Fundamentals of Local Planning

David V. Lu

ROSCon2017

LOCUS
Central Dogma of ROS Navigation

Current Location + Goal Location

Global Planner
Global Plan
Local Planner
Command Velocities
Fundamentals of Local Planning

Fundamentals / Theory

ROS Navigation Today

ROS Navigation Tomorrow
What is the best command velocity?

What command velocities are available?

What would the commands do?

How do we define the "best" command?
What Command Velocities are Available?
Velocity Search Space

- $x, \theta$ - Non-holonomic Robots
- $x, y, \theta$ - Holonomic Robots
- $x, y, z, \text{roll}, \text{pitch}, \text{yaw}$ - Drones
A Brief History Interlude:
Vector Fields - 1986

Oussama Khatib
Artificial Intelligence Laboratory
Stanford University
Stanford, California 94305

Real-Time Obstacle Avoidance for Manipulators and Mobile Robots

A Brief History Interlude:
DWA - 1997

The Dynamic Window Approach to Collision Avoidance

Dieter Fox† Wolfram Burgard† Sebastian Thrun‡‡
†Dept. of Computer Science III, University of Bonn, D-53117 Bonn, Germany
‡Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, P A 15213
Email: {fox,wolfram}@uran.cs.uni-bonn.de, thrun@cs.cmu.edu

23-33.

\[ v_{\text{min}} = v_0 - a \times t \]
\[ v_{\text{max}} = v_0 + a \times t \]
What Command Velocities are Available?

Sampling Search

vel_x, vel_theta

vx_samples = 5
ttheta_samples = 9
What is the best command velocity?

What command velocities are available?

What would the commands do?

How do we define the "best" command?
What would the commands do?

Trajectory Generation

pose:  (0.0, 0.42, 0.0) /map
vel:    (0.0, 0.0, 0.0)
cmd_vel: (1.0, 0.0, -0.8)

t=0.0  
(0.0, 0.42, 0.0)

t=0.75  
(0.63, 0.31, -0.51)

t=1.50  
(1.12, -0.09, -1.03)

Key Parameters:
sim_time (t=1.5)
discretization parameters
What is the best command velocity?

What command velocities are available?

What would the commands do?

How do we define the "best" command?
How do we define the "best" command?

Trajectory Scoring

- Moves towards goal
- Doesn't hit obstacles

A: 10.0
B: 40.0
C: -1.0

Score Rules:
Negative is Invalid
Lower is Better
How do we define the "best" command?

Critic and Scores

Critics produce scores

Final Score = Weighted Sum of Scores

For each critic:
  if critic.raw_score < 0: return critic.raw_score
  score += critic.scale * critic.raw_score
How do we define the "best" command?

Example Critics

Cost on the Costmap (Obstacles)
How do we define the "best" command? 
Example Critics

Cost on the Costmap (Obstacles)

Distance to Global Path (PathDist)
How do we define the "best" command?

Example Critics

- Cost on the Costmap (Obstacles)
- Distance to Global Path (PathDist)
- Distance to Goal (GoalDist)
How do we define the "best" command?

Example Critics

- Cost on the Costmap (Obstacles)
- Distance to Global Path (PathDist)
- Distance to Goal (GoalDist)
- Orientation To Goal (GoalAlign)
How do we define the "best" command?

Example Critics

- Cost on the Costmap (Obstacles)
- Distance to Global Path (PathDist)
- Distance to Goal (GoalDist)
- Orientation To Goal (GoalAlign)
- Orientation To Path (PathAlign)
What is the best command velocity?

What command velocities are available?
Sampling search in dynamic window

What would the commands do?
Trajectory generation with kinematic model

How do we define the "best" command?
Critics implementing heuristic scoring functions.
ROS Navigation Today: Theory vs. Practice

**Theory:**
ROS Navigation is a universal black box that runs robot navigation for hundreds of robots.

**Practice:**
ROS Navigation was built nearly 9 years ago to control the PR2, and it working well for other platforms is almost incidental.
ROS Navigation Today
Current core local planners

**base_local_planner** - 2009 - Eitan Marder-Eppstein
Pre-turtle-names

**dwa_local_planner** - 2011/2012 - Eitan & Thibault Kruse
Diamondback
ROS Navigation Today
Sources of Data

Current Location
Current Velocity

Global Plan
Goal Location

Costmap2D

`nav_core/base_local_planner.h`
void initialize(string name, TransformListener* tf, Costmap2DROS* costmap_ros);
bool setPlan(vector<PoseStamped> plan);
bool computeVelocityCommands(Twist& cmd_vel);
bool isGoalReached();
ROS Navigation Tomorrow
Why the fork not?

- Testable
- Customizable
- Backwards Compatible
- Clean
The structure of the classes in the code should match the conceptual pieces of the algorithm, which should match the ROS interfaces.
ROS Navigation Tomorrow
The Next Generation

github.com/locusrobotics/robot_navigation
ROS Navigation Tomorrow

**nav_core2**

```cpp
void initialize(string name, TransformListener::Ptr tf,
                 Costmap2DROS::Ptr costmap_ros);
void setPlan(Path2D path);
Twist2DStamped computeVelocityCommands(
    Pose2DStamped pose, Twist2D velocity)
bool isGoalReached(Pose2DStamped pose,
                   Twist2D velocity);
```

Error handling via Exceptions

Parameterized isGoalReached

More explicit computeVelocityCommands
ROS Navigation Tomorrow

dwb_local_planner

- Velocity Iterator & Trajectory Generator
- Trajectory Critic
- Goal Checker
void startNewIteration(Twist2D current_velocity);
bool hasMoreTwists();
Twist2D nextTwist();

Trajectory2D generateTrajectory(Pose2D start_pose,
Twist2D start_vel, Twist2D cmd_vel);
dwb_local_planner
TrajectoryCritic

void onInit();
bool prepare(Pose2D pose, Twist2D vel,
    Pose2D goal, Path2D global_plan);
double scoreTrajectory(Trajectory2D traj);
double getScale();
void debrief(Twist2D cmd_vel);
dwb_local_planner
plugins. plugins everywhere.

CostmapLayer
TrajectoryGenerator
TrajectoryCritic
GoalChecker
[dwb_msgs/GenerateTrajectory.srv]:
geometry_msgs/Pose2D start_pose
nav_2d_msgs/Twist2D start_vel
nav_2d_msgs/Twist2D cmd_vel
---
dwb_msgs/Trajectory2D traj
  nav_2d_msgs/Twist2D velocity
duration duration
geometry_msgs/Pose2D[] poses
dwb_local_planner
Debug Local Plan

[dwb_msgs/LocalPlanEvaluation.msg]:
std_msgs/Header header
dwb_msgs/TrajectoryScore[] twists
dwb_msgs/Trajectory2D traj
dwb_msgs/CriticScore[] scores
  string name
  float32 raw_score, scale
  float32 total
uint16 best_index, worst_index
ROS Navigation Tomorrow
Backwards Compatibility

- Interface Compatibility
  - Use `nav_core2` local planners as a `nav_core` plugin using `nav_core_adapter/local_planner_adapter`

- Parameter Compatibility
  - By default, will load plugins needed to replicate `dwa_local_planner`.
DWB Example
Precise Plan Following
DWB Example
Drunken Path Planning