# Information-Theoretic Sensor-Based Predictive Control for Autonomous Vehicle Navigation: A Proof of Concept

David Perez-Morales, Vincent Frémont

# Information-Theoretic Sensor-Based Predictive Control for Autonomous Vehicle Navigation: A Proof of Concept

David Pérez-Morales[1] and Vincent Fremont[2]

***Abstract*— This paper explores the feasibility of an Information-Theoretic Sensor-Based Predictive Control (IT-SBPC) approach for autonomous navigation in presence of pedestrians. Our technique relies solely in sensor data expressed relative to the vehicle and therefore no localization is inherently required. By combining the advantages of the information-theoretic framework and sensor-based formalism, the proposed technique drives the vehicle safely and smoothly towards the desired goal. Several real-time simulated scenarios, showing that the car is able to reach the goal with centimeter-level accuracy, validate the effectiveness of our approach.**

## I. INTRODUCTION

Vehicles and Intelligent Transportation Systems have followed a quick mutation thanks to the growing use of electric technology and the emergence of autonomous navigation techniques. The hype created around self-driving vehicles even lead parts of the industry to overestimate their arrival [1] as commercial products. In spite of the efforts from the industry and academia to reach fully autonomously driving capabilities (level 5), there's not yet a system with such capacities [2]. According to different studies, pedestrian populating environments present the major challenges due to the ethical, social and legal considerations that arise [3], [4]. Moreover, crowded environments present challenges even for experienced drivers. Therefore, being able to navigate safely and effectively under such conditions would represent a major milestone towards fully autonomous driving.

In order to deal with pedestrian populating environments, Model Predictive Path Integral (MPPI) Control techniques appear as an interesting path to explore due to the prediction step inherently considered. MPPI control algorithms and its derivatives have been extensively investigated for autonomous navigation tasks [5], [6], [7]. Some of the key features of this type of techniques are that it is a derivative-free method optimal control method that can naturally deal with non-linear, non-convex and non-smooth system dynamics and cost functions. Moreover, it is straightforward to impose constraints on both the state and control inputs of the system.

Another interesting alternative for autonomous navigation tasks is the use of sensor-based control techniques. In particular, they have been proven to be valid for navigation [8], dynamic obstacle avoidance [9] and for parking applications in static [10] and dynamic environments [11]. A key benefit of these techniques is their great robustness against modeling and calibration errors, particularly when then pose of the robot doesn't need to be reconstructed. Moreover, because they rely only in locally perceived information, they do not suffer from potential localization issues.

In this work we aim to provide a proof of concept that combines the advantages of the Information-Theoretic Model Predictive Control (ITMPC) framework [6] and the sensor-based formalism [12] by proposing an Information-Theoretic Sensor-Based Predictive Control approach capable of navigating towards a given goal while avoiding collision with (potentially dynamic) surrounding obstacles. Given that this work is a proof of concept, a single obstacle is considered in the environment. Moreover, dynamic obstacles are assumed to be pedestrians.

In the next section the kinematic model of the vehicle, the multi-sensor formalism and the interaction model that allows to describe the navigation task and collision avoidance constraints are presented. Afterwards, the control strategy is presented in Section III. The simulation setup is described in Section IV and the obtained results are presented in Section V. Finally, some conclusions are given in Section VI.

## II. MODELING AND NOTATION

### A. Car-like robot model and notation

Considering the well-known kinematic model of a car with rear-wheel driving [13], the vehicle's twist is defined by the following column vector (elements separated by a semicolon):

$$\mathbf{v}_m = [v_{x_m}; \dot{\theta}_m], \tag{1}$$

where $v_{x_m}$ and $\dot{\theta}_m$ are, respectively the longitudinal (along $x_m$) and rotational velocities expressed in the moving base frame $\mathcal{F}_m$. Additionally, one can link the steering angle $\delta$ to $\dot{\theta}_m$ using the following equation:

$$\dot{\theta}_m = \frac{v_{x_m} \tan \delta}{l_{wb}}. \tag{2}$$

where $l_{wb}$ is the distance between the front and rear wheel axles. Therefore, it is possible to consider as the actual control input of the robotized vehicle the following expression:

$$\mathbf{v}_r = [v_{x_m}; \delta] \tag{3}$$

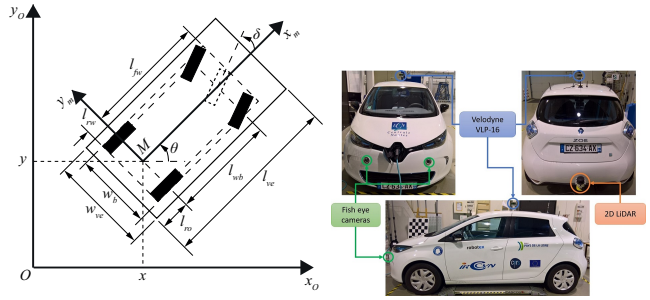
Nonetheless, in order to minimize the chattering produced by the stochastic nature of the proposed approach, the time derivative of (3) is considered as the input of the transition model introduced in the next section:

$$\mathbf{u} = [\dot{v}_{x_m}; \dot{\delta}] \tag{4}$$

[1] David Pérez-Morales and [2] Vincent Fremont are with LS2N, Laboratoire des Sciences du Numérique de Nantes, École Centrale de Nantes, 1 rue de la Noë, 44321 Nantes, France

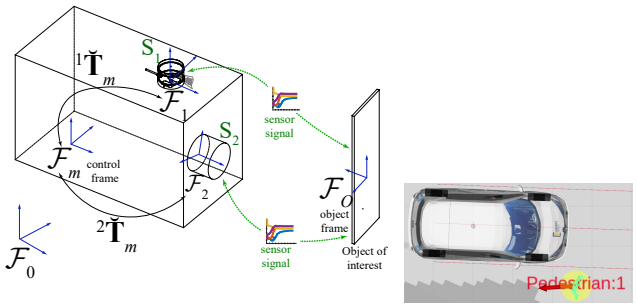
[1] David.PerezMorales@ls2n.fr

[2] Vincent.Fremont@ls2n.fr




(a) (b)

Fig. 1. (a) Kinematic model diagram for a car-like rear-wheel driving robot. (b) Robotized Renault ZOE used as a basis for the simulation model




(a) (b)

Fig. 2. (a) Multi-sensor model. (b) Perceived pedestrian and its estimated orientation.

As such, it is possible to retrieve (3) from (4) with a simple numerical integration:

$$\mathbf{v}_r = \mathbf{v}_r + t_s \mathbf{u} \tag{5}$$

where $t_s$ is the sampling period.

It should be noted that, thanks to the multi-sensor-based formalism considered (introduced in the next subsection), our closed-loop control law does not need to have any knowledge about the Cartesian pose of the vehicle.

The vehicle used for simulation, represented by its bounding rectangle in Fig. 1a, is a Renault ZOE (Fig. 1b). Its relevant dimensional parameters are presented in Table I.

TABLE I

DIMENSIONAL VEHICLE PARAMETERS

| *Parameters* | *Notation* | *Value* |
|---|---|---|
| Wheelbase: Distance between the front and rear wheel axles | $l_{wb}$ | 2.588 m |
| Rear overhang: Distance between the rear wheel axle and the rear bumper | $l_{ro}$ | 0.657 m |
| Total length of the vehicle | $l_{ve}$ | 4.084 m |
| Total width of the vehicle | $w_{ve}$ | 1.945 m |

### B. Multi-sensor modeling

For the sake of clarity, the considered multi-sensor modeling (detailed in [14]) is recalled in this subsection.

Let us consider a robotic system equipped with $k$ sensors (Fig. 2a) that provide data about the robot pose in its environment. Each sensor $S_i$ gives a signal (sensor feature) $\mathbf{s}_i$ of dimension $\mathbb{d}_i$ with $\sum_{i=1}^{k} \mathbb{d}_i = \mathbb{d}$.

In a static environment, the sensor feature derivative can be expressed as follows:

$$\dot{\mathbf{s}}_i = \check{\mathbf{L}}_i \check{\mathbf{v}}_i = \check{\mathbf{L}}_i {}^i\check{\mathbf{T}}_m \check{\mathbf{v}}_m \tag{6}$$

where $\check{\mathbf{L}}_i$ is the interaction matrix [12] of $\mathbf{s}_i$ ($\dim(\check{\mathbf{L}}_i) = \mathbb{d}_i \times 6$) and ${}^i\check{\mathbf{T}}_m$ is the 3D screw transformation matrix that expresses the sensor twist $\check{\mathbf{v}}_i$ (which is expressed in its corresponding frame $\mathcal{F}_i$) with respect to the robot twist $\check{\mathbf{v}}_m$, expressed in the mobile frame $\mathcal{F}_m$ which is used as control frame.

Denoting $\mathbf{s} = (\mathbf{s}_1; \ldots; \mathbf{s}_k)$ the $\mathbb{d}$-dimensional signal of the multi-sensor system, the signal variation over time can be linked to the moving vehicle twist:

$$\dot{\mathbf{s}} = \check{\mathbf{L}}_\mathbf{s} \check{\mathbf{v}}_m \tag{7}$$

with $\check{\mathbf{L}}_\mathbf{s} = \check{\mathbf{L}} \check{\mathbf{T}}_m$, where $\check{\mathbf{L}}$ and $\check{\mathbf{T}}_m$ are obtained by concatenating either diagonally or vertically, respectively, matrices $\check{\mathbf{L}}_i$ and ${}^i\check{\mathbf{T}}_m \; \forall \; i \in [1 \; \ldots \; k]$.

*Planar world assumption:* Assuming that the vehicle to which the sensors are rigidly attached evolves in a plane and that the sensors and vehicle have vertical parallel $z$ axes, all the twists are reduced to $[v_{x_i}; v_{y_i}; \dot{\theta}_i]$ hence the reduced forms (denoted by the accent $\check{}$) of the various matrices and vectors (denoted by the accent $\check{}$) are considered. $\check{\mathbf{L}}_i$ is of dimension $\mathbb{d}_i \times 3$, $\check{\mathbf{v}}_m = [v_{x_m}; v_{y_m}; \dot{\theta}_m]$ and ${}^i\check{\mathbf{T}}_m$ is defined as:

$${}^i\check{\mathbf{T}}_m = \begin{bmatrix} \cos({}^m\theta_i) & \sin({}^m\theta_i) & x_i \sin({}^m\theta_i) - y_i \cos({}^m\theta_i) \\ -\sin({}^m\theta_i) & \cos({}^m\theta_i) & x_i \cos({}^m\theta_i) + y_i \sin({}^m\theta_i) \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where ${}^m\mathbf{t}_i = [x_i; y_i]$ and ${}^m\theta_i$ are, respectively, the position and orientation of $S_i$ (frame $\mathcal{F}_i$) with respect to $\mathcal{F}_m$ expressed in $\mathcal{F}_m$. Furthermore, since in the considered model the control frame $\mathcal{F}_m$ is attached to the vehicle's rear axle with origin at the point $M$ (Fig. 1a), the robot twist $\check{\mathbf{v}}_m$ can be further reduced to (1). Thus, it is possible to write:

$$\dot{\mathbf{s}} = \mathbf{L}_\mathbf{s} \, \mathbf{v}_m \tag{9}$$

where $\mathbf{L}_\mathbf{s}$ is composed of the first and third columns of $\check{\mathbf{L}}_\mathbf{s}$.

Moreover, if the appropriate considerations are taken in the interaction model, sensor-based control strategies are capable of dealing with moving features [15]. (7) can be modified to account for the velocity of moving sensor features as follows:

$$\dot{\mathbf{s}} = \check{\mathbf{L}}_\mathbf{s} (\check{\mathbf{v}}_m - \check{\mathbf{v}}_{\text{mo}}) \tag{10}$$

where $\check{\mathbf{v}}_{\text{mo}}$ is the twist of the moving feature expressed in the vehicle frame. Furthermore, if at some point a moving feature becomes static (i.e. $\check{\mathbf{v}}_{\text{mo}}$ is null), (10) would simply become $\dot{\mathbf{s}} = \check{\mathbf{L}}_\mathbf{s} \check{\mathbf{v}}_m$, expression that can be easily reduced to (9) with no further impact on the modeling. For this work we assume that $\check{\mathbf{v}}_{\text{mo}}$ is known.

### C. Interaction model

For the interaction model it is assumed that the system is capable of perceiving its desired goal and (potentially dynamic) surrounding obstacles as points in, respectively, polar $(\rho, \theta)$ and Cartesian $(x, y)$ coordinates. Therefore, the features in polar coordinates and the associated interaction matrix are denoted as [16]:

$$\mathbf{s}_{\rho\theta} = [\rho; \theta], \tag{11}$$

$$\check{\mathbf{L}}_{\rho\theta} = \begin{bmatrix} -\cos(\theta) & -\sin(\theta) & 0 \\ \frac{\sin(\theta)}{\rho} & -\frac{\cos(\theta)}{\rho} & -1 \end{bmatrix}. \tag{12}$$

Similarly, the features in Cartesian coordinates and the associated interaction matrix are denoted as [12]:

$$\mathbf{s}_{xy} = [x; y], \tag{13}$$

$$\check{\mathbf{L}}_{xy} = \begin{bmatrix} -1 & 0 & y \\ 0 & -1 & -x \end{bmatrix}. \tag{14}$$

Without entering in matters of social interaction, the moving obstacle is assumed to be a pedestrian. It is modeled as a point in Cartesian coordinates with a given orientation (Fig. 2b).

## III. INFORMATION-THEORETIC SENSOR-BASED PREDICTIVE CONTROL

The ITMPC [6] strategy, which our work is based upon, is a stochastic Model Predictive Control method that can be applied to non-linear (with respect to the control input or the state) systems and with non-convex cost objectives. The key idea of the approach is to evaluate the cost of thousands of trajectories (rollouts) sampled from the system dynamics using a Monte-Carlo simulation in real time.

Let us consider a discrete time dynamical system of the form

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{w}_t) \tag{15}$$

where $\mathbf{s} \in \mathbb{R}^{\mathsf{d}}$ is the state of the system at time $t$, $\mathbf{w}_t \in \mathbb{R}^{\mathsf{n}}$ is the input to the system at time $t$ and $f$ is the state-transition function of the system. It is assumed that we have no direct control over the input variable $\mathbf{w}_t$ and instead $\mathbf{w}_t$ is a random vector generated by a white-noise proces with density function $\mathbf{w}_t \sim \mathcal{N}(\mathbf{u}_t, \Sigma)$ in which we have a direct control over the mean $\mathbf{u}_t$. This assumption translates to the low-level controller achieving the set-point with some error that satisfies a Gaussian distribution.

Given a finite-time horizon $t \in \{0, 1, 2, \ldots, T-1\}$, the stochastic optimal control aims to find a control input sequence $\tilde{\mathbf{u}} = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}\} \in \mathbb{R}^{n \times T}$ that minimizes the expectations $\mathbb{E}\left[S(\mathbf{s}_t, \tilde{\mathbf{w}})\right]$ with respect to all the simulated rollouts generated using (15). The optimal control problem can be formulated as:

$$\begin{aligned} J &= \min_{\tilde{\mathbf{u}} \in U} \mathbb{E}\left[S(\mathbf{s}_t, \tilde{\mathbf{w}})\right] \\ &= \min_{\tilde{\mathbf{u}} \in U} \mathbb{E}\left[\phi(\mathbf{s}_T) + \sum_{t=0}^{T-1} (q(\mathbf{s}_t) + \gamma \mathbf{u}_t^{\mathsf{T}} \Sigma^{-1} (\mathbf{u}_t - \mathbf{w}_t))\right] \end{aligned} \tag{16}$$

where $U$ is the set of admissible command sequences, $S(\mathbf{s}_t, \tilde{\mathbf{w}})$ is the state-dependent cost to go of a rollout while $\phi(\mathbf{s}_T)$ and $q(\mathbf{s}_t)$ are, respectively, the terminal and the state-dependent running costs, and $\gamma = \lambda(1-\alpha)$ where $\lambda$ is the inverse temperature and $0 <= \alpha <= 1$ is a parameter that balances the requirements of low energy ($\alpha = 0$, $\tilde{\mathbf{u}}$ is pushed to zero) and control smoothness ($\alpha = 1$, $\tilde{\mathbf{u}}$ is kept near the current planned control sequence). As noted in [6], this formulation is closely related to the well-known policy gradient theorem typically used in reinforcement learning.

**Algorithm 1:** Information-Theoretic Control

**Data:** $f$: Transition mmodel;
$K$: Number of rollouts;
$T$: Number of time steps;
$\tilde{\mathbf{u}} = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}\}$: Initial control sequence;
$\phi, q, \Sigma, \gamma, \alpha, \lambda$: Cost functions/parameters;
SGF: Savitzky-Golay convolutional filter;

**while** *task not completed* **do**
  $\mathbf{s}_0 \leftarrow$ GetState();
  $S_k \leftarrow$ InitTrajectoryCost(), $S_k \in \mathbb{R}^K$;
  **for** $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{s} \leftarrow \mathbf{s}_0$;
    Sample $\varepsilon^k = \{\epsilon_0^k, \epsilon_{T-1}^k\} \in \mathcal{N}(0, \Sigma)$;
    **for** $t \leftarrow 0$ **to** $T-1$ **do**
      **if** $k \geq (1-\alpha)K$ **then**
        $\mathbf{w}_t = \mathbf{u}_t + \epsilon_t^k$;
      **else**
        $\mathbf{w}_t = \epsilon_t^k$;
      $\mathbf{s} \leftarrow f(\mathbf{s}, \mathbf{w_t})$;
      $S_k += q(\mathbf{s}) + \gamma \mathbf{u}_t^{\mathsf{T}} \Sigma^{-1} (\mathbf{u}_t - \mathbf{w}_t)$;
    $S_k += \phi(\mathbf{s})$;
  $\beta \leftarrow \min_k[S_k]$;
  $\eta \leftarrow \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda}(S_k - \beta)\right)$;
  **for** $k \leftarrow 0$ **to** $K-1$ **do**
    $w_k \leftarrow \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}(S_k - \beta)\right)$;
  **for** $t \leftarrow 0$ **to** $T-1$ **do**
    $\tilde{\mathbf{u}} \leftarrow \tilde{\mathbf{u}} + \sum_{k=0}^{K-1} w_k \varepsilon^k$
  $\tilde{\mathbf{u}} \leftarrow$ SGF($\tilde{\mathbf{u}}$);
  $\mathbf{v}_r \leftarrow$ NumericalIntegration($\mathbf{v}_r, \mathbf{u}_0$);
  SendToActuators($\mathbf{v}_r$);
  **for** $t \leftarrow 1$ **to** $T-1$ **do**
    $\mathbf{u}_{t-1} = \mathbf{u}_t$;
  $\mathbf{u}_{T-1} =$ Init($\mathbf{u}_{T-1}$);

Our Information-Theoretic Sensor-Based Predictive Control strategy is detailed in Algorithm 1. It starts by getting the current state of the system and then producing $K$ trajectory samples. Each rollout is generated by simulating forward the dynamics using a randomly generated control sequence. A cost based on the state and inputs is computed for each rollout. Having computed the cost of each trajectory (which is initialized to zero), the control sequence is updated based on the minimum sampled cost and the probability-weighted average over all the perturbation sequences. Afterwards, the updated control sequence is passed through a Savitzky-Golay Filter (SGF). A simple numerical integration (5) is used to obtain the control input to be sent to the vehicle. Finally, the control sequence of is slid down in order to warm-start the next control loop iteration.

The key difference with the work presented in [6] is the use of a numerical integration step in the transition model (Algorithm 2) and to obtain the final control input $\mathbf{v}_r$ to be sent to the vehicle. As acknowledged in [6], the stochastic

**Algorithm 2:** Transition Model $f$

**Data:** $\mathbf{s}, \mathbf{w}$: State and input;
$\mathbf{v}_r$: Current control input;
$v_{\max}, \delta_{\max}$: Maximum control input values;
$t_s$: Sampling period;

$\mathbf{v}_r \leftarrow \text{NumericalIntegration}(\mathbf{v}_r, \mathbf{w})$;
$\mathbf{v}_r \leftarrow \text{ClampMaxValues}(\mathbf{v}_r)$;
$\mathbf{v}_m \leftarrow \text{BuildTwist}(\mathbf{v}_r)$;
$\mathbf{s}_{\rho\theta} \leftarrow \mathbf{s}_{\rho\theta} + \mathbf{L}_{\mathbf{s}_{\rho\theta}}\, t_s\, \mathbf{v}_m$;
$\mathbf{s}_{xy} \leftarrow \mathbf{s}_{xy} + \check{\mathbf{L}}_{\mathbf{s_{xy}}}\, t_s\, (\check{\mathbf{v}}_m - \check{\mathbf{v}}_{\text{mo}})$;

TABLE II
CONTROL-RELATED PARAMETERS

| *Parameter* | *Value* |
|---|---|
| Control frequency | 20 Hz |
| $T$ | 80 |
| $K$ | 4500 |
| $\lambda$ | 3.5 |
| $\alpha$ | 0.99 |
| $\Sigma$ | diag(0.00125,0.0035) |
| Init($\mathbf{u}_{T-1}$) | (0,0) |
| InitTrajectoryCost() | 0 |
| $v_{\max}$ | $\leq 2.7778\,\mathrm{m/s}$ |
| $\delta$max | $0.5236\,\mathrm{rad}$ |

nature of the Information-Theoretic Model Predictive Control strategy can lead to significant chattering. We found that even after applying a SGF to the control sequence, the non-smooth behavior of the control was still strongly present, especially when the vehicle starts relatively far away from the goal. By using the aforementioned numerical integration step (5) we managed to considerably reduce this effect.

## IV. Simulation Setup

In order to evaluate the performance of our proposed IT-SBPC approach we make use of an in-house developed fast prototyping environment using the same software architecture as the one inside the real vehicle. In addition to behaving nearly identically (form a software architecture point of view) to the real car, this fast prototyping environment simulates as well the dynamics of the vehicle leading to rather realistic simulations. Moreover, to simulate the pedestrian we make use of a modified version [17] of the open source crowd simulator pedsim_ros [18]. The pedestrian generated by pedsim_ros is aware of the vehicle and thus its behavior/motions are influence by it.

Considering that the task is to reach a goal while avoiding collision, the state-dependent cost function is defined as:

$$q(\mathbf{s}) = \mathbf{e}^{\mathsf{T}}\mathbf{Q}\mathbf{e} + \mathbf{v}_m^{\mathsf{T}}\mathbf{R}\mathbf{v}_m + 10^4 C \tag{17}$$

where $\mathbf{e} = \mathbf{s}_{\rho\theta} - \mathbf{s}^*_{\rho\theta}$, with $\mathbf{s}^*_{\rho\theta}$ being the desired value (i.e. the goal to be reached) of $\mathbf{s}_{\rho\theta}$. $\mathbf{Q}$ and $\mathbf{R}$ are weighted matrices that regulate the influence of, respectively, the different elements of the task sensor features and of the vehicle's twist. These matrices are defined as:

$$\mathbf{Q} = \text{diag}(0.55, 1.0) \quad (18a) \qquad \mathbf{R} = \text{diag}(2.5, 30.0) \quad (18b)$$

Finally, to avoid collision with the obstacle perceived as $\mathbf{s}_{xy}$, $C = w(x)w(y)$, where $w(s)$ is a generic smooth weighting function (Fig. 3) that depends on the current value of a given sensor feature. The activation boundaries $s^{s^{\pm}}, s^{\pm}$ of $w(s)$ are defined such that they describe two bounding rectangles for the vehicle $1.0\,\mathrm{m}$ and $4.0\,\mathrm{m}$ longer and $0.7\,\mathrm{m}$ and $3.7\,\mathrm{m}$ wider than $l_{ve}$ and $w_{ve}$ respectively. These bounding rectangles are denoted by the red and orange ones in Figs. 4a-7a. The size of the smaller bounding rectangle has a direct impact on the safety distance to the obstalce while the size of the larger one affects the distance at which the vehicle starts to react to the obstacle. The sizes chosen were found empirically aiming to balance safety and comfort requirements while avoiding being overly convservative. The minimum $w^-$ and maximum $w^+$ values of $w(s)$ are, respectively 0 and 1. The rest of the control-related parameters are given in Table II.

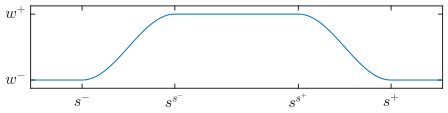



Fig. 3. Generic smooth weighting function $w(s)$

It should be noted that, contrary to the typical real-time implementations of MPPI-like techniques which make use of GPUs [19], [6], [20], our implementation runs on a single core of an AMD Ryzen 9 5950X.

Finally, all the simulations presented in the next section start with the goal placed at $51\,\mathrm{m}$ straight ahead of the car and $\mathbf{s}^*_{\rho\theta} = [1; 0]$. The goal placement was chosen in order to let the car smoothly reach its maximum allowed velocity before having to perform any avoidance maneuver. $\mathbf{s} = [\mathbf{s}_{\rho\theta}; \mathbf{s}_{xy}]$ is assumed to be already expressed in the control frame $\mathcal{F}_m$.

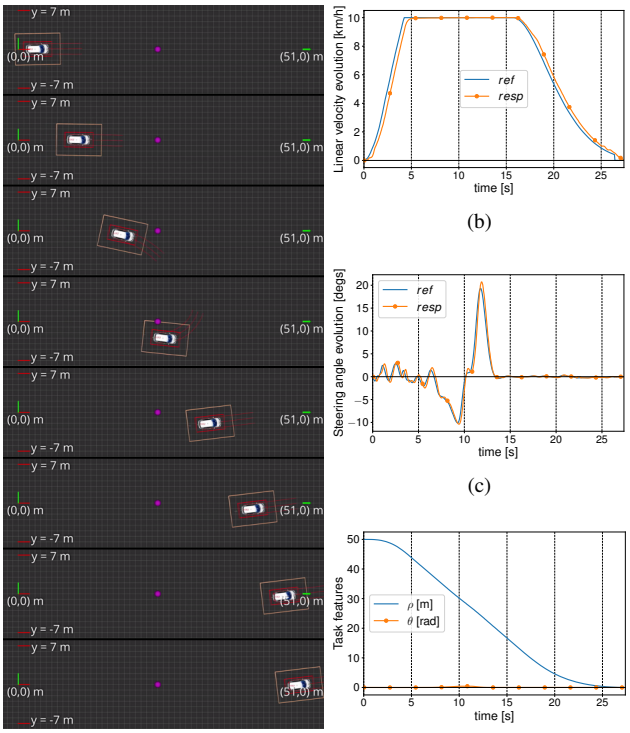
## V. Results

### A. Static obstacle

We start the assessment of our approach with the easiest case: A static obstacle placed between the starting and desired positions, specifically at 25 m straight ahead of the car. As it can be seen in Fig. 4, the vehicle successfully and smoothly (Fig. 4d) reaches the goal while avoiding collision with the static obstacle by slightly turning to the right to then later turn left to keep moving towards the goal. The linear velocity $v_{x_m}$ (Fig. 4b) is generally smooth. The steering angle $\delta$ (Fig. 4c) starts slightly unstable but it gets more stable as the linear velocity approaches the maximum value and it remains stable even when avoiding the obstacle. The final error is $\mathbf{e} = [-0.0162; -0.0057]$.

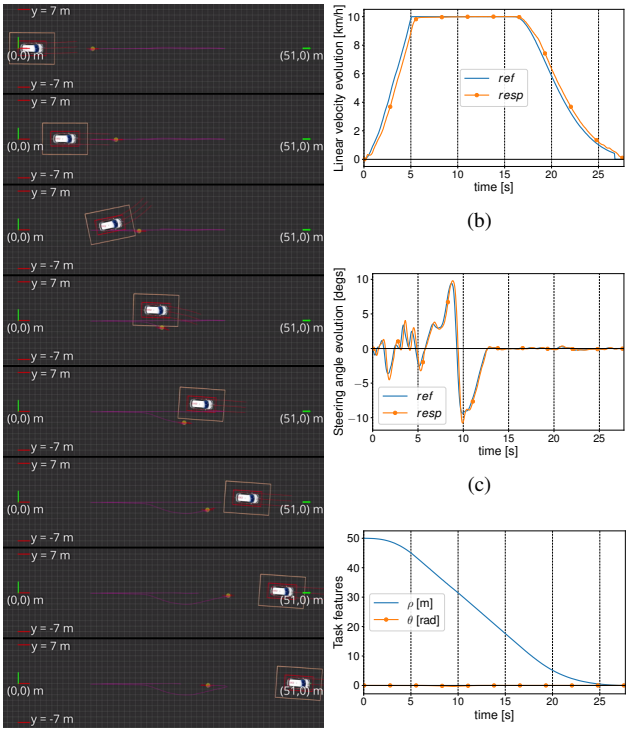
### B. Pedestrian with parallel motion, same orientation

We now move to a case where a simulated pedestrian is present in the environment. Its motion is parallel to what the ideal motion of the vehicle would be if no obstacles were to be present. At the moment when the vehicle has to avoid the pedestrian, both agents have mostly the same orientation. Similarly to the previous case, the vehicle successfully reaches the goal while avoiding collision (Fig. 5), this time by slightly turning to the left to then later turn right




(a) (b) (c) (d)

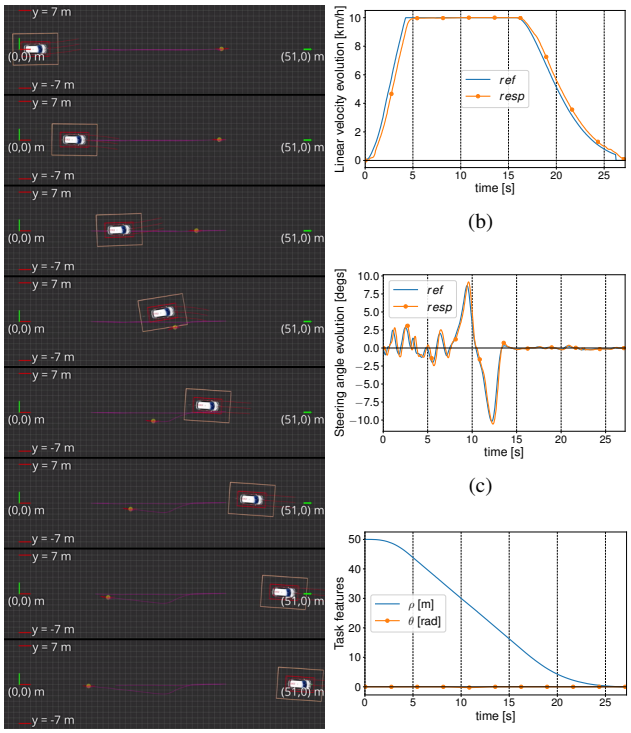
Fig. 4. Static obstacle




(a) (b) (c) (d)

Fig. 5. Pedestrian with parallel motion, same orientation

to keep moving towards the goal. The collision avoidance maneuver seems natural (Fig. 5a), i.e. not steering away from the pedestrian too soon nor too late, thanks to the pedestrian's motion being considered in the predictions. The comments on the control and task features signals from the




(a) (b) (c) (d)

Fig. 6. Pedestrian with parallel motion, opposite orientation

previous case apply for this one as well. The final error is $\mathbf{e} = [-0.0210; -0.0001]$.

### C. Pedestrian with parallel motion, opposite orientation

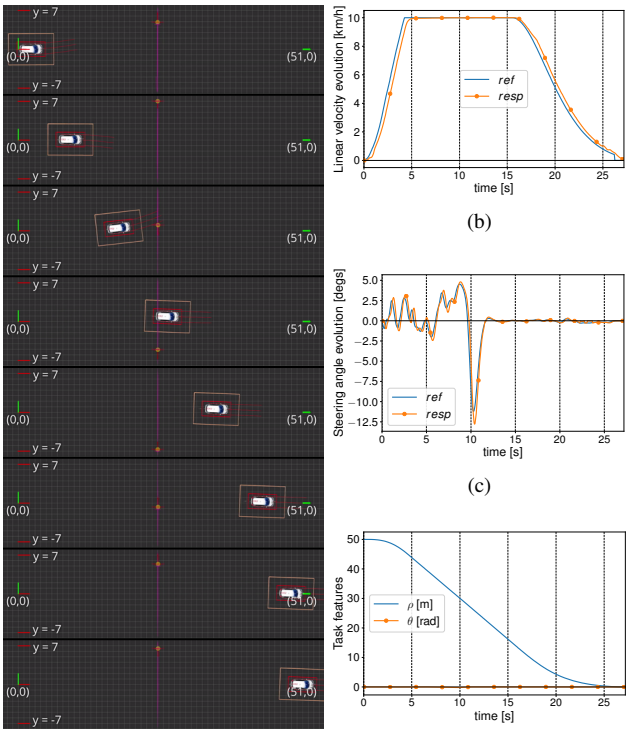
We now present a different case considering the same simulated environment as in the previous one. This time the vehicle and the pedestrian have mostly opposite orientations at the moment of the collision avoidance maneuver. Again, the vehicle successfully reaches the goal while avoiding collision (Fig. 6). The evolution of the control and task features signals is very similar to the previous case. The final error is $\mathbf{e} = [-0.0193; -0.0019]$.

### D. Pedestrian with perpendicular motion

Finally we consider a case where the pedestrian and vehicle motions' are mostly perpendicular. In this case as well the car successfully reaches the goal while avoiding collision (Fig. 7). This time our approach predicts that, given the pedestrian's motion, the best option to avoid collision is to turn slightly to the left, even if that means that at some point the car steers towards the pedestrian (Fig. 7a) because, by the time the car would reach the line traced by the pedestrian, it would have already moved downwards and thus no collision occurs. The general comments on the evolution of the control and task features signals from the previous cases apply as well for this one. The final error is $\mathbf{e} = [-0.0160; -0.0083]$.

## VI. CONCLUSIONS

In this work we have presented an approach that combines information-theoretic and sensor-based control techniques




(a) (b) (c) (d)

Fig. 7. Pedestrian with perpendicular motion

in order to navigate towards a given goal while avoiding collision with obstacles. On the one hand, the sensor-based formalism allows to define the navigation task and collision avoidance constraints without inherently requiring any knowledge of the car's nor pedestrian's position in a reference fixed frame. On the other hand, the information-theoretic framework allows to naturally handle non-linear and non-smooth dynamics by forward simulating thousands of trajectories in order to find an optimal control sequence.

The obtained results are rather encouraging as the vehicle consistently manages to reach the goal without colliding with the obstacle in many different scenarios. Moreover, the car consistently achieves rather small final $||\mathbf{e}|| \leq 0.021$ values.

Thanks to the relatively lightweight computations required by the sensor-based formalism, our implementation is able to run on a single CPU core of an AMD Ryzen 9 5950X. Nonetheless, in order to handle multiple obstacles future implementations would require some level of parallelization, be it on CPU or GPU. Future work will focus on this issue.

Another interesting research path is to investigate different cost functions to deal with arbitrarily shaped obstacles and social interaction with pedestrians and other vehicles in order to deal with the navigation challenges (safety, comfort, efficiency) that multiple surrounding agents (pedestrians, vehicles, etc) would cause.

## ACKNOWLEDGMENT

This research is part of the HIANIC project, funded by the French National Research Agency (ANR-17-CE22-0010).

## REFERENCES

[1] A. Khalid, "Ford CEO says the company 'overestimated' self-driving cars," 2019. [Online]. Available: https://www.engadget.com/2019/04/10/ford-ceo-says-the-company-overestimated-self-driving-cars/

[2] E. Ackerman, "Toyota's Gill Pratt on Self-Driving Cars and the Reality of Full Autonomy," 2017. [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/self-driving/toyota-gill-pratt-on-the-reality-of-full-autonomy

[3] L. Adouane, "Toward fully autonomous vehicle navigation: From behavioral to hybrid multi-controller architectures," in *11th International Workshop on Robot Motion and Control*. Wasowo,Poland: IEEE, 2017, pp. 85–98.

[4] T. Bellet, M. Cunneen, M. Mullins, F. Murphy, F. Pütz, F. Spickermann, C. Braendle, and M. F. Baumann, "From semi to fully autonomous vehicles: New emerging risks and ethico-legal challenges for human-machine interactions," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 63, no. 2019, pp. 153–164, 2019.

[5] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2016-June, 2016, pp. 1433–1440.

[6] ——, "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[7] R. Kusumoto, L. Palmieri, M. Spies, A. Csiszar, and K. O. Arras, "Informed Information Theoretic Model Predictive Control," in *2019 International Conference on Robotics and Automation (ICRA)*, vol. 2019-May. IEEE, may 2019, pp. 2047–2053.

[8] D. A. de Lima and A. C. Victorino, "Sensor-Based Control with Digital Maps Association for Global Navigation: A Real Application for Autonomous Vehicles," in *IEEE 18th International Conference on Intelligent Transportation Systems*, Las Palmas, Spain, 2015, pp. 1791–1796.

[9] Y. Kang, D. A. de Lima, and A. C. Victorino, "Dynamic obstacles avoidance based on image-based dynamic window approach for human-vehicle interaction," in *IEEE Intelligent Vehicles Symposium*, Seoul, South Korea, 2015, pp. 77–82.

[10] D. Perez-Morales, O. Kermorgant, S. Dominguez-Quijada, and P. Martinet, "Laser-Based Control Law for Autonomous Parallel and Perpendicular Parking," in *Second IEEE International Conference on Robotic Computing*, Laguna Hills, USA, 2018, pp. 64–71.

[11] ——, "Multi-Sensor-based Predictive Control for Autonomous Parking in Presence of Pedestrians," *16th IEEE International Conference on Control, Automation, Robotics and Vision, ICARCV 2020*, pp. 406–413, 2020.

[12] F. Chaumette and S. Hutchinson, "Visual servo control, part I : Basic Approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. December, pp. 82–90, 2006.

[13] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot motion planning and control*, 1998, pp. 171–253.

[14] O. Kermorgant and F. Chaumette, "Dealing with constraints in sensor-based robot control," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 244–257, 2014.

[15] N. Shahriari, S. Fantasia, F. Flacco, and G. Oriolo, "Robotic visual servoing of moving targets," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 77–82.

[16] F. Chaumette and S. Hutchinson, "Visual servo control, part II: advanced approaches'," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[17] M. Prédhumeau, J. Dugdale, and A. Spalanzani, "Modeling and Simulating Pedestrian Social Group Behavior with Heterogeneous Social Relationships," in *Spring Simulation Conference (SpringSim 2020)*, 2020.

[18] B. Okal, T. Linder, D. Vasquez, S. Wehner, O. Islas, and L. Palmieri, "pedsim_ros GitHub repository." [Online]. Available: https://github.com/srl-freiburg/pedsim{_}ros

[19] G. Williams, A. Aldrich, and E. A. Theodorou, "Model Predictive Path Integral Control: From Theory to Parallel Computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[20] I. S. Mohamed, G. Allibert, and P. Martinet, "Model Predictive Path Integral Control Framework for Partially Observable Navigation: A Quadrotor Case Study," apr 2020.