The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

Dynamic Window-based Obstacle Avoidance in the Presence of Moving Obstacles

H. Harada[1], M. Yokomichi[2], and O. Sato[2]

[1] Graduate School of Enginnering, University of Miyazaki, Miyazaki 889-2155, Japan
[2] Faculty of Engineering, University of Miyazaki, Miyazaki 889-2155, Japan
(yokomich@cs.miyazaki-u.ac.jp)

Abstract: This paper proposes a reactive control method for mobile robots in the presence of moving obstacles. The method is based on the dynamic window algorithm and extends it in order to avoid moving obstacles efficiently. Firstly, the future collision is detected based on the generalized velocity obstacles. Secondly, input value can be varied within the prediction horizon. This means that better input can be selected such that the robot starts passing maneuver ealier. However, this causes the dimension of the search space larger. In order to reduce the computation time, GDS (gradually dense-sparse) discretization and randomized sampling method similar to RRT is adopted. By means of these extensions, the robot can avoid moving obstacles with simpler cost function and reasonable computation time. Performance of the proposed method is evaluated by numerical simulations.

Keywords: Dynamic Window, Receding Horizon Control, Randomized MPC GDS discretization

1 INTRODUCTION

The problems of path-planning and collision avoidance are computing a collision-free path for a robot moving among obstacles and are crucial tasks for mobile robots. The application area has been spread to wide areas such as, automated transportation systems, automated factories, and robot human interactions. In many cases, the environment is uncertain and dynamic, thus, the path-planning and obstacle avoidance system should perform in such environments. When restricted to the collision avoidance, there have been proposed many reactive approaches in the last thirty years. Originally, these approaches have been applied to the static environment, then, some of them have been extended to the dynamic case recently.

The dynamic window approach, which was originally proposed by Fox et al in [1], is one of the most popular reactive collision avoidance methods. The original algorithm was extended by Brock and Kathib [2] such that the global path planning such as A* is integrated in the cost function. Furthermore, Seder and Petrović [4] applied the FD* search algorithm in order to consider the dynamically changing environment. However, in the case where the obstacles are moving, the planning should be time-dependent especially in the predicting phase, thus, more complicated computation for path planning is needed.

On the other hand, because of its *look ahead* nature, the dynamic window approach can be thought as one of the *Receding Horizon Control* problem (RHC, for short) in the model predicted control. This point has been pointed out by Ogren and Leonard [3]. In the general RHC problem, it is assumed that the input may change within the predicting horizon when the optimal input is searched.

But, in many works on the dynamic window approach, the use of constant input value is assumed in order to reduce the dimension of search space and decrease the computation time. Recently, Kiss and Tecesz [7] applied the RHC approach to dynamic window and they permitted piecewise constant input in the prediction phase, however, they did not show the concrete method to compute the optimal input sequence and, in addition, considered only static problem.

In this paper, the authors propose a method to extend the dynamic window approach to the moving obstacle avoidance with piecewise constant input. In the proposed method, the obstacles are divided to two classes; static and dynamic. The goal-intended term in the cost function is calculated by path planning method only for static obstacles. The collision to the dynamic obstacles are evaluated by computing the generalized velocity obstacles which is proposed by Wilkie et al [5]. In order to reduce the computation time, a randomized planning method with the gradually dense-sparse discretization is applied. The randomized planning method used in this work is proposed by Brooks et al [9] for the static path planning problem and the authors modify it in order to fit to the dynamic obstacle avoidance. The gradually dense-sparse discretization [10] is one of the methods to discritize the time horizon in the RHC problem and can reduce the search space without the loss of control performance. By applying these methods, the robot can avoid the obstacles with better behavior and with admissible computation time. The effectiveness of the proposed approach is evaluated by numerical simulations.

2 PRELIMINARIES

In this paper, it is supposed that the robot moves on a horizontal plane and its kinematics is nonholonomic differential-drive type. The shape of the robot and the obstacles are as-

sumed to be approximated by circles. The state vector of the robot is given by $p(t) := (x(t), y(t), \theta(t))^T$, where $q := (x, y)$ and θ are the position and the orientation of the robot. The continuous time kinematics of the robot is described as

$$\begin{aligned} \dot{x} &= \cos(\theta)v \\ \dot{y} &= \sin(\theta)v \\ \dot{\theta} &= w \end{aligned} \tag{1}$$

where the input $u = (v, w)$ consists of the translational and the angular velocities. The solution of (1) for $p(t_0) = p_0$ and $u(t); t \in [t_0, t_1]$ is denoted as $p(t_1; t_0, p_0, u)$. Furthermore, it is assumed that the values and instantaneous variations of these inputs are limited. It is also supposed that the static map of the environment and the movement of the temporally obstacles is known.

2.1 Dynamic Window Approach

The dynamic window approach, proposed by Fox et al. [1] is a velocity space based local reactive avoidance technique where the optimal control command is searched in the subspace of the velocity space. The search space is reduced by kinematic and dynamic constraint of the robot and the constraint that the robot with the input does not collide with the obstacles, and the reduced search space is called *Dynamic Window* and denoted as V.

Suppose that the time is discretized at equal intervals Δt and the input is constant within each interval. In addition, the predicting time horizon T is set as $T = n\Delta t$ for some integer $n > 1$ and it is assumed that the input is constant with in the predicting horizon in evaluating the input.

the cost function in [1] is defined as a linear combination of *heading* term, *distance* term, and *velocity* term. If the position trajectory by the input is directed towards the goal, then the heading term takes high value. The distance term is the smallest distance to the obstacles within the horizon, and the velocity term is simply the value of v.

2.2 Extensions of Dynamic Window

Using the original dynamic window approach proposed in [1], robust obstacle avoidance is achieved in many cases, but there exist several shortcomings. These are summarized as :

- The heading function may cause local minima when the shape of the free space is complicated .
- It does not consider the dynamic environment.

To overcome the first problem, Brock and Khatib modified the cost function such that the free space connectivity information is taken into account [2]. They used a navigating function (NF) which is a local minima-free function defined on the discretized configuration space. NF is computed by the global path planning algorithm such as A* or wavefront propagation, so their approach is called *global* dynamic window (GDW, for short). Suppose that NF is denoted as $N(q)$ and the current state p_0, input $u \in V$, and the corresponding trajectory $p(t; t_0, p_0, u)$, $t \in [t_0 : t_0 + T]$ is given. Then, the modified cost function in [2] has the following form:

$$\begin{aligned} \Omega_g(p, u) &= \alpha \cdot n_1(p, u) + \beta \cdot vel(u) \\ &\quad + \gamma \cdot goal(p, u) + \delta \cdot n_2(p, u), \end{aligned} \tag{2}$$

where n_1 takes high value when the difference between $\theta(t_0 + T; t_0, p_0, u)$ and the direction of $\nabla N(q(t_0 + T; t_0, p_0, u))$ is large, *goal* is a binary function (1 if the trajectory $q(t; t_0, p_0, u)$ pass through the goal area), and

$$n_2(p, u) = N(q_0) - N(q(t_0 + T; t_0, p_0, u)).$$

Because the original heading term is replaced with NF, the resulting maneuvers can avoid local minima so that the robot can move along the optimal path.

It should be noted that, in (2), the *distance* term is deleted. This is because the local minima may caused by the trade-off between *heading* term and this term.

The second problem is more serious. The framework in [1] can be easily extended to the dynamic environment by means of the velocity obstacles[8], or its generalizations [5]. However, if the moving obstacles is not considered in the path planning phase, trade-off problem may occurs. On the other hand, the method in [2] needs the global free-space information. this means, for all time instances, the position and shape of all obstacles should be known. When the environment is static, it is sufficient that the path planning is made at the first phase only. However, when the environment is dynamic, on-line re-planning is needed. Furthermore, when the obstacles move, the path planning problem should be time-dependent, such that, the search space is extended to $\mathscr{CT}$-space[6].

In [4], FD* algorithm is applied to GDW and its effectiveness are shown. But, even if the FD* is used, the replanning needs large computation time, thus it is not easy to apply this method to the case where the moving obstacles are large, or when the obstacles move fast.

2.3 Optimality issue

Many of the previous researches about DW assume that through the predicting time horizon, the input should be constant. As mentioned above, this is because the search space can be reduced so that the computation time becomes small. In the case where the environment is static and where GDW is used, this yields satisfactory results in many cases because the optimal path is already given by path planning phase, thus the time horizon can be made rather short. However, especially in the dynamic environment without re-planning, this assumption causes not only non-optimal but also dangerous behaviors.

3 EXTENSION TO PIECEWISE CONSTANT INPUTS

Based on the discussions in the previous section, it is preferred to use piecewise constant inputs for search instead of constant input throughout the predicting time horizon. As mentioned above, this extension causes the increase of the dimension of the search space such that the computation time increases exponentially. Therefore, some kind of computationally efficient solver is required.

It has been reported in [3], DW can be regarded as a special kind of *Receding Horizon Control* approach in the model predictive control (MPC), and several real-time solvers has been proposed for some problems including the trajectory generation problem for mobile robots. In this paper, Randomized MPC approach in [9] and GDS (Gradual Dense-Sparse) discretization are applied to DW for moving obstacle avoidance.

The randomized MPC is a kind of input space-based tree search for RHC, which is similar to RRT for path planning problem. For a system

$$\dot{x} = f(x,u) \tag{3}$$

with state variable $x \in X \subset \Re^n$ and input $u \in U \subset \Re^m$.

For given time horizon $[t_0 : t_0 + t']$, suppose that its discretization

$$t_0, t_1, t_2, \ldots, t_{r-1}, t_r = t_0 + t'$$

is given. We suppose that the input is piecewise constant and let $u_{t_0:t_r} = \{u_0, u_1, \ldots, u_{r-1}\}$ denotes a input sequence. Let $c(x(t_0), u_{t_0:t_r})$ denotes a cost function for initial condition and control sequence. The task is to find the control sequence $u^*_{t_0:t_r}$ which maximize (minimize) $c(x(t_0), u^*_{t_0:t_r})$. The basic ideas of the randomized MPC is similar to an RRT. A tree T is grown in which each node q consists of the tuple $< x_q, t_i, u_q >$: a state, a time, and a control input. This means, for the parent node $q' =< x_{q'}, t_{i-1}, u_{q'} >$, x_q is a solution of (3) at $t = t_i$ whose initial position is $x_{q'}$, and to which input u_q is applied. Thus, algorithm iteratively expands the tree by selecting an existing node q' and a control input u_q, then adding new nodes by integrating (3) forwards in time until t_q. Let $c_T(q_f)$ denotes a cost of the control input $u_{t_0:t_r}$ from the root node to a final leaf q_f at $t = t_0 + t'$. The basic procedure in [9] is summarized in Algorithm 1. Furthermore, in the next time, the previous optimal control sequence is used for the initial tree for effective search.

In the Algorithm 1, some subroutines should be specified. The usual RRT implementations of **SELECT_NODE** and **SELECT_CONTROL** sample a point in state space at first, select the nearest node as parent, then select the control which grows the tree from the node toward the sampled point. By means of these procedures, uniformly sampling or exploring the state space is achieved, however, these requires high computational time especially when the nearest neighbor node is found. In [9], parent node is selected uniformly from the existing nodes, and input is also selected uniformly from U. In addition, the method to discritize the time horizon is not addressed. These simplifications yields the low computational cost, but, may causes biased search in the state space. In order to achieve wider searching region with small number of node, Gradual Dense-Sparse discretization (GDS discretization) for predicting time horizon [10] is effective. In GDS discretization, the discretization time interval is made short in the early phase, and is made larger incrementally.

Algorithm 1 Tree Expansion for Randomized MPC

```
initialize: T ← q_root =< x(t_0), t_0, u_0 >, c_max ← 0;
for i = 1, ..., N do
    q ← SELECT_NODE(T);
    u ← SELECT_CONTROL(T, q);
    t ← t_i, x ← x_q;
    while t < t_0 + t' do
        x ← x + ∫_{t_i}^{t_{i+1}} f(x,u)dt;
        t ← t_{i+1};
        if x ∉ X then
            break;
        end if
        add q_new =< x, t, u > to T;
        if t = t_0 + t' then
            c_q ← c_T(q_new);
            if c_q > c_max then
                c_max ← c_q, q_best ← q_new;
            end if
        end if
    end while
end for
```

4 SIMULATION RESULTS

The proposed algorithms have been implemented in *Player/Stage* software tools. In order to illustrate the effectiveness of the proposed algorithm, a result of a test is presented in Fig. 1. In Fig. 1 red rectangle is the controlled

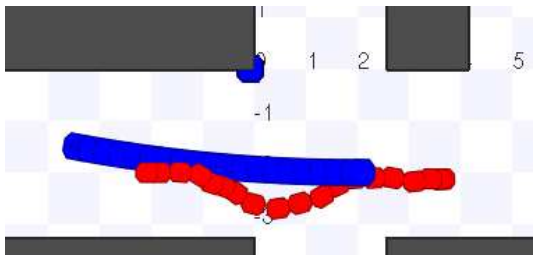

Fig. 1. Trajectories of the robot and the obstacle.

robot and blue one is the obstacle. The obstacle is also governed by the kinematic model (1) and the input is supposed to be constant $(v,w) = (0.3(m/s), -0.01(rad/s))$. The sampling interval and the prediction horizon are set to 0.1(s) and 3.2(s), respectively. The discretization intervals for GDS are set to 0.2, 0.2, 0.2, 0.2, 0.8, 1.6(s) and the number of nodes for randomized MPC is set to 4,200. In this case, the average computation time for about 16(s) execution was 0.012(s) by core i5 2.3GHz processor. The navigating function $N(q)$ for static environment was a L_1-based distance to the goal point, which was obtained by wavefront propagation algorithm a priori. It should be noted that, in the situation depicted in Fig. 1, the navigating function for the static map is the same as the simple L_1 distance. The cost function used at simulation is

$$\Omega_g(p,u) = \alpha \cdot n_1(p,u) + \gamma \cdot goal(p,u) + \delta \cdot n_2(p,u),$$

that is. less terms than (2) are used. This means that the parameter tuning becomes easier than GDW. In addition, the result is compared with those that the constant inputs are selected in Fig. 2, where additional *distance* term is added to the cost function. In Fig. 2, the line "single T=2" means

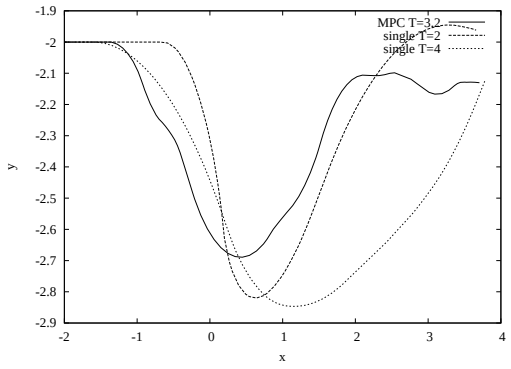

Fig. 2. Trajectories of the robot and the obstacle.

that the optimal constant input is searched in each time with prediction horizon $T = 2$. From the Fig. 2, the robot based on the proposed method starts avoiding maneuver as early as $T = 4$, and pass the obstacle more quickly than others.

5 CONCLUSIONS

In this paper, an extension of Dynamic Window approach is proposed for dynamic environment, and its effectiveness is examind by numerical simulations. By admitting piecewise constant input in the predicting and searching step, better performances are obtained. In addition, by means of randomized searching with GDS discretization, real-time calculation is also possible. However, the obtained trajectory seems to be nonsmooth and its modification is one of the future issues.

REFERENCES

[1] D.Fox, W.Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, Vol. 4(1), 23-33, 1997.

[2] O.Broch and O.Khatib, "High-speed navigation using the global dynamic window approach," *Proc. of the Int. Conf. on Robotics and Automation*, 341-346, 1999.

[3] P.Ögren and N.E.Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Trans. Robot.*, Vol.21, 188-195, 2005.

[4] M.Seder and I.Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstaecles," *Proc. of the Int. Conf. on Robotics and Automation*, 1986-1991, 2007.

[5] D.Wilkie, J.v.d.Berg, and D.Manocha, "Generalized velocity obstacles," *Proc. IEEE/RSJ int. Conf. Intelligent Robots and Systems*, 5573-5578, 2009.

[6] J. Latombe, *Robot motion planning*, Kluwer Academic Publishers, 1991.

[7] D.Kiss and G.Tevesz, "A receding horizon control approach to obstacle avoidance," *Proc. IEEE Int. Symp. on Applied Computational Intelligence and Infomatics*, 397-402, 2011.

[8] P.Fiorini and Z.Shiller, "Motion planning in a dynamic environments using velocity obstacles," *Int. J. of Robotics Research*, Vol.17(7), 760-772, 1998.

[9] A.Brooks, T.Kaupp, and A.Makarenko, "Randomised MPC-based motion-planning for mobile robot obstacle avoidance," *Proc. Int. Conf. on Robotics and Automation*, 3962-3967, 2009.

[10] P.Ögren and J.W.C.Robinson, "Receding horizon control of UAVs using gradual dense-sparse discretizations," *Proc. AIAA Guidance, Navigation and Control Conf.*, 2010.