

POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in Automazione

Dipartimento di Elettronica, Informazione e Bioingegneria



Thesis

Title

Advisor: Prof. Name SURNAME

Co-Advisor: Ing. Name SURNAME

Thesis by:

Name SURNAME Matr. 000000

Academic Year 2016–2017

To someone very special...

Acknowledgments

Abstract

Sommario

Contents

Acknowledgments	5
Abstract	7
Sommario	9
List of figures	13
List of tables	15
Introduction	17
1 State of the art	19
1.1 Parametrized trajectory	20
1.2 Virtual time	22
1.3 Consensus law	23
1.4 Network topology	25
1.5 Convergence property	26
2 Consensus	29
3 System architecture	31
4 Consensus node	33

5	Simulation results	35
6	Experimental results	37
	Conclusions	39
A	First appendix	43
B	Second appendix	45
C	Third appendix	47

List of Figures

List of Tables

Introduction

Chapter 1

State of the art

The literature about the "Consensus" has grown significantly in the latest years because of the increasing presence of autonomous vehicles. This, in accordance with the new improvements in robotics, has brought to growing interest in consensus between multiple agents which have to accomplish a mission in cooperative or adversarial scenarios.

The "Consensus" theory has its roots in Graph Theory and Automatics and it can be used to coordinate a mission in order to achieve the synchronization between the vehicles even during unforeseen events which force one or more components of the mission to change the planned trajectory or task. In this case, the other components identify this variation and they act to preserve the synchronization.

The "Consensus" algorithm is a distributed algorithm which can sometimes be simulated in a centralized fashion because of the reduced computational power of the machines involved in the mission, which are equipped with low power hardware to account for the crucial issue of power consumption. In this scenario, a centralized server simulates the algorithm and communicates with all the machines.

The most common "Consensus" application is a spatial and timing consen-

sus: in this implementation, each vehicle of the formation has to travel along a specified trajectory and the completion of the mission occurs when all the agents reach the final positions of their spatial paths. The algorithm has to guarantee that the difference between the ending time at which all the vehicles finish their tasks is minimized and asymptotically goes to zero, when the execution time goes to infinity and no other unforeseen events happens. We also consider formations of Unmanned Aerial Vehicles, but the key concepts can be freely applied to other categories of robots when they are able to follow a trajectory. This study is focused on this kind of application, and further simulation results and experimental achievements are presented in chapters 5 and 6.

In the next sections we refer to the main work of Venanzio Cichella [3] in the field of UAV consensus, in order to provide an homogeneous state of the art about the "Consensus". The paper considers all the details needed to build a "Consensus" system. The main components of this kind of system are listed and explained below:

- Parametrized trajectory
- Virtual time
- Consensus law
- Network topology

1.1 Parametrized trajectory

The trajectory is a spatial path with an associated timing law and it is used to identify the position of the center of mass of our agent at a given time. We can start with the following definition of a generic trajectory $p_{d,i}(t_d)$ for N vehicles:

$$p_{d,i} : [0, t_{d,i}^f] \rightarrow \mathbb{R}^3, \quad i = 1, 2, \dots, N \quad (1.1)$$

where $t_d \in [0, T_d]$, with $T_d := \max\{t_{d,1}^f, \dots, t_{d,N}^f\}$, is the time variable of the trajectory, while $t_{d,i}^f \in \mathbb{R}^+$ are the individual final mission times of the vehicles obtained during the planning phase. Usually all this final times are equal and therefore $t_{d,1}^f = \dots = t_{d,N}^f = T_d$, but we introduced the notation for the sake of completeness. Obviously, the trajectories need to be collision free and must comply with spatial and temporal constraints due to the dimension of the vehicle and its maximum velocities and accelerations. The trajectory can account also for the orientation, and therefore, the function might take images in \mathbb{R}^m , where m is the number of dimensions considered. In the following sections, we will refer only to the position, but a more general theory can also be developed.

We now parametrize the trajectory using a dimensionless variable $\zeta_i \in [0, 1]$, related to the time t_d . In this way we can specify a function $\theta(\cdot)$, which represents the timing law associated with the spatial path $p_{d,i}(\zeta_i)$. We can specify this timing law using the dynamic relation of the form:

$$\theta(t_d) = \frac{d\zeta_i}{dt_d} \quad (1.2)$$

where $\theta(t_d)$ is a smooth and positive (the parameter increases when the time increases) function.

As it allows a one-to-one correspondence between the time variable t_d and the parameter ζ_i , an analytical expression for the function $\zeta_i(t_d)$ is desirable. Using the timing law defined in (1.2), the map $\zeta_i(t_d)$ is given by the integral:

$$\zeta(t_d) = \int_0^{t_d} \theta_i(\tau) d\tau \quad (1.3)$$

Usually, all this functions are defined as polynomials in order to make quicker and easier the evaluation process, since multiplication and addition are the basic operations in a digital processor. However, it is not mandatory to use them and a generic shape for the functions can be designed. We have defined all the elements of a trajectory and we go into detail about the trajectory generation

phase. Further information about boundary conditions and flyable trajectory, which satisfy the dynamic constraints of the vehicles, can be found in [3] and is extensively analyzed in [2], [4], [5].

1.2 Virtual time

Given N collision free trajectories, we want each vehicle to follow a *virtual target*, moving along the path computed offline by the trajectory generation algorithm. The objective can be achieved introducing a *virtual time*, γ_i , which is used to evaluate the trajectory and can be adjusted online to reach the synchronization even when external disturbances occur. Thus, the position of the i^{th} virtual target is denoted by $p_{d,i}(\gamma_i(t))$ and the i^{th} vehicle tries to follow it, by reducing to zero a suitably defined error vector using control inputs.

Considering the trajectory $p_{d,i}(t_d)$ produced by the trajectory generation algorithm, we consider the virtual time γ_i as a function of time t , which relates the actual time t to mission planning time t_d .

$$\gamma_i : \mathbb{R}^+ \rightarrow [0, T_d], \quad \text{for all } i = 1, 2, \dots, N \quad (1.4)$$

We can now define the virtual target's position, velocity and acceleration, which have to be followed by the i^{th} vehicle at time t

$$\begin{aligned} p_{c,i}(t) &= p_{d,i}(\gamma_i(t)) \\ v_{c,i}(t) &= \dot{p}_{d,i}(\gamma_i(t), \dot{\gamma}_i(t)) \\ a_{c,i}(t) &= \ddot{p}_{d,i}(\gamma_i(t), \dot{\gamma}_i(t), \ddot{\gamma}_i(t)) \end{aligned} \quad (1.5)$$

With the above formulation, if $\dot{\gamma} = 1$, then the speed profile of the virtual target is equal to the desired speed profile computed at trajectory generation level. Indeed, if $\dot{\gamma} = 1$, for all $t \in [0, T_d]$, with $\gamma_i(0) = 0$, it implies that $\gamma_i(t) = t_d$ for all t and thus:

$$p_{c,i}(t) = p_{d,i}(\gamma_i(t)) = p_{d,i}(t) = p_{d,i}(t_d)$$

In this particular case, the desired and commanded trajectories coincide in every time instant and also the velocity profiles coincide with the ones chosen at the trajectory generation time. If instead $\dot{\gamma}_i > 1$, it implies a faster execution of the mission; on the other hand, $\dot{\gamma}_i < 1$ implies a slower one.

We can now normalize γ_i in order to have a range which is $[0, 1]$. We simply need to divide all by T_d . In this way, we could use Bezier curves in order to represent the spatial path. This kind of curves offer interesting properties for computing minimum distances between two of them and allow the computation of smooth trajectories. In any case, it is not mandatory to use them and a general function could be used instead.

The second order derivative of γ_i , $\ddot{\gamma}_i$, is a free parameter used to achieve the consensus. In the next section we will introduce the control law which commands its evolution during time and we will explain how it is possible to implement a distributed algorithm.

1.3 Consensus law

Now, we formally state the path following problem. We define as $p_i(t) \in \mathbb{R}^3$ the position of the center of mass of the i^{th} agent and since $p_{c,i}(t)$ describes the commanded position to be followed by the agent at time t , the errors are defined as:

$$\begin{aligned} e_{p,i}(t) &= p_{c,i}(t) - p_i(t) \in \mathbb{R}^3 \\ e_{v,i}(t) &= v_{c,i}(t) - \dot{p}_i(t) \in \mathbb{R}^3 \end{aligned} \tag{1.6}$$

Then, the objective reduces to that of regulating the error defined in (1.6) to a neighbourhood of zero. This task is solved with an autopilot capable of following the set points computed from the desired trajectory at specified instances of time.

The virtual time is the parameter used to reach consensus between multiple vehicles. In fact, since the trajectories are parametrized by γ_i , the agents are synchronized at time t when:

$$\gamma_i(t) - \gamma_j(t) = 0 \quad \text{for all } i, j \in 1, \dots, N, \quad i \neq j \quad (1.7)$$

We can also control the rate of progression of the mission using a parameter $\dot{\gamma}_d \in \mathbb{R}$, which represents the velocity of the virtual time with respect to the real time. All the agents share this variable and they proceed at the same rate of progression if:

$$\dot{\gamma}_i(t) - \dot{\gamma}_d(t) = 0 \quad \text{for all } i \in 1, \dots, N \quad (1.8)$$

Adjusting $\dot{\gamma}_d$ we can decide the speed of the mission: for instance, if we set $\dot{\gamma}_d = 1$ and (1.7) and (1.8) are satisfied for all the vehicles, then the mission is executed at the speed originally planned in the trajectory generation phase. If instead we use $\dot{\gamma}_d > 1$ or $\dot{\gamma}_d < 1$ we carry out the mission faster or slower. This term can be changed in real time in order to avoid moving objects or unplanned obstacles, which make it necessary to change one of the path of the agents. For the purpose of "Consensus", the parameter is only a reference command, rather than a control input.

Now we introduce the coordination control law which regulates the evolution of $\ddot{\gamma}_i(t)$ during the time and determines $\gamma_i(t)$:

$$\begin{aligned} \ddot{\gamma}_i(t) &= \ddot{\gamma}_d(t) - b(\dot{\gamma}_i(t) - \dot{\gamma}_d(t)) - a \sum_{j \in \mathbb{N}_i} (\gamma_i(t) - \gamma_j(t)) - \bar{\alpha}_i(e_{p,i}(t)) \\ \dot{\gamma}_i(0) &= \dot{\gamma}_d(0) = 1 \\ \gamma_i(0) &= \gamma_d(0) = 0 \end{aligned} \quad (1.9)$$

where a and b are positive coordination control gains, while $\bar{\alpha}_i(e_{p,i}(t))$ is defined

as:

$$\bar{\alpha}_i(e_{p,i}(t)) = \frac{v_{c,i}^T(t)e_{p,i}(t)}{\|v_{c,i}(t)\| + \epsilon} \quad (1.10)$$

with ϵ being a positive design parameter, $e_{p,i}$ the position error vector defined in (1.6) and \aleph_i the set of the neighbors which can communicate with the i^{th} vehicle (we will see details later). In the equation (1.9) we have four terms. The feedforward term $\ddot{\gamma}_d$ allows the virtual target to follow the acceleration profile of γ_d . The second term $-b(\dot{\gamma}_i(t) - \dot{\gamma}_d(t))$ reduces the error between the speed profile imposed by $\dot{\gamma}_d(t)$ and $\dot{\gamma}_i(t)$, which corresponds to the control objective given in (1.8). In particular, if $\dot{\gamma}_d(t)$ is one, then the virtual target converges to the desired speed profile chosen in the trajectory generation phase. The third term $-a \sum_{j \in \aleph_i} (\gamma_i(t) - \gamma_j(t))$ ensures that all the vehicles are coordinated with their neighbors as specified in (1.7). Finally, the fourth term $-\bar{\alpha}_i(e_{p,i}(t))$ is a correction term used to take into account for the path following errors of the agent. Indeed, if the vehicle is behind its target, the term is not zero and the target slows down in order to wait for the real vehicle.

With this control law, we want our vehicles to be synchronized and to proceed at a desired rate of progression, in order to accomplish the mission even when some unforeseen disturbances occur during the execution phase.

1.4 Network topology

To achieve time-coordination objective, agents must exchange information over a supporting communication network. To analyze the information flow, we need to consider some tools from algebraic graph theory, whose key concepts can be found in [1].

We assume that a vehicle i exchanges information with only a subset of all vehicles, denoted as $\aleph_i(t)$. We assume that arcs of the network are bidirectional

and that there are no network delays. The information exchanged is composed by the virtual time of the agents, $\gamma_i(t)$.

The topology of the graph $\Gamma(t)$ that represents the communication network must comply with the following constraint in order to guarantee the convergence of the "Consensus" algorithm:

$$\frac{1}{NT} \int_t^{t+T} QL(\tau)Q^T d\tau \geq \mu I_{N-1}, \quad \text{for all } t \geq 0 \quad (1.11)$$

where $L(t) \in \mathbb{R}^{N \times N}$ is the Laplacian of the graph $\Gamma(t)$ and $Q \in \mathbb{R}^{(N-1) \times N}$ is a matrix such that $Q1_N = 0$ and $QQ^T = I_{N-1}$, with 1_N being a vector in \mathbb{R}^N whose components are all 1. In (1.11), the parameters $T > 0$ and $\mu \in (0, 1]$ represent a measure of the level of connectivity of the communication graph. This condition requires the graph $\Gamma(t)$ to be connected only in an integral sense, not pointwise in time. Therefore, even if the graph were disconnected during the mission at some interval of time, the convergence of the "Consensus" algorithm would still be possible. With this condition, we can capture also packets dropouts, loss of communication and switching topologies, which can all occur during the mission, but these events do not necessary break the convergence property.

1.5 Convergence property

The control law given by (1.9) guarantees that the error of the "Consensus" algorithm converges to zero exponentially. It can be shown that the maximum convergence rate is given by the sum of the convergence rate of the path following error and the term

$$\frac{a}{b} \frac{N\mu}{T(1 + (a/b)NT)^2} \quad (1.12)$$

which depends on the control gains a and b , the number of vehicles N and the quality of service of the communication network, characterized by the parameters

T and μ . If we fix the gains and the number of the vehicles, the convergence rate depends only on the amount of information which the agents exchange each other over time.

Chapter 2

Consensus

Chapter 3

System architecture

Chapter 4

Consensus node

Chapter 5

Simulation results

Chapter 6

Experimental results

Conclusions

Bibliography

- [1] N. Biggs. *Algebraic Graph Theory*. New York: Cambridge Univ. Press, 1993.
- [2] R. Choe, J. Puig-Navarro, V. Cichella, E. Xargay, and N. Hovakimyan. Trajectory generation using spatial Pythagorean Hodograph Bezier curves. *in Proc. AIAA Guidance, Navigation and Control Conf., Kissimmee, FL*, 2015.
- [3] V. Cichella, R. Choe, S. B. Mehdi, E. Xargay, N. Hovakimyan, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, and A. P. Aguiar. Safe coordinated maneuvering of teams of multirotor unmanned aerial vehicles. *IEEE Control systems magazine*, August 2016.
- [4] V. T. Taranenko. Experience of Employment of Ritz’s, Poincare’s, and Lyapunov’s Methods for Solving Flight Dynamics Problems. *Moscow: Air Force Engineering Academy Press*, 1968.
- [5] O. A. Yakimenko. Direct method for rapid prototyping of near-optimal aircraft trajectories. *J. Guidance, Control Dyn.*, 23(5):865–875, Sept.-Oct. 2000.

Appendix A

First appendix

Appendix B

Second appendix

Appendix C

Third appendix

