Multi-UAV Mission Coordination using Signal Temporal Logic Specifications

Nuno Ricardo Ferreira Duarte
nuno.ferreria.duarte@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

September 14, 2016

Abstract

Temporal Logic has successfully been used for specifying a wide range of behaviors for systems and environments. Previous works using Linear Temporal Logic (LTL) have synthesized controllers when specifying properties of discrete time signals. However, these controllers are inconvenient to handle continuous signals with time dependency, more specifically, when optimizing trajectories under environmental uncertainties, such as static or dynamical obstacles, known and unknown obstacles. To address these issues, we use Signal Temporal Logic (STL) instead and our aim is to design a reactive strategy and trajectory planning for a search and rescue mission in a known environment; the mission involves more than one autonomous aerial vehicle (UAV) working together to satisfy tasks such as periodically monitoring areas, path planning, staying safe, and avoiding obstacles that can either be object in the environment or a different autonomous vehicle that is attempting to satisfy its own tasks. To avoid obstacles we will compare more that one solution using different methods such as STL specifications that take into account the localization of all vehicles in the environment or a centralized node that checks the localization of two vehicles in order to prevent any possible future collisions. The simulations show that using a controller developed using this approach can generate trajectories for a mission involving 3 UAVs performing a cooperative mission of grabbing 3 individual objects and 1 object that requires 2 UAVs for carrying with successful results and without any collisions.

Keywords: Logic, Planning, UAV, STL.

1. Introduction

The focus of this thesis is on multi-UAV mission planning using signal temporal logic (STL) specifications. Mission coordination has been studied by many researchers and in various problem settings [2], [12], [11]. It is clear advantageous that it is to develop a coordinated team of unmanned vehicles to achieve a goal instead of using a single vehicle. Additionally, missions involving surveillance, rescue and reaching goals have been increasing in complexity. In this context, it is important to develop new aways of describing the missions specifications. STL is a formal language used to define constraints and specifications of continuous signals. The use of this language instead of other well-known languages ([7], [5] and [6]), is due to its application in continuous and hybrid systems. Combining a formal language to specify the tasks of multiple autonomous vehicles will provide tremendous insight to the performance of such specifications in a real-life applications. Mission coordination has been discussed for several years, either for humanitarian missions or with a more military view [2], [12], [11]. Nonetheless, using aerial vehicles in a group instead of a single quadrotor is more advantageous in terms of allowing solutions for more complex tasks and also being economically reliable than using a single UAV with larger proportions and higher quality sensors and actuators. Mission coordination involving multi-UAV has been a topic of tremendous research using several different perspectives, and describing formal specifications with mixed-integer linear constraints of discrete or continuous signals has been one of the most popular. Logic languages are the most intuitive and easy to understand even though it can be very computationally demanding for complex problems. Temporal Logic such as Linear Temporal Logic (LTL), Metric Temporal Logic (MTL), or Computational Tree Logic (CTL) are some of the most well used to describe specifications for path planning. In this thesis, some background to the work done with some of those languages will be presented and comparisons to the alternative chosen for this work.

2. Signal Temporal Logic

Most of the notation in this section mirrors that in [9] and [10], and missing details can be found in either works.

2.1. Discrete-Time Systems

We consider discrete-time systems of the form

$$x_{t+1} = f(x_t, u_t, w_t) \tag{1}$$

where $t = 0, 1, \dots$ indicate the time steps, $x_t \in \mathcal{X} \subseteq (\mathbb{R}^{n_c} \times \{0,1\}^{n_l}$ denotes the states, $u_T \in U \subseteq (\mathbb{R}^{m_c} \times \{0,1\}^{m_l})$ denotes the control inputs, $w_t \in W \subseteq (\mathbb{R}^{e_c} \times \{0,1\}^{e_l}$ denotes the external inputs or disturbances, and the initial state is $x_0 \in \mathcal{X}$. Note that all the variables may have real-valued or binary components. A *run* $\sigma = (x_0u_0w_0)(x_1u_1w_1)(x_2u_2w_2)\dots$ is an infinite sequence of state, input, and disturbance components at each time step t. Similar to [9], the horizon-N run of a system modeled by (1), denoted by $\sigma = (x_0, \mathbf{u}^N, \mathbf{w}^N) = (x_{0,0}, w_0)(x_1, u_1, w_1)(x_2, u_2, w_2)\dots(x_N, u_N, w_N)$ is assumed to be unique for an initial state $x_0 \in X$, a control input sequence $\mathbf{u}^N = u_0u_1u_2\dots u_{N-1} \in U^N$ and a sequence of environment inputs $\mathbf{w}^N = w_0w_1w_2\dots w_{N-1} \in W^N$. We also consider a generic cost function $J(\sigma(x_0, \mathbf{u}.\mathbf{w}))$ that maps (infinite and finite) runs to $\mathbb{R}$.

2.2. Syntax

STL formulas are defined recursively according to the grammar

$$\varphi ::= \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Box_{[a,b]}\ \psi \mid \varphi\, \mathcal{U}_{[a,b]}\ \psi$$

where μ is a predicate whose value is determined by the sign of a function of an underlying signal $\mathbf{x}$, i.e., $\mu \equiv \mu(x) > 0$, and ψ is an STL formula. The predicate $\neg\mu$ is the negation of the predicate μ, using boolean operators such as conjunction ($\wedge$), disjunction ($\vee$), or until ($\mathcal{U}$) we can write a set of predicates denominated as a STL formula ψ. From [1] temporal operators $\Diamond$ (eventually) and $\Box$ (always) can be derived from the previous boolean operators.

All the STL formulas can be defined with respect to signal $\mathbf{x}$ at time t in the following manner

$$\begin{aligned}
(\mathbf{x}, t) \models \mu &\Leftrightarrow \mu(x_t) > 0\\
(\mathbf{x}, t) \models \neg\mu &\Leftrightarrow \neg((\mathbf{x}, t) \models \mu)\\
(\mathbf{x}, t) \models \varphi \wedge \psi &\Leftrightarrow (\mathbf{x}, t) \models \varphi \wedge (\mathbf{x}, t) \models \psi\\
(\mathbf{x}, t) \models \varphi \vee \psi &\Leftrightarrow (\mathbf{x}, t) \models \varphi \vee (\mathbf{x}, t) \models \psi\\
(\mathbf{x}, t) \models \Box_{[a,b]}\varphi &\Leftrightarrow \forall t' \in [t+a, t+b], (\mathbf{x}, t') \models \varphi\\
(\mathbf{x}, t) \models \varphi\, \mathcal{U}_{[a,b]}\ \psi &\Leftrightarrow \exists t' \in [t+a, t+b] \text{ s.t. } (\mathbf{x}, t') \models \psi\\
&\quad \wedge \forall t'' \in [t, t'], (\mathbf{x}, t') \models \varphi
\end{aligned}$$

A signal $\mathbf{x} = x_0x_1x_2\dots$ satisfies φ, denoted by $\mathbf{x} \models \varphi$, if $(\mathbf{x}, 0) \models \varphi$. Informally, $\mathbf{x} \models \Box_{[a,b]}\varphi$ if φ holds at every time step before a and b, and $\mathbf{x} \models \varphi\, \mathcal{U}_{[a,b]}\ \psi$ if φ holds at some time step between a and b until ψ becomes true. Additionally, $\Diamond_{[a,b]}\varphi = \top\, \mathcal{U}_{[a,b]}\ \varphi$, so that $\mathbf{x} \models \Diamond_{[a,b]}\varphi$ if φ holds at some time step between a and b. STL formulas refer to discrete time intervals of discrete-time systems.

An STL formula φ is *bounded-time* if it contains no unbounded operators; the *bound* of φ is the maximum over the sums of all nested upper bounds on the temporal operators, and provides a conservative maximum trajectory length required to decided its satisfiability. For example, for $\Box_{[0,4]}\Diamond_{[3,6]}\varphi$, a trajectory of length $N \geq 5 + 4 = 9$ is sufficient to determine whether the formula is satisfiable.

2.3. Robust Satisfaction of STL formulas and MILP encoding

In this work, we consider the quantitative or robust semantics for STL, which assigns a real value to a predicate as an indication of to what extend a formula is satisfied. Following [9], the robustness function is defined as a real-valued function ρ^φ of signal $\mathbf{x}$ and t such that $\rho^\varphi(\mathbf{x}, t) > 0 \equiv (\mathbf{x}, t) \models \varphi$ and they are computed recursively as follows

$$\begin{aligned}
\rho^{\mu}(\mathbf{x}, t) &= \mu(x_t)\\
\rho^{\neg\mu}(\mathbf{x}, t) &= -\mu(x_t)\\
\rho^{\varphi\wedge\psi}(\mathbf{x}, t) &= \min(\rho^{\varphi}(\mathbf{x}, t), \rho^{\psi}(\mathbf{x}, t))\\
\rho^{\varphi\vee\psi}(\mathbf{x}, t) &= \max(\rho^{\varphi}(\mathbf{x}, t), \rho^{\psi}(\mathbf{x}, t))\\
\rho^{\Box_{[a,b]}\varphi}(\mathbf{x}, t) &= \min_{t' \in [t+a, t+b]} \rho^{\varphi}(\mathbf{x}, t')\\
\rho^{\varphi\, \mathcal{U}_{[a,b]}\ \psi}(\mathbf{x}, t) &= \max_{t' \in [t+a, t+b]} (\min(\rho^{\psi}(\mathbf{x}, t'),\\
&\qquad \min_{t'' \in [t,t']\rho^{\varphi}(\mathbf{x}, t''))}
\end{aligned}$$

For example, the robust satisfaction of $\mu_1 \equiv x - 5 > 0$ at time 0 is $\rho^{\mu_1}(x, 0) = x_0 - 5$.

To avoid the curse of dimensionality of the state space abstraction, STL specifications can be encoded as mixed-integer linear constraints [9]. Accordingly, we restrict ourselves to discrete time systems and to linear or affine predicates. Note that in case of dealing with continuous dynamics, we can approximate them by their equivalent discretized model using an appropriate sampling time. Using this encoding, the optimal input sequence for system 1 (as will be shown in the next section) can be obtained by solving a mixed integer linear programming (MILP) optimization problem at each time step. The interested reader is referred to [9] for the details of this encoding.

The advantage of this *robustness-based* encoding is that it allows us to obtain a trajectory that maximizes or minimizes the robustness of satisfaction. The objective function for the optimization problem can be then defined as the robustness function, which will be maximized, or the robustness function can be a complement to domain-specific objectives on runs of the system.

As mentioned in [9], MILPs are NP-hard, and hence impractical when the dimensions of the problem grow. We present the computational costs of the above encoding in terms of the number of variables and constraints in the resulting MILP. If P is the set of predicates used in the formula and $|\varphi|$ is the length (i.e. the number of operators), and N is the number of STL formulas, then $O(N|P|) + O(N|\psi|)$ continuous variables are introduced. In addition, $O(N)$ binary variables are introduced for every instance of a Boolean operator, i.e. $O(N|\varphi|)$ Boolean variables.

3. Synthesization of STL specifications

This encoding consists of system constraints and STL constraints [9].

3.1. System Constraints

3.1.1 Propositional calculus and linear integer programming

Consider X_i as a *literal* which can take values of "T" (true) or "F" (false). With this *literals* we use boolean operators such as $\wedge$ (and), $\vee$ (or), $\neg$ (not), $\Rightarrow$ (implies), $\Leftrightarrow$, (if and only if), $\oplus$, (exclusive or), to write statements like the one represented below.

$$(X_1 \vee X_2) \wedge X_3 \Rightarrow X_4 \Leftrightarrow \neg(X_5 \oplus X_6)$$

Correspondingly one can associate with a literal X_i a *logical variable* $\delta_i \in \{0,1\}$, which has a value of either 1 if $X_i = \mathrm{T}$, or 0 otherwise. In [4] we get that integer programming was considered an efficient way to perform automated deduction. By converting propositional logic problems into linear inequalities involving logical variables δ_i by means of a linear integer program. Here are some examples given that can prove that equivalence.

$$X_1 \vee X_2 \text{ is equivalent to } \delta_1 + \delta_2 \geq 1 \tag{2a}$$

$$X_1 \wedge X_2 \text{ is equivalent to } \delta_1 = 1, \delta_2 = 1 \tag{2b}$$

$$\neg X_1 \text{ is equivalent to } \delta_1 = 0 \tag{2c}$$

$$X_1 \Rightarrow X_2 \text{ is equivalent to } \delta_1 - \delta_2 \leq 0 \tag{2d}$$

$$X_1 \Leftrightarrow X_2 \text{ is equivalent to } \delta_1 - \delta_2 = 0 \tag{2e}$$

As we are interested in systems which have both logic and dynamics, we wish to establish a link between the two worlds. For this purpose, we end up with *mixed-integer linear inequalities*, i.e linear inequalities involving both *continuous variables* $x \in \mathbb{R}^n$ and logical (*indicator*) variables $\delta \in \{0,1\}$. Consider the statement $X \triangleq [f(x) \leq 0]$, where $f : \mathbb{R}^n \Rightarrow \mathbb{R}$ is linear, assume that $x \in \mathcal{X}$, where $\mathcal{X}$ is a given bounded set, and define

$$M \triangleq \max_{x \in \mathcal{X}} f(x) \tag{3a}$$

$$m \triangleq \min_{x \in \mathcal{X}} f(x) \tag{3b}$$

M and m are over and under estimates, respectively. Therefore equation $y = \delta f(x)$ is equivalent to

$$\begin{aligned} &y \leq M\delta \\ &y \geq m\delta \\ &y \leq f(x) - m(1-\delta) \\ &y \geq f(x) - M(1-\delta) \end{aligned} \tag{4}$$

3.1.2 Mixed Logical Dynamical Systems

The tools obtained in the previous section will be used now to express relations describing the evolution of systems where physical laws, logic rules, and operating constraints are interdependent. We are able to do this by describing mixed logical dynamical (MLD) systems through the following linear relations [3]:

$$\begin{aligned} x(t+1) &= A_t x(t) + B_{1t} u(t) + B_{2t} \delta(t) + B_{3t} z(t) \\ y(t) &= C_t x(t) + D_{1t} u(t) + D_{2t} \delta(t) + D_{3t} z(t) \\ E_{1t} u(t) &+ E_{2t} \delta(t) + E_{3t} z(t) + E_{4t} x(t) \leq E_{5t} \end{aligned} \tag{5}$$

where $t \in \mathbb{Z}$,

$$x = \begin{bmatrix} x_c \\ x_l \end{bmatrix}, x_c \in \mathbb{R}^{n_c}, x_l \in \{0,1\}^{n_l}, n \triangleq n_c + n_l$$

is the state of the system, whose components are distinguished between continuous x_c and 0-1 x_l;

$$y = \begin{bmatrix} y_c \\ y_l \end{bmatrix}, y_c \in \mathbb{R}^{p_c}, y_l \in \{0,1\}^{p_l}, p \triangleq p_c + p_l$$

is the output vector,

$$u = \begin{bmatrix} u_c \\ u_l \end{bmatrix}, u_c \in \mathbb{R}^{m_c}, u_l \in \{0,1\}^{m_l}, m \triangleq m_c + m_l$$

is the command input, collecting both continuous commands u_c, and binary (on/off) commands u_l; $\delta \in \{0,1\}^{r_l}$ and $z \in \mathbb{R}^{r_c}$ represent respectively auxiliary logical and continuous variables. The continuous variable is defined as follows $z(t) = \delta(t)x(t)$.

3.2. Signal Temporal Logic (STL) constraints

Given a formula φ, we introduce a variable z_t^{φ}, whose value is tied to a set of mixed integer linear (MIL) constraints represented above. In other words, z_t^{φ} has an associated set of MILP constraints such that $z_t^{\varphi} = 1$ if and only if φ holds at position t. For z_0^{φ} represents the formula φ in the initial state and whether or not it holds.

The predicates are represented by constraints on system state variables. For each predicate $\mu \in P$, a binary variable $z_t^{\mu} \in \{0,1\}$ is introduced. The following constraints enforce that $z_t^{\mu} = 1$ if and only if $\mu(x_t) > 0$:

$$\begin{aligned} \mu(x_t) &\leq M_t z_t^{\mu} - \in_t \\ -\mu(x_t) &\leq M_t(1 - z_t^{\mu}) - \in_t \end{aligned} \tag{6}$$

where M_t is a sufficiently large positive number, and $\in_t$ a sufficiently small positive number that serve to bound $\mu(x_t)$ away from 0. M_t is just like M and m from section 3.1 instead that $M = M_t$ and $m = -M_t$.

The boolean operations on MILP variables described above are defined by logical operations. The logical operations on variables $z_t^{\varphi} \in [0,1]$ are the following:

Negation: $z_t^{\varphi} = \neg z_t^{\psi}$ which can also be represented

$$\text{as } z_t^{\varphi} = 1 - z_t^{\psi},\ z_t^{\varphi} \leq z_{t_i}^{\psi_i}, i = 1, \ldots, m$$

Conjunction: $z_t^{\varphi} = \wedge_{i=1}^{m} z_{t_i}^{\psi_i}$ which can be represented

$$\text{as } z_t^{\varphi} \geq 1 - m + \sum_{i=1}^{m} z_{t_i}^{\psi_i},\ z_t^{\varphi} \geq z_{t_i}^{\psi_i}, i = 1, \ldots, m$$

Disjunction: $z_t^{\varphi} = \vee_{i=1}^{m} z_{t_i}^{\psi_i}$ which can be represented

$$\text{as } z_t^{\varphi} \leq \sum_{i=1}^{m} z_{t_i}^{\psi_i}$$

Temporal operators $\Box$ and $\Diamond$ also get appropriate encoding:

Always: $\varphi = \Box_{[a,b]}\psi$ which can also be represented

$$\text{as } z_t^{\varphi} = \wedge_{i=a_t^N}^{b_t^N} z_i^{\psi}$$

Let $a_t^N = \min(t + a, N)$ and $b_t^N = \min(t + b, N)$ where N is the receding horizon.

Eventually: $\varphi = \Diamond_{[a,b]}\psi$ which can also be

$$\text{represented as } z_t^{\varphi} = \vee_{i=a_t^N}^{b_t^N} z_i^{\psi}$$

Until: $\varphi = \psi_1 \mathcal{U}_{[a,b]} \psi_2$

The definition of until is not relevant for this thesis so we don't represent in succinct detailed. However, its definition can be found in [9].

The combination of the STL constraints, system constraints and loop constraints gives the MILP encoding, and this enables checking feasibility of this set and find a solution using an MILP solver. After determining an objective function of the system, it is possible to find the optimal trajectory that satisfies the STL specifications.

3.3. Robustness-base encoding

In this section we give a short representation of the robustness function discussed in the section 2.3. The robustness can be computed recursively on the structure of the formula in conjunction with the generation of the constraints. In addition, since max and min operations can be expressed in an MILP formulation using additional binary variables, this does not add complexity to the encoding.

The encodings of the temporal operators work upon the encodings defined above. For each predicate $\mu \in P$, it is now introduced variables r_t^{μ} for time indices $t = 0, 1, \ldots, N$, and set $r_t^{\mu} = \mu(x_t)$. For r_t^{φ} where φ is a Boolean formula, it is assumed that each operand φ has a corresponding variable $r_t^{\varphi} = \rho^{\varphi}(\mathrm{x}, t)$. The Boolean operations are defined as

Negation: $r_t^{\varphi} = \neg r_t^{\psi}$ which can also be represented

$$\text{as } r_t^{\varphi} = -r_t^{\psi}$$

$$\text{Conjunction: } r_t^{\varphi} = \wedge_{i=1}^{m} r_{t_i}^{\psi_i}$$

$$\sum_{i=1}^{m} \rho_{t_i}^{\psi_i} = 1$$

$$r_t^{\varphi} \leq z_{t_i}^{\psi_i}, i = 1, \ldots, m$$

$$r^{\psi_i} t_i - (1 - \rho_{t_i}^{\psi_i})M \leq r_t^{\varphi} \leq r^{\psi_i} t_i + M(1 - \rho_{t_i}^{\psi_i})$$

Together, these constraints enforce that $r_t^{\varphi} = \min(r_{t_i}^{\psi_i})$.

$$\text{Disjunction: } r_t^{\varphi} = \wedge_{i=1}^{m} r_{t_i}^{\psi_i}$$

$$\sum_{i=1}^{m} \rho_{t_i}^{\psi_i} = 1$$

$$r_t^{\varphi} \geq z_{t_i}^{\psi_i}, i = 1, \ldots, m$$

$$r^{\psi_i} t_i - (1 - \rho_{t_i}^{\psi_i})M \leq r_t^{\varphi} \leq r^{\psi_i} t_i + M(1 - \rho_{t_i}^{\psi_i})$$

Together, these constraints enforce that $r_t^{\varphi} = \max(r_{t_i}^{\psi_i})$. The encoding for bounded temporal operators is defined as in section 2.3; The advantage of this encoding is that it allows to maximize robustness of satisfaction. Additionally, an encoding based on robustness has the advantage of allowing the STL constraints to be softened or hardened as necessary. However, due to additional binary variables is more computational expensive. On the other hand, the robustness constraints are more easily relaxed, which allows a simpler cost function to solve the same problem if it didn't had robustness in the objective function.

4. Results

After all problems have been addressed it is time to add everything together with the purpose of reproducing the **Challenge 3** of the international competition MBZIRC 2017. The mission, as it was described previously, involves a coordinated mission between 3 quadrotors to search and grab 3 individual objects and one object cooperatively in a dynamical environment. The first part consists of the 3 quadrotors starting from a known location and searching the objects in the environment. The high level controller developed in this thesis allows us to focus primarily on the trajectory generated of each model leaving the more low-end priorities such as locating the objects and its physical characteristics for the low level controllers. As such, we assume a priori that the exact location of each object is given to us before starting the mission. The starting point of the mission can be seen in Figure 1(a). The first mission is completed when each quadrotor reaches its object and grabs it (Figure 1(b)). The second objective, performed in Figure 1(c), is to relocate the objects to a new position and, when the final destination of the object has been reached, the quadrotor can drop the object. This summarizes the first part of the mission we set out to do, where all the different problems discussed in the previous section can occur and where the alternative solutions can provide a feasible option that avoids any possible failing situations. The next and final part is allocated for the cooperative mission involving 2 quadrotors grasping and transporting of a fairly large size object (200×20×20 cm). This object respects the dimensions provided by the MBZIRC 2017 competition guidelines. The object will be grabbed by one quadrotor on each end to avoid any unpredictable collisions.

This cooperative team work is achieved by expressing the desired movement of the two quadrotors present in this objective. In a cooperative mission two or more vehicles are working together to fulfill a goal, in our case is to grab an object that can't be transported by just one quadrotor. Knowing that the current environment contains both types of objects (individual objects and big objects), the quadrotors need to be able to distinguish between the two in order to perform the right mission. For the scope of this thesis we don't address properties of the object such as weight, type of material, etc. We assume that the quadrotor identifies automatically what object is dealing with in contact. The mission describes then a first quadrotor

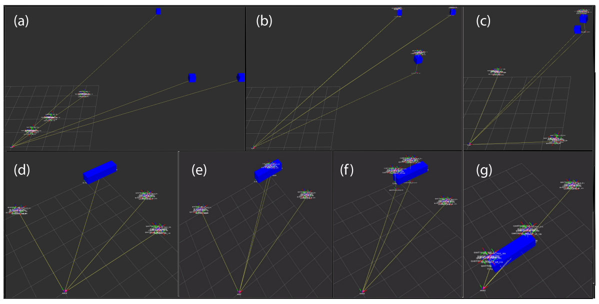

Figure 1: The final mission illustrated step-by-step representing every major event. In (a) is the starting point of the 3 quadrotors with the initial position of the 3 individual objects; (b) shows the quadrotors reaching the objects and transporting them to a new location; (c) illustrates the new location for all 3 objects and the new starting points of quadrotor 1 and 2; (d) is where the new mission starts with the big object initialized, and quadrotor 3 is shown in its final position where it will stay for the remaining of the time; (e), (f), and (g) are the 3 step process of moving the big object: quadrotor 1 goes to the object to evaluate its size, then quadrotor 1 waits for quadrotor 2 to arrive since it can't transported by itself, and finally both quadrotors moving the object to its final location. YouTube video: `https://www.youtube.com/watch?v=Y-KO_NPfYvw`

reaching its closest object, identifies that it is in fact a big object, and waits for the second quadrotor to come to its assistance. Note that the STL specifications written next are simplified to just one possible case. This is the case when Object 1 is closer than Object 2 or 3. However, this may not always be the case as such additional STL specifications are added to deal with all possible scenarios.

$$\begin{aligned}
&\Diamond_{[0,3]}(x_{1_x}(t) - Obj_{1_x}(t))^2 + (x_{1_y}(t) - Obj_{1_y})^2 \\
&\quad < (x_{1_x}(t) - Obj_{2_x}(t))^2 + (x_{1_y}(t) - Obj_{2_y})^2 \wedge \\
&\quad (x_{1_x}(t) - Obj_{1_x}(t))^2 + (x_{1_y}(t) - Obj_{1_y}(t))^2 \\
&\quad < (x_{1_x}(t) - Obj_{3_x}(t))^2 + (x_{1_y}(t) - Obj_{3_y} t))^2 \;\; => \\
&\quad \Diamond_{[0,3]}(x_{1_x}(t) \approx Obj_{1_x}(t) \wedge x_{1_y}(t) \approx Obj_{1_y}(t) \\
&\quad \wedge x_{1_z}(t) \approx 0)
\end{aligned}$$

$$\begin{aligned}
&(x_{1_x}(t) \approx Obj_{1_x}(t) \wedge x_{1_y}(t) \approx Obj_{1_y}(t) \wedge x_{1_z}(t) \approx 0) \\
&\quad => \Box_{[4,\infty]}(x_{1_x} + 1 < x_{2_x}(t) < x_{1_x} + 2 \;\; \wedge \\
&\quad x_{1_y} < x_{2_y}(t) < x_{1_y} + 0.2 \wedge x_{1_z} < x_{2_z}(t) \\
&\quad < x_{1_z} + 0.2) \;\; \wedge \\
&\quad \Diamond_{[4,8]}(x_{1_x}(t) \approx Goal_x(t) \wedge x_{1_y}(t) \approx Goal_y(t) \\
&\quad \wedge x_{1_z}(t) \approx Goal_z(t))
\end{aligned}$$

$$\Diamond_{[0,3]}(x_{i_z}(t) > 3 \wedge x_{i_z}(t) < 5)$$

The solution to the problem described above is represented in Figure 2. The first part of the equations describes simply the first quadrotor reaching its closest object. The same identical STL specification described in all missions involving quadrotors attempting to grasp the closest individual object to its position. The second equation is what allows the mission to adapt to the situation and call for backup when the object requires a second quadrotor to be lifted. Again, we assume the moment quadrotor

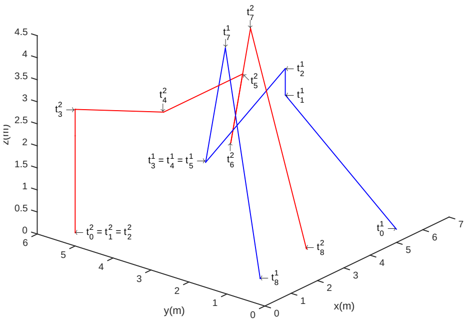

Figure 2: The trajectories path of the quadrotors involved in the cooperative mission of transporting a big object.

1 reaches the object (t_3^1) it has on-board sensors that allow to distinguish the current object from the ordinary single object as well as send a signal to quadrotor 2 that it requires its assistance. After those low-level conditions have been established the second part of the equation can be used. This equation will only hold true when the quadrotor 1 is on top of the object. After that the STL specifications applied to quadrotor 2 and the cooperative mission will be in play. Note that quadrotor 2 is in its initial position from time $t = 0$ to $t = 2$, this is due to quadrotor 1 not having reached the object at those time steps. When at time $t = 3$ quadrotor 1 reaches the object then the cooperative mission starts, as the second equation holds true and orders are given. This specifications are to the right of the boolean operator (=>) to represent the next steps to make after quadrotor 1 reaches the object. This orders quadrotor 2 to converge its position to the location of quadrotor 1 (t^2 reach t_3^1), at the same time keep a safe distance to it and also respect the object dimensions so it won't miss the grasping. This is the reason why t_3^1 and t_6^2 are not the exact same location, due to the dimension of the object defined previously. The last part of the STL specification is regarding the new goal position of the object to be dropped. Since the previous rule needs to be kept from $t = 4$ until the end of the simulation ($\Box_{[4,\infty]}(x_{1_x} + 1 < x_{2_x}(t) < x_{1_x} + 2 \wedge x_{1_y} < x_{2_y}(t) < x_{1_y} + 0.2 \wedge x_{1_z} < x_{2_z}(t) < x_{1_z} + 0.2)$) by ordering quadrotor 1 to move to that new goal, quadrotor 2 will automatically follow the same path creating the so desired cooperative mission between two quadrotors transporting a big object as seen from Figure 1(f) until it is dropped viewed in subpicture (g). However to replicate this same parallel motion into the Gazebo simulator, some changes needed to be done to the way data is sent and processed between nodes. As a result, from the moment the two quadrotors reach the object and the object is grabbed the goal positions from the trajectory generated by the STL solver will be sent at the same time synchronized by each STLnode# of each quadrotor. This prevents one of the quadrotors to move faster than the other one, making it a smooth transportation of the object without any disturbances.

5. Conclusions

The major achievements of the present work can be summarized to being able to show the advantages of using Signal Temporal Logic language to describe the mission specifications on a high-end level controller. Previous research papers have been written on using different techniques with LTL language that prove the feasibility of multi-vehicle coordinated missions [7] [8]. In this thesis we present the differences between the existing work using LTL with the now new alternative language that is STL. The advantageous being that its variables are continuous-time signals rather than discrete-time signals allowing for a more robust satisfaction parameter than the previous yes or no answer for the satisfiability in discrete-time. Additionally, the use of a MPC controller that allows us to adapt in real time to the current situation to disturbances that may happen ahead of time, giving us results less prone to feasibility problems than the ones given by previously researched methods.

Even though the major goals of this thesis were achieved, it must be noted that this is still a very particular example and it is only fully functional in a completely known environment with perfect conditions. It is true that the goal of this thesis was to solve the problem **Challenge 3** from the international competition MBZIRC 2017. Nevertheless, it would be interesting to adapt our STL specifications to a mission with more than 3 quadrotors. As some of the missions developed in this thesis are limited to only 3 vehicles. Additionally, it would be interesting to understand how to deal with unknown or partially known object locations as well as uncertainties in the quadrotor current location. Athough a complete controller was developed and simulated it was not possible to run the tests with the real system. Due to the lack of knowledge on the real hardware and not enough time to implement my own external algorithm, it was only possible to test the controller. The last thing to be noted is that, despite only being tested for environments adapted to the **Challenge 3** problem, the contribution of this thesis can be used as a development environment and testbed for future work.

Acknowledgements
As I am writing these last words to finish my thesis, the realization that this document will be the very last milestone for graduating from my study at Instituto Superior Técnico, is slowly sicking in. It has been an exciting, sometimes difficult, but always very interesting journey, and my thesis project was no different. First of all I would like to thank my family and friends, for not only supporting me during the complete time of my study in IST, but also for being understanding whenever I was stressed out or easily irritable, especially during the final weeks of my thesis project. Then to my two supervisors Prof. Pedro Lima and Prof. Richard Murray whose support was instrumental for this thesis and who was available to me whenever I needed. I want to give my special thanks to Prof. Richard Murray and Samira Farahani for giving me the opportunity to work on a summer internship in Caltech. Without it I would never have learned about Signal Temporal Logic and I wouldn't have done a master thesis on the subject. I would also like to thank all my friends who supported me and made me feel happy especially during the difficult times.

References

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT press, 2008. ISBN 978-0-262-02649-9.

[2] J. Bellingham, M. Tillerson, M. Alighanbari, and J. How. Cooperative path planning for multiple uavs in dynamic and uncertain environments. *41st IEEE Conference on Decision and Control Las Vegas NV*, 3:2816 – 2822, Dec. 2002. doi:10.1109/CDC.2002.1184270.

[3] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:347–536, Mar. 1999.

[4] T. M. Cavalier, P. M. Pardalos, and A. L. Soyster. Modeling and integer programming techniques applied to propositional calculus. *Computers and Operations Research*, pages 561–570, 1990.

[5] M. Faied, A. Mostafa, and A. Girard. Dynamic optimal control of multiple depot vehicle routing problem with metric temporal logic. *American Control Conference Hyatt Regency Riverfront St. Louis MO USA*, June 2009.

[6] S. Karaman. Optimal planning with temporal logic specifications. MIT, 2009.

[7] S. Karaman and E. Frazzoli. Linear temporal logic vehicle routing with applications to multi-uav mission planning. *International Journal of Robust and Nonlinear Control*, 21(12):1372?1395, Aug. 2011. doi: 10.1002/rnc.1715.

[8] B. Lacerda and P. Lima. Ltl-based decentralized supervisory control of multi-robot tasks modelled as petri nets. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[9] V. Raman, A. Donzé, M. Maasoumy, R. Murray, A. Sangiovanni-Vincentelli, and S. Seshia. Model predictive control with signal temporal logic specifications. *53rd IEEE Conference on Decision and Control CDC Los Angeles CA USA*, pages 81–87, Dec. 2014. doi:10.1109/CDC.2014.7039363.

[10] V. Raman, A. Donzé, D. Sadigh, R. Murray, and S. Seshia. Reactive synthesis from signal temporal logic specifications. *Hybrid Systems: Computation and Control HSCC Seattle WA USA*, pages 239–248, Apr. 2015. doi:10.1145/2728606.2728628.

[11] A. Richards, J. Bellingham, M. Tillerson, and J. How. Coordination and control of multiple uavs. *AIAA Guidance Navigation and Control Conference*, 2002.

[12] M. Roberts, T. Apker, B. Johnson, B. Auslander, B. Wellman, and D. W. Aha. Coordinating robot teams for disaster relief. *28th International Florida Artificial Intelligence Research Society Conference*, 2014.