A Fast Monte Carlo Algorithm for Collision Probability Estimation

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

Dominique Gruyer, Guillaume Saint Pierre
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 modeling. The probability of collision is computed from a sum of integral of a product of Gaussians. The integrals takes into account the uncertain configurations and the volume of both the vehicle and the obstacles.**

I. Introduction

The anticipation of 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 way 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 judgment 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 judgment. [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. The farthest is the collision and the highest is the 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 so as to compute a time to collision. [4] computes a distance to undecided regions (unknown region) or to nearby obstacles. Next they use this distance information to compute a speed.

We think that only using a measured distance to collision [4][5] is not sufficient as the real distance could be quite different and lead to unexpected collision. 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]) 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%. It is a shame to define a threshold and to 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 a probability of collision. Such an approach have been followed in [6] 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. That is why we propose an algorithm in section 4. In section 5 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 θ_v is its orientation. All variables are defined with respect to the global frame.

The obstacles configuration is denoted $\mathbf{x_o} = (x_o, y_o, \theta_o)^T$. Both vehicle and obstacles are represented as polygonal lines in a 2D world map and their surfaces 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 $\mathbf{x}$ having a $\mathbf{C}$ covariance matrix and an $\hat{\mathbf{x}}$ mean is :

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

where n is the $\mathbf{x}$ dimension. We consider that $\mathbf{x}$ dimension is 3 for the sake of simplicity : $\mathbf{x} = (x, y, \theta)^{\mathrm{T}}$(nevertheless the $\mathbf{x}$ dimension can be higher).

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

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

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

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

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

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 :

$$p_{coll}(v,o) = \iiint_{\mathbb{R}^2\times[0,2\pi[} p_v(x,y,\theta)\,.\,p_o(x,y,\theta).dxdyd\theta \tag{5}$$

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 $\mathcal{N}_d\left(\widehat{\mathbf{x}}_v, \mathbf{\Sigma}_v\right)$ distribution, and $p_o(x,y,\theta)$ the density of a $\mathcal{N}_d\left(\widehat{\mathbf{x}}_o, \mathbf{\Sigma}_o\right)$. Let us denote $\mathbf{x} = (x,y,\theta)$, $\mathbf{A} = \left(\mathbf{x}-\widehat{\mathbf{x}}_v\right)'\mathbf{\Sigma}_v^{-1}\left(\mathbf{x}-\widehat{\mathbf{x}}_v\right) + \left(\mathbf{x}-\widehat{\mathbf{x}}_o\right)'\mathbf{\Sigma}_o^{-1}\left(\mathbf{x}-\widehat{\mathbf{x}}_o\right)$ and $\mathbf{B} = \left[\mathbf{x}-\mathbf{m}\right]'\mathbf{\Sigma}^{-1}\left[\mathbf{x}-\mathbf{m}\right]$ with:

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

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

We have

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

with

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

Equation (8) has been used to compute the probability of collision of figure 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 other way (left part of the figure). The vehicle was moving from the bottom to the top of the figure in 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 of both the obstacle (0.16) and the vehicle (0.13). Consequently the pdf of the collision has been multiplied by 100 on figure 1 for a better visualization. The probability of collision computed using equation (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 distance between the vehicles. We have not investigate the interest of equation (8) when the volume are not null because the following approach (see the next section) provide good enough results both in term of computing time and precision.

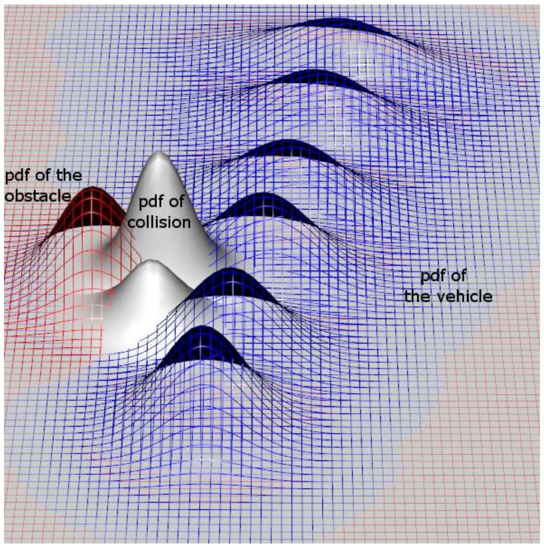


Figure 1. pdf of the collision between an obstacle and a vehicle moving in straight line

IV. Probability of collision between 2 objects with Gaussian uncertainties

A. Analytical description of the problem

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

$$\begin{aligned}p_{coll}(v,o) = \int_D\ & p_v(x_v,y_v,\theta_v). \\ & p_o(x_o,y_o,\theta_o)dx_vdy_vd\theta_vdx_ody_od\theta_o\end{aligned} \tag{10}$$

with

$$\begin{aligned}D = \{\ & (x_v,y_v,x_o,y_o)\in\mathbb{R}^4, (\theta_v,\theta_o)\in[0,2\pi[^2 \\ & \setminus \mathcal{V}_v(x_v,y_v,\theta_v)\cap\mathcal{V}_o(x_o,y_o,\theta_o)\neq\emptyset\}\end{aligned} \tag{11}$$

If v and o are punctual objects then equation 10 turn to equation 5. If v and/or o have an infinite volume then v and o always collide and equation 10 equals one.

Algorithm 1 Probability of collision between v and o

```
1:  function SlowProbabilityOfCollision($v, o$)
2:      $p_{coll}(v,o) \leftarrow 0$
3:      for $i \leftarrow 1$ to $N$ do
4:          $\mathbf{x}_v \leftarrow randc(\widehat{\mathbf{x}}_v, \mathbf{\Sigma}_v)$
5:          for $j \leftarrow 1$ to $N$ do
6:              $\mathbf{x}_o \leftarrow randc(\widehat{\mathbf{x}}_o, \mathbf{\Sigma}_o)$
7:              if $\mathcal{V}_v(\mathbf{x}_v) \cap \mathcal{V}_o(\mathbf{x}_o) \neq \emptyset$ then
8:                  $p_{coll}(v,o) \leftarrow p_{coll}(v,o) + 1$
9:              end if
10:         end for
11:     end for
12:     $p_{coll}(v,o) \leftarrow \frac{p_{coll}(v,o)}{N^2}$
13:     return($p_{coll}(v,o)$)
14: end function
```

Algorithm 2 Faster computing of the probability of collision between v and o

```
1:  function FastProbabilityOfCollision($v, o$)
2:      $p_{coll}(v,o) \leftarrow 0$
3:      for $j \leftarrow 1$ to $N$ do
4:          $\mathbf{x}_v \leftarrow randc(\widehat{\mathbf{x}}_v, \mathbf{\Sigma}_v)$
5:          $\mathbf{x}_o \leftarrow randc(\widehat{\mathbf{x}}_o, \mathbf{\Sigma}_o)$
6:          if $\mathcal{V}_v(\mathbf{x}_v) \cap \mathcal{V}_o(\mathbf{x}_o) \neq \emptyset$ then
7:              $p_{coll}(v,o) \leftarrow p_{coll}(v,o) + 1$
8:          end if
9:      end for
10:     $p_{coll}(v,o) \leftarrow \frac{p_{coll}(v,o)}{N}$
11:     return($p_{coll}(v,o)$)
12: end function
```

B. Monte Carlo solution

As we have no analytical solution to equation 10, we propose to use a MC (Monte Carlo) method.

Firstly, we need to rewrite Eq. (10) as :

$$p_{coll}(v,o) = \int_0^{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}[p_v(x_v,y_v,\theta_v). \int_0^{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p_o(x_o,y_o,\theta_o). \Upsilon.dx_o dy_o d\theta_o].dx_v dy_v d\theta_v \quad (12)$$

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:

$$\Upsilon(\mathcal{V}_v(x_v,y_v,\theta_v),\mathcal{V}_o(x_o,y_o,\theta_o)) = \begin{cases} 1 & \text{if } \mathcal{V}_v(x_v,y_v,\theta_v) \cap \mathcal{V}_o(x_o,y_o,\theta_o) \neq \emptyset \\ 0 & \text{if } \mathcal{V}_v(x_v,y_v,\theta_v) \cap \mathcal{V}_o(x_o,y_o,\theta_o) = \emptyset \end{cases} \quad (13)$$

Secondly we know that we can approximate the integral of the product of two functions f and h by :

$$\int f(y)h(y)dy = \frac{1}{m}\sum_{j=1}^{m} h(y_j) \quad (14)$$

Inside this Monte Carlo approximation, $(y_1,...,y_m)$ are samples generated by following the pdf $f(y)$.

Thanks to Eq. (14) we can rewrite the right part of Eq. (12) as:

$$\int_0^{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p_o(x_o,y_o,\theta_o).\Upsilon.dx_o dy_o d\theta_o = \frac{1}{m}\sum_{j=1}^{m}\Upsilon(\mathcal{V}(x_v,y_v,\theta_v),\mathcal{V}(x_{o_j},y_{o_j},\theta_{o_j})) \quad (15)$$

and using Eq. (14) and (15) allows us to rewrite Eq. (12) as:

$$p_{coll}(v,o) = \frac{\sum_{i=1}^{m}}{m}\frac{\sum_{j=1}^{m}}{m}\Upsilon\left(\mathcal{V}(x_{v_i},y_{v_i},\theta_{v_i}),\mathcal{V}(x_{o_j},y_{o_j},\theta_{o_j})\right) \quad (16)$$

Which can be rewritten as:

$$p_{coll}(v,o) = \frac{1}{m^2}\sum_{i,j=1}^{m}\Upsilon\left(\mathcal{V}(x_{v_i},y_{v_i},\theta_{v_i}),\mathcal{V}(x_{o_j},y_{o_j},\theta_{o_j})\right) \quad (17)$$

Where $((x_{v_1},y_{v_1},\theta_{v_1}),...,(x_{v_m},y_{v_m},\theta_{v_m}))$ and $((x_{o_1},y_{o_1},\theta_{o_1}),...,(x_{o_m},y_{o_m},\theta_{o_m}))$ are samples generated by following the pdf $p_v(x_v,y_v,\theta_v)$ and $p_o(x_o,y_o,\theta_o)$. Drawing of samples is done as explained in appendix B. We assume that such a drawing is done by a randc() function.

Algorithm 1 is the implementation of equation 17. The complexity of this algorithm is $0(N^2)$ where $N = m$ is a number that determine the accuracy of the integral computation. Unfortunately we need a value of N of the order of 10^4 to obtain "good results" (see figures 2, 3, 4 and section IV-C). Consequently we propose to rewrite Eq. (17) as :

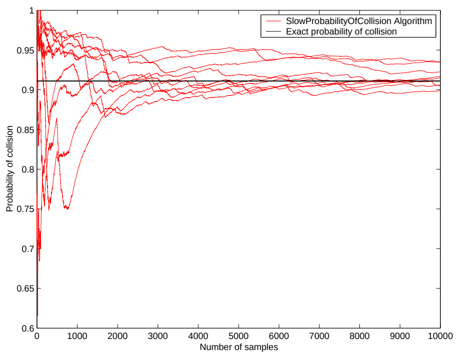
$$p_{coll}(v,o) = \frac{1}{m}\sum_{i=1}^{m}\Upsilon\left(\mathcal{V}(x_{v_i},y_{v_i},\theta_{v_i}),\mathcal{V}(x_{o_i},y_{o_i},\theta_{o_i})\right) \quad (18)$$

The sum of sum has been replaced by a single sum and the samples are now drawn simultaneously (in Eq. (17) we drawn m samples of o for 1 sample of v). The equation (18) leads to algorithm 2 with a linear complexity in $O(N)$ which is much more better than algorithm 1. Unfortunately we need an higher number for N (in algorithm 2 comparatively to algorithm 1) in order to achieve a good accuracy. Nevertheless we are going to see in the next section that the N value needed in algorithm 2 is very inferior to N^2 value needed in algorithm 1.

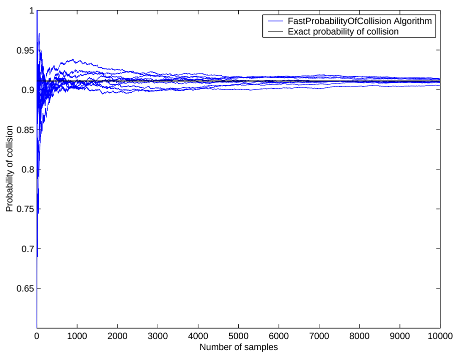
C. Experimental results

Results provided by algorithms 1 and 2 have been compared on figures 2, 3 and 4. The two algorithms have approximated 3 different values of the probability of collision : an high (figure 2), an medium (figure3) and a low value (figure4).

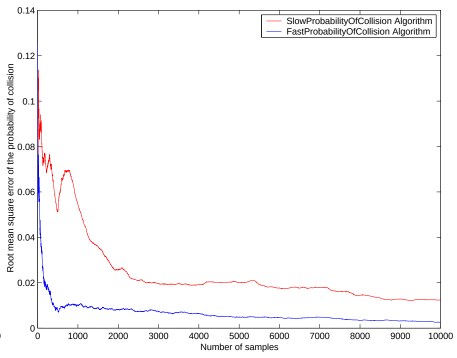
For each value the algorithms have been ran 10 times (although one run is sufficient to obtain an estimate of the



(a) "Slow Probability Of Collision" algorithm

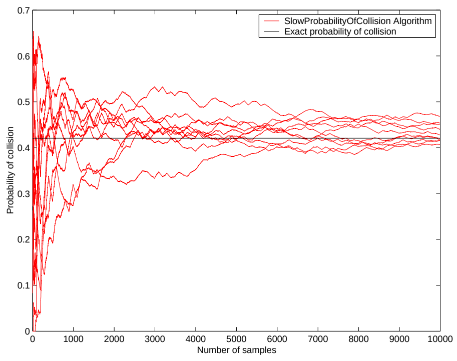


(b) "Fast Probability Of Collision" algorithm

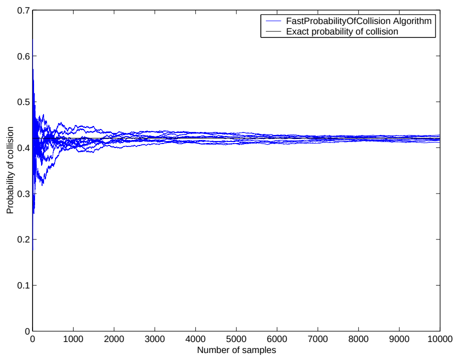


(c) Comparison of errors between both algorithms

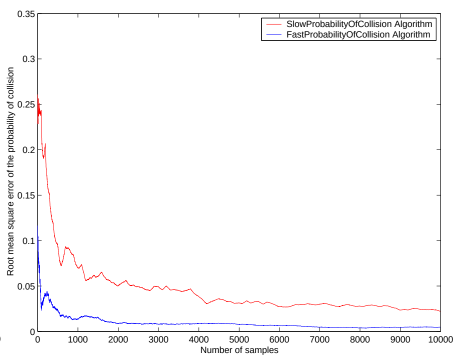
Figure 2. Comparison between the Slow and Fast algorithms for a high probability of collision



(a) "Slow Probability Of Collision" algorithm



(b) "Fast Probability Of Collision" algorithm



(c) Comparison of errors between both algorithms

Figure 3. Comparison between the Slow and Fast algorithms for a medium probability of collision

probability) with until 10^4 samples for each run ($N = 100$ in algorithm 1 and $N = 10^4$ in algorithm 2). Consequently subfigures .a and .b have 10 curves (plus the line of the exact value) and we can analyze various results of the algorithms on three typical situations. For each figure the "exact probability" is the mean between the two algorithms after 10^6 samples.

On each figure the slow algorithm defines a large corridor around the exact value whereas the fast algorithm defines a thinner corridor. It shows the property of the fast algorithm to define (in a lower time) better and more regular results than the slow algorithm. It is underlined on figures 2.c, 3.c and 4.c where the root mean square error (RMSE) is computed for both algorithms. RMSE is always lower (2 to 10 times after more than 500 samples) for the fast algorithm than for the slow algorithm on each of our attempts. For high and medium probability of collision, the accuracy (maximum error) after 1000 samples is about 0.15 for the slow algorithm whereas it is less than 0.05 for the fast algorithm. Considering the low (0.011) probability of collision (figure 4), the accuracy is 0.03 for the slow algorithm and 0.005 for the fast algorithm.

Both algorithms compute the probability of collision with 10^4 samples (one run) in about 0.01 second on a Pentium IV processor (2 Ghz). Consequently, the fast algorithm outperforms the slow algorithm considering both computing time and precision.

Approximating the probability of collision takes 1 millisecond if we consider that only 10^3 samples are necessary (for algorithm 2). Consequently such an algorithm can be embedded on a vehicle and deals with multiple obstacles as described in the next section.

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

A. Analytical description of the problem

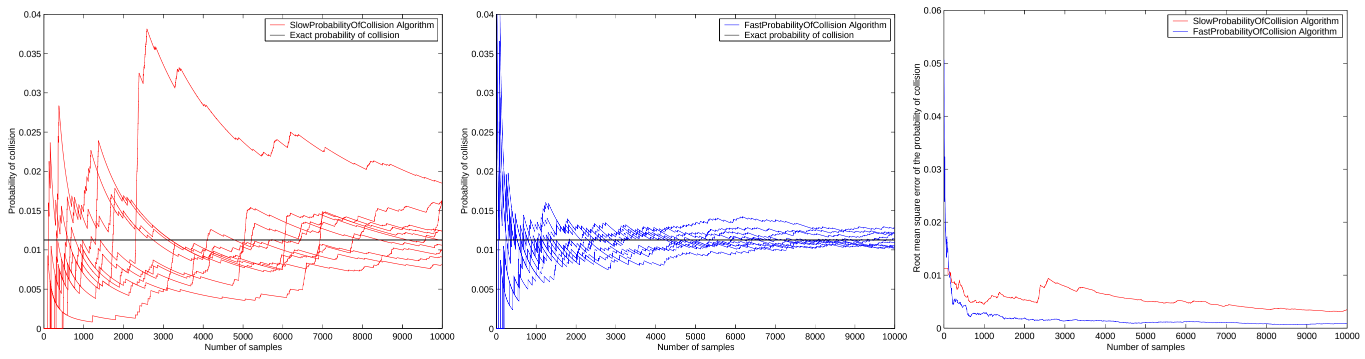
The probability that v collides with at least one obstacle can be calculated through the probability that v do not collides with any obstacles :

$$p_{coll}(v, o_1 ... o_n) = 1 - \overline{p_{coll}(v, o_1)} \cdot \ldots \cdot \overline{p_{coll}(v, o_n)} \quad (19)$$

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

$$\begin{aligned}\overline{p_{coll}(v, o_i)} = &\int_0^{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}[p_v(x_v, y_v, \theta_v).\\ &\int_0^{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p_{o_i}(x_{o_i}, y_{o_i}, \theta_{o_i}).\\ &\overline{\Upsilon}.dx_{o_i}dy_{o_i}d\theta_{o_i}].dx_v dy_v d\theta_v \quad (20)\end{aligned}$$

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



(a) "Slow Probability Of Collision" algorithm (b) "Fast Probability Of Collision" algorithm (c) Comparison of errors between both algorithms

Figure 4. Comparison between the Slow and Fast algorithms for a low probability of collision

B. Monte Carlo solution

Using equations 19 and 20 (equation 20 being computed via a MC algorithm like algorithm 2) leads to algorithm 3 which is explained beneath.

- Lines 2-4: The probability $\overline{p_{coll}(v, o_i)}$ 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 $\overline{p_{coll}(v, o_i)}$ for each of the n obstacles.
- Lines 6 and 12: This second loop computes $\overline{p_{coll}(v, o_i)}$ using a new pair of samples at each iteration. The bigger is the number N of samples the better is 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 put on $\mathbf{x}_v$ does not intersect the volume of the i obstacle put on $\mathbf{x}_o$ then the variable $\overline{p_{coll}(v, o_i)}$ rises in increment of 1. When $\overline{p_{coll}(v, o_i)}$ will be divided by the number of samples it corresponds to the probability that the v vehicle and the o_i obstacle does not collide.
- Line 14 : The probability $\overline{p_{coll}(v, o_{1..n})}$ that the vehicle does not collide with the obstacles is initialized to 1.
- Lines 15-17: $\overline{p_{coll}(v, o_{1..n})}$ is updated according to $\overline{p_{coll}(v, o_i)}$ (a variable which is proportional to the probability of collision with each of the obstacles).
- Line 18: The probability $p_{coll}(v, o_1 ... o_n)$ that the vehicle collides with the obstacles is computed. $\overline{p_{coll}(v, o_{1..n})}$ 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 $0(nN)$ which is n time the complexity of algorithm 2. This is verified by experimental results (with $N = 10^3$) where the computing time is n millisecond.

Algorithm 3 Probability of collision between v and $o_1, ..., o_n$

```
1: function ProbabilityOfCollisions($v, o_1, ..., o_n$)
2:     for $i \leftarrow 1$ to $n$ do
3:         $\overline{p_{coll}(v, o_i)} \leftarrow 0$
4:     end for
5:     for $i \leftarrow 1$ to $n$ do
6:         for $j \leftarrow 1$ to $N$ do
7:             $\mathbf{x}_v \leftarrow randc(\widehat{\mathbf{x}}_v, \mathbf{\Sigma}_v)$
8:             $\mathbf{x}_o \leftarrow randc(\widehat{\mathbf{x}}_{o_i}, \mathbf{\Sigma}_{o_i})$
9:             if $\mathcal{V}_v(\mathbf{x}_v) \cap \mathcal{V}_{o_i}(\mathbf{x}_o) = \emptyset$ then
10:                $\overline{p_{coll}(v, o_i)} \leftarrow \overline{p_{coll}(v, o_i)} + 1$
11:            end if
12:        end for
13:    end for
14:    $\overline{p_{coll}(v, o_{1..n})} \leftarrow 1$
15:    for $i \leftarrow 1$ to $n$ do
16:        $\overline{p_{coll}(v, o_{1..n})} \leftarrow \overline{p_{coll}(v, o_{1..n})} \cdot \overline{p_{coll}(v, o_i)}$
17:    end for
18:    $p_{coll}(v, o_1 ... o_n) \leftarrow 1 - \frac{\overline{p_{coll}(v, o_{1..n})}}{N^n}$
19:    return($p_{coll}(v, o_1 ... o_n)$)
20: end function
```

VI. Conclusion

We have defined the probability of collision for a vehicle in a cluttered environment as a sum of integral of a product of Gaussian. The integrals takes into account the uncertainties and the volume of both the vehicle and the obstacles. We have proposed a Monte Carlo method to compute the integrals because we have no analytical solution. Next we have enhanced this MC algorithm (in term of computing time and quality of result). Experimental results show the soundness of the enhanced algorithm.

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] 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.

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

[6] 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.

[7] R.Y. Rubinstein, *Simulation and the Monte Carlo method.* John Wiley & Sons, 1981.

[8] S. Kirkpatrick and E. Stoll, “A very fast shift-register sequence random number generator,” *Journal of Computational Physics*, vol. 40, pp. 517–526, 1981.

[9] G. Box and M. Muller, “A note on the generation of random normal deviates,” *Annals Math. Stat*, vol. 29, pp. 610–611, 1958.

APPENDIX

A. Multivariate Gaussian distribution on R^d

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

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

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

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

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

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

with $\begin{array}{l}\alpha_1 = \left(x-m_1\right)'\Sigma_1^{-1}\left(x-m_1\right)\\ \alpha_2 = \left(x-m_2\right)'\Sigma_2^{-1}\left(x-m_2\right)\end{array}$

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

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

One can show that

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

With

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

and

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

Therefore

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

With

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

B. Generating Gaussian pseudo-random numbers

Most authors (see [7]) use a classical random number function in order to generate Gaussian pseudo-random numbers. The name of such a function is rand() in most programming language; we made the same choice for the following sections. The rand() function returns a number from a uniform distribution in the range from 0 to 1. If it it is available, we recommend the use of the R250 algorithm as the rand() function for most applications ([8]).

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

a) The Box-Muller transformation: There are many ways of generating Gaussian pseudo-random numbers with zero mean and a standard deviation of one (see for example [7] for an extensive discussion of this topic) but we will only go into one important method here. The famous Box-Muller [9] transformation 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:

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

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.
- 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. Standard deviations are then multiplied and/or means added per the user specifications.