A Mobile Robot Path Planning Algorithm Based on Improved A* Algorithm and Dynamic Window Approach

Yonggang Li, Rencai Jin, Xiangrong Xu, et al.

Presented by Khang Luong, Ong Xuan Son

October 11, 2025

Outline

Problem and Proposed Solution

Limitations of Traditional Algorithms:

- **Traditional A* Algorithm:** Often produces paths with many turning points, redundant nodes, and large turning angles. It is unable to avoid dynamic obstacles.
- **Dynamic Window Approach (DWA):** As a local algorithm, it is prone to getting trapped in local optima and may fail to reach the target position.

Proposed Solution:

- This paper presents a **hybrid algorithm** that fuses an **improved A*** with the **DWA**.
- **A* Algorithm Improvements:**
  - An **adaptive adjustment step algorithm** to enhance flexibility based on obstacle density.
  - A **three-time Bezier curve** to smooth the trajectory, reducing turning points and angles.
- **Fusion:** The improved A* algorithm plans the global path, and the DWA is then used for local planning and real-time dynamic obstacle avoidance.

Problem Formulation

Input:

- A global map of the static environment.
- Start and target coordinates (x_{start}, y_{start}) and (x_{target}, y_{target}).
- Real-time sensor data (e.g., from Lidar) for detecting dynamic obstacles.

Output:

- A safe, smooth, and optimal (or near-optimal) trajectory from the start to the target.
- A sequence of velocity commands (v, ω) for the robot to follow the trajectory.

Environment Model:

- The environment is represented using a **grid method** (raster modeling).
- The space is divided into equal, disjoint grid cells.
- Each cell is classified as either **free** (white) or **occupied** (black).
- The model contains both static obstacles (known on the global map) and dynamic obstacles (detected locally).

Outline

Traditional A* Algorithm

The A* algorithm is a heuristic search method for global path planning.

Cost Function: The cost of a node n is evaluated by the function:

$$f(n) = g(n) + h(n)$$

$f(n)$: The total estimated cost of the path through node n.
$g(n)$: The **actual cost** from the start node to the current node n.
$h(n)$: The **heuristic estimated cost** from node n to the target node.

Search Process:

- It uses 2 lists: an **Open list** for unexpanded nodes and a **Closed list** for expanded nodes.
- It repeatedly selects the node with the smallest $f(n)$ value from the Open list, expands it, and moves it to the Closed list until the target is found.
- The search typically explores neighbors in four or eight directions.

Improvement 1: Adaptive Step Size

To increase flexibility, the algorithm adapts its step size based on the surrounding environment.

Threat Function: The obstacle density is quantified by a threat function $f(x_1, x_2)$:

$$f(x_1, x_2) = \begin{cases} \frac{1}{k_1 x_1 + k_2 x_2 + c} & \text{if } d = 0 \\ 1 & \text{if } d \neq 0 \end{cases}$$

- x_1, x_2: The number of static obstacles in immediate and nearby areas, respectively.
- d: The number of dynamic obstacles in the direction of motion.
- k_1, k_2, c: Weighting coefficients.

Adaptive Step Size Formula: The step length l is then calculated as:

$$l = \begin{cases} f(x_1, x_2) \cdot l_{\max} & \text{if } d = 0 \\ f(x_1, x_2) \cdot l_{\min} & \text{if } d \neq 0 \end{cases}$$

This allows the robot to take larger steps in open spaces and smaller, safer steps near obstacles.

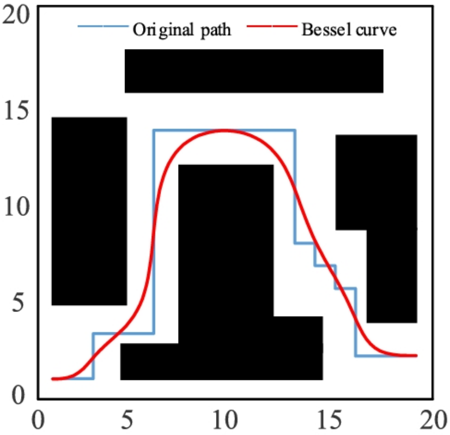
Improvement 2: Path Smoothing with Bezier Curves

Cubic Bezier Curve Formula:

- A curve segment is defined by a start point P_0, an end point P_3, and two control points P_1, P_2.
- The coordinates $B(t)$ on the curve at time $t \in [0, 1]$ are given by:

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3$$

This smoothing process ensures the robot can travel smoothly, reducing motor strain and satisfying motion constraints.



Outline

DWA Overview and Evaluation Function

1. **Velocity Sampling:** Sample multiple pairs of linear (v) and angular (ω) velocities within a "dynamic window" constrained by motor performance, acceleration limits, and a safe braking distance.
2. **Trajectory Simulation:** Predict the robot's trajectory for each sampled velocity pair over a short time interval.
3. **Evaluation and Selection:** Use an evaluation function to score valid (non-colliding) trajectories and select the one with the highest score.

DWA Evaluation Function:

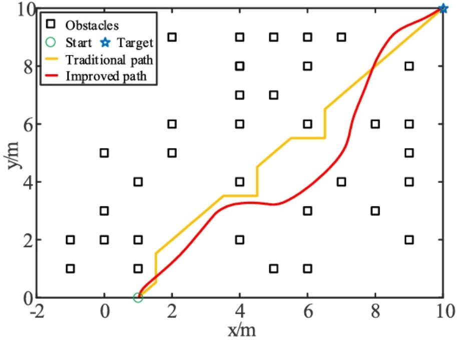
$$G(v,\omega) = \sigma[\alpha \cdot head(v,\omega) + \beta \cdot stob(v,\omega) + \delta \cdot dyob(v,\omega) + \gamma \cdot velo(v,\omega)]$$

$head(v,\omega)$: **Azimuth**, measuring angular deviation from the global path.
$stob(v,\omega)$: **Distance** to the nearest static obstacle.
$dyob(v,\omega)$: **Distance** to the nearest dynamic obstacle.
$velo(v,\omega)$: The robot's forward **velocity**.

Outline

Fusing Improved A* and DWA

1. **Global Planning:** The improved A* algorithm (with adaptive step and Bezier smoothing) plans a globally optimal path on the static map.
2. **Local Target Points:** Key points from the global path serve as temporary sub-goals for the local planner.
3. **Local Planning:** The DWA algorithm navigates toward the current local target, performing real-time avoidance of both static and dynamic obstacles using sensor data.
4. **Update and Repeat:** As the robot moves, the local target point is continuously updated along the global path until the final destination is reached.



Hybrid Algorithm Performance

Quantitative Comparison:

Algorithm	Turning	Smoothness	Dynamic Avoid	Path Length
Traditional A*	8	No	No	14.07
Improved A*	6	Yes	No	11.92
DWA	-	Yes	Yes	Not reached
Hybrid Algorithm	**4**	**Yes**	**Yes**	**13.56**

Table: Performance comparison of the different algorithms.

- Compared to the traditional A* algorithm, the hybrid method reduces the number of turns by **50%** and the path length by **3.62%**.
- The algorithm successfully solves the inability of A* to avoid dynamic obstacles and prevents DWA from getting trapped in local optima.

Outline

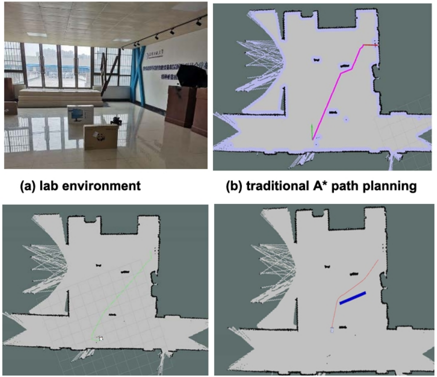
Real-World Robot Experiments

Setup:

- **Software:** Ubuntu 18.04 with ROS (Melodic).
- **Robot Hardware:** Equipped with Lidar, Camera, IMU, and Mecanum wheels.
- **Environment:** An 80m^2 lab space for physical tests.

Key Findings (Average of 10 runs):

- The hybrid algorithm reduced the average time consumption by **10.27%**.
- The number of path inflection points was reduced by **57.14%**.
- The accuracy at the end point was higher by **33.33%** compared to the traditional algorithm.



(a) lab environment (b) traditional A* path planning

(c) fusion algorithm path planning (d) path planning under dynamic obstacles.

Outline

Conclusion and Future Work

Summary of Contributions:

- An **improved A* algorithm** was developed using an adaptive step size and Bezier curve smoothing, which reduces run time and the number of turning points.
- A **hybrid algorithm** was created by fusing the improved A* with DWA, successfully combining global optimality with real-time dynamic obstacle avoidance.
- Extensive simulations and real-world experiments validated that the proposed algorithm has good applicability and security for complex dynamic environments.

Future Work:

- The authors suggest further exploring the path planning of mobile robots in multi-task complex scenes.
- Future research will also investigate combining the algorithm with deep learning and machine vision.