Planning Local Trajectories in Unreliable Environments using a Parallel Hierarchical Approach

1st Cristiano Souza de Oliveira
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Aldo von Wangenheim
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Planning feasible trajectories in low structured environments, where planning conditions can abruptly change, is one of the key challenges in autonomous ground vehicle navigation. Many techniques of feasible path discovery wore developed to overcome some of the problems in those scenarios, each bringing some advantages and setbacks. We propose an approach based on first allowing multiple path planning solutions to run in parallel and then choosing the right planned path based on defined criteria. We show that modifying the criteria or the set of local planners used to create the decision candidates can alter the behavior of the vehicle, enabling this approach to be adapted or to self-adapt to different scenarios. We compare this approach with adopting each local planner as a standalone solution both in terms of performance and of final planned path quality, where the parallel approach was able to outperform the standalone solutions in 87% of the experiments.**

***Index Terms*—path planning, trajectory planning, unstructured environments, off-road, local planning, parallel**

I. Introduction

A. Background

Planning feasible trajectories in low structured environments is one of the key challenges in autonomous ground vehicle navigation. Any unmanned ground vehicle (UGV) must at least account for immediate driving, such as perceiving and following roads or lanes, detecting obstacles and producing local paths capable of avoiding collision or performing emergency breaks.

A driving scenario may vary significantly in unstructured environments, affecting the perception conditions. In the same mission, depending on the type of road being followed, satellite positioning can be susceptible to blockage by trees or other natural elements, so the vehicle must rely on other types of navigation such as vision-based. In opened terrain satellite data can flow unblocked, but road boundaries may be undefined. Vision-based navigation can suddenly fail in the presence of raindrops, while radar data is unaffected. On the other hand, radar data has range limitations that can prevent a vehicle to drive in a desired speed. Thus, a single solution approach is very limiting, as planning conditions in off-road scenarios may vary abruptly.

It seems unlikely that any local path planning algorithm alone can account for all of the unexpected variations, which opens a path to try solutions based on multiple local planning approaches. That potentially raise concerns about the overall execution time, which may affect the driving experience. The current evolution of embedded systems, where multicore CPUs, GPUs and APUs, a combination of CPU and GPU on the same chip, are being available allow the execution of multiple processing threads in real parallel pipelines.

B. Related Work

In [3], a Determined Finite Automata (DFA) is used to select a local planning approach based on the vehicle perceived state. This state is determined by statistical analysis of vehicle states and environment sensor data. Typical conditions include the distribution of obstacles, the slope of the road and the current speed of the vehicle. To avoid oscillating planning mode switching, a probabilistic analysis on perception results within a continuous period is applied as the trigger condition.

In [1] the local planning is defined before the mission starts. In order to follow the planned path, a parallel approach is proposed as a navigation method based on a DCS (Distributed Control System). Several navigation modules are executed in parallel, and they independently control the robot by using magnetic and geometric landmarks.

C. Our work

Our research focuses on planning local trajectories that can adapt to the changing conditions in low structured environments, by executing several local planners while keeping a low response time.

When defining a local trajectory for an UGV, it is crucial that a maximum time constraint is followed, to allow the vehicle to perform a mission without risking it to navigate disoriented while waiting for planner to respond. Executing more than one local planner can be a problem if the first executor is unable to find a feasible path due to operation outside boundary conditions. If another local planner must now takeover, that means that the time frame for total execution is

Fig. 1. Architecture of the parallel hierarchical execution

Fig. 2. Architecture of the parallel hierarchical execution

the sum of the two stacked planning solutions present in the system. If many solutions are present, them the total execution time is the sum of all of them in the worst case scenario.

In order to use several local planners while following the time constraint, our approach relies on executing them in parallel and then selecting the final resulting path based on some metrics and on execution hierarchy.

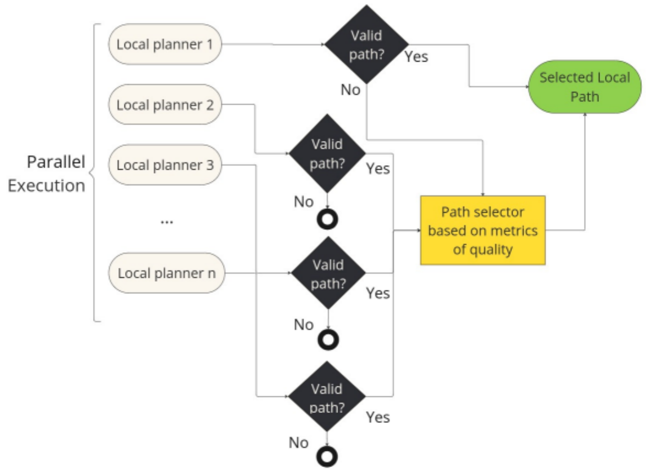
The main advantage of this approach is that a new local solution that responds to a specific set of boundary conditions can be stacked with the current known ones, expanding the vehicle's ability to navigate in rough conditions, while the time execution constraint can still be followed. Similarly, a solution based on machine learning can be stacked to learn from the best selected local solutions at any time, working as a self-adapting mechanism for unexpected future situations.

II. Overview

In this section, we present an overview of the system architecture. The goal of a local path planner is defined as to produce a feasible, ideally optimal and smooth, path from data gathered by environment perception, which complies with driving limitations. This data is often stored in some data structure which we define here as data representation.

Assuming that any local planner x in the local planner set can use the same data representation format and all planners are able to read the perception data in parallel, which implies that there will be no serial access due to mutex or semaphore locking, then the maximum execution time is equals to the execution time of the slower local planner in the set. Assuming that every local planner in the set is preemptive, which means that the current execution can be stopped at any time, which implies that a non-optimal or a null response can be given, then the maximum execution time is equal to a timeout threshold.

$$\begin{aligned} &max\_time = max(exec\_time(x), x \in local\_planners) \\ &exec\_time = min(max\_time, TIMEOUT) \end{aligned} \tag{1}$$

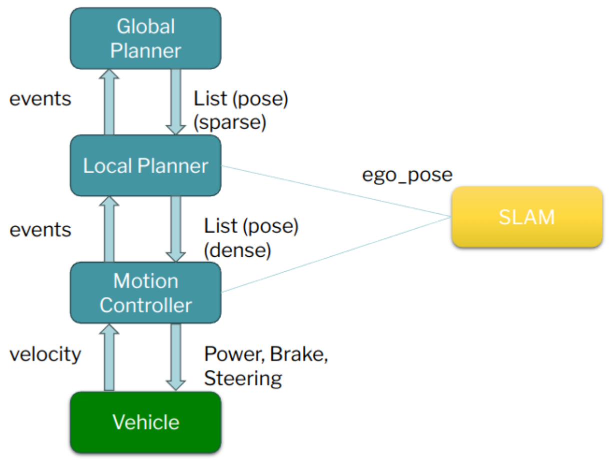


After the execution time, it is possible that more than one local planner was able to provide a feasible path. As shown in 1 an hierarchy can provide an order of preference, where a planner's response will always be picked if feasible. If not, a metric system can be defined to select the best response among the other planners for the current scenario. If any local planner reached a valid path, that means that a blockage is found and recovering measures must take place.

Recovering from an invalid path means finding alternative routes to reach the goal. This task is not reachable from the perspective of the local planning, since the goal should not be on the vehicle's vicinity. A full stop is then expected here, while another system module, often called The Global Planner, which have access to offline data such as a digital map, tries to find a new route. If this task also fails, then driving is not possible anymore and a human operator must intervene.

III. Testing the Architecture

To test our solution, we adopted a three-layer architecture: a simple Global Planner provides a fixed list of goal points, being unable to plan or to re-plan global paths; a Local Planner, which implements our parallel hierarchical local planning approach with different combinations of local planner executors and a Motion Controller, which is responsible to make the vehicle follow the selected local path by using feedback mechanisms to correct variations between the planned trajectory and the real resulting motion. Our motion controller uses a PID longitudinal controller and the Stanley lateral controller [2] for error correction. Additionally, a simple SLAM module is responsible for finding the current vehicle (ego) pose using GNSS positioning, odometer, IMU and compass data. Those data are fused using a Kalmann Filter [4] approach. We define a pose here as a 3-tuple (latitude, longitude, heading). Even though the Global Planner does not plan or re-plan paths, it still receives invalid path events. The resulting action for this event during or testing is to simply perform an emergency stop.



Experience shows that depending on the choice of a data representation model, it may affect the availability of already implemented local planning executors solutions. We've decided to adopt the Occupancy grid (OG) as the data representation, since a Systematic Review we've performed of the last 12 years of path planning in low and non-structured environments pointed out that it is the mostly adopted approach. Therefore, we've chosen the algorithms in table 1 to compose this testing.

IV. Results and Discussion

V. Conclusion