Motion Planning for Autonomous All-Terrain Vehicle

Guan-Horng Liu, Samuel Wang, Shu-Kai Lin, Chris Wang, Tiffany May
Advisor : Mr. George Kantor
OUTLINE

Platform Introduction

Motivation & Challenges

Problem Formulation

OMPL Framework

ROS Planning PipeLine
  Collision Check Module
  RRT-Based Planner Module

Final Demo
Platform Introduction

Project Overview
General purpose autonomous technology development for off-road driving in wilderness environment.

Vehicle Platform
2016 YAMAHA Viking VI side-by-side ATV

On-board Sensors
Novatel, Velodyne 64, Multisense S21

Software Modules
Classification, Pose Estimation, Global Planner, Local Planner

Parter
Field Robotics Center
YAMAHA Motor Company, Japan
YAMAHA Motor Corporation, USA
Motivation & Challenges

Motivation
Propose a new local planner for off-road navigation with
- Static obstacles avoidance
- High-speed maneuvering in complex vehicle dynamic

Kinodynamic Planning in Control Space

Challenges
Uncertainty in Vehicle Dynamic Response Modeling
e.g. Wheel-terrain interaction...

Real-Time Implementation
e.g. Anytime planning, Computational efficiency
Problem Formulation

Testing Scenario Design
S-shape maneuvering with static obstacles avoidance
Vehicle velocity with at least 20kph
Model-based planner

Available Module/Sensor
YAMAHA Velocity Controller (YVCA)
LiDAR (point cloud), GPS/INS (position & velocity)
Show Time First
20kph
Platform Introduction

Motivation & Challenges

Problem Formulation

OMPL Framework

ROS Planning PipeLine
  Collision Check Module
  RRT-Based Planner Module

Final Demo
OMPL :: Framework

### OMPL Classes and Their Roles

- **ControlSampler**
  - Implements sampling of controls for a specific ControlSpace.
- **ControlSpace**
  - Represents the control space the planner uses to represent inputs to the system being planned for.
- **DirectedControlSample**
  - Sample controls that take the system towards a desired state.
- **StateSpace**
  - Represents the state space in which planning is performed; implements topology specific functions; distance, interpolation, state (de)allocation.
- **StateSampler**
  - Implements uniform and Gaussian sampling of states for a specific StateSpace.
- **StatePropagator**
  - Returns the state obtained by applying a control to some arbitrary initial state.
- **ProjectionEvaluator**
  - Computes projections from states of a specific StateSpace to a low-dimensional Euclidean space.
- **MotionValidator**
  - Provides the ability to check the validity of path segments using the interpolation provided by the StateSpace.
- **ValidStateSampler**
  - Provides the ability to sample valid states.
- **MotionValidator**
  - Provides routines typically used by motion planners; combines the functionality of classes it depends on.
- **SpaceInformation**
  - Provides a simple way of setting up all needed classes without limiting functionality.
- **StateValidityChecker**
  - Decides whether a given state from a specific StateSpace is valid.
- **OptimizationObjective**
  - Encodes a path cost that is used by optimizing planners.
- **ProblemDefinition**
  - Specifies the instance of the planning problem; requires definition of start states and a goal.
- **Path**
  - Representation of a path; used to represent a solution to a planning problem.
- **Goal**
  - Representation of a goal.

### User Code

- **User code**
  - User must instantiate this class.
  - User must instantiate this class unless SimpleSetup is used.
  - User can instantiate this class, but defaults are provided.
OMPL :: OMPL.app

RRT Planner

KPIECE Planner
OMPL :: Benchmark

Solving Time among multi planners

Path Difference among multi planners
Collision Check Module

Simplified Occupancy Grid

Mesh

RANSAC Segmentation

Height Map Algorithm

Implemented Approach
Collision Check :: Simplified Occupancy Grid

- Count only increases
- If ( count > threshold ) → obstacle
Collision Check :: Mesh

- Use Open Dynamic Engine (ODE) to cast ray from 4 wheels down to the mesh
- Calculate vehicle pitch and roll from 4 contact points
- Collision condition:
  - Pitch > threshold
  - Roll > threshold
  - Mesh interact with the ray between two wheels
Collision Check :: RANSAC Segmentation

- Use RANSAC to obtain a cloud fitting to the plane model
- Get point cloud outliers to extract obstacles
Collision Check :: Height Map Algorithm

- Calculate the height difference of multiple points within one grid
- If (height_difference > threshold)
  → It’s an obstacle
Collision Check :: Implemented Approach

- **Requirement:** Efficiency + Reliability
  - **Efficiency**
    1. Using height map algorithm as final approach for efficiency
    2. Bit-wise operation for faster multiplication
  - **Reliability**
    1. Using obstacle counter for more robust detection
    2. Dilate obstacle size
RRT-Based Planner Design

Why RRT?
- High dimensional planning w/ complex dynamic model
- Smooth maneuvering

6 DoF control input $u = \text{(forward & angular vel.)}$

$SE2 + R^3$ vehicle-frame velocity
RRT-Based Planner Design

1. Data-Driven Vehicle Dynamic Response Model
   Predictive Model: $\dot{x} = f(x, u, t)$

- **6 DoF state $x$**
  - $SE2 + R^3$ vehicle-frame velocity

- **2 DoF control input**
  - $u = (\text{forward} & \text{angular vel.})$
1. Data-Driven Vehicle Dynamic Response Model

2. Control Shooting Method w/ Random Control Duration
   Flexible constraints between connect and extend RRT
   Steering Mechanism Constraint in Control Space

Distance = $\frac{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}}{\sqrt{(\dot{x}_1 - \dot{x}_2)^2 - (\dot{y}_1 - \dot{y}_2)^2}}$
RRT-Based Planner Design

1. Data-Driven Vehicle Dynamic Response Model

2. Control Shooting Method w/ Random Propagation Steps

3. Sub-Optimization w/ Minimal Traveling Time
   One more step needed to reach RRT* in control space
   Connect extended node to minimum node

RRT-Based Planner Design

1. Data-Driven Vehicle Dynamic Response Model
2. Control Shooting Method
   w/ Random Propagation Steps
3. Sub-Optimization w/ Minimal Traveling Time
4. Replanning
   Solving loop in 20Hz, planner loop in 2Hz
   Estimate start state using dynamic model in 1.
   Path consistence among each planner loop

Update start, collision map 0.5 sec

0.05 sec

Publish the lowest cost path
Final Demo Video

Vehicle operation vel:
10 ~ 40 kph
Base line requirement:
20 kph
Final Demo here:
~ 30 kph