fetch robot DWA

The term **"Fetch robot DWA"** refers to the use of the **Dynamic Window Approach (DWA) local planner** for navigation with the **Fetch mobile manipulator robot**. The Fetch robot commonly operates using the Robot Operating System (ROS), which includes the `dwa_local_planner` package for autonomous navigation.

Dynamic Window Approach (DWA) Explained

DWA is a widely used algorithm for local path planning and obstacle avoidance in mobile robots.

- **Function:** It is responsible for connecting a global path plan to the robot's physical movement by generating collision-free trajectories in real-time.
- **Mechanism:** The algorithm works by sampling a range of possible velocities (the "dynamic window") and simulating the robot's movement for each velocity over a short time horizon. It then selects the trajectory that best achieves the overall goal while avoiding potential obstacles.
- **ROS Integration:** In the ROS navigation stack, the `dwa_local_planner` package provides an implementation of this controller.

Fetch Robot Context

The Fetch robot, a mobile manipulation platform originally designed for logistics and research, utilizes the ROS framework for its operations. When performing autonomous navigation tasks (like moving from point A to point B in a warehouse or

Ask anything

The Fetch robot is used in various research environments and simulation tasks, often involving navigation challenges where DWA plays a key role.

What parameters can be tuned in DWA for the Fetch robot?

- **`max_vel_x`** / **`min_vel_x`** : The maximum and minimum translational velocity in the forward (x) direction (e.g., 0.5 m/s).
- **`max_vel_y`** / **`min_vel_y`** : The maximum and minimum translational velocity in the sideways (y) direction. For non-holonomic robots like the Fetch, these are typically zero.
- **`max_rot_vel`** / **`min_rot_vel`** : The maximum and minimum rotational velocity (e.g., 1.0 rad/s).
- **`acc_lim_x`** / **`acc_lim_theta`** : The linear (x) and angular (theta) acceleration limits. Setting these correctly is crucial for physically realistic motion.
- **`max_trans_vel`** / **`min_trans_vel`** : Absolute limits on the overall translational speed, often used to enforce a consistent speed range.

Oct 27, 2020 — * New in navigation ROS Diamondback, navigation_experimental ROS CTurtle. * The dwa_local_planner package provides a c...

Show all

2. Forward Simulation Parameters

These parameters control how the DWA algorithm predicts the robot's future movements to evaluate potential paths.

- **`sim_time`** : The amount of time in the future (in seconds) to simulate trajectories. A longer time allows the robot to plan further ahead but might make it difficult to enter tight spaces.
- **`vx_samples`** / **`vtheta_samples`** : The number of samples to use when discretizing the velocity search space for linear (x) and rotational (theta) velocities. More samples lead to a finer search but increase computation time.
- **`sim_granularity`** : The step size used in the forward simulation.

3. Trajectory Scoring Parameters (Biases)

tendency to cut corners.

- **`occdist_scale`**: A weighting factor for obstacle avoidance costs. A higher value makes the robot more cautious and maintain a larger buffer around obstacles.
- **`forward_point_distance`**: The distance ahead of the robot to look for the "forward point" used in scoring, which helps prevent collisions when the robot is wider than it is long.

4. Goal Tolerance Parameters

These define the conditions under which the robot considers itself to have reached the target.

- **`xy_goal_tolerance`**: The distance threshold the robot must be within from the goal position in the x-y plane.
- **`yaw_goal_tolerance`**: The angular threshold the robot must be within from the final goal orientation.
- **`latch_xy_goal_tolerance`**: A boolean that, when true, stops translational movement once the robot is within the `xy_goal_tolerance`, allowing it to rotate in place to achieve the final `yaw_goal_tolerance`.

5. Oscillation Prevention Parameters

These help the robot escape situations where it gets stuck oscillating back and forth between two states (e.g., stuck between two obstacles).

- **`oscillation_reset_dist`**: The distance the robot must move away from a previous position to reset the oscillation detection.

What are some best practices for tuning DWA on the

let's focus on Robot Configuration and Velocity Limits....

How Robot Configuration and Velocity Limits Work

The **Robot Configuration and Velocity Limits** parameters are physical

- **Safety:** Setting these correctly is vital. If `acc_lim_x` is set too high, the DWA planner might command speeds that the robot's motors cannot physically reach or safely stop from before hitting an obstacle.

Changing Parameters using ROS

In ROS, you can modify DWA parameters *dynamically* while the system is running using the `dynamic_reconfigure` mechanism. This prevents the need to restart the entire navigation stack for every test.

Using `rqt_reconfigure` GUI (Recommended for tuning)

The easiest way is using the graphical interface:

1. Launch the `rqt_reconfigure` GUI:

```bash
rqt_reconfigure
```

Use code with caution.

2. In the GUI window, find and select the node name associated with your local planner (usually something like `/move_base` or `/dwa_local_planner`).
3. Expand the list of parameters and navigate to the relevant limits (e.g., `max_vel_x`, `acc_lim_theta`).
4. Adjust the slider or text box values in real-time. The changes are immediately sent to the active node, and the robot's behavior will change instantly.

Use code with caution.

Jun 25, 2021 — * Robot Configuration Parameters. maxvelx: 0.5 #The maximum x velocity for the robot ( ) minvelx: -0.2 ...

ROS Answers

Note: Replace `/move_base` with your actual DWA node name if it differs.

How Parameters Affect Robot Motion & Test Command

parameter tuning for dwa_local_planner: unable to turn properly

Jun 8, 2015 — dwalocalplanner, params.yaml DWAPlannerROS: acc_lim_x: 2.0 acc_lim_y: 0 acc_lim_th: 3.0 max_trans_vel: 0.5 min_trans_vel...

ROS Answers

Changing these parameters has direct, observable effects on the Fetch robot's movement characteristics.

Parameter	Effect of Increasing Value	Effect of Decreasing Value
`max_vel_x`	Robot moves faster when traveling in a straight line toward a goal.	Robot moves more slowly, increasing total navigation time.
`max_rot_vel`	Robot rotates faster in place or while turning corners.	Robot turns slowly, resulting in wider turns and slower overall speed in cluttered areas.
`acc_lim_x`	Robot accelerates and decelerates more aggressively; starts and stops faster.	Robot movement is smoother and more gradual, reducing jerky motions.

A New Local Planner - Robot & Chisel

Oct 13, 2024 — All of these controllers have many parameters, but the ones for DWA are especially difficult to tune because they are ...

www.robotandchisel.com

Show all

Robot Command to Test:

To observe the effects, you need to send a navigation goal to the robot's navigation stack. The `move_base` node listens for goals on the `/move_base_simple/goal` topic.

You can send a goal using the **RViz interface** by clicking the "2D Nav Goal" button

Use code with caution.

Metrics for Performance Evaluation

Evaluating performance involves quantitative and qualitative metrics to determine if a set of parameters performs "well" in your specific environment (e.g., a lab or warehouse).

Quantitative Metrics

1. **Time to Goal:** The total time taken for the robot to reach the destination. Lower is generally better for efficiency.
2. **Success Rate:** The percentage of navigation attempts that reach the goal without failing (e.g., getting stuck, oscillating, or hitting an obstacle).
3. **Path Efficiency/Length:** Comparing the actual distance traveled by the robot versus the optimal, shortest global path distance.
4. **Average Velocity:** The average speed maintained throughout the journey (calculated from odometry data).

Qualitative Metrics & Tools

1. **Smoothness of Motion:** Observe if the robot moves fluidly or if it exhibits jerky starts, stops, or frequent oscillations.
2. **Obstacle Clearance:** Visually confirm the distance the robot maintains from obstacles.
3. **Visualization Tools:** Use **RViz** to visualize the DWA's "trajectory cloud" and the local costmap. This helps you see which trajectories are being scored highly and

Tell me more about using RViz for DWA