2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Self-triggered Adaptive Planning and Scheduling of UAV Operations

Esen Yel, Tony X. Lin and Nicola Bezzo

Abstract— **Modern unmanned aerial vehicles (UAVs) rely on constant periodic sensor measurements to detect and avoid obstacles. However, constant checking and replanning are time and energy consuming and are often not necessary especially in situations in which the UAV can safely fly in uncluttered environments without entering unsafe states. Thus, in this paper, we propose a self-triggered framework that leverages reachability analysis to schedule the next time to check sensor measurements and perform replanning while guaranteeing safety under noise and disturbance effects. Further, we relax sensor checking and motion replanning operations by leveraging a risk-based analysis that determines the likelihood to reach undesired states over a certain time horizon. We also propose an online speed adaptation policy based on the planned trajectory curvature to minimize drift from the desired path due to the system dynamics. Finally, we validate the proposed approach with simulations and experiments for a quadrotor UAV motion planning case study in a cluttered environment.**

I. Introduction

Over the last few years, unmanned aerial vehicles (UAVs) have gained a lot of attention and popularity in both civilian and military operations. Some of the applications in which UAVs are particularly suitable over other cyber-physical system technologies are for example delivery services, inspection, search and rescue, reconnaissance, and surveillance.

Most of these applications require the UAV to travel to one or multiple goals following trajectories, while sharing the airspace with other objects (i.e., obstacles and other vehicles). In order to guarantee safety under different uncertainties like disturbance and noise, traditional motion planning techniques employ periodic sensor checking to maintain a desired trajectory (e.g., by means of IMU and GPS sensors) and to detect and avoid obstacles (e.g., by using lidar or camera sensors), and periodic planning and replanning operations to adapt the motion of the UAV accordingly. However, this periodicity constraint could be relaxed if the system can assess and predict future states under different uncertainties; this would minimize unnecessary checking and replanning operations which are energy and computation consuming. Let us consider the pictorial representation in Fig. 1: a UAV flying in an obstacle free environment needs less computation time and can move faster than the same UAV navigating in a cluttered environment, in which it will need more computation time to elaborate sensor data and plan its trajectory to avoid obstacles. Its speed could also be adapted while going through different environments, moving faster in open and obstacle free areas while slowing down in cluttered spaces since it will need to closely follow a trajectory between obstacles which is usually curved.

Esen Yel, Tony X. Lin and Nicola Bezzo are with the Departments of Systems and Information Engineering and Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA {esenyel, txl5gd, nbezzo}@virginia.edu

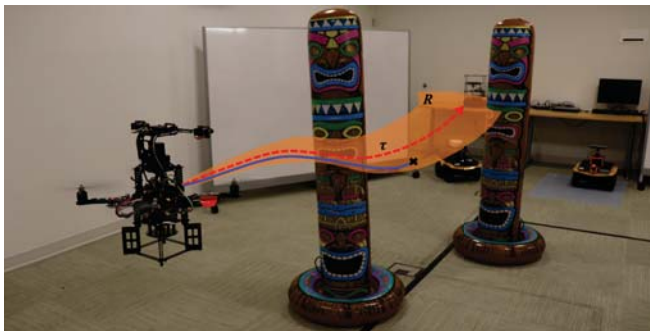


Fig. 1. Pictorial representation of the proposed problem in which a quadrotor must navigate through a cluttered environment without constant sensor checking and adapt its speed accordingly. At t_p, our proposed framework computes the desired trajectory τ (red dashed curve) and a reachable set R (the orange shaded region) to decide the next monitoring time t_{p+1} that guarantees safety and liveness constraints. The blue solid curve represents the actual trajectory followed by the quadrotor which deviates from the desired trajectory due to disturbance, noise, and jitter in the system. A similar setup is considered during experiments.

To deal with these considerations, we propose an online adaptive framework to minimize both computation and energy consumptions on mobile aerial robotic applications while guaranteeing safety (i.e., something "bad" will never happen) and liveness (i.e., something "good" will eventually happen). The proposed framework is based on the following consideration: *periodic checking, sensor monitoring and replanning are not always necessary, and they can be relaxed if we can predict and plan the operation of a robot correctly*. This paper targets to solve the following challenges:

- how to minimize sensor checking and replanning operations, while guaranteeing safety and liveness properties;
- how to plan and replan a UAV operation, including adapting its speed, while solving the previous challenge.

To this end, our proposed approach uses a combination of reachability analysis, self-triggered scheduling, replanning policies, and risk analysis to: i) predict the future states of the system, ii) schedule next sensor monitoring time, and iii) replan the motion, and adapt the speed of a UAV that is navigating in cluttered and unknown environments under the effect of noise and unforeseen disturbance phenomena.

II. Related Work

At the core of UAV technologies, motion planning in particular targeted to obstacle and collision avoidance is one of the most important capabilities for safe and persistent autonomous operations. In the literature we find numerous works using different perception techniques to deal with this problem [1], [2], [3]. In all of these works, obstacle detection is achieved via high-frequency periodic sensor data acquisition and elaboration, which can be too conservative when the environment is free from obstacles.

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

In this work, we leverage reachability analysis to predict the future states of the system and relax the sensor checking and replanning operations while guaranteeing safety and liveness constraints. In the literature, there are a few works in which reachability analysis is utilized for UAV operations. In [4], reach-avoid sets are calculated using Hamilton-Jacobi-Isaacs equations and are utilized to replan the motion control inputs of an autonomous helicopter under disturbances. In [5], forward stochastic reachable sets are constructed and used to plan the motion of a robot avoiding dynamic obstacles. Reachable sets are also utilized to detect the possible collisions [6], [7]. In [8], Robust Control Invariant (RCI) tubes, defined as regions in which the system state is guaranteed to stay, are used to create a collision free trajectory. Differently from these previous works, we use reachable sets to: i) detect possible obstacle collisions over a future time horizon and ii) schedule next sensor checking and replanning events considering disturbance and uncertainties in the environment. The key aspect and novelty in our work is that we schedule next sensor checking and replanning times aperiodically when they are mostly needed. In our previous work [9], we introduce an online self/event-triggered scheduling policy to decide the best time in the future to replan the motion of a UAV while minimizing computational energy. In [10], we proposed a self-triggered scheduling policy to navigate a UAV via an open-loop controller in a cluttered environment and schedule next sensor checking time. In this paper we build on these previous works and propose a framework to decide the next sensor checking time and to relax unnecessary replanning operations using a risk-based approach. We also introduce a policy to adapt the speed of the UAV at replanning time based on the specific trajectory that the UAV needs to follow. This last problem is solved leveraging the results on trajectory curvature analysis presented in [11]. Different from these previous works in which a curvature analysis was performed to generate optimal trajectories, in our case the trajectory is first selected and the optimal speed is computed based on the curvature of the desired trajectory, to avoid drifts due to the UAV dynamics.

Overall, we contribute to the current literature on UAV motion planning by: i) providing a novel self-triggered scheduling approach using reachability analysis, ii) defining a risk-based trajectory replanning scheduling policy and iii) adapting the speed of the UAV based on the curvature of the path to follow.

The rest of the paper is organized as follows: in Section III, we formally define the problem and in Section IV, we define the UAV motion, noise, and disturbance models. Our self-triggered scheduling, risk-based replanning, and curvature based speed adaptation techniques are then presented in Section V, and validated with simulations and experiments in Sections VI and VII, respectively. Finally, we draw conclusions and discuss future work in Section VIII.

III. Problem Formulation

In this work we are interested in finding an online policy to schedule next sensor monitoring and replanning operations. Formally, the problem can be casted as follows:

Problem 1: *Self-triggered Replanning*: A UAV has the objective to visit one or more goal locations in a cluttered environment. The obstacle positions are detected using onboard range sensors and a trajectory with the desired goals is generated online according to the detected obstacles. The aim is to find a scheduling policy to determine the next sensor monitoring and replanning time t_{p+1}, such that the following safety and liveness requirements are satisfied between two consecutive replanning times:

1) *Safety Constraint:* The UAV should avoid collisions with obstacles between two consecutive replanning times, or mathematically:
$$\|\boldsymbol{p}(t) - \boldsymbol{o}_i\| > 0, \forall t \in [t_p, t_{p+1}], \forall i \in [0, n_o] \quad (1)$$
in which $\boldsymbol{p}(t) = \begin{bmatrix} x, y \end{bmatrix}^\mathsf{T}$ is the position of the UAV and $\boldsymbol{o}_i = \begin{bmatrix} o_x, o_y \end{bmatrix}^\mathsf{T}$ is the position of the i^{th} obstacle in the $x-y$ plane with n_o the number of obstacles in the environment.
2) *Liveness:* The UAV should stay within a certain proximity of the planned trajectory:
$$\|\boldsymbol{p}(t) - \boldsymbol{p}_\tau(t)\| \leq \lambda_d, \forall t \in [t_p, t_{p+1}] \quad (2)$$
where $\boldsymbol{p}_\tau(t)$ is the desired position of the UAV on the trajectory at time t, and λ_d is the allowed deviation threshold.

To minimize unnecessary replanning operations we introduce the following problem:

Problem 2: *Replanning Relaxation*: Given the assumptions in Problem 1, we want to find a policy to decide the 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 defined in (1) and (2) respectively hold between t_p and t_{s+1}.

In order to avoid collisions with obstacles, a UAV may need to follow a path with very high curvature which may lead to a deviation from the desired trajectory, especially when traveling at high speed. Formally:

Problem 3. *Speed Adaptation*: Given the UAV dynamics and assumptions listed in the previous problems, at replanning time t_p, after defining a trajectory, we would like to find a policy to determine the optimal 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 trajectory having maximum curvature κ_m, ξ_t is a user defined average deviation threshold, and v_{min} and v_{max} are the minimum and maximum UAV speeds, respectively.

IV. System Models

In this section, we present the dynamical system, noise and disturbance models that will be used to control the UAV and compute reachable sets. Here, we follow the architecture shown in Fig. 2 and describe its main elements as follows.

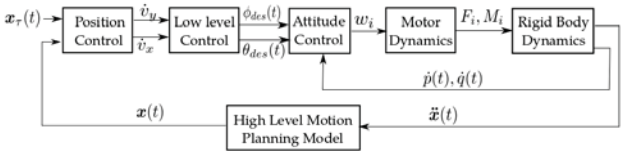


Fig. 2. UAV planning and control architecture

A. Quadrotor Dynamical Model

A quadrotor can be modeled using the following 12^{th} order state vector:

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

where $\boldsymbol{p}_q = [x\ y\ z]^{\mathsf{T}}$ is the world frame position, v_x, v_y and v_z are the world frame velocities, ϕ, θ and ψ are the roll, pitch and yaw Euler angles and ω_x, ω_y and ω_z are the body frame angular velocities [10], [9].

The dynamics of the quadrotor can be described as:

$$\dot{\boldsymbol{p}}_q^{\mathsf{T}} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}$$

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi \\ \cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} u_1$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

B. High-Level Motion Planning Model

The following high-level motion planning model is used to capture the quadrotor position and velocity evolutions in the $x-y$ plane. This simplified model will be used for the reachability analysis presented in the next section.

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

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

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

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

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

C. Position, Low Level, and Attitude Controls

Following the architecture in Fig. 2, the position and low-level controllers (implemented as a series of PID loops after linearizing (IV-A), [12]) generate the necessary angle inputs to the attitude control in order to follow the desired trajectory $\boldsymbol{x}_\tau$. The attitude controller along with the motor dynamics calculate the thrust and moment values that each rotor should provide for these angle inputs [9], [12]. Finally the rigid body dynamics generate the response of the quadrotor to these thrust and moment values in terms of acceleration. The position and low level controller errors are represented as input noise in the high level motion planning model (3). This is described in more detail in the following section.

D. Noise and Disturbance Model

When a quadrotor is flying, internal factors like sensor or actuator noises or external factors like wind disturbance may cause it to deviate from its planned behavior and they all should be taken into consideration for safety. In this work, we assume that wind disturbance, actuator noise, and state uncertainty are all bounded by ellipsoids ϵ and uniformly distributed. The state uncertainty caused by the sensor measurement noise is represented by $\eta_x \in \epsilon(0, N_x)$ which represents an ellipsoid centered in the origin and bounded by a shape matrix N_x. The sources of the input uncertainties η_u are the actuator and process noises resulting from the low-level controllers and the mechanical uncertainties from the rotors, gears, and propellers. The input uncertainty is assumed to be bounded by an ellipsoid, $\eta_u \in \epsilon(0, N_u)$ where N_u is the bound. The external disturbance $\boldsymbol{d}(t)$ is also centered around the origin and bounded by an ellipsoid $\boldsymbol{d}(t) \in \epsilon(0, N_d)$ where N_d is the disturbance bound.

V. Self-triggered Scheduling and Replanning

In this section, we describe our online scheduling and replanning framework whose architecture is shown in Fig. 3.

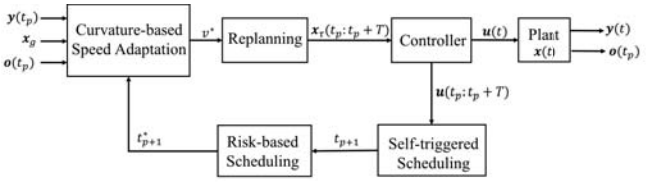


Fig. 3. Overall architecture followed for our risk-based self-triggered scheduling and replanning framework with speed adaptation.

To efficiently use the available resources, it is desirable to limit sensor monitoring and replanning actions to instances in which the system really needs attention. Unfortunately, the classical time-triggered control paradigm performs sensing and actuation actions periodically in time (without considering the state information) rather than when the system actually needs them. Instead of periodically replanning, we consider self-triggered scheduling policies that determine, based on currently available information, at what time in the future the UAV will need to interrogate its sensors and replan its motion. This idea is closely related to the work on self-triggered control presented in [13] in which the authors outline a technique that enables a controller to decide its next execution time while maintaining stability. We extend this methodology to consider the replanning case and instead of stability issues here we consider safety constraints.

Next replanning time decision will be taken considering the current state, actions, and the predicted future states calculated using reachability analysis.

A. Reachability Analysis for UAV Operations

A *reachable set* or *reach set* computed at time t_0 for a future time t_f and represented by $R(\boldsymbol{x}_0, \boldsymbol{u}(t), t_f)$ is an ellipsoid ϵ that contains all the states $\boldsymbol{x}$ reachable at a future time $t_f > t_0$ where the initial set $\boldsymbol{x}_0 \in \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ is bounded by an ellipsoid with center $\boldsymbol{x}_0$ and shape matrix $\boldsymbol{X}_0$ and the input $\boldsymbol{u}(t) \in \epsilon(\boldsymbol{u}(t), \boldsymbol{U})$ is bounded by an ellipsoid with center $\boldsymbol{u}(t)$ and shape matrix $\boldsymbol{U}$. The actual state $\boldsymbol{x}(t_f)$

and the estimated state $\tilde{\boldsymbol{x}}(t_f)$ will lie inside the reachable ellipsoid. The shape matrix $\boldsymbol{X}_0$ of the initial state ellipsoid depends on the state uncertainty η_x whereas the shape matrix of the input ellipsoid $\boldsymbol{U}$ depends on the input uncertainty η_u as explained in Section IV-D. $R_p(\boldsymbol{x}_0, \boldsymbol{u}(t), t_f)$ is the projection of the reachable sets on the position space and a reachable tube $R(\boldsymbol{x}_0, \boldsymbol{u}(t), [0, T])$ is the set of all reachable sets over the time interval $\Delta T = [0, T]$.

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

$$\begin{aligned} R^+(t, t_0, \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)) = (\boldsymbol{\Phi}(t, t_0)\epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0) \\ \oplus \int_{t_0}^{t} \boldsymbol{\Phi}(t, \zeta)\boldsymbol{B}\epsilon(\boldsymbol{u}(\zeta), \boldsymbol{U})d\zeta) \\ \ominus \int_{t_0}^{t} \boldsymbol{\Phi}(t, \zeta)(-\boldsymbol{G})\epsilon(0, N_d)d\zeta \end{aligned}$$

where $\boldsymbol{\Phi}(t, t_0) = e^{\boldsymbol{A}(t-t_0)}$ and the symbols $\oplus$ and $\ominus$ represent geometric sum and geometric difference respectively.

In this work, we calculate the reachable sets of the states of a quadrotor tracking a trajectory $\boldsymbol{x}_\tau(t_p : t_{p+1})$ generated over the interval $\Delta t_p = [t_p, t_{p+1}]$ where t_p is the current planning time and t_{p+1} is the next planning time, initially set to $t_p + T$ before the rescheduling operation. We generate minimum jerk trajectory [14], and an online simulator implementing a PD controller is used to find the acceleration inputs to track this trajectory as if there is no disturbance and noise in the environment. These computed inputs are used to calculate the reachable tubes. In Fig. 4, a position reeachable tube calculated for $\Delta t_p = 2$s is shown for a quadrotor moving in the x direction with sensor measurement noise $\eta_x = 0.001$ and input noise $\eta_u = 0.054$. Although the actual path of the quadrotor (blue dotted curve) is different from its desired trajectory (red star curve) due to wind disturbance $\boldsymbol{d} = [0.1, 0.1]$m/s, it is contained inside the reachable tube, since uncertainties and disturbances are taken into account during the reachability calculation.

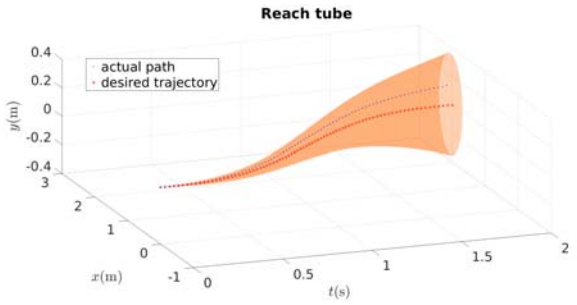


Fig. 4. The position reachable tube for a quadrotor following a straight line trajectory for 2s.

B. Self-Triggered Scheduling

In this section we describe a self-triggered scheduling policy to decide the next replanning time t_{p+1} leveraging the reachability analysis outlined in the previous section to guarantee safety between replanning operations.

The predicted first time that a collision may occur, t_c, is calculated using the position reachable tube presented in Section V-A as follows:

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

where $\boldsymbol{O} \in \mathbb{R}^{2 \times n_o}$ is the set of obstacle positions and n_o is the number of obstacles in the environment.

The earliest time that the UAV can leave the region covered by the field of view of its range sensor, denoted by t_l is calculated as follows:

$$t_l = \min(t_k | R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [t_p, t_p + T]) \not\subset \boldsymbol{r}(t_p)) \quad (5)$$

where $\boldsymbol{r}(t_p)$ is the set of all points in the $x-y$ plane covered by the range sensor of the UAV at the planning time t_p.

If there is a possibility to collide with an obstacle or leave the region covered by the range sensor, the UAV needs to check its state to decide whether replanning is necessary or not. Therefore, it needs to check its state either at time t_c, or at the time t_l, whichever is the minimum. If it is not possible to collide with any obstacle or leave the sensor field of view within the planned horizon T, then monitoring and replanning happen at time T.

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

where t_r is the amount of time necessary for the replanning calculation. Using this approach, we can guarantee to stay within the sensor field of view and avoid collisions with any obstacles, as formally described in the following lemma.

Lemma 1: Given t_p and t_{p+1} computed in (6), the UAV is guaranteed to stay within the region covered by the range sensor field of view and avoid collision with any obstacle detected at t_p.

Proof: By definition, $R(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [0, T])$ is the set of all states $\boldsymbol{x}$, such that there exists an input $\boldsymbol{u} \in \epsilon(\boldsymbol{u}(t_k), \boldsymbol{U})$ and an initial state $\boldsymbol{x}_0 \in \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ which steer the UAV from $\boldsymbol{x}_0$ to $\boldsymbol{x}$ in time t_k. $R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [0, T])$ is the projection of $R(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [0, T])$ onto the position space. Considering a time value t^* between t_p and t_{p+1}, $t_p < t^* < t_c$ and $t_p < t^* < t_l$, by the definitions of t_c in (4) and t_l in (5),

$$\begin{aligned} R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [t_p, t^*) \cap \boldsymbol{o} = \emptyset \\ R_p(\boldsymbol{x}_0, \boldsymbol{u}(t_k), t_k \in [t_p, t^*]) \subset \boldsymbol{r}(t_p) \end{aligned}$$

which proves that for $t_p < t^* < t_{p+1}$, there doesn't exist an input $\boldsymbol{u}(t^*) \in \epsilon(\boldsymbol{u}(t^*), \boldsymbol{U})$ and an initial state $\boldsymbol{x}_0 \in \epsilon(\boldsymbol{x}_0, \boldsymbol{X}_0)$ that steer the system from $\boldsymbol{x}_0$ to any state $\boldsymbol{x}$ inside an obstacle or outside the region covered by the range sensor ■

C. Risk Based Replanning

Reachable tubes are meant to capture the worst case scenario in terms of disturbance and noise. In fact, when the UAV is following its trajectory closely, replanning at each time that the reachable tube collides with an obstacle may be over-conservative. Therefore, replanning t_{p+1} can be postponed to a later t^*_{p+1}. In this section, we propose a risk-based strategy to decide whether replanning is necessary or not, and if necessary, when it should be scheduled. To this end, we define a risk measure, which is a combination of risk

of deviating from the desired trajectory (liveness) and risk of colliding with any obstacle (safety). For example, in the situation presented in Fig. 5, the UAV is deviating from its trajectory and getting closer to the obstacle whose position over time is shown by a black line. In this case, replanning might be necessary, because the risk of drifting and hitting an obstacle is high.

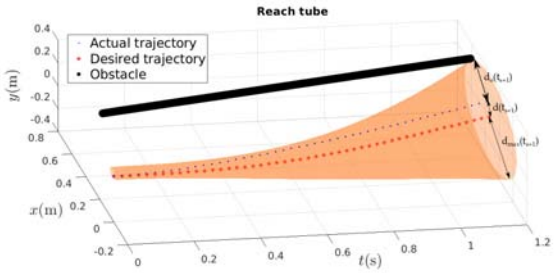


Fig. 5. Risk calculation at t_s depending on the state of the UAV.

Let us define t_s as the time in which the UAV is computing the risk. The risk associated with the deviation from the desired trajectory is calculated as follows:

$$r_d = \frac{d(t_s)}{d_{max}(t_s)} \tag{7}$$

where $d(t_s) = |\boldsymbol{p}(t_s) - \boldsymbol{p}_\tau(t_s)|$ is the distance between the position of the UAV $\boldsymbol{p}(t_s)$ and the desired position on the trajectory $\boldsymbol{p}_\tau(t_s)$ at time t_s. $d_{max}(t_s)$ is the maximum possible deviation from the trajectory at t_s which is equal to the radius of the position reachable set at that time.

The risk of obstacle collision is calculated as follows:

$$r_o = \min(1, \frac{k_r d(t_s)}{d_o(t_s)}) \tag{8}$$

where $d_o(t_s) = |\boldsymbol{p}(t_s) - \boldsymbol{o}^*|$ is the distance between the position of the UAV, $\boldsymbol{p}(t_s)$, and the position of the closest obstacle $\boldsymbol{o}^*$ and k_r is a constant parameter that in our case was set to 10. Finally the total risk is calculated as follows:

$$r = \alpha r_d + \beta r_o \quad \alpha, \beta \in \mathbb{R}^+,\ \alpha + \beta = 1 \tag{9}$$

where α and β are weight factors that depend on the importance that the user wants to pose to safety vs. liveness.

When the UAV is deviating a lot from the desired trajectory or is very close to the obstacle, the risk exceeds the maximum allowable threshold, which triggers a replanning operation because it becomes too risky for the UAV to continue. When the risk is smaller than a minimum threshold, then it can continue until the end of its planning horizon or sensor field of view. If the risk is in between two thresholds, then the next replanning time is scheduled before it leaves the region covered by its range sensor or before the end of its planning horizon. Putting everything together, next replanning time t^*_{p+1} is calculated as follows:

$$\begin{cases} t^*_{p+1} = t_s, & \text{if } r > \rho^+ \\ t^*_{p+1} = T^*, & \text{if } r < \rho^- \\ t_{s+1} = t_s + (1-r)(T^* - t_s) & \text{if } \rho^- \le r \le \rho^+ \end{cases} \tag{10}$$

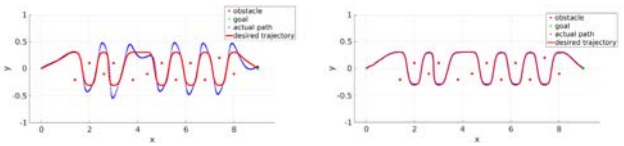
where T^* is the time that the UAV is expected to leave the region covered by range sensor or the planning horizon, whichever is minimum. ρ^- and ρ^+ are the minimum and maximum risk thresholds respectively. When the risk is between ρ^- and ρ^+, a *Zeno phenomenon* [15] may occur and lead to infinite scheduling of monitoring times between t_s and T^*. In order to prevent such behavior, we consider the sampling time δt and include the following constraint:

$$t_{s+1} = T^*, \quad \text{if } (1-r)(T^* - t_s) < \delta t \tag{11}$$

Equation (11) shows that when state checking time becomes very close to the replanning time, the UAV replans its motion again at T^*, preventing infinite number of state checks and risk calculations caused by the Zeno phenomenon.

D. Curvature Based Speed Adaptation

In cluttered environments, a UAV may need to follow winding trajectories with high curvatures, to avoid obstacles along its path. Although a UAV can follow these trajectories very closely with low speeds, tracking becomes harder with high speeds. In contrast, trajectories with low curvatures can be closely tracked even at high speeds. For example, in Fig. 6, a UAV is tasked to follow a trajectory (red colored continuous curve) with different speeds: $v = 1$m/s in Fig. 6(a) and $v = 0.25$m/s in Fig. 6(b). Intuitively and as can be clearly noted by comparing the two figures, tracking is better achieved when the speed is lower.



(a) (b)

Fig. 6. Comparison between drifts on a path with high curvature. In (a), the UAV moves with v_{max} = 1m/s drifting on average 9.05cm. In (b), the UAV moves with v_{max} = 0.25m/s drifting on average 1.5cm.

To decide the optimal speed, the UAV adapts its speed according to the maximum curvature of the computed obstacle avoidance trajectory at replanning times. The curvature of the trajectory is calculated based on [11], as follows:

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

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 to avoid obstacles.

The average deviation d_{avg} from the desired trajectory can be over-approximated using the maximum curvature of the trajectory and the speed of the quadrotor as follows:

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

where Ω is a parameter obtained experimentally for the specific UAV (in our case $\Omega = 15$) and κ_m is the maximum curvature over the trajectory: $\kappa_m = \max_{i \in [1, n-1]} \kappa_i$.

The UAV's optimal velocity v^* is then selected such as to keep the average deviation under a certain threshold and make the robot reach its goal as fast as possible:

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

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

VI. Simulations

The case study investigated in this work is a UAV waypoint navigation in a cluttered environment under sensor and process noises, and wind disturbance. More specifically, here we consider a quadrotor UAV equipped with a localization sensor (e.g. GPS) and a range sensor (e.g. a lidar) with a limited 10m range and a 180° field of view. The UAV is tasked to visit each corner of a square trajectory with 30m sides starting at the origin with zero velocity. A constant wind disturbance of magnitude $\boldsymbol{d} = [-0.07, 0.07]^\mathsf{T}$m/s is present everywhere in the environment and unknown a priori. In Fig. 7(a), the path of a quadrotor controlled in open-loop without checking neither the localization nor the range sensors is shown. The maximum speed of the quadrotor is $v = 0.5$m/s and just for this case it is assumed that the exact position of all the obstacles are known a priori. As expected, due to the disturbance and the lack of feedback, the quadrotor drifts away from its trajectory and collides with multiple obstacles.

In the simulations shown in Fig. 7(b-e) the UAV is assumed to navigate in the environment for the first time without previous knowledge of the obstacles locations. In Fig. 7(b), the UAV monitors both its localization and its range sensors only at the times scheduled by the proposed reachability-based self-triggered approach, however it moves using an open-loop controller with maximum speed $v = 0.5$m/s between replanning times. In this case, replanning occurred 121 times (black '$\times$' in Fig. 7(b)). In Fig. 7(c), the UAV monitors its localization sensor constantly (usually not computationally demanding) and schedules the next range sensor checking and replanning operations aperiodically using our proposed self-triggered approach described in Section V-B. Speed is also adapted between $v_{min} = 0.25$m/s and $v_{max} = 3$m/s, according to the curvature of the trajectory as described in Section V-D. The UAV was able to complete the waypoint navigation safely, with 58 replanning operations following the trajectory very closely even in the presence of external disturbance. The reachable tubes generated during the first edge of the trajectory are shown in Fig. 7(f). In the case presented in Fig. 7(d), we also relax replanning operations based on the risk-based approach introduced in Section V-C, to avoid unnecessary sensor checking and replanning operations. At the points shown by magenta '$\diamond$' symbols in Fig. 7(d), the UAV calculates the risk, and decides the next replanning time. Minimum and maximum risk thresholds were set to $\rho^- = 0.1$ and $\rho^+ = 0.4$ respectively. Using this approach, the UAV performed only 21 replanning operations (black '$\times$') following its desired trajectory closely with an average deviation of 6.74cm. We have also compared our approach with traditional motion planning, Fig. 7(e), in which the range sensor is periodically checked at 40Hz rate– typical of modern lidars– and replanning occurs whenever the UAV detects an obstacle along its path.

Table I summarizes the number of replanning events, range sensor checking, and maximum and average deviations for the different cases simulated in Fig. 7. As expected, using periodic checking and replanning, Fig. 7(e), the quadrotor is able to track better a trajectory, however the vehicle ends up checking its range sensor unnecessarily 10233 times, thus using more computation and consuming more energy than in the other cases. The self-triggered risk based replanning approach (Fig. 7(d)) behaves overall better than the others considering the good tracking performance and minimum number of sensor monitoring and replanning operations.

TABLE I
Comparison between the different cases simulated in Fig.7

	Number of replanning/ range sensor checking	Max. deviation	Avg. deviation
Fig. 7(a)	1/1	247.2cm	123.1cm
Fig. 7(b)	121/121	59.27cm	14.54cm
Fig. 7(c)	58/58	25.43cm	5.54cm
Fig. 7(d)	21/21	20.24cm	6.74cm
Fig.7(e)	54/10233	9.23cm	3.51cm

VII. Experiments

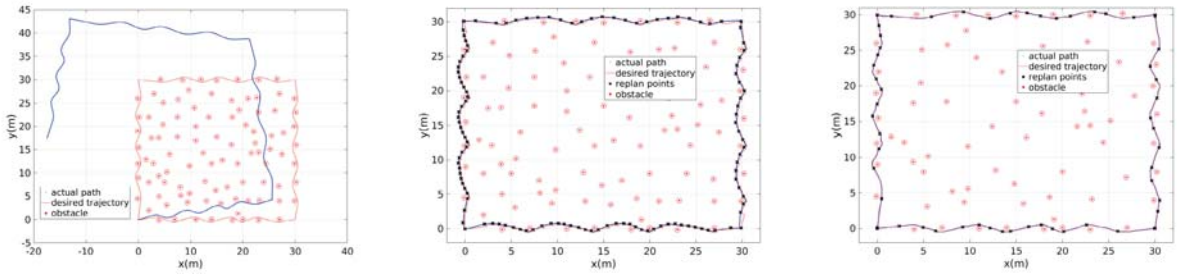
The proposed approach was validated experimentally using an AscTec Pelican quadrotor UAV equipped with i7 CPU and an Hokuyo Lidar range sensor. We followed the architecture depicted in Fig. 8. Since experiments were conducted indoors, we used our Vicon motion capture system to monitor the state $\tilde{\boldsymbol{x}}$ of the quadrotor, while obstacle positions $\tilde{\boldsymbol{o}}$ were estimated using the on-board lidar. Our self-triggered approach was implemented in Matlab. The ellipsoid toolbox [16] was used to calculate reachable tubes while the quadrotor was controlled using the Robotics System Toolbox to interface Matlab with ROS. Fig. 9(a) shows an overlapped sequence of snapshots for a waypoint navigation experiment in which the UAV is tasked to visit the four corners of a rectangular trajectory. Two obstacles (inflated poles) are present along the path. At run time the quadrotor builds a trajectory to avoid the obstacles, adapts its speed, and schedules next lidar sensor checking and replanning operations using the proposed reachability-based self-triggered approach. Replanning was performed online taking 0.02s over a prediction time horizon T = 8s.

In Fig. 9(b), the actual path of the quadrotor is shown by a blue solid curve while its desired trajectory is shown by a red curve. Sensor checking and replanning occured only 11 times while avoiding both obstacles. By using our self-triggered scheduling approach, CPU utilization was decreased to 2.7% against the 9% recorded when running a periodic 40Hz lidar checking and motion planning controller.

VIII. Conclusions and Future Work

In this work, we have presented a self-triggered policy for UAV motion planning in cluttered environments under the effect of uncertainties and disturbances. Our approach leverages reachability analysis to schedule the next replanning operations, risk analysis to relax unnecessary sensor checking and replanning operations, and curvature analysis to adapt the speed of the vehicle. Using the proposed techniques, we demonstrated that it is possible to minimize computation and thus energy consumption and guarantee safety and liveness conditions without constantly monitoring sensors and replan the vehicle's operations.

In our current work, we are extending this approach to more complicated scenarios with dynamic obstacles. In the future we also plan to use a similar framework to optimize



(a) Open-loop (b) Self-triggered with open-loop (c) Self-triggered with closed-loop

(d) Risk-based replanning (e) Periodic obstacle detection (f) Reachable tubes generated along the first edge of the path shown in Fig. 7(c)

Fig. 7. Simulation Results

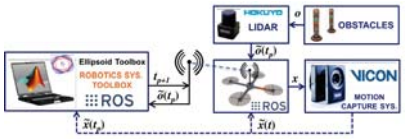
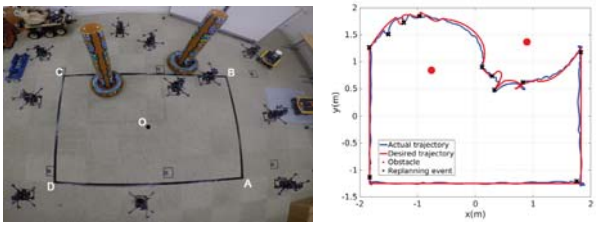


Fig. 8. Architecture of the experiment setup.



(a) Overlapped snapshots sequence. (b) Desired vs. actual trajectory.

Fig. 9. Waypoint navigation experimental results.

energy consumption leaving sensors in idle mode when not in use. Further, we plan to incorporate machine learning techniques to consider cases in which the model of the vehicle is unknown or changes due to failures.

ACKNOWLEDGMENTS

This work is based on research sponsored by ONR under agreement number N000141712012.

REFERENCES

[1] M. C. P. Santos, L. V. Santana, A. S. Brando, and M. Sarcinelli-Filho, "Uav obstacle avoidance using rgb-d system," in *International Conference on Unmanned Aircraft Systems*, June 2015, pp. 312–319.

[2] S. Roelofsen, D. Gillet, and A. Martinoli, "Reciprocal collision avoidance for quadrotors using on-board visual detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 4810–4817.

[3] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, July 2017.

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

[5] A. P. Vinod, B. HomChaudhuri, and M. M. Oishi, "Forward stochastic reachability analysis for uncontrolled linear systems using fourier transforms," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control (HSSC)*, 2017, pp. 35–44.

[6] Y. Lin and S. Saripalli, "Sampling-based path planning for uav collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–14, 2017.

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

[8] S. Singh, A. Majumdar, J. J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5883–5890.

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

[10] E. Yel, T. X. Lin, and N. Bezzo, "Reachability-based self-triggered scheduling and replanning of uav operations," in *NASA/ESA Conference on Adaptive Hardware and Systems*, July 2017, pp. 221–228.

[11] A. Ahmadzadeh, A. Jadbabaie, V. Kumar, and G. J. Pappas, "Stable multi-particle systems and application in multi-vehicle path planning and coverage," in *46th IEEE Conference on Decision and Control*, Dec 2007, pp. 1467–1472.

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

[13] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2030–2042, 2010.

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

[15] M. Egerstedt, K. H. Johansson, S. Sastry, and J. Lygeros, "On the regularization of zeno hybrid automata," *Systems and Control Letters*, vol. 38, pp. 141–150, 1999.

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