PROBABILITY-BASED OPTIMAL PATH PLANNING FOR TWO-WHEELED MOBILE ROBOTS

Jaeyeon Lee
Department of Electrical Engineering
University of Texas at Dallas
Richardson, Texas 75080
Email: jaeyeon.lee@utdallas.edu

Wooram Park
Department of Mechanical Engineering
University of Texas at Dallas
Richardson, Texas 75080
Email: wooram.park@utdallas.edu

ABSTRACT
Most dynamic systems show uncertainty in their behavior. Therefore, a deterministic model is not sufficient to predict the stochastic behavior of such systems. Alternatively, a stochastic model can be used for better analysis and simulation. By numerically integrating the stochastic differential equation or solving the Fokker-Planck equation, we can obtain a probability density function of the motion of the system. Based on this probability density function, the path-of-probability (POP) method for path planning has been developed and verified in simulation. However, there are rooms for more improvements and its practical implementation has not been performed yet. This paper concerns formulation, simulation and practical implementation of the path-of-probability for two-wheeled mobile robots. In this framework, we define a new cost function which measures the averaged targeting error using root-mean-square (RMS), and iteratively minimize it to find an optimal path with the lowest targeting error. The proposed algorithm is implemented and tested with a two-wheeled mobile robot for performance verification.

1 Introduction
In modeling dynamic systems, it is common to ignore the system uncertainty and use the deterministic model for motion analysis, control and simulation. The stochasticity of systems is easily ignored in many cases when employing the deterministic model even if there exists the stochasticity such as model errors and disturbances in actual world. There is no doubt that some amount of discrepancy between a theoretical model and an actual system is inevitable. One easy way to tackle this problem is to apply the feedback control, and a variety of control methods are used to compensate the associated errors. Nevertheless, it has been shown that we can include the stochastic effects in analysis and simulation of dynamic systems. This paper focuses on applying the stochastic effects into a path planning problem.

Along this line, the system uncertainty and stochasticity have been studied in many ways. The stochasticity was included in the theoretical model for better analysis as well as control in [1–3]. Latombe et al. [4] developed a motion planning in an environment with errors existing in control and sensing. Alterovitz et al. [5] proposed a new motion planning algorithm for a flexible medical needle with the consideration of system uncertainty. There exist a number of studies on path planning methods based on the stochastic model [6–11]. These methods generate paths for the system by searching the highest probability for the system to reach a target. In other words, they find a path to maximize the probability for the system to reach a target. In authors’ previous work [12], we firstly used a targeting error instead of probability to find the best configuration of robot manipulator. Inspired by the work in [12], this paper proposes a method to find an optimal path that minimizes the expected error of the mobile robot position to the target. Although the optimality in path planning can be defined in various ways, we define it using minimum of targeting errors in this paper.

Even though the path-of-probability (POP) method for path generation has been developed, improved and verified with computer simulation [6–11], there has not been implementation with actual hardware. In this paper, we solve the path planning prob-
lem for two-wheeled mobile robot with the targeting error estimation. Additionally, we apply two improvement ideas for the classical POP method and verify the algorithm with both computer simulation and actual experiments with iRobot Create.

This paper is organized as follows. In Section 2, we review the stochasticity-based modeling for the two-wheeled mobile robot. Two improvements of the classical POP for the path generation are proposed in Section 3. Multiple simulations based on the proposed method and implementation with iRobot Create are provided in Sections 4 and 5, respectively. Finally, conclusion is given in Section 6.

2 Stochastic Modeling

As a prior step to control a mobile robot system, we need to have a reference path that the robot should follow in path planning. There have been many research activities to obtain better paths according to a couple of different purposes. One example is the maximum probability approach [6]. Since the probability of the system motion stems from the stochasticity of the system, a mathematical model for the system with the consideration of the stochasticity should be firstly obtained. In this section, we model a two-wheeled mobile robot with the stochasticity.

2.1 Stochastic differential equation

As aforementioned, actual systems always contain a certain amount of errors such as uncertainty, noise and disturbance. We can include this stochastic effect into the theoretical model equation to express motion of the system more accurately by introducing a stochastic differential equation (SDE). The SDE is able to contain the stochastic effects when it expresses actual system’s behavior. The stochastic differential equation in \( \mathbb{R}^n \) is written as

\[
dx(t) = h(x(t), t)dt + H(x(t), t)dW(t)
\]  

where \( x \in \mathbb{R}^n \) and \( W \in \mathbb{R}^m \). We can obtain this expression by perturbing the deterministic system \( dx/dt = h(x, t) \) by noise or random force at every time. The random force is denoted by \( W(t) \) that is a vector of uncorrelated Wiener processes with unit strength for each dimension and a matrix \( H \in \mathbb{R}^{n \times m} \) scales and couples these noises.

The system variable \( x(t) \) can be obtained by integrating the equation (1). However, the integration results will not be consistent every time because of the stochastic term. We can handle this issue by employing a probability density function \( \rho(x, t) \) of \( x(t) \). One of the methods to obtain the probability density function from the SDE is to use the Fokker-Planck equation. This special partial differential equation governs the time evolution of the probability density function. For the SDE in (1), the Fokker-Planck equation is written as

\[
\frac{\partial \rho(x, t)}{\partial t} + \sum_{i=1}^{n} \frac{\partial}{\partial x_i} (h_i(x, t)\rho(x, t)) - \frac{1}{2} \sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j} \left( \sum_{k=1}^{m} H_{ik} H_{kj} \rho(x, t) \right) = 0
\]  

(2)

Fokker and Planck have found this theory firstly and the advanced versions were introduced in [13–16]. The analytic solution for the probability density function can be obtained by solving this Fokker-Planck equation [17] in principle. However, getting a closed form solution is challenging and sometimes impossible.

Another way to obtain the probability density function \( \rho(x, t) \) from the SDE is to numerically integrate the SDE (1) multiple times and build a histogram. This histogram represents the probability density function. The numerical integration of the SDE can be achieved by the Euler-Maruyama method [18]. The numerical integration for the general SDE (1) can be given as [6, 12]

\[
x_{i+1} = x_i + h(x_i, t_i)\Delta t_i + H(x_i, t_i)\Delta W_i,
\]

(3)

where

\[
x_i = x(t_i),
\]

\[
\Delta t_i = t_{i+1} - t_i,
\]

\[
\Delta W_i = W(t_{i+1}) - W(t_i).
\]

The simple update procedure represented by these equations is the Euler-Maruyama method. Since \( W(t) \) is the Wiener process, we can consider that \( \Delta W_i \) is sampled from a Gaussian distribution with zero mean and variance \( \Delta t_i \), which means

\[
\Delta W_i \sim \sqrt{\Delta t_i}\mathcal{N}(0, 1).
\]

2.2 Stochastic model for two-wheeled mobile robot

Figure 1(a) shows schematic representation of a two-wheeled mobile robot. The variables \( (x, y) \) and \( \theta \) denote the position and the orientation of the robot, respectively. The distance between two wheels measured along the axis is \( L \), the radius of the wheel is \( r \), the rotation angle of the right wheel rotates is \( \phi_1 \), and the rotation angle of the left wheel rotates is \( \phi_2 \). Both angles are measured counterclockwise viewed along the axis from outside to inside of the mobile robot. This model can be applied to most two-wheeled mobile robots including iRobot Create. According to the work in [7], we can assume no-lateral-slip
of the two-wheeled mobile robot with an assumption that the strength of the Wiener process. Let us define $2\delta$ as the difference between the angular velocities of the two wheels such that $\omega_1 = \omega + \delta$ and $\omega_2 = \omega - \delta$. By adopting the stochastic behavior of the system, the infinitesimal motion in (4) can be rewritten as

$$d\phi_i = \omega_i dt + \sigma dW_i$$

where $dW_i$ is the increment of unit strength Wiener processes, $\sigma$ is the strength of the Wiener process. Let us define $2\delta$ as the difference between the angular velocities of the two wheels such that $\omega_1 = \omega + \delta$ and $\omega_2 = \omega - \delta$. By adopting the stochastic behavior of the system, the infinitesimal motion in (4) can be rewritten as

$$\begin{pmatrix} dx \\ dy \\ d\theta \end{pmatrix} = \begin{pmatrix} \frac{x}{y} \cos \theta - \frac{y}{x} \cos \theta \\ \frac{x}{y} \sin \theta - \frac{y}{x} \sin \theta \\ \frac{d\phi_1}{d\phi_2} \end{pmatrix} \begin{pmatrix} d\phi_1 \\ d\phi_2 \end{pmatrix} dt$$

$$+ \sigma \begin{pmatrix} \frac{x}{y} \cos \theta - \frac{y}{x} \cos \theta \\ \frac{x}{y} \sin \theta - \frac{y}{x} \sin \theta \\ \frac{dW_1}{dW_2} \end{pmatrix}$$

where $\gamma = 2\delta / L$. This equation (6) is able to express the motion of the two-wheeled mobile robot with an assumption that the stochastic effects are applied to the angular velocities of the two wheels.

3 Path Generation with Stochasticity

In this section, we define the targeting error for a system to reach a target and show its use for the path planning of stochastic systems. The proposed method is applied to the two-wheeled mobile robot and then, finally, we will discuss potentials of the POP algorithm to be improved for better performance.

3.1 Path generation with a new cost function

To generate a path for a stochastic system, we can use the POP method. This method uses the probability density function to decide the locally optimal intermediate paths. In the previous implementations [6–11], the maximum probability was pursued during the path generation. Therefore, their research focused on finding an optimal path with the maximum probability to reach a goal. However according to [12], the targeting error instead of targeting probability is also a meaningful measure to determine if the planning is successful. The targeting error of stochastic systems can be evaluated using the probability density function. We use this new measure as a cost function for the POP method.

Suppose that a random variable $X$ is drawn from a probability density function (PDF) $\rho(x)$. The root-mean-square (RMS) distance is computed as

$$L = \sqrt{\frac{1}{N} \sum_{i}(x_i - x_d)^2} \approx \sqrt{\sigma^2 + (\mu - x_d)^2}$$

where $x_i$ is the $i$th sampled value, $N$ is the number of samples, $x_d$ is the target, and $\mu$ and $\sigma^2$ are the mean and the variance of the PDF, respectively. We can modify this formulation to our problem. Since the mobile robot explores on 2D space and we are focusing on the minimization of the cost, a new cost can be defined as

$$L'(\mu_x, \mu_y, \sigma_x, \sigma_y) = \sigma_x^2 + \sigma_y^2 + (\mu_x - x_d)^2 + (\mu_y - y_d)^2.$$ 

Therefore, the optimal path in this paper can be defined as a path with the minimum targeting RMS error to reach a goal.

3.2 Path generation for two-wheeled mobile robot

This statistical approach can be applied to the two-wheeled mobile robot to generate the optimal path as follows. We firstly assume a simple input angular velocity $\omega = 1$ with $\delta = 0$. The robot will follow the intermediate path defined by angular velocity $\omega = 1$ and $\delta = 0$ in a deterministic way during a small
time period $\Delta t$. Then, in the sampling stage, we generate multiple random path samples from the current intermediate state to a target by applying stochasticity to obtain a probability density function. We can calculate the cost that the future path reaches the target using this probability density function. Similarly, we can test with other inputs $(\omega, \delta) = (1, \pm \alpha)$ and $(\omega, \delta) = (1, \pm \beta)$ to evaluate the cost function. After obtaining all five costs using five candidates for $\delta$, we compare the costs to find which intermediate path is the best one for the next intermediate path in terms of the minimum targeting error. One small path with which the future path has the lowest targeting error will be selected. The algorithm runs until a pre-defined total number of path segments are obtained. Eventually, we can have an optimal path from initial position to a goal location by stitching the chosen intermediate paths. In the simulation and implementation of this paper we suggest the following options for each intermediate path’s candidates.

$$\delta = -\alpha \text{ or } -\beta \text{ or } 0 \text{ or } \beta \text{ or } \alpha. \quad (7)$$

The proposed algorithm will choose one optimal $\delta$ among candidates $(-\alpha, -\beta, 0, \beta, \alpha)$ by searching the minimum targeting error to reach a goal.

### 3.3 Improved POP method

According to the POP method, we choose the total number of path segments based on the distance between the initial position and the target position before applying the POP method because the algorithm runs until the fixed number of intermediate paths are obtained. Therefore, we should carefully decide the number of intermediate paths to get a meaningful full path reaching a target accurately. Otherwise, the full path will not reach the target when the total length of the intermediate paths is too short or the full path may pass over the target when the total length is too long.

To overcome this problem, we generate the random path samples with shorter and longer paths in the sampling stage. For example, we generate the random path samples with 9 and 11 segments as well as 10 segments in the case that we pre-defined total 10 path segments for the POP method. Due to these two additional sampling sets (with 9 and 11 segments), the algorithm can estimate whether the remained path segments is proper, short, or long. Therefore, the algorithm can actively adjust the length of segments. Figure 2 shows the probability density function with three different numbers of the path segments when a system is moving from $(0,0)$ to the right direction. The left PDF is generated by total 9 segments, the middle PDF is generated by total 10 segments, and the right PDF is generated by total 11 segments of the intermediate paths.

Moreover, there is another room for improvement of the POP algorithm. The idea is that we can iteratively update the optimal path. We generate multiple candidates of the optimal full paths by repeatedly applying the POP algorithm and evaluating them using the RMS targeting errors. To do this, after obtaining the first optimal path (after the 1st iteration is done), we apply the same algorithm again using the path that we obtained in the previous iteration. We repeat this iteration until the result converges to certain boundary.

### 4 Numerical Simulation

In this section, we verify the proposed algorithm with computer simulation. Compared to the classical POP algorithm, the simulation with the proposed method shows two improvements as mentioned in the previous section. For each simulation case, we test the algorithm to generate the optimal path with the lowest targeting error with and without obstacles. The two-wheeled mobile robot starts to move at $(0,0)$ for every simulation and all simulation parameters we used are identical to the actual parameters of the iRobot Create for the purpose of consistency between simulation and actual implementation that will be given in the next section.

#### 4.1 Simulation I : Active segments

Figure 3 shows simulation results of the proposed algorithm with and without obstacles. For the first simulation without an obstacle in Figure 3(a), the target point is at $(3800,900)$. The POP algorithm is applied and the optimized path in terms of RMS errors to hit the target is generated. A full path is composed of 13 segments even though default number of total segments is set by 10. The number of segments is actively changed depending on the position of the goal during the simulation. For each step to get the intermediate path, the algorithm evaluates five candidates of $\delta$ and selects one that brings the lowest RMS error for the robot to hit the target. The empirical parameters are selected as $\sigma = 0.07$ and
\[ \delta = [-0.1, -0.05, 0.0, 0.05, 0.1]. \]

Figure 3(b) shows the second simulation result with two obstacles. The target point is at (3850, -300) and two obstacles are at (1200, 200) and (2850, -350). The strategy to avoid obstacles is to exclude the paths that are overlapped with the obstacles when generating multiple random path samples. Therefore, the path generated by this strategy intrinsically avoids the obstacle.

4.2 Simulation II : Iterative method

As aforementioned, we can update the path through the iterative application of the POP method. In this iterative approach, the multiple random path samples in the sampling stage are generated based on the previous optimal path. We generate a set of paths through the iterative application of the POP method until one of three conditions is satisfied. The conditions are:

- The current path segments are more than the past path segments.
- The current targeting error is higher than the past targeting error.
- The number of iteration exceeds 50.

Figure 4 shows the result of the new path generation with the iterative approach. Figure 4(a) shows the optimal path generation.
without obstacles. The optimal path from the first iteration is denoted by the blue straight line and the final iteration result is denoted by the red dotted line. The black dotted line denotes the paths obtained during the iteration. Figure 4(b) shows the optimal path generation result with iteration with two circular obstacles. With this iteration, the targeting error decreases.

5 Implementation and Experiments

We implement the proposed algorithm with iRobot Create based on the simulation results to verify the algorithm with an actual hardware system. The iRobot Create is a circular two-wheeled mobile robot with approximately 169mm radius, 88mm height and 2.9kg weight. Two wheels are 30mm in radius and the distance between the two wheels is $L = 262mm$. Figure 5 shows the test environment with the iRobot Create. The test area represented by the blue rectangle is the size of 4000mm $\times$ 2000mm. The motion of the robot is captured by a camera. The white circles represent the obstacles.

The robot moves and rotates according to the result of the proposed path planning algorithm for each intermediate step. Different from the simulation, each intermediate position and orientation are updated by the iRobot’s encoder which means the actual values with noise are employed. Figure 6 shows our first implementation test. This result shows that the robot follows the optimal path with the lowest targeting error to reach the target without obstacles. Note that this is an implementation of the first simulation result shown in Figure 3(a). The final targeting error is measured as 51mm. Figure 7 shows an implementation test with two obstacles. The robot follows the optimal path with the lowest targeting error with obstacles. Similarly, this is an implementation of the second simulation result in Figure 3(b). The targeting error is measured as 72mm.

Additionally, we implement the iterative application. We make the robot follow the optimal path after the iteration process is finished. In this process, the multiple random path samples in the sampling stage are generated by the final optimal path that we obtained in the iteration process. Even though this implementation strategy doesn’t mean that the robot always reaches to the target with the minimum targeting error, we can expect the

FIGURE 6. Implementation of the algorithm with iRobot Create (no obstacle). The same parameters for the target and obstacles as in Fig. 3(a) are used.

FIGURE 8. Implementation of the algorithm with iRobot Create (no obstacle) with iteration approach. The same parameters for the target and obstacles as in Fig. 4(a) are used.

FIGURE 7. Implementation of the algorithm with iRobot Create (obstacles). The same parameters for the target and obstacles as in Fig. 3(b) are used.

FIGURE 9. Implementation of the algorithm with iRobot Create (obstacles). The same parameters for the target and obstacles as in Fig. 4(b) are used.
higher possibility to reach the goal with the minimum RMS error to hit the target than the implementation without the iterative approach. Figure 8 shows the implementation result that reflects the simulation result in Figure 4(a). The actual error is slightly decreased to 45 mm compared to the implementation without iteration. The last result shown in Figure 9 is implementation of the result in Figure 4(b). The final targeting error is measured as 65 mm. Note that the error in implementation is slightly larger than the simulation result because we used actual intermediate position and rotation values obtained by the internal encoder to calculate next intermediate path. In other words, we considered the actual uncertainty from the environment to verify the proposed method in the actual implementation.

According to the implementations, we can guarantee the proposed algorithm can generate the optimal path with respect to the lowest targeting error in an environment with and without obstacles. Additionally, we can verify that the improved method with iterative approach can possibly decrease the targeting error. Consequently, the proposed method shows the performance to make an actual two wheeled mobile robot follow the optimal path with effective obstacle avoidance.

6 Conclusion

In this paper, we improved a path planning method to generate optimal paths with the consideration of system stochasticity. Then, we applied it to an actual mobile robot platform. The improved algorithm generates paths that minimize the targeting error. To do this, we defined a new cost function by applying the root-mean-square error to hit the target, and adapt it in the path-of-probability (POP) method. We also improved the POP method by actively adjusting the number of intermediate paths so that the number of intermediate paths is automatically adjusted during the optimal path generation process. In addition, we modified the algorithm to operate iteratively. This iterative approach is justified by the fact that the POP algorithm finds only locally optimal answers with the first iteration. We verified the performance of the path planning algorithm with simulation and actual implementation. We finally applied this theoretic approach to the iRobot Create and confirmed that the algorithm works successfully in path planning for the two-wheeled mobile robot. For the future, we will investigate other types of criteria for generating optimal paths such as shortest paths and minimum travel time.

REFERENCES
