A Markov Decision Process-based approach for trajectory planning with clothoid tentacles

Hafida Mouhagir, Reine Talj, Véronique Cherfaoui, Franck Guillemard, François Aioun

A Markov Decision Process-based approach for trajectory planning with clothoid tentacles

Hafida Mouhagir[1,2], Reine Talj [1], Véronique Cherfaoui[1] , Franck Guillemard[2], François Aioun[2]

***Abstract*—The work presented in this paper focuses on reactive local trajectory planning which plays an essential role for future autonomous vehicles. The challenge is to avoid obstacles in respect to road rules while following a global reference trajectory. The planning approach used in this work is the method of clothoid tentacles generated in the egocentered reference frame related to the vehicle. Generated tentacles in a egocentered grid represent feasible trajectories by the vehicle, and in order to choose the right one, we formulate the problem as a Markov Decision Process.**

I. INTRODUCTION

The area of autonomous navigation has seen significant advances in recent years, specially thanks to the efforts of the scientific and engineering teams participating in the DARPA Urban Challenge [1], as well as other contests [2]. Concerning the trajectory planning, mainly approaches at a local on-road level have been inventoried and summarized in the survey [3], some of them are presented below.

The RTT algorithm (Rapidly exploring Random Trees) generates open-loop trajectories for nonlinear systems with state constraints. RRTs are suited for path planning problems that involve obstacles and non-holonomic constraints. This method ensures kinematic feasibility and can easily be implemented in real-time [4]. However, under the presence of many obstacles or heavy traffic, RRTs will check every possible collision for every expanded node which may lead to computational complexity.

Another local trajectory planning approach is Lattice planners [5] which is an extension of grid-based planners. It defines a state lattice that is discretized in all state parameters of interest, such as position, heading, curvature, and velocity. A trajectory generator is used to precompute feasible motions between states in the lattice in the absence of obstacles. The states define nodes in a graph, where edges describe motions. They are generally well-suited for non-holonomic and highly constrained environments, such as the road environment. Lattice planners are resolution complete. This means that the control space can automatically be adjusted for every resolution change and the space is explored consistently. Lattice planners also guarantee optimality and smoothness because they do not introduce discontinuities related to back-pointers but the main drawback was found to be curvature discontinuity.

At a local on-road level, one of the most popular technique is based on a search space which contains a certain geometric curve (e.g. clothoids or splines) and several lateral shifts of this curve [1][6][7]. Each candidate path is then evaluated through a cost function with several considerations, such as distance and time costs, acceleration and collision checking. The Tentacles method uses a set of virtual antennas that are called tentacles and an egocentric occupancy grid around the vehicle [8]. The occupancy grid expresses the state of the environment surrounding the vehicle and contains the obstacles if they exist. The tentacle is a geometric shape that models a possible trajectory of the vehicle and we can find in the literature multiple shapes of tentacles. In [8], the shape adopted for tentacles is circular. The weakness of this approach appears in considering all the tentacles generated for a certain speed as trajectory candidates even if their curvature is not well-suited to the current vehicle steering angle. An improvement has been made by introducing clothoid tentacles method [9]. The clothoid approach considers the current dynamical state of the vehicle and makes a smooth variations in the vehicle dynamic variables.

Decision making for autonomous driving is a challenging task due to the uncertainty of the complex environment surrounding the vehicle. Partially Observable Markov Decision Process (POMDP) [10], [11] and [12] offers a framework for autonomous robot navigation in dynamic environments. With this approach, the state of a car's environment can be estimated and the development of traffic situations can be predicted. Unfortunately, the implementation of these methods requires expensive collection of training data.

In this paper, we propose an original method based on trajectory planning with clothoid tentacle, to avoid obstacles and follow the reference trajectory, coupled with a decision process inspired from well known MDP (Markovian Decision Process) model.

The paper is organized as follows: Section II presents our trajectory planning strategy with the method of clothoid tentacles. In Section III, we detail the principal of Markov Decision Process and explain the proposed MDP like model for trajectory planning with clothoid tentacles. The simulation results based on data taken from Scaner-Studio simulator are discussed in Section IV . Finally, conclusions and perspectives are given is Section V.

The authors are with [1]*Sorbonne universités, Université de Technologie de Compiègne (UTC), CNRS Heudiasyc UMR* 7253, [2]*PSA Peugeot Citroën, Direction scientifique, Centre technique de Vélizy, France.* E-mail: {hafida.mouhagir, reine.talj, veronique.cherfaoui}@hds.utc.fr, {franck.guillemard, françois.aioun}@mpsa.com

II. Trajectory planning with the method of clothoid tentacles

Local trajectory planning is based on following a desired global reference trajectory defined on a global map while avoiding collisions over time by using the perception information that represent the environment. Our trajectory planning strategy can be divided into three main steps (Fig. 1):

- Creating and updating occupancy grid with data coming from exteroceptive sensors.
- Generating tentacles which will represent dynamically feasible trajectories.
- Choosing of the best tentacle that the vehicle will execute.

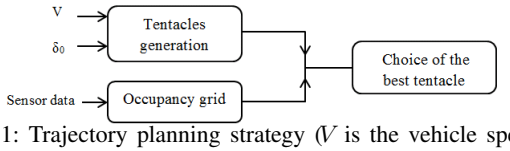


Figure 1: Trajectory planning strategy (V is the vehicle speed and δ_0 is the current steering angle)

A. Occupancy grid

The occupancy grid is a metric and discrete representation used in robotics algorithms such as path planning. This grid can be used to represent and visualize a vehicle workspace. The occupancy grid is built by a mapping process that integrate sensor data coming from exteroceptive sensor (Camera, Lidar, Radar) and the pose of the vehicle. They are used in mapping applications, as finding collision-free paths, performing collision avoidance, and calculating localization.

In our work, a 2D egocentered grid is constructed as a discrete representation of the environment around the vehicle by a set of square cells. Each cell contains information about the occupancy of the corresponding surface (Fig. 2).

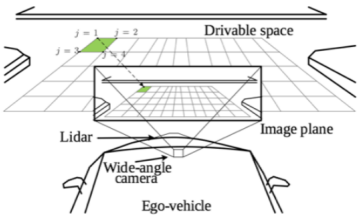


Figure 2: Occupancy grid principle

There are various frameworks used for creating and updating occupancy grid like Bayesian framework and evidential framework. Fig. 3 shows an occupancy grid [13] example based on evidential theory. The green color in the occupancy grid shows the navigable space, the red shows the occupied space, while the blue represents conflicting cells and the black represents unexplored cells. The color intensity reflects the certainty degree. In this paper, only binary grids are considered (free/occupied cells) but future work will take into account of the uncertainties modeled in the grid.

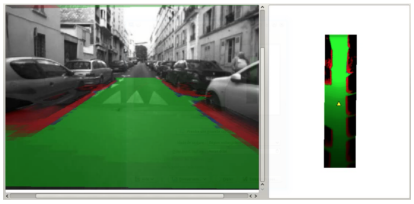


Figure 3: Occupancy grid example of a road

B. Generating tentacles

The method of tentacles consists in using a set of virtual antennas called tentacles in the egocentered reference frame related to the vehicle. Tentacles are a geometrical shape which models the dynamically feasible trajectories of the vehicle.

Several forms of tentacles exist:

1) Circular tentacles: In [8], tentacles are represented by arcs of circle. For a fixed speed, all the tentacles begin at the center of gravity of the vehicle and take the shape of arcs of circles. Every arc represents a trajectory with a specific steering angle. The most bent tentacles (those of the extremity) correspond respectively to the positive and negative maximal value of the steering angle which the vehicle can make at the current speed without losing stability. The length of tentacles increases with the increase of the speed (Fig. 4)

Figure 4: The tentacle's lengths increase with higher speed [8]

All the tentacles generated for a given speed are estimated as candidates for the execution of the movement, even if their curvature is not well adapted to the current steering angle of the vehicle.

2) Clothoid tentacles: Clothoid is a curve whose curvature varies linearly with curvilinear abscissa, also known as an Euler spiral, Cornu spiral or linarc. Its expression is presented by (1):

$$\rho = \frac{2}{k^2} s \tag{1}$$

where ρ is the clothoid curvature, s is the curvilinear abscissa and k is a constant, representing the clothoid parameter.

In [9] and [14], a set of tentacles is generated at each speed. All the tentacles are represented in the vehicle local coordinate system. They start at the center of gravity of the vehicle and take the form of clothoids.

We assume that all tentacles generated for a given speed V_x have the same length:

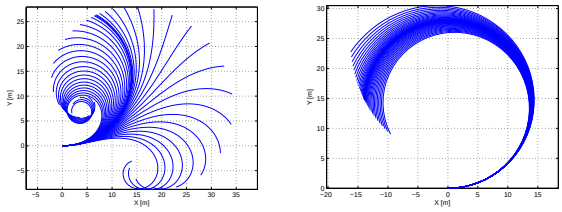
$$L_{tentacle}(m) = \begin{cases} t_0\, V_x - L_0 & V_x > 1(m/s) \\ 2(m) & V_x \leq 1(m/s) \end{cases} \tag{2}$$

where $t_0 = 7s$ and $L_0 = 5m$.

The initial curvature ρ_0 of the tentacles is calculated from the current vehicle steering angle δ_0 .

$$\rho_0 = \frac{\tan \delta_0}{L}$$

where L is the vehicle's wheelbase. The set of calculated clothoids depends on the vehicle speed and corresponds to feasible trajectories of the vehicle without losing stability till some defined distance (Fig. 5). A tentacle is considered navigable if there is no obstacle within the collision distance that corresponds to the distance needed to stop the vehicle with a comfortable deceleration.



(a) $V_x = 6\,m/s\,, \delta_0 = 0.3\,rad$ (b) $V_x = 10\,m/s\,, \delta_0 = 0.3\,rad$

Figure 5: Examples of set of clothoid tentacles

In our work, we use clothoid tentacles because this method considers the current dynamical state of the vehicle and make a smooth variations in the vehicle dynamic variables such as the yaw rate, the sideslip angle and the steering angle [14].

C. Choosing the best tentacle

For both circular and clothoid tentacles, after generating the tentacles and to guarantee a secure navigation, a security region is generated around each tentacle [8], [14]. This area takes into account the width of the vehicle, plus a security margin. Using the superposition of the security area with the occupancy grid, the navigable tentacles are obtained. The final step of the method is to choose the best tentacle from several navigable tentacles using different criteria. The best tentacle is then considered as a trajectory to execute. If there is no navigable tentacle, the authors choose the tentacle having the greatest distance to the first obstacle and they proceed to brake the vehicle with a constant deceleration along this tentacle.

Three criteria are used to choose one tentacle among navigable tentacles, the clearance criterion allows to estimate the distance to the first obstacle if it exists, the curvature criterion shows if the tentacle corresponds to a smoothly varying steering angle and the trajectory criterion reflects the closeness of the tentacle to the reference trajectory. The criteria are normalized to the interval $[0, 1]$ and then linearly combined into a single function to be minimized, namely,

$$V_{combined} = a_0 V_{clearance} + a_1 V_{curvature} + a_2 V_{trajectory} \tag{3}$$

where a_0, a_1 and a_2 are weighting parameters that can be used to prefer a criterion than another. Though the method is simple and easy to understand, it is difficult to adjust the weighted coefficients in different environments. To solve this problem, a geometrical ripple tentacles technique is used to choose a tentacle as a sub-optimal path [15].

In this paper, we propose to use a method inspired from Markov Decision Process model to choose the best tentacle.

III. Proposed MDP approach for trajectory planning with tentacles

A. Markov Decision Process principle

A Markov Decision Process (MDP) is a discrete-time state-transition system. The system is assumed to be in a state at any given time. The agent observes the state and performs an action accordingly. The system then makes a transition to the next state and the agent receives some reward.

It can be described formally with 5 components (S, A, T, R, γ):

- States: S is the set of states. The state usually captures the complete configuration of the system. Once the state of the system is known, the future of the system is independent from all previous system transitions. This means that the state of the system is a sufficient statistic of the history of the system.
- Actions : $A(s) : S \rightarrow A$ is the set of actions allowed in each state where A is the set of all actions.
- Transition Probabilities : $T : S \times S \times A \rightarrow [0, 1]$ defines the transition probabilities of the system. This function specifies how likely it is to end up at any state s', given the current state s and a specific action a performed by the agent.
- Rewards : $R : S \times A \rightarrow \mathbb{R}$. Depending on the current state of the system and the action taken, the agent will receive a reward drawn from this model.
- Discount Factor : $\gamma \subset [0, 1)$ is the discount rate used to calculate the long-term attenuation.

The agent starts in an initial state $s_0 \in S$. At each time step t, an action $a_t \in A(s)$ is taken by the agent. The system then makes a transition to s_{t+1} with the probability $T(s_t, s_{t+1}, a_t)$ and the agent receives an immediate reward $r_t \sim R(s_t, a_t)$. The goal of the agent is to maximize the discounted sum of rewards over the planning horizon h (which could be infinite). This is usually referred to as the value (V):

$$V = \sum_{t=0}^{h} \gamma^t r_t \tag{4}$$

There are several algorithms used to calculate an optimal policy in MDP problems, the most popular ones are: value iteration introduced in [16] and policy iteration introduced three years later in [17].

The MDP framework with completely known dynamics has been studied within artificial intelligence (AI) as a planning problem. AI planning thus reduced to finding a single action sequence taking the agent from a start state to a goal state. An example is shown in Figure 6. In this gridworld, states are presented by the grid cells and the primitive actions are steps up, right, and left, which usually cause the agent to move to the corresponding cell (but a third of the time they cause it to move to one of the other three neighbor cells).

In this planning problem, the agent is told that a positive reward can be obtained only at the goal state (cell a_4) and a negative reward is obtained at state (cell b_4). In this

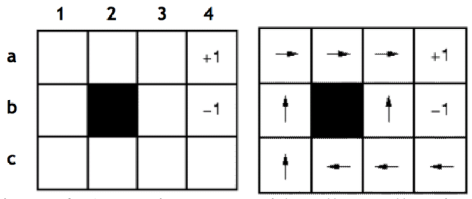


Figure 6: An environment with cell-to-cell actions

example (11 states and 3 actions), and with a policy iteration algorithm, it would take 3 iterations to find an optimal policy.

For autonomous vehicle trajectory planning strategy, the size of grid is around 800*800 cells and actions that could be executed by the vehicle are defined by the steering angle and/or acceleration. Hence, to formalize the planning problem with MDP the states are defined by the grid cells and actions as the variation of steering angle and/or acceleration. Due to numerous states and actions, the calculation time is not suited for real application. To reduce the complexity, we propose exploiting the possible trajectories defined by the tentacle method.

B. Proposed decision approach based on MDP

As described just before, we have to define the states of the system, the actions that can be taken, the transition matrix and the reward that the agent will receive for each action.

- States: are represented by circles s_i around the tentacles (Fig. 7), their diameter represent the width of the vehicle with a margin of security. Each tentacle is composed of n_s states, and we dispose of n_t tentacles.
- Actions: we dispose of n_t actions because each tentacle represents an action.
- Transition probabilities: in this work, we assume that we don't have a possible transition from one tentacle to another one. Thus we choose deterministic transitions; for all states s and all $a \in A(s)$ we assume that $p(s' \mid s, a) = 1$ for a unique $s' \in S$, while $p(s'' \mid s, a) = 0$ for all $s'' \neq s'$. Thus each action leads deterministically from one state to another (or the same) one.
- Reward: we will define a different reward for every state depending on its occupancy degree and its closeness to the reference trajectory. Moreover, we add a positive reward if the tentacle is one of the left side tentacles because the overtaking is always done on the left side of the vehicle that precede us, in order to avoid ambiguity in case of total symmetry of tentacles ($\delta_0 = 0$) when executing an overtaking manoeuver.

Since we have deterministic transition, the number of states can be reduced. After affecting for every state a reward regarding if it's free, occupied and close from the reference trajectory, we merge the states s_i of every tentacle in order to have just one state s_f per tentacle. Hence the total number of states s_f is n_t plus the initial state which is the actual position of the vehicle.

We consider a safety distance which corresponds to the distance travelled during two seconds with a given speed,

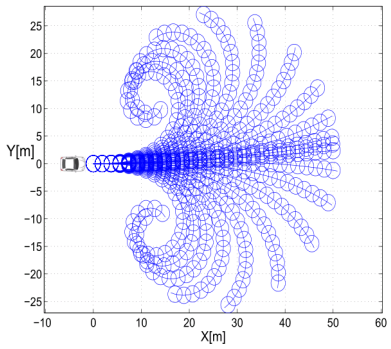


Figure 7: States of MDP model with clothoid tentacles

if a state of a tentacle is occupied within this distance, the tentacle is classified as non navigable. The next step of the method is to choose the best tentacle from several navigable tentacles to execute. If there is no navigable tentacle, we choose the tentacle having the greatest distance to the first obstacle and they proceed to brake the vehicle with a constant deceleration along this tentacle.

In order to define rewards, we use the occupancy grid.

Since there is no transition from a tentacle to another, we consider a tentacle as a state s_f, and we calculate its reward as following:

$$R(s_f) = \sum_{k=0}^{n_s} \gamma_t^k R(s_{i_k|trajectory}) + \sum_{k=0}^{n_s} \gamma_o^k R(s_{i_k|occupied}) + \sum_{k=0}^{n_s} \gamma_f^k R(s_{i_k|free}) + R(s_f \mid left) \quad (5)$$

where γ_t , γ_o and γ_f are discount factors that can be used to change the behavior of our approach, and that represent distance attenuation of each kind of reward.

To calculate rewards, we observe the cells of the occupancy grid that are superposed with the state. The state is considered as occupied if more than f_s (threshold) cells inside the state are occupied, otherwise it is considered as a free state.

In case of occupied state, the vehicle risks collision, so the corresponding tentacle receives a negative reward $R(s_{i_k|occupied}) = R_o$, and the left tentacles receive a positive reward $R(s_f \mid left)$ since the overtaking is done by the left. If the state is free, it receive a positive reward $R(s_{i_k|free}) = R_f$.

To calculate the reward to be attributed to states with regard to their closeness of the global reference trajectory, defined for example by GPS waypoints and a global map, we calculate the lateral displacement between the tentacles and the reference trajectory at different levels of the crash distance.

The crash distance l_c is the distance needed to stop a vehicle traveling with a speed V_x , with a maximum longitudinal deceleration $a_m = 1.5\, m/s^2$ that maintains passenger comfort, it is calculated by:

$$l_c = \frac{V_x^2}{2a_m} + l_s \quad (6)$$

where l_s is a security marge.

For each tentacle, a set of measurements d_i is calculated (Fig. 8) by taking both the lateral distance a_i and relative tangent orientations α_i between the tentacle and the reference trajectory [8]. Each d_i is calculated at a longitudinal distance $\kappa_i l_c$ from the vehicle position.

$$d_i = a_i + c_\alpha \alpha_i \tag{7}$$

here, c_α represents a scale between the linear distance and the tangent orientation.

We combine all the distances as follows:

$$d = \sum_{i=1}^{n} \lambda_i d_i \tag{8}$$

with λ_i is weighting constants.

All the states of the same tentacle receive a reward $R(s_{|trajectory})$ according to the distance d:

$$R(s_{|trajectory}) = R_{trajectory} - d \tag{9}$$

with $R_{trajectory} = R_t$, a fixed constant reward.

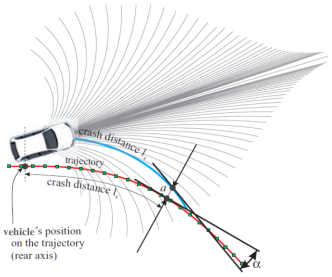


Figure 8: Distance between a tentacle and the reference trajectory [8]

The main advantage of the proposed algorithm is the significant number of freedom degrees which can allow us to change the behavior of the approach.

IV. Simulation Results

A. System set-up

In this part we report the simulation results based on data taken from Scaner-Studio simulator. The data taken from the Scaner-Studio was processed and simulated in Matlab. With these data, we generate a global map indicating the reference trajectory with its right and left borders and obstacles with red circles (Fig. 9). We indicated the navigable space of the road by black cells having the value "0" and the not navigable space by white cells having the value "1".

At every sampling period (100 ms in our case) the sensor provides a new measure and the occupancy grid, considered to be ego-centered around the vehicle, is built. This 2D occupancy grid is constituted of 800*800 cells. The size of each cell is 25cm* 25cm. As said before, grid mapping is not in the scope of this study, so we do not accumulate data and suppose that the frequency used for grid generation is sufficient to ensure safe navigation.

In our MDP-like model, states are represented by circles around the tentacles, their diameter is fixed to $3\,m$. Each

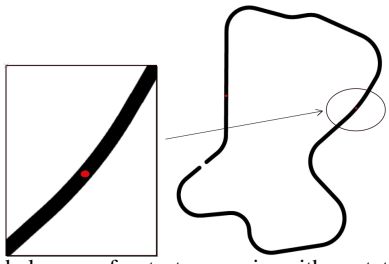


Figure 9: Global map of a test scenario with a static obstacle in Scaner-Studio simulator

tentacle is composed of $n_s = 16$ states, and we dispose of $n_t = 41$ tentacles. If a state is free, it receives a reward $R_f = 1$, if it's occupied it receives a negative reward $R_o = -50$ and we fixed $R_t = 30$ and $R_l = 0.5$. We calculate the distance between the tentacles and the reference trajectory at 3 different levels (κ_1, κ_2 and κ_3) of the crash distance, and we fixe λ_1, λ_2 and λ_3 in order to give more importance to near points of the tentacle.

The values of other parameters are in the table I below:

c_α	γ_t	γ_o	γ_f	κ_1	κ_2	κ_3	λ_1	λ_2	λ_3
0.7	0.99	0.95	0.99	$\frac{1}{10}$	$\frac{1}{2}$	1	10	2	$\frac{1}{3}$

Table I: The values of various parameters

B. Results

We evaluate the presented approach with two different scenarios (Fig. 10): following the reference trajectory, and executing an overtaking manoeuver of a static obstacle. The vehicle speed is set to $6\,m/s$, we position the vehicle on the map, and we generate the local map and apply our algorithm to choose the tentacle to follow at each sampling step.

In the simulation, an occupied cell will have the value 1 and a free cell will have the value 0. We defined a threshold $f_s = 1$ used to decide if a state s_i is occupied or free.

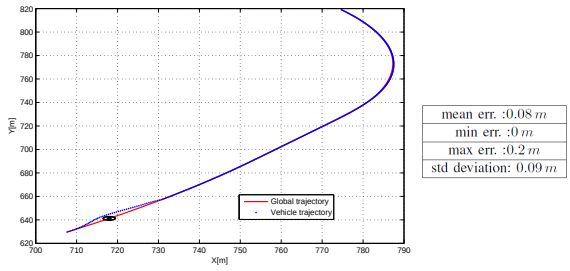
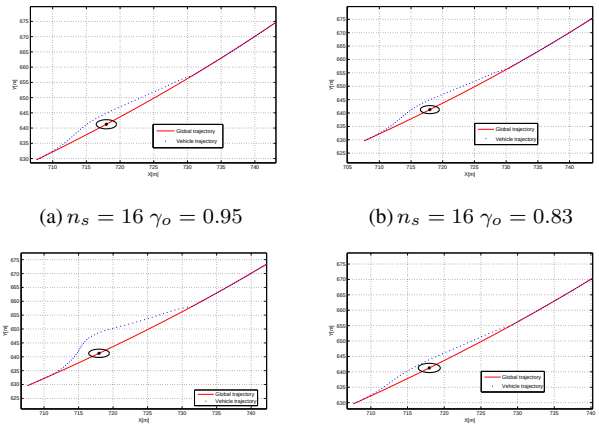


Figure 10: Overtaking manoeuver and following the reference trajectory and different performance measures

In Figure 10, the reactions of the resulting MDP like model policy for two both driving situations are shown. We observe that the vehicle can make an overtaking manoeuver and follow the reference trajectory. The table next to Figure 10 shows the performances of the algorithm like the mean error between the global reference trajectory and the vehicle trajectory (the calculation of this error begins after the overtaking manoeuver).

In Figure 11, we show the influence of the parameters γ_o and n_s. γ_o represents the attenuation with regard to the obstacle proximity. We noticed from Fig.(11a) and Fig. (11b) that, with a bigger γ_o, the lateral displacement is more important while executing an overtaking manoeuver, due to the non-attenuation of the obstacle. n_s represents the number of states per tentacle, Figures (11c) and (11d) show that an increase in the number of states, that corresponds to farther vision horizon, allows to return faster to the reference trajectory after an overtaking manoeuver.



(a) $n_s = 16$ $\gamma_o = 0.95$ (b) $n_s = 16$ $\gamma_o = 0.83$

(c) $n_s = 10$ $\gamma_o = 0.95$ (d) $n_s = 25$ $\gamma_o = 0.95$

Figure 11: Influence of parameters γ_o and n_s with $V = 6\,m/s$

The main advantage of the proposed algorithm is the significant number of freedom degrees which can allow us to change the behavior of the approach. For example we can attenuate the presence of distant obstacles or the importance of following the reference trajectory in the presence of an obstacle. With a MDP model we can introduce the uncertainty on actions by using stochastic transitions instead of deterministic ones. Thus, we can add a possible transition from a clothoid to another. At the last, the method has the capacity to develop more efficient reward calculation taking into account the uncertainty of the environment surrounding the vehicle by using the occupancy grid based on evidential theory.

V. Conclusion and Perspectives

In this work, we introduced a promising method for vehicles autonomous trajectory planning and decision. The method is based on a MDP like model for trajectory planning with clothoid tentacles. The idea is to generate realistic trajectories with tentacles method and select the best thanks to the MDP like process. The simulation results show good performance of our algorithm in avoiding obstacles and following the reference trajectory. In fact, this reactive algorithm does not accumulate data and it takes into account vehicle dynamics and real road structure by using clothoid shape. Among the perspectives, we look to consider the uncertainty of the environment surrounding the vehicle and to include moving obstacles such as moving cars or pedestrians. This improvement would be at the perception level, by using an occupancy grid based on evidential theory, that can be considered in the reward calculation. Further, an interesting extension of our method would be to consider the uncertainty on actions by introducing stochastic transitions instead of deterministic ones.

We also look to implement our algorithm in a robotized vehicle in order to highlight the validity of the algorithm.

References

[1] Jan Becker, Hendrik Dahlkamp, Dmitri Dolgov, Gabe Hoffmann, Doug Johnston, Stefan Klumpp, Dirk Langer, David Orenstein, Johannes Paefgen, Isaac Penny, and Anna Petrovskaya. Junior : The Stanford Entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[2] S. Singh M. Buehler, K. Iagnemma. *The 2005 DARPA grand challenge: the great Robot race*. Springer Trats in Advanced Robotics, 2007.

[3] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015.

[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research (29), 485-501*, 2010a.

[5] M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. *International Symposium on Artificial Intelligence, Robotics, and Automation in Space, 1-7.*, 2005.

[6] P. Broggi, A.and Medici, P. Zani, A. Coati, and M. Panciroli. Autonomous vehicles control in the vislab intercontinental autonomous challenge. *Annual Reviews in Control (36), 161-171.*, 2012.

[7] L. Ma, J. Yang, and M. Zhang. A two-level path planning method for on-road autonomous driving. *In: 2012 2nd International Conference on Intelligent System Design and Engineering Application, 661-664*, 2012.

[8] Felix von Hundelshausen, Michael Himmelsbach, Falk Hecker, Andre Mueller, and Hans-Joachim Wuensche. Driving with tentacles: Integral structures for sensing and motion. *Journal of Field Robotics*, 25(9):640–673, sep 2008.

[9] Michael Himmelsbach, Thorsten Luettel, Falk Hecker, Felix von Hundelshausen, and Hans-Joachim Wünsche. Autonomous Off-Road Navigation for MuCAR-3. *Künstliche Intelligenz (KI)*, 25(2):145–149, feb 2011.

[10] Amalia Foka. Real-Time Hierarchical POMDPs for Autonomous Robot Navigation 2 Partially Observable Markov Decision Processes (POMDPs). (July), 2005.

[11] S. Brechtel, T. Gindele, and R. Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, 28, 2014.

[12] T. Gindele, S. Brechtel, and R. Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *Intelligent Transportation Systems Magazine, IEEE*, (January):69–79, 2015.

[13] J. Moras, V. Cherfaoui, and P. Bonnifait. moving objects detection by conflict analysis in evidential grids. *IEEE Intelligent Vehicles Symposium, vol(6)*, 2011.

[14] A. Chebly, G. Tagne, R. Talj, and A. Charara. A local trajectory planning and tracking for autonomous vehicle navigation using clothoid tentacles method. *International IEEE Conference on Intelligent Vehicles Symposium (IV), Seoul, South Korea, June*, 2015.

[15] H. Yu, J. Gong, K. Iagnemma, Y. Jiang, and J. Duan. Robotic wheeled vehicle ripple tentacles motion planning method. *Intelligent Vehicles Symposium (IV), pp 1156-1161, June*, 2012.

[16] R. Bellman. Dynamic programming. *Princeton University Press*, 1957.

[17] R.A. Howard. Dynamic programming and markov processes. *MIT Press, Cambridge, Massachussets*, 1960.