Journal of Intelligent & Robotic Systems
https://doi.org/10.1007/s10846-020-01192-2

Computation-Aware Adaptive Planning and Scheduling for Safe Unmanned Airborne Operations

Esen Yel[1] · Tony X. Lin[2] · Nicola Bezzo[1,3]

Received: 1 October 2019 / Accepted: 26 March 2020
© Springer Nature B.V. 2020

Abstract
Modern unmanned aerial vehicles (UAVs) rely on high-frequency periodic sensor measurements in order to safely operate in cluttered environments with both static and dynamic obstacles. However, periodic sensor checking operations are time and computation consuming and they are often not needed, especially in situations where the UAV can operate without violating the safety constraint (e.g., in uncluttered free space). In this paper, we introduce a computation-aware framework that limits sensor checking and replanning operations to instances in which such operations could be necessary. To this end, we propose an approach that utilizes reachability analysis to capture the future states of a UAV operating under the effects of noise and disturbance and performs self-triggered scheduling for sensor monitoring and replanning operations while guaranteeing safety. The replanning operation is further relaxed by performing an online reachable tube shrinking. This approach is supplemented with an online speed adaptation policy based on the curvature of the planned path to minimize deviation from the desired trajectory due to complex system dynamics and controller limitations. The proposed technique is validated with both simulations and experiments focusing on a quadrotor motion planning operation in environments consisting of both static and dynamic obstacles.

Keywords Self-triggered control · Reachability analysis · Scheduling and replanning · Unmanned aerial vehicles

1 Introduction

Unmanned aerial vehicles (UAVs) have been gaining a lot of attention and popularity in the last decade thanks to the myriad of operations in which they can be deployed ranging both civilian and military applications. Their technological advancements which include more agile platforms, increasingly precise sensors and actuators, have only augmented their popularity and deployment in our daily lives. Although these vehicles have seen an incremental technological improvement, a big challenge still remains on how to guarantee safety during their missions while the airspace is shared with other objects (i.e., other aerial vehicles and obstacles).

Typically, a vehicle *periodically* monitors 1) its pose and configuration using pose sensors like IMU, GPS, speedometers, motion capture systems and 2) its distance to the obstacles using range sensors like lidar, radar, sonar, and IR sensors, in order to avoid collisions under external disturbances and noises. This information is then used to plan and replan the vehicle motion accordingly to guarantee the desired objective, in a robust, optimal, and safe fashion. However, periodic sensor checking and planning brings computational burden to the system, and can be relaxed if the future states of the system under different uncertainties can be predicted reliably. For example, let's consider the pictorial representation in Fig. 1: when a UAV moves in

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-020-01192-2) contains supplementary material, which is available to authorized users.

✉ Esen Yel
esenyel@virginia.edu

Tony X. Lin
tlin339@gatech.edu

Nicola Bezzo
nbezzo@virginia.edu

1 Autonomous Mobile Robots Lab, Link Lab, Department of Engineering Systems and Environment, University of Virginia, Charlottesville, VA, USA

2 School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA

3 Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, USA

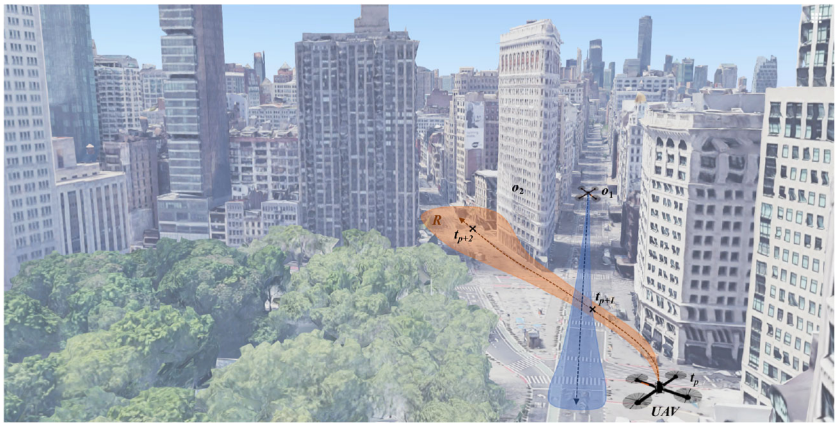


Fig. 1 Pictorial representation of the problem presented in this paper: a quadrotor in navigating in a cluttered environment in which it needs to avoid collision with both static and dynamic obstacles $\{o_1, o_2\}$ while minimizing periodic sensor checking under uncertainties like wind disturbances, noise, and jitter. R represents the reachable region of the UAV with t_{p+1} the first predicted time that a collision with o_1 may occur and t_{p+2} the time of a second possible collision with o_2

an obstacle-free environment, it may act without checking its sensors for a longer time and avoid collisions, whereas when the vehicle gets close to the obstacles or to the other aerial vehicles, it needs to monitor its sensors more often. Furthermore, whereas the vehicle can move faster in obstacle-free areas, it would need to slow down to closely follow winding trajectories between obstacles.

To deal with these issues, we propose an online adaptive framework which allows a UAV to limit its sensor monitoring times to the instances in which it is necessary, thus minimizing computation while guaranteeing safety (i.e., something "bad" will never happen) and liveness (i.e., something "good" will eventually happen) . The proposed approach addresses the following challenges:

1. how to minimize sensor monitoring and replanning operations while satisfying safety and liveness conditions in cluttered and dynamic environments;
2. how to plan and replan a UAV operation, including adapting its speed, while solving the previous challenge.

Our proposed approach leverages reachability analysis to predict the future states of the system, utilizes self-triggered scheduling methods to compute the next sensor monitoring and replanning time, and introduces a replanning approach to adapt the speed of the UAV that is navigating in cluttered and unknown environments with static and dynamic obstacles under the effect of noises and external disturbances.

The reminder of the paper is organized as follows: Section 2 provides a brief background on related work. Section 3 introduces the notation used throughout this paper and formally defines the problem. The UAV motion, noise and disturbance models are defined in Section 4. A general framework of the proposed approach is introduced in Section 5 and detailed discussion about the approach along with extensive simulation and experimental results for static and dynamic environments are provided in Sections 6 and 7 respectively. Finally, conclusions and future work are discussed in Section 8.

2 Related Work

This work introduces a reachability-based self-triggered scheduling and motion replanning approach in environments cluttered with static and dynamic obstacles. In this section, we review motion planning approaches, particularly the ones focusing on obstacle avoidance. Then we provide a discussion about reachability analysis techniques utilized for various kinds of UAV operations and we state the contributions of this work.

2.1 Motion Planning

In the literature of mobile robotics, motion planning for collision avoidance has been studied heavily to enable safe and persistent autonomous operations in known and unknown environments. Sampling-based methods are commonly used in robotics operations to generate collision free paths. Rapidly-exploring Random Tree (RRT) based approaches, such as RRT^X [21] and RRT* [5, 22] have been leveraged for UAV navigation tasks. In [11], a random waypoint sampling method is proposed to navigate a vehicle in environments with dynamic obstacles by predicting its future states. Other widely used motion planning approaches for collision-free UAV navigation are vector field-based methods. For example, potential functions have been widely utilized to make autonomous mobile robots move only in obstacle-free regions [25]. Similarly, in [23, 24], vector based navigation is used while taking in consideration the field of view limitations of the range sensors for obstacle detection. In [28], the traditional artificial potential field method is improved to deal with the target unreachability problem while treating other vehicles as obstacles in cooperative UAV operations. Recently, trajectory optimization [20], model predictive control [19] and reinforcement learning [8] techniques have also gained popularity for UAV navigation in dynamic environments. For the sake of safety guarantees, [13] introduces the concept of Safe Flight Corridor (SFC) and generate motion plans inside these corridors. For smoothness, [1, 30] take curvature analysis into consideration. In all of these works, obstacle detection is performed using different kinds of sensors such as RGB-D cameras [19, 25, 26], fish-eye cameras [23], low cost sensors [9] or lidars [13] all running periodically at high frequency, which brings computational burden to the systems. However, when there are no obstacles present in the environment, this high-frequency periodic sensor data acquisition and elaboration is too conservative and can be relaxed while maintaining the same safety constraints (i.e., obstacle collision). In the work presented in this paper we elaborate this idea which to the best of our knowledge is not well covered in the robotic literature and propose methods to relax and adapt sensor checking.

2.2 Reachability Analysis

When dealing with safety, a well known theory is reachability analysis which consists in predicting the future states covered by a system over a certain future time horizon under uncertainties. Hamilton-Jacobi reachability has been widely used to compute the reachable sets of hybrid systems and to provide safety guarantees for optimal system trajectories [6]. Forward stochastic reachability analysis is another method that leverages Fourier transforms for LTI systems with uncertain dynamics [29]. In [16], these stochastic forward reachable sets are utilized in a roadmap-based approach to avoid moving obstacles. In [12], an aircraft is tasked to avoid other aircrafts by detecting collisions using reachable sets and sampling based methods are used to generate new trajectories to avoid obstacles. In [34], reachable sets are used to detect collisions with other UAVs and to generate control laws for collision avoidance. In [27], a region where the system is guaranteed to stay while following a nominal trajectory is characterized offline and utilized to plan safety-guaranteed trajectories at run-time. In [15], funnels are created for motion primitives, which are analogous to reachable tubes, and the motion of the UAV is planned in real time, using the funnel library. There are two main differences between these previous works and our proposed approach: i) instead of using reachable sets to design safe trajectories, we utilize them to guarantee the safety of the planned trajectories by detecting possible obstacle collisions under disturbances and uncertainty and ii) we minimize unnecessary sensor monitoring and replanning operations while satisfying safety and liveness conditions. These ideas build on our previous work [4], in which a self/event-triggered approach was deployed to schedule the best time to replan a UAV operation considering energy constraints. Similarly, in [32], we proposed a self-triggered scheduling policy to minimize computation while guaranteeing safety of a UAV navigating in open-loop without state feedback. In [33], we extended this work by introducing risk-based replanning and self-triggered speed adaptation policies and in [31] we considered dynamic obstacles. This paper extends our previous studies and presents a complete and unison framework for computation-aware adaptive planning and scheduling of unmanned aerial vehicles, considering different scenarios within static and dynamic environments.

2.3 Contributions

The contribution of this work is fivefold: 1) We develop a reachability-based self-triggered scheduling approach to decide next sensor monitoring and replanning operations and provide safety and liveness guarantees; 2) We propose a method to update the reachable tubes based on the observed state of the system to further minimize replanning operations while guaranteeing safety; 3) We present a speed adaptation approach that considers the curvature of the trajectory to minimize drift; 4) We propose a reachability-based self/event-triggered approach to deal with dynamic environments and minimize the number of sensor checking and replanning operations while guaranteeing inter-agent collision avoidance; and 5) Our final contribution is on the validation using realistic simulators and by experiments with real aerial vehicles in both static and dynamic environments.

To the best of our knowledge, the aforementioned challenges in motion planning have not been extensively covered by the current literature. This work provides a complete and unison framework to save computational resources in autonomous operations in static and dynamic environments.

3 Problem Formulation

Through this paper, bold lower case italic letters (e.g., $\boldsymbol{q}$) are used to denote vectors and bold upper case italic letters (e.g, $\boldsymbol{A}$) are used to denote matrices. $\|.\|$ represents the Euclidean norm.

The state vector of the quadrotor is:

$$\boldsymbol{q} = [\boldsymbol{p}_q^\mathsf{T} \Phi \theta \psi v_x v_y v_z \omega_x \omega_y \omega_z]^\mathsf{T}$$

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. $\boldsymbol{x} \in \mathbb{R}^4$ refers to the position and velocity part of the state in x-y direction and $\boldsymbol{p} \in \mathbb{R}^2$ denotes only the position in x-y direction [18].

In this work, we assume a standard UAV equipped with sensors capable of observing its angular position and velocity. It is also assumed that the UAV can observe its position and velocity in the $x - y$ plane (i.e. $\boldsymbol{x}$) via pose sensor at scheduled times. The position of an obstacle in the $x - y$ plane is denoted by $\boldsymbol{o} \in \mathbb{R}^2$ with a subscript indicating the index of the obstacle and it is assumed to be observed via an on-board range sensor at scheduled times. We neglect the third dimension because we assume that the robot moves on a plane at desired z level, however, the proposed approach is still valid when the z position of the robot varies.

The problems that we address are formally defined as follows:

Problem 1: ***Self-triggered Replanning*****:** A UAV has an objective to visit one or more goal locations in a cluttered environment. The positions of the obstacles are detected using an on-board range sensor and a trajectory to the desired goals is generated online considering obstacle locations. The UAV dynamics are a function of its state $\boldsymbol{q}$, input $\boldsymbol{u}$, and disturbance $\boldsymbol{d}$,

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

We consider two cases within the scope of this problem. In the first case the UAV does not monitor its pose and range sensors and moves with open loop controller between planning times. Since monitoring the pose sensors is not computationally as expensive as range sensors, we introduce also a second case in which the vehicle monitors its pose sensor periodically and moves with a closed loop controller between replanning times.

Case 1: *Open Loop Scheduling and Replanning*: Find a policy to schedule the next pose and range sensor monitoring and replanning time t_{p+1}, while the UAV is operating with a precomputed sequence of inputs in open loop guaranteeing both safety and liveness constraints between replanning operations as mathematically represented in Eqs. 2 and 3.

Case 2: *Closed Loop Scheduling and Replanning*: Find a policy to schedule next range sensor monitoring and replanning time t_{p+1}, while the UAV is checking periodically its pose sensor and operating in closed loop guaranteeing both safety and liveness constraints between replanning operations as mathematically represented in Eqs. 2 and 3.

In both cases, the next replanning time is determined such that the following safety and liveness requirements between replanning operations are met:

1. *Safety Constraint:* Collisions with obstacles should be avoided between two consecutive replanning times, or mathematically:
$$\|\boldsymbol{p}(t) - \boldsymbol{o}_i(t)\| > r_o^i, \forall t \in [t_p, t_{p+1}], \forall i \in \{1, \cdots, n_o\} \tag{2}$$
in which $\boldsymbol{p}(t) = [x(t), y(t)]^\mathsf{T}$ is the position of the quadrotor and $\boldsymbol{o}_i(t) = \left[x_o^i(t), y_o^i(t)\right]^\mathsf{T}$ is the position of the i^{th} obstacle in the $x - y$ plane at time t with n_o the number of obstacles in the environment and r_o^i is the radius of the i^{th} obstacle. Note that in static environments, the obstacle positions do not change over time.
2. *Liveness:* The UAV should stay within a certain proximity of the planned trajectory:
$$\|\boldsymbol{p}(t) - \boldsymbol{p}_\tau(t)\| \leq \lambda_d, \forall t \in [t_p, t_{p+1}] \tag{3}$$
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.

If the UAV is following its trajectory without deviating too much, replanning operations can be relaxed while still operating safely. To minimize unnecessary replanning operations we introduce the following problem:

Problem 2: ***Replanning Relaxation*****:** Given the assumptions in Problem 1, find a policy to decide next time t_{s+1} to monitor the state of the system and postpone next replanning time, obtained in Problem 1, to a later $t^*_{p+1} \geq t_{p+1}$ such that $t_{p+1} \leq t_{s+1} \leq t^*_{p+1}$ and the same safety and liveness constraints hold:

1. *Safety Constraint:* $\|\boldsymbol{p}(t) - \boldsymbol{o}_i(t)\| > r_o^i, \forall t \in [t_p, t_{s+1}], \forall i \in \{1, \cdots, n_o\}$
2. *Liveness Constraint:* $\|\boldsymbol{p}(t) - \boldsymbol{p}_\tau(t)\| < \lambda_d, \forall t \in [t_p, t_{s+1}]$

In cluttered environments, avoiding obstacles may require a UAV to follow winding trajectories which may lead to a significant drift from the planned trajectory, especially when the speed of the vehicle is high. We formally cast this problem as follows:

Problem 3. ***Speed Adaptation*****:** Given Eq. 1, and assumptions listed in the previous problems, at replanning time t_p, after defining a trajectory τ, find a policy to determine the maximum speed v^* such that the following conditions are satisfied:

$$d_{avg}(\kappa_m, v^*) \leq \xi_t, \text{with } v^* \in [v_{min}, v_{max}]$$

where d_{avg} is the average deviation from the planned trajectory with maximum curvature κ_m, ξ_t is a deviation threshold defined by the user, and v_{min} and v_{max} are the minimum and maximum allowed UAV speeds, respectively.

4 System Models

In this section, we present the dynamical model of the system, and the noise and disturbance models considered for reachability analysis.

4.1 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. As also described in [4, 18], the thrust (F_i) and moment (M_i) produced by each rotor is proportional to their angular speed:

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

where κ_f and κ_m are proportionality constant for thrust and moment respectively. The net thrust and moments generated on the quadrotor is calculated by:

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

where d is the arm length of the quadrotor.

The dynamics of the quadrotor are then described as follows:

$$\begin{aligned} \dot{\boldsymbol{p}}_q{}^{\mathsf{T}} &= \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \\ \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \cos\Phi\cos\psi\sin\theta + \sin\Phi\sin\psi \\ \cos\Phi\sin\theta\sin\psi - \cos\psi\sin\Phi \\ \cos\theta\cos\Phi \end{bmatrix} u_1 \\ \begin{bmatrix} \dot{\Phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} 1 & \sin\Phi\tan\theta & \cos\Phi\tan\theta \\ 0 & \cos\Phi & -\sin\Phi \\ 0 & \sin\Phi\sec\theta & \cos\Phi\sec\theta \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \\ \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} &= \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \end{aligned} \quad (4)$$

During the simulations, we linearized the dynamics of the quadrotor [4].

4.2 High-Level Motion Planning Model

In this work, we use the following high-level motion planning model for the quadrotor to capture the evolution of its position and velocity states in the $x - y$ plane. This simplified model will be used to compute the reachable tubes in Section 6.1.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}(\boldsymbol{u}(t) + \boldsymbol{\eta}_u + \boldsymbol{\eta}_d) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{\eta}_y \end{aligned} \quad (5)$$

where $\boldsymbol{x} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^{\mathsf{T}}$ is the state, $\boldsymbol{y}$ is the position and velocity measurement with sensor noise $\boldsymbol{\eta}_y$. $\boldsymbol{u}$ is the input acceleration with noise $\boldsymbol{\eta}_u$ and the effect of disturbance $\boldsymbol{\eta}_d$.

$\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ matrices are given by:

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

where $\boldsymbol{I}$ is a 4×4 identity matrix.

This model assumes that the position of the quadrotor along the z axis is constant and the yaw angle is equal to zero.

4.3 Position, Low Level, and Attitude Controls

By adopting the framework in Fig. 2, in order to follow the desired trajectory $\boldsymbol{x}_\tau = \begin{bmatrix} x_\tau & y_\tau & v_{x,\tau} & v_{y,\tau} \end{bmatrix}^{\mathsf{T}}$, the position controller which is implemented as a series of PD loops generates the desired acceleration inputs:

$$\begin{aligned} \ddot{x}_{des}(t) &= K_p(x_\tau(t) - x(t)) + K_d(v_{x,\tau}(t) - v_x(t)) \\ \ddot{y}_{des}(t) &= K_p(y_\tau(t) - y(t)) + K_d(v_{y,\tau}(t) - v_y(t)) \end{aligned}$$

where K_p and K_d are proportional and derivative coefficients of the controller respectively. To provide these desired acceleration inputs, low-level controllers generate the necessary angle inputs to the attitude control. The necessary angular speeds are calculated by the attitude controller and the thrust and moment values that each rotor should provide are calculated through motor dynamics. Finally, the response of the quadrotor to these thrust and moment values in terms of its state is generated by rigid body dynamics.

4.4 Noise Models

The behavior of a quadrotor can be negatively affected by various factors such as sensor and process noises, and external disturbances which may cause the vehicle to drift from its planned trajectory and possibly collide with obstacles. Thus, to guarantee safety, these factors should be taken into account during planning.

In this work, the effect of disturbances, noises and uncertainties are assumed to be uniformly distributed and bounded by ellipsoids ϵ. It should be noted that beside simplifying the calculation of reachable sets, this is a valid assumption because high uncertainties with very low probability can be neglected and thus the value can be bounded. For ease of discussion and computation, the noise

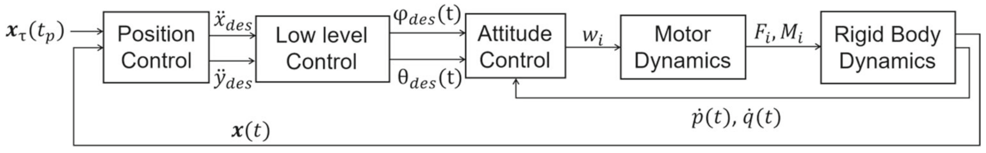


Fig. 2 The control diagram for a quadrotor trajectory following operation [18]

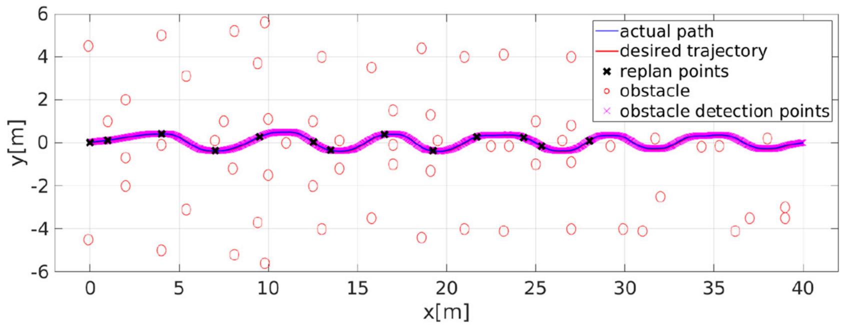


Fig. 3 Periodic sensor monitoring and obstacle detection

values are drawn from a uniform distribution. Note that this assumption is valid because we want to treat all uncertainty values in the same way from a safety point of view. It should be also noted that the presented framework is independent from the distribution of the noise values as long as they can be bounded by ellipsoids. The state uncertainty caused by the sensor measurement noise is represented by $\boldsymbol{\eta}_y \in \epsilon(\mathbf{0}, \boldsymbol{Y})$ which represents an ellipsoid with center around the origin and shape matrix $\boldsymbol{Y}$. The combination of the mechanical uncertainties from the rotors, gears and propellers, and the low-level controllers $\boldsymbol{\eta}_u$, as well as the effect of the external wind disturbance $\boldsymbol{\eta}_d$ are considered as noise on the applied input. The total uncertainty on the input is assumed to be bounded by an ellipsoid $(\boldsymbol{\eta}_u + \boldsymbol{\eta}_d) \in \epsilon(\mathbf{0}, \boldsymbol{U})$ centered around the origin with shape matrix $\boldsymbol{U}$.

4.5 Sample Scenario

Throughout this work, we use a sample case study as a baseline reference to our proposed approach: a UAV is tasked to travel 40m along the x direction with constant speed $v = 1.0$m/s in a cluttered environment under wind disturbance $\boldsymbol{d} = [-0.1, 0.1]^{\mathsf{T}}$m/s. The UAV is equipped with an omnidirectional range sensor with 10m range which is used to detect the obstacles in the environment. It is assumed that the vehicle is able to detect all the obstacles within the sensor range.

In traditional motion planning methods, the state of the system and the obstacle positions are monitored at high frequency and replanning occurs whenever the UAV detects a new obstacle along its path [2, 14]. Monitoring the range or vision sensors to detect the obstacles at high frequency brings unnecessary computational burden to the system. In Fig. 3, the UAV travels to its goal position using this traditional motion planning approach under the conditions described in the sample scenario. The desired trajectory of the UAV is shown by a red curve and its actual path is shown by a blue dotted line. The maximum deviation from its desired trajectory is recorded as 11.42cm. Replanning (depicted by black cross marks $\times$) happens 13 times through the execution of the operation. In order to detect and avoid obstacles, pose and range sensor are monitored periodically at 40Hz rate to resemble the rate of real lidars and are depicted by magenta diamonds 1721 times in Fig. 3.

5 Framework

In order to solve the problems listed in Section 3, we propose a reachability-based self-triggered scheduling and replanning approach which can be broken down in to several components, as depicted in the framework shown in Fig. 4. The framework consists of reachability analysis both for UAVs and dynamic obstacles, self-triggered scheduling,

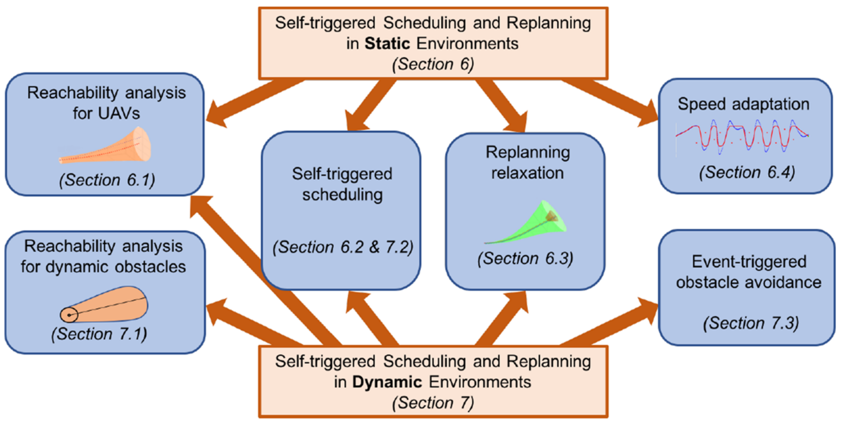

Self-triggered Scheduling and Replanning in Static Environments (Section 6)
Reachability analysis for UAVs (Section 6.1)
Self-triggered scheduling (Section 6.2 & 7.2)
Replanning relaxation (Section 6.3)
Speed adaptation (Section 6.4)
Reachability analysis for dynamic obstacles (Section 7.1)
Event-triggered obstacle avoidance (Section 7.3)
Self-triggered Scheduling and Replanning in Dynamic Environments (Section 7)

Fig. 4 Overall reachability-based self-triggered scheduling and replanning framework for dynamic and static environments

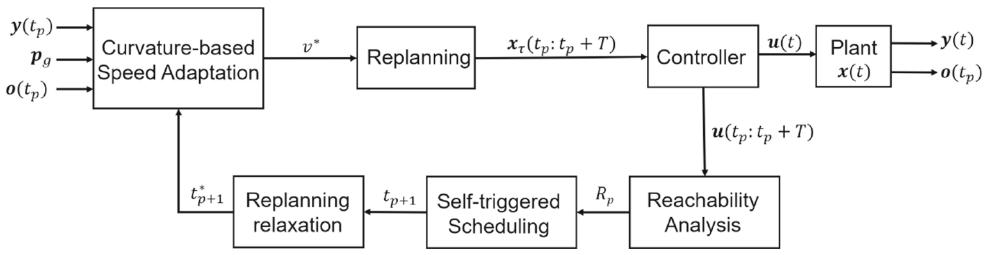


Fig. 5 Our self-triggered scheduling and replanning framework with speed adaptation

replanning relaxation, curvature-based speed adaptation and event-triggered obstacle avoidance. These individual components are analyzed in detail in Sections 6 and 7.

6 Self-Triggered Scheduling and Replanning for Static Environments

In order to use the available computational resources in an efficient way, we introduce scheduling policies to make decisions about next sensor monitoring and replanning times. Our approach leverages reachability analysis to predict the future states of the system and utilizes self-triggered control for scheduling. In this section, we also introduce i) an approach to update reachable sets and relax replanning operations to further minimize computation and ii) a curvature-based speed adaptation method for better tracking performance. The overall framework is shown in Fig. 5.

6.1 Reachablity Analysis for Trajectory Tracking on Quadrotors

A *reachable set* or *reach set* of a system is defined as the set of states that can be reached from a given initial state with an admissible input over a certain time horizon [10]. 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 possible future states $\boldsymbol{x}(t)$ for $t_0 \le t \le t_f$ where the initial set $\epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ is 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 $\boldsymbol{U}$. $R^+(\boldsymbol{x}_0, \boldsymbol{u}(t), t_f)$ is the external bound of the reachable set and $R^+_p(\boldsymbol{x}_0, \boldsymbol{u}(t), t_f)$ is the projection of the external bound of the reachable set on the position space.

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

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

where $\boldsymbol{\Phi}(t, t_0) = e^{A(t-t_0)}$, and the symbol $\oplus$ represents geometric sum. A reachable tube $R(\boldsymbol{x}_0, \boldsymbol{u}(t), [t_0, t_0 + T])$ is the set of all reachable sets over the time interval $\Delta T = [t_0, t_0 + T]$ and its external bound is described as follows:

$$R^+(\boldsymbol{x}_0, \boldsymbol{u}(t))|_{t_0}^{t_0+T} = R^+(\boldsymbol{x}_0, \boldsymbol{u}(t), [t_0, t_0+T]) = \bigcup_{t_0}^{t_0+T} R^+(\boldsymbol{x}_0, \boldsymbol{u}(\zeta), \zeta)d\zeta$$

which is the union of all reachable sets from time t_0 to t_0+T. $R^+_p(\boldsymbol{x}_0, \boldsymbol{u}(t), [t_0, t_0 + T])$ is the projection of the external bound of the reachable tube on to the position space.

For a UAV following a desired trajectory $\boldsymbol{x}_\tau(t_p : t_{p+1})$, the reachable sets are generated over the time interval $\Delta t_p = [t_p, t_{p+1}]$ where t_p is the current replanning time and t_{p+1} is the next replanning time, initially set to $t_p + T$ before the rescheduling operation. The desired trajectories are computed minimizing jerk [17], and they contain the desired positions and velocities that the UAV has to track along the path. Using a PD controller, the acceleration input required to track the trajectory is calculated as follows:

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

where $t \in [t_p, t_{p+1}]$. In order to calculate the set of inputs that would be applied to the UAV during its motion, an online simulation is run as if there is no disturbance and noise in the environment. The control input $\boldsymbol{u}(t)$ is calculated using this PD controller and applied to the simulated system for $t \in [t_p : t_{p+1}]$. As a result, a set of control inputs $\boldsymbol{u}(t_p : t_{p+1})$ is generated.

The set of inputs calculated using the online simulator are used to construct the reachable tubes considering the

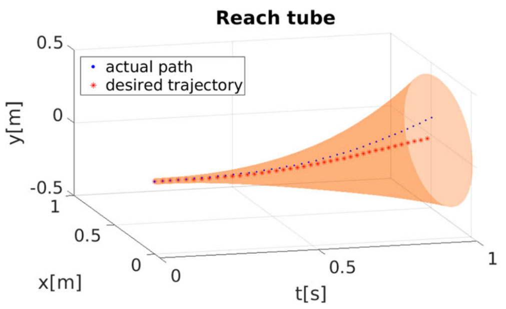


Fig. 6 The reachable tube (orange) associated with a planned trajectory (red starred curve) for a quadrotor moving in the $+x$ direction over a time interval [0,1.0]s. The blue dotted curve indicates the actual trajectory followed by the UAV drifted from the planned trajectory due to wind disturbance but still contained inside the reachable tube

disturbances and uncertainties. A position reachable tube constructed from $t_p = 0$ to $t_{p+1} = 1.0$s for a quadrotor following a straight 1m long trajectory in the $+x$ direction in open loop is shown in Fig. 6. The actual path of the quadrotor (blue dotted curve) deviates from the desired trajectory (red start curve) due to the presence of wind disturbance $\boldsymbol{d} = [0, 0.45]$m/s. Nevertheless, the path of the quadrotor is contained inside the reachable tube since the system uncertainties and disturbances are taken into consideration during the reachable set computation. To perform such reachability analysis, we leveraged the Ellipsoidal Toolbox [10] for ease of integration with our Matlab simulations and physical experiments. However, any other reachability tool could be used within our framework.

6.2 Self-Triggered Scheduling

In order to schedule next sensor monitoring time while guaranteeing safety and liveness between replanning operations, we leverage the reachable tubes calculated in the previous section. In this section we introduce our self-triggered scheduling approach to guarantee safety and liveness between aperiodic replanning operations.

6.2.1 Case 1: Open Loop Scheduling and Replanning

Our novel self-triggered scheduling policy consists of deciding the next sensor monitoring and motion replanning time t_{p+1} such that safety and liveness of the UAV are guaranteed between replanning times t_p and t_{p+1} without monitoring its pose and range sensors between replanning times, for example in case of GPS signal loss. The safety requirement is violated whenever a collision with an obstacle becomes possible. We calculate the first time in which the UAV can collide with an obstacle, t_c, using a reachable tube as follows:

$$t_c = \min(t | R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t \in [t_p, t_p + T]) \cap \boldsymbol{O} \neq \emptyset) \quad (6)$$

where $\boldsymbol{O}$ is the set of obstacles detected at time t_p.

As the UAV does not constantly check its range sensor while moving towards its goal, it may leave the region that is detected by the range sensor. This situation raises a potential danger of obstacle collisions because the UAV does not have information about the environment beyond its sensory range. The earliest time that the UAV can leave the region sensed by the range sensor field of view, t_l is calculated as follows:

$$t_l = \min(t | R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t \in [t_p, t_p + T]) \not\subset \boldsymbol{r}(t_p)) \quad (7)$$

where $\boldsymbol{r}(t_p)$ is the region covered by the field of view of the range sensor of the UAV at the planning time t_p.

Liveness constraint might be violated if it is possible for the UAV to deviate more than a threshold λ_d from its desired trajectory. The first time in which the liveness condition might be violated, t_d, is calculated as follows:

$$t_d = \min(t | \| R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t \in [t_p, t_p + T]) - \boldsymbol{p}_\tau(t)\| > \lambda_d) \quad (8)$$

where $\boldsymbol{p}_\tau(t)$ is the desired position of the UAV along the computed trajectory at time $t \in [t_p, t_p + T]$.

Before the end of the planning horizon $t_p + T$, if one or more of the following conditions occur:

– a collision with an obstacle becomes possible at $t_c < t_p + T$
– the deviation could be larger than the permitted threshold t time $t_d < t_p + T$
– the vehicle may leave the region covered by the range sensor at time $t_l < t_p + T$

then the UAV needs to check its state at one of these three times, whichever is the earliest. Otherwise, the next replanning time is scheduled at the end of the time horizon.

$$t_{p+1} = \begin{cases} \min(t_c, t_d, t_l) - t_r, & \text{if } t_c < t_p + T \text{ or } t_d < t_p + T \text{ or } t_l < t_p + T \\ t_p + T - t_r, & \text{otherwise} \end{cases} \quad (9)$$

where t_r is the amount of time necessary for the replanning calculation.

Lemma 1 *Given t_{p+1} as defined in Eq. 9, the UAV is guaranteed to stay within λ_d proximity of its planned trajectory, not to collide with any obstacle and to stay within the region covered by the range sensor field of view.*

Proof By definition, $R(\boldsymbol{x}_0, \boldsymbol{u}(t), t \in [t_p, t_p + T])$ is the set of all states $\boldsymbol{x}$, such that there exists an input $\boldsymbol{u} \in \epsilon(\boldsymbol{u}(t), \boldsymbol{U})$ and an initial state $\boldsymbol{x}_0 \in \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ which steers the UAV from $\boldsymbol{x}_0$ to $\boldsymbol{x}$ in time t [10]. $R_p^+(\boldsymbol{x}_0, \boldsymbol{u}(t), t \in [t_p, t_p + T])$ is the projection of the external bound of $R(\boldsymbol{x}_0, \boldsymbol{u}(t), t \in [t_p, t_p + T])$ onto the position space. Considering a time value t^* between t_p and t_{p+1}, $t_p < t^* < t_c$, $t_p < t^* < t_l$, and $t_p < t^* < t_d$, by the definitions of t_c in Eq. 6, t_l in Eq. 7 and t_d in Eq. 8,

$$R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t_k), t \in [t_p, t^*]) \cap \boldsymbol{O} = \emptyset$$
$$\| R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t \in [t_p, t^*]) - \boldsymbol{p}_\tau(t)\| < \lambda_d$$
$$R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t \in [t_p, t_p + T]) \subset \boldsymbol{r}(t_p)$$

which proves that for a $t_p < t^* < t_{p+1}$, there doesn't exist an input $\boldsymbol{u}(t^*) \in \epsilon(\boldsymbol{u}(t^*), \boldsymbol{U})$ and an initial state $\boldsymbol{x}(t_p) \in \epsilon(\boldsymbol{x}(t_p), \boldsymbol{X}_0)$ which makes the system reach a state $\boldsymbol{x}$ from $\boldsymbol{x}(t_p)$ such that the position of the system intersects with an obstacle, or leaves the region sensed by the range sensor, or deviates from the desired trajectory more than λ_d. □

In Fig. 7, we demonstrate the results of this approach for our sample case. The desired trajectory of the UAV is shown by red curve and the actual trajectory followed by the UAV is shown by blue dots. The deviation threshold is

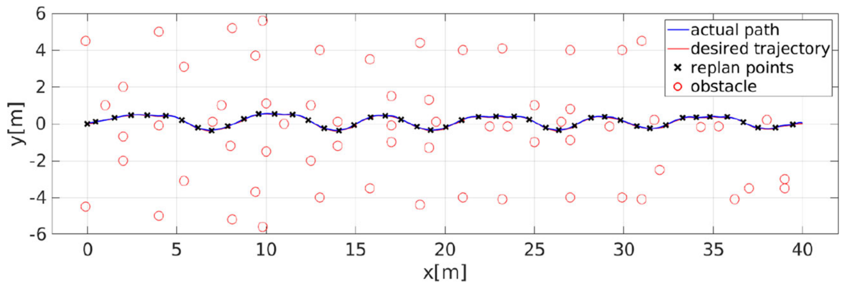


Fig. 7 Desired trajectory (red curve) and actual path (blue dots) of the quadrotor with self-triggered for position, velocity and obstacle monitoring

picked as $\lambda_d = 50\text{cm}$ and the wind disturbance is constant everywhere $\boldsymbol{d} = [-0.1, 0.1]^\mathsf{T}\text{m/s}$. The UAV checks its states and replans its motion only at the points shown by black crosses 46 times and travels with an open loop controller in between replanning points. The maximum deviation along the path is recorded as 16.69cm. As can be noticed, the UAV never collides with an obstacle or deviates from its desired trajectory more than the permitted threshold thanks to our self-triggered approach.

6.2.2 Case 2: Closed Loop Scheduling and Replanning

Compared to monitoring range sensors, we note that monitoring pose sensors is usually computationally negligible and is necessary for closed loop trajectory-tracking operations. Here, we consider the case in which the UAV can monitor its position and velocity periodically while checking for obstacles is scheduled aperiodically at replanning times. Since the UAV checks its position sensor constantly, the liveness constraint is not taken into consideration in this case. If it becomes possible for the UAV to collide with an obstacle or to leave the region sensed by the range sensor before the end of its time horizon, the UAV monitors its range sensor and replans its trajectory at either t_c or t_l, whichever is earlier.

$$t_{p+1} = \begin{cases} \min(t_c, t_l) - t_r, & \text{if } t_c < t_p + T \text{ or } t_l < t_p + T \\ t_p + T - t_r, & \text{otherwise} \end{cases} \tag{10}$$

Using this approach, the UAV is guaranteed to stay within its sensor field of view and avoid collisions with an obstacle, following the same reasoning in Lemma 1.

In Fig. 8, we demonstrate the sample case, where the UAV monitors its position and velocity periodically and schedules next obstacle monitoring time based on the proposed self-triggered scheduling approach. The UAV checks its range sensor to detect the obstacles and replans its motion at points shown by black crosses only 37 times. The desired trajectory is shown by a red curve and the actual path of the UAV is shown by blue dots. The maximum deviation from the desired trajectory is recorded as 11.61cm. To further decrease the number of replanning operations, we update the reachable tubes based on the observed state of the system which is discussed in the next section.

6.3 Reachable Set Shrinking and Replanning Relaxation

Reachable tubes are constructed to capture the worst case scenario in terms of external disturbances and noises. In ideal conditions, without disturbance, the system can follow its desired behavior very closely, and therefore replanning (i.e., computing a new reachable tube) at each time that the reachable tube collides with an obstacle might be over-conservative and computationally expensive. To overcome this limitation and postpone next monitoring and replanning operations, we leverage the deviation from the center of the reachable set at the replanning time t_{p+1} to update the

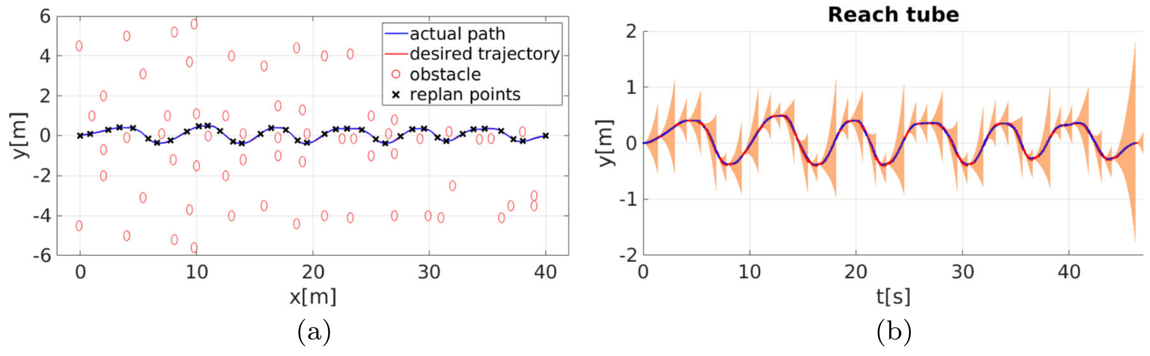


Fig. 8 Self-triggered scheduling for obstacle detection approach applied to the sample scenario. **a** Desired trajectory (red curve) and actual path (blue dots) of the quadrotor with selftriggered scheduling for obstacle monitoring. **b** Corresponding reachable tubes

reachable tube without recalculating them. The deviation from the center of the position reachable tube can be calculated as follows:

$$d(t_{p+1}) = \|\boldsymbol{p}(t_{p+1}) - \boldsymbol{p}_r(t_{p+1})\|$$

where $\boldsymbol{p}(t_{p+1})$ is the position of the UAV at time t_{p+1} and $\boldsymbol{p}_r(t_{p+1})$ is the center of the position reachable tube at t_{p+1}. In the ideal conditions (perfect system model and controller), the position reachable tube would be centered around the desired trajectory. The radius of the external bound of the reachable tube $R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t))|_{t_p}^{t_p+T}$ can be reduced by $r_t(t_{p+1}) - d(t_{p+1}) - \eta_p$ where $r_t(t_{p+1})$ is the radius of the position reachable tube at time t_{p+1} and η_p is the position measurement uncertainty bound. The reasoning behind this shrinking is that if the system has not deviated from the center of the reachable set by a large amount, it becomes impossible for the system to reach the previously computed reachable set border.

By shrinking, we obtain an updated position reachable set $\bar{R}_p^+(\boldsymbol{x}(t_{p+1}), \boldsymbol{u}(t), t)$ with a smaller external bound than the original reachable set. Figure 9a is a pictorial representation of the reachable tube update procedure where the original position reachable tube $R_p^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t))|_{t_p}^{t_p+T}$ is shown by green region and the updated position reachable tube $\bar{R}_p^+(\boldsymbol{x}(t_{p+1}), \boldsymbol{u}(t))|_{t_{p+1}}^{t_p+T}$ is shown by dark orange region.

Using this approach, the UAV is guaranteed to stay within the updated reachable tube, as formally described in Lemma 2:

Lemma 2 *Given that $\boldsymbol{x}(t_{p+1}) \in R^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t))|_{t_p}^{t_p+T}$, the state of the UAV is guaranteed to stay within the updated reachable tube at any future time $t_{p+1} \le t \le t_p + T$:*

$$\boldsymbol{x}(t) \in \bar{R}^+(\boldsymbol{x}(t_{p+1}), \boldsymbol{u}(t))|_{t_{p+1}}^{t_p+T}, \ \forall t \in [t_{p+1}, t_p + T]$$

Proof To prove this lemma, we leverage the geometric properties of reachable sets following the representation in Fig. 9b. The external bound for the reachable tube calculated at planning time t_p, $R_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t)$ is given as follows:

$$R_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t) \ = e^{\boldsymbol{A}(t-t_p)}\epsilon(\boldsymbol{x}(t_p), X_0) \oplus \int_{t_p}^{t} e^{\boldsymbol{A}(t-\zeta)}\boldsymbol{B}\epsilon(\boldsymbol{u}(\zeta), \boldsymbol{U})d\zeta$$

for $t_p \le t \le t_p + T$. At the time t_{p+1}, a reachable set from the measured state $\boldsymbol{y}(t_{p+1})$ with the same input sequence can be calculated as follows:

$$R_2^+(\boldsymbol{y}(t_{p+1}), \boldsymbol{u}(t), t) \ = e^{\boldsymbol{A}(t-t_{p+1})}\epsilon(\boldsymbol{y}(t_{p+1}), X_0) \oplus \int_{t_{p+1}}^{t} e^{\boldsymbol{A}(t-\zeta)}\boldsymbol{B}\epsilon(\boldsymbol{u}(\zeta), \boldsymbol{U})d\zeta$$

for $t_{p+1} \le t \le t_{p+1} + T$. The reachable set R_1^+ after t_{p+1} can also be written as follows:

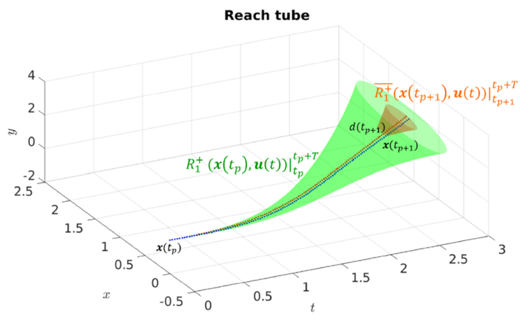
$$R_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t) \ = e^{\boldsymbol{A}(t-t_{p+1})}R_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t_{p+1}) \oplus \int_{t_{p+1}}^{t} e^{\boldsymbol{A}(t-\zeta)}\boldsymbol{B}\epsilon(\boldsymbol{u}(\zeta), \boldsymbol{U})d\zeta$$

for $t_{p+1} \le t \le t_p + T$. By definition, the reachable set at R_1^+ at time t_{p+1} contains the measured state:

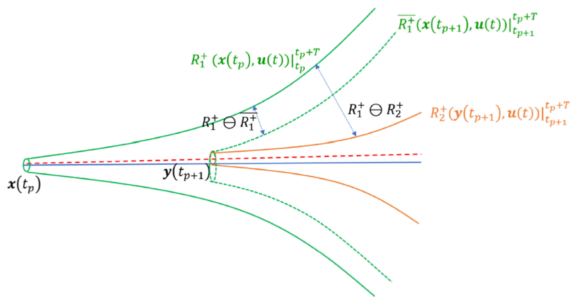
$$R_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t_{p+1}) \supset \epsilon(\boldsymbol{y}(t_{p+1}), X_0)$$

Therefore, at time t_{p+1}, the reachable set R_1^+ contains R_2^+. By the definition of the external bounds of these reachable sets, it can be seen that the radii of both reachable sets grow exponentially over time and at time t_{p+1} the radius of R_1^+ is larger. Therefore, the difference between the reachable set areas increases quadratically with the radius of the reachable sets. Our shrinking procedure consists in reducing the reachable set radius by a constant value $r_t(t_{p+1}) - d(t_{p+1}) - \eta_p$, therefore the initial shrunk set also contains the measured state $\boldsymbol{y}(t_{p+1})$:

$$\bar{R}_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t) \supset \epsilon(\boldsymbol{y}(t_{p+1}), X_0)$$



(a)


$\overline{R_1^+}(\boldsymbol{x}(t_{p+1}), \boldsymbol{u}(t))|_{t_{p+1}}^{t_p+T}$
$R_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t))|_{t_p}^{t_p+T}$
$R_1^+ \ominus \overline{R_1^+}$
$R_1^+ \ominus R_2^+$
$R_2^+(\boldsymbol{y}(t_{p+1}), \boldsymbol{u}(t))|_{t_{p+1}}^{t_p+T}$
$\boldsymbol{x}(t_p)$
$\boldsymbol{y}(t_{p+1})$

(b)

Fig. 9 Reachable tube update procedure. **a** Updating the reachable tubes based on the deviation from the reachable tube center. **b** Pictorial explanation of the procedure to update the reachable tubes online

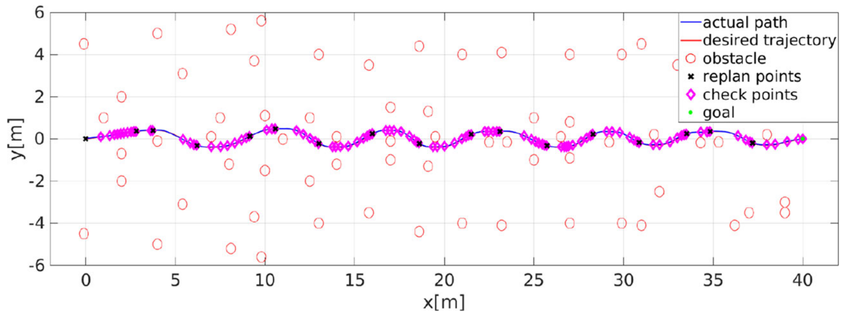


Fig. 10 Replanning relaxation approach applied to sample scenario

As the shrinking is done with a constant reduction, the difference between the reachable set R_1^+ area and the shrunk reachable set $\bar{R}_1^+$ area grows linearly with the reachable set radius. This concludes that the shrunk reachable set always contains the reachable set if a new set is calculated from the observed state:

$$\bar{R}_1^+(\boldsymbol{x}(t_p), \boldsymbol{u}(t), t) \supset R_2^+(\boldsymbol{y}(t_{p+1}), \boldsymbol{u}(t), t) \supset \boldsymbol{x}(t), \forall t \in [t_{p+1}, t_p + T]$$

□

The next scheduling time to update the reachable tube is calculated as defined in Section 6.2. When the difference between consecutive scheduling times gets smaller, a *Zeno phenomenon* [7] may occur and lead to an infinite number of reachable tube updates before t^*_{p+1}. In order to prevent such behavior, we consider time threshold δt and include the following constraint:

$$t^*_{p+1} = t_{s+1} \quad \text{if } t_{s+1} - t_s \leq \delta_t \tag{11}$$

Eq. 11 shows that when the scheduled times to update the reachable tube become very close to each other, a new reachable tube is calculated as described in Section 6.1, which prevents infinite number of reachable tube updates caused by the Zeno phenomenon.

In Fig. 10, using the same sample case described in Section 4.5, we demonstrate the results of the UAV motion where the next obstacle monitoring time is scheduled using the shrunk tubes similar to the one shown in Fig. 9a. The UAV checks its range sensor to detect the obstacles and replans its motion at points shown by black crosses only 17 times whereas without replanning relaxation, replanning occurs 37 times as shown in Section 6.2.2. The desired trajectory is shown by red curve and the actual path of the UAV is shown by blue dots. The maximum deviation from the desired trajectory is recorded as 11.80cm.

6.4 Curvature Based Speed Adaptation

In order to guarantee liveness constraints, the UAV needs to follow its desired trajectory closely. In cluttered environments, trajectories may become very curvy in order to avoid the obstacles. Even though a UAV can perform better tracking performance with low speeds, it becomes very hard to achieve tracking with high speeds without drifting from the planned trajectory. In contrast, trajectories with low curvatures can be closely followed even with high speeds. For example, in Fig. 11, the actual path of a UAV is compared when it is following the same desired trajectory (red curve) with different speeds: v = 1m/s in Fig. 11a and v = 0.25m/s in Fig. 11b. As expected and can be noticed by comparing the two figures, the UAV is able to follow its trajectory with less deviation with a lower speed.

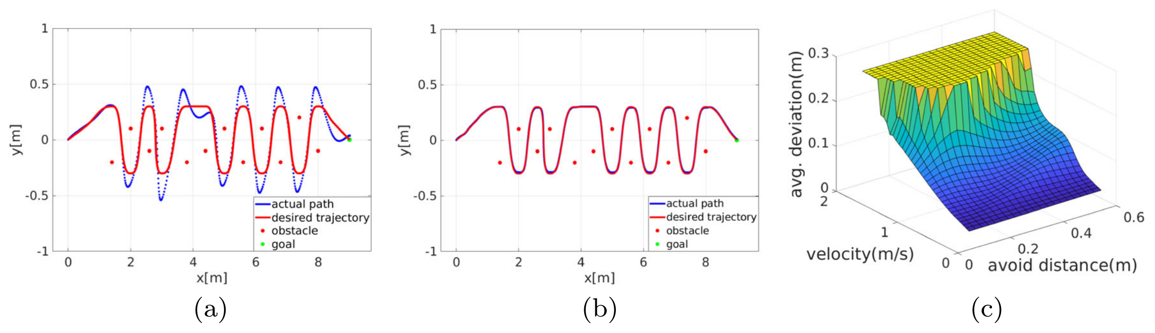


Fig. 11 Effects of the velocity on the deviation, [33]. **a** The path of the UAV moving with v = 1m/s on a path with high curvature, drifting on average 9.05cm. **b** The path of the UAV moving with v = 0.25m/s on the same path, drifting on average 1.50cm. **c** The experimental relationship between velocity, avoid distance from obstacle and the average deviation from the trajectory

In order to decide the optimal speed to use for a given path, we create a policy which adapts the desired speed to follow an obstacle avoidance path based on the maximum curvature along the path. For curvature computation, we leverage the analysis presented in [1], as follows:

$$\kappa_i = \frac{4A}{d_{(i-1)i} d_{i(i+1)} d_{(i-1)(i+1)}}, \quad i \in \{1, \cdots, n-1\} \tag{12}$$

where d_{ij} is the distance between two waypoints i and j, A is the area of the triangle formed by three consecutive waypoints, and n is the total number of waypoints. Here, waypoints are the points that the UAV needs to visit in order to avoid the obstacle. In Fig. 11c, the experimental results demonstrating the relationship between the velocity, desired avoiding distance from the obstacle (which closely affects the curvature of the path) and average deviation from desired trajectory are shown. As can be noticed, the average deviation from the desired trajectory increases with the velocity. When the avoid distance from the obstacles are set around 0.3m, the curvature becomes the maximum, resulting in more average deviation from the trajectory for the same speed. These results suggest an exponential relationship between the estimation of average deviation from the desired trajectory and the maximum curvature of the path and the speed of the UAV as follows:

$$d_{avg}(\kappa_m, v) = \frac{1}{\Omega} e^{\kappa_m \cdot v} \tag{13}$$

where d_{avg} is the average deviation from the trajectory during its motion. The parameter Ω is determined using the experimental data so that the estimation is an over-approximation, and it is different for different types of vehicles. κ_m is the maximum curvature along the trajectory:

$$\kappa_m = \max_{i \in [1, n-1]} \kappa_i$$

The objective of the UAV is to finish its task as fast as possible without deviating too much from the planned trajectory. Therefore, the optimal velocity for the UAV v^* is calculated as the largest possible velocity which keeps the average deviation under a given threshold ξ_t:

$$\begin{aligned} \mathcal{V} &= \left\{ v : d_{avg}(\kappa_m, v) < \xi_t,\ v_{min} < v < v_{max} \right\} \\ v^* &= \max(\mathcal{V}) \end{aligned} \tag{14}$$

where $\mathcal{V}$ is the set of allowable velocities between v_{min} and v_{max}, that keeps the average deviation d_{avg} below ξ_t.

6.5 Simulation Results

In this section, we show and compare the simulation results of each approach presented above for a UAV waypoint navigation case study under sensor and process noises, and wind disturbance in an environment cluttered with circular obstacles. The quadrotor UAV that we considered during these simulations uses a localization sensor (e.g., GPS) to

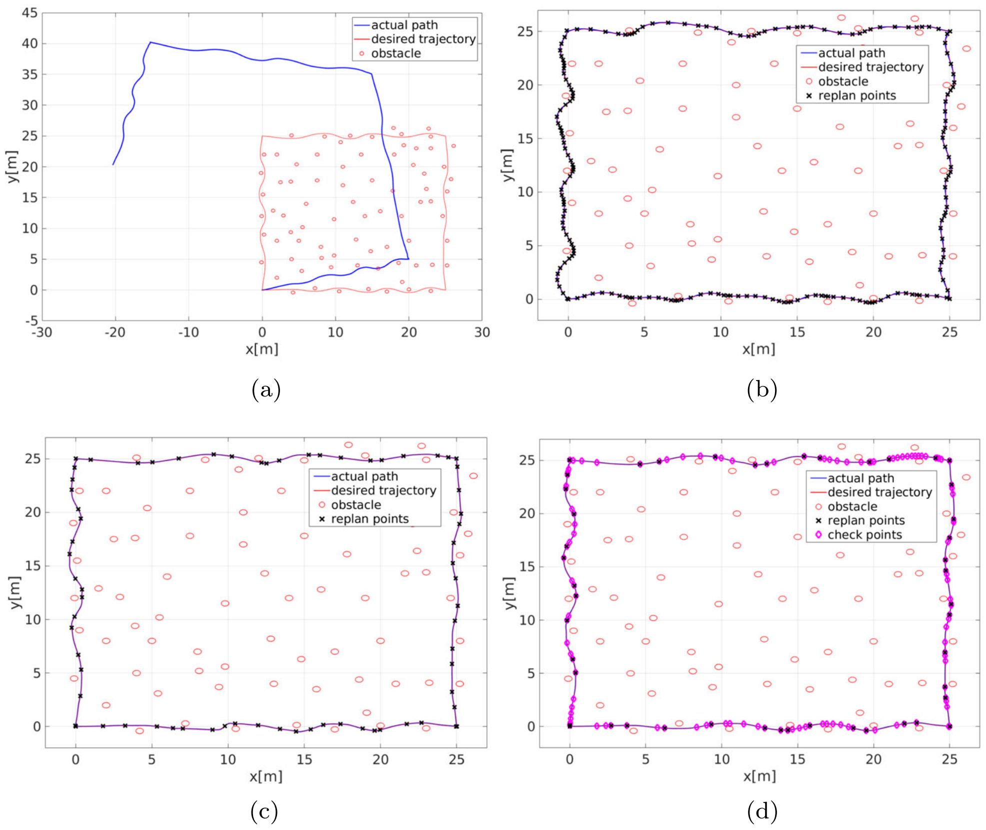


Fig. 12 Comparison between the simulation results of different techniques. **a** Open-loop. **b** Self-triggered scheduling for position, velocity and obstacle monitoring. **c** Self-triggered scheduling for obstacle monitoring. **d** Replanning relaxation

monitor its position in the environment and a range sensor (e.g., a lidar) with a limited 10m range and a 360° field of view to detect the obstacles. The UAV is initialized at the origin with zero velocity. Its mission consists in navigating a square trajectory with 25m sides. The wind disturbance $\boldsymbol{d} = [-0.1, 0.1]^{\mathsf{T}}$ is present everywhere in the environment and its value is unknown to the vehicle.

In Fig. 12a, the trajectory of a quadrotor is planned in the beginning of its motion assuming that the exact positions of all the obstacles are known a priori. The trajectory with constant desired speed of 0.5m/s is controlled in open-loop without getting feedback from neither the localization nor the range sensors. Since the quadrotor does not observe its position while moving under the effect of disturbance, it deviates from its planned trajectory significantly and collides with multiple obstacles.

In the simulations demonstrated in Fig. 12b–d the UAV does not have any prior knowledge about the obstacle locations in the environment. In Fig. 7, the UAV computes its reachable sets considering the disturbance and noise bounds and uses these reachable sets to schedule the times that it needs to monitor its range and pose sensor as described in Section 6.2.1. The quadrotor moves in open loop between replanning times and follows a trajectory with constant desired speed of 0.5m/s. In this case, the replanning occurs 194 times at points shown by black '×' symbols in Fig. 12b. Even though the UAV does not monitor its sensors between replanning times, it is able to complete its task without colliding with any obstacles or deviating too much from the desired trajectory. In Fig. 12c, the pose sensor is monitored periodically (usually not computationally demanding) and the range sensor is monitored aperiodically at times scheduled according to proposed self-triggered approach described in Section 6.2.2. Additionally, depending on the curvature of the trajectory, the speed of the vehicle is also adapted between $v_{min} = 0.25$m/s and $v_{max} = 1.25$m/s as explained in Section 6.4. The replanning and range sensor detection happens only 70 times at the points shown by black '×' symbols and the vehicle completes its task without colliding with any obstacles. To further minimize the sensor checking and replanning operations, we apply the proposed reachable set update method presented in Section 6.3 in the case shown in Fig. 12d. At the points shown by magenta '⋄' symbols, the UAV updates the reachable sets based on its position and schedules the next raplanning time. With the proposed approach, the UAV completed its task by planning its trajectory only 47 times (black '×') and by tracking its desired trajectory closely with an average deviation of 6.18cm. If instead, the UAV was using periodic sensor monitoring, it would have needed over 4000 range sensor checks, which is avoided thanks to our proposed self-triggered approach.

Table 1 Comparison between the different cases simulated in Fig. 12

	Number of replanning and sensor monitoring	Max. deviation	Avg. deviation
Fig. 12a	1	2878.84cm	1432.67cm
Fig. 12b	194	16.71cm	4.27cm
Fig. 12c	70	13.66cm	5.63cm
Fig. 12c	47	14.61cm	6.18cm

The number of replanning operations, number of range sensor checks, and the maximum and average deviation values for different cases are compared in Table 1. The best results are achieved using the self-triggered replanning approach with relaxation (Fig. 12d). The UAV performs a reasonably good tracking performance with the minimum number of sensor monitoring and replanning operations.

6.6 Experimental Results

In order to validate our proposed self-triggered replanning approach, we implemented a series of experiments: i) self-triggered scheduling for both pose and range sensor monitoring (Case 1 presented in Section 6.2.1) and ii) self-triggered scheduling for only range sensor monitoring (Case 2 presented in Section 6.2.2).

6.6.1 Case 1: Open Loop Scheduling and Replanning

The proposed self-triggered approach to schedule the pose sensor monitoring times was validated using an AscTec Hummingbird quadrotor UAV. The range sensor scheduling is not addressed during this experiment. As shown by the framework in Fig. 13, the Matlab ellipsoidal toolbox [10] is used to calculate the reachable sets and the control commands to the quadrotor are sent through ROS. The communication between Matlab and ROS was bridged using the Robotics System Toolbox. The self-triggered scheduling and replanning framework was implemented in Matlab and the position and the velocity of the quadrotor in $x - y$ plane at each scheduled monitoring time were monitored using a Vicon motion capture system. During this experiment, the obstacle is assumed to be static with a known position.

The starting position of the robot is at $(-1.75, -0.4, 1)$m and the goal is located at $(1.75, -0.4, 1)$m. The quadrotor aims to track the planned obstacle-free trajectory without performing periodic sensor monitoring for its position and velocity in $x - y$ plane. The next sensor monitoring time is scheduled considering only the safety condition during this experiment. In Fig. 14a, a sequence of snapshots of the quadrotor is shown in which the quadrotor monitor its position with the rate of 40Hz. In Fig. 15a, the

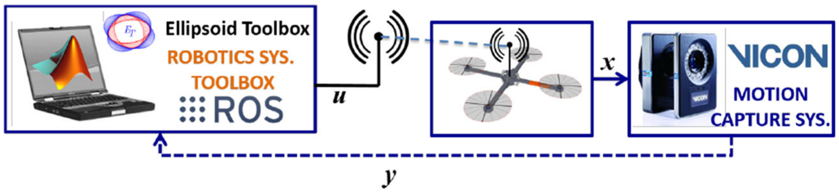


Fig. 13 Framework of the experimental setup, [32]

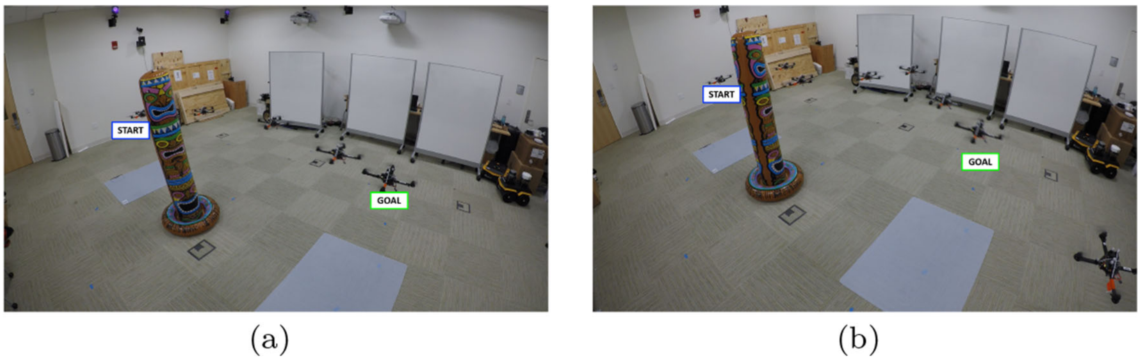


Fig. 14 Obstacle avoidance experiment results with **a** periodic position and velocity monitoring and **b** self-triggered scheduling for sensor monitoring and replanning considering only safety constraint, [32]

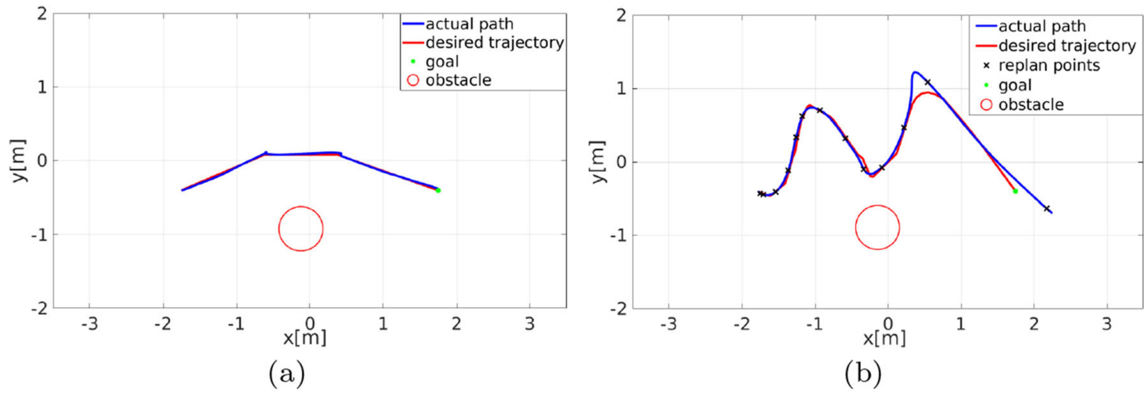


Fig. 15 The desired and actual path of the quadrotor for the experiments shown in Fig. 14. **a** periodic position and velocity monitoring and **b** self-triggered scheduling and replanning with only safety constraint, [32]

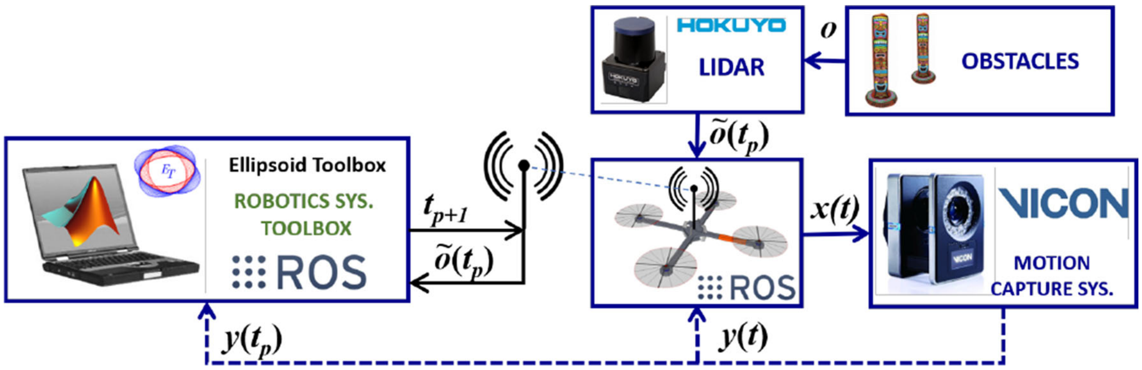


Fig. 16 Framework of the experiment setup, [33]

corresponding obstacle-free desired trajectory (red curve) and the actual path of the UAV (blue curve) are shown. Thanks to periodic sensor checking, the actual and the desired trajectory are very close to each other. In Fig. 14b, a sequence of snapshots of the quadrotor while it is performing self-triggered scheduling and replanning is presented. In Fig. 15b, the actual and the desired trajectory of the quadrotor is shown. The quadrotor monitors its state and replans its trajectory only 13 times at points shown by black '$\times$' symbols. Lack of periodic sensor monitoring caused the quadrotor to deviate from its planned trajectory by nearly 0.5m, however our proposed approach prevented it from colliding with any obstacles. These experiments show that even though periodic sensor monitoring provides better trajectory tracking, it is not necessary as the safety can be guaranteed using the proposed approach with aperiodic monitoring.

6.6.2 Case 2: Closed Loop Scheduling and Replanning

The proposed self-triggered scheduling for range sensor monitoring and replanning approach with speed adaptation was validated using an AscTec Pelican quadrotor UAV. The UAV has an i7 CPU on board for computation and a Hokuyo Lidar range sensor for obstacle detection. The framework presented in Fig. 16 was followed during the experiments in which our Vicon motion capture system was used to monitor the state of the UAV. The quadrotor used its on-board lidar to estimate the positions of the obstacle. Our self-triggered approach was implemented in Matlab similar to the previous case study.

The UAV was tasked to complete a rectangular trajectory by visiting the corners in order: $\{O \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow A \rightarrow O\}$. Two obstacles (inflated poles) were positioned on the top edge of the rectangle. At runtime, the quadrotor built a trajectory to move to the desired waypoints and to avoid the obstacles and computed its reachable sets. Using the proposed approach, the next sensor monitoring and replanning time was scheduled to the instance in which the reachable sets collide with an obstacle for the first time. Since the range of the lidar sensor is long enough to cover the workspace, the time that the system can leave the field of view of the lidar is not considered during this experiment. The speed of the UAV was also adapted when it was moving around the obstacles.

Figure 17a shows an overlapped sequence of snapshots of this experiment and Fig. 17b shows the actual path (blue curve) and the desired trajectory (red curve) of the UAV. The average deviation is recorded as 5.35cm and its speed is adapted in the range of $0.125 - 0.5$ m/s. Sensor checking and replanning occurred only 11 times and the UAV was able to complete its trajectory without colliding with any obstacles. We compared the results of our approach with results of traditional replanning approach with periodic sensor monitoring shown in Fig. 17c. The UAV moved with constant speed $v = 0.125$m/s and it monitored for the obstacles periodically with 40Hz frequency. In this case the average deviation decreased to 3.10cm. However, with the proposed approach, the CPU utilization decreased to 2.7% from 9% with periodic 40Hz lidar monitoring.

7 Self/Event-Triggered Scheduling and Replanning in Dynamic Environments

So far, we have considered scheduling and replanning of UAV operations in static environments, however the UAV may operate in dynamic environments, for example in the presence of other aerial vehicles. For safe operations in dynamic environments, we extend the framework discussed in the previous sections and propose a self/event-triggered scheduling and replanning approach which follows the framework depicted in Fig. 18 [31].

Based on the initial observations about the obstacle positions, a trajectory to the desired goal position $\boldsymbol{p}_g$ is generated. The future states of the UAV are predicted using the reachable tubes calculated as outlined in Section 6.1. If the distance $d_o(t_p)$ between the UAV and the closest

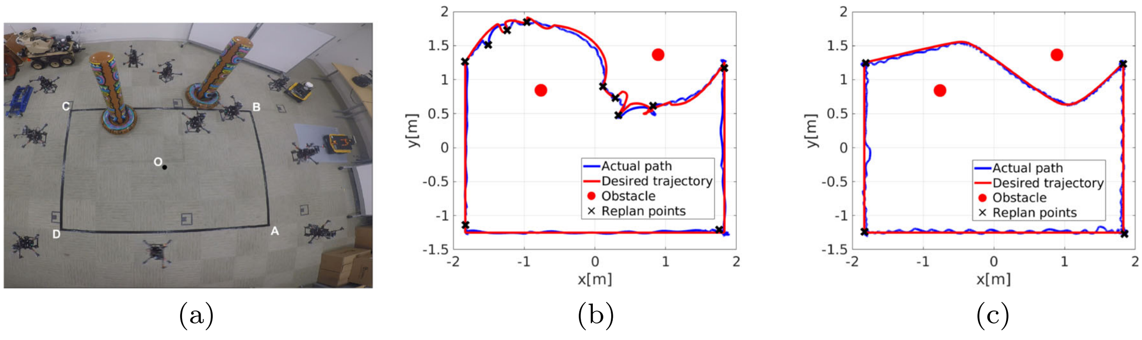


Fig. 17 Waypoint navigation experimental results, [33]. **a** Overlapped sequence of snapshots. **b** Desired vs. actual trajectories of the UAV during the experiment. **c** Desired vs. actual trajectories of the UAV with periodic obstacle detection

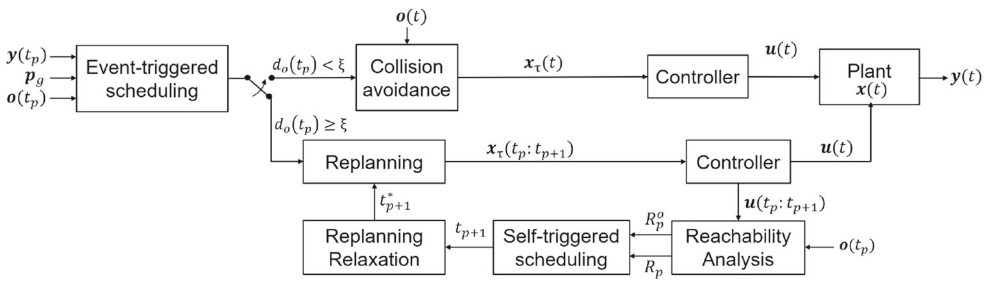


Fig. 18 Overall self/event-triggered scheduling and replanning framework in dynamic environments

mobile obstacle is smaller than a user defined threshold ξ, a collision avoidance behavior is triggered in which sensor checking switches to periodic. Collision avoidance is then performed using repulsive potential fields, [3].

7.1 Dynamic Obstacle Reachability Analysis

In order to guarantee safety of the UAV operations in dynamic environments without monitoring the sensors periodically, it is required to predict the future states of the dynamic obstacles. To capture the possible future states of a dynamic obstacle, its reachable sets are calculated based on the available information about its state, maximum velocity and direction of movement. For the sake of simplicity, we assume that the obstacle has a known maximum speed $v_{o,max}$, and a heading which is not known a priori. Therefore, at the initial planning time t_0, the position reachable set of the obstacle, $R^o_p(\boldsymbol{o}_i(t_0), [t_0 + T])$, is a circle centered around the obstacle position as shown in Fig. 19b, which contains all the points that can be reached within the planning horizon T. The direction and velocity of the mobile obstacle at time t_p, $\vec{\boldsymbol{v}}_{oi}(t_p)$, can be estimated based on previous observations about its position, as follows:

$$\vec{\boldsymbol{v}}_{oi}(t_p) = \frac{\boldsymbol{o}_i(t_p) - \boldsymbol{o}_i(t_{p-1})}{\|\boldsymbol{o}_i(t_p) - \boldsymbol{o}_i(t_{p-1})\|}, \quad \forall i \in [0, n_o] \tag{15}$$

Its reachable set can be constructed along its trajectory while taking the estimation errors and noise into account as depicted in Fig. 19c. Due to the measurement noise and uncertainties, the actual direction of the mobile obstacle might be different from the estimated one. The minimum and maximum limits of its direction of motion can be calculated as follows:

$$\vec{\boldsymbol{v}}^+_{oi}(t_p) = \frac{\boldsymbol{o}^+_i(t_p) - \boldsymbol{o}_i(t_{p-1})}{\|\boldsymbol{o}^+_i(t_p) - \boldsymbol{o}_i(t_{p-1})\|}, \quad \forall i \in [0, n_o] \tag{16}$$

$$\vec{\boldsymbol{v}}^-_{oi}(t_p) = \frac{\boldsymbol{o}^-_i(t_p) - \boldsymbol{o}_i(t_{p-1})}{\|\boldsymbol{o}^-_i(t_p) - \boldsymbol{o}_i(t_{p-1})\|}, \quad \forall i \in [0, n_o] \tag{17}$$

where $\vec{\boldsymbol{v}}^+_{oi}(t_p)$ and $\vec{\boldsymbol{v}}^-_{oi}(t_p)$ are upper and lower limits of the direction of the obstacle where $\boldsymbol{o}^+_i(t_p)$ and $\boldsymbol{o}^-_i(t_p)$ are two extreme points where the obstacle can be at t_p due to uncertainties. $\boldsymbol{o}^+_i(t_p)$ and $\boldsymbol{o}^-_i(t_p)$ are shown in Fig. 19a, where η_x is the maximum measurement noise.

The updated reach set of the obstacle for the time horizon T is shown in Fig. 19c with a blue shaded region. As can be noticed, the reachable set of the obstacle grows along its direction with a rate proportional to the maximum velocity of the vehicle.

7.2 Self/Event-Triggered Scheduling and Replanning

In a dynamic environment, the first time a collision may occur, $t_{c,o}$, is calculated as follows:

$$\begin{aligned} t_{c,o} &= \min(t_k | R^+_p(\boldsymbol{x}(t_p), \boldsymbol{u}(t_k), t_k \in [t_p, t_p + T]) \\ &\quad \cap\ R^o_p(\boldsymbol{o}_i(t_p), t_k \in [t_p, t_p + T]) \neq \emptyset) \end{aligned}$$

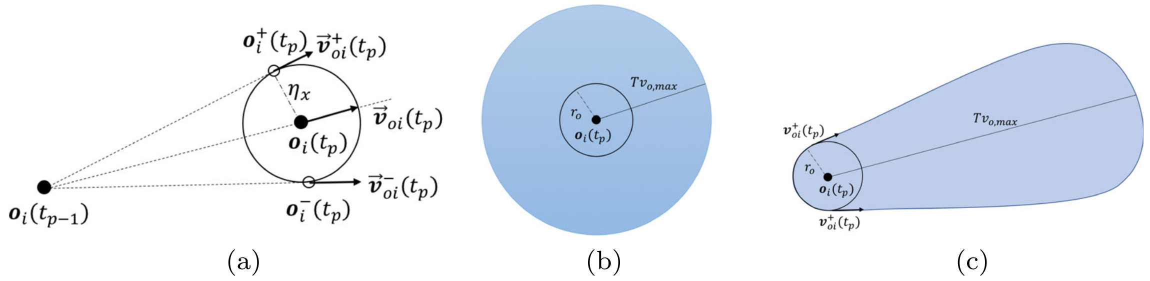


Fig. 19 Reachable tube calculation for dynamic obstacles. **a** Calculation of the obstacle direction based on the previous position of the obstacle and the magnitude of uncertainties. **b** Initial reach set of the obstacle. **c** Reach set of the obstacle updated based on the observed direction

where $R_p^o(\boldsymbol{o}_i(t_p), t_k \in [t_p, t_p + T])$ is the geometric sum of the position reachable sets of the obstacle between t_p and $t_p + T$. In Fig. 20, reachable sets of the UAV (blue circles) and reachable sets of the obstacle (magenta region) are shown at $t_{c,o}$, the time that they collide for the first time.

Similar to the case with static obstacles, the next sensor checking time is scheduled to t_l (which is calculated in Eq. 7) or to $t_{c,o}$ whichever is minimum:

$$t_{p+1} = \begin{cases} \min(t_{c,o}, t_l) - t_r, & \text{if } t_{c,o} < t_p + T \text{ or } t_l < t_p + T \\ t_p + T - t_r, & \text{otherwise} \end{cases} \tag{18}$$

where t_r is the amount of time necessary for the replanning calculation. In case that the UAV doesn't detect any obstacle in the environment, next sensor checking time can be computed considering the worst case scenario, that is when the obstacle is right on the boundary of the sensor field of view.

It should be noted that the same safety guarantees presented in the static case apply while minimizing sensor checking operations also in the dynamic environments.

7.3 Dynamic Obstacle Repulsive Potential Field Collision Avoidance

If $d_o(t_p) < \xi$ a collision avoidance is initiated following a repulsive potential filed approach. The repulsive potential field around the reachable set of the obstacle $W_O(\boldsymbol{p}(t))$ can be computed as follows:

$$W_O(\boldsymbol{p}(t)) = \begin{cases} \frac{1}{2}\alpha_i \left(\frac{1}{\rho(\boldsymbol{p}(t))} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho(\boldsymbol{p}(t)) \le \rho_0 \\ 0, & \text{if } \rho(\boldsymbol{p}(t)) > \rho_0 \end{cases} \tag{19}$$

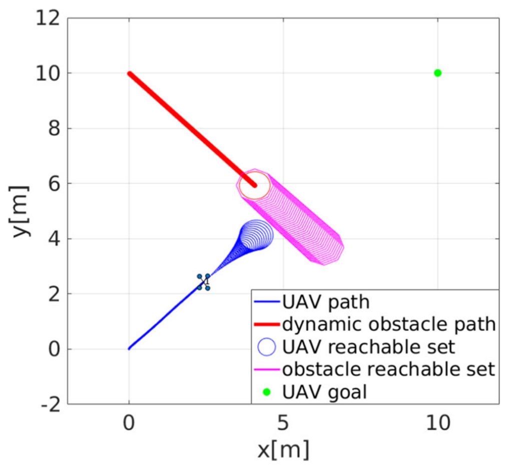


Fig. 20 Collision between the reachable tube of the UAV and reachable set of the obstacle at time $t_{c,o}$

where $\rho(\boldsymbol{p}(t))$ is the shortest distance to the obstacle reachable tube from the UAV position $\boldsymbol{p}(t)$, ρ_0 is the distance threshold for the repulsive field and α_i is a positive constant. Then the repulsive force $F_O(\boldsymbol{p}(t))$ is equal to the negative gradient of $W_O(\boldsymbol{p}(t))$:

$$F_O(\boldsymbol{p}(t)) = \alpha_i \left(\frac{1}{\rho(\boldsymbol{p}(t))} - \frac{1}{\rho_0}\right) \frac{1}{\rho(\boldsymbol{p}(t))^2} \nabla\rho(\boldsymbol{p}(t)) \tag{20}$$

The attractive potential field $W_G(\boldsymbol{p}(t))$ to go to the goal position is calculated as follows:

$$W_G(\boldsymbol{p}(t)) = \frac{1}{2}\zeta_i(\|\boldsymbol{p}(t) - \boldsymbol{p}_g\|^2) \tag{21}$$

where $\boldsymbol{p}_g$ is the position of the goal and ζ_i is a positive constant. The attractive force is the gradient of the attractive field at $\boldsymbol{p}(t)$:

$$F_G(\boldsymbol{p}(t)) = -\zeta_i(\boldsymbol{p}(t) - \boldsymbol{p}_g) \tag{22}$$

Finally, the UAV moves with the combination of repulsive and attractive forces:

$$F(\boldsymbol{p}(t)) = F_G(\boldsymbol{p}(t)) + F_O(\boldsymbol{p}(t)) \tag{23}$$

As soon as $d_o(t_p) \ge \xi$ the UAV switches back to the self/event-triggered scheduling policy presented above.

7.4 Simulation Results

The case study investigated in this section is a UAV waypoint navigation through a simple environment with one mobile obstacle. We consider a similar UAV described in Section 6.5, with sensor range of 20m. The same wind disturbance described in Section 6.5 is used during these simulations.

In the simulations shown in Fig. 21, the UAV is tasked to go to a goal point $\boldsymbol{p}_g = [10, 10]$m shown by a green circle. The red circle represents a dynamic obstacle which and both the vehicle and the obstacle move towards the center of the environment to reach their goal. As the obstacle moves faster than the UAV, their trajectories do not collide with each other. The obstacle and UAV positions before, near and after the intersection point of their paths are displayed in the first three figures in Fig. 21. As can be seen, since the obstacle passes the intersection point before the UAV, the UAV doesn't perform any avoidance action and it keeps following its originally planned trajectory. The replanning occurs only 3 times at the points shown by black '×' symbols in Fig. 21d. At the points shown by magenta 'diamond' symbols, the UAV checks its sensors, updates its reachable sets but it does not replan its trajectory. In this simulation, the average and maximum deviation from the desired trajectory is recorded as 6.36cm and 9.53cm respectively.

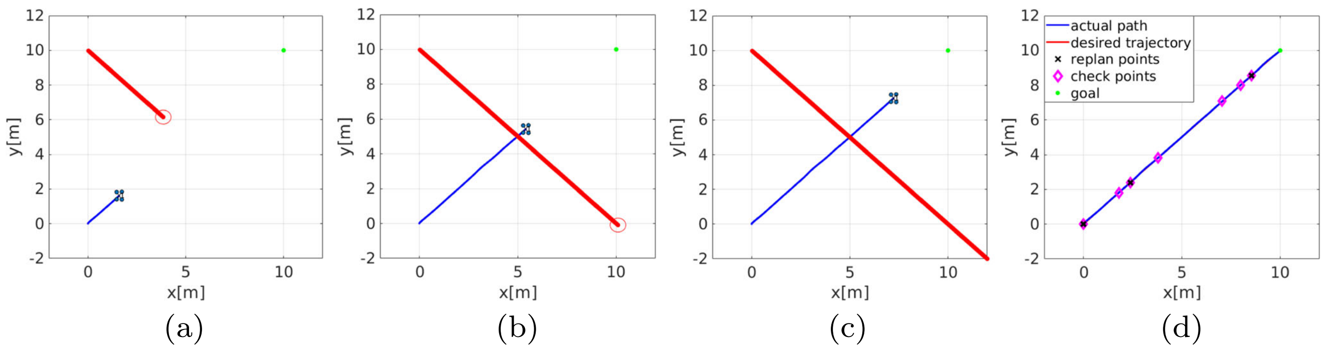


Fig. 21 Simulation Results in which the paths of the UAV and the obstacle intersects but collision doesn't happen because the obstacle passes the intersecting point earlier than the UAV. **a** Before the intersection point. **b** Near the intersection point. **c** After the intersection point. **d** Complete path of the UAV

In the case presented in Fig. 22, the UAV is tasked to the same goal location $\boldsymbol{p}_g = [6, 10]$. A dynamic obstacle starting from a different position move towards the path of the UAV and this case a collision would occur at the intersection point if the trajectory of UAV is not replanned. The first three figures in Fig. 22 shows the movement of UAV and the obstacle. When the distance between the UAV and the obstacle gets smaller than the threshold, the obstacle avoidance behavior is triggered by our event-triggered replanning approach and the UAV performs an obstacle avoidance maneuver. Once the obstacle is passed and far away, the UAV original self-triggered planning approach moving towards the goal. In this case, the UAV replans its trajectory only 3 times and checks its sensors at the points shown by magenta 'diamond' symbols in Fig. 22d. As can be noticed, while avoiding the obstacle, the UAV periodically checks its sensor. In this simulation, the average and maximum deviation from the desired trajectory is recorded as 4.02cm and 9.01cm respectively.

In Fig. 23, the UAV is tasked to go to a goal point $\boldsymbol{p}_g = [10, 10]$m. A dynamic obstacle approaches the UAV from the opposite direction. The first figure in Fig. 23 shows the paths of the obstacle and the UAV before reaching the intersection point. Similar to the previous case, as they get closer to each other, our event-triggered replanning approach triggers a collision avoidance behavior. After the UAV passes the obstacle, as shown in Fig. 23c, it goes back to the original self-triggered behavior towards its goal. In this case, the UAV replans its trajectory only 4 times at the black '$\times$' marks in Fig. 23d. In this simulation, the average and maximum deviation from the desired trajectory is recorded as 5.25cm and 9.76cm respectively.

7.5 Experimental Results

The self/event-triggered scheduling and replanning approach in dynamic environments was validated using two AscTec Hummingbird quadtrotor UAVs. The second UAV has a predefined trajectory to follow and the first UAV needs to update its trajectory according to our self-triggered scheduling and replanning approach to avoid the second UAV when necessary. The framework followed in this experiment is shown in Fig. 24. Similar to the previous sections, the reachable tubes for the first UAV are calculated using Matlab ellipsoidal toolbox [10] and both UAVs are controlled using ROS framework. A Vicon motion capture

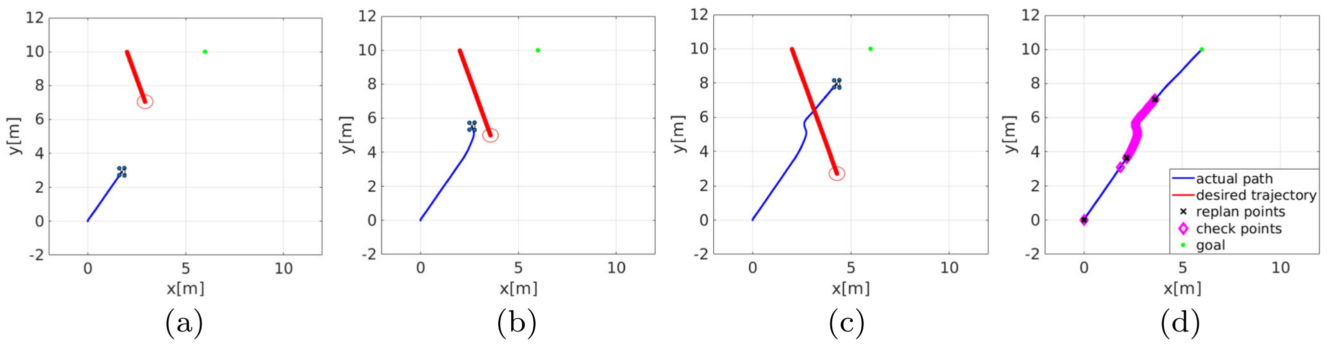


Fig. 22 Simulation results of collision avoidance. The obstacle and the UAV could collide with each other without replanning. **a** Before the intersection point. **b** Near the intersection point. **c** After the intersection point. **d** Complete path of the UAV

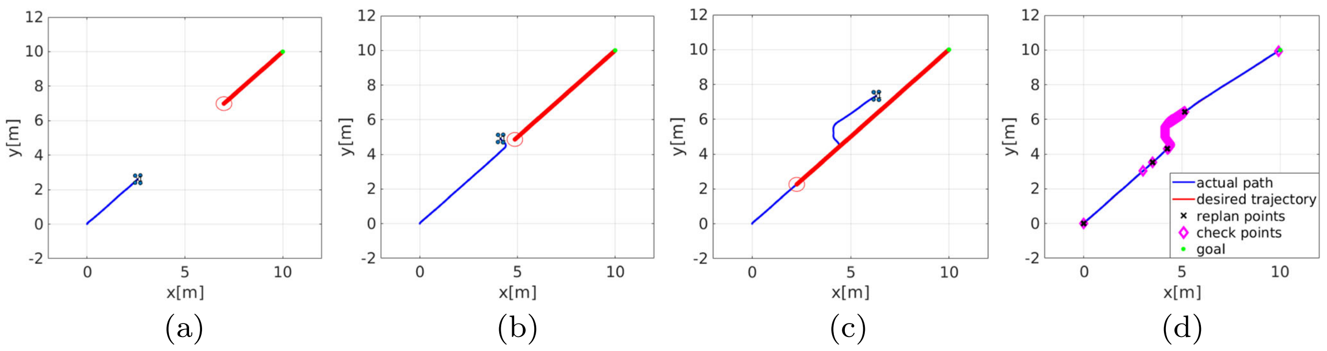


Fig. 23 Simulation results of collision avoidance in which the obstacle and the UAV move towards each others. **a** Before the intersection point. **b** Near the intersection point. **c** After the intersection point. **d** Complete path of the UAV

system was used to track the position and velocity of both UAVs at the scheduled times.

In Fig. 25a, a sequence of overlapped snapshots of the two quadrotors is shown. The first UAV starts its motion at $(-2.0, 0.0, 1.0)$m and it is tasked to navigate to its goal position at $(2.0, 0.0, 1.0)$m and the second UAV starts its motion at $(0.0, 1.0, 1.0)$m and it is tasked to go to its goal position at $(0.0, -1.0, 1.0)$m. During the experiments, it is assumed that the direction of the movement of the second UAV is known and it does not change over time. Both UAVs travel with average velocity 0.3m/s in this case. Since the trajectory of the second UAV is shorter, it passes the intersection point before the first UAV. Therefore, the first UAV doesn't need to update its trajectory to avoid the second one. In Fig. 25c, we compare the desired and actual trajectories of the two UAVs during the mission. The first UAV checks for the position of the second UAV only 7 times at the points shown by black '$\times$' symbols. At these points, since the first UAV is following its trajectory closely, it keeps following the existing trajectory and it recalculates the reachable tubes of both vehicles. The average and maximum deviation from its desired trajectory are recorded as 5.28cm and 12.41cm.

In Fig. 25b, both UAVs have the same initial and goal positions as in the previous case. The first UAV moves with the average velocity 0.3m/s and the second UAV moves with 0.05m/s. In this case, a collision between two UAVs would have occurred but the first UAV adapts its trajectory to avoid the second UAV to prevent collision when the distance between the two vehicles is less than $\rho_0 = 1.5$m and when they are approaching each others. In Fig. 25d, we compare the desired and actual trajectories of the two UAVs during the mission. When the reachable sets of the two UAVs collides and when they are close to each other, the first UAV monitors the position of the second UAV at points shown by magenta color, which happens 122 times, while following the adapted trajectory. Similar to the previous case, at the points shown with black $\times$ symbols, the UAV monitors the position of the second UAV only 8 times and it recalculates the reachable tubes of both vehicles. The average and maximum deviation of the first UAV from its desired trajectory are recorded as 2.88cm and 8.75cm in this case. If instead, traditional periodic monitoring was used, the deviation would have been less, however it would have required to monitor the position of the second UAV 2225 times as opposed to 130 times with our approach.

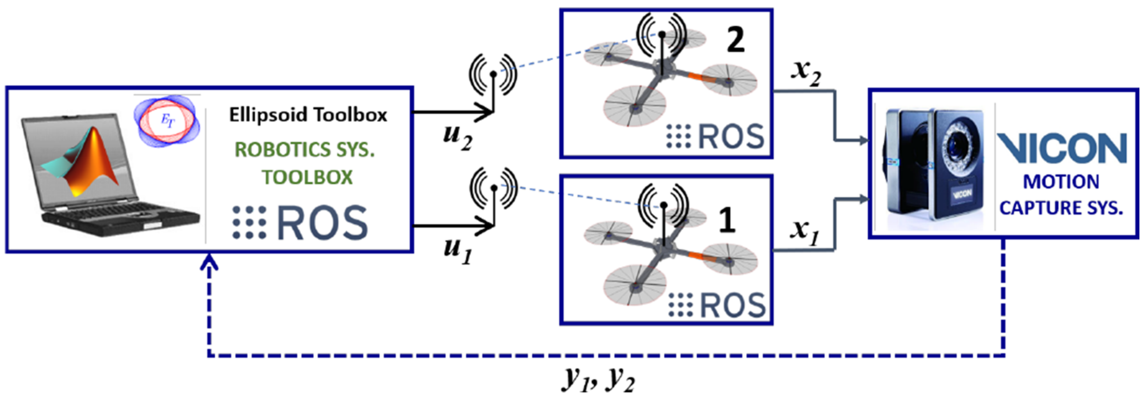


Fig. 24 Framework of the experiment setup

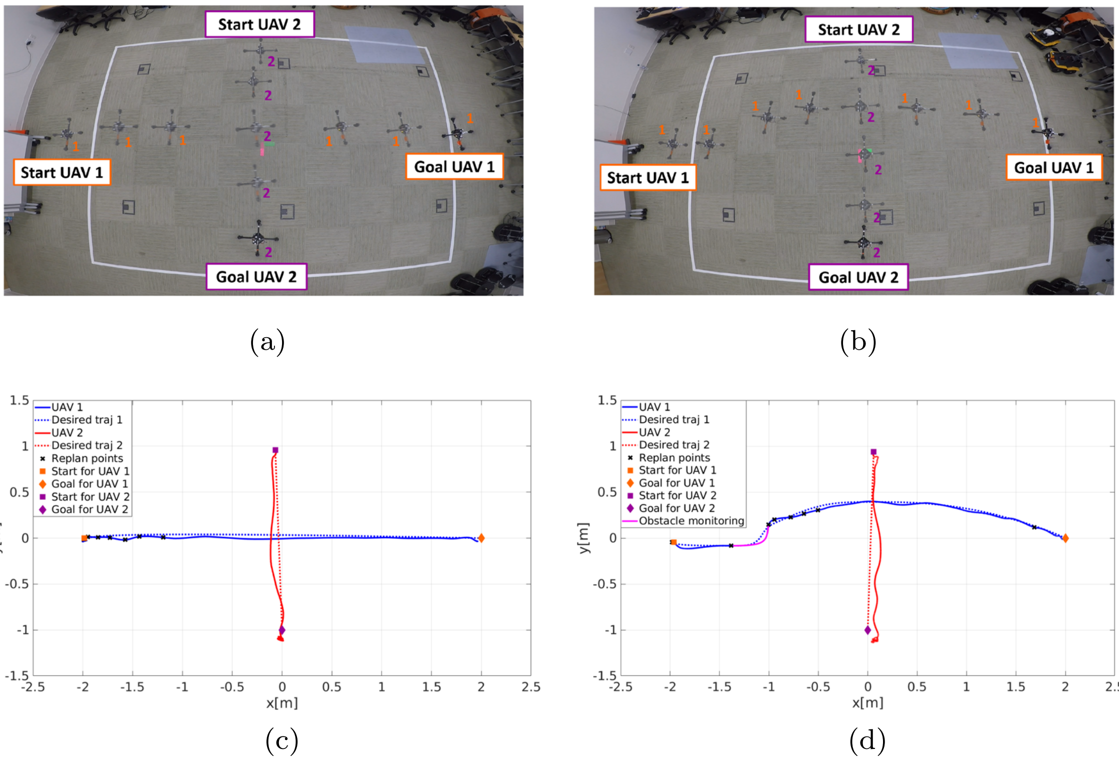


Fig. 25 Comparison of experimental results with self/event-triggered approach in dynamic environments. **a** Overlapped sequence of screenshots for non-colliding trajectories. **b** Overlapped sequence of screenshots for colliding trajectories. **c** Trajectories of the both UAVs for noncolliding trajectories. **d** Trajectories of the both UAVs for colliding trajectories

8 Conclusions and Future Work

In this work, we have presented an adaptive scheduling and replanning framework for UAV operations in cluttered and dynamic environments. The future states of the system under bounded external disturbances and system noises are computed by leveraging reachability analysis. Reachable sets are utilized to schedule next sensor monitoring and replanning times while guaranteeing safety and liveness. We also presented a computationally effective way of updating the reachable tubes based on the monitored system state to further minimize the replanning and sensor checking operations. The speed of the vehicle is also adapted using a curvature based approach to limit the deviation while the system is moving in cluttered environments.

In our future work, we plan to consider more realistic dynamics for the moving obstacles and use a similar framework to optimize energy consumption by leaving sensors in idle mode when not in use. Currently we are also exploring machine learning techniques for fast reachability analysis for more complex system dynamics at runtime, and to perform replanning for safety guaranteed operations in partially known environments. We also plan to extend this approach to systems with uncertain models, and to cases in which the model of the system changes over time.

Acknowledgements This material is based upon work supported by the Air Force Research Laboratory and the Defense Advanced Research Projects Agency under Contract No. FA8750-18-C-0090, ONR under agreement number N000141712012, and NSF under grant #1816591. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Air Force Research Laboratory (AFRL), the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, or the United States Government.

References

1. Ahmadzadeh, A., Jadbabaie, A., Kumar, V., Pappas, G.J.: Stable multi-particle systems and application in multi-vehicle path planning and coverage. In: 46th IEEE Conference on Decision and Control, pp. 1467–1472 (2007)
2. Al-Kaff, A., Meng, Q., Martín, D., de la Escalera, A., Armingol, J.M.: Monocular vision-based obstacle detection/avoidance for unmanned aerial vehicles. In: IEEE Intelligent Vehicles Symposium (IV), pp. 92–97 (2016). https://doi.org/10.1109/IVS.2016.7535370
3. Bezzo, N., Griffin, B., Cruz, P., Donahue, J., Fierro, R., Wood, J.: A cooperative heterogeneous mobile wireless mechatronic system. IEEE/ASME Trans. Mechatron. **19**(1), 20–31 (2014)
4. Bezzo, N., Mohta, K., Nowzari, C., Lee, I., Kumar, V., Pappas, G.: Online planning for energy-efficient and disturbance-aware uav operations. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5027–5033 (2016)

5. Bouzid, Y., Bestaoui, Y., Siguerdidjane, H.: Quadrotor-uav optimal coverage path planning in cluttered environment with a limited onboard energy. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 979–984 (2017). https://doi.org/10.1109/IROS.2017.8202264
6. Ding, J., Gillula, J.H., Huang, H., Vitus, M.P., Zhang, W., Tomlin, C.J.: Hybrid systems in robotics. IEEE Robot. Autom. Mag. **18**(3), 33–43 (2011)
7. Egerstedt, M., Johansson, K.H., Sastry, S., Lygeros, J.: On the regularization of zeno hybrid automata. Syst. Control Lett. **38**, 141–150 (1999)
8. Faust, A., Chiang, H.T., Rackley, N., Tapia, L.: Avoiding moving obstacles with stochastic hybrid dynamics using pearl: Preference appraisal reinforcement learning. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 484–490 (2016)
9. Gageik, N., Benz, P., Montenegro, S.: Obstacle detection and collision avoidance for a uav with complementary low-cost sensors. IEEE Access **3**, 599–609 (2015). https://doi.org/10.1109/ACCESS.2015.2432455
10. Kurzhanskiy, A.A., Varaiya, P.: Ellipsoidal toolbox (et). In: Proceedings of the 45th IEEE Conference on Decision and Control, pp. 1498–1503 (2006)
11. Lin, Y., Saripalli, S.: Sampling based collision avoidance for uavs. In: American Control Conference (ACC), 2016, pp. 1353–1358. IEEE (2016)
12. Lin, Y., Saripalli, S.: Sampling-based path planning for uav collision avoidance. IEEE Trans. Intell. Transport. Syst. **PP**(99), 1–14 (2017). https://doi.org/10.1109/TITS.2017.2673778
13. Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C.J., Kumar, V.: Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. IEEE Robot. Autom. Lett. **2**(3), 1688–1695 (2017). https://doi.org/10.1109/LRA.2017.2663526
14. Mac, T.T., Copot, C., Hernandez, A., Keyser, R.D.: Improved potential field method for unknown obstacle avoidance using uav in indoor environment. In: IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 345–350 (2016)
15. Majumdar, A., Tedrake, R.: Funnel libraries for real-time robust feedback motion planning. Int. J. Robot. Res. **36**(8), 947–982 (2017)
16. Malone, N., Lesser, K., Oishi, M., Tapia, L.: Stochastic reachability based motion planning for multiple moving obstacle avoidance. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control, HSCC '14, pp. 51–60. ACM, New York (2014). https://doi.org/10.1145/2562059.2562127
17. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: IEEE International Conference on Robotics and Automation, pp. 2520–2525 (2011)
18. Michael, N., Mellinger, D., Lindsey, Q., Kumar, V.: The grasp multiple micro-uav testbed. IEEE Robot. Autom. Mag. **17**(3), 56–65 (2010)
19. Odelga, M., Stegagno, P., Bulthoff, H.H.: Obstacle detection, tracking and avoidance for a teleoperated uav. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2984–2990 (2016)
20. Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., Galceran, E.: Continuous-time trajectory optimization for online uav replanning. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5332–5339 (2016)
21. Otte, M., Frazzoli, E., RRT, X.: Real-time motion planning/replanning for environments with unpredictable obstacles. In: Algorithmic Foundations of Robotics XI, pp. 461–478. Springer (2015)
22. Pereira, G.A.S., Choudhury, S., Scherer, S.: A framework for optimal repairing of vector field-based motion plans. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 261–266 (2016). https://doi.org/10.1109/ICUAS.2016.7502525
23. Roelofsen, S., Gillet, D., Martinoli, A.: Reciprocal collision avoidance for quadrotors using on-board visual detection. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4810–4817 (2015)
24. Roelofsen, S., Martinoli, A., Gillet, D.: 3d collision avoidance algorithm for unmanned aerial vehicles with limited field of view constraints. In: IEEE 55th Conference on Decision and Control (CDC), pp. 2555–2560 (2016)
25. Santos, M.C.P., Rosales, C.D., Sarcinelli-Filho, M., Carelli, R.: A novel null-space-based uav trajectory tracking controller with collision avoidance. IEEE/ASME Trans. Mechatron. **22**(6), 2543–2553 (2017). https://doi.org/10.1109/TMECH.2017.2752302
26. Santos, M.C.P., Santana, L.V., Brandão, A.S., Sarcinelli-Filho, M.: Uav obstacle avoidance using rgb-d system. In: International Conference on Unmanned Aircraft Systems (ICUAS), pp. 312–319 (2015). https://doi.org/10.1109/ICUAS.2015.7152305
27. Singh, S., Majumdar, A., Slotine, J.J., Pavone, M.: Robust online motion planning via contraction theory and convex optimization. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 5883–5890 (2017). https://doi.org/10.1109/ICRA.2017.7989693
28. Sun, J., Tang, J., Lao, S.: Collision avoidance for cooperative uavs with optimized artificial potential field algorithm. IEEE Access **5**, 18382–18390 (2017). https://doi.org/10.1109/ACCESS.2017.2746752
29. Vinod, A.P., Homchaudhuri, B., Oishi, M.M.K.: Forward stochastic reachability analysis for uncontrolled linear systems using fourier transforms. arXiv:abs/1610.04550 (2016)
30. Yang, K., Sukkarieh, S.: An analytical continuous-curvature path-smoothing algorithm. IEEE Trans. Robot. **26**(3), 561–568 (2010). https://doi.org/10.1109/TRO.2010.2042990
31. Yel, E., Bezzo, N.: Reachability-based adaptive uav scheduling and planning in cluttered and dynamic environments. In: Workshop on Informative Path Planning and Adaptive Sampling at ICRA (2018). http://robotics.usc.edu/wippas/program.html
32. Yel, E., Lin, T.X., Bezzo, N.: Reachability-based self-triggered scheduling and replanning of uav operations. In: NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 221–228 (2017)
33. Yel, E., Lin, T.X., Bezzo, N.: Self-triggered adaptive planning and scheduling of uav operations. In: IEEE International Conference on Robotics and Automation (ICRA) (2018)
34. Zhou, Y., Raghavan, A., Baras, J.S.: Time varying control set design for uav collision avoidance using reachable tubes. In: IEEE 55th Conference on Decision and Control (CDC), pp. 6857–6862 (2016)

Esen Yel is currently a Ph.D. student in System Engineering at the Autonomous Mobile Robots Laboratory and the Link Lab at the University of Virginia under the supervision of Prof. Nicola Bezzo. She received the B.S. and M.S degrees in Electrical and Electronics Engineering from Boğaziçi University, Istanbul, Turkey in 2014 and 2016 respectively. Her current research interests include safe motion planning, reachability analysis, runtime monitoring, and self-triggered scheduling and control of autonomous aerial vehicles.

Tony X. Lin is currently a Ph.D. student in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. He earned his M.S. degree in Computer Engineering in May 2018 and B.S. degree in Mechanical Engineering from the University of Virginia in May 2016. He is currently conducting research on bio-inspired decentralized autonomy algorithms.

Nicola Bezzo is an Assistant Professor with the Department of Engineering Systems and Environment and the Department of Electrical and Computer Engineering at the University of Virginia (UVA). Prior to joining UVA in 2016, he was a Postdoctoral Researcher at the PRECISE Center, in the Department of Computer and Information Science at the University of Pennsylvania (UPenn) where he worked on topics related to robotics and cyber-physical systems security. He received a Ph.D. degree in Electrical and Computer Engineering from the University of New Mexico where he focused on the development of theories for motion planning of heterogeneous aerial and ground robotic systems under communication constraints. Prior to his Ph.D., he received both M.S. and B.S. degrees in Electrical Engineering with honors (summa cum laude) from Politecnico di Milano, Italy. At UVA he leads the Autonomous Mobile Robots Lab with research focused on safe and resilient motion planning and control of autonomous vehicles under uncertainties. He is also part of the Link Lab.