Near-to-near decentralized algorithm for vehicle platooning

Jano Yazbeck, Alexis Scheuer and François Charpillet
Université de Lorraine, LORIA
Inria, MAIA team - FRANCE
[Jano.Yazbeck | Alexis.Scheuer | Francois.Charpillet]@loria.fr

***Abstract*— In the last decades, intelligent transportation systems noticed an important evolution due to the introduction of intelligent electric vehicles. This paper deals with the platooning problem, a technique which aims at steering a convoy along a path without collision nor lateral deviation. This work focuses on the lateral control in a local decentralized approach. We present here two control algorithms for platooning which aim at reducing the lateral deviation between the leader's and each of the followers' paths. Experimental results show an improvement in the lateral behavior of the followers.**

I. Introduction

Vehicle platooning presents nowadays an attractive solution for traffic congestion and unoptimized space occupation. Intelligent vehicles moving closely would increase the road network capacity, improving the traffic flow and reducing fuel consumption.

Platooning can be approached by several methods:

- centralized approaches: a central computer (usually located at the leading vehicle) collects data from all vehicles using communication. Then, it computes and sends the commands for each one (e. g. [1], [2]). In Sartre project [3], the leader's actions on accelerating, decelerating and steering are transmitted to the followers by the communication system to avoid string instabilities. However, in case the leader crashes or faces technical problems, the consequences on the platoon should be considered.
- global decentralized approaches: each vehicle collects data relative to the state of the platoon and computes its own commands (as [4], [5], [6]).
- near-to-near decentralized approaches: each vehicle perceives its neighboring environment and computes the corresponding behavior ([7], [8], [9]).

Centralized and global decentralized approaches achieve accurate tracking since the computed commands depend on the state of the whole convoy. A reliable communication is then required for data exchange. A brief survey on platooning [10] presented several works based on inter-vehicle communication. The authors pointed out that delays due to the use of communication can be a considerable problem. Effects on platooning should be studied. These delays can be caused by packet loss, data transmission time and their analysis time.

LORIA: Lab. Lorrain de Recherche en Informatique et ses Applications.
INRIA: Institut National de Recherche en Informatique et Automatique.

To achieve a communication-free platooning where robots do not rely on exchanged data or received commands, we choose to develop a decentralized local approach. In these kind of approaches, the vehicles are fully autonomous as they can join or leave a convoy without warning a supervisor. Although, their main drawback due to the local perception is the lack of precision since an accumulated lateral error propagates along the platoon generating important oscillations.

A previous paper [11] presented a lateral controller where this lateral deviation was drastically reduced by aiming previous positions of the predecessor instead of its current position. However, oscillations around the leader's path appear in case the target position was too close to the follower. This method is called the $Memo-LAT$ algorithm (for path Memorization and Look-Ahead Target selection).

The $Memo-LAT$ algorithm was improved into a new method called the NOC algorithm presented in another paper [12] where the curvature of the predecessor's path was considered in the lateral control computation. By applying the NOC method, the robots follow their leader's path without oscillations.

This paper presents a summary on the two previous algorithms. It describes how they work and shows experimental results.

In order to present more precisely our contribution, we first formalize the considered problem in Section II. Then, we explain the $Memo-LAT$ and NOC algorithms in Sections III and IV. In Section V, we discuss experimental results. Finally, we conclude and present our future works in Section VI.

II. Assumptions

Urban vehicles in the city centers must move at low speed for safety reasons. Thus, dynamic effects can be neglected and only a kinematic model is considered to represent their movement. Given that the proposed algorithm is a high level controller, it can be applicable on any kind of robots. Although, experiments were run on an unicycle kinematic model which verifies the equations of movement 1. In Sections III and IV, we present the algorithms for a two-robots convoy. Then, we show in Section V experimental results on four-robots convoy. In this paper, we consider a convoy moving at a constant velocity v_0 where the angular

velocity is bounded.

$$\begin{cases} \dot{x} = v_0 \cos\theta \\ \dot{y} = v_0 \sin\theta \\ \dot{\theta} = \omega \end{cases} \quad (1)$$

III. $Memo - LAT$ ALGORITHM

A vehicle under the $Memo - LAT$ algorithm acquires and stores its predecessor's path as a set of points. Instead of aiming the current predecessor's position, the controller chooses, among the stored positions, a closest one to aim for (see Fig. 1). The lateral error to the exact predecessor's path is thus reduced. The chosen target is located at a look-ahead

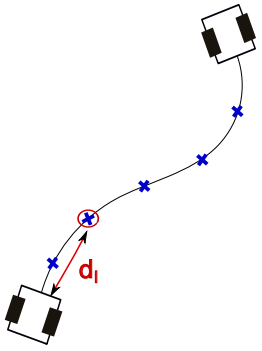

Fig. 1. The chosen target according to the $Memo - LAT$ algorithm.

distance d_l smaller than the inter-distance between the two successive robots. Then, the controller computes an angular velocity in order to reduce the lateral deviation between the follower and its target. If this deviation is important, the follower turns with an important angular velocity generating oscillations around its predecessor's path.

The value of the look-ahead distance d_l influences on the platooning accuracy: in high curvature paths, an important value of d_l leads to an important lateral deviation between the follower and its leader's path. On the contrary, a small value of d_l reduces the lateral deviation but creates oscillations around a path with a relatively small curvature. We run several simulations while varying the path curvature and the initial robots configurations (position, orientation and velocity). But we were not able to establish an analytic relation allowing us to set dynamically the value of d_l according to the state of the system. So, we propose in Section IV the NOC algorithm.

IV. NOC ALGORITHM

NOC algorithm improves the $Memo - LAT$ method by reducing oscillations and lateral deviations. As in $Memo - LAT$ method, each follower stores its predecessor's path as a set of points. In order to achieve a precise tracking, the follower aims at reducing the surface between its path and its predecessor's path by computing a sequence of angular velocities $\{\omega_1^*, \omega_2^*, ..., \omega_i^*\}$ as shown in Fig. 2. As we noticed in $Memo - LAT$ algorithm, the steering velocity needs to be computed according to the shape of the predecessor's path. So, we propose in the NOC algorithm to consider the

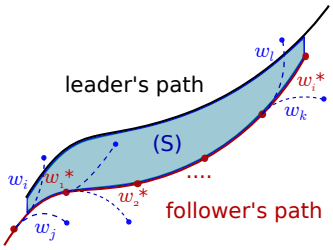

Fig. 2. The lateral deviation represented by the surface between the paths of two successive robots.

curvature of the target path in order to optimize the follower's movement.

A. Algorithm overview

NOC algorithm is resumed to the following steps:

- acquire and store the new position of the leader ;
- choose among the stored positions the best target to aim for ;
- approximate the path around the target by a line or a circle and deduce the corresponding curvature ;
- consider the approximated curvature to compute the angular velocity which reduces the lateral and angular errors while avoiding oscillations.

Instead of choosing a position located at d_l as in $Memo - LAT$ algorithm, the controller under NOC method looks for the closest position of the predecessor's path which allows the follower to converge without crossing its approximated path around this position.

B. Optimization criteria

In order to optimize its lateral movement, the follower needs to reduce an error function. Once the follower approximates its predecessor's path by a circle or a line, it computes the lateral and angular errors to the approximated path. Then, it tries to compute an angular velocity which reduces the error function based on these lateral and angular errors. This error function is defined as follows:

$$\mathcal{E} = k_y \epsilon_y + k_\theta \epsilon_\theta \quad (2)$$

where ϵ_y and ϵ_θ are respectively the lateral and angular errors, and k_y and k_θ are the corresponding weights.

C. On-line path generation

Once the target position is chosen, the follower approximates the path around the target by a line or a circle and deduces the curvature. So instead of tracking the target position, the follower will compute angular velocity to follow the approximated path. We consider two cases:

- if the follower aims for the initial position of its predecessor, the target path is the line joining the initial positions of the two robots (leader-follower).
- if the target position is not the first position of the predecessor, the follower picks the two neighboring positions from the predecessor's path (the one before and the one after the target point). If these three points are aligned,

the target path is a line and the approximated curvature is equal to zero. Otherwise, the circle generated by these three points is the target path and the curvature around the target position is the inverse of the circle radius.

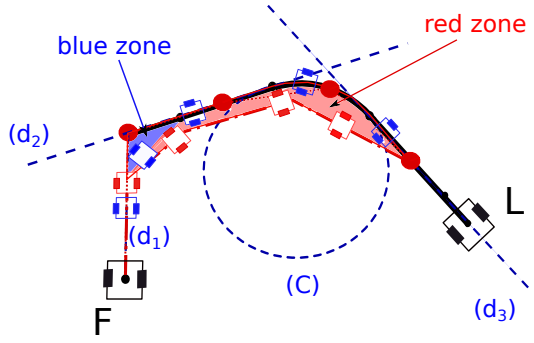

Fig. 3. Approximation of the predecessor's path (in black): by a succession of lines and circles (in blue) under NOC algorithm, by a succession of lines under $Memo-LAT$ algorithm (in red).

In Fig. 3, we compare the behavior of a following robot under $Memo-LAT$ and NOC algorithms. As we can see in $Memo-LAT$ algorithm, once the inter-distance between the follower and its target is smaller than the look-ahead distance d_l, the follower aims for the next target. The higher the look-ahead distance is, the more important the lateral deviation (represented by the red surface) between the leader and follower paths is. On the contrary, NOC algorithm approximates the leader's path by a succession of lines and circles and computes a sequence of angular velocities to follow this approximated path. By anticipating its orientation, the follower aims at reducing the red surface into the blue one.

D. Angular velocity computation

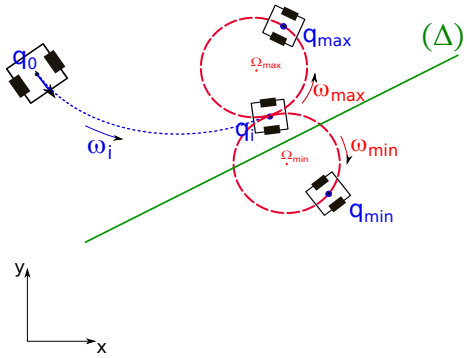

Fig. 4. Condition Γ_i^+: prediction of a possible intersection between the follower turning with ω_{max} and the leader's linear path in the future time.

After choosing its new target, the robot approximates the path around it by a line $[\mathcal{D}]$ or a circle $(\mathcal{C})$. Then, the robot retains, from the discretized domain $\{\omega_{min}, ..., \omega_{max}\}$, the angular velocities which, by generating the corresponding arc of a circle $(\mathcal{C}_i)$ or segment line $[\mathcal{D}_i]$, verify the following conditions:

- Condition Γ_i: during Δt, no intersection between the arc of a circle $(\mathcal{C}_i)$ (or the segment line $[\mathcal{D}_i]$) and $(\mathcal{C})$ (or (Δ)).
- Condition Γ_i^+: if ω_i is applied, the robot will have the new configuration q_i. Then, starting at q_i, the robot generates a circle by turning with the maximum angular velocity (ω_{max} or ω_{min}). this entire generated circle must not cross the approximated path $(\mathcal{C})$ (or (Δ)).

From these retained angular velocities, the best command is the one which minimizes the predefined error function $\mathcal{E}$.

E. Refinement of the angular velocity

The chosen angular velocity may not be the best command due to the discretization. We note ω_c the angular velocity which verifies the conditions Γ_i and Γ_i^+ while minimizing the error function $\mathcal{E}$. We also note $\omega_\cap$ the angular velocity which minimizes $\mathcal{E}$ while satisfying only Γ_i. The robot crosses its predecessor's path while turning with ω_{max} at the next time step. By discretizing the new domain $[\omega_\cap,\ \omega_c]$, a better command can be found: ω^* satisfies both Γ_i and Γ_i^+ and reduces the error function more than ω_c.

V. Experimental results

We consider a four-robots convoy moving at a constant velocity $v_0 = 8\ m/s$. Their maximum angular velocity is $\omega_{max} = 1\ rad/s$, so the minimum turning radius is $8m$. The weights to reduce the lateral and angular errors are respectively $k_y = 0.9$ and $k_\theta = 0.1$. The angular velocity domain $[-\omega_{max}, \omega_{max}]$ is discretized into 50 values. Also, the new angular velocity domain for the refinement is discretized into 100 values.

A. A four-robots convoy platooning along a rounded square

We consider a platooning along a rounded square. As Figure 5 shows, the robots are initially at L, F_1, F_2, F_3 and they move according to the direction of the arrow.

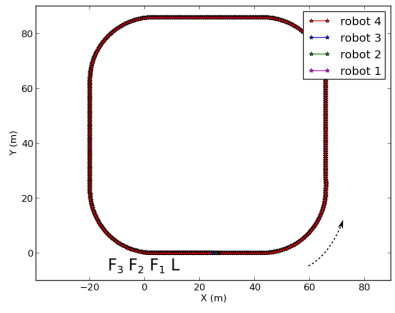

Fig. 5. Platooning of a four-robots convoy along a rounded square.

B. Comparison with the $Memo-LAT$ algorithm

We presented previously the $Memo-LAT$ algorithm [11]. In $Memo-LAT$ algorithm, the tracked positions are located at a look-ahead distance $d_l = 0.5\ m$ from the current position of the follower. While the $Memo-LAT$ algorithm generates oscillations around the leader's

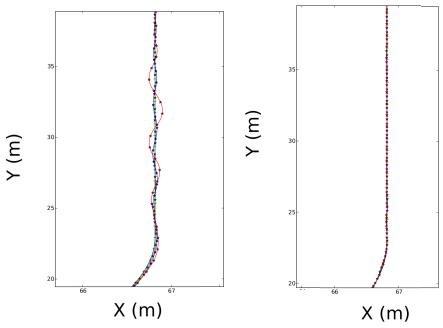

Fig. 6. Comparison between $Memo - LAT$ (on the left) and NOC (on the right) algorithms: zooming around a part of the leader's path during a platooning of a four-robots convoy along a rounded square under the $Memo - LAT$ algorithm. The look-ahead distance is $d_l = 0.5m$.

path (Fig.6), NOC algorithm reduces them and allows the follower to have a smoother movement.

C. Re-convergence of a suddenly disturbed follower towards its target path

We consider a two-robots convoy moving along a spiral path. At an instant t_i, the follower is suddenly disturbed by injecting a spatial perturbation ($\Delta x = 1m,\ \Delta y = 1m$). It is transposed from its position before disturbance F_b to its new position after disturbance F_a (Fig. 7).

As we can notice, this lateral deviation is quickly reduced as the follower converges towards its leader's path and follows it precisely. We inject the same perturbation on the

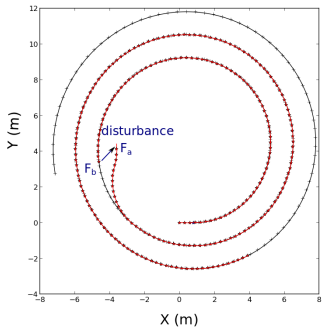

Fig. 7. NOC algorithm: Re-convergence of a suddenly disturbed follower during a platooning along a spiral path.

follower and we observe its behavior in Fig. 8 under the $Memo - LAT$ algorithm. The follower tries to re-converge quickly but important oscillations are induced around the spiral path of the leader.

VI. Conclusion and Future Works

In this paper, we presented a summary on two control algorithms for a platooning where the leader's analytic path is unknown to the followers. Instead of aiming at the current position of its predecessor, the robot under $Memo - LAT$ algorithm aims at a previous stored position of the predecessor. This method drastically reduces the lateral deviation but creates occasionally important oscillations around the

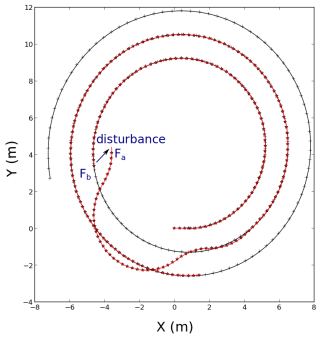

Fig. 8. $memo - LAT$ algorithm: Re-convergence of a suddenly disturbed follower during a platooning along a spiral path.

leader's path. NOC algorithm considers the curvature of the predecessor's path in the computation of the angular velocity: the oscillations are thus reduced. In the future works, we will try to improve the NOC algorithm by finding a more intelligent method to discretize the angular velocity domain. Also, we will combine the longitudinal and lateral controllers and study the effects of the longitudinal velocity on the lateral behavior of the convoy during platooning.

Acknowledgement

This work was partially supported by INTRADE european project.

References

[1] G. Antonelli and S. Chiaverini, "Kinematic control of platoons of autonomous vehicles," *IEEE Trans. Robotics*, vol. 22, no. 6, pp. 1285–1292, Dec. 2006.

[2] C. Smaili, M. El Badaoui El Najjar, and F. Charpillet, "Multi-sensor fusion method using bayesian network for precise multi-vehicle localization," in *11th International IEEE Conference on Intelligent Transportation Systems*, Beijing (CN), Oct. 2008, pp. 906—-911.

[3] E. Coelingh and S. Solyom, "All aboard the robotic road train," *IEEE Spectrum*, pp. 26–31, 2012.

[4] J. Bom, B. Thuilot, F. Marmoiton, and P. Martinet, "Nonlinear control for urban vehicles platooning, relying upon a unique kinematic gps," in *International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp. 41–46.

[5] P. Avanzini, B. Thuilot, T. Dallej, P. Martinet, and J. Derutin, "On-line reference trajectory generation for manually convoying a platoon of automatic urban vehicles," in *IROS*, 2009, pp. 1867–1872.

[6] P. Avanzini, B. Thuilot, and P. Martinet, "Obstacle avoidance and trajectory replanification for a group of communicating vehicles," in *ITST*, 2009, pp. 267 – 272.

[7] J.-M. Contet, F. Gechter, P. Gruer, and A. Koukam, "Multiagent system model for vehicle platooning with merge and split capabilities," in *3rd International Conference on Autonomous Robots and Agents (ICARA)*, Palmerston North (NZ), Dec. 2006.

[8] S.-Y. Yi and K.-T. Chong, "Impedance control for a vehicle platoon system," *Mechatronics*, vol. 15, no. 5, pp. 627–638, June 2005.

[9] P. Daviet and M. Parent, "Longitudinal and lateral servoing of vehicles in a platoon," in *Proc. of the IEEE Int. Symp. on Intelligent Vehicles*, Tokyo (JP), Sept. 1996, pp. 41–46.

[10] P. Kavathekar and Y. Chen, "Draft: Vehicle platooning: a brief survey and categorization," in *Proceedings of The ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2011.

[11] J. Yazbeck, A. Scheuer, O. Simonin, and F. Charpillet, "Improving near-to-near lateral control of platoons without communication," in *IROS*, 2011, pp. 4103–4108.

[12] J. Yazbeck, A. Scheuer, and F. Charpillet, "Optimized lateral control for a decentralized near-to-near platooning," INRIA, Tech. Rep., 2013.