Multi-level Adaptation for Automatic Landing with Engine Failure under Turbulent Weather

Haotian Gu* and Hamidreza Jafarnejadsani†
Stevens Institute of Technology, Hoboken, New Jersey, 07030

This paper addresses efficient feasibility evaluation of possible emergency landing sites, online navigation, and path following for automatic landing under engine-out failure subject to turbulent weather. The proposed Multi-level Adaptive Safety Control framework enables unmanned aerial vehicles (UAVs) under large uncertainties to perform safety maneuvers traditionally reserved for human pilots with sufficient experience. In this framework, a simplified flight model is first used for time-efficient feasibility evaluation of a set of landing sites and trajectory generation. Then, an online path following controller is employed to track the selected landing trajectory. We used a high-fidelity simulation environment for a fixed-wing aircraft to test and validate the proposed approach under various weather uncertainties. For the case of emergency landing due to engine failure under severe weather conditions, the simulation results show that the proposed automatic landing framework is robust to uncertainties and adaptable at different landing stages while being computationally inexpensive for planning and tracking tasks.

I. Introduction

The unmanned aerial vehicles (UAVs) technology, which is moving towards full autonomous flight, requires operation under uncertainties due to dynamic environments, interaction with humans, system faults, and even malicious cyber attacks. Ensuring security and safety is the first step to making the solutions using such systems certifiable and scalable. In this paper, we introduce an autopilot framework called “Multi-level Adaptive Safety Control" (MASC) for the resilient control of autonomous UAVs under large uncertainties and employ it for engine-out automatic landing under severe weather conditions.

A. MASC Architecture

In 2009, an Airbus A320 passenger plane (US Airways flight 1549) lost both engines minutes after take-off from LaGuardia airport in New York City due to severe bird strikes [1]. Captain Sullenberger safely landed the plane in the nearby Hudson River. Inspired by this story, we aim to equip UAVs with the capability of human pilots to determine if the current mission is still possible after a severe system failure. If not, the mission is re-planned so that it can be accomplished using the remaining capabilities. This is achieved by the proposed autopilot framework, MASC, which is capable of performing safe maneuvers that are traditionally reserved for human pilots.

Fig. 1 (left) Multi-level Adaptive Safety Control (MASC) framework, (right) Simulation studies using X-Plane® program

*Ph.D. Student, Department of Mechanical Engineering, Hoboken, New Jersey, USA, AIAA Student Member.
†Assistant Professor, Department of Mechanical Engineering, Hoboken, New Jersey, USA, AIAA Member.

From a mission control architecture perspective, we aim to replace the traditional top-down, one-way adaptation, that starts with mission planning and cascades down to trajectory generation, tracking, and finally stabilizing controller, with an integrated top-down and bottom-up architecture that allows for two-way adaptation between planning and control to improve the stability and robustness of the system. To this end, we build the MASC framework upon the Simplex fault-tolerant architecture [2–6], which is recognized as a useful approach for the protection of cyber-physical systems against various software failures. By integrating the MASC framework with the Simplex architecture, we aim to enable cyber-physical systems to handle large uncertainties originating from the physical world. The MASC framework, shown in Figure 1, consists of the following components:

- Normal Mode Controller: equipped with complex functionalities to operate the system under normal conditions.
- Safe Mode Controller with Multi-level Adaptation: a simple and verified controller that ensures safe and stable operations of the system with limited levels of performance and reduced functionalities. The control architecture consists of three levels: i) offline landing trajectory prediction; ii) mission feasibility evaluation and re-planning; and iii) online trajectory generation and path following control.
- Monitoring and Capability Auditing: uses a model considering the cyber-physical nature of autonomous systems for estimation and fault detection. The model identifies the remaining capabilities of the system and its decision logic triggers a switch from Normal Mode to Safe Mode.

Under the engine-out flight scenario with turbulent weather, the proposed architecture adapts the mission to the new constraints by i) auditing the remaining capability of the crippled aircraft and providing feedback to the other layers, ii) updating the flight envelope, iii) evaluating the feasibility of potential reachable destinations and selecting the low risk one; and iv) planning the flight path online and then employing a robust autopilot controller to track the path, meantime making sure to stay within the flight envelope of the crippled UAV. Feedback provided from the lower layers to the higher layers such as the mission planner allows for the interaction of the MASC modules and a two-way adaptation between the planning and control.

Emergency landing due to an engine failure under severe weather conditions is challenging even for an experienced pilot. Our proposed approach, referred to as the MASC framework, provides the autopilot with the agility required to compensate for uncertainties by adaptations in planning and control. The framework can be employed on dependable computing architectures for the safety control design. We utilize a computationally-efficient trajectory generation and tracking control approach, which can be computed with low latency on a low-cost real-time embedded platform onboard most UAVs. Also, the framework allows for the evaluation of reachable areas for an emergency landing. We tested the framework in a high-fidelity flight simulation environment for validation and verification. We achieved successful landings under a wide range of initial conditions (i.e., altitude, distance, and orientation relative to the landing site), windy weather, and turbulence without re-tuning the autopilot parameters.

B. Related Work: Engine-Out Emergency Landing

UAVs play a significant role in a wide range of industries, including defense, transportation, and agriculture, to name a few [7–9]. Engine failure is one of the most hazardous situations for UAVs [10, 11]. While engine failures are not common in passenger aircraft, an engine-out accident is more probable for a low-cost commercial UAV. The safety risks are even higher if a UAV crashes over a populated area endangering people and infrastructure on the ground. Numerous approaches for planning and control are proposed in the literature to mitigate the risks due to engine failure. In [12], an adaptive flight planner (AFP) presented for landing an engine-out aircraft. The AFP approach for loss of thrust case performs the two main flight-planning tasks required to get a crippled aircraft safely on the ground: i) select a landing site and ii) construct a post-failure trajectory that can safely reach that landing site. An adaptive trajectory generation scheme with a certain presumed best glide ratio and bank angle for turns is proposed in [13]. Additionally, trajectory planning based on flight envelope and motion primitives is proposed in [14] for damaged aircraft. A reachable set for auto-landing is calculated by using optimal control theory in [15]. Most of the related studies do not address emergency landing under additional weather uncertainties, and simulation results are mainly based on simplified models for the aircraft and environment.

The rapidly-exploring random trees (RRT) method is a popular sampling-based path planning algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree. The algorithm creates a search tree containing a set of nodes and the connecting path edges set. In [16], a path planning scheme is developed based on the optimal sampling-based RRT algorithm to generate a landing trajectory in real-time and also examines its performance for simulated engine failures occurring in mountainous terrain. However, RRT-based algorithms are computationally demanding for planning large-scale smoother path [17]. The demand increases with the dimensions of

the searched state-space. Another motion planning for emergency landing is based on the Artificial Potential Field (APF) [18] method and greedy search in the space of motion primitives. In APF [19], the UAV's path is calculated based on the resultant potential fields from the initial point to the target point. However, the conventional APF [20] may encounter the trap of a local minimum when the attractive force and repulsive force reach a balance, which means that the UAV stops moving towards the target.

This paper is organized as follows. The components of the Multi-level Adaptive Safety Control (MASC) framework are presented in Section II. Particularly, monitoring and capability auditing is discussed in Section II.A, the low-level safety controller is presented in Section II.B, and the mission adaptation is described in Section II.C. Section III.D describes the high-fidelity software-in-the-loop (SITL) simulation environment for a fixed-wing aircraft and presents the simulation results. Finally, Section IV concludes the paper.

II. Multi-level Adaptive Safety Control (MASC)

This section presents the components of the Multi-Level Adaptive Safety Control (MASC)* framework.

A. Monitoring and Capability Auditing

The monitoring and capability auditing module has a set of stored expected models $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$ where the triple

$$\mathcal{D}_j = \{A_j, B_j, \Theta_j\} \tag{1}$$

represents the plant matrices $(A_j,\ B_j)$, and the uncertainty set Θ_j. In particular, each model is represented as

$$\mathcal{D}_j : \begin{cases} \dot{x}(t) = A_j x(t) + B_j(u(t) + f_j(x(t), t)), \\ y(t) = Cx(t), \quad x(t_0) = x_0, \end{cases} \tag{2}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, and $y(t) \in \mathbb{R}^q$ is the available output measurement. The term $f_j \in \Theta_j,\ \forall (x, t) \in \mathbb{R}^n \times [0, \infty)$, represents unknown system uncertainties and disturbances subject to local Lipschitz continuity assumption. Control input $u(t)$ is the robust low-level controller that stabilizes the model $\mathcal{D}_j$ with guaranteed robustness margins for *apriori* given bounds on the uncertainties.

Remark 1 *It is worth mentioning that $\mathcal{D}$ is a set of nominal/representative fault models. The models do not need to be perfectly accurate, and any modeling error is expressed as $f_j \in \Theta_j$. Given a nominal model, the safe mode controller will deal with any model mismatch or external disturbance.*

Monitoring and capability auditing is an integral part of the MASC framework (shown in Figure 1), which performs the task of fault detection and isolation (FDI), i.e., to notice the existence of a fault, and to further identify the fault model. Since the autopilot is characterized as a cyber-physical system, regardless of the location of the faulty elements, the effect of the fault is always reflected in the physical world. Leveraging the measurement of physical state, we employ a model-based FDI approach used in control literature [21][22]. Identifying the new model is critical for stabilizing the UAV, and it should be prioritized computationally, while the mission re-planning algorithm can take longer to converge to a feasible trajectory.

In the particular case of engine malfunction, monitoring the sensors such as the engine's RPM indicator and onboard accelerometer provides sufficient information for the monitoring and capability auditing module to detect the engine failure. Then, the module activates the planning and control task specifically designed for emergency landing within the Safe Control Mode module. Having identified the faults and updated the system model $\mathcal{D}_j$, another critical task of the monitoring and capability auditing module is to determine the safe flight envelope for the mission re-planing. Specifically, for protecting the flight envelope during the emergency landing, it is very crucial to maintain the forward airspeed around the optimal gliding speed V_{opt} and the corresponding best slope γ_{opt} that the aircraft manufacturer recommends. The optimal speed ensures maximum gliding distance without stalling the aircraft. In addition, the following constraints are considered for motion planning:

$$\begin{aligned} V_{\min} < V < V_{\max}, \quad & p_{\min} < p < p_{\max}, \\ \theta_{\min} < \theta < \theta_{\max}, \quad & q_{\min} < q < q_{\max}, \\ \phi_{\min} < \phi < \phi_{\max}, \quad & r_{\min} < r < r_{\max}, \end{aligned} \tag{3}$$

*Open source code for the MASC framework on Github: (`https://github.com/SASLabStevens/MASC-Architecture.git`)

where V, θ, and ϕ are the forward airspeed, pitch angle, and roll angle, respectively. Also, p, q, and r are roll, pitch, yaw rates, respectively.

B. Safe Mode Path-Following Controller

Path following controller modifies the control commands to the low-level longitudinal and lateral controllers to follow the reference path. Monitoring and Capability Auditing will compute the mission feasibility, given the states and the model of a damaged UAV. Large uncertainty mitigation requires mission adaptation and selection of a new trajectory that is still feasible, given the remaining capabilities. For large uncertainties outside the design bounds, i.e., $f_j \notin \Theta_j$ in (2), the control inputs can saturate and drive the system to unsafe states. Modification of the reference command $r_{\mathrm{d}}[i]$ based on the updated objectives is another layer of defense for maintaining safety by *satisfying flight envelope constraints*. Therefore, we consider a control structure that consists of a path following controller, where the generated reference commands to the low-level controller are limited by saturation bounds to maintain the closed-loop system within operational safety envelope.

Let the reference command be constrained to a convex polytope as a safe operational region, defined by the set

$$\mathcal{R} = \{r_{\mathrm{d}} \in \mathbb{R}^q \,|\, \|W r_{\mathrm{d}}\|_\infty \leq 1\}\,, \tag{4}$$

where $W = \mathrm{diag}\{\mathrm{r}_{\mathrm{max}_1}^{-1}, \ldots, \mathrm{r}_{\mathrm{max}_\mathrm{q}}^{-1}\}$, and the positive constants $\mathrm{r}_{\mathrm{max}_\mathrm{i}}$'s are the saturation bounds on the reference commands. Then, the weighted reference command is bounded by

$$\|W r_{\mathrm{d}}[i]\|_\infty \leq 1, \quad i \in \mathbb{Z}_{\geq 0}.$$

In this paper, the reference command $r_{\mathrm{d}}[i],\ i \in \mathbb{Z}_{\geq 0}$, which is generated by the path following control law, is given by

$$r_{\mathrm{d}}[i] = (1-\alpha) W^{-1} \mathrm{sat}\left\{\frac{1}{1-\alpha} W F_{\mathrm{z}} \left(z_{\mathrm{m_d}}[i] - z_{\mathrm{d}}[i]\right)\right\}, \tag{5}$$

where $\mathrm{sat}\{\cdot\}$ denotes the saturation function, $F_{\mathrm{z}} \in \mathbb{R}^{q\times p}$ is the state-feedback gain, and $\alpha \in (0,\ 1)$ is a constant. Also, $z_{\mathrm{m_d}}[i]$ is the desired trajectory variable generated by the mission planner and $z_{\mathrm{d}}[i]$ is the actual state of the aircraft. In the case of an emergency glide landing, the desired heading angle ψ and flight path angle γ are the variables that are generated by the mission planner. The path following controller in (5) is equivalent to a PI controller subject to a saturation function. This control law ensures that the roll and pitch commands stay within the safe flight envelope during the glide landing, and hence the possibility of a stall decreases.

C. Mission Adaptation

In the MASC framework, we present a mission planner in the Safe Control Mode that generates the landing trajectory to a safe landing site. The planner also evaluates mission feasibility considering the damage of the aircraft and environmental constraints. In the case of a fault/failure detection, once the capability auditing provides the new/altered model $\mathcal{D}_j$, two concurrent steps will be taken: *i)* MASC initiates the feasibility evaluation and landing trajectory estimation, and *ii)* autopilot controller is activated based on its ability to stabilize the system around a pre-calculated $r_{\mathrm{d}}^{th}[\cdot]$ reference command. The first step is taken to ensure that the system does not violate its stability/safety bounds while the mission is being re-planned. Once the mission is re-planned, it is fed into the path following controller, which then accordingly alters the $r_{\mathrm{d}}^{th}[\cdot]$ reference command provided to the j^{th} low-level controller to execute the new mission. In the second step, re-planning of the mission, it is crucial to compute the *reachable area*, which is defined as all the spatial points the aircraft is capable of reaching given the dynamic constraints, potential and kinetic energy, and available fuel if the engine is still partially working. To this end, provided the information and new constraints by the monitoring and capability auditing module, a set of candidate locations are initially considered. This initial set may include nearby airports and empty lands. Leveraging the updated model of the UAV, MASC evaluates the feasibility of safe landing in the candidate areas and identifies the most likely safe location for an emergency landing. Any violation of the safety constraints (for instance (3) and (4)) during the evaluation process rules out a candidate area as a safe reachable area.

The feasibility evaluation for landing site selection is summarized in Algorithm 1, and the landing trajectory planning is summarized in Algorithm 2. The feasibility evaluation can be viewed as an offline trajectory planning from the initial state of the engine-out aircraft to a few possible landing coordinates. For the offline trajectory generation, a simplified and reduced model of the UAV dynamics and autopilot is used to estimate the trajectory and travel time to each landing site. Consequently, the more desirable landing coordinate is selected, and it is passed to the online trajectory planning

for the online execution. The block diagram for feasibility evaluation of the landing trajectories using a simplified UAV model is shown in Figure 2. Both online and offline planning use the carrot chasing based guidance logic for trajectory following [23]. The carrot chasing method uses a pseudo target moving along the desired flight path while generating the desired heading angle using the reference point [24], and it is robust to disturbances [25].

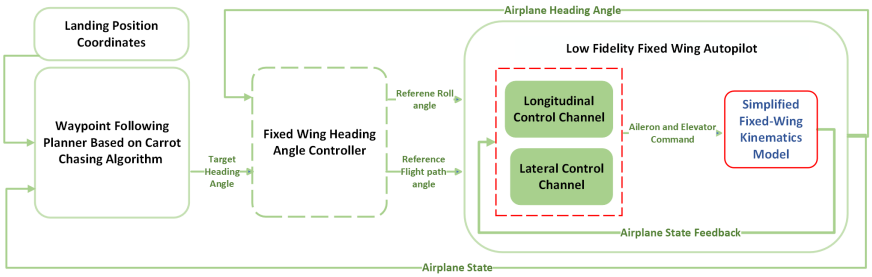

Fig. 2 The block diagram for evaluating feasibility of the landing trajectories using a simplified UAV model in Matlab

The landing mission is divided into three phases: *Phase I: Cruising, Phase II: Loitering, and Phase III: Approach* as illustrated in Figure 3. In Algorithm 2, the emergency glide landing procedure starts by cruising towards the loitering center with coordinates $(x_\mathrm{l}, y_\mathrm{l})$ near the landing site. After the aircraft is close enough to the loitering center, i.e., $\sqrt{(x_\mathrm{l}-x)^2+(y_\mathrm{l}-y)^2} < R_\mathrm{l}$, the mission enters the loitering phase. While loitering, the aircraft losses altitude in a spiral trajectory. When the cut-off altitude, z_a, is reached, i.e., $z < z_a$, the mission enters the approach phase. Notice that the mission is allowed to progress in one direction, similar to a directed acyclic graph (DAG), and it is possible that the first and second phases are skipped altogether depending on the initial states of the aircraft, as illustrated in Figure 3. The variables defined for the three flight phases in Algorithm2 are described in the following.

Phase I: During the cruising phase, the aircraft cruises towards the loitering circle. The desired heading angle is calculated as given in Algorithm 2. The line-of-sight (LOS) θ denotes the angle in the Euclidean plane, given in radians, between the positive x-axis and the ray to the point $(W_\mathrm{ipl}, W_\mathrm{i})$ which connects the initial engine malfunction position to the loiter center. $path_\mathrm{g}$ is the Euclidean distance of the current coordinates and the start of the reference path. R_u is the distance between the start of the reference path and the current particle projected on the straight reference path. The airspeed should remain close to the best gliding speed V_opt while the UAV cruises to the landing site.

Phase II: In the loitering phase, the aircraft loiters near the landing site in a spiral trajectory to lose any excessive altitude for the final approach. In Algorithm 2, O denotes the global coordinates of the loiter center. Also, θ will gradually approach the chord tangent angle of loiter as the moving trajectory converges to the reference circle. In addition, δ is the look-ahead distance. With an increase in δ, the flight path of the UAV can converge to the reference trajectory quickly, reducing the cross-track error [26].

Phase III: In the approach phase, the aircraft aligns itself with the runway and tracks the desired flight path angle for the final approach. The trajectory point renew scheme is the same as the Phase I. The difference is θ denotes the angle in the Euclidean plane, given in radians, between the positive x-axis and the ray to the points $(W_\mathrm{ipl}, W_\mathrm{i})$ which connects optimal landing area coordinates to $(x_\mathrm{u}, y_\mathrm{u})$. If the ground distance of the aircraft to the landing site is larger than a constant R, i.e., $\sqrt{(x-x_\mathrm{f})^2+(y-y_\mathrm{f})^2} > R$, we have

$$\begin{aligned} x_\mathrm{u} &= x_\mathrm{l} + R_\mathrm{l} \times cos(\psi_\mathrm{f} - \pi), \\ y_\mathrm{u} &= y_\mathrm{l} + R_\mathrm{l} \times sin(\psi_\mathrm{f} - \pi), \end{aligned} \tag{6}$$

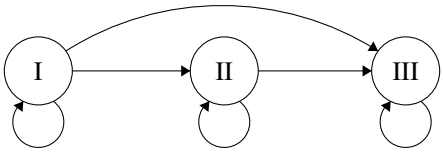

Fig. 3 Landing mission phase transition diagram. Phase I: Cruising, Phase II: Loitering, and Phase III: Approach.

where R_l is loiter diameter, and ψ_f is the heading angle of the runway. Also, $(W_{ipl},\ W_i)$ connects the $(x_u,\ y_u)$ start of the reference line and landing position coordinates $(x_f,\ y_f)$ in Euclidean plane.

Algorithm 1: algorithm for feasibility evaluation for candidate landing areas.

Input: Initial engine out coordinates, coordinates of the back up landing areas
Output: Predicted landing trajectory and estimated landing time
Procedure For Feasibility Evaluation of Landing Areas:
If(Engine malfunction == true)
    MASC Framework initiates the Feasibility Evaluation process.
    GO TO Landing area feasibility evaluation and landing trajectory and time prediction
    1: Feed the global coordinates where engine is out and of landing areas;
    2: Implement offline path planning in acceleration mode;
    3: Get the predicted flight trajectories and estimated landing time;
    4: Determine the most suitable landing site;
  do
    MASC(Online Navigation) Initiate
  while(The determined optimal landing coordinates)
else
    Conducting the Normal Flight mode;

III. Software-in-the-loop Simulation Study

This section presents a software-in-the-loop (SITL) simulation scheme to evaluate and validate the proposed MASC[†] framework for online path planning and navigation under the emergency landing case. The SITL architecture for the MASC autopilot, shown in Figure 4, has three primary components: i) high-fidelity physical simulation environment (in X-Plane), and ii) MASC autopilot (in MATLAB/Simulink) consisting of a nonlinear logic based mission planner and proportional heading angle regulation scheme, and iii) a user datagram protocol (UDP). The X-Plane program has been certified by the Federal Aviation Administration (FAA) as a simulation software to train pilots. The aircraft model used in this simulation is Cessna 172SP as shown in Figure 1.

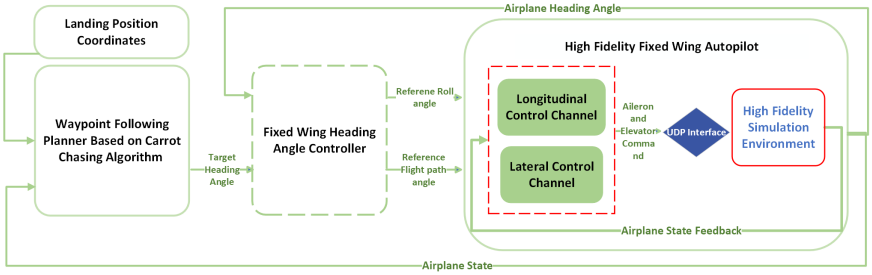

Fig. 4 The block diagram for implementing MASC framework in the X-Plane high-fidelity flight simulation environment

A. UDP Receiver and Sender Interface

X-Plane adopts the User Datagram Protocol (UDP) to communicate with the third-party software and external processes. Unlike the Transmission Control Protocol (TCP), UDP assures that data packages will arrive completely and orderly. Also, the communication via UDP can achieve high-speed data traffic compared to other protocols by efficient use of bandwidth, which is an advantageous characteristic for the simulation under consideration. Correspondingly, DSP toolbox in MATLAB/Simulink supports the UDP communication through DSP System Toolbox. This toolbox can query an application using UDP to send real-time data from the Simulink model to the corresponding channel in

[†]Open source code for the MASC framework on Github: (`https://github.com/SASLabStevens/MASC-Architecture.git`)

Algorithm 2: Algorithm for emergency landing using the nonlinear guidance logic [23].

Input: Initial Engine out coordinates, reachable landing coordinates and runway direction.
Output: Desired Heading Angle
SIL simulation Procedure:
if (Engine malfunction == true)
if ($Distance > R_l$ and $z > z_a$)
begin if
1: Initialize: $W_\mathrm{i} = (x_\mathrm{i},\ y_\mathrm{i})$, $W_\mathrm{ipl} = (x_\mathrm{ipl},\ y_\mathrm{ipl})$, $P_\mathrm{new} = (x_\mathrm{new},\ y_\mathrm{new})$, ψ_des, δ, θ ;
2: $path_\mathrm{g} = \|W_\mathrm{i} - P_\mathrm{new}\|$;
3: $\theta = \left\|W_\mathrm{ipl} - W_\mathrm{i}\right\|$;
4: $\theta_\mathrm{u} = \|P_\mathrm{new} - W_\mathrm{i}\|$;
5: $l_\mathrm{d} = \theta - \theta_\mathrm{u}$;
6: $R_\mathrm{u} = \sqrt{(path_\mathrm{g})^2 + (path_\mathrm{g} \times \sin(l_\mathrm{d})))^2}$;
7: $(x_\mathrm{t},\ y_\mathrm{t}) = ((R_\mathrm{u} + \delta) \times \cos(\theta),\ (R_\mathrm{u} + \delta) \times \sin(\theta))$;
8: $\psi_\mathrm{des} = \mathrm{atan2}(y_\mathrm{t} - y_\mathrm{new},\ x_\mathrm{t} - x_\mathrm{new})$;
end if
else if ($Distance < R_l$ and $z > z_a$)
begin if
1: Initialize: $O = (x_\mathrm{l},\ y_\mathrm{l})$, $P = (x,\ y)$, ψ_des, λ, κ;
2: $d = \|O - P\| - R_\mathrm{c}$;
3: $\theta_\mathrm{l} = \mathrm{atan2}\ (y - y_\mathrm{l},\ x - x_\mathrm{l})$;
4: $(x_\mathrm{t},\ y_\mathrm{t}) = (x_\mathrm{l} + R_\mathrm{c} \times \cos(\lambda + \theta_\mathrm{l}),\ y_\mathrm{l} + R_\mathrm{c} \times \sin(\lambda + \theta_\mathrm{l}))$;
5: $\psi_\mathrm{des} = \mathrm{atan2}(y_\mathrm{t} - y,\ x_\mathrm{t} - x)$;
end if
else
begin if
1: Initialize: $W_\mathrm{i} = (x_\mathrm{i},\ y_\mathrm{i})$, $W_\mathrm{ipl} = (x_\mathrm{ipl},\ y_\mathrm{ipl})$, $P_\mathrm{new} = (x_\mathrm{new},\ y_\mathrm{new})$, ψ_des, δ, θ;
2: $path_\mathrm{g} = \|W_\mathrm{i} - P_\mathrm{new}\|$;
3: $\theta = \left\|W_\mathrm{ipl} - W_\mathrm{i}\right\|$;
4: $\theta_\mathrm{u} = \|P_\mathrm{new} - W_\mathrm{i}\|$;
5: $l_\mathrm{d} = \theta - \theta_\mathrm{u}$;
6: $R_\mathrm{u} = \sqrt{(path_\mathrm{g})^2 + (path_\mathrm{g} \times \sin(l_\mathrm{d})))^2}$;
7: $(x_\mathrm{t},\ y_\mathrm{t}) = (R_\mathrm{u} + \delta) \times \cos(\theta),\ (R_\mathrm{u} + \delta) \times \sin(\theta))$;
8: $\psi_\mathrm{des} = \mathrm{atan2}(y_\mathrm{t} - y_\mathrm{new},\ x_\mathrm{t} - x_\mathrm{new})$;
end if
else
Fly in Normal Mode;
end;

X-Plane. Also, the UDP object allows performing byte-type and datagram-type communication using a UDP socket in the local host. For the implementation, we consider designing subscriber and publisher to guarantee data transfer between the X-Plane and MATLAB/Simulink in real time. For subscriber, we used two Simulink blocks: an embedded MATLAB function and a byte unpack. For publisher, we use a byte pack and encoder, and both are linked via a bus module in Simulink.

B. Engine-Out Landing under Clear Weather

The simulation results for the automatic landing process under clear weather are shown in Figure 5, illustrating three different viewpoints of the real-time landing trajectory. In this emergency landing simulation, we randomly initiate the process from five different positions given in Table 1. The final landing coordinates are set to North $21822m$, East $-9751.8m$, Height $140m$, and Heading direction of the runway $24.18deg$. As we can see from the results, the MASC for emergency landing during engine failure can plan a path online to safely navigate the aircraft to the configured landing site from various initial conditions. Also, the aircraft's airspeed remains well above the stall speed, which is around 30 m/s. Throughout the simulations, the weather condition is set to Clear, with no wind, i.e., the best weather condition in X-Plane®.

Trial	North (m)	East (m)	Height (m)	Heading (Deg)
1	13163	-7164.9	3000	78.5
2	13353	-14380	4000	110.3
3	23429	-6675.6	5000	85.6
4	20719	-11652	3000	256.74
5	21323	-11021	2000	69.594

Table 1 The initial position coordinates of the engine-out aircraft in the simulation trials

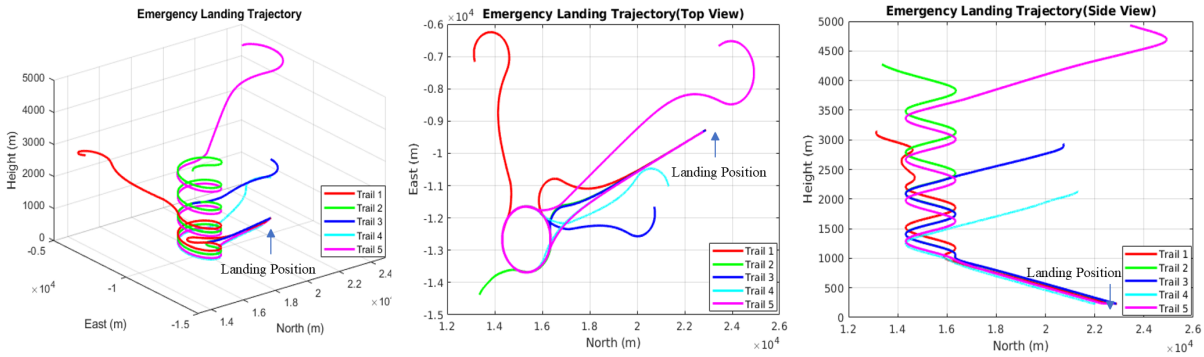

Fig. 5 Automatic engine-out landing under clear weather from different initial positions (simulation video link)

C. Engine-Out Landing under Windy and Turbulent Weather

To further demonstrate the robustness of the MASC framework under large wind and turbulence uncertainties, we simulated the emergency landing task under different severe weather settings in the X-Plane program with parameters listed in Table 2. We varied the parameters Wind Direction, Wind Speed, Turbulence, Gust Speed Increase, and Total Wind Shear to different levels in the high fidelity simulation environment. Figure 6 shows the real-time trajectories for landing in different weather conditions. The initial and final coordinates of the aircraft set to that of the first trial in Table 1. We note that the initial coordinates of the engine-out aircraft are slightly different but relatively close to each other in these simulations because of how we initialize these test runs. As our results suggest, MASC can navigate the aircraft to the configured landing site in each test run under severe weather conditions. Therefore, our approach can robustly plan a landing trajectory and safely navigate the aircraft to a landing site under large wind and turbulence uncertainties.

Trial	Wind Direction (deg)	Wind Speed (kts)	Turbulence (%)	Gust Speed Increase (kts)	Total Wind Shear
1	20	14	10	10	10
2	0	3	8	14	8
3	4	5	10	8	14
4	14	7	12	22	8
5	27	12	4	9	4

Table 2 The weather settings in the X-Plane program

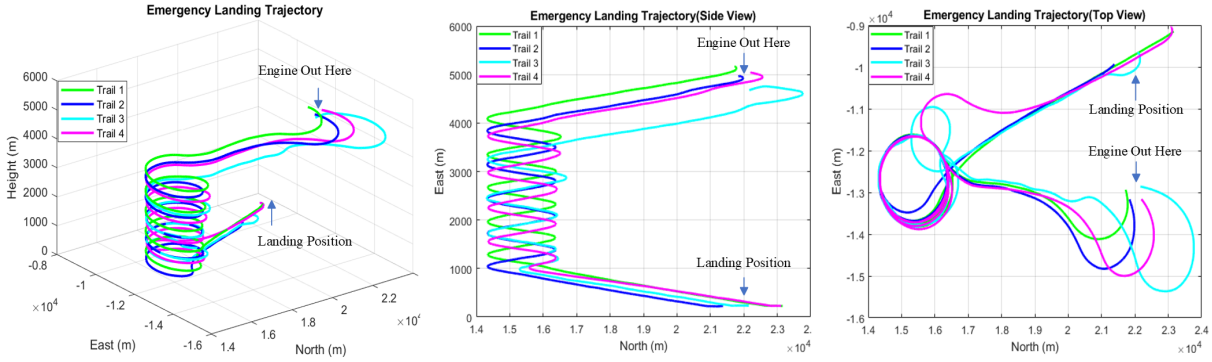

Fig. 6 Automatic engine-out landing under different windy and turbulent weather settings in the X-plane program (simulation video link)

D. Feasibility Evaluation for Landing Site Selection

In the MASC framework, the mission adaptation starts with offline trajectory planning for reachability and feasibility evaluation of candidate landing sites. The offline planning architecture, presented in Figure 2, computes feasible emergency landing trajectories for each nearby landing site and selects the best landing coordinates. In our implementation, the control laws and simplified UAV model are implemented in MATLAB/Simulink in acceleration mode. The feasible landing site selection consists of three tasks: offline path planning, landing time estimation, and optimal landing site selection. We build a mathematical model to obtain the discretized reference trajectory points utilized to calculate the landing path and expected landing time for each landing site. To avoid unnecessary calculations, we opt-in for a low density of reference trajectory points (coarser trajectories). The simulations show that the approach is sufficiently fast and computationally inexpensive, and the computation time for each trajectory is 3.083 s on average (corresponding to a path that takes about 10 min to complete). However, we note that the computation time depends on the computing hardware used to run these simulations.

Landing Coordinates	North (m)	East (m)	Height (m)	Runway Direction (Deg)	Landing Time (s)
1	21822	-9751.8	235	24.17	650
2	11822	-6751.8	235	130	700
3	46000	-39751.8	235	40	N/A
4	36000	-19751.8	235	40	800

Table 3 Candidate landing coordinates

In this simulation, our goal is to select the landing site with the shortest landing time among four different potential sites listed in Table 3. Figure 7 shows the trajectories leading to each of the landing sites labeled by 1, 2, 3, and 4, and the corresponding landing times are summarized in Table 3. We note that landing sites 1, 2, and 4 are reachable, but landing site 3 is infeasible, i.e., the engine-out aircraft crashes before reaching the landing site 3. It takes 650 s to land at

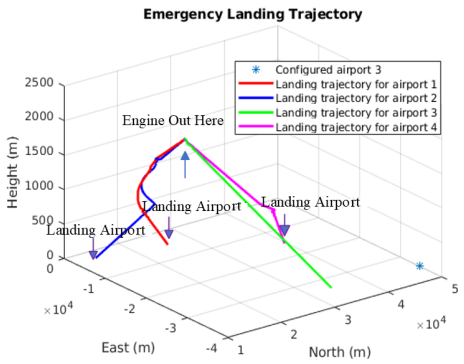

Fig. 7 Trajectories generated for four different landing sites

landing site 1, which is the shortest time. The emergency landing also takes about 10.08 *min* in the SITL simulation, which means the offline planning method is accurate enough to predict the actual time needed for the emergency landing process.

E. Comparison between the Offline and Online Trajectories

We use a simplified UAV model for offline trajectory generation, and therefore, we expect some discrepancies between the predicted trajectory and the actual flight path of the UAV in the high-fidelity simulation. In the final part of our simulation study, we compare the online and offline planned paths to observe possible discrepancies. To do so, we chose the landing site 1 in SITL simulation and test set for offline path planning. The trajectory planned offline coincides with the online one, as shown in Figure 8. Due to fewer turns needed for the aircraft in the SITL simulation to match the heading direction of the reference straight line compared to the offline path planning, the diameter of the path planned offline at the start of Phase I does not entirely match the SITL trajectory. The landing time in the SITL simulation is 6.8 min, while the estimated landing time is 6 min for offline path planning in Matlab/Simulink. However, these differences do not considerably impact the main outcomes, which are predicting the feasibility of the selected path and estimation of landing time.

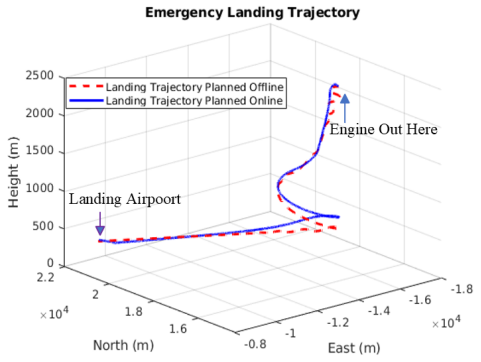

Fig. 8 Comparison of online and offline trajectories

IV. Conclusion

This paper proposes a robust and lightweight navigation and control architecture that enables UAVs to perform an automatic emergency landing under severe weather conditions. A Multi-level Adaptation approach in mission planning, path tracking, and stabilizing control was presented within this framework. The proposed framework can also evaluate the feasibility of landing at the nearby landing sites and select the optimal one. Using a high-fidelity

simulation environment, the effectiveness of the approach is verified by successfully landing a fixed-wing aircraft under engine failure for a wide range of initial aircraft states and weather uncertainties. In the future, we plan to address other challenges such as vision-based landing and obstacle avoidance when flying over a complex urban area. In this paper, rudder control has not been used. We plan to incorporate rudder control to track a desired angle of sideslip and provide the fixed-wing aircraft with more agility and maneuvering performance.

V. Acknowledgement

This research is partially supported by the National Science Foundation (award no. 2137753).

References

[1] "Sullenberger Made the Right Move, Landing on the Hudson," , [posted 05-May-2010]. URL `https://www.wired.com/2010/05/ntsb-makes-recommendations-after-miracle-on-the-hudson-investigation/`.

[2] Sha, L., "Dependable system upgrade," *Real-Time Systems Symposium, 1998. Proceedings. The 19th IEEE*, IEEE, 1998, pp. 440–448.

[3] Sha, L., "Using Simplicity to Control Complexity," *IEEE Software*, Vol. 18, No. 4, 2001, pp. 20–28. https://doi.org/10.1109/MS.2001.936213.

[4] Crenshaw, T. L., Gunter, E., Robinson, C. L., Sha, L., and Kumar, P., "The simplex reference model: Limiting fault-propagation due to unreliable components in cyber-physical system architectures," *The 28th IEEE International Real-Time Systems Symposium*, 2007, pp. 400–412.

[5] Wang, X., Hovakimyan, N., and Sha, L., "L1Simplex: fault-tolerant control of cyber-physical systems," *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, ACM, 2013, pp. 41–50.

[6] Yoon, M.-K., Liu, B., Hovakimyan, N., and Sha, L., "VirtualDrone: Virtual Sensing, Actuation, and Communication for Attack-Resilient Unmanned Aerial Systems," *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems*, 2017.

[7] D'Andrea, R., "Guest Editorial Can Drone Deliver?" *IEEE Transactions on Automation Science and Engineering*, Vol. 11, No. 3, 2014, pp. 647–648.

[8] Puri, V., Nayyar, A., and Raja, L., "Agriculture drones: A modern breakthrough in precision agriculture," *Journal of Statistics and Management Systems*, Vol. 20, No. 4, 2017, pp. 507–518.

[9] Lin, C. A., Shah, K., Mauntel, L. C. C., and Shah, S. A., "Drone Delivery of Medications: Review of the Landscape and Legal Considerations," *American Journal of Health-System Pharmacy*, Vol. 75, No. 3, 2018, pp. 153–158.

[10] Jourdan, D., Piedmonte, M., Gavrilets, V., Vos, D., and McCormick, J., "Enhancing UAV survivability Through Damage Tolerant Control," *AIAA Guidance, Navigation, and Control Conference*, 2010.

[11] Ayhan, B., Kwan, C., Budavari, B., Larkin, J., and Gribben, D., "Path planning for UAVs with engine failure in the presence of winds," *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2018, pp. 3788–3794.

[12] Atkins, E. M., Portillo, I. A., and Strube, M. J., "Emergency Flight Planning Applied to Total Loss of Thrust," *Journal of Aircraft*, Vol. 43, No. 4, 2006, pp. 1205–1216. https://doi.org/10.2514/1.18816, URL https://doi.org/10.2514/1.18816.

[13] Atkins, E., "Emergency Landing Automation Aids: An Evaluation Inspired by US Airways Flight 1549," *AIAA Infotech@ Aerospace 2010*, 2010, p. 3381.

[14] Asadi, D., Sabzehparvar, M., Atkins, E. M., and Talebi, H. A., "Damaged airplane trajectory planning based on flight envelope and motion primitives," *Journal of Aircraft*, Vol. 51, No. 6, 2014, pp. 1740–1757.

[15] Bayen, A. M., Mitchell, I. M., Osihi, M. K., and Tomlin, C. J., "Aircraft Autolander Safety Analysis Through Optimal Control-based Reach Set Computation," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 1, 2007, pp. 68–77.

[16] Choudhury, S., Scherer, S., and Singh, S., "RRT*-AR: Sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter," *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013.

[17] Sláma, J., "Emergency Landing Guidance for an Aerial Vehicle with a Motor Malfunction," 2018.

[18] Chen, Y.-B., Luo, G.-C., Mei, Y.-S., Yu, J.-Q., and Su, X.-L., "UAV path planning using artificial potential field method updated by optimal control theory," *Int. J. Syst. Sci.*, Vol. 47, No. 6, 2016, pp. 1407–1420.

[19] Dai, J., Wang, Y., Wang, C., Ying, J., and Zhai, J., "Research on hierarchical potential field method of path planning for UAVs," *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, IEEE, 2018.

[20] Budiyanto, A., Cahyadi, A., Adji, T. B., and Wahyunggoro, O., "UAV obstacle avoidance using potential field under dynamic environment," *2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, IEEE, 2015.

[21] Hwang, I., Kim, S., Kim, Y., and Seah, C. E., "A survey of fault detection, isolation, and reconfiguration methods," *IEEE transactions on control systems technology*, Vol. 18, No. 3, 2009, pp. 636–653.

[22] Gao, Z., Cecati, C., and Ding, S. X., "A survey of fault diagnosis and fault-tolerant techniques—part I: Fault diagnosis with model-based and signal-based approaches," *IEEE Trans. Ind. Electron.*, Vol. 62, No. 6, 2015, pp. 3757–3767.

[23] Sujit, P., Saripalli, S., and Sousa, J. B., "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless," *IEEE Control Systems Magazine*, Vol. 34, No. 1, 2014, pp. 42–59.

[24] Park, S., Deyst, J., and How, J., "A New Nonlinear Guidance Logic for Trajectory Tracking," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, 2004. https://doi.org/10.2514/6.2004-4900, URL https://doi.org/10.2514/6.2004-4900.

[25] Sujit, P. B., Saripalli, S., and Sousa, J. B., "An evaluation of UAV path following algorithms," *2013 European Control Conference (ECC)*, IEEE, 2013.

[26] Bemporad, A., and Morari, M., "Robust model predictive control: A survey," *Robustness in identification and control*, Springer, 1999, pp. 207–226.