Reachability-based Self-triggered Scheduling and Replanning of UAV Operations

Esen Yel[1], Tony X. Lin[2], and Nicola Bezzo[1,2]

[1]Department of Systems and Information Engineering
[2]Department of Electrical and Computer Engineering
University of Virginia
Email: {esenyel, txl5gd, nbezzo}@virginia.edu

***Abstract*—Modern unmanned aerial vehicles (UAVs) rely on constant periodic sensor measurements to help avoid obstacles, deal with system disturbances and correct uncertainties. However, constant checking is time and energy consuming and is often not necessary especially in situations in which the UAV can safely fly using open loop control without entering unsafe states. Thus, in this paper we focus on two main problems: i) how to predict the possible reachable states of the UAV considering disturbances and system noise and ii) when to schedule the next state monitoring and replanning time to avoid unnecessary executions. To this end, we propose a self-triggered framework in which we consider realistic UAV dynamics and reachability analysis to predict future events while scheduling replanning times to guarantee safe high performance behavior under various conditions. Finally we validate the proposed approach with simulations and experiments in which a quadrotor UAV is tasked to navigate to a goal while avoiding obstacles in the environment.**

I. Introduction

The use of unmanned aerial vehicles (UAVs) is rapidly increasing in both civilian and military applications. The technological advancements in materials, electronics, computation units, and sensors, along with the research to make UAVs more energy efficient and agile are only increasing the interest in using such systems for diverse applications such as search and rescue missions, surveillance, transportation, and photography.

In most applications, UAVs are required to "go-to-goal" and track trajectories in order to perform tasks. Since they are sharing the same airspace with other objects, they must guarantee collision avoidance despite disturbance and uncertainty due to sensor measurements and actuation. By using sensors like lidars and cameras we can detect and avoid obstacles and thus guarantee safety constraints. This operation however is time, energy, and computation consuming and is often not necessary. Let us consider the pictorial representation in Fig. 1. The UAV is initially located in an open space, far from obstacles. Depending on its actions, it could fly without checking its state for a certain amount of time before getting close to its first obstacle. Inside a cluttered environment, however, the same UAV may need to monitor its state at a much higher frequency. Thus, in this work we propose an adaptive strategy that decides the next time to monitor the UAV states and replan the trajectory to guarantee safety conditions (i.e,. something "bad" will never happen) and liveness conditions (i.e., something "good" will eventually happen).

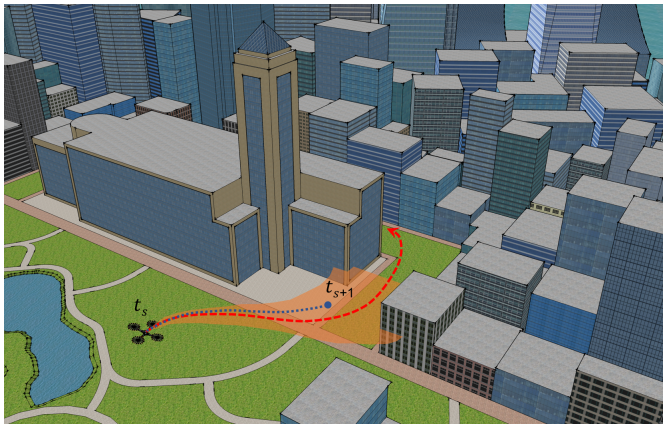


Fig. 1. Pictorial representation of the proposed problem in which a quadrotor must navigate through a cluttered environment without constant sensor checking. At t_s we compute the desired trajectory (red dashed curve). Our proposed framework computes a reachable set (the orange shaded region) to decide the next monitoring time t_{s+1} that guarantees safety and liveness constraints. The blue dotted curve in the figure represents the actual trajectory followed by the quadrotor which deviates from the desired trajectory due to wind disturbance, noise, and jitter in the system.

To this end we have created a self-triggered scheduling and replanning policy in which we leverage reachability analysis to i) predict future states of the system under different disturbance and noise conditions, ii) plan and replan the motion of the UAV, and iii) schedule the next replanning time.

The specific case study presented in this work consists of a single UAV following an obstacle free trajectory from a starting point to multiple goal positions. The aim of the robot is to follow this trajectory and reach the goal without constantly monitoring its state. However, since the robot does not monitor its state constantly, it can deviate from its planned trajectory as a result of uncertainties and disturbances leading to potential collisions. In this work, we counteract these issues by computing the next monitoring and trajectory replanning time with reachability analysis constrained by our safety and liveness requirements and finally we replan the UAV trajectory accordingly, as depicted in Fig. 1.

A. Related Work

Studies on unmanned aerial vehicles (UAVs), especially quadrotors, are gaining popularity among researchers in the

978-1-5386-3439-4/17/$31.00 ©2017 IEEE

robotics and control fields due to their potential applications. One of the most important problems that the robotics community has always been interested in solving is to safely operate UAVs in the presence of objects or other vehicles in the environment. Various computer vision, machine learning and control techniques have been leveraged to solve this problem in recent years. For example, in [1], obstacle detection with limited field of view and navigation control using potential fields are introduced. In [2], a RGB-D camera is used with the Bin-Occupancy filter to detect obstacles and a Model Predictive Control (MPC) approach is used to avoid obstacles. Reinforcement learning is used in [3] to deal with dynamic obstacles. In [4], a sampling method is used in order to create trajectories which are checked against collision at every step during the UAV navigation. In [5], an online trajectory replanning algorithm for obstacle avoidance in structurally unknown environments is performed using trajectory optimization. In all of these works, the focus is on the obstacle detection and avoidance by using high frequency periodic data acquisition and elaboration from sensors, which may be too conservative when the environment is free from obstacles.

In this work, we build a framework in which we relax constant state checking and periodic motion planning while guaranteeing both safety and liveness for a UAV that is navigating in cluttered environments while under the effects of disturbances and system uncertainties. To achieve this goal, we leverage reachability analysis. In the literature, we can find a few works in which reachable sets are used for UAV operations. In [6], continuous time Hamilton-Jacobi reachability is used to calculate the reachable sets which are then used to design the controllers to guarantee safety. For this reason, the reach-avoid reachable sets are calculated to avoid the obstacles. Similarly, in [7], Hamilton-Jacobi reachable sets are utilized to replan the motion control inputs of an autonomous helicopter under disturbances. In [8], a stochastic approach to calculate the reachable sets is proposed and applied on double integrator robot models. In [9], an aircraft is tasked with avoiding other aircrafts by using reachable sets to regularly check for collisions while sampling based methods are used to generate new trajectories. In [10], reachable sets are used to check for collisions with other UAVs and to generate control laws for collision avoidance. In [11], the authors generate reachable sets called funnel libraries to create a collision free trajectory when an airplane moves in a nominal direction in the presence of noise and disturbance. In environments with dense obstacles, in order to generate collision free paths, such funnel libraries grow very large thus becoming computationally expensive.

In this work, differently from previous related work, we use reachable sets to: i) detect possible obstacle collisions in the future and ii) adapt the checking frequency in the presence of disturbance and uncertainties in the environment and sensors. We leverage our previous work on online replanning for energy efficient UAV operations [12] in which self-triggered and self/event-triggered scheduling policies were derived to plan the best time in the future to replan the motion of a UAV while minimizing the computational energy. In this work, we contribute to the current literature on UAV motion planning by: i) providing a novel self-triggered scheduling approach using reachability analysis and ii) developing a complete framework to plan UAV operations without monitoring the state and obstacle positions constantly, while guaranteeing safety and liveness requirements. To the best of our knowledge, the proposed approach has never been investigated for UAV operations.

The rest of the paper is organized as follows: in Section II, we formally define the problem and in Section III, we define the UAV and disturbance models. In Section IV we introduce reachability analysis. Our self-triggered scheduling and replanning technique is then introduced in Section V and validated with simulations and experiments in Sections VI and VII, respectively. Finally, we draw conclusions and discuss future work in Section VIII.

II. Problem Statement

In this work we are interested in finding a policy that decides the next monitoring and replanning time for a UAV navigating in a cluttered environment. Formally the problem can be casted as follows.

Problem 1. ***Self-triggered Scheduling:*** *A UAV has the objective to visit one or more goal positions in a cluttered environment. The obstacle positions are detected using onboard sensors and a trajectory with the desired goals is generated considering the obstacle positions. Given the UAV dynamics as a function of its state $\boldsymbol{x}$, its input $\boldsymbol{u}$, and the disturbance $\boldsymbol{d}$,*

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) \tag{1}$$

we want to find a scheduling policy to determine the next time t_{s+1} to monitor the state of the UAV and replan its trajectory. t_{s+1} should be selected in such a way that the UAV will satisfy both of the following safety and liveness requirements between t_s and t_{s+1}:

1) *Safety: The UAV should be guaranteed to avoid collisions with obstacles between two consecutive replanning times t_s and t_{s+1}.*

$$\|\boldsymbol{p}(t) - \boldsymbol{p}_{oi}\| > 0, t \in [t_s, t_{s+1}], i \in [0, n]$$

where $\boldsymbol{p}(t) = \begin{bmatrix} x, y \end{bmatrix}^\mathsf{T}$ is the position of the quadrotor and $\boldsymbol{p}_{oi}$ is the position of the i^{th} obstacle in the $x-y$ plane and n is the number of obstacles in the environment.

2) *Liveness: The UAV should be guaranteed to stay within a certain proximity of the planned trajectory between two consecutive replanning times t_s and t_{s+1}.*

$$\|\boldsymbol{p}(t) - \boldsymbol{p}_\tau(t)\| < \lambda_d, t \in [t_s, t_{s+1}]$$

where $\boldsymbol{p}_\tau(t)$ is the desired position of the the quadrotor on the trajectory at time t and λ_d is the allowed deviation threshold.

Due to unforeseen disturbances and uncertainties, the UAV may deviate from its obstacle-free trajectory and reach unsafe regions (i.e. obstacles). Solving Problem 1 will allow the UAV to fly safely within a certain radius of its trajectory without checking its state until t_{s+1}. At time t_{s+1}, a new obstacle-free trajectory $\boldsymbol{x}_\tau$ to the next goal positions should be generated since it may have deviated from its desired trajectory. Thus, by solving Problem 1, it is possible to decide the optimal time to monitor the UAV's state and replan its trajectory accordingly.

III. System Models & Architecture

In this section, we explain the dynamical, noise, and disturbance models used to calculate the reachable sets and to control the quadrotor.

A. Quadrotor Dynamical Model

A quadrotor has four rotors with two rotating clockwise and two rotating counter-clockwise. The angular speed of each rotor is denoted by ω_i. The thrust and moment produced by each rotor is proportional to their angular speed [12], [13]

$$F_i = \kappa_f \omega_i^2, \quad M_i = \kappa_m \omega_i^2, \quad i = 1, \cdots, 4$$

where F_i and M_i are the thrust and the moment produced by each rotor respectively with the proportionality constant for thrust κ_f and for moment κ_m. For a quadrotor with arm length d, the net thrust and moments generated on the quadrotor are given by:

$$\begin{bmatrix} F \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \kappa_f & \kappa_f & \kappa_f & \kappa_f \\ 0 & d\kappa_f & 0 & -d\kappa_f \\ -d\kappa_f & 0 & d\kappa_f & 0 \\ \kappa_m & -\kappa_m & \kappa_m & -\kappa_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}$$

The state vector of the quadrotor is:

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{p}_q^\mathsf{T} & \phi & \theta & \psi & v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^\mathsf{T}$$

where $\boldsymbol{p}_q = [x\ y\ z]^\mathsf{T}$ is the world frame position, v_x, v_y and v_z are the world frame velocities, ϕ, θ and ψ are the roll, pitch and yaw Euler angles and ω_x, ω_y and ω_z are the body frame angular velocities.

The dynamics of the quadrotor are then described as follows:

$$\begin{aligned} \dot{\boldsymbol{p}}_q^\mathsf{T} &= \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \\ \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi \\ \cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} u_1 \\ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \\ \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} &= \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \end{aligned} \tag{2}$$

B. High-Level Motion Planning Model

In this work, we use the following high-level motion planning model for the quadrotor:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}(\boldsymbol{u}(t) + \eta_u) + \boldsymbol{G}\boldsymbol{d}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \eta_x \end{aligned} \tag{3}$$

where $\boldsymbol{x} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^\mathsf{T}$ is the state, $\boldsymbol{y}$ is the sensor measurement vector with sensor noise vector η_x. $\boldsymbol{u}$ is the input acceleration with input noise η_u, and $\boldsymbol{d} = \begin{bmatrix} w_x & w_y \end{bmatrix}^\mathsf{T}$ is the disturbance. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{G}$ and $\boldsymbol{C}$ matrices are given by:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ k_x & 0 \\ 0 & k_y \end{bmatrix} \quad \boldsymbol{C} = \boldsymbol{I}$$

where k_x and k_y are the drag parameters in x and y direction respectively and $\boldsymbol{I}$ is a 4×4 identity matrix.

This model assumes that the quadrotor is hovering at a constant z level with a zero yaw constraint. The block diagram in Fig. 2 shows the relationship between the complete quadrotor dynamics and the high level motion planner model.

C. Position, Low Level, and Attitude Controls

Following the architecture in Fig. 2, the position and low-level controllers (implemented as a series of PID loops after linearizing (2), [13]) generate the necessary angle inputs to the attitude control in order to follow the desired trajectory $\boldsymbol{x}_\tau$. In this work the quadrotor does not monitor constantly its state. During the phases in which the quadrotor is flying without checking its sensors, an open loop control is used to generate the necessary acceleration and angle inputs to make it reach the next state in its desired trajectory, hence assuming that it is following its trajectory perfectly. The attitude controller along with the motor dynamics calculate the thrust and moment values that each rotor should have. Finally the rigid body dynamics generate the response of the quadrotor to these thrust and moment values in terms of acceleration. The position and low level controller errors are represented as actuator noise in the high level motion planning model (3). In the following section, we explain in more details the noise and disturbance models for this system.

D. Noise and Disturbance Model

When a quadrotor is flying, there are many factors that may cause it to deviate from its planned behavior. These factors can be internal like sensor or actuator noises, as well as external like wind disturbance. In order to guarantee safety these factors should be taken into consideration when planning the motion of the UAV. In this work, we assume that wind disturbance, actuator noise, and state uncertainty due to sensor noise are all bounded by ellipsoids ϵ. The state uncertainty caused by the sensor measurement noise is represented by $\eta_x \in \epsilon(0, N_x)$ which represents an ellipsoid centered in the origin and bounded by a shape matrix N_x. The source of the input uncertainties are the actuator and process noise

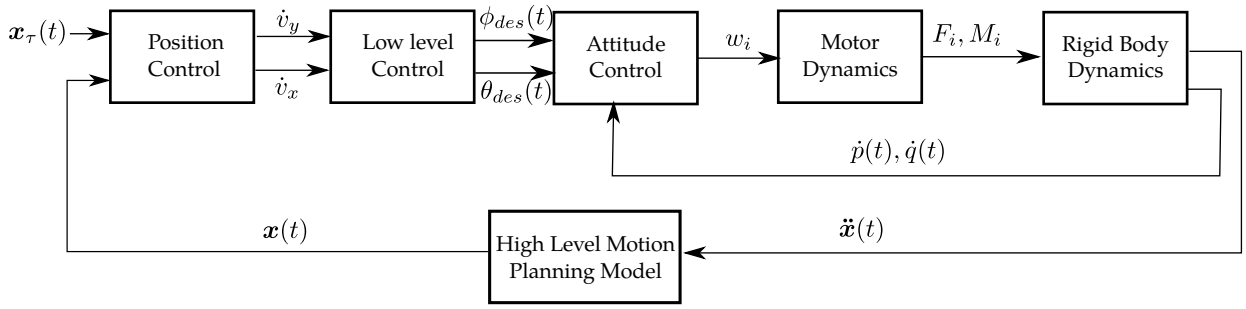


Fig. 2. Quadrotor control diagram showing the relationship between the double integrator model for motion planning and the nonlinear model for the quadrotor dynamics.

resulting from the low-level controllers and the mechanical uncertainties from the rotors, gears, and propellers. Wind disturbance is also represented as an input uncertainty during reachability analysis and the planning phases. Both actuator and disturbance uncertainties are captured by input uncertainty $\eta_u \in \epsilon(0, N_a + N_d)$ where N_a is the bound on the actuator noise and N_d is the bound on the disturbances. Noises and disturbances are assumed to be uniformly distributed over the corresponding ellipsoids.

IV. Reachability Analysis

A. Reachable Sets and Tubes

A *reachable set* or *reach set* of a system is defined as the set of states such that for each of these states there exists an input which drives the initial state to this state over a certain time window [14]. A reachable set computed at time t_0 for a future time t_f and represented by $R(\boldsymbol{x}_0, \boldsymbol{u}(t), t_f)$ is an ellipsoid ϵ that contains all the states $\boldsymbol{x}$ reachable at a future time $t_f > t_0$ where the initial set $\boldsymbol{x}_0 \in \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ is bounded by an ellipsoid with center $\boldsymbol{x}_0$ and shape matrix $\boldsymbol{X}_0$ and the input $\boldsymbol{u}(t) \in \epsilon(\boldsymbol{u}(t), \boldsymbol{U})$ is bounded by an ellipsoid with center $\boldsymbol{u}(t)$ and shape matrix $\boldsymbol{U}$. The actual state $\boldsymbol{x}(t_f)$ and the estimated state $\tilde{\boldsymbol{x}}(t_f)$ will lie inside the reachable ellipsoid. The shape matrix $\boldsymbol{X}_0$ of the initial state ellipsoid depends on the state uncertainty η_x whereas the shape matrix of the input ellipsoid $\boldsymbol{U}$ depends on the input uncertainty η_u. $R_p(\boldsymbol{x}_0, \boldsymbol{u}(t), t_f)$ is the projection of the reachable sets on the position space and $R_v(\boldsymbol{x}_0, \boldsymbol{u}(t), t_f)$ is the projection on the velocity space. A reachable tube $R(\boldsymbol{x}_0, \boldsymbol{u}(t), [0, T])$ is the set of all reachable sets over the time interval $\Delta T = [0, T]$.

The external bound for the reach set at time t starting from an initial time t_0 is calculated based on the initial state ellipsoid, the plant model, and the input ellipsoid, as follows:

$$R^+(t, t_0, \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)) = \boldsymbol{\Phi}(t, t_0)\epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0) \oplus \int_{t_0}^{t} \boldsymbol{\Phi}(t, \zeta)\boldsymbol{B}(\zeta)\epsilon(\boldsymbol{u}(\zeta), \boldsymbol{U})d\zeta$$

where $\boldsymbol{\Phi}(t, t_0) = e^{\boldsymbol{A}(t-t_0)}$.

B. Reachable Tubes for a Trajectory Tracking Quadrotor

In this work, we use reachability analysis to calculate the reachable sets of the states of a quadrotor tracking a trajectory $\boldsymbol{x}_\tau(t_s : t_{s+1})$ generated over the interval $\Delta t_s = [t_s, t_{s+1}]$ where t_s is the current scheduled time and t_{s+1} is the next scheduled monitoring time, initially set to $t_s + T$ before the rescheduling operation. We generate minimum acceleration trajectories [15] which compute desired positions and velocities at each time step along the trajectory. A PD controller is used to find the acceleration input to track the trajectory as follows:

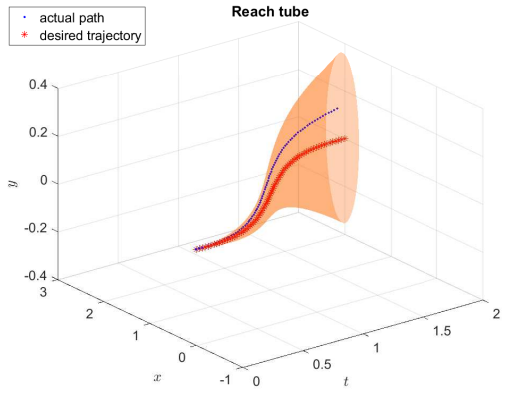
$$\boldsymbol{u}(t) = K_P(\boldsymbol{x}_\tau(t) - \boldsymbol{x}(t)) + K_D(\dot{\boldsymbol{x}}_\tau(t) - \dot{\boldsymbol{x}}(t))$$

where $t \in [t_s, t_{s+1}]$. Since the robot does not monitor its state constantly but just at the scheduled monitoring times, in order to calculate the inputs required to follow this trajectory over the interval Δt_s, an online simulation is run as if the quadrotor is tracking the trajectory perfectly. As a result, a set of control inputs $\boldsymbol{u}(t_s : t_{s+1})$ is constructed.

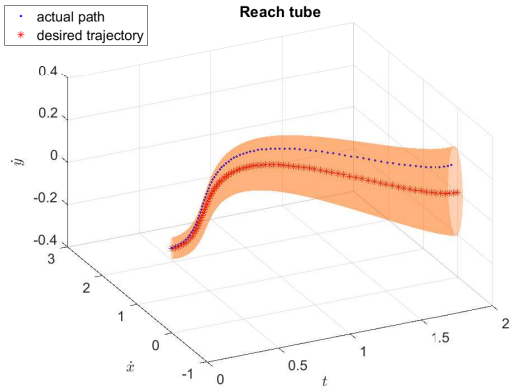
Reachable sets are calculated to capture the motion of the quadrotor tracking a trajectory without monitoring its state continuously. To this end, the set of inputs generated during the online simulation are applied in open loop to the quadrotor during Δt_s. The same set of inputs, considering the noise resulting from disturbances and actuator uncertainties, are used to calculate the reachable sets. In Fig. 3, a position and velocity reachable tube calculated from $t_s = 0$ to $t_{s+1} = 2$s is shown for a quadrotor following a straight line trajectory from $(0, 0)$m to $(2, 0)$m with sensor measurement noise $\eta_x = 0.001$ on the initial state and input noise $\eta_u = 0.054$. Fig. 3(a) shows the position reachable tubes while Fig. 3(b) shows the velocity reachable tubes. The blue dotted curve shows the path of the quadrotor whereas the red star curve shows the desired trajectory. The desired trajectory and the actual path are different due to the presence of wind disturbance $\boldsymbol{d} = [0.1, 0.1]$m/s however the actual path is contained inside the reachable tubes, since system and sensor uncertainties as well as disturbances are considered when calculating such reachable tubes.

V. Self-Triggered Scheduling and Replanning

In this section we describe our framework to solve the self-triggered scheduling and replanning problems using reachable sets. The architecture followed in this work is shown in Fig. 4. The overall architecture is composed of self-triggered



(a) Position reachable tubes.



(b) Velocity reachable tubes.

Fig. 3. The reachable tubes for the position of the quadrotor following a straight line trajectory over a time interval [0,2]s for an initial position $x_p = [0, 0]^T$. The blue dotted curve is the actual trajectory followed by the robot while the red star curve represents the desired trajectory.

scheduling, replanning, and control. First, a trajectory from the current state of the quadrotor to a goal position $\boldsymbol{x}_g$ is generated. The controller described in Section IV-B generates the necessary input values $\boldsymbol{u}(t_s : t_s + T)$ to follow this trajectory over a time horizon T. Our self-triggered scheduling policy uses reachable sets to compute the next replanning time t_{s+1}. Reachable tubes are calculated using the state of the robot observed from some sensor measurements $\boldsymbol{y}$ at t_s, the sensor noises η_x, the set of planned inputs $\boldsymbol{u}(t_s : t_s + T)$ to be applied to the quadrotor between t_s and a finite fixed horizon T, and the input uncertainty η_u resulting from the low-level controllers and the disturbance in the environment. The input sequence $\boldsymbol{u}(t_s : t_{s+1})$ is then applied to the quadrotor until time t_{s+1}. The state of the system is finally monitored through sensor measurements and a new trajectory to the goal is replanned at time $(t_{s+1} - t_r)$, in which we have introduced t_r as the time needed to calculate the reachable tube and for replanning.

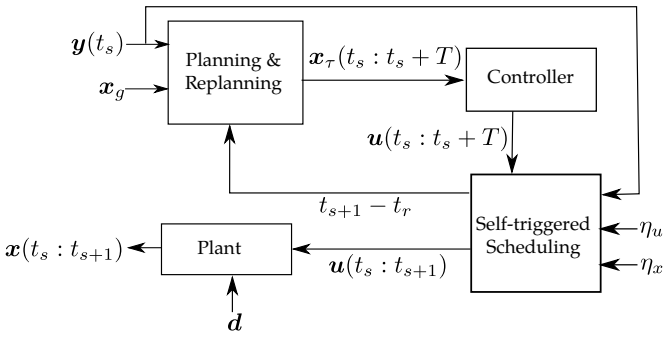


Fig. 4. Overall architecture of our self-triggered scheduling and replanning framework.

A. *Self-triggered Scheduling*

Our novel self-triggered scheduling policy consists of deciding the next replanning time t_{s+1} such that safety and liveness of the quadrotor are guaranteed until t_{s+1}. The safety requirement is violated when there is a possible collision with an obstacle. The set of obstacle positions is shown by $\boldsymbol{o} \in R^{2\times n}$ where $\boldsymbol{p}_{oi} \in \boldsymbol{o}, i \in [0, n]$ and n is the number of obstacles in the environment. The predicted first time in which a collision may occur, t_c, is calculated using a reachable tube as follows:

$$t_c = \min(t_k | R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [t_s, t_s + T]) \cap \boldsymbol{o} \neq \emptyset) \quad (4)$$

The liveness requirement is violated when the UAV position deviates from the desired trajectory more than a certain threshold λ_d. The predicted first time in which the UAV can deviate from the trajectory is shown by t_d and calculated as:

$$t_d = \min(t_k | \| R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [t_s, t_s{+}T]) - \boldsymbol{p}_\tau(t_k) \| > \lambda_d) \quad (5)$$

where $\boldsymbol{p}_\tau(t_k)$ is the desired position of the UAV along the computed trajectory at time $t_k \in \Delta t_s$.

If there exists the possibility to collide with an obstacle at time $t_c < T$ and/or to deviate from the desired trajectory at time $t_d < T$, the UAV needs to check its state at one of these times, whichever is minimum. If the UAV cannot collide with any obstacle or deviate from the trajectory more than the allowed threshold, the next replanning time is the initial time horizon T.

$$t_{s+1} = \begin{cases} \min(t_c, t_d), & \text{if } t_c < T \text{ or } t_d < T \\ T, & \text{otherwise} \end{cases} \quad (6)$$

Lemma 1. *Given t_s and t_{s+1}, the UAV is guaranteed to stay within λ_d proximity of its planned trajectory and not to collide with any obstacle.*

Proof. By definition, $R(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [0, T])$ is the set of all states $\boldsymbol{x}$, such that there exists an input $\boldsymbol{u} \in \epsilon(\boldsymbol{u}(t_k), \boldsymbol{U})$ and an initial state $\boldsymbol{x}_0 \in \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ which steers the UAV from $\boldsymbol{x}_0$ to $\boldsymbol{x}$ in time t_k [14]. $R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [0, T])$ is the projection of $R(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [0, T])$ onto the position space.

Considering a time value t^* between t_s and t_{s+1}, $t_s < t^* < t_c$ and $t_s < t^* < t_d$, by the definitions of t_c in (4) and t_d in (5),

$$R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [t_s, t^* + t_s]) \cap \boldsymbol{o} = \emptyset$$

$$\|R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [t_s, t^* + t_s]) - \boldsymbol{p}_\tau(t_k)\| < \lambda_d$$

which proves that for a $t_s < t^* < t_{s+1}$, there doesn't exist an input $\boldsymbol{u}(t^*) \in \epsilon(\boldsymbol{u}(t^*), \boldsymbol{U})$ and an initial state $\boldsymbol{x}_0 \in \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ which steers the system from $\boldsymbol{x}_0$ to a state $\boldsymbol{x}$ such that the position part of the state intersects with an obstacle or deviates from the planned trajectory more than the permitted threshold λ_d.

□

B. Trajectory Replanning

The self-triggered scheduling procedure calculates the next state monitoring and replanning time. In self-triggered replanning, the path of the UAV is replanned from its state to the next goal point. In this work, we assume that the positions of all obstacles can be detected by sensors and that they do not change over the time interval that the sensors are not checked. The self-triggered scheduling and replanning procedures are summarized in Algorithm 1 where δt is the step time for the controller. At the time in which the quadrotor checks its state, $t = t_{s+1} - t_r$, $\boldsymbol{y}(t)$ is monitored from the sensors and a minimum acceleration trajectory is generated [15]. By running an online simulation, the set of inputs are calculated based on the trajectory that the quadrotor is supposed to follow for a time horizon T as explained in Section IV-B. These input values are used to calculate the next replanning time. When the distance between the final goal position $\boldsymbol{p}_g$ and the position of the quadrotor obtained from the sensors measurements $\boldsymbol{y}_p(t_s)$ is smaller than a certain threshold λ_g, the quadrotor is considered as reached the goal.

Algorithm 1 Self-triggered Scheduling & Trajectory Replanning

```
Initialize the system: t = 0, x(0) = x_0, y_p(0) = p(0), s = 0
Calculate the initial trajectory x_τ(t) for t ∈ [t_s, t_s + T]
Calculate the set of inputs u(t_s : t_s + T)]
Calculate the initial reachable tube R(x_0, u(t), t ∈ [t_s, t_s + T])
Calculate t_{s+1}
while ‖y_p(t_s) − p_g‖ > λ_g do
    Apply u(t) ∈ u(t_s : t_{s+1}) to the system defined in (2)
    t = t + δt
    if t = t_{s+1} − t_r then
        s = s + 1
        Update y_p(t_s)
        if ‖y_p(t_s) − p_g‖ > λ_g then
            Calculate the trajectory x_τ(t_s : t_s + T)
            Calculate the set of inputs u(t_s : t_s + T)
            Calculate R(x(t_s), u(t), t ∈ [t_s, t_s + T])
            Calculate t_{s+1}
        end if
    end if
end while
```

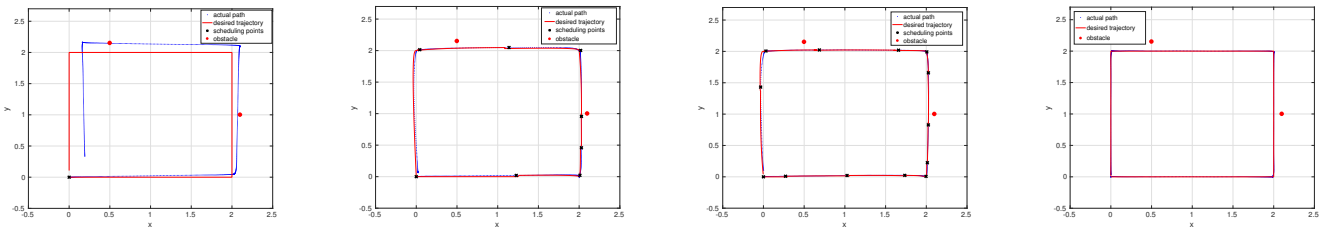
VI. Simulation Results

The case study investigated in this work is with a single UAV under disturbance, process noise, and uncertainties in sensor measurements. In our specific case, we consider a quadrotor starting from its initial position with zero velocity $\boldsymbol{x}(0) = (0, 0, 0, 0)^\mathsf{T}$ trying to visit each corner of a square path with 2m sides. The disturbance is assumed to be constant everywhere in the environment $\boldsymbol{d} = [0.01, 0.01]^\mathsf{T}$m/s. The maximum speed of the quadrotor is assumed to be $v_{max} = 0.75$m/s and λ_g is set to 0.05m. We consider a case where there are two single point obstacles in the environment close to the path at positions $\boldsymbol{p}_{o1} = (0.50, 2.15)$m and $\boldsymbol{p}_{o2} = (2.10, 1.00)$m.

In Fig. 5(a), the quadrotor generates an obstacle free trajectory to visit each corner of the square. A set of inputs is calculated to follow such trajectory and applied with an open loop controller. In the case that there are no disturbances or uncertainties, the quadrotor should be able to follow the trajectory perfectly. However, due to disturbances and system uncertainties, it deviates from its desired trajectory, and collides with both obstacles as shown in Fig. 5(a). The red continuous curve shows the desired trajectory to track while the blue dotted curve is the real path followed by the quadrotor. Both safety and liveness requirements are violated in this case since there are obstacle collisions and the robot deviates from the trajectory more than the allowed threshold which was set to 0.25m for this simulation.

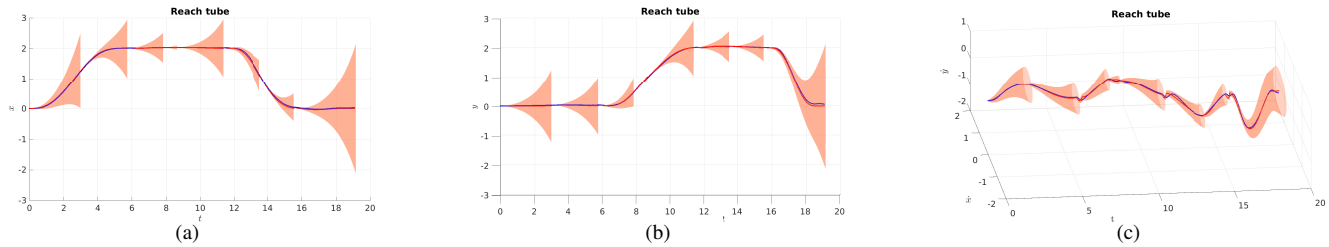
In Fig. 5(b) we show the case in which only the safety requirements need to be satisfied, which means that next replanning time $t_{s+1} = t_c$ (i.e., the quadrotor checks its state and regenerates the trajectory at times in which a collision may occur). The points in which the quadrotor checks its state are marked by 'x' symbols on the path of the quadrotor. By using our approach the quadrotor is able to avoid collisions with the obstacles. The quadrotor also starts to follow the trajectory more closely since it checks its state and corrects its path accordingly during replanning times. Fig. 6 shows the reachable tubes generated for the case presented in Fig. 5(b) when only safety requirements need to be met. As can be noticed, the reachable tube calculations are aperiodic. The next reachable tube calculations are performed whenever the current reachable tube overlaps with an obstacle position. When it gets closer to the obstacle, the quadrotor check its state more often.

In Fig. 5(c) both liveness and safety requirements are considered. In this case, the quadrotor checks its state more often than in the previous case to also meet the liveness requirements. As a result the trajectory is tracked more closely than in the previous two simulations. The quadrotor checks its state before it deviates $\lambda_d = 0.25$m from the desired trajectory, because the reachable sets bounds act as worst case scenarios increasing over time, however the actual deviation of the system may not be that large.



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

Fig. 5. Comparison between three simulations in which a quadrotor is tasked to follow a square trajectory. In (a), the quadrotor doesn't check its state, in (b) only safety requirements are considered for self-triggered scheduling, in (c), both safety and liveness requirements are considered and in (d), closed loop control is applied where the robot checks its state at 40Hz. In all cases, a red curve corresponds to the desired trajectory, a blue curve corresponds to the actual path of the robot, obstacle positions are shown by red dots, and black 'x' depict the positions where the UAV checks its state and replan its trajectory.



(a) (b) (c)

Fig. 6. The reachable tubes obtained with the self-triggered scheduling policy considering only safety constraints shown in Fig. 5(b). The reachable tubes are projected into the x axis in (a) and into y axis in (b). (c) shows the velocity reachable tubes. The reachable tube calculation is aperiodic, therefore, reachable tubes have different sizes.

In Fig. 5(d), we show the path of the quadrotor when its state is periodically checked at 40Hz and a close loop controller is applied. As can be noted, the robot is able to follow the trajectory very closely. In Table I, the number of state checks and the maximum drift is compared between these three cases. As can be seen, the maximum drift decreases from 32.94cm to 7.85cm when only the safety requirements are considered and using only 8 state checks. With the introduction of the liveness requirements, the number of state checks increased to 13 and the maximum drift decreased to 7.23cm. In close loop control, the robot checks its state 836 times during its motion, and the maximum drift is measured as 4.87cm. These results show that, with our self-triggered scheduling and replanning procedure, the safety and liveness requirements are guaranteed without checking for the state and obstacle positions constantly.

TABLE I
COMPARISON BETWEEN NUMBER OF STATE CHECKS AND MAXIMUM DRIFT.

	Number of State checks	Max. drift (cm)
No checking	1	32.94
Safety	8	7.85
Safety and liveness	13	7.23
Closed loop	836	4.87

VII. EXPERIMENTS

In order to validate the proposed self-triggered scheduling and replanning approach, we implemented a series of experiments with an AscTec Hummingbird quadrotor UAV. The architecture followed in this work is shown in Fig. 7. Here, reachable sets were calculated using the Matlab ellipsoidal toolbox [14]. The quadrotor control was achieved using our ROS framework and communication was bridged using the Robotics System Toolbox. The self-triggered scheduling and replanning scheme were implemented in Matlab while acceleration inputs were sent to the quadrotor through ROS. In order to monitor the state of the quadrotor at each scheduling time, a Vicon motion capture system was used.

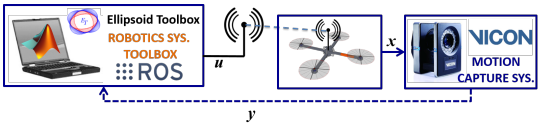


Fig. 7. Architecture of the experimental setup.

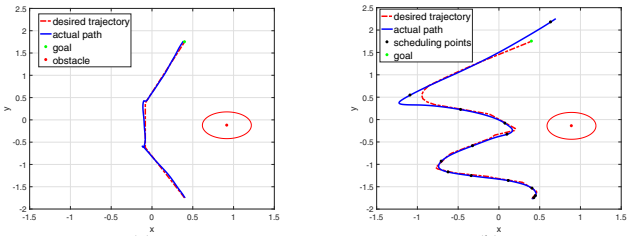
The UAV starts its motion at $(0.4, -1.75, 1)$m and is tasked to navigate to a goal position at $(0.4, 1.75, 1)$m. The objective of the quadrotor is to follow the trajectory avoiding the obstacle without checking its state constantly. Due to the limited workspace, in this specific experiment we demonstrate only the safety requirement case with $\lambda_g = 0.5$. The quadrotor initially creates a trajectory and decides next monitoring time using reachable tubes as described in the previous sections. To minimize computation times, the number of reachable sets calculated in each tube was decreased from 200 to 20.

In Fig. 8(a), we show a sequence of snapshots of the quadrotor going to the goal while using a periodic checking rate of 40Hz. In Fig. 9(a), the obstacle-free trajectory generated

is shown with a red dashed curve and its actual path with a blue colored curve. As can be noted, the actual path of the quadrotor is very close to the desired one due to the frequent checking of the state. In Fig. 8(b), we show a sequence of snapshots of the quadrotor going to the goal while performing self triggered scheduling and replanning and the planned and actual paths of the quadrotor during its motion are shown in Fig. 9(b). During runtime, the quadrotor checks its state and replans its trajectory 13 times only. As can be noted, it deviates from its trajectory by nearly 0.5m, but it does not collide with the obstacle. Similar to our simulations, most of the state checks happens in the close neighborhood of the obstacle since the reachable tubes interesect more frequently with the obstacle (i.e., the probability of collision is higher near the obstacle). It should be noted that the desired trajectory of the robot changes over time according to its current position, therefore it is different than its initial trajectory. These two experiments demonstrate that constant checking of state, although while capable of more accurate tracking, is not necessary and safety constraints can be guaranteed by using our proposed framework.

(a) (b)

Fig. 8. Obstacle avoidance experiments for (a) constant checking and tracking of the trajectory and (b) self-triggered scheduling and replanning with only safety constraint.



(a) (b)

Fig. 9. Results for the experiments shown in Fig. 8. (a) Constant checking and tracking and (b) self-triggered scheduling and replanning with only safety constraint

VIII. Conclusion and Future Work

In this work, we have presented a self-triggered scheduling scheme for UAV navigation in cluttered environments under external disturbances and sensor and system uncertainties. Our framework uses reachability analysis and a self-triggering policy to schedule replanning times while guaranteeing safety and liveness properties. The proposed approach was validated both with simulations and experiments on a quadrotor motion planning case study. The proposed framework is general for any type of systems and thus could also be used for ground and underwater vehicles, as well as more complicated cyber-physical systems.

In our future work we plan to consider more realistic scenarios in which the internal state of the system is monitored periodically and lidar and camera data are checked aperiodically as discussed in this paper. Further, we plan to consider a similar framework to minimize energy consumption and to adapt the speed of the UAV based on its state in relation to environmental conditions. We also plan to consider more complicated scenarios with dynamic obstacles.

Acknowledgments

This work is based on research sponsored by DARPA under agreement number FA8750-12-2-0247 and ONR under agreement number N000141712012.

References

[1] S. Roelofsen, A. Martinoli, and D. Gillet, "3d collision avoidance algorithm for unmanned aerial vehicles with limited field of view constraints," in *IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 2555–2560.

[2] M. Odelga, P. Stegagno, and H. H. Blthoff, "Obstacle detection, tracking and avoidance for a teleoperated uav," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2984–2990.

[3] A. Faust, H. T. Chiang, N. Rackley, and L. Tapia, "Avoiding moving obstacles with stochastic hybrid dynamics using pearl: Preference appraisal reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 484–490.

[4] Y. Lin and S. Saripalli, "Sampling based collision avoidance for uavs," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 1353–1358.

[5] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5332–5339.

[6] J. Ding, J. H. Gillula, H. Huang, M. P. Vitus, W. Zhang, and C. J. Tomlin, "Hybrid systems in robotics," *IEEE Robotics Automation Magazine*, vol. 18, no. 3, pp. 33–43, Sept 2011.

[7] J. Ding, E. Li, H. Huang, and C. J. Tomlin, "Reachability-based synthesis of feedback policies for motion planning under bounded disturbances," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2160–2165.

[8] A. P. Vinod, B. Homchaudhuri, and M. M. K. Oishi, "Forward stochastic reachability analysis for uncontrolled linear systems using fourier transforms," *CoRR*, vol. abs/1610.04550, 2016. [Online]. Available: http://arxiv.org/abs/1610.04550

[9] Y. Lin and S. Saripalli, "Collision avoidance for uavs using reachable sets," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 226–235.

[10] Y. Zhou, A. Raghavan, and J. S. Baras, "Time varying control set design for uav collision avoidance using reachable tubes," in *IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 6857–6862.

[11] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *arXiv preprint arXiv:1601.04037*, 2016.

[12] N. Bezzo, K. Mohta, C. Nowzari, I. Lee, V. Kumar, and G. Pappas, "Online planning for energy-efficient and disturbance-aware uav operations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5027–5033.

[13] 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, Sept 2010.

[14] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal toolbox (et)," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec 2006, pp. 1498–1503.

[15] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.