value) and we can analyze various results of the algorithm on three typical situations. For each figure the "exact probability" corresponds to the mean after $10^6$ samples. On each subfigure the algorithm defines a corridor around the exact value.

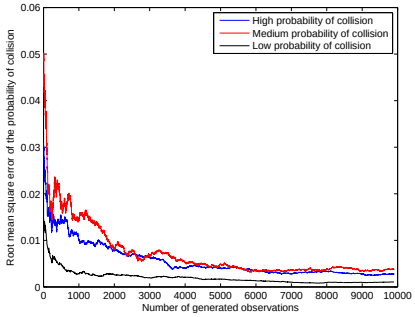



Fig. 3. Comparison of errors for an high, a medium and a low probability of collision

The root mean square error (RMSE) is computed on Fig. 3 from the 3 subfigures of Fig. 2. The RMSE value is acceptable for the high and the medium probability values (less than 0.02 for 0.91 and 0.5 mean values after 1000 generated observations). Nevertheless the RMSE is big regarding the low probability value (less than 0.005 for 0.01 mean value after 1000 generated observations).

The proposed algorithm computes the probability of collision with $10^4$ samples (one run) in about 0.01 second on a Pentium IV processor (2 Ghz). Approximating the probability of collision takes 1 millisecond if we consider that only $10^3$ samples are necessary. Consequently such algorithm can be embedded on a vehicle and deals with multiple obstacles as described in the next section.

## VI. Probability of collision between an object and other objects with Gaussian uncertainties

### A. Analytical description of the problem

The probability that $v$ collides with at least one obstacle can be calculated through the probability that $v$ does not collide with any obstacles :

$$\mathbb{P}_{coll}(v, o_1...o_n) = 1 - \overline{\mathbb{P}_{coll}(v, o_1)} \cdot ... \cdot \overline{\mathbb{P}_{coll}(v, o_n)} \tag{20}$$

The probability that $v$ do *not* collide with $o_i$ obstacle is

$$\overline{\mathbb{P}_{coll}(v,o_i)} = \int_{\mathbb{R}^6} \overline{\Upsilon} \cdot p_v(x_v, y_v, \theta_v) \cdot p_{o_i}(x_{o_i}, y_{o_i}, \theta_{o_i}) \, dx_v dy_v d\theta_v dx_o dy_o d\theta_o \quad (21)$$

with $\overline{\Upsilon} = \overline{\Upsilon(\mathcal{V}_v(x_v, y_v, \theta_v), \mathcal{V}_{o_i}(x_{o_i}, y_{o_i}, \theta_{o_i}))}$.

### B. Monte Carlo solution

Using Eq.s (20) and (21) (Eq. 21 being computed via a MC algorithm like algorithm 1) leads to algorithm 2 which is explained beneath.

- Lines 2-4: The probability $\overline{\mathbb{P}_{coll}(v,o_i)}$ that the $v$ vehicle does not collide with the $o_i$ obstacle is initialized to 0 for each of the $n$ obstacles.
- Lines 5 and 13: This first loop activates the computation of $\overline{\mathbb{P}_{coll}(v,o_i)}$ for each of the $n$ obstacles.
- Lines 6 and 12: This second loop computes $\overline{\mathbb{P}_{coll}(v,o_i)}$ using a new pair of samples at each iteration. The bigger the number $N$ of samples the better the accuracy.
- Lines 7 and 8: A $\mathbf{x}_v$ (respectively $\mathbf{x}_o$) sample is drawn following the pdf of the vehicle (respectively the $i$ obstacle).
- Lines 9-11: If the volume of the vehicle on $\mathbf{x}_v$ does not intersect the volume of the $i$ obstacle on $\mathbf{x}_o$ then the variable $\overline{\mathbb{P}_{coll}(v,o_i)}$ rises in increment of 1. The division of $\overline{\mathbb{P}_{coll}(v,o_i)}$ by the number of samples corresponds to the probability that the $v$ vehicle and the $o_i$ obstacle do not collide.
- Line 14 : The probability $\overline{\mathbb{P}_{coll}(v,o_{1..n})}$ that the vehicle does not collide with the obstacles is initialized to 1.
- Lines 15-17: $\overline{\mathbb{P}_{coll}(v,o_{1..n})}$ is updated according to $\overline{\mathbb{P}_{coll}(v,o_i)}$ (a variable which is proportional to the probability of collision with each of the obstacles).
- Line 18: The probability $\mathbb{P}_{coll}(v,o_1...o_n)$ that the vehicle collides with the obstacles is computed. $\overline{\mathbb{P}_{coll}(v,o_{1..n})}$ is divided $n$ times by $N$ where $N$ is the number of samples and $n$ is the number of obstacles.
- Line 19 returns the result.

The complexity of this algorithm is O($nN$) which is $n$ times the complexity of algorithm 1 This is verified by experimental results (with $N = 10^3$) where the computing time is $n$ milliseconds.

## VII. Conclusion

We have defined the probability of collision for a vehicle in a cluttered environment as a product of integrals of a product of Gaussians. The probability of collision takes into account the uncertainties and the volume of both vehicle and obstacles. We have proposed a Monte Carlo method to compute the integrals because we do not have analytical solution. Experimental results show the soundness of our algorithm which can deal with any shape (and modelling) of both vehicle and obstacles. As our algorithm relies on a classical geometrical collision test, it can easily replace any classical collision test by a probabilistic one.

**Algorithm 2** Probability of collision between $v$ and $o_1, ..., o_n$

```
1:  function ProbabilityOfCollisions(v, o_1, ..., o_n)
2:      for i ←1 to n do
3:          P̄_coll(v, o_i) ← 0
4:      end for
5:      for i ←1 to n do
6:          for j ←1 to N do
7:              x_v ← randc(x̂_v, Σ_v)
8:              x_o ← randc(x̂_{o_i}, Σ_{o_i})
9:              if V_v(x_v) ∩ V_{o_i}(x_o) = ∅ then
10:                 P̄_coll(v, o_i) ← P̄_coll(v, o_i) + 1
11:             end if
12:         end for
13:     end for
14:     P̄_coll(v, o_{1..n}) ← 1
15:     for i ←1 to n do
16:         P̄_coll(v, o_{1..n}) ← P̄_coll(v, o_{1..n}) · P̄_coll(v, o_i)
17:     end for
18:     P_coll(v, o_1...o_n) ←1 − P̄_coll(v,o_{1..n}) / N^n
19:     return(P_coll(v, o_1...o_n))
20: end function
```

## References


[1] H. Hu, M. Brady, and P. Probert, "Navigation and control of a mobile robot among moving obstacles," in *Proc. of the 30th IEEE International Conference on Decision and Control (CDC'91)*, Brighton, England, 11-13 Dec 1991, pp. 698–703.

[2] R. Zapata, P. Lepinay, and P. Thompson, "Reactive behaviors of fast mobile robots," *Journal of Robotic Systems*, vol. 11, pp. 13–20, 1994.

[3] L. Codewener and D. Meizel, "On line speed monitoring of mobile robots tasks," *Engineering applications of artificial intelligence*, vol. 7(2), pp. 152–160, 1994.

[4] A. Lambert and N. LeFort-Piat, "Safe task planning integrating uncertainties and local maps federation," *International Journal of Robotics Research*, vol. 19(6), pp. 597–611, 2000.

[5] J. Miura, Y. Negishi, and Y. Shirai, "Adaptive robot speed control by considering map and motion uncertainty," *Robotics and Autonomous Systems*, vol. 54(2), pp. 110–117, 2006.

[6] K. M. Krishna, R. Alami, and T. Simeon, "Safe proactive plans and their execution," *Robotics and Autonomous Systems*, vol. 54, pp. 244–255, 2006.

[7] P. Burlina, D. DeMenthon, and L. Davis, "Navigation with uncertainty: reaching a goal in a high collision risk region," in *Proc. of the IEEE International Conference on Robotics and Automation*, Nice, France, 12-14 May 1992, pp. 2440–2445 vol.3.

[8] J. V. Terza and U. Welland, "A comparison of bivariate normal algorithms," *Journal of Statist. Comput. Simul.*, vol. 19, pp. 115–127, 1988.

[9] M. Patefield and D. Tandy, "Fast and accurate computation of owen's t-function," *Journal of Statistical Software*, vol. 5 (5), 2000. [Online]. Available: http://www.jstatsoft.org

[10] A. Genz, "Numerical computation of multivariate normal probabilities," *J. Comput. Graph. Statist.*, vol. 1, pp. 141–150, 1992.

[11] M. Schervish, "Multivariate normal probabilities with error bound," *Applied Statistics*, vol. 33, pp. 81–87, 1984.

[12] J. Berntsen, T. O. Espelid, and A. Genz, "Algorithm 698: Dcuhre-an adaptive multidimensional integration routine for a vector of integrals," *ACM Transactions on Mathematical Software*, vol. 17, pp. 452–456, 1991.

[13] J. Sheil and I. O'Muircheartaigh, "Algorithm as 106: The distribution of non-negative quadratic forms in normal variables," *Applied Statistics*, vol. 26, pp. 92–98, 1977.

[14] C.P. Robert and G. Casella, *Monte Carlo statistical methods*. Springer Verlag, 2004.

[15] R.Y. Rubinstein, *Simulation and the Monte Carlo method*. John Wiley & Sons, 1981.
[16] G. Box and M. Muller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, vol. 29(2), pp. 610–611, 1958.

## APPENDIX

### A. Multivariate Gaussian distribution on $R^d$

$$\mathcal{N}_d(x; m, \Sigma) \sim \frac{\exp\left(-\frac{1}{2}(x-m)'\Sigma^{-1}(x-m)\right)}{\sqrt{(2\pi)^d \det(\Sigma)}},$$

with $m \in \mathbb{R}^d$, and $\Sigma$ a symmetric, positive semi-definite square matrix of dimension $d$.

Let $X \sim \mathcal{N}_d(m_1, \Sigma_1)$ and $Y \sim \mathcal{N}_d(m_2, \Sigma_2)$ where $\mathcal{N}_d(m, \Sigma)$ denotes the d-dimensional Gaussian distribution with mean $m$ and covariance $\Sigma$, with density $\mathcal{N}_d(x; m, \Sigma)$. We need to evaluate the following integral :

$$\int \mathcal{N}_d(x; m_1, \Sigma_1)\, \mathcal{N}_d(x; m_2, \Sigma_2)\, dx$$

where $x \in \mathbb{R}^d$. This should be written

$$\int \frac{\exp\left(-\frac{1}{2}[\alpha_1 + \alpha_2]\right)}{(2\pi)^d \sqrt{\det(\Sigma_1)}\sqrt{\det(\Sigma_2)}} dx$$

with $\alpha_1 = (x-m_1)'\Sigma_1^{-1}(x-m_1)$
$\alpha_2 = (x-m_2)'\Sigma_2^{-1}(x-m_2)$

Let's study the term $\int \exp\left(-\frac{1}{2}(A)\right) dx$ where

$$A = (x-m_1)'\Sigma_1^{-1}(x-m_1) + (x-m_2)'\Sigma_2^{-1}(x-m_2).$$

One can show that

$$A = B + m_1'\Sigma_1^{-1}m_1 + m_2'\Sigma_2^{-1}m_2 - \left(\Sigma_1^{-1}m_1 + \Sigma_2^{-1}m_2\right)'\left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1}\left(\Sigma_1^{-1}m_1 + \Sigma_2^{-1}m_2\right).$$

With

$$B = [x-m]'\Sigma^{-1}[x-m],$$

and

$$m = \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1}\left(\Sigma_1^{-1}m_1 + \Sigma_2^{-1}m_2\right)$$
$$\Sigma^{-1} = \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right).$$

Therefore

$$\int \frac{\exp\left(-\frac{1}{2}A\right)}{(2\pi)^d \sqrt{\det(\Sigma_1)}\sqrt{\det(\Sigma_2)}} dx = \frac{\exp\left[-\frac{1}{2}(A-B)\right]}{(2\pi)^d \sqrt{\det(\Sigma_1)}\sqrt{\det(\Sigma_2)}} \times \int \exp\left(-\frac{1}{2}B\right) dx.$$

With

$$\int \exp\left(-\frac{1}{2}B\right) dx = \sqrt{(2\pi)^d}\sqrt{\det(\Sigma)} \int \frac{\exp\left(-\frac{1}{2}(x-m)'\Sigma^{-1}(x-m)\right)}{\sqrt{(2\pi)^d}\sqrt{\det(\Sigma)}} dx$$
$$= \sqrt{(2\pi)^d}\sqrt{\det(\Sigma)}$$

### B. Generating Gaussian pseudo-random numbers

Most authors (see [15]) use a classical random number function (that returns a number from an uniform distribution in the range from 0 to 1) in order to generate Gaussian pseudo-random numbers.

*1) Generating Gaussian numbers with zero mean and a standard deviation of one:*

*a) The Box-Muller transformation:* The most poupular ways of generating Gaussian pseudo-random numbers with zero mean and a standard deviation of one (see [15]) is the famous Box-Muller [16] transformation. It allows us to transform uniformly distributed random variables, to a new set of random variables with a Gaussian (or Normal) distribution. The most basic form of the transformation looks like:

$$y_1 = \sqrt{-2ln(x_1)} \times \cos(2\pi x_2)$$
$$y_2 = \sqrt{-2ln(x_1)} \times \sin(2\pi x_2)$$

We start with two independent random numbers: $x_1$=rand() and $x_2$=rand(). Then apply the above transformations to get two new independent random numbers which have a Gaussian distribution with zero mean and a standard deviation of one. This particular form of the transformation has two problems with it : it is slow because of many calls to the math library and it can have numerical stability problems when $x_1$ is very close to zero.

*b) The polar form of the Box-Muller transformation:* The polar form of the Box-Muller transformation is both faster and more robust numerically then the Box-Muller transformation. The algorithmic description of it is:

```
do {
     x1 = 2.0 * rand() - 1.0;
     x2 = 2.0 * rand() - 1.0;
     w = x1 * x1 + x2 * x2;
} while ( w >= 1.0 );
w = sqrt( (-2.0 * ln( w ) ) / w );
y1 = x1 * w;
y2 = x2 * w;
```

The polar form is faster because it does the equivalent of the sine and cosine geometrically without a call to the trigonometric function library. But because of the possibility of many calls to rand(), the uniform random number generator should be fast. That's why we recommend the use of the R250 algorithm.

*2) Random multivariate normal numbers:* The random multivariate normal numbers are produced by multiplying a vector of random univariate normal numbers by the Cholesky decomposition of the correlation matrix according to the formula: $Y = LX$ where

- $Y$ = a vector of random multivariate normal numbers
- $L$ = the Cholesky decomposition of the correlation matrix.
- $X$ = a vector of random univariate normal numbers

Here the Cholesky decomposition is stored in the lower triangle and main diagonal of a square matrix; elements in the upper triangle of the matrix are 0.