Reactive Controller Synthesis for UAV Mission Planning

Kyunghoon Cho, Yunho Choi and Songhwai Oh
Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul, Korea
(Tel : +82-2-880-1512; E-mail: {kyunghoon.cho,yunho.choi,songhwai.oh}@cpslab.snu.ac.kr)

***Abstract*—This paper presents an UAV control scheme under mission specifications. A mission is specified in linear temporal logic (LTL) formula. A point-to-point local planner is designed to search a trajectory satisfying the given mission specification so that the low-level controller of a quadrotor can follow the trajectory. The designed planner adopts control Lyapunov functions, and has the following properties: (1) It reactively generates a trajectory with respect to current quadrotor state. (2) The generated trajectory reflects the reference path produced during the offline planning procedure. The proposed method is flexible in responding to disturbances and guides the quadrotor to satisfy the given mission requirement.**

***Keywords*—Mission planning, Unmanned aerial vehicle, Linear temporal logic**

1. Introduction

Unmanned aerial vehicles (UAVs) have been attracting attention in broad area including both military and civilian applications. Recently Amazon has launched a delivery system which utilizes UAVs to deliver packages to customers, and such services are anticipated to prevail in the near future. Advances in guidance technologies allow UAVs to perform from simple missions such as reaching goal to complicated ones like the vehicle routing problem.

Temporal logics describe high-level specifications with temporal and logical constraints. Especially, linear temporal logic (LTL) is widely used to specify robot task specifications [1]–[3]. For instance, in pick-up and delivery problems, "(eventually) pick item A and deliver it to customer B" can be written in LTL formula $\phi = \Diamond(P_A \wedge \Diamond D_B)$.

In this paper, we design a local planner that guides a quadrotor to meet a mission specification, where missions are represented as a syntactically co-safe LTL formula. The designed planner utilizes control Lyapunov functions so that the desired trajectory can be searched with a single-step computation. This property allows the planner reactively generates the trajectory with respect to quadrotor's state in real-time. We adopted the Lyapunov function design method in [4], where the Lyapunov functions reflect the reference trajectories sought in the off-line planning procedure. The proposed planner can handle disturbances flexibly and lead the quadrotor to fulfill the given mission requirement.

2. Mathematical Framework

2.1. System Model

The quadrotor system is modeled as a rigid body with a twelve-dimensional state vector

$$\mathrm{x} = [\xi, \dot{\xi}, \nu, w].$$

$\xi = (x, y, z)$ is a position in the three-dimensional Cartesian space, and $\dot{\xi} = (\dot{x}, \dot{y}, \dot{z})$ is a linear velocity. $\nu = (\psi, \phi, \theta)$ corresponds to Euler angles (roll, pitch, yaw), and $w = (p, q, r)$ is an angular velocity. The position of quadrotor evolves according to the dynamic model:

$$\begin{aligned} m\ddot{\xi} &= \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum_i F_i \end{bmatrix} \\ R &= \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi c\theta & s\phi & c\phi c\theta \end{bmatrix}, \end{aligned}$$

where m is the mass, F_i is the force of each rotor and R is the rotation matrix with $c\theta$ and $s\theta$ denoting $\cos\theta$ and $\sin\theta$ respectively. The relationship between the angular velocity w and time derivative of Euler angles can be described as below:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}.$$

The quadrotor is controlled by the force of each motor F_i. We omit the detailed dynamic model of the quadrotor which is shown in [5].

$\mathcal{W} \in \mathbb{R}^3$ denotes the robot's workspace which describes the physical space where a robot operates. The mapping function $h : \mathrm{x} \to \xi \in \mathcal{W}$ extracts position from quadrotor state.

2.2. Linear Temporal Logic (LTL)

Linear temporal logic is a logical formalism, which is suited for specifying linear time properties. The basic ingredients of an LTL formula are a set of atomic propositions (APs), and Boolean and temporal operators. LTL formulas are formed according to the following grammar [6]: $\phi ::= true \,|\, a \,|\, \phi_1 \vee \phi_2 \,|\, \neg\phi \,|\, \bigcirc\phi \,|\, \phi_1 \mathrm{U} \phi_2$, where $a \in AP$, $\bigcirc$ is the *next* operator, and U is the *until* operator. There are also derived operators such as $\Box$ (*always*), $\Diamond$ (*eventually*), and $\Rightarrow$ (*implication*).

An atomic proposition is a statement or assertion which must be true or false. We use Π to denote a collection of all atomic propositions, i.e., $\Pi = \{\pi_0, \pi_1, ..., \pi_N\}$. The semantics of LTL are defined over infinite traces of a given system, where a trace σ is a sequence of atomic propositions. For given trace σ, the notation $\sigma \vDash \phi$ denotes that σ satisfies ϕ.

A. Syntactically Co-Safe LTL: In this work, we treat a restricted form of LTL formulas, which is called syntactically co-safe LTL formulas (sc-LTL). An LTL formula ϕ is co-safe if any infinite trace that satisfies ϕ has a finite prefix which also satisfies ϕ [7]. Syntactically co-safe LTL formulas are

the LTL formulas that contain only $\bigcirc, \Diamond, \mathrm{U}$ operators, when written in a positive normal form ($\neg$ appears only in front of atomic propositions).

B. Automaton Representation: Given a set of atomic propositions Π and a syntactically co-safe LTL formula ϕ, it is known that a nondeterministic finite automaton (NFA) can be constructed [8]. There exists off-the-shelf software which enables a fast translation from an LTL formula to NFA [6]. An NFA can be converted to a deterministic finite automaton (DFA), which is more convenient in computation. A DFA $\mathcal{A}_\phi$, corresponding to ϕ, is defined below.

Definition 1: A deterministic finite automaton DFA is a tuple $\mathcal{A}_\phi = (Q, \Sigma, \delta, q_{init}, Q_{acc})$, consisting of (i) a finite set of states Q, (ii) a finite alphabet $\Sigma = 2^\Pi$, (iii) a transition relation $\delta : Q \times \Sigma \to Q$, (iv) a set of initial states $q_{init} \subseteq Q$, and (v) a set of accepting states $Q_{acc} \subseteq Q$.

C. LTL Semantics over Trajectories: Let the workspace $\mathcal{W}$ contain regions of interest $P = \{P_1, ..., P_n\}$, and each atomic proposition $\pi_j \in \Pi$ corresponds to the region of interest P_j. A labeling function $L : \mathcal{W} \to 2^\Pi$ maps a state in the workspace to a set of true propositions. π_0 holds true for all workspace beside regions of interest. For given $\pi_i \in \Pi$, $\neg\pi_i$ holds true for $\{w \in \mathcal{W} \,|\, \pi_i \notin L(w)\}$.

For a discretized trajectory $\mathbf{x}_{\Delta t}(x_0) = \mathrm{x}_0, \mathrm{x}_1, ..., \mathrm{x}_m$ starting from x_0 with time step Δt, a trace can be defined as follows:

$$trace(\mathbf{x}_{\Delta t}(\mathrm{x}_0)) = L(h(\mathrm{x}_0)), L(h(\mathrm{x}_1)), ..., L(h(\mathrm{x}_m)).$$

For a given trace $trace(\mathbf{x}_{\Delta t}(\mathrm{x}_0, \mathbf{u}))$ and a DFA $\mathcal{A}_\phi$, automaton states are sequentially updated from the initial automaton state with respect to the transition relation in $\mathcal{A}_\phi$ [9]. If a reached automaton state is one of accepting states, then the trajectory $\mathbf{x}_{\Delta t}(\mathrm{x}_0)$ satisfies ϕ, denoted as $\mathbf{x}(\mathrm{x}_0))\models_{\Delta t}\phi$.

2.3. Candidate Lyapunov Function

Let us consider the system model described as below:

$$\dot{\mathbf{x}} = v,$$

where $\mathbf{x}(t) \subset \mathbb{R}^d$ defines a state at time $t \in \mathbb{R}$. A construction of valid Lyapunov candidates can be done in various ways. We construct Lyapunov candidates from demonstrations , which was introduced in [4]. The Lyapunov function is modeled as weighted sum of asymmetric quadratic functions (WSAQF):

$$\mathrm{V}(\mathbf{z}) = \mathbf{z}^T P^0 \mathbf{z} + \sum_{l=1}^{L} \beta^l(\mathbf{z})(\mathbf{z}^T P^l(\mathbf{z} - \mu_l))^2,$$

where $\mathbf{z} = \mathbf{x} - \mathbf{x}^*$ and $\mathbf{x}^*$ is a target state. L is the number of used asymmetric quadratic functions, μ_l are mean vectors to shape the asymmetry of the functions and $P^i \in \mathbb{R}^{d\times d}$ are positive definite matrices. The coefficient β^l is defined as follows:

$$\beta^l(\mathbf{z}) = \begin{cases} 1 & : \mathbf{z}^T P^l(\mathbf{z} - \mu_l) \geq 0 \\ 0 & : \mathbf{z}^T P^l(\mathbf{z} - \mu_l) < 0. \end{cases}$$

The main advantage of the WSAQF is the existence of an unique global minimum at $\mathbf{z} = 0$. Through learning parameters P^l and μ_l, demonstrated trajectories can be reflected in the designed function, and it can be done by solving the optimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N_{traj}} \sum_{k=1}^{N_i} \frac{1+\bar{w}}{2}\text{sign}(\psi^{i,k})(\psi^{i,k})^2 + \frac{1-\bar{w}}{2}(\psi^{i,k})^2 \\ \text{subject to} \quad & P^l \succ 0, \quad l = 0, ..., L, \end{aligned}$$

where $\succ$ denotes the positive definiteness of matrix and $\bar{w}$ is a small positive scalar. The function ψ is defined as

$$\psi^{i,k} = \frac{\nabla \mathrm{V}(\mathbf{x}_k^i)^T v_k^i}{\|\mathrm{V}(\mathbf{x}_k^i)\| \cdot \|v_k^i\|}.$$

3. Proposed Approach

The proposed approach consists of two major steps: offline and online planning. In the first step, a reference path satisfying the mission specification is searched by the offline planner. Based on the reference path, a set of Lyapunov functions and temporal goal points are constructed. The role of Lyapunov functions is to find trajectories reaching to the assigned targets, which leads the robot to satisfy the given sc-LTL formula. The second step is the online planning, a short-horizon trajectory is computed based on the Lyapunov functions and robot's current state. It gradually guides the robot to meet the given mission specification, even under disturbances.

3.1. Offline Planning

The offline procedure is given in Algorithm 1. A DFA for the sc-LTL formula ϕ is constructed and a satisfied path is searched by the offline planner (line 1-2). There exist various works for solving path planning problem under LTL specifications. We use one of the previous work [9] as the offline planner.

The found reference path Ξ is decomposed into a set of path segments. Let $P_{i_1}, ..., P_{i_N}$ is a sequence of interesting regions which the found reference path must pass to meet the mission specification. If p_i is a point in the region of interest P_i, then a sequence of points $p_{i_1}, ..., p_{i_N}$ satisfies the sc-LTL formula $trace([p_{i_1}, ..., p_{i_N}]) \models \phi$. A set of temporal target points ξ_k^* is defined as

$$\xi_k^* = \arg\min_{\xi\in\Xi} \|\xi - p_{i_k}\|, \ \forall k = 1, ..., N.$$

Path segments $\Xi_1, ..., \Xi_N$ are defined according to ξ_k^*, where each end point of Ξ_k is ξ_k^* for all $k = 1, ..., N$. The Lyapunov functions $\mathrm{V}_1, ..., \mathrm{V}_k$ are constructed from the path segments and the temporal targets [4]. Figure 1 describes the offline procedure for the LTL formula $\phi = \Diamond(a \wedge \Diamond b)$, which means "(eventually) reach region a, then b". One can notice that the contour map of Lyapunov function reflects the path segment. It has lower cost near path segment and the lowest at the end of path segment. Gradient flows of Lyapunov function supports the fact that ξ_k^* is an unique global minimum point.

3.2. Online Planning

The online procedure is shown in Algorithm 2. An online path planner is designed based on the Lyapunov functions (line 3). At each time step, $OnlinePlanner$ generates the desired

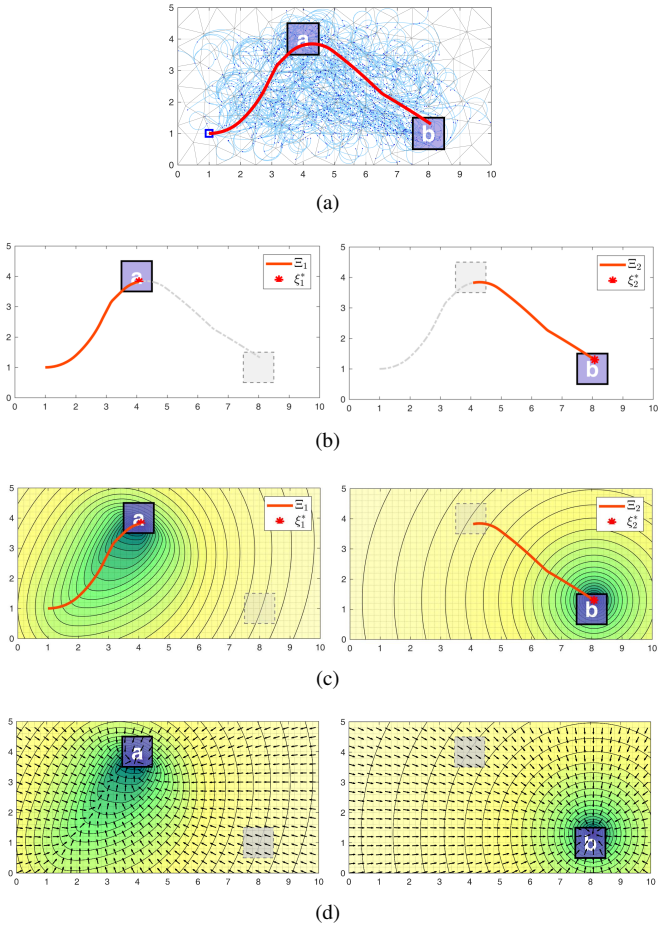


Fig. 1. The offline procedure. (a) The reference solution (orange line) found by the offline planner. (b) Path segments and temporal targets from the reference path. (c) Contour maps of Lyapunov functions reflecting path segments. (d) Gradient vector flows of Lyapunov function.

Algorithm 1 Offline Step $(\phi, \mathrm{x}_{init})$

1: $\mathcal{A}_\phi \leftarrow$ Automaton(ϕ)
2: $\Xi \leftarrow$ OfflinePlanner$(\phi, \mathrm{x}_{init})$
3: $\Xi_1, ..., \Xi_N, \xi_1^*, ..., \xi_N^*$
4: $\quad \leftarrow$ PathDecomposition$(\Xi, q_{init}, \mathcal{A}_\phi)$
5: **for** $k = 1$, $k{+}{+}$, while $k \leq N$ **do**
6: $\quad \mathrm{V}_k \leftarrow$ ConstructLyapunovFunc(Ξ_k, ξ_k^*)
7: **end for**

path with a short horizon, which leads the robot to satisfy the given LTL specification.

In order to find the desired path $\mathbf{x}_{des}$, a goal point $\xi_{des} \in \mathcal{W}$ is defined as

$$\xi_{des} = h(\mathrm{x}) + \eta \Delta\xi,$$

where η is a scaling constant between 0 and 1. $\Delta\xi$ is computed by solving the following optimization problem:

$$\begin{aligned} \underset{\Delta\xi}{\text{minimize}} \quad & \frac{1}{2}(\Delta\xi)^T(\Delta\xi) \\ \text{subject to} \quad & (\nabla \mathrm{V}_\xi(\xi))^T \Delta\xi < -\rho, \end{aligned} \tag{1}$$

where $\rho \geq 0$. We set $\rho = \|\nabla \mathrm{V}_\xi\|$, which is a slight variation of Sontag's control Lyapunov formula [10]. Notice that (1) is

Algorithm 2 Online Step $(\mathrm{x}_{init}, V, \xi^*)$

1: $\mathrm{x} \leftarrow \mathrm{x}_{init}$, $i \leftarrow 1$
2: **while** $i \leq N$ **do**
3: $\quad \mathbf{x}_{des} \leftarrow$ OnlinePlanner$(\mathrm{x}, V_i, \xi_i^*)$
4: $\quad \mathrm{x} \leftarrow$ Controller$(\mathrm{x}, \mathbf{x}_{des})$
5: $\quad$ **if** $\|h(\mathrm{x}) - \xi_i^*\| < \sigma_{thres}$ **then**
6: $\quad\quad i \leftarrow i + 1$
7: $\quad$ **end if**
8: **end while**

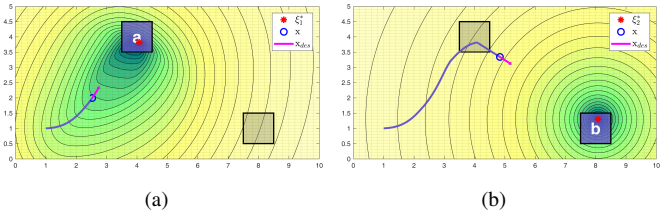


Fig. 2. Snapshots of the online procedure for the the LTL formula $\phi = \Diamond(a \wedge \Diamond(b))$. Different Lyapunov functions are used with respect to the current state of quadrotor.

a convex problem and has the global optimum point which can be solved analytically

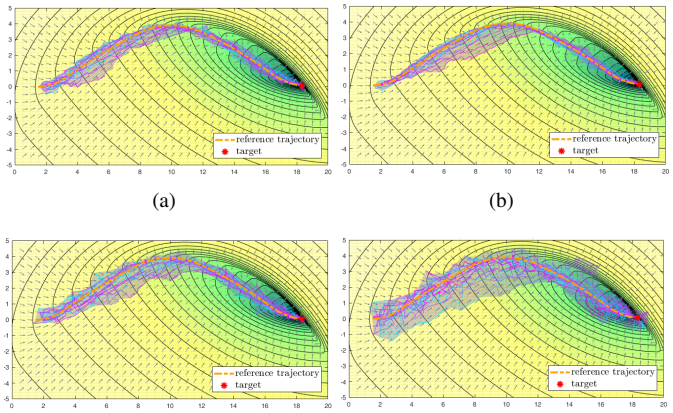
$$\Delta\xi = \begin{cases} 0 & : \xi = \xi^* \\ -\dfrac{\rho}{(\nabla V_\xi(\xi))^T(\nabla V_\xi(\xi))} \nabla \mathrm{V}_\xi(\xi) & : \xi = \mathcal{W} \backslash \xi^*. \end{cases}$$

The desired trajectory $\mathbf{x}_{des}$ is generated by smoothly interpolating two points $h(\mathrm{x})$ and ξ_{des}. The trajectory $\mathbf{x}_{des}$ is defined as a 7th-order polynomials of time t. It travels between a pair of points $h(\mathrm{x})$ and ξ_{des}, and takes a known amount of time T. $\mathbf{x}_{des}(t)$ can be represented as $\mathbf{x}_{des}(t) = \alpha_0 + \alpha_1(t/T) + \alpha_2(t/T)^2 + ... + \alpha_7(t/T)^7$. Coefficients of the polynomial are found under the following constraints:

$$\begin{aligned} &\mathbf{x}_{des}(0) = h(\mathrm{x}), && \mathbf{x}_{des}(T) = \xi_{ref}, \\ &\dot{\mathbf{x}}_{des}(0) = h_v(\mathrm{x}), && \dot{\mathbf{x}}_{des}(T) = 0, \\ &{\mathbf{x}_{des}}^{(k)}(0) = 0 \quad \forall k = 2, ..., 6, \end{aligned}$$

with ${\mathbf{x}_{des}}^{(k)}$ denoting the k-th derivative of $\mathbf{x}_{des}$. The coefficients are computed by solving the linear equation with the above constraints.

The low-level controller for a quadrotor in [5] is used (line 4). The quadrotor is controlled independently by nested feedback loops. The inner attitude control loop uses onboard accelerometers and gyros to control the roll, pitch, and yaw angles, while the outer position control loop uses the estimates of position and velocity of the center of mass to control the quadrotor. If the current state is reached to the target ξ_i^*, then the Lyapunov function is updated (line 5-7). The threshold distance σ_{thres} is suitably decided so that points within a ball of radius σ_{thres} centered at $h(\mathrm{x})$ are included in the corresponding region of interest. Updating the Lyapunov function changes how the online planner behaves so that it can continuously guide the robot to satisfy the given LTL specification.



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

Fig. 3. Simulated trajectories under white Gaussian noises with different standard deviations. (a) 0.02 (b) 0.04 (c) 0.1 (d) 0.2

(a) (b)

Fig. 4. (a) Crazyflie 2.0 quadrotor. (b) Vicon MX motion capture system.

4. Experimental Results

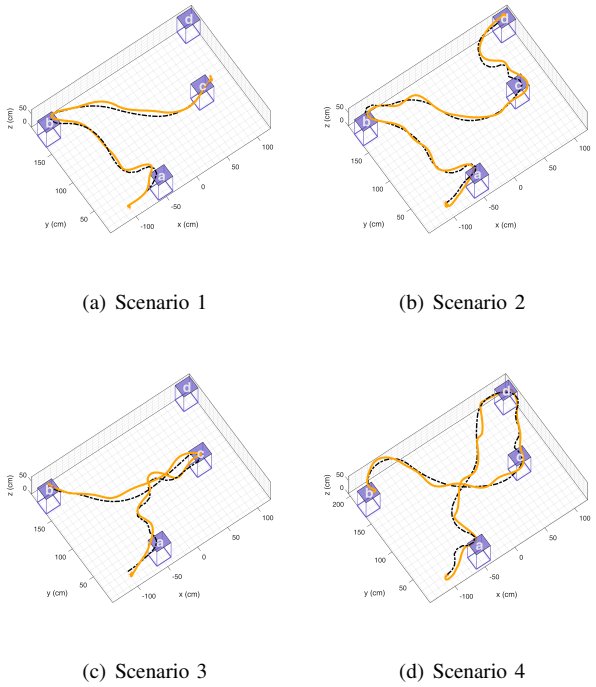
In this section, we show simulation and experiment results. We ran simulation on the simple scenario of goal reaching problem in order to show how the proposed planner behaves well under disturbances. A 2D plane quadrotor model is tested in MATLAB, and the disturbance is modeled as an white Gaussian noise affecting on the position of quadrotor. Figure 3 shows the simulation result with different standard deviation of white Gaussian noise. Notice that the proposed planner flexibly responds to external noises and guides the quadrotor to reach the target point.

Also we have conducted experiments using quadrotor robots. The robot platform used in the experiment is Crazyflie 2.0, an open source nano quadrotor platform developed by Bitcraze. We consider a quadrotor moving inside the workspace of $3.4\,m \times 2.4\,m$. Both the position and orientation of a quadrotor are measured by a Vicon MX motion capture system (Figure 4).

We consider four different scenarios with different task specifications. For Scenario 1 and 2, sequential missions are assigned and coverage missions are given in Scenario 3 and 4. LTL formulas for assigned missions are states as below.

$$\begin{aligned}
&\text{Scenario } 1 : \phi = \Diamond(a \wedge \Diamond(b \wedge (\Diamond c))) \\
&\text{Scenario } 2 : \phi = \Diamond(a \wedge \Diamond(b \wedge \Diamond(c \wedge (\Diamond d)))) \\
&\text{Scenario } 3 : \phi = \Diamond(a) \wedge \Diamond(b) \wedge \Diamond(c) \\
&\text{Scenario } 4 : \phi = \Diamond(b) \wedge \Diamond(c) \wedge \Diamond(d)
\end{aligned}$$

The results of experiments are shown in Figure 5. It shows that a quadrotor has successfully carried out assigned missions in real environments.



(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Fig. 5. Results from experiments with Crazyflie 2.0 quadrotor robots. The black dashed line is the reference trajectory from the offline planner. The orange line is the actual traveled trajectory of a quadrotor.

Acknowledgement

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT Future Planning (NRF-2017R1A2B2006136).

References

[1] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. of the IEEE International Conference on Robotics and Automation*, Apr. 2005.

[2] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, Dec. 2006.

[3] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *Proc. of the IEEE Conference on Decision and Control*, Dec. 2009.

[4] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[5] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[6] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press Cambridge, 2008.

[7] A. P. Sistla, "Safety, liveness and fairness in temporal logic," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 495–511, Sep. 1994.

[8] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, Nov. 2001.

[9] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Proc. of the IEEE International Conference on Robotics and Automation*, May 2010.

[10] E. D. Sontag, "A universalconstruction of artstein's theorem on nonlinear stabilization," *Systems & control letters*, vol. 13, no. 2, pp. 117–123, 1989.