An Architecture for Automated Driving in Urban Environments

Gang Chen, Thierry Fraichard

An Architecture for Automated Driving in Urban Environments

Gang Chen and Thierry Fraichard

Inria Rhône-Alpes & LIG-CNRS Lab., Grenoble (FR)
firstname.lastname@inrialpes.fr

Summary. This paper presents a novel navigation architecture for automated car-like vehicles in urban environments. Motion safety is a critical issue in such environments given that they are partially known and highly dynamic with moving objects (other vehicles, pedestrians...). The main feature of the navigation architecture proposed is its ability to make *safe* motion decisions *in real-time*, thus taking into account the harsh constraints imposed by the type of environments considered. The architecture is based upon an efficient publish/subscribe middleware system that allows modularity in design and the easy integration of the key functional components required for autonomous navigation, namely perception, localisation, mapping, real-time motion planning and motion tracking. After an overall presentation of the architecture and its main modules, the paper focuses on the "motion" components of the architecture. Experimental results carried out on a simulated Cycab vehicle are presented.

1 Introduction

Autonomous navigation of intelligent vehicles in urban environments requires to solve a number of challenging problems in domains as different as perception, localization, environment modelling, reasoning and decision-making, control, *etc.* The problem of designing and integrating these functionalities within a single navigation architecture is of fundamental importance. Since Shakey's pioneering attempts at navigating around autonomously in the late sixties [1], the number and variety of autonomous navigation architectures that have been proposed is large (see [2]). From the motion determination perspective, these navigation architectures can be broadly classified into *deliberative* (*aka motion planning-based*) versus *reactive* approaches: deliberative approaches aim at computing a complete motion all the way to the goal using motion planning techniques, whereas reactive approaches determine the motion to be executed during the next time-step only. Deliberative approaches have to solve a motion planning problem [3]. They require a model of the environment as complete as possible and their intrinsic complexity is such that it

may preclude their application in dynamic environments: indeed, the vehicle has only a limited time to determine its future course of action (by standing still for too long, it might be collided by one of the moving objects). Reactive approaches on the other hand can operate on-line using local sensor information: they can be used in any kind of environment whether unknown, changing or dynamic. This accounts for the large number of reactive approaches that have been developed over the years, *eg* [4, 5, 6, 7], *etc.* Most of today's reactive approaches however face a major challenge: as shown in [8], motion safety in dynamic environments is not guaranteed (in the sense that the vehicle may end up in a situation where a collision inevitably occurs at some point in the future).

The primary contribution of this paper is a motion planning module that takes into account these two constraints, namely the *real-time* and *safety* constraints. It is achieved thanks to the two concepts of Partial Motion Planning (PMP) [9] and Inevitable Collision States (ICS) [10]. PMP is a planning scheme that takes into account the real-time constraint explicitly by generating partial motions iteratively at each time-step. Like reactive decision schemes, PMP faces the safety issue. ICS are called upon to address this issue. An ICS is a state for which, no matter what the future trajectory followed by the vehicle is, a collision with an object eventually occurs. For obvious safety reasons, a vehicle should never end up in an ICS. By computing *ICS-free partial motion* at each time-step, the vehicle safety *can be guaranteed* in real-time.

The secondary contribution of this paper is a presentation of the navigation architecture hosting the PMP-ICS motion planner. It is based upon an efficient publish/subscribe middleware system named DDX [11] that allows modularity in design and the easy integration of the key functional components required for autonomous navigation, namely perception, localization, world modelling, motion planning and motion tracking.

The paper is organised as follows: an overall description of the navigation architecture is given first in §2. The three layers of this architecture are respectively detailed in §3–5. Experimental results are finally presented in §6.

2 Navigation Architecture Overview

The architecture presented in this paper is a DDX-based real-time modular architecture. DDX is a publish/subscribe middleware [11] that is used to provide the navigation modules with an abstract view of the Cycab, its sensors and its environment. Fig. 1 depicts the overall architecture. Below the DDX layer is the Cycab layer: it features the Cycab, its sensors and the environment either in simulation or for real. Above the DDX layer is the Navigation layer: it features the different key modules required for autonomous navigation: localization, world modelling, motion planning and motion tracking (these modules are detailed in §5). To complete the architecture, a Geographic Information

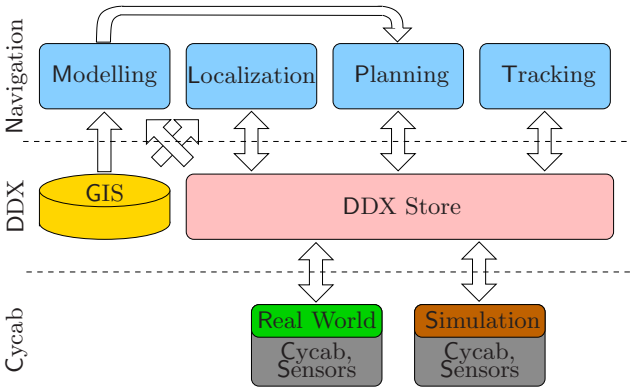


Fig. 1. Functional view of the navigation architecture.

System (GIS) is used to provide static information about the environment (road geometry and topology, traffic signs and traffic rules...). The dynamic information about the environment (other vehicles, pedestrians...) is computed by the Cycab layer through sensor-data processing (or directly by the simulator). The following sections describe the Cycab, the DDX and the Navigation layers respectively.

3 Cycab Layer

Fig. 2. The Cycab vehicle (left) and the Cycab simulator (right).

The Cycab vehicle is a lightweight urban electric vehicle which is specifically designed for downtown areas (Fig.2). It integrates four motor wheels and a motorized mechanical jack for steering. Micro-controllers are used to control the motor-wheels and the steering mechanism. An embedded PC under Linux RTAI is used for the overall control of the vehicle. Two CAN (Controller Area Network) buses are used for communication between the different

hardware components of the vehicle. It can be equipped with various sensors such as GPS, IMU, video cameras and range sensors (see http://www-lara.inria.fr/cycaba). A Cycab Simulator has been designed to facilitate the design and test of the automated driving algorithms that will be implemented on the real Cycab. It is based upon the MGengine simulation engine (http://mgengine.sourceforge.net) and permits the simulation of the Cycab vehicle, its sensors and its environment (see http://cycabtk.gforge.inria.fr).

4 DDX Layer

Table 1. DDX Store: who is using what?

Module	Input/Output Data
Localization	Input: CycabPose, CycabState, GIS, Landmarks Output: CycabPose
World Modelling	Input: GIS, Moving Objects Output: Future Model
Motion Planning	Input: Future Model Output: Trajectory
Motion Tracking	Input: CycabPose, Trajectory Output: CycabCommand

DDX provides an efficient communication mechanism to allow multiple processes to share data. It is implemented as a *Store*, *ie* a block of shared memory (possibly distributed over several computers), that is used to store shared information. The *Catalog* function is used to ensure the coherence of the information contained in the different stores (using the UDP/IP communication protocol). As far as the navigation architecture proposed is concerned, the data contained in the DDX store comprises four main data structures concerning either the Cycab or its environment:

- **Cycab:** information concerning the Cycab:
  - *CycabState*: encoder values, wheel velocities.
  - *CycabCommand*: actuator commands (speed, steering angle).
  - *CycabPose*: position and orientation of the Cycab.
- **Landmarks:** position of the observed landmarks, *ie* the salient features of the environment used for absolute localization.
- **Trajectory:** nominal trajectory that is to be executed by the vehicle (see §5.2). It is a sequence of (state, time) couples.
- **Moving objects:** list of the moving objects observed in the environment. Each moving object is characterized by it shape, position, orientation and velocity.

Table 1 summarizes how these data structures are used by the different navigation modules. The GIS data used by Localization is the list of the landmarks' position. World Modelling on the other hand gets the road geometry from GIS. Future Model is a description of the current state of the environment (fixed and moving objects) plus a prediction of the future motion of the moving objects (see §5.1).

5 Navigation Layer

This section presents the main modules used for autonomous navigation. There are four of them: *World Modelling* and *Localization* that deals with building a model of the vehicle's environment and localizing the vehicle inside this model. *Motion Planning* and *Motion Tracking* respectively deals with computing and executing a trajectory. *World Modelling* and *Motion Planning* are described in the next two sections. Due to lack of space, *Localization* and *Motion Tracking* are not described here, the reader is referred to [12] and [13] instead.

5.1 World Modelling

The primary purpose of the World Modelling module is to build a model of the environment of the vehicle that can be used for autonomous navigation purposes. Road-like environments feature both *fixed objects* (such as the road limits) and *moving objects* (such as other vehicles and pedestrians) and the World Model must represent them both.

Static information about the environment, *eg* limits of the roadway, geometry of the obstacles, are assumed to be known a priori and made available in a Geographic Information System (GIS). For the purpose of navigation in urban environments, a 2D map of the environment (*ie* a set of polygonal obstacles) suffices (Fig. 3-top right). The structure of the roadway is also included in the GIS: it is represented as oriented lanes connected together in a network. This structure is exploited by the Motion Planning module to determine the route that is to be followed in order to reach a given goal.

In the architecture proposed, it is assumed that the information about the moving objects (linear and angular velocity/acceleration, *etc.*) is a direct output of a sensor-processing step corresponding to the different sensors used. In other words, the detection and tracking of the moving objects is performed in the Cycab layer, not the Navigation layer. Experiments have been carried out using video cameras and laser range sensor, and the tracking techniques presented in [14] and [9] (Fig. 3-bottom left).

Finally, noting that motion planning involves a certain degree of reasoning about the future, motion prediction regarding the future behaviour of the moving objects is required. In the architecture proposed, motion prediction

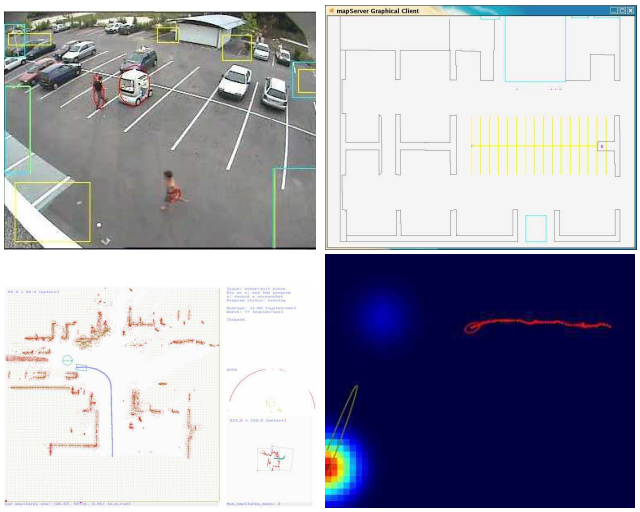


Fig. 3. View of the parking lot (top left), of the corresponding map (top right), of the moving objects tracking process (bottom left), and of the prediction of the future motion of one moving object (bottom right).

relies upon the assumption that pedestrians and vehicles do not move randomly but follow typical "motion patterns" which may be learned and then used in a prediction phase (Fig. 3-bottom right; see [15] for details on the prediction scheme).

5.2 Motion Planning

The Motion Planning module is the key component of the solution proposed for motion autonomy in dynamic environments. Its purpose is to compute the trajectory that is to be executed by the vehicle in order to reach its goal. The Motion Planning module takes as input the model of the future provided by the World Modelling module, computes a trajectory and places it into the DDX store where it is available for the Motion Tracking module.

When placed in a dynamic environment, a vehicle cannot stand still since it might be collided by one of the moving objects. In a situation like this, a *real-time constraint* is imposed to the vehicle: it only has a limited time to determine its future course of action. The time available is a function of what is called the *dynamicity* of the environment which is directly related to the dynamics of both the moving objects and the robotic system. *Partial Motion Planning* (PMP) is a planning scheme that takes into account the real-time constraint explicitly: when the time available is over, PMP is interrupted and it returns a partial motion, *ie* a motion that may not necessarily reach the goal. Of course, since only a partial motion is computed, it is necessary to

iterate the partial motion planning process until the goal is reached. The iterative nature of PMP is doubly required since the model of the future is based upon predictions whose validity duration is limited in most cases. An iterative planning scheme permits to take into account the unexpected changes of the environment by updating the predictions at a given frequency (which is also determined by the environment dynamicity). Let us consider the PMP cycle starting at time t_i, it comprises three steps:

(1) An updated model of the future is acquired (provided by the World Modelling module).
(2) The state-time space of the vehicle is searched using an incremental exploration method that builds a tree rooted at the state $s(t_{i+1})$ with $t_{i+1} = t_i + \delta_p$ where δ_p is the planning time available.
(3) At time t_{i+1}, the current cycle is over, the best partial trajectory $\Pi(i)$ of the tree is selected according to a given criterion (safety, length, *etc.*). It is discretized and placed into the DDX store.

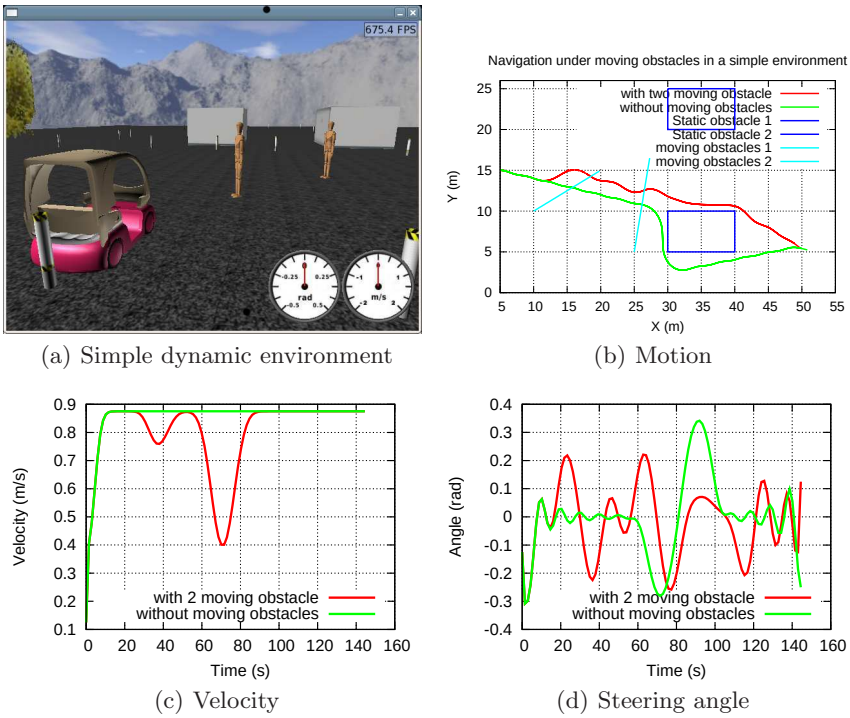
PMP cycles until the last state of the planned trajectory reaches a neighbourhood of the goal state. An incremental search method is used to explore the state-space. It is based upon the Rapidly-Exploring Random Tree (RRT) technique [3] that incrementally expands a tree rooted at the start state. This method being incremental in nature, it can be interrupted at any time. Classically, RRT computes collision-free trajectories. In the approach proposed, the usual geometric collision-checker is replaced by an Inevitable Collision State-checker [16] that ensures that the vehicle will never end up in a situation eventually yielding a collision later in the future.

6 Experiments

The different modules of the navigation architecture are implemented in C++ under Linux. The DDX framework allows the different navigation functionalities/modules to be distributed over different computers. When the real Cycab is used, its embedded core software communicates with the rest of the application through wireless Ethernet. Experiments on autonomous navigation has been carried out in simulation. So far, only the localization and tracking modules have been tested on the real Cycab. Autonomous navigation experiments with the real Cycab are underway. Simulations for two different scenarios have been studied to test the real-time planning performance of PMP. The PMP cycle time has been set to $1s$ and the motion tracking cycle time is $50ms$.

6.1 Test Environment

The first experiment is done in a dynamic 2D environment (size $60 \times 30m$), featuring two static rectangular objects and two moving disk objects



(a) Simple dynamic environment (b) Motion

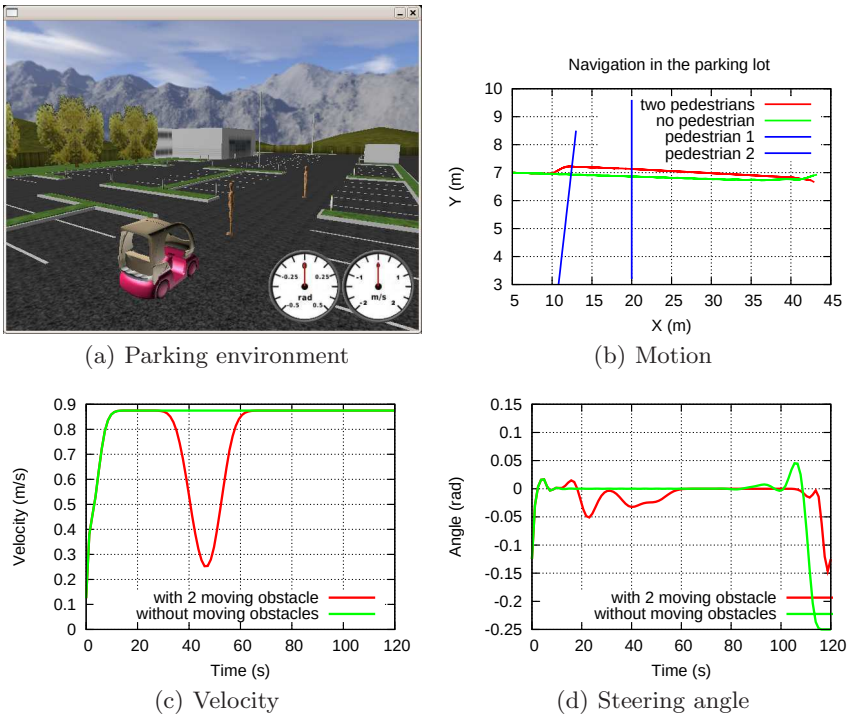
(c) Velocity (d) Steering angle

Fig. 4. Experiment in a test environment.

(Fig. 4(a)). The starting and the goal pose of the Cycab are $(5, 15, 0)$ and $(50, 5, 0)$ respectively. The moving objects are programmed to move with a constant velocity (moving upwards). Fig. 4(b) shows the setup of this experiment and the output of the motion planning process. The safe motion planned for the Cycab is the red line passing between the two rectangular objects. In comparison, the green line passing below the two rectangular objects is the trajectory obtained when the moving objects are not present. Figs 4(c) and 4(d) depicts the velocity and steering angle profile along both trajectories. The difference between the two trajectories is clearly due to the presence of the moving objects. Notice how the vehicle modify its course and slows down twice in order to give way to the moving objects.

6.2 Parking Lot of Inria Rhône-Alpes

The second experiment is done in a 2D model of the parking lot of Inria Rhône-Alpes (Fig. 5(a)). This environment is cluttered with twenty-six fixed objects and two pedestrians. From the motion planning point of view, this



(a) Parking environment (b) Motion

(c) Velocity (d) Steering angle

Fig. 5. Experiment in the parking lot of Inria Rhône-Alpes.

environment imposes more collision-avoidance constraints than the first scenario. The starting and the goal pose of the Cycab are $(5, 7, 0)$ and $(43, 7, 0.1)$ respectively. The pedestrians move upwards on the roadway. Fig. 5(b) shows the setup of this experiment and the output of the motion planning process. It also features the trajectory obtained when the moving objects are not present. Figs 5(c) and 5(d) depicts the velocity and steering angle profile along both trajectories. In this scenario, because of the extra constraint imposed by the fixed objects, the two trajectories are geometrically close (there is little room for manoeuvring). Most of the differences occur in the velocity profile.

7 Conclusions and Future Works

This paper has presented a novel navigation architecture for automated car-like vehicles in urban environments. The main feature of this navigation architecture is its ability to make *safe* motion decisions *in real-time*, thus taking into account the harsh constraints imposed by the type of environments considered (partially known with highly dynamic moving objects). Experimental

results carried out on a simulation platform in a parking environment has demonstrated the ability to navigate safely in dynamic environments. Future works will include further experiments with a real vehicle.

References

1. N. J. Nilsson, "Shakey the robot," AI Center, SRI International, Menlo Park, CA (US), Technical note 323, Apr. 1984.
2. I. R. Nourbakhsh and R. Siegwart, *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.
3. S. Lavalle, *Planning Algorithms*. Cambridge University Press, 2006.
4. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, 1986.
5. D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
6. O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *IEEE Trans. on Robotics and Automation*, Detroit, MI (US), May 1999, pp. 341–346.
7. J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, Feb. 2004.
8. T. Fraichard, "A short paper about safety," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome (IT), Apr. 2007.
9. R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Toward urban driverless vehicles," *Int. Journal of Vehicle Autonomous Systems*, In press.
10. T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.
11. P. Corke, P. Sikka, J. Roberts, and E. Duff, "DDX: A distributed software architecture for robotic systems," in *Proc. of the Australian Conf. on Robotics and Automation*, Canberra (AU), Dec. 2004.
12. C. Pradalier and S. Sekhavat, "Simultaneous localization and mapping using the geometric projection filter and correspondence graph matching," *Advanced Robotics*, 2004.
13. G. Chen and T. Fraichard, "A real-time navigation architecture for automated vehicles in urban environments," in *Proc. of the IEEE Intelligent Vehicles Symposium*, Istanbul (TR), June 2007.
14. D. Vasquez and T. Fraichard, "A novel self organizing network to perform fast moving object extraction from video streams," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing (CN), October 2006.
15. D. Vasquez, T. Fraichard, O. Aycard, and C. Laugier, "Intentional motion online learning and prediction," *Machine Vision and Applications*, In press.
16. R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma (IT), Apr. 2007.