Reactive motion planning in a dynamic world*

Th. Fraichard, M. Hassoun, C. Laugier†
IMAG-LIFIA
46, av. Félix Viallet, 38031 Grenoble Cedex, France

Abstract—This paper deals with the problem of planning and controlling the motion of a car like vehicle moving in a dynamic and roadway like environment. The contribution presented here is a motion controller which executes in a reactive way a given nominal motion plan. Such a plan is made up of a smooth trajectory $\mathcal{C}$ and of time constraints of the type "reach location l at time t_l". Data concerning the actual environment of the vehicle considered are assumed to be obtained through perception. In order to get the required reactivity, we have developed a motion controller with two main components: the *pilot* which analyses the current situation and adapts the nominal plan accordingly, and the *executor* which generates the required motion commands. The pilot operates at a symbolic level using a set of behavioural rules. The executor makes use of a potential field approach to generate the motion commands.

1 Introduction

1.1 The problem

This paper deals with the problem of planning and controlling the motions of a non holonomic vehicle in a dynamic world—i.e. in a time varying environment including other vehicles and various static and moving obstacles whose locations and behaviours are partly known beforehand. The framework of this research program is the European Prometheus Eureka project whose purpose is to design new cars and new road infrastructures to solve some critical points of today's traffic—mainly safety and congestion problems. A system including a *motion planner* (made up of a smooth trajectory planner and a temporal planner) and a *motion controller* has thus been developed. The whole system architecture and the temporal planner are presented in [3]. This paper focuses on the motion controller which deals with the reactive execution of the nominal plan periodically provided by the motion planner to the vehicle. This nominal plan is made up of a geometric trajectory and of an associated set of time constraints of the type "reach location l at time t_l". Data concerning the actual environment of the vehicle considered are assumed to be obtained through perception—by perception, we mean either direct perception through sensors or communication with the other vehicles. Since there may be a discrepancy between the hypotheses made at planning time and the actual environment, the nominal plan has to be adapted at execution time.

1.2 Our approach

Let M be the car like vehicle considered, let $\mathcal{W}$ be the actual environment—i.e. a planar surface—and let $\mathcal{P}$ be the nominal motion plan periodically provided by the motion planner. $\mathcal{P}$ is made up of (1) a geometric trajectory—i.e. a curve $\mathcal{C}$ of $\mathcal{W}$—and (2) time constraints of the type "reach location l at time t_l"—i.e. t-uples (l, t_l). The purpose of the motion controller of M is to generate the appropriate motion commands which are required to execute $\mathcal{P}$ while constantly adapting the behaviour of M to the characteristics of $\mathcal{W}$ (unpredicted events in particular). The on line adjustment of $\mathcal{P}$ must obviously respect the set of behaviour rules (highway code) prevailing in $\mathcal{W}$. Since $\mathcal{W}$ is a dynamic environment, the commands to apply must simultaneously control the position, orientation and velocity changes. In order to satisfy the previous requirements, we have developed a hierarchical system with two main modules (see figure 1): the *pilot* which analyses the current situation and adapts the nominal plan accordingly, and the *executor* which generates the required motion commands.

The pilot enables M to react appropriately to unexpected events. It is implemented as a symbolic layer enforcing a set of behavioral rules. This approach consisting in combining a "rule layer" and a "process layer" to get an appropriate reactive behaviour has already been proposed in [2].

The executor represents the numerical layer which is needed to generate the required motion commands at every time interval δt. It has been implemented using a "dynamic potential field" which combines classical information about the task (distance to the current subgoal and to the static obstacles), but also information about the nominal plan (trajectory and time constraints) and data about the dynamic obstacles (position and velocity).

The paper is organized as follows: §2 and §3 respectively describe the pilot and the executor. Experimental results are finally discussed in §4.

*Work supported by the French Ministry of Research and Technology and the European Prometheus Eureka project.

†Research Director at INRIA

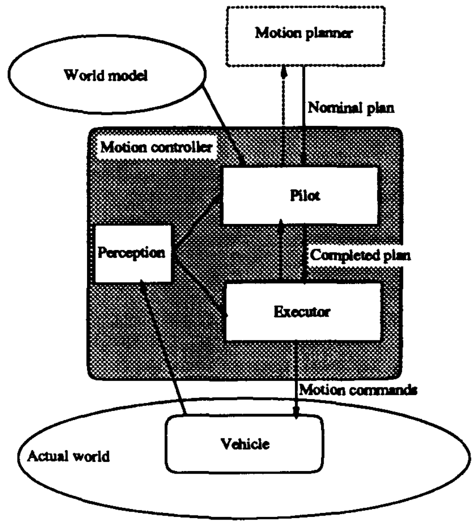

Figure 1: the system architecture

2 The pilot

2.1 General presentation

The pilot of the vehicle M can be seen as the symbolic layer of a reactive motion planner. Its main task is to adapt the nominal motion plan $\mathcal{P}$ to the characteristics of the actual environment of M. For that purpose, the pilot is activated through an asynchronous mechanism based upon the perception data and the feedback from the executor. For instance, the pilot is activated when a new obstacle is detected ahead of M, or when M is about to cross an intersection. or when the current velocity of M becomes very different of the nominal velocity. Once activated, the pilot analyses the current situation, checks for the validity of $\mathcal{P}$ and possibly adapt $\mathcal{P}$ using a set of behaviour rules. Since our system deals with road traffic, these behaviour rules stem from the highway code and take into account such things as the priority rules, the traffic lights, the speed limits and so on.

The inputs of the pilot are respectively $\mathcal{P}$, perception data about the actual environment of M and an a priori model of $\mathcal{W}$. $\mathcal{W}$ is a subset of the road network. As such, it is a highly structured environment; this structure is explicitly represented through a connected set of regions: *lanes*, *conflict* and *post conflict* regions. Basically the roadway is an organized set of *lanes*. A lane is characterized by a main spine along which the vehicles are supposed to move in a given direction. A *conflict region* represents the sub part of the roadway corresponding to the intersection of two or more lanes. It is clear that only one vehicle at a time can occupy such a region. This means that a vehicle heading for a particular conflict region r has to check for the availability of r before entering it. Besides, in order not to block the traffic flow, the vehicle must also check whether it is possible to cross r without stopping. In other words, there must be no obstacle right after r. Checking for such a condition led us to define appropriate regions called *post conflict regions*. In the symbolic model of $\mathcal{W}$, the connectivity of the roadway along with the associated motion constraints are represented using a labelled oriented graph structure. The information labelling the graph is either known a priori or obtained at execution time through perception (for instance, when detecting a beacon indicating the presence of a traffic light or informing the vehicle that it will soon cross a higher priority lane). In the current version of the system, a priori information about $\mathcal{W}$ is provided to the system. Information about unpredicted events and the surrounding vehicles (position and velocity parameters) is obtained using a simulation process.

The main output of the pilot is a "completed" motion plan $\mathcal{P}^+$ made up of: (1) the geometric trajectory $\mathcal{C}$, (2) time constraints (l, t_l) and (3) symbolic behaviour instructions of the type "stop at l", "overtake"... As we will see further, the symbolic instructions are used by the executor to tune some parameters of its potential field.

2.2 The approach

As explained earlier, the task of the pilot is to analyze the current situation and to adapt $\mathcal{P}$ accordingly when necessary. In a first step, the system must determine whether the event which activated the pilot will generate a failure or a degenerated execution of $\mathcal{P}$—i.e. a collision, a non satisfied time constraint or an excessive delay. The second step of the reasoning is applied when a local adjustment of $\mathcal{P}$ must take place. It leads the pilot to temporarily modify the current symbolic instruction of $\mathcal{P}^+$ and/or the next time constraint (l, t_l). The reasoning is based upon a process which simulates the execution of the planned maneuver. This simulation process assumes that the behaviour of the other vehicles will remain constant in the near future.

Five types of behaviour have been considered in the current implementation of the system: normal and cautious—i.e. when approaching an intersection—road driving, overtaking, stopping at a given location and crossing an intersection. Each behaviour is characterized by a set of preconditions, a set of symbolic instructions and a set of postconditions indicating that the symbolic instructions have been carried out.

For instance, the preconditions of an overtaking maneuver are: $(M\,follows\,X) \wedge (|v - v_d| < k_1) \wedge (delay > k_2)$ where v and v_d are respectively the current and the desired velocity of M, *delay* is the delay associated with the nominal time constraints and X is an other vehicle moving in the same lane as M. The symbolic instruction "Overtake" added to $\mathcal{P}^+$ will have several effects at the executor level: the attractive potential field associated with the nominal trajectory $\mathcal{C}$ will be weakened and the desired velocity v_d will be increased in order to overtake as fast as possible. The postcondition associated with the overtaking maneuver is characterized by: $d(M, \mathcal{C}) < \epsilon$ where d is the euclidean distance.

The crossing intersection behaviour is activated by conflictual situations—i.e. when a conflict region aimed by M is already occupied. Solving such conflicts requires in general to introduce waiting delays in $\mathcal{P}^+$: M must stop before a conflict region or behind a preceding vehicle and wait until the involved conflict and post conflict regions are free. A consequence of such a behaviour is usually to generate a time delay Δt incompatible with the next time constraints of $\mathcal{P}$. An obvious solution to this problem is to ask the motion planner for a new set of time constraints. This is not implemented yet in the current version of the system; we simply add Δt to the next time constraints.

3 The executor

3.1 General presentation

The executor of the vehicle M can be seen as a motion commands generator. At every time interval δt, the executor has to generate the position, orientation and velocity changes which are required first to follow as closely as possible the nominal trajectory, second to satisfy the associated time constraints and finally to avoid any collision with the close obstacles.

The inputs of the executor are respectively the completed motion plan $\mathcal{P}^+$ and a description of the local environment of M. The description of the local environment of M includes an appropriate representation of the geometry and the instantaneous position and velocity parameters of each obstacle surrounding M (the static obstacles are obviously represented using only geometric and position characteristics). It is assumed that all relevant information about the surrounding obstacles is provided by the perceptive system of M. A static obstacle is represented using a simple bitmap description as in [1]. The model of a dynamic obstacle is completed by a vector representing its instantaneous velocity.

The main output of the executor is a motion command. Let us see what is a motion command. The natural control parameters of a car like vehicle are the velocity (throttle and brake) and the steering angle (driving wheel). If we assume M to be front wheel driven, these control parameters can be characterized by the pair (v, ϕ) where v and ϕ are respectively the module and the orientation of the instantaneous velocity $\vec{v}$ applied at the front axle midpoint F (see figure 2). In order to deal with the dynamic aspect of the vehicle's motions, we chose to use the time derivatives of v and ϕ to represent the motion commands: $(\dot{v}, \dot{\phi}) \in [-\dot{v}_{max}, +\dot{v}_{max}] \times [-\dot{\phi}_{max}, +\dot{\phi}_{max}]$ where $\dot{v}_{max}$ and $\dot{\phi}_{max}$ are respectively the maximum acceleration and the maximum angular velocity associated with the steering angle of M.

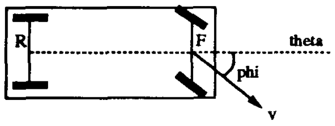

Figure 2: a car like vehicle

Another output of the executor is a feedback to the pilot about the current status of M—non satisfied time constraints, excessive delay...

3.2 The approach

Let $\mathcal{K} = [-\dot{v}_{max}, +\dot{v}_{max}] \times [-\dot{\phi}_{max}, +\dot{\phi}_{max}]$ be the control space of M. The problem to be solved by the executor is to select at time t the best motion command $(\dot{v}, \dot{\phi})$ to be applied to M during the time interval $[t, t']$ where $t' = t + \delta t]$. Selecting such a command requires in theory to analyze the effects of all the motion commands allowed at time t; in practice, this is achieved by reasoning about a discretized representation of $\mathcal{K}$. In the current implementation of the system, we use the following discretization: $\{-\dot{v}_{max}, 0, +\dot{v}_{max}\} \times \{-\dot{\phi}_{max}, 0, +\dot{\phi}_{max}\}$. Let $(q, \vec{v})$ be the instantaneous state of M at time t; q represents the current configuration of M—i.e. the t-uple (x, y, θ) where (x, y) are the coordinates of the rear axle midpoint R and θ is the orientation of the main axis of M—and $\vec{v}$ is the instantaneous velocity of M. Let $(\dot{v}, \dot{\phi})$ be a motion command chosen in the previous representation of $\mathcal{K}$. Assuming that $(\dot{v}, \dot{\phi})$ is instantaneously executed by M (this assumption is consistant since δt is small) then the state of M during the time interval $[t, t']$ is characterized by the pair $v' = v + \delta t.\dot{v}$ and $\phi' = \phi + \delta t.\dot{\phi}$. Computing the configuration q' of M at time t' is carried out using the kinematic equations of M.

In order to guide the choice of the next motion command to be applied, the executor makes use of a potential field U. Since $\mathcal{W}$ is time varying, U depends on three parameters: the time t and the instantaneous configuration q of M and the instantaneous velocity $\vec{v}$ of M. Thus U is a combination of three terms U_1, U_2 and U_3 whose purpose are respectively to guide M along the nominal trajectory, to avoid the dynamic obstacles and to satisfy the time constraints. We refer the reader to [4] for a complete presentation of the potential fields U_1, U_2 and U_3. Being given U, our algorithm to select the next motion command to be applied at time t is:

1. Compute all the pairs $(q', \vec{v}')$ obtained after application of the allowed motion commands $(\dot{v}, \dot{\phi})$. Any motion command yielding a velocity $\vec{v}')$ which does

not verify the kinematic constraints of M is discarded (M is subject to constraints on its acceleration, its steering angle and its centrifugal acceleration).

2. Choose the pair $(q', \vec{v}')$—and the associated motion command $(\dot{v}, \dot{\phi})$—which minimizes the value of U without generating any collision in $\mathcal{W}$.

4 Experiments

A prototype of the motion controller is currently being developed and tested. It is implemented in C on a SPARC Sun workstation and it includes a simulator for a car like vehicle. The executor is almost totally implemented. The pilot includes the five behaviours presented earlier but only the overtaking and intersection crossing behaviours have been tested.

As mentioned earlier, $\mathcal{W}$ is represented by a bitmap. U_1 is also represented by a bitmap and is computed each time a new geometric trajectory $\mathcal{C}$ is provided to the motion controller. U_2 and U_3 are dynamic and must obviously be computed at every time step δt.

Several examples have been processed in simulation. Figure 3 presents some snapshots of the simulation of the motion of several vehicles within an intersection. The snapshot 1 represents the initial situation, the numbered black rectangles represent the six vehicles present in the intersection at time 0. The numbered white disks represent the goals of the vehicles. In this simulation, each vehicle is equipped with our control system. $\mathcal{W}$ is represented by a 500×500 bitmap. It takes about 30s to compute the bitmap associated with U_1 and the motion commands are generated every 0.5s.

Let us study for instance the behaviour of the vehicle 0. In the snapshot 2, the vehicle 0 begins to overtake the vehicle 1 because the speed of the vehicle 1 is too slow. The attraction associated with the nominal trajectory of the vehicle 0 has been weakened and its velocity will increase (this is visible in the snapshot 3). At the end of the overtaking maneuver, the attraction associated with the nominal trajectory is set back to its normal value and the vehicle 0 then crosses the intersection following its nominal plan. As for the vehicle 2, it is located in the lower priority road of the intersection. Therefore, the vehicle 2 gives way successively to the vehicles 4 and 0 before to cross the intersection (snapshots 3 and 4).

The experiments have shown that the system works fairly well. However the choice of the correct coefficients for the function combining U_1, U_2 and U_3 is so far empirical. It means that the introduction of a new behaviour requires to experiment new coefficients in order to get motion commands consistant with this behaviour. This point has to be improved in the future. An other problem stems from the choice of the discretization for $\mathcal{W}$. A finer discretization would provide the vehicles with a smoother behaviour but at the expense of the computation load. In order to improve this point, we are studying the possibility to "wire" the algorithms used to compute U.

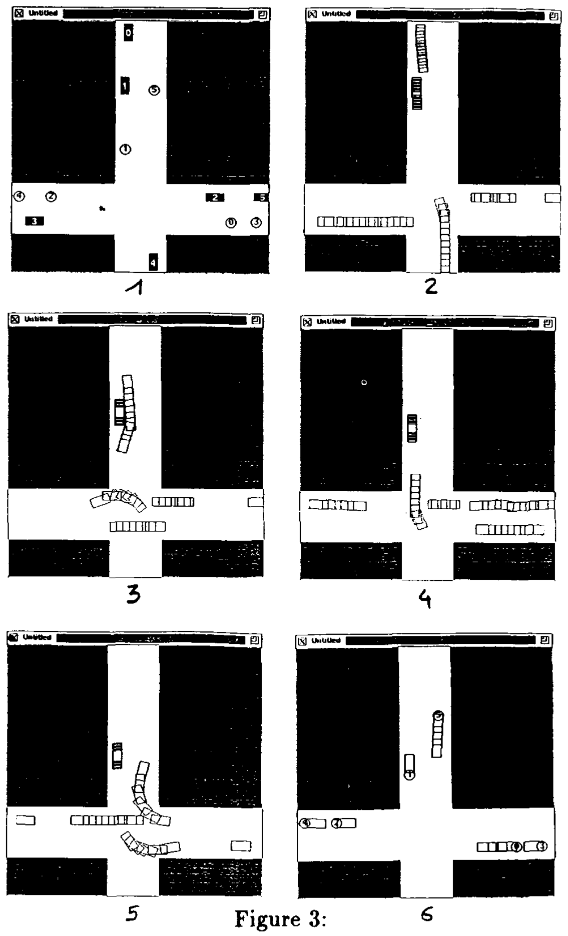

Figure 3:

References

[1] J. Barraquand and J-C Latombe. On non holonomic mobile robots and optimal maneuvering. *Revue d'intelligence Artificielle*, 3(2):77–103, 1989.

[2] P. Caloud. Distributed motion planning and motion coordination for multiple robots. Working paper, Computer Science Dept, Stanford Univ, Ca (USA), 1990.

[3] Th. Fraichard and C. Laugier. Planning movements for several coordinated vehicles. In *Proc. of the Int. Workshop on Intelligent Robots and Systems*, pages 466–472, Tsukuba (Japan), Sep. 1989. IEEE/RSJ.

[4] Th. Fraichard and C. Laugier. On line reactive planning for a non holonomic mobile in a dynamic world. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 432–437, Sacramento, Ca (USA), Apr. 1991. IEEE.