Safe Navigation for a three wheels mobile robot through Hamilton-Jacobi reachability analysis

Gioele Migno, Filippo Ficarola, Leonardo Pio Lo Porto

***Abstract*—Hamilton-Jacobi reachability analysis is a powerful tool able to solve safe navigation problems, in the literature it is mainly applied to aviation scenarios such as collision avoidance. In this paper we introduce in detail the theoretical concepts behind this tool and how to successfully apply it to solve a wheeled mobile robot control problem.**

I. Introduction

During system design, a crucial step is to determine if the system works according to the desired specifications such as performance and safety. This verification step is challenging for many reasons, the first one is the need to take into account all possible system behaviors, this makes necessary the use of a formal verification method since most simulation-based approaches are insufficient. The other main ones are the presence of unpredictable disturbance in practical systems, and the complexity of system dynamics that in general is nonlinear, evolves in continuous time, and has high dimensional state space. Hamilton-Jacobi Reachability Analysis (HJ-RA) is a verification method for guaranteeing performance and safety properties of systems based on reachability analysis, which computes the set of states from which the system can be driven to a target set, while satisfying possible state constraints at all times. HJ-RA overcomes some of the above challenges, it is applicable to general nonlinear systems and easily handles control and disturbance variables, however using a standard implementation, its cost is exponential with respect to the state space dimension.

In this paper we will first introduce all the theoretical fundamentals necessary to HJ-RA (Section II), then how to model and solve a generic reachability problem (Section III) and finally an application of this tool to a wheeled mobile robot (Sections IV, V).

II. Theoretical Fundamentals

In this section, we will introduce all the theoretical fundamentals necessary to understand the theory behind the Hamilton-Jacobi Reachability Analysis.

A. Reachability Analysis

The goal of reachability analysis is to compute the reach-avoid set (RAS) defined as the set of initial states from which the system, using an optimal input, can be driven to a target set within a finite time horizon and satisfying time-varying state constraints at all times. In a reachability problem formulation, the target set can represent a set of undesired states (unsafe), or a set of desired states, in the first case, the RAS contains states to avoid since there exists an optimal input (disturbance) that leads the state into an unsafe region. In the second case instead, the RAS represents safe states from which, applying an optimal input (control law), the system can reach a desired state. In the literature, the just described reach-avoid set, is also called capture basin [7] or backward reachable set (BRS) [1]. In alternative to it, in some cases one might be interested in computing a forward reachable set (FRS), defined as the set of all states that the system can reach from a given initial set of states after a finite time horizon. To understand the difference between BRS and FRS, consider a reachability problem in which the target set contains unsafe states, the FRS can be used to check whether the set of possible future states of the system includes undesired states, the BRS instead, can be used to compute, by starting from known unsafe conditions, those initial states that must be avoided. In this paper HJ-RA is used to compute the BRS of the system, however, it can be used also to compute the FRS. All systems in the real world are subject to a disturbance, hence they have two different kinds of inputs, a controllable one u (control) and another uncontrollable d (disturbance); for that reason the computation of the BRS can be formulated in terms of a two-player game. For instance, consider an aircraft that has to follow a trajectory to complete a task, the system has two inputs: a control input (Player 1) and a disturbance (Player 2), in this scenario, the disturbance could be the wind. Suppose now that there is a goal position to reach (target set) along the trajectory, therefore the control input tries to bring the state at the target and the disturbance to steer it away, in this case, the BRS contains all the initial states for which exists an optimal control command that despite the worst disturbance, brings the system at the goal position. Suppose instead of a goal, there is an obstacle along the trajectory, now the target set is defined by all states of the system that correspond to a collision with the obstacle, therefore, the BRS contains those states which could lead to a collision despite the best possible control action. In both cases, the BRS can be computed by studying the outcome of the game between the two players. Due to the way they are formulated, those two games are "games of kind", namely games in which the outcome is binary: system reaches or not the target set. In order to solve this type of game, it is necessary to translate it into a "games of degree" in which players want to optimize a cost function $J(\cdot)$, with opposite goals: one tries to maximize and the other to minimize it. An approach we will see later, called Level Set Method, can perform this kind of transformation by translating the problem into a standard differential game.

B. Differential Games

In this section we will introduce the basic concepts of the game theory related to a two person, zero-sum differential game, namely a mathematical representation in game theory of a situation which involves two sides, where the result is an advantage for one side and a loss for the other. If the total gains of the participants are added up, and the total losses are subtracted, they will sum to zero.

1) Definition of the differential game: Consider the system dynamics $\dot{x} = f(x(s), u(s), d(s))$ where $s \in [t, T]$ represents the time variable, $x(\cdot) \in \mathbb{R}^n$, $u(\cdot) \in U \subseteq \mathbb{R}^m$, $d(\cdot) \in D \subseteq \mathbb{R}^p$, $f : \mathbb{R}^n \times U \times D \to \mathbb{R}^n$. We use s as the time variable to agree with the standard notation in game theory in which the game starts at time $s = t$ and ends at $s = T$ [6]. The inputs $u(\cdot)$, $d(\cdot)$ represent Player 1 (control) and Player 2 (disturbance) respectively, we assume that they are drawn from the set of measurable functions [6][1]:

$$u(\cdot) \in \mathcal{U}_{[t,T]} \triangleq \{\sigma : [t, T] \to U | \sigma(\cdot) \text{ is measurable } \}$$

$$d(\cdot) \in \mathcal{D}_{[t,T]} \triangleq \{\sigma : [t, T] \to D | \sigma(\cdot) \text{ is measurable } \}$$

Assume U, D are compact, $f(\cdot)$ is bounded and Lipschitz continuous in $x(\cdot)$ and continuous in $u(\cdot)$ and $d(\cdot)$, therefore, the system dynamics admits an unique trajectory $x(\cdot)$ that represents the response of the system to the inputs $u(\cdot)$, $d(\cdot)$ starting from an initial state $x(t) = x$. The differential equation associated to the game is then the following one:

$$\begin{cases} \dot{x}(s) = f(x(s), u(s), d(s)) \quad s \in [t, T] \\ x(t) = x \end{cases} \tag{1}$$

In order to complete the game definition is necessary to introduce the payoff function of the game $J(\cdot)$ and to do that we use two Lipschitz continuous functions $c : \mathbb{R}^n \to \mathbb{R}$ and $q : \mathbb{R}^n \to \mathbb{R}$:

$$J(x, t, u(\cdot), d(\cdot)) = \int_t^T c(x(s), d(s), u(s), s)\, ds + q(x(T)) \tag{2}$$

The payoff function represents the reward/cost obtained by the two players at the end of the differential game (1) started at time t with an initial state x, and subject to the commands $u(\cdot)$, $d(\cdot)$ chosen by Player 1 and Player 2 respectively. The function $c(\cdot)$ indicates a running cost, therefore the first term of (2) is the reward gained during the game (trajectory). The second term instead, uses $q(\cdot)$ to evaluate the final state $x(T)$ reached.

2) Value of the game: In order to solve the game, namely computes its outcome, it's necessary to define the goal and the information pattern of each player. Since we are referring to a zero-sum game, the aims of the players must be opposite, without loss of generality we assume Player 1 ($u(\cdot)$) wants to minimize the cost function $J(\cdot)$ and Player 2 ($d(\cdot)$) tries to maximize it. The information pattern of a game indicates which information a player has respect to its opponent.

In the case of simple games, usually there exists a dominant strategy for each player, namely an optimal strategy that is better than other ones independently to the opponent's strategy. In these cases it is easy to solve the game, since assuming rational players, we know a priori the strategies that will be chosen and therefore we can compute the game outcome in advance. In more complex games like the differential game (1) in which no longer exists dominant strategies, we cannot predict the outcome therefore we have to find another way to solve the game. For this reason in game theory two quantities called lower value $V^-(\cdot)$ and upper value $V^+(\cdot)$ are defined. The lower value $V^-(\cdot)$ indicates the lowest possible outcome of the game (the lowest value of $J(\cdot)$), the upper value $V^+(\cdot)$ is instead the highest one, these values can be defined by giving to a player an advantage respect to the other one. As done in [6] [3] [1] to give a strategic advantage to a player, we impose to him the use of a non-anticipative strategy. This is an advantage because during the game, we assume that each player can choose its own input by knowing the current state of the system (system feedback), and in addition to this information, if a player can use a nonanticipative strategy then it knows also the current input chosen by its opponent [5].

The lower value of the game is defined by giving to the player that wants to minimize the cost function $J(\cdot)$ the strategic advantage, therefore we define a nonanticipative strategy for $u(\cdot)$ in the following way [6]:

$$\begin{aligned} \gamma \in \Gamma \triangleq \Big\{ & \alpha : \mathcal{D}(t) \to \mathcal{U}(t) \,|\, d(r) = \hat{d}(r) \in \mathcal{D}(t) \\ & \text{for almost every } r \in [t, s] \\ & \implies \alpha[d](r) = \alpha[\hat{d}](r) \\ & \text{for almost every } r \in [t, s] \} \end{aligned} \tag{3}$$

What the formula says is that, a map $\alpha : \mathcal{D}(t) \to \mathcal{U}(t)$ is a nonanticipative strategy if for any $s \geq t$, for any $d(\cdot)$ and $\hat{d}(\cdot)$ belonging to $\mathcal{D}$ such that $d(\cdot)$ and $\hat{d}(\cdot)$ coincide almost everywhere on $[t, s]$, the image $\alpha(d(\cdot))$ and $\alpha(\hat{d}(\cdot))$ coincide almost everywhere on $[t, s]$ This restriction means that if player 1 cannot distinguish between input signals $\alpha(d(\cdot))$ and $\alpha(\hat{d}(\cdot))$ of player 2 until after time s, then player 1 cannot respond differently ($\alpha[d](r) = \alpha[\hat{d}](r)$) to those signals until after time s. Or in other words, player 1 cannot respond differently to two different player 2 disturbances until they become different.

Now we can define the lower value of the game $V^-(x, t)$ by allowing $u(\cdot)$ to use $\gamma(\cdot)$:

$$V^-(x, t) \triangleq \inf_{\gamma(\cdot) \in \Gamma(\cdot)} \sup_{d(\cdot) \in \mathcal{D}(\cdot)} J(x, t, \gamma[d](\cdot), d) \tag{4}$$

As mentioned before, in general $V^-(x, t)$ is not the outcome but represents the lower outcome possible of the game started at time t with initial state x. However, it is important to highlight that if we assume the use of the information pattern just described (Player 1 uses nonanticipative), then $V^-(x, t)$ is the actual outcome of the game.

Similarly we can define the upper value of the game $V^+(x,t)$, this time Player 2 ($d(\cdot)$) uses a nonanticipative strategy defined as:

$$\begin{aligned}\delta \in \Delta \triangleq \{\alpha : \mathcal{U}(t) \to \mathcal{D}(t) \,|\, u(r) = \hat{u}(r) \in \mathcal{U}(t) \\ \text{for almost every } r \in [t,s] \\ \implies \alpha[u](r) = \alpha[\hat{u}](r) \\ \text{for almost every } r \in [t,s]\}\end{aligned} \tag{5}$$

The upper value is then defined by giving to $d(\cdot)$ (player that wants maximizes $J(\cdot)$) the strategic advantage:

$$V^+(x,t) \triangleq \sup_{\delta(\cdot)\in\Delta(\cdot)} \inf_{u(\cdot)\in\mathcal{U}(\cdot)} J(x,t,u,\delta[u](\cdot)) \tag{6}$$

For theroretical completeness, we note that $V^-(x,t) \leq V^+(x,t)$ for any (x,t). In those cases in which equality holds, the game is said to have value and $V(x,t) \triangleq V^-(x,t) = V^+(x,t)$ is called the value of the game. [7]

3) Differential Game in Reachability Analysis: Let's try now to understand how differential games theory is related to a reachablity problem, to do that assume we are in the case in which we have a set to reach, namely the target set is a desired set of states. In this case Player 1 tries to bring the system towards the target with its input $u(\cdot)$ and in the meanwhile the disturbance (Player 2) tries to steer the system away the target with its input $d(\cdot)$. This problem can be through as a game between the two players, however in this form, the game theory just described cannot be used since the reward function is a simple boolean value: at the end of the game the system state reaches (Player 1 wins) or not the target set, so we have $J(\cdot) = \{0 \text{ (win)}, 1 \text{ (loss)}\}$. To solve this issue we can easily use the Level Set Method that allows to rewrite $J(\cdot)$ as a real function, we will see this later. For the moment assume Player 1 wants to minimize a real value function $J(\cdot)$ and Player 2 tries to maximize it. Now we have to understand why the lower and the upper value of the game are important for our purpose. As said before, $V^-(x,t)$ or $V^+(x,t)$ is the actual outcome of the game only if we give a strategic advantage to one player, therefore introducing an information pattern for our game, we can easily define formally the outcome through one of them. In order to consider always the worst possible scenario, the strategic advantage is usually given to the disturbance $d(\cdot)$, then since its aim is to maximize the cost function, the outcome of the game is defined by $V^+(x,t)$. In a different scenario in which $u(\cdot)$ maximizes and $d(\cdot)$ minimizes $J(\cdot)$ then, since the strategic advantage must be assigned to the disturbance, the outcome this time is given by $V^-(x,t)$. In this paper we always refer to the first scenario.

As we will see later the BRS is related to the outcome of the game ($V^+(x,t)$), this allows us to solve our reachability problem using the upper value of the game. One important result of dynamic programming is that the value functions $V^{\pm}(x,t)$ are the viscosity solutions of two particular variational inequalities in the form of a Hamilton-Jacobi-Isaacs equation. We will see this formally later, let's now introduce what is a HJI equation, a viscosity solution and how to use the Level Set Method.

C. Hamilton-Jacobi Equation

In physics, the Hamilton-Jacobi equation is an alternative formulation of classical mechanics. It is a first-order nonlinear partial differential equation of the form $H(x, u_x(x,\alpha,t),t) + u_t(x,\alpha,t) = K(\alpha,t)$ with independent variables $(x,t) \in \mathbb{R}^n \times \mathbb{R}$ and parameters $\alpha \in \mathbb{R}^n$. It has wide applications in many scientific fields like mechanics. Its solutions determine infinite families of solutions of Hamilton's ordinary differential equations, which are the equations of motion of a mechanical system. In our case, the family of solutions that come from the computation of the HJI PDE and the variational inequality is the viscosity solution.

D. Viscosity Solution

Viscosity solution is a concept introduced in [8]; it is a notion of solution that allows it to be, for example, nowhere differentiable but for which strong uniqueness theorems, stability theorems and general existence theorems are all valid. It is a generalization of the solution to a partial differential equation (PDE). It has been found that the viscosity solution is that kind of solution to use in many applications of PDE's, including for example first order equations arising in dynamic programming (the Hamilton–Jacobi–Bellman equation) or differential games (the Hamilton–Jacobi–Isaacs equation). This means that, given the classical concept of PDE

$$F(y, u(y), Du(y)) = 0 \tag{7}$$

Under the viscosity solution concept, a certain variable u does not need to be everywhere differentiable. There may be points where Du does not exist and yet u satisfies the equation in an appropriate generalized sense.

Taking the main concepts of viscosity solutions from [8], let u be a function from O into $\mathbb{R}$ and let y_0 belong to $\mathcal{O}$. The superdifferential and subdifferential of u at y_0, denoted, respectively, by $D^+u(y_0)$ and $D^-u(y_0)$ are the set of points for which, respectively, the following two inequalities hold:

$$\limsup_{y\to y_0}(u(y) - u(y_0) - p_0 \cdot (y - y_0))|y - y_0|^{-1} \leq 0 \tag{8}$$

and

$$\liminf_{y\to y_0}(u(y) - u(y_0) - p_0 \cdot (y - y_0))|y - y_0|^{-1} \geq 0 \tag{9}$$

where $p_0 \cdot (y - y_0)$ is the Euclidean scalar product of p_0 and $(y - y_0)$. Given these concepts it is possible to define what is a viscosity solution mathematically. If $D^+u(y_0)$ and $D^-u(y_0)$ are nonempty at some x and u is differentiate at x, a viscosity solution of (7) is a function u belonging to $C(\mathcal{O})$ satisfying the two following conditions:

$$F(y, u(), p) \leq 0 \forall y \in \mathcal{O}, \forall p \in D^+u(y) \tag{10}$$

$$F(y, u(), p) \geq 0 \forall y \in \mathcal{O}, \forall p \in D^-u(y) \tag{11}$$

E. Level Set Method

As mentioned before, in order to compute the BRS is necessary to solve a game of kind where the outcome is boolean: the system state either reaches the target set or not. Level Set Method can be used to translate this game into a game of degree, where players share an objective function to optimize. The basic idea of this approach is to encode the boolean outcome through a quantitative function $g(\cdot)$ and compare its value at the end of the game to a threshold value, usually zero, to determine whether the system reached the target set. The first step is to define a Lipschitz function $g(x(\tau))$, where $x(\tau)$ represents the current system state ($s = \tau \in [t, T]$), such that the target set R corresponds to the zero sublevel set of $g(x(\tau))$, that is, $x(\tau) \in R \Leftrightarrow g(x(\tau)) \leq 0$. We indicated the target set with R (reach) since from now on we suppose that the set contains goal states, namely states to reach. Now we can define the real value cost function of the game $J(\cdot)$, we are not interested in any kind of running cost, therefore we consider only the value of $g(\cdot)$ at the end of a game started at time $s = t$ and initial state x:

$$J(x, t, u(\cdot), d(\cdot)) = g(x(T)) \tag{12}$$

Assuming our reachability problem consists only of reaching a target set R, then the goal of our control input $u(\cdot)$ is to minimize $J(\cdot)$ while the disturbance $d(\cdot)$ tries to do the opposite. Then giving the strategic advantage to $d(\cdot)$, namely force it to use the non-anticipative strategy $\delta[d](\cdot)$, we define the outcome of the game through its upper value:

$$\begin{aligned} V^+(x,t) &= \sup_{\delta(\cdot)\in\Delta(\cdot)} \inf_{u(\cdot)\in\mathcal{U}(\cdot)} J(x,t,u,\delta[u](\cdot)) \\ &= \sup_{\delta(\cdot)\in\Delta(\cdot)} \inf_{u(\cdot)\in\mathcal{U}(\cdot)} g(x(T)) \end{aligned} \tag{13}$$

However, in practical scenarios, along the trajectory of a dynamical system there may be both goals to reach and obstacles to avoid. The goals to reach can be represented by the target set R as previously done, the set of states to avoid instead, can be defined with another set A (avoid) that contains all the system state x that corresponds to an object collision. This kind of formulation is adopted in [3] however, we will use another one from [7] in which instead of A, its complementary set K is defined. This choice is due to the fact that [7] provides a more advanced problem formulation allowing sets R and K to be time-varying, namely allows to define a reachability problem in which both the goals and the obstacles can vary their positions over time. According to this new formulation, since R can vary over time, also the function $g(\cdot)$ must be time-dependent. The new set K is characterized similarly to R by intoducing another function $h(\cdot)$. Formally: consider the sets R, K related respectively to the level sets of two Lipschitz continuous and bounded functions $g : \mathbb{R}^n \times s \in [t, T] \to \mathbb{R}$, $h : \mathbb{R}^n \times s \in [t, T] \to \mathbb{R}$, then the two sets can be characterized as:

$$R = \{(x, s) \in \mathbb{R}^n \times [t, T] \,|\, g(x, s) \leq 0\} \tag{14}$$

$$K = \{(x, s) \in \mathbb{R}^n \times [t, T] \,|\, h(x, s) \leq 0\} \tag{15}$$

The most common choice for the functions $g(\cdot)$ and $h(\cdot)$ is to use the distance between the current state x and the set of interested, namely:

$$g(x, s) = \begin{cases} -d(x, s, R^c) & \text{if } x \in R \\ d(x, s, R) & \text{if } x \in R^c \end{cases} \tag{16}$$

$$h(x, s) = \begin{cases} -d(x, s, K^c) & \text{if } x \in K \\ d(x, s, K) & \text{if } x \in K^c \end{cases} \tag{17}$$

Since $g(\cdot)$, $h(\cdot)$ must be bounded, we will see later why, we can introduce two constants C_g C_h to impose a saturation to the distance functions, or alternatively, we can use the arctangent of the signed distance, in this way the resulting functions are bounded and also globally Lipschitz [2].

In the next section we will see how the value function $V^+(\cdot)$ is formulated when we have both a reach R and a constraints K set, and most importantly how it can be computed in order to calculate the BRS. In the following sections we will refer to the BRS as a reach-avoid set (RAS) to highlight the fact that there is both a set to reach and one to avoid, this last is implicity defined by its complementary K, namely $A \triangleq K^c$.

III. Computation Reach-Avoid set

In this section we will see how all the previous theoretical concepts can be applied to solve a reachability problem, in which the system state has to reach a desired set R while remains in the constraints set K that represents the complementary of the set of states A that correspond to a collision between the system and an obstacle in the real world.

Given the previous specifications, we can define two different kinds of reachability problems. In the first one we are interested in reaching safely R exactly at the end of the game ($s = T$), in the second instead, the system can reach safely R at any s inside the time horizon $[t, T]$. In the following we will focus our attention only on this last one.

Consider the system dynamics (1), the two functions $g(\cdot)$, $h(\cdot)$ defined in (16)(17) and used to represent the target set R and the constraint set K respectively. As said before, under these conditions, the system dynamics admits a unique trajectory $x(s)$ from an initial state x at time t under inputs $u(\cdot)$ and $d(\cdot)$. We denote this solution as:

$$\phi(s; x, t, u(\cdot), d(\cdot)) : [t, T] \to \mathbb{R}^n$$

A. Payoff Function

We have already seen the payoff function of a reachability problem in which there is only a set to reach (12), now we have to consider also the constraints set K and to do that we can use the following cost function [7]:

$$J(x, t, u(\cdot), d(\cdot)) = \min_{\tau\in[t,T]} \{\mathcal{J}(\tau, x, t, u(\cdot), d(\cdot))\} \tag{18}$$

$$\mathcal{J}(\tau, x, t, u(\cdot), d(\cdot)) = \max\left\{g(\mathcal{X}(\tau), \tau), \max_{r\in[t,\tau]} h(\mathcal{X}(\tau), r)\right\} \tag{19}$$

Where:

$$\mathcal{X}(\tau) = \phi(\tau; x, t, u(\cdot), d(\cdot))$$

The $J(\cdot)$ function is used to evaluate the game and therefore determines if Player 1 ($u(\cdot)$) won or lost. The expression (18) considers the value of another function $\mathcal{J}(\cdot)$ for all time $\tau \in [t, T]$ during the game. For each instant τ, $\mathcal{J}(\cdot)$ takes the maximum between two quantities, the first one uses $g(\cdot)$ to determine if the current system state is inside the reach set R and the second instead, uses $h(\cdot)$ to check whether or not the state had ever left the constraints set K so far, namely on time interval $[t, \tau]$. The maximum value of $h(\cdot)$ is taken in order to keep in mind a previous collision before τ, indeed we recall that $(x, s) \in K \Leftrightarrow h(x, s) \leq 0$, therefore a positive value of $h(\cdot)$ indicates an exit from K (obstacle collision). So, for a given $\tau \in [t, T]$, we have $\mathcal{J} \leq 0$ if and only if, the system trajectory started at time t with initial state x, reaches the target R at time τ without ever having collided with an obstacle on $[t, \tau]$, if this situation takes place for any $\tau \in [t, T]$ then Player 1 wins. For that reason, $J(\cdot)$ takes the minimum time τ, in this way the control input $u(\cdot)$ can win at any moment during the game. It is important to highlight that $J(\cdot)$ does not take into account any collision after the system reached the target set R.

Once defined the cost function of the game, $V^-(x, t)$ and $V^+(x, t)$ are defined in the same way of (4) (6) respectively.

B. Definition Reach-Avoid Set

We are finally ready to define formally the RAS. As said before, we assume $u(\cdot)$ tries to minimize the distance of the system state to R, and $d(\cdot)$ tries to do the opposite, potentially also bringing the state outside the constraints set $K = A^c$. In order to consider the worst possible case, we give the strategic advantage to the disturbance, in this way the outcome of the game is given by $V^+(x, t)$ and using it we can compute RAS^+. For completeness, we also report the definition of RAS^- namely the reach-avoid set computed through the lower value of the game $V^-(x, t)$.

Given an information pattern, we define the reach-avoid set as the set of point (x, t) for which the system trajectory $\phi(\cdot)$, starting at time t with initial state x and both players acting optimally, reaches the target set R at some instant $\tau \in [t, T]$ while remaining inside the constraints set K for all $s \in [t, \tau]$. We indicate with $\mathcal{R}_s$, $\mathcal{K}_s$ the target set R and the constraints set K at time s respectively.

When the strategic advantage is given to the player that wants to maximize the outcome ($d(\cdot)$), we indicate the reach-avoid set with RAS^+ that is formally defined as [7]:

$$\begin{aligned} RAS^+ = \big\{(x,t) \in \mathbb{R}^n \times [0,T] \,|\, \exists u^*(\cdot) \in \mathcal{U}_{[0,T]}, \\ \forall \delta(\cdot) \in \Delta_{[0,T]}, \exists \tau \in [t,T], \phi(\tau; x, t, u^*(\cdot), \delta(\cdot)) \in \mathcal{R}_\tau \quad \wedge \\ \forall s \in [t,\tau], \phi(s; x, t, u^*(\cdot), \delta(\cdot)) \in \mathcal{K}_s\big\} \end{aligned} \tag{20}$$

Similarly, when the player that tries to minimize uses a nonanticipative strategy:

$$\begin{aligned} RAS^- = \big\{(x,t) \in \mathbb{R}^n \times [0,T] \,|\, \exists \gamma^*(\cdot) \in \Gamma_{[0,T]}, \\ \forall d(\cdot) \in \mathcal{D}_{[0,T]}, \exists \tau \in [t,T], \phi(\tau; x, t, \gamma^*(\cdot), d(\cdot)) \in \mathcal{R}_\tau \quad \wedge \\ \forall s \in [t,\tau], \phi(s; x, t, \gamma^*(\cdot), d(\cdot)) \in \mathcal{K}_s\big\} \end{aligned} \tag{21}$$

These two formulations of the reach-avoid set are related to the outcome of the game through the following proposition [7]:

Proposition 1. *The reach-avoid set $RAS^\pm$ is given by the zero sublevel set of the game's outcome:*

$$RAS^\pm = \big\{(x,t) \in \mathbb{R}^n \times [0,T] \,|\, V^\pm(x,t) \leq 0\big\}$$

These two reach-avoid sets are related, indeed the reach-avoid set computed when the strategic advantage is given to the disturbance it is a subset of the one computed given the advantage to the control input [7]:

$$RAS^+ \subseteq RAS^- \tag{22}$$

This relation is the reason why we have chosen to give the advantage to the disturbance, in this way we increase the safety margin.

C. Hamilton-Jacobi-Isaacs Equation

Given an information pattern, the reach-avoid set $RAS^\pm$ is described by the zero sublevel of the game's outcome $V^\pm(x, t)$. This last is the viscosity solution of a specific Hamilton-Jacobi-Isaacs equation [7]:

Theorem 1. *Assume $f(\cdot)$ defined as done in (1), $g(x, t)$, $h(x, t)$ globally Lipschitz continuous. Then the value function $V^\pm(x, t)$ is the unique viscosity solution of the following variational inequality in the form of a Hamilton-Jacobi-Isaacs equation:*

$$\max\{\mathcal{H}, \Delta h\} = 0$$

$$\mathcal{H} \triangleq \min\left\{\frac{\partial V^\pm(x,t)}{\partial t} + H^\pm\left(x, \frac{\partial V^\pm(x,t)}{\partial x}, t\right), \Delta g\right\}$$

$$\Delta h \triangleq h(x,t) - V^\pm(x,t), \quad \Delta g \triangleq g(x,t) - V^\pm(x,t)$$

$$t \in [0,T] \quad x \in \mathbb{R}^n$$

with terminal condition

$$V^\pm(x,T) = \max\{g(x,T), h(x,T)\} \quad x \in \mathbb{R}^n$$

The upper and lower Hamiltonians $H^\pm$ are given by:

$$H^+(x,p,t) = \min_{u\in U} \max_{d\in D} p^T f(x,u,d) \tag{23}$$

$$H^-(x,p,t) = \max_{d\in D} \min_{u\in U} p^T f(x,u,d) \tag{24}$$

The algorithm of the numerical method able to compute the value function $V^\pm(x, t)$ in Th. 1 is described in [7]. Please notice here we use different notations for target and constraints functions.

IV. Case of Study

In order to use HJ-RA to solve a reachability problem in the case of a robot, we have to follow several steps. The first one is the definition of the game dynamics, that in the case of wheeled mobile robot, is usually given by its kinematic model where control input is the wheels velocities. The second step is the definition and computation of the Hamiltonian, in our case the definition is (24) since we assume u wants to minimize and d to maximize. In addition to that, the numerical method [7] able to compute $V^+(x,t)$ requires also computation of its partial derivative w.r.t. the vector p. The third step is the environment definition, namely definition of $g(x,s)$ and $h(x,s)$ that represent respectively the reach set $\mathcal{R}_s$ and the constraint set $\mathcal{K}_s$ at the time s. The final step is the numerical computation of $V^+(x,t)$ using the algorithm in [7], to do that we can use two Matlab libraries: ToolboxLS [11] and helperOC [1].

Once computed $V^+(x,t)$, in a limited and discretized state-time space, we can use this value function to define a safe control law from each feasible initial states x_0 to the reach set.

A. Definition Game Dynamics

Our case of study focuses on a three-wheeled omnidirectional mobile robot [4] [12] which is a holonomic robot that has the ability to move simultaneously and independently in translation and rotation. The robot has three omni-wheels equally disposed at 120°, and have a distance L from their center and the robot's center of mass $r \equiv O_m$.

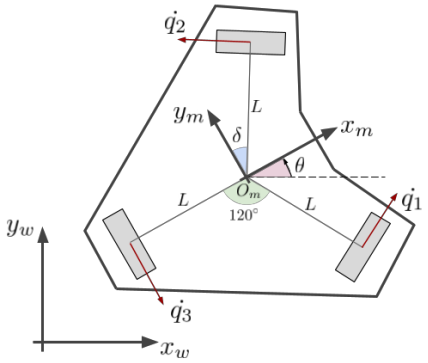

Fig. 1. Three-wheeled omnidirectional mobile robot

As shown in Fig. 1, there are two different reference frames: the first $RF_m \triangleq [O_m, X_m, Y_m]$ is a mobile frame placed at the robot's center of mass, the second one is the fixed world coordinate system $RF_w \triangleq [O_w, X_w, Y_w]$. Robot orientation w.r.t to RF_w is given by the angle θ. The angle δ is equal to 30° and indicates the wheel orientation w.r.t. RF_m.

The kinematic model w.r.t. RF_w is deduced as follows:

$$\dot{x} = \begin{bmatrix} \frac{2}{3}\cos(\theta+\delta) & -\frac{2}{3}\cos(\theta-\delta) & \frac{2}{3}\sin\theta \\ \frac{2}{3}\sin(\theta+\delta) & -\frac{2}{3}\sin(\theta-\delta) & -\frac{2}{3}\cos\theta \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \dot{q} \quad (25)$$

$$+ \begin{bmatrix} -sin\theta \\ cos\theta \\ 0 \end{bmatrix} d$$

Where system state is given by $x = [^w x_r, ^w y_r, \theta]^T$, and the control input $u = [u_1, u_2, u_3]^T = \dot{q} = [\dot{q}_1, \dot{q}_2, \dot{q}_3]^T$ The second term represents an external disturbance given by a velocity with module d and perpendicular to robot orientation θ (i.e. it lies on Y_m). We assume both inputs to be limited in an interval as required by game theory: $u \in U = [u_{min}, u_{max}]$, $d \in D = [d_{min}, d_{max}]$. The procedure to calculate (25) is shown in the Appendix.

B. Hamiltonian

The game dynamics $f(x(\cdot), u(\cdot), d(\cdot))$ is given by (25), therefore the upper Hamiltonian can be computed as follows:

$$H^+(x,p,t) = \min_{u\in U}\max_{d\in D}[p_x, p_y, p_\theta] f(x,u,d)$$

Developing and collecting terms w.r.t. inputs [1]:

$$\begin{aligned} H^+(\cdot) = \min_{u\in U}\max_{d\in D} & \left(\tfrac{2}{3}c(\theta+\delta)p_x + \tfrac{2}{3}s(\theta+\delta)p_y + \tfrac{1}{3L}p_\theta\right) u_1 \\ & + \left(-\tfrac{2}{3}c(\theta-\delta)p_x - \tfrac{2}{3}s(\theta-\delta)p_y + \tfrac{1}{3L}p_\theta\right) u_2 \\ & + \left(\tfrac{2}{3}s(\theta)p_x - \tfrac{2}{3}c(\theta)p_y + \tfrac{1}{3L}p_\theta\right) u_3 \\ & + (c(\theta)p_y - s(\theta)p_x)d \end{aligned}$$

For the sake of brevity:

$$H^+(\cdot) = \min_{u\in U}\max_{d\in D} \sigma_{u1}u_1 + \sigma_{u2}u_2 + \sigma_{u3}u_3 + \sigma_d d$$

Since inputs appear linearly in the Hamiltonian, it is easy to solve the min-max problem, in other cases could be necessary to use optimization numerical tools:

$$H^+(\cdot) = \sigma_{u1}u_1^* + \sigma_{u2}u_2^* + \sigma_{u3}u_3^* + \sigma_d d^* \quad (26)$$

Where $u^* = [u_1^*, u_2^*, u_3^*]^T$ and d^* represent the optimal moves for the two players and are given by:

$$u_i = \begin{cases} u_{min} & \text{if } \sigma_{ui} \geq 0 \\ u_{max} & \text{Otherwise} \end{cases} \quad (27)$$

$$d = \begin{cases} d_{min} & \text{if } \sigma_d \leq 0 \\ d_{max} & \text{Otherwise} \end{cases} \quad (28)$$

Please notice that these optimal inputs, as well as the Hamiltonian, cannot be computed at prior since first they require the computation of $V^+(x,t)$ necessary to calculate $p = [p_x, p_y, p_\theta]^T$.

Finally, as required from the numerical method able to compute $V^+(x,t)$, we calculate the partial derivative of (26)

[1] $c(\cdot) = \cos(\cdot)$, $s(\cdot) = \sin(\cdot)$

w.r.t. to p. Please refer to [7] [10] for the mathematics reasons behind it.

$$\frac{\partial H^+(x,p,t)}{\partial p} = \left[\frac{\partial H^+(\cdot)}{\partial p_x}, \frac{\partial H^+(\cdot)}{\partial p_y}, \frac{\partial H^+(\cdot)}{\partial p_\theta}\right] \tag{29}$$

$$\begin{aligned}\frac{\partial H^+(x,p,t)}{\partial p_x} =& \frac{2}{3}cos(\theta+\delta)u_1^* - \frac{2}{3}cos(\theta-\delta)u_2^* \\ &+ \frac{2}{3}sin(\theta)u_3^* - sin(\theta)d^*\end{aligned}$$

$$\begin{aligned}\frac{\partial H^+(x,p,t)}{\partial p_y} =& \frac{2}{3}sin(\theta+\delta)u_1^* - \frac{2}{3}sin(\theta-\delta)u_2^* \\ &- \frac{2}{3}cos(\theta)u_3^* + cos(\theta)d^*\end{aligned}$$

$$\frac{\partial H^+(x,p,t)}{\partial p_\theta} = \frac{u_1^* + u_2^* + u_3^*}{3L}$$

We treat the final two steps of the case of study, namely environment definition and implementation, in the next section.

V. Simulation

In all simulations, $L = 1$, control input u is limited in $U = [-\overline{u}, +\overline{u}]$ where $\overline{u} = 1.5$ and the disturbance d in $D = [-\overline{d}, +\overline{d}]$ where $\overline{d} = 0.15$. In order to take into account robot dimensions in the configuration space ($\mathcal{C}$-space), we approximate its shape as a circle with $radius = (4/3)L$.

Our Simulations are divided in two main cases:

- The first one takes place in a static environment, where we have fixed obstacles and a fixed target set
- The second one takes place in a dynamic environment with moving obstacles and a moving target set.

The second case contains three sub-cases:

- The Robot starts near the target set
- The Robot starts distant from the target set
- The Robot starts outside the RAS (Reach-Avoid Set)

In addiction, for some simulations we test two disturbances strategies:

- Optimal Disturbance (28)
- Random Disturbance

The random disturbance is sampled from a Normal Gaussian distribution, with mean $\mu = 0$ and standard deviation $\sigma = \overline{d}/3$. σ has this form since in this way most of the samples (99.7%) are in D, those outside are clipped.

All simulations have been implemented in Matlab using both ToolboxLS [11] and helperOC [1] libraries. The code (*main.m*) is available in the code folder of https://github.com/gioelemigno/AMR_project.

A. Static Case

As mentioned before in this case the robot has to reach a fixed target set which is among some rectangular fixed obstacles. The simulation develops in a time of 10 seconds. The first step is the computation of the value function $V^+(x,t)$ from which we can obtain the RAS, Figs. (2), (3), (4) show the static environment in $\mathcal{C}$-space and RAS regression during the game in different instants of time. The blue contour represents the border of RAS, so from any initial configuration $x_0 = [x_{r0}, y_{r0}, \theta_{r0}]^T$ inside the blue border our robot is expected to reach the target set in the time limit of 10 seconds despite the disturbance.

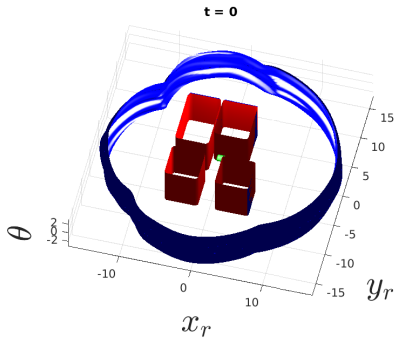

Fig. 2. Static environment and RAS at game start. The two horizontal axes represent the two first robot state components (x_r, y_r), while the third axis represents the third component θ, namely the angle that the X-axis of RF_m forms with RF_w. The small green cuboid represents the target set and the four red cuboids represents the obstacles. Since this representation takes place in $\mathcal{C}$-space the obstacles are bigger than their real dimensions in the workspace and the target set seems smaller. Blue contour indicates the border of RAS

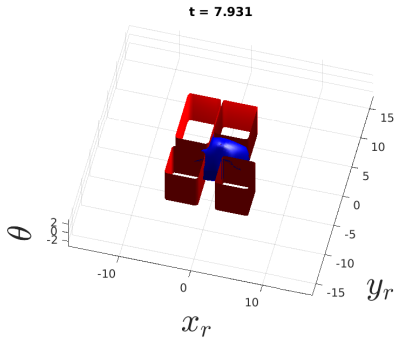

Fig. 3. Static environment and RAS at time 7.931s

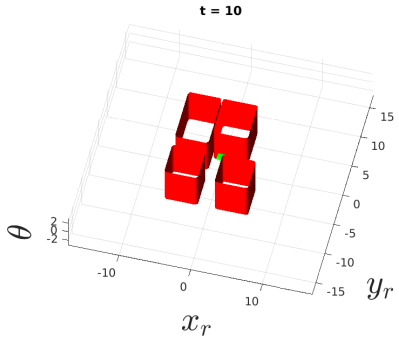

Fig. 4. Static environment and RAS (empty) at end game

In $\mathcal{C}$-space the target set has a limited height $\theta \in [-0.1, 0.1]$ since we want the robot reaches the target set with a desired orientation θ_d in a small range around $0\,rad$.

Now that the RAS is defined we can see how the robot, starting from an initial state x_0 at time $t = 0$, reaches the target set avoiding obstacles despite the disturbance. So since the robot's initial position $x_0 = [-5, -8 - 2]$ is in the limits of the RAS we can see that it reaches, after the needed time, the target set in the desired orientation, avoiding the obstacles in its path. Fig. (8) shows the trajectory in both configuration and work-space and also the optimal player moves at each time instant of the game. In Fig. (9) is shown the same experiment but instead of an optimal disturbance we use a random one.

B. Dynamic Case

Now we will see some more complex cases where reach set and avoid set are not fixed, but they move in the environment. Here we simulated a moving platform which represent a recharging station for the robot. So the robot has to access the platform while this platform is moving downward along the Y-axis with a constant velocity $v = -1$. The platform is not accessible from everywhere, but only from the right side, while the other sides are surrounded by barriers (obstacles). The final desired orientation is the same of static environment ($\theta_d \in [-0.1, 0.1]$), so the target set is not restricted only in small area of the X-axis and Y-axis, but also in terms of the third component θ.

As done before, the first step is the computation of RAS at each instant of the game, in this case we are going to see the RAS regressing over time in a particular direction since the target set, always represented by a green area, can be reached from only one side, see Figs. (5), (6), (7).

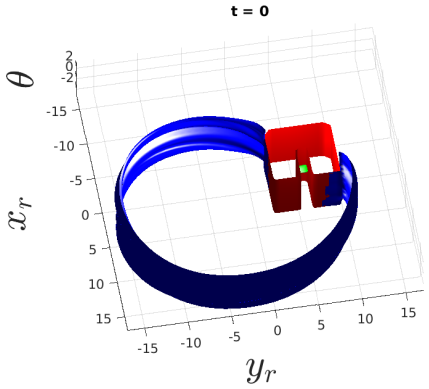

Fig. 5. Dynamic environment and RAS at game start

As mentioned before, we simulated three different cases for the Dynamic experiments.

1) Case 1: In this first case the initial position of the robot $x_0 = [0, 0, -3]$ is relatively near the target set and inside the RAS at time 0, so we know from the theory that exists a trajectory which can lead the robot inside the target set in the maximum given time (10 seconds). In Fig. (10) we can see the particular trajectory of the robot necessary to reach the moving target set despite an optimal disturbance. As expected, at time 10 the robot is in the target set with the desired orientation without collisions with the obstacles.

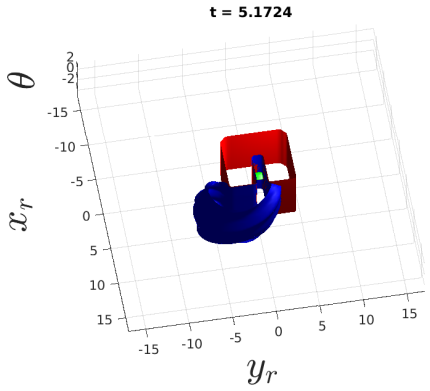

Fig. 6. Dynamic environment and RAS at time 5.1724s

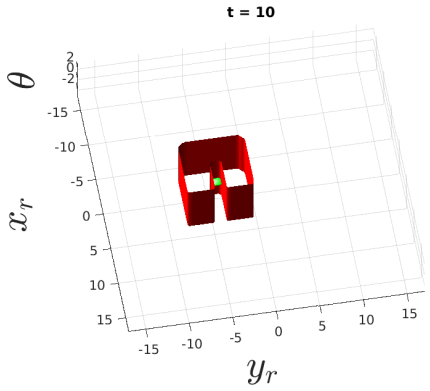

Fig. 7. Dynamic environment and RAS (empty) at end game

2) Case 2: The second case is similar to the previous one, the only difference is the initial position of the robot ($x_0 = [0, -7, -3]$). Here the robot is relatively distant from the target set, but still in the RAS. Even in this case the robot successfully reached the target set with the desired angle θ_d, playing against the optimal disturbance and avoiding any obstacle (Fig. (11)). The robot trajectory when the disturbance acts random is shown in Fig. (12).

3) Case 3: In this last case we see for completeness what happens if the initial position of the robot is outside the RAS. The initial position of the robot is $x_0 = [14, 14, -3]$, so completely outside the bounds of the RAS, obviously we can already say that the robot is not going to reach the target set.

As we can see from Fig. (13), robot tries to reach the target but at time 10 it is still outside.

VI. Conclusion

In this paper we first introduced all the basic concepts of differential games necessary to Hamilton-Jacobi reachability, then we showed how to model a reachability problem in the more complex case, namely in the presence of a reach set and an avoid set, both time-varying. Then we concluded with the case of study and simulations of a three-wheeled robot in two different type of environments (static and dynamic).

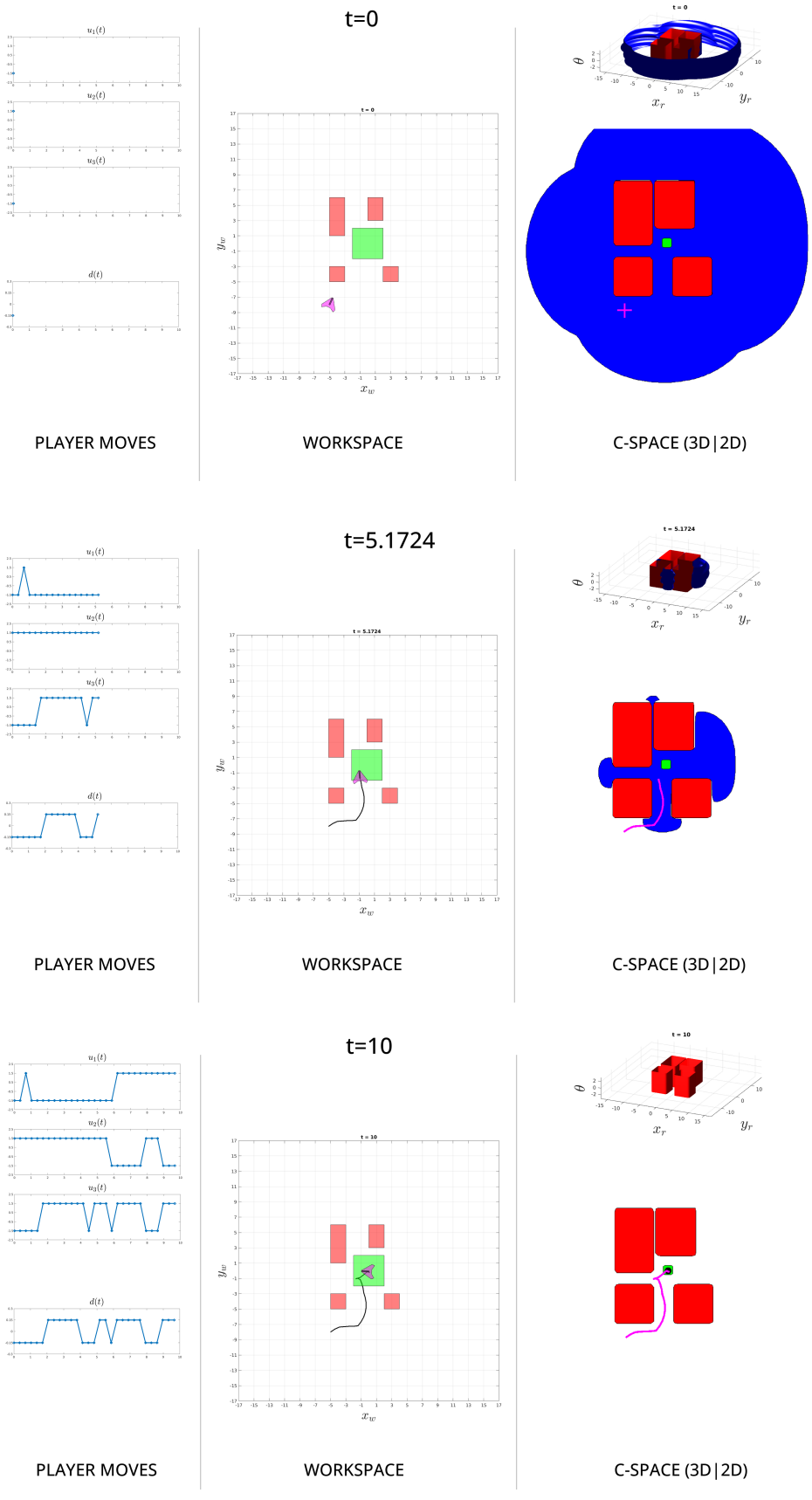

Fig. 8. Static environment, optimal disturbance. $x_0 = [-5, -8 - 2]$, $\theta_d \in [-0.1, 0.1]$

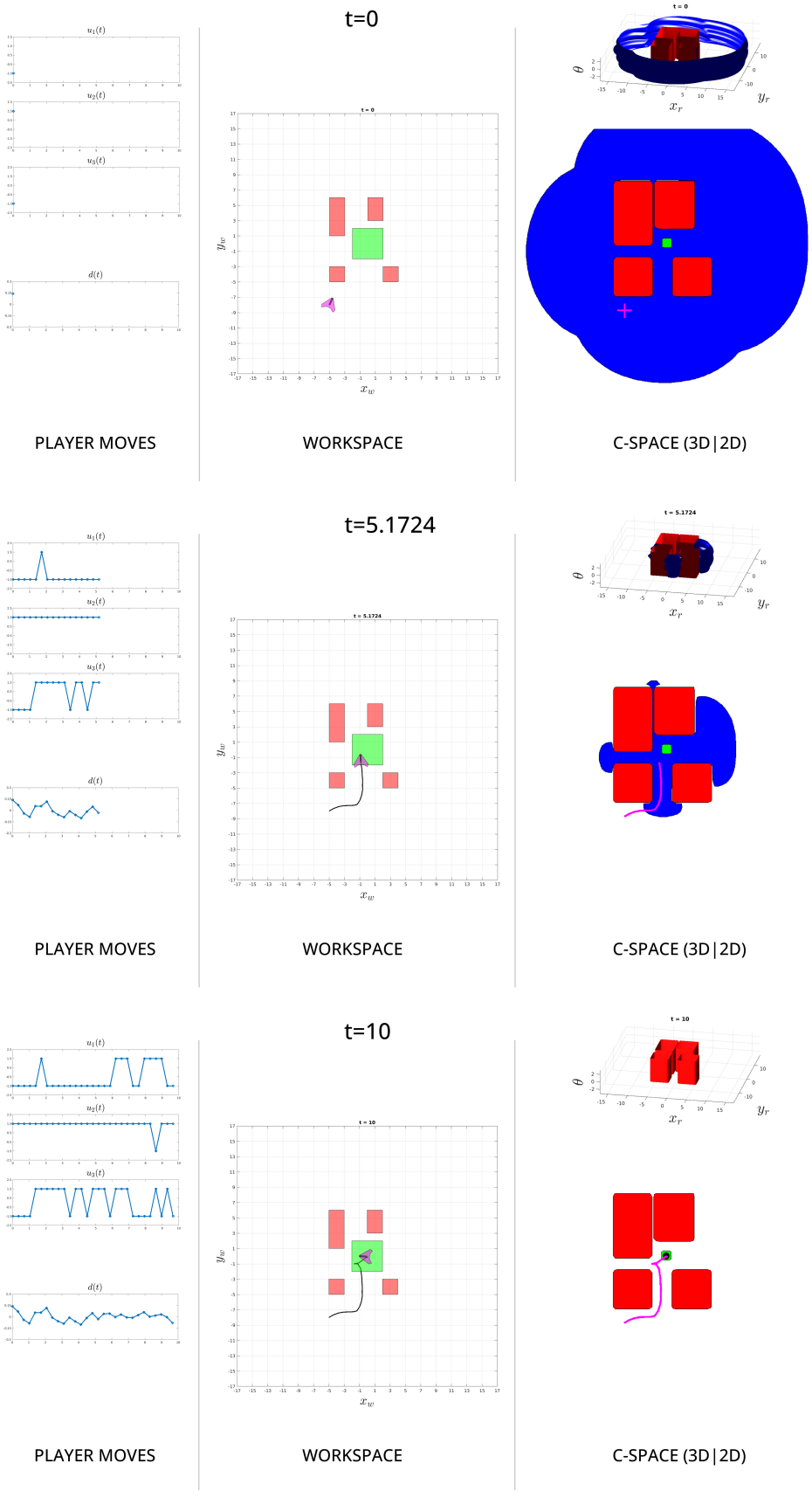

Fig. 9. Static environment, random disturbance. $x_0 = [-5, -8 - 2]$, $\theta_d \in [-0.1, 0.1]$

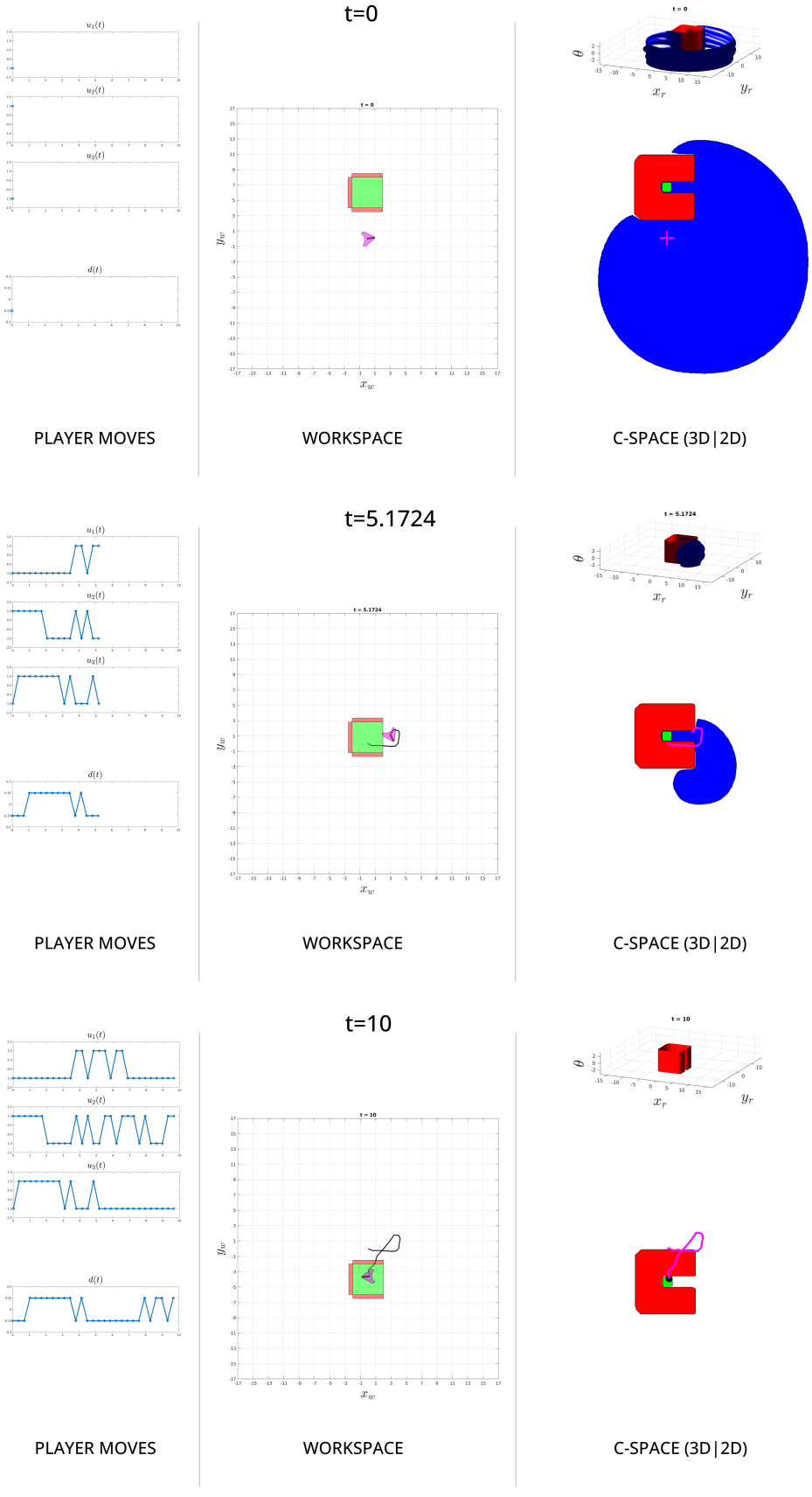

Fig. 10. Dynamic environment, optimal disturbance. $x_0 = [0, 0, -3]$, $\theta_d \in [-0.1, 0.1]$

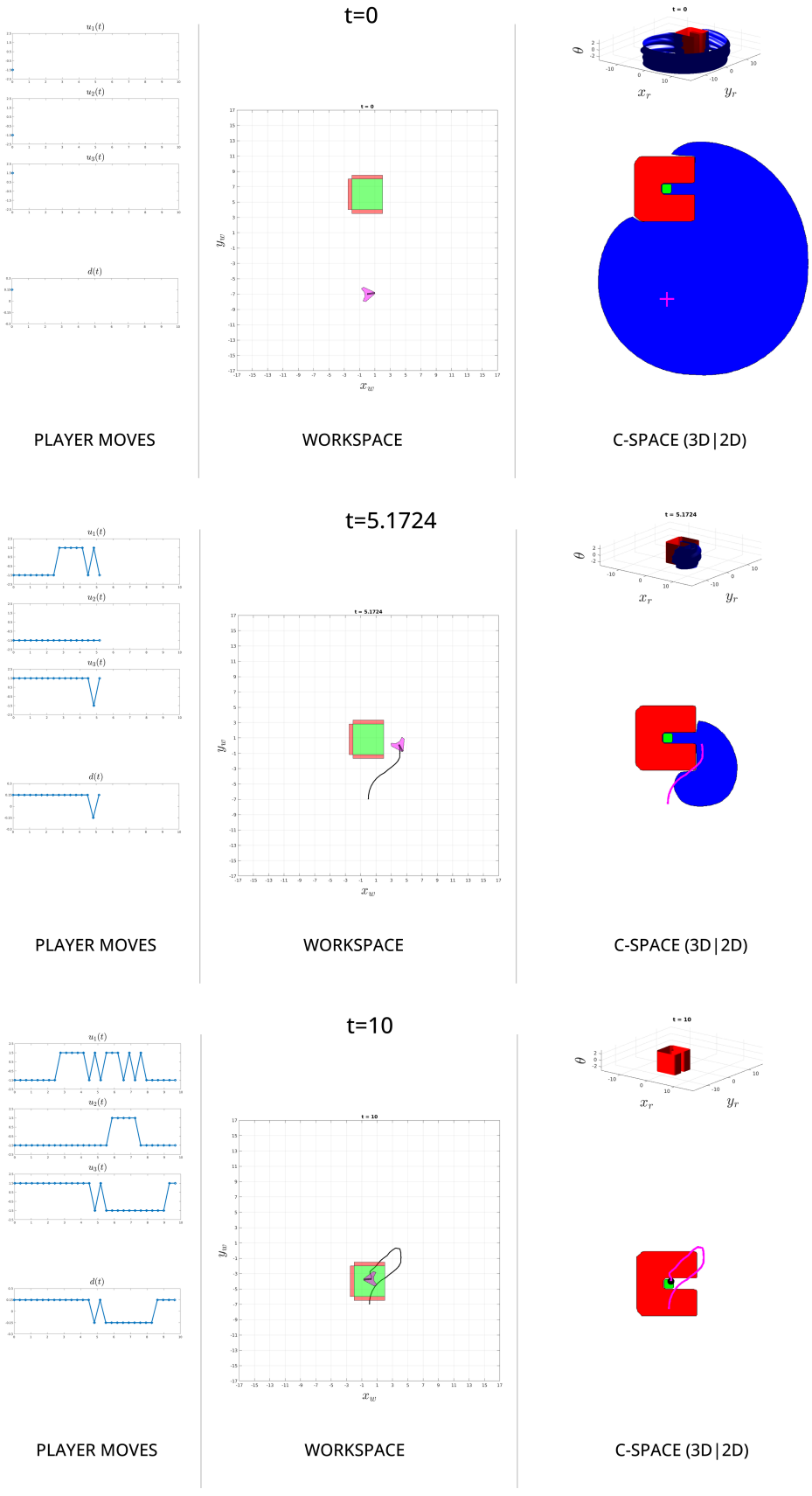

Fig. 11. Dynamic environment, optimal disturbance. $x_0 = [0, -7, -3]$, $\theta_d \in [-0.1, 0.1]$

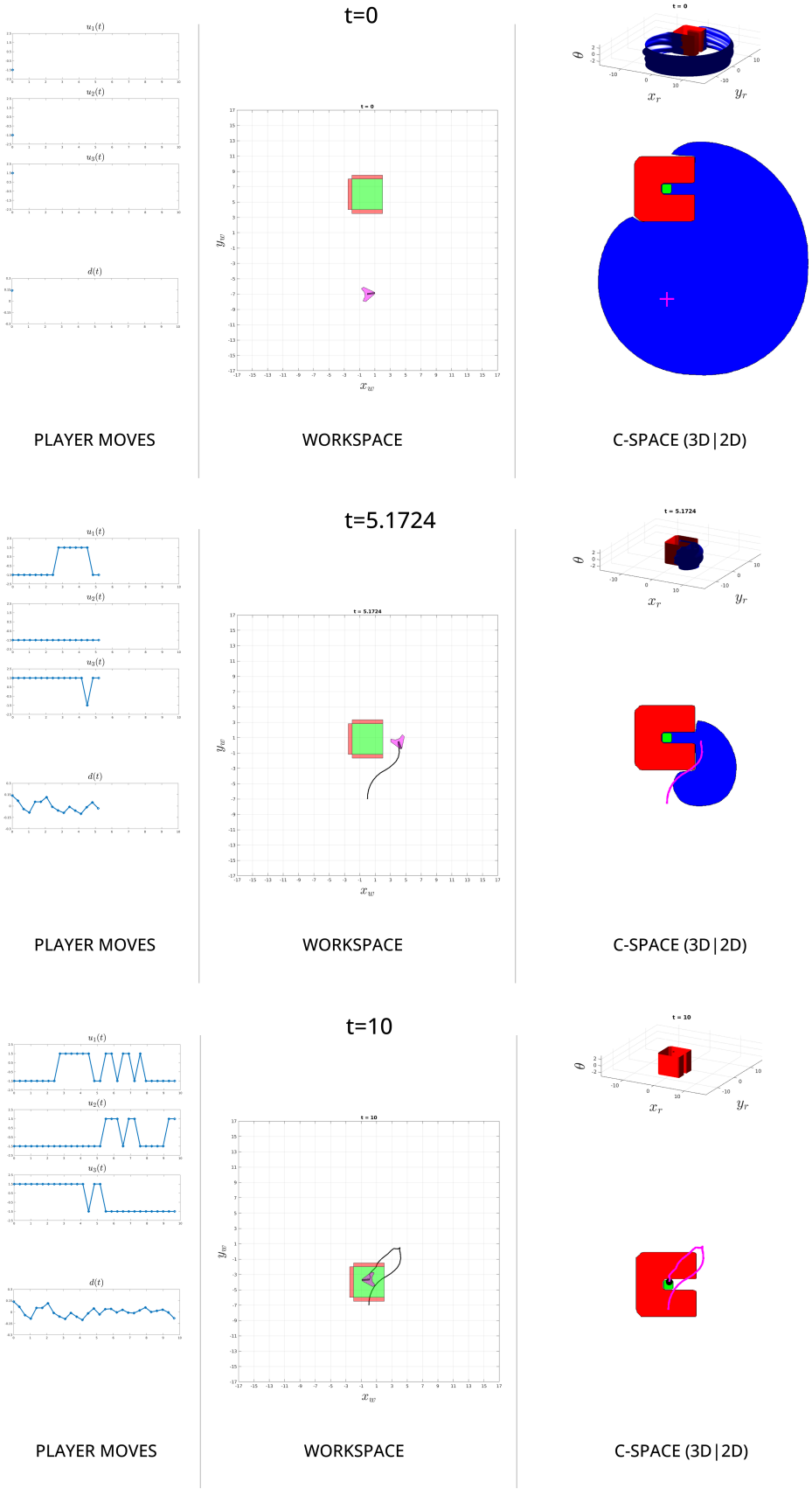

Fig. 12. Dynamic environment, random disturbance. $x_0 = [0, -7, -3]$, $\theta_d \in [-0.1, 0.1]$

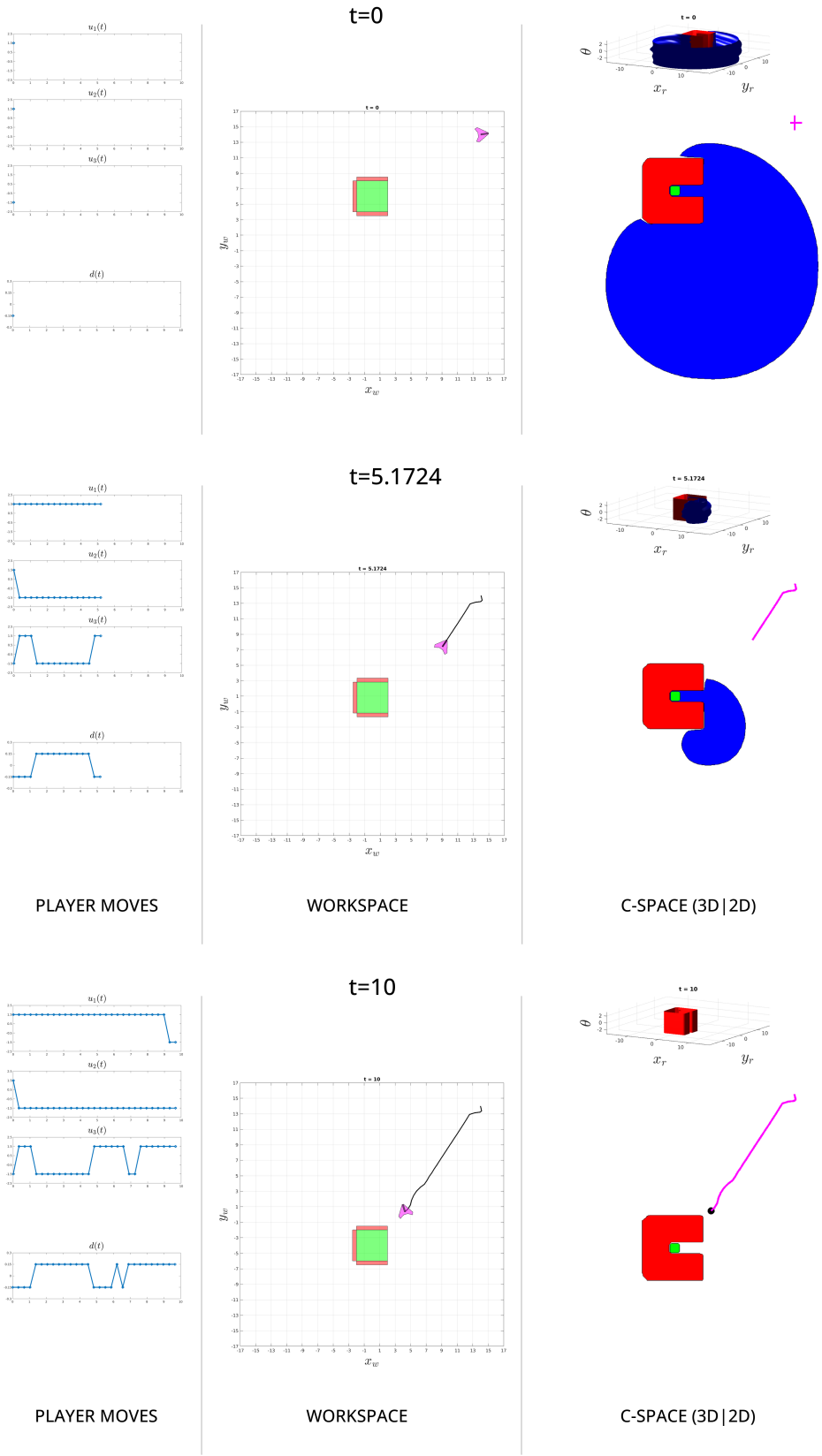

Fig. 13. Dynamic environment, optimal disturbance. $x_0 = [14, 14, -3]$, $\theta_d \in [-0.1, 0.1]$ (outside RAS)

APPENDIX

A. Kinematic Model Calculation

The three-wheeled mobile robot is used in other two papers [4] [12] however, they do not provide the procedure to compute its kinematic model, therefore for completeness we show here the steps necessary to obtain (25). The initial steps are similar to the proof shown in [9].

The type of onmi-wheel used by the robot is shown in Fig.14, please notice axes of rollers are perpendicular (90°) to the wheel axis, we made this clarification to avoid confusion with Mecanum wheel (a.k.a. Swedish wheel) in which axes of rollers have an angle of 45°, this last type is usually used in four-wheeled robot.

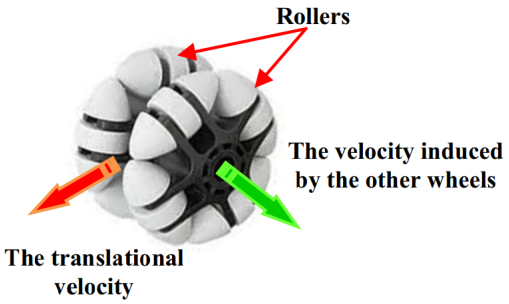

Fig. 14. Omnidirectional wheel mounted on three-wheeled robot. Credit: [9]

The first step is to compute the velocity induced to other wheels when a wheel rotates around its axis. Once computed it we can easily obtain the kinematic model w.r.t. to the robot's coordinate system RF_m and then w.r.t. RF_w.

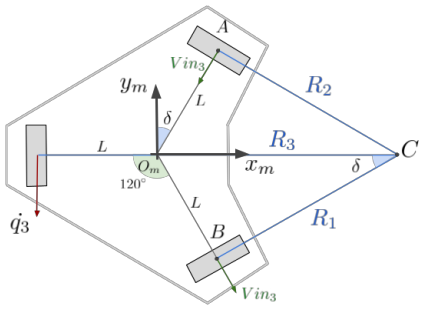

Fig. 15. Induced velocity by the third wheel to the other ones

1) Induced velocity: Assuming that only the third wheel rotates with a velocity $\dot{q}_3$ and the other two wheels are inactive, we define as induced velocity Vin_3 the velocity that the first and second wheel gain from the third wheel. Using as reference Fig.15, the robot turns around point C with an angular velocity ω that can be expressed in two equivalent ways:

$$\omega_3 = \frac{\dot{q}_3}{R_3} = \frac{V_{in3}}{R_1} \tag{30}$$

R_3 and R_1 can be easily computed from $\widehat{OBC}$ triangle:

$$R_3 = L + \overline{OC} = L + 2L = 3L$$

$$R_1 = \frac{sin(\pi/2 - \delta)}{sin(\delta)} L = \frac{cos(\delta)}{sin(\delta)} L$$

Then from (30) we can express Vin_3 in terms of $\dot{q}_3$:

$$Vin_3 = \frac{R_1}{R_3} \dot{q}_3$$

Due to the symmetrical mounting of the wheels, we can extend the previous result to a generic wheel i:

$$Vin_i = \frac{1}{3} \frac{cos(\delta)}{sin(\delta)} \dot{q}_i$$

2) Kinematic model in robot reference frame: In the general case we have all the three wheels rotating with a specific velocity $\dot{q}_i$ and inducing a velocity Vin_i to the others. We define a local frame $RF_i = [O_i, X_i, Y_i]$ for each wheel, O_i is placed at the wheel's center, Y_i is parallel to $\dot{q}_i$ and X_i is placed in order to make RF_i coherent with the direction of the z-axis of RF_m, see Fig. 16.

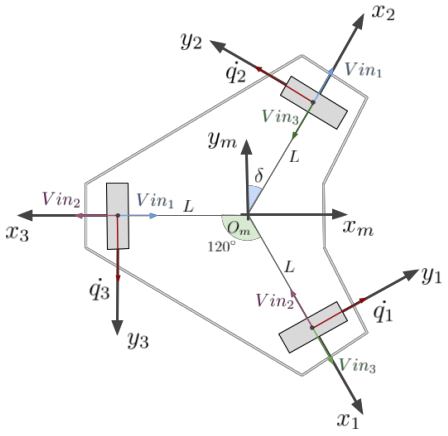

Fig. 16. Resulting velocities w.r.t. RF_i

The resulting velocities of each wheel i w.r.t to RF_i are the following:

$${}^1V_1 = \begin{pmatrix} Vin_3 - Vin_2 \\ \dot{q}_1 \end{pmatrix} \tag{31}$$

$${}^2V_2 = \begin{pmatrix} Vin_1 - Vin_3 \\ \dot{q}_2 \end{pmatrix} \tag{32}$$

$${}^3V_3 = \begin{pmatrix} Vin_2 - Vin_1 \\ \dot{q}_3 \end{pmatrix} \tag{33}$$

We can rewrite each of them w.r.t. RF_m using:

$$ {}^mV_1 = {}^mR_1\,{}^1V_1, \quad {}^mV_2 = {}^mR_2\,{}^2V_2, \quad {}^mV_3 = {}^mR_3\,{}^3V_3 $$

Where:

$$ {}^mR_1 = R_z\left(-\frac{\pi}{2}+\delta\right), {}^mR_2 = R_z\left(\frac{\pi}{2}-\delta\right), {}^mR_3 = R_z(\pi) $$

$$ R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} $$

The whole angular velocity of the robot can be obtained by summing each single contribution ω_i (30):

$$ \omega = \omega_1 + \omega_2 + \omega_3 $$

Therefore, the kinematic model w.r.t. the RF_m is given by:

$$ \begin{cases} \begin{pmatrix} {}^m\dot{x}_r \\ {}^m y_r \end{pmatrix} = \frac{{}^mV_1 + {}^mV_2 + {}^mV_3}{3} \\ \omega = \frac{1}{3L}\left(\dot{q_1} + \dot{q_2} + \dot{q_3}\right) \end{cases} \tag{34} $$

3) Model in the world coordinate system: Finally, starting from (34), we can compute the kinematic model w.r.t to the world reference frame RF_w. The angular velocity remains the same, for the position we use a rotation around z-axis:

$$ \begin{cases} {}^w\dot{x}_r = R_z(\theta)\,{}^m\dot{x}_r \\ {}^w\dot{y}_r = R_z(\theta)\,{}^m\dot{y}_r \\ \dot{\theta} = \omega \end{cases} \tag{35} $$

Carrying out the calculations with $\delta = 30°$, from (35) we obtain (25). [2]

[2]Matlab notebook *proof_model.mlx* in code folder shows the computation

REFERENCES

[1] S. Bansal, Mo Chen, S. Herbert and C. J. Tomlin, "Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances", *arxiv.org/abs/1709.07523*.

[2] K. Margellos and J. Lygeros, "Hamilton-Jacobi formulation for reach-avoid problems with an application to air traffic management", *Proceedings of the 2010 American Control Conference*, 2010, pp. 3045-3050, doi: 10.1109/ACC.2010.5530514.

[3] K. Margellos and J. Lygeros, "Hamilton–Jacobi Formulation for Reach–Avoid Differential Games," *in IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1849-1861, Aug. 2011, doi: 10.1109/TAC.2011.2105730.

[4] Datar, Mandar & Ketkar, Vishwanath & Mejari, Manas & Gupta, Ankit & Singh, N.M. & Kazi, Faruk, "Motion Planning for the Three Wheel Mobile Robot using the Reachable Set Computation under Constraints, (2014)" *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 3. 10.3182/20140313-3-IN-3024.00178.

[5] I. M. Mitchell, A. M. Bayen and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *in IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947-957, July 2005, doi: 10.1109/TAC.2005.851439.

[6] Lawrence C. Evans and Panagiotis E. Souganidis, "Differential Games and Representation Formulas for Solutions of Hamilton-Jacobi-Isaacs Equations", *Indiana University Mathematics Journal (1983), volume 33, pages 773-797*.

[7] Fisac, Jaime, Chen, Mo, Tomlin, Claire, Sastry, Shankar, "Reach-Avoid Problems with Time-Varying Dynamics, Targets and Constraints (2014)", *https://arxiv.org/abs/1410.6445*.

[8] Crandall, Michael G., and Pierre-Louis Lions, "Viscosity Solutions of Hamilton-Jacobi Equations", *Transactions of the American Mathematical Society, vol. 277, no. 1, American Mathematical Society, 1983, pp. 1–42*.

[9] Nacer Hacene and Boubekeur Mendil, "Motion Analysis and Control of Three-Wheeled Omnidirectional Mobile Robot", *Journal of Control, Automation and Electrical Systems, vol. 30, 2019, pp. 194-213*.

[10] Ian Mitchell, "Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems", *https://www.cs.ubc.ca/ mitchell/Papers/thesisMitchell.pdf*

[11] Ian Mitchell, "A toolbox of level set methods. 2004", *http://www.cs.ubc.ca/ mitchell/ToolboxLS*

[12] Xiang Li, Andreas Zell, "Motion Control of an Omnidirectional Mobile Robot", *https://www.scitepress.org/Papers/2007/16448/16448.pdf*