How to synchronize an out of sync system?

Kahina LOUADJ ^{1,2} and Philippe MARTHON ²

Abstract—Synchronization with other machines is one of those tasks that an intelligent machine should be able to perform. To do so, a general method of synchronization must be defined and that is the ambition of this article. For this purpose, we recall what the main concepts of systemic modelling consist of. Then we define what a Synchronization Problem is and distinguish three types of synchronization problems : predetermined (PSP), stochastic (SSP) and asymptotic (ASP). So, to find the best solution, we define three optimized synchronization problems : Optimized Predetermined Synchronization Problems, Optimized Stochastic Synchronization Problems (OSSP) and Optimized Asymptotic Synchronization Problems (OASP). A general method for solving an optimized synchronization problem and thus synchronize an out of sync system, is then given; it includes four steps : step 1, identify the synchronization functions; step 2, identify the synchronization case : predetermined, stochastic or asymptotic; step 3, select an optimization criterion and solve the optimized synchronization problem corresponding to the chosen synchronization case; step 4, find a control which permits to track the system trajectory, optimal solution of the selected optimized synchronization problem. In the next section, we present a heuristic algorithm that checks the constraints of an ASP for any initial state of the system to be synchronized. The principle of this algorithm is to pursue or track a perfectly synchronized solution. Finally, we present and solve two synchronization problems in the field of Mobile Robot Systems: a synchronized satelliteization problem and the Horse Carousel Problem. We apply to these two problems the general method given above by choosing the asymptotic synchronization case and our tracking algorithm. Simulation results show the efficiency of the method and the relevance of using a tracking algorithm.

Index Terms—Machine Intelligence, Mobile Robot System, Trajectory Tracking, Motion Synchronization, Synchronization Theory, Pursuit Algorithm, Horse Carousel Problem.

1 INTRODUCTION

TO fix the ideas, let's start by giving a typical example of a synchronization problem: the Horse Carousel Problem (HCP).

Definition 1.1 (The Horse Carousel Problem). *Let's suppose that a carousel contains n horses and that n children want to climb on it. Now, let's assume that the merry-go-round is spinning, that it won't stop, and that the intrepid children still want to ride it. What are the trajectories of the n children that would allow them to ride them?*

Let's start by noticing that the children *synchronized* on the carousel so they could ride their horses. Also note it is necessary that when the children climb on their horses, their velocity vectors are approximately the same as that of the horses. Finally, it seems natural to look for the trajectory that minimizes the sum of the lengths of the paths taken by the children to reach their horses.

In what follows, we will give a solution to this problem and illustrate it with a similar example, that of the synchronized satelliteization of three satellites.

In this paper, we are interested by the general system synchronization problem (SP). Synchronization means make two or more things happen at the same time or in a precise sequence. Synchronization is also the coordination of events to operate a system in unison. For example, the conductor of an orchestra keeps the orchestra synchronized. In the

case of a Multi-Robot System (MRS), synchronize it means synchronize the trajectories with each other. A crucial point is obviously to avoid collision between robots of the MRS.

Surprisingly, we did not find any articles in the literature dealing with synchronization problems. There does not seem to be any real theory on the subject.

Synchronization remains above all an art; however, some realizations such as the shows of ballets of drones given in the presentations of the Korean Olympic winter PyeongChang 2018 [1] and Japanese Olympic summer Tokyo 2020 [2] games, demonstrate a definite know-how.

One of the objectives of this study is show that *system modelling* is a fundamental prerequisite to formulate a synchronization problem. It also permits to propose solutions.

So, this paper is organized as follows : first, we give a definition of systemic modeling (SM). Secondly, we give the three main SM applications: simulation, optimization and identification. In section 4, we define what is the trajectory of a Mobile Robot System and the two conditions that it must meet and in section 5, we present synchronization theory which contains fundamental definitions, and define different synchronization problems and after, we give three types of optimized synchronization problems. Section 6 presents a general method to synchronize an out of sync system and in section 7 a pursuit algorithm for solving an asymptotic synchronization problem (ASP) is given. In section 8, we present and solve two synchronisation problems in the field of Mobile Robot Systems: a synchronized satelliteization and the Horse Carrousel Problem.

- ¹K.Louadj Bouira University. Faculty of Sciences and applied sciences. Department of Mathematics. Bouira. Algeria.
E-mail: louadj_kahina@yahoo.fr, Philippe.Marthon@enseeiht.fr
- ²P.Marthon Federal University of Toulouse Midi-Pyrénées, France. EN-SEIHT Site of the Toulouse Research Institute in Computer Science.

2 SYSTEMIC MODELLING

Systemic modelling consists in representing the phenomena in the form of systems, that is to say of sets of agents interacting on each other. So, systemic modelling is a description of a real phenomenon and the changes that are produced by a sequence of events resulting from actions and interactions. For this, there are two important factors that are as follows:

- ◇ **The agents** are substantial beings that are at the origin of these actions, centers of actions.
- ◇ **States** are associated to these agents and change under the effect of the produced actions.

2.1 Description of a system

The description of a system, we will formulate it in a mathematical language from three entities, the agents, the states and the actions.

Agents and States

As previously seen, the agents are the action centers of the system and their states are the properties and interactions they have at a given instant. These states will evolve over time under the effect of three factors:

- 1) the history of the agent, i.e. all the states through which it has passed since its creation.
- 2) the actions generated by the agent and those it is influenced by the other agents in the system.
- 3) the action that the environment of the system applies to it.

To summarize, we will have systems and their environments, inside these systems agents that remain fixed during their life periods and the states, associated to these agents, that are varying. In the same system, agents can appear or disappear during the life of this system.

Actions and controls

We must now discuss the actions and interactions that take place within the system. As we have indicated, at the origin of an action is the agent of the system which has generated this action. This action has for consequence to modify one or several states of the agents composing the system. In what follows, we will distinguish two types of variable and invariable action:

- 1) A variable action will be called a "control". The agent who generates it can modify it at will and thus modify the evolution of the system (its trajectory).
- 2) An invariable action will always be the same.

In other terms, the controls allow to act on the trajectory of the system, whereas invariant actions do not.

These actions depend on the present state of the agents that generated them. They can even depend on the complete history of these agents when they have a memory.

We thus end up with our general model of a system, which we will name AR model for action-reaction, and which we can describe in the form of an algorithm.

AR Algorithm

At the instant of creation of the system, all the agents of this system are in a given state. From this initial instant, actions will be generated which will modify the state of the system until its final death. Hence the algorithm:

- 1) Define the agents of the system and their states
- 2) Determine the initial state of the system
- 3) While the system is in life, do
 - a) Calculate the actions generated by all the agents of the system, these actions depending themselves on the current state of the system
 - b) Modify the current state of the system according to the generated actions and the history of the system
- 4) End While

3 SYSTEMIC MODELLING APPLICATIONS

3.1 Simulation

The simulation of a system will consist in calculating its trajectory during its lifetime. This will be done by running the AR algorithm above.

3.2 Optimization

As we have just proved, the evolution of a system depends on the actions that are exerted on it (within it and from its environment). However, it will be possible to modify these actions when these actions are controls. The question is then that of the choice of these controls, which supposes a criterion of choice. Once this criterion is established, the resulting controls will be qualified as optimal if they optimize this criterion. So, an optimized system is a system whose evolution will make it possible to best achieve the function for which it was designed.

An optimized system is called to last, a non-optimized system is called to disappear or to be reformed. Optimization is at the heart of Operations Research and Optimal Control.

3.3 Identification

It is a question here of determining the trajectory of the system when this one is indeterminate. This problem is at the heart of Artificial Intelligence.

4 MRS TRAJECTORY

Definition 4.1. The state $X(t)$ of a Mobile Robot System at an instant t , is the n -uplet $(X_1(t), X_2(t), \dots, X_n(t)) \in \mathbb{R}^N$ where $X_i(t) = (x_i(t); \dot{x}_i(t))$ is the state of the mobile robot i at t ; $x_i(t)$ and $\dot{x}_i(t)$ are the position and the speed of the robot i .

Definition 4.2. The Trajectory of a Mobile Robot System is the function which at any instant t makes the state $X(t)$ of the system at t . A trajectory can be identified to a path parameterized by time.

Definition 4.3. Trajectory tracking is concerned with the design of control laws that force a mobile robot to converge to a time parameterized reference [4].

4.1 General constraints on MRS trajectories

In all cases, a trajectory of a mobile robot must meet two conditions:

- 1) A Feasibility condition
- 2) A Non-collision condition

Remark that these two conditions are independent.

4.1.1 Feasibility condition

This condition means that there is a command that will allow the robot to track its trajectory (definition 4.3). It is therefore necessary to know precisely the physical characteristics of the robot to know if a trajectory is feasible (realizable). This feasibility condition will be translated by geometrical constraints on the path traveled by the robot (e.g. an increase in curvature) and by kinematic constraints (e.g. an increase in the intensity of the robot's speed).

4.1.2 Non-collision condition

This condition means that the robot, while following its trajectory, does not collision with obstacles, whether fixed or mobile. This condition is often difficult to meet because it implies that the robot is able to describe its environment at any time, which is itself changing.

5 SYNCHRONIZATION THEORY

5.1 Fundamental Definitions

Definition 5.1 (Synchronization). Synchronization means to make two or more things happen at the same time or in a precise sequence.

We rephrase this informal definition in systemic terms. Suppose a system S with n agents (robots for example) and note $X_i(t)$ the state of agent i at instant t . Note $X(t) = (X_1(t), X_2(t), \dots, X_n(t)) \in \mathbb{R}^N$ the state of the system at instant t .

For instance $N = mn$ if $X_i(t) \in \mathbb{R}^m$, $\forall i = \overline{1, n}$.

Definition 5.2 (Synchronization function and System Synchronization). Let g a function $\mathbb{R}^+ \times \mathbb{R}^N \mapsto \mathbb{R}^p$. g is a synchronization function of the system S if the following equivalence relation is verified:

$\forall t \in [t_0, t_1]$, $g(t, X(t)) = 0 \iff$ all the agents of the system happen at the same time or in a precise sequence on the time interval $[t_0, t_1]$.

The system is then perfectly synchronized on the interval $[t_0, t_1]$.

Note that a synchronization function is not unique in general. For example, for the Horse Carousel Problem (HCP), there is $n!$ synchronization functions corresponding to the number of permutations from $\{1, \dots, n\}$ in itself.

Definition 5.3 (Perfect Synchronized System). A Perfect Synchronized System on an interval $[t_0, t_1]$ is such that there exists a synchronization function g checking $\forall t \in [t_0, t_1]$, $g(t, X(t)) = 0$.

Definition 5.4 (Asymptotic Synchronized System). An Asymptotic Synchronized System is such that there exists a synchronization function g checking $g(t, X(t)) \rightarrow 0$, when $t \rightarrow +\infty$.

5.2 Synchronization Problems

Definition 5.5 (Out Of Sync). An out of sync system is a system which at start up ($t = 0$) verifies $g(0, X(0)) \neq 0$.

Definition 5.6 (Perfect and Asymptotic Synchronization Problem). A perfect (resp. asymptotic) synchronization problem of an out of sync system is to find a feasible and free-collision trajectory X of this system such as $\exists t_0 > 0 \forall t \geq t_0$, $g(t, X(t)) = 0$ (resp. $g(t, X(t)) \rightarrow 0$ when $t \rightarrow +\infty$) where g is a synchronization function.

If t_0 is predetermined, a perfect synchronization problem will be called a Predetermined Synchronization problem (PSS); if not, a Stochastic Synchronization Problem (SSP). An Asymptotic Synchronization Problem will be noted ASP.

5.3 Optimized Synchronization Problems (OSP)

In general, there are infinite solutions to synchronization problems. It is therefore necessary to determine one of them, "the best", and for this purpose we propose to introduce an optimization criterion f . We then end up with the three types of optimized synchronization problems corresponding to the three types of synchronization previously mentioned.

Definition 5.7 (Optimized Predetermined Synchronization Problem). An Optimized Predetermined Synchronization Problem (OPSP) is a constrained optimization problem where the variable X is the trajectory of a system and where the initial state of the system is fixed. The other constraints are :

- 1) X is feasible.
- 2) X is collision-free.
- 3) X verifies the conditions of a trajectory of a perfect synchronized system from a predetermined instant t_0 .

. In mathematical words, an OPSP can be written as follows:

$$(OPSP) \begin{cases} \text{Opt} f(X) \\ X \text{ feasible} \\ X \text{ collision-free} \\ g(t, X(t)) = 0, \forall t \geq t_0 \\ X(0) = X_0 \end{cases}$$

where f is the criterion to be optimized and g is the synchronization function.

Definition 5.8 (Optimized Stochastic Synchronization Problem). An Optimized Stochastic Synchronization Problem (OSSP) is a constrained optimization problem where the variable X is the trajectory of a system and where the initial state of the system is fixed. The other constraints are :

- 1) X is feasible.
- 2) X is collision-free.
- 3) X verifies the conditions of a trajectory of a perfect synchronized system from a stochastic instant t_0 .

In mathematical words, an OSSP can be written as follows:

$$(OSSP) \begin{cases} \text{Opt} f(X) \\ X \text{ feasible} \\ X \text{ collision-free} \\ \exists t_0 \mid g(t, X(t)) = 0, \forall t \geq t_0 \\ X(0) = X_0 \end{cases}$$

where f is the criterion to be optimized and g is the synchronization function.

Definition 5.9 (Asymptotic Synchronization Problem). *An Optimized Asymptotic Synchronization Problem (OASP) is a constrained optimization problem where the variable X is the trajectory of a system and where the initial state of the system is fixed. The other constraints are :*

- 1) X is feasible.
- 2) X is collision-free.
- 3) X verifies the conditions of a trajectory of an asymptotic synchronized system.

. In mathematical words, an OASP can be written as follows:

$$(OASP) \begin{cases} \text{Opt}f(X) \\ X \text{ feasible} \\ X \text{ collision-free} \\ g(t, X(t)) \rightarrow 0, \text{ when } t \rightarrow +\infty \\ X(0) = X_0 \end{cases}$$

where f is the criterion to be optimized and g is the synchronization function.

6 A GENERAL METHOD TO SYNCHRONIZE AN OUT OF SYNC SYSTEM

At the beginning, our system is out of sync at $t = 0$. The general method for solving an optimized synchronization problem breaks down into four steps:

- ◇ Step 1: Model the synchronization constraints, identify the trajectories of perfect synchronized systems from $t = 0$ and the synchronization functions g .
- ◇ Step 2: Identify the synchronization case (perfect or asymptotic).
 - Case 1: Perfect Synchronization at a predetermined instant t_0 . Find a trajectory X such that $g(t, X(t)) = 0, \forall t \geq t_0$.
 - Case 2: Perfect Synchronization at stochastic instant t_0 . Find a trajectory X , such that $\exists t_0 > 0, g(t, X(t)) = 0, \forall t \geq t_0$.
 - Case 3: Asymptotic Synchronization. Find a trajectory X such that $g(t, X(t)) \rightarrow 0$ when $t \rightarrow +\infty$.
- ◇ Step 3: Choose an optimization criterion f and solve the synchronization problem corresponding to the chosen synchronization case (see below).
- ◇ Step 4: Find a control U which permits to track the trajectory $\bar{X}$, optimal solution of the synchronization problem above. We refer the reader to the references [3], [4] for specific solutions.

7 A PURSUIT ALGORITHM FOR SOLVING AN ASP

We begin to present a heuristic algorithm capable of finding a solution which checks the constraints of an ASP¹ for any initial state X_0 .

1. A solution which checks the constraints of an optimization problem is called a *feasible solution*

Its principle is to get nearer to a perfect synchronized system by modifying the speed of each of the mobiles so that it is directed towards the new position of the corresponding perfectly synchronized mobile. In addition, we make sure that the intensity of the acceleration is constant (noted γ). Moreover, once the change of speed is made, we correct its intensity to be equal to the corresponding perfectly synchronized mobile.

Algorithm 1 The Pursuit Algorithm

input: Constants, Perfectly Synchronized System, Initial Positions, Initial Speeds

Constants : n, γ, t_{max}
Perfectly Synchronized System: $X^*(n, t_{max}), V^*(n, t_{max})$
Initial positions : $X(n, 0)$
Initial speeds : $V(n, 0)$

```

for  $t \leftarrow 0$  to  $t_{max} - 1$  do
  for  $i \leftarrow 1$  to  $n$  do
    /* Calculate the attractive
       acceleration exerted on the mobile  $i$ 
       by the Perfectly Synchronized System
       */
     $\Gamma(i, t) \leftarrow \gamma * \frac{X^*(i, t+1) - X(i, t)}{\|X^*(i, t+1) - X(i, t)\|_2}$ 
    /* Calculate the new speed of the
       mobile  $i$ 
       */
     $V(i, t+1) \leftarrow \|V^*(i, t+1)\|_2 \frac{V(i, t) + \Gamma(i, t)}{\|V(i, t) + \Gamma(i, t)\|_2}$ 
    /* Calculate the new position of the
       mobile  $i$ 
       */
     $X(i, t+1) \leftarrow X(i, t) + V(i, t+1)$ 
  end
end

```

In this algorithm, $X(i, t)$ is the position of the mobile i at t and $V(i, t)$ its speed.

8 TWO EXAMPLES OF SYNCHRONIZATION PROBLEMS IN THE FIELD OF MRS

We present and solve two synchronization problems in the field of Mobile Robot Systems (MRS) : a synchronized satelliteization problem and the Horse Carousel Problem.

8.1 A Synchronized Satellite Problem (SSP)

Suppose that the synchronization problem is to try to put n satellites (the agents of the system) on the same circular orbit (center $(0, 0)$ and radius R), these satellites rotating in the trigonometric direction (Constraint C_1). When these satellites will be into the orbit, they must verify the two constraints of synchronization:

- ◇ Constraint C_2 : They move on the orbit with a constant velocity Rv .
- ◇ Constraint C_3 : They remain equidistant when they move on the orbit.

We want also that before reaching their orbit, they move to a speed near to Rv ; this constraint allows to avoid a discontinuity of the velocity intensity during the passage

in orbit (Constraint C_4).

We apply now the general methodology given in section 6.

- ◇ Step 1: It is easily shown that the trajectories X_i^* of n synchronized virtual satellites (verifying the constraints C_2 and C_3) must check:

$$X_i^*(1)(t) = x_i^*(t) = (R \cos(v(t - \frac{2(i-1)\pi}{nv}) + \phi), R \sin(v(t - \frac{2(i-1)\pi}{nv}) + \phi)), \forall t \geq t_0, i = 1, \dots, n$$

where Rv is the constant velocity and $\phi \in [0, 2\pi[$ is the phase.

So the synchronization function is $g = (g_1, g_2, \dots, g_n)$ with where

- P is a permutation of $\{1, \dots, n\}$ in itself.
- $X_i(t) = (X_i(1)(t); X_i(2)(t)) = (x_i(t); \dot{x}_i(t))$.

- ◇ Step 2: We choose to apply an asymptotic synchronization of the satellites.

- ◇ Step 3: As a criterion of optimization, we choose to minimize the maximum curvature of the system trajectory which is the largest curvature of the trajectories of n satellites.

Consequently, the OASP is written as follows:

$$(OASP) \begin{cases} \min_{\substack{\phi \in [0, 2\pi/n] \\ P \in \mathcal{P}}} K(\phi, P) \\ X_{feasible} \\ X_{collision-free} \\ g(t, X(t)) \rightarrow 0, \text{ when } t \rightarrow +\infty \\ X(0) = X_0 \end{cases}$$

where

- $K(\phi, P) = \max_{i=1, \dots, n} \max_{t \geq 0} \kappa(i, t)$.
- $\mathcal{P}$ is the set of permutations $\{1, 2, \dots, n\}$ into itself.
- $\kappa(i, t)$ is the curvature at time t of the trajectory of the satellite $P(i)$ which pursues the corresponding virtual synchronized satellite i , ($i = \overline{1, n}$).
- g is given by the equation (1).

To solve this OASP, firstly, we release the two constraints X feasible and X collision-free and solve the released problem by applying the pursuit algorithm (1) to each perfectly synchronized system defined by the pair (ϕ, P) and compute the value $K(\phi, P)$ for this pair. Finally, we compute the minimum. Note that we assume here that the tracking algorithm provides an optimal trajectory for each pair (ϕ, P) .

In the following, we present the results of a numerical application with $n = 3$.

The initial conditions are $X_1(0) = ((0, -20); (0, 0))$, $X_2(0) = ((20, 30); (0, 0))$, $X_3(0) = ((-20, -20); (0, 0))$. The radius of the circle $R = 10$, $v = 0.1$ so that the intensity of the speed Rv of each satellite on the orbital circle is equal to 1.

There are $3! = 6$ possibilities to associate each trajectory X_i

to its goal (the corresponding virtual synchronized satellite trajectory) X_j^* .

Since $g(t, X^*(t)) = 0, \forall t \geq 0$

$$X_1^*(1)(t) = (R \cos(vt + \phi), R \sin(vt + \phi)) \quad (2)$$

$$X_2^*(1)(t) = (R \cos(v(t - \frac{2\pi}{3v}) + \phi), R \sin(v(t - \frac{2\pi}{3v}) + \phi)) \quad (3)$$

$$X_3^*(1)(t) = (R \cos(v(t - \frac{4\pi}{3v}) + \phi), R \sin(v(t - \frac{4\pi}{3v}) + \phi)) \quad (4)$$

according to equation (1).

The permutation of $\{1, 2, 3\}$ in itself is noticed $P = (i, j, k)$ and means that X_i is associated to X_1^* , X_j to X_2^* and X_k to X_3^* .

The figures (1) – (6) show the paths associated to these 6 permutations when we apply the pursuit algorithm. The blue color is that of the path of the satellite which pursues the virtual satellite 1, the red color is that of the path of the satellite which pursues the virtual satellite 2 and the yellow color is that of the path of the satellite that pursues the virtual satellite 3 and this for a phase $\phi = 0$. The 6 figures represent permutations between the three virtual and real satellites ($3! = 6$). The diamonds identify the positions of the three real satellites after a simulation time $t_{max} = 100$ with $\gamma = 1$. Note that at this date the three real satellites are in orbit, are located on the vertices of an equilateral triangle and therefore, quasi-perfectly synchronized (cf figures (14) and (15)).

For $n = 3$, the problem OASP is written as follows:

$$(OASP) \begin{cases} \min_{\substack{\phi \in [0, 2\pi/3] \\ P \in \mathcal{P}}} K(\phi, P) \\ X_{feasible} \\ X_{collision-free} \\ g(t, X(t)) \rightarrow 0, \text{ when } t \rightarrow +\infty \\ X(0) = X_0 \end{cases}$$

The previous study leads us to search for an optimal solution of our OASP problem among the solutions that pursue the corresponding perfectly synchronized system defined by (2), (3) and (4).

To solve OASP, firstly, we release the two constraints X feasible and X collision-free and solve the released problem ($OASP^*$)

$$(OASP^*) \begin{cases} \min_{\substack{\phi \in [0, 2\pi/3] \\ P \in \mathcal{P}}} K(\phi, P) \\ g(t, X(t)) \rightarrow 0, \text{ when } t \rightarrow +\infty \\ X(0) = X_0 \end{cases}$$

by applying the pursuit algorithm (1) to the perfectly synchronized system defined by the pair (ϕ, P) .

By calculating the maxima of the (non-oriented) curvatures for different values of ϕ varying from 0 to $\frac{2\pi}{3}$ with a step of $\Delta\phi = \frac{2\pi}{303} = 0.0207rd (=1.881deg)$ and for different permutations of the three satellites and taking the minimum, we get the optimal phase $\bar{\phi} = 0.41473rd$ and the optimal permutation. $\bar{P} = (2, 1, 3)$.

$$\begin{aligned}
g_i(t, X(t)) &= X_{P(i)}(t) - X_i^*(t) \\
&= \begin{bmatrix} X_{P(i)}(1)(t) - X_i^*(1)(t) \\ X_{P(i)}(2)(t) - X_i^*(2)(t) \end{bmatrix} \\
&= \begin{bmatrix} X_{P(i)}(1)(t) - (R \cos(v(t - \frac{2(i-1)\pi}{nv}) + \phi), R \sin(v(t - \frac{2(i-1)\pi}{nv}) + \phi)) \\ X_{P(i)}(2)(t) - (R \sin(v(t - \frac{2(i-1)\pi}{nv}) + \phi), -R \cos(v(t - \frac{2(i-1)\pi}{nv}) + \phi)) \end{bmatrix}
\end{aligned} \tag{1}$$

Fig. 1: (1,2,3)-Permutation

Fig. 3: (2,1,3)-Permutation

Fig. 2: (1,3,2)-Permutation

Fig. 4: (2,3,1)-Permutation

In other words,
the real satellite 1 pursues the virtual satellite 2 whose trajectory is defined by:

$$X_2^*(t) = (R \cos(v(t - \frac{2\pi}{3v}) + 0.41473), R \sin(v(t - \frac{2\pi}{3v}) + 0.41473))$$

the real satellite 2 pursues the virtual satellite 1 whose trajectory is defined by:

$$X_1^*(t) = (R \cos(vt + 0.41473), R \sin(vt + 0.41473))$$

the real satellite 3 pursues the virtual satellite 3 whose trajectory is defined by:

$$X_3^*(t) = (R \cos(v(t - \frac{4\pi}{3v}) + 0.41473), R \sin(v(t - \frac{4\pi}{3v}) + 0.41473))$$

The optimal curvature is $K(\bar{\phi}, \bar{P}) = 0.16361 \text{ pix}^{-1}$. It means that the curvatures of the three trajectories $\bar{X}_i(t)$ ($i = 1, 2, 3$) are always less or equal to 0.16361. It must be noticed that this value is near the theoretic minimum 0.1 (the curvature of the orbital circle). The optimal paths are given by figure (7). The simulation time $t_{max} = 100$.

Figures (8) and (9) represent the paths with respect to x and respect y and the figures (10) and (11) represent speeds with respect to x and to y correspond to $\phi = 0.41473$.

Secondly, we verify if the optimal system trajectory $\bar{X}$ satisfy the two constraints :

◇ $\bar{X}$ feasible:

To answer this question, we must know the mechanical characteristics of the system and apply to it the physical laws.

Fig. 5: (3,1,2)-Permutation

Fig. 6: (3,2,1)-Permutation

Fig. 7: The Optimal Paths

◇ $\bar{X}$ collision-free: The simulation allows to know if there are collisions between the satellites when they travel their respective trajectories. Of course, it cannot predict collisions with other unknown fixed or moving obstacles.

Fig. 8: The Trajectories of x Fig. 9: The Trajectories of y Fig. 10: The Speeds of x Fig. 11: The Speeds of y

8.2 The Horse Carousel Problem (HCP)

The only difference with the previous example is that the goals (the horses) to reach are real and not virtual (the satellites). The phase ϕ is determined here by the angle that the horse 1 makes at the instant $t = 0$. So there are only $n!$ synchronization functions instead of an infinity.

Fig. 12: The Optimal Curvatures

Fig. 13: The Gradients of Optimal Curvatures

9 CONCLUSION

In this paper we have presented a general method to synchronize an out of sync system. Then, we have presented a pursuit algorithm capable of finding a solution which checks the constraints of an Asymptotic Synchronization Problem. Applied to a problem of synchronizing three satellites on the same orbit, it works very well.

Fig. 14: Distances to the center of the orbit

Fig. 15: The salient angles in degrees between two satellites versus time (the angle between 1 and 2 is colored blue, the one between 1 and 3 is colored red and the one between 2 and 3 is colored yellow).

REFERENCES

- [1] <https://www.youtube.com/watch?v=Wm00ASff6u0>.
- [2] <https://www.youtube.com/watch?v=dukMp9xorWo>.
- [3] R. Maslim, H. Chaoyi, Z. Yixi, J. Linhao, B. B. Kocer, and E. Kayacan, *Performance evaluation of adaptive and non adaptive fuzzy structures for 4d trajectory tracking of quadrotors : A comparative study*, 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–7, août 2015.
<http://doi.org/10.1109/FUZZ-IEEE.2015.7337945>
- [4] Z. Zeng, H. Lu, Z. Zheng, *High-Speed Trajectory Tracking Based on Model Predictive Control for Omni-directional Mobile Robots*, 25th Chinese Control and Decision Conference (CCDC), 2013.
<http://doi.org/10.1109/CCDC.2013.6561493>