AIAA SciTech Forum
11–15 & 19–21 January 2021, VIRTUAL EVENT
AIAA Scitech 2021 Forum

10.2514/6.2021-0378

Autonomous MAV Landing on a Moving Platform with Estimation of Unknown Turbulent Wind Conditions

Aleix Paris*, Andrea Tagliabue†, and Jonathan P. How‡
Massachusetts Institute of Technology, Cambridge, MA, 02139

This paper presents an autonomous landing of a micro aerial vehicle (MAV) on a moving platform immersed in turbulent wind conditions. We estimate the 3D wind vector acting on the vehicle using a model-based and a deep learning-based approach. A disturbance-aware boundary layer sliding controller then uses this estimation to generate a control input that provides trajectory tracking guarantees in the presence of unknown, but bounded disturbances. The approach presented integrates our previous works on control and estimation, and we show its performance in a challenging setting. The experiments show that our methods enable fast landing on a moving platform in turbulent, unknown wind conditions.

I. Nomenclature

$\mathbf{a}$	=	Subvector of the MAV's state $\mathbf{x}$ representing the acceleration
$a_{\max}$	=	Constant that limits the maximum acceleration
B	=	Body-fixed reference frame, placed in the center of mass (CoM) of the MAV
c	=	Drag coefficient of the MAV
d	=	Disturbance affecting the system (acceleration)
d_{land}	=	Distance from the MAV to the tag on the landing platform
$\hat{f}$	=	Estimated acceleration caused by drag along a generic component (x, y, or z)
$\mathbf{f}_{\text{touch}}$	=	Interaction forces (excluding the aerodynamic drag force)
ℓ	=	Quadratic cost function for the MPC problem
N	=	Timestep when the MAV has to reach the final state
$\mathbf{p}_{\text{plat}}$	=	2D coordinates of the landing platform
$\mathbf{p}$	=	Position of the MAV expressed in the inertial frame W
$\mathbf{q}_W^B$	=	Attitude quaternion of the MAV w.r.t. the inertial frame W
$\mathbf{R}_W^B$	=	Rotation matrix associated with the attitude quaternion $\mathbf{q}_W^B$
$\mathbf{R}_{W,G}^B$	=	Rotation matrix of the ground vehicle w.r.t. the inertial frame
$\bar{\mathbf{u}}$	=	Jerk of the planned trajectory
$\bar{u}_{\max}$	=	Constant that limits the maximum jerk
u	=	Control input along a generic component (x, y, or z)
$\mathbf{u}_{\text{wind}}$	=	Unit vector in the direction of the wind
v_p	=	Magnitude of the landing platform's velocity
$\mathbf{v}_{\text{wind}}$	=	Wind-speed expressed in world frame
$\mathbf{v}$	=	Velocity of the MAV in world frame
$\mathbf{v}_\infty$	=	Relative airflow of the MAV
W	=	Inertial (World) reference frame
$\mathbf{w}_U$	=	Uniform simulated wind
$\mathbf{w}_G$	=	Ground vehicle simulated wind
$\mathbf{w}^T$	=	Total simulated wind experienced by the MAV
$\mathbf{w}_n$	=	Nominal wind speed of the simulated wind
$\mathbf{w}_\sigma$	=	Standard deviation vector of the simulated wind
$\mathbf{x}$	=	State of the MAV (position, velocity, acceleration)

*Graduate Research Assistant, Department of Aeronautics and Astronautics, MIT
†Graduate Research Assistant, Department of Aeronautics and Astronautics, MIT.
‡Richard C. Maclaurin Professor of Aeronautics and Astronautics, Department of Aeronautics and Astronautics, MIT

$\boldsymbol{x}_{\text{w.e.}}$	=	State vector of the wind estimator based on an UKF (+ LSTM)
$\boldsymbol{x}_{\text{tracker}}$	=	State vector of the landing platform tracker
x_d	=	Desired position along a generic component (x, y, or z)
θ	=	Orientation angle of the landing platform w.r.t. the x-axis of the inertial frame W
$\boldsymbol{\phi}_i$	=	Vector containing the roll and pich angles of the i-th whisker-like wind sensor
$\boldsymbol{\sigma}$	=	Vector containing the standard deviation of the turbulent wind speed per component
${}_B\boldsymbol{\omega}$	=	Angular velocity of the MAV

II. Introduction

Landing a micro aerial vehicle (MAV) autonomously on a moving platform in challenging conditions is a problem of interest for applications that require collaboration of MAVs and ships [1, 2] or ground vehicles, such as truck-drone delivery systems [3, 4]. Under these circumstances, precise control robust to disturbances is of utmost importance to ensure the success of the mission. Previous works dealing with autonomous MAV landing did not make special considerations to compensate the turbulent environment around a quickly moving ground vehicle [5–8] or ship [9], or the wind conditions had to be measured beforehand [10].

Actively taking into account in the controller the effects of the drag force caused by the wind can make the MAV more robust to unknown atmospheric conditions and wind gusts. To estimate the wind conditions affecting a MAV, some approaches use pressure sensors [11], ultrasonic sensors [12], or whisker-like sensors [13] to directly measure the wind conditions affecting the MAV. Another strategy is to employ model-based approaches [14, 15], learning-based approaches [16, 17], or hybrid (model-based and learning-based) solutions [18] that leverage the inertial effects produced by the wind on the MAV to obtain an estimate of the drag force. Ref. [19] compares model predictive and PID controllers that use a disturbance estimator capable of rejecting wind of up to 12 m/s. Both an extended and unscented Kalman filter that estimate the wind are analyzed. In [20], an adaptive robust controller based on sliding mode control is presented, which is capable of rejecting wind disturbances of up to 3.8 m/s. In our previous work [21] we introduced a model- and a deep-learning based strategy to differentiate between aerodynamic and interaction forces with bio-inspired sensors, and presents a comparison of both approaches.

In this work we integrate our previous approaches about agile landing on a moving platform [10, 22] and wind estimation [21, 23] providing a comprehensive solution for the problem of landing on a moving platform in unknown wind conditions. In the proposed approach, the wind is estimated by fusing the measurements provided by whisker-like sensors [23] with inertial and rotors speed data via an unscented Kalman filter (UKF) and a recurrent neural network (RNN) [21]. Then, a disturbance-aware boundary layer sliding controller calculates the control inputs by taking into account the characteristics of the estimated turbulent conditions [10, 22]. The main contribution of this work consists in combining the previously presented control and estimation strategies, and in providing a thorough evaluation of the approach via simulation results, in landing experiments in turbulent wind conditions.

The rest of this paper is organized as follows. Section III details the main components of our approach, except for the disturbance-aware boundary layer sliding controller and the model- and deep-learning based wind estimation strategy, which are explained in Section IV and Section V, respectively. Our approach is evaluated in Section VI and, finally, Section VII concludes this paper.

III. Approach

Our approach for agile landing on a moving platform in turbulent wind conditions is divided in two main steps. First, the MAV approaches the landing platform and detects it via the on-board camera. Once the platform is detected, the MAV initiates a dynamic landing trajectory generated by a planner based on model predictive control (MPC), while disturbance rejection is guaranteed thanks to the wind estimate provided by a UKF which is used by our control strategy. We use a tracker based on an extended Kalman filter (EKF) to generate a smooth estimate of the landing target, and a state machine coordinates the transition between the phases of the maneuver.

This section provides a system-level overview of the main components of the approach. A system-level diagram is illustrated in figure Fig. 1. The controller used in this work is explained in detail in Section IV, and Section V describes the model- and deep-learning based wind estimation strategies used in this paper.

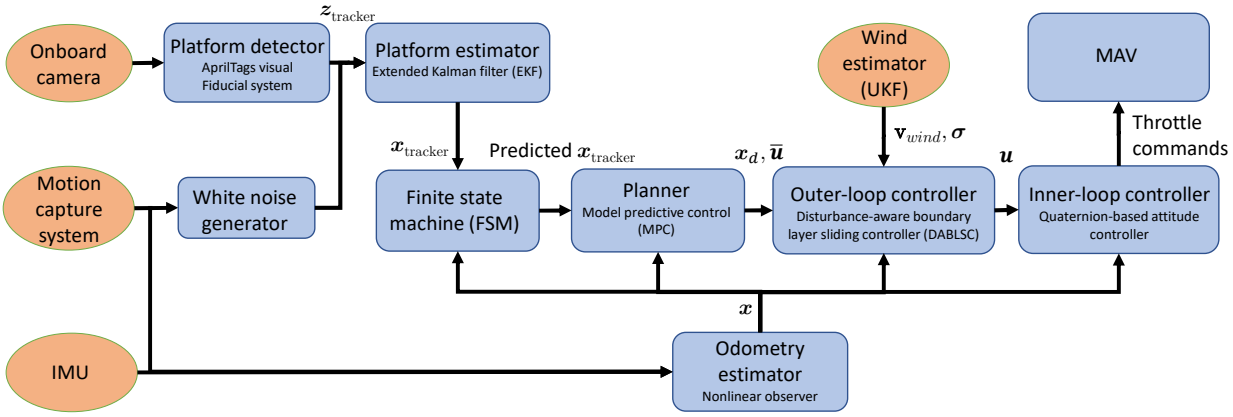

Figure 1 Diagram of the system architecture. The inputs are represented with orange circles, while blue rectangles represent the components. The direction and contents of the exchanged information is indicated with labeled arrows.

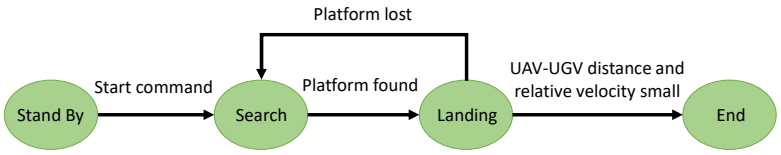

Figure 2 Diagram of the finite state machine.

A. Finite State Machine

The landing maneuver is implemented and executed via a four-states finite state machine (FSM), as Fig. 2 shows. The FSM states are:

1) *Stand By*. After taking off, the MAV hovers waiting for the command to start the maneuver.
2) *Search*. The MAV receives GPS coordinates of the landing platform, the unmanned ground vehicle (UGV), and predicts a rendez-vous position assuming constant linear and angular velocity of the landing platform and a nominal cruise speed of the MAV. This position is then offset by a small distance backwards in the direction of the UGV to ensure target detection by the front-facing camera. At all moments, the MAV's onboard camera images are analyzed to detect the landing platform. When this occurs, the state is switched to *Landing*.
3) *Landing*. The planner (see Section III.B) generates a smooth, dynamic landing trajectory from the MAV to the UGV. If the platform is lost at any time during this state, it switches back to *Search*. Otherwise, when the relative MAV-UGV position and velocity are small enough, the state switches to *End*.
4) *End.* The motors are killed and the maneuver is finished.

B. Dynamic Landing Planner

The planner solves a convex optimization problem with objectives and constraints that depend on the FSM state. In *Search* mode, the planner generates a minimum-time trajectory from the MAV to the predicted rendez-vous point with the UGV. Minimum time is chosen because of its usefulness in applications such as drone delivery, to reduce turnarounds. In *Landing* mode, the planner generates a trajectory with a final position corresponding to the platform's position offset a few cm backwards and predicted ahead by considering the computation time, assuming constant linear/angular velocities for the platform. The final velocity of the trajectory is set to match the UGV's. In this mode, we initially minimize the jerk to obtain a trajectory which facilitates adequate tag acquisition by (implicitly) minimizing motion blur on the vision-based target detection. Such minimum jerk approach produces smooth trajectories and has a long heritage for planning MAV paths [5, 24, 25]. As the UAV approaches the target, the effect of disturbances increases

so the planner minimizes the time spent in the turbulent area.

The trajectory is re-planned using an MPC approach every time a new estimate of the platform's position is obtained. CVXGEN [26] generated the code to solve the following convex optimization problem

$$\begin{aligned} \min_{\mathbf{x},\bar{\mathbf{u}}} J &= \sum_{t=0}^{N} \ell\left(\bar{\mathbf{u}}, d_{\text{land}}\right)[t] \\ \text{subject to } \mathbf{x}[t+1] &= \mathbf{A}\mathbf{x}[t] + \mathbf{B}\bar{\mathbf{u}}[t] \qquad \text{for } t = 0 \ldots N \\ |\mathbf{a}[t]|_\infty &\leq a_{\max} \\ |\bar{\mathbf{u}}[t]|_\infty &\leq \bar{u}_{\max} \\ \mathbf{x}[0] &= \mathbf{x}_0 \\ \mathbf{x}[N] &= \mathbf{x}_f \end{aligned} \tag{1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the state and input matrices for a triple integrator, and $\mathbf{x}^\top = [\mathbf{p}^\top, \mathbf{v}^\top, \mathbf{a}^\top]$ (expressed in the inertial frame W). Note that it is possible to plan using this linear model for a nonlinear system because the nonlinear dynamics of the MAV are canceled by the ancillary controller, which is detailed in Section IV.

C. Landing Platform Detection and Estimation

In order to rendez-vous, detect, and land on the moving platform, a method for estimating and predicting the platform's motion is necessary. Our approach is to use AprilTags for detection, and an EKF for the estimation part.

The AprilTags visual fiducial system [27, 28] is used to detect the landing platform with increased precision (with respect to GPS) when the MAV-UGV distance is smaller than approximately 3.5 m. This algorithm analyzes the images and attempts to find a tag bundle. If one is found, it outputs the relative pose (position and orientation) of the MAV and the bundle, which is used in the estimator explained next.

To estimate the landing platform's state, the EKF in [10] is used. We assume that the platform only moves along a planar surface, and its motion can be modeled as the one of a unicycle with constant linear and angular velocity. The state vector of the platform is

$$\boldsymbol{x}_{\text{tracker}}^\top = [\mathbf{p}_{\text{plat}}{}^\top, v_p, \theta, \dot{\theta}] \tag{2}$$

while the measurement vector provided by the AprilTags detector is

$$\boldsymbol{z}_{\text{tracker}}^\top = [\mathbf{p}_{\text{plat}_m}{}^\top, \theta_m] \tag{3}$$

where the subscript m (x_m) denotes the measurement of a variable (x). When a measurement is received, the EKF approach is used to perform an update of the estimated state. The measurements are obtained in two ways. First, when the MAV is far from the platform (that is, in the *Search* state defined in Section III.A), these measurements are obtained by simulated GPS measurements that a ground vehicle could provide to the UAV for the rendezvous. Note that receiving θ_m is not necessary to estimate the orientation of a moving platform because its motion could be used to infer that quantity. Nevertheless, θ_m measurements are used in this work, which enables testing for static platform experiments. The update frequency is 2 Hz, which is realistic for UAV applications [29]. These first set of measurements are simply used to help the UAV locate the ground vehicle, but that information could be estimated without requiring a link between the two vehicles.

Second, and most importantly, when the MAV is near the ground vehicle and detects the platform (via its onboard tag), both the simulated-GPS and the visual detection measurements are fused to estimate $\boldsymbol{x}_{\text{tracker}}$. The 2D position $\mathbf{p}_{\text{plat}_m}$ and the heading angle θ_m are then used to update the platform's state, and the estimated velocity $\hat{v}_p$ is incorporated into the vector $\mathbf{x}_f$ in (1) to ensure the MAV matches the moving platform's speed at the landing point.

IV. Position Controller Based on Boundary Layer Sliding Control

In order to allow flight and landing in challenging wind conditions, a robust control strategy is required. The role of the outer-loop controller presented in this section is to track the smooth trajectory generated by the planner, using the wind velocity estimate provided by the UKF presented in Section V. Then, a quaternion-based inner-loop attitude controller generates the throttle commands for each motor. In this work, we use the disturbance-aware boundary layer sliding controller (DABLSC) [10] due to its demonstrated efficacy and robustness to unknown, but bounded, disturbances [30].

The following derives the DABLSC along one of the coordinates of the inertial frame W. Note that, for simplicity, the indices representing each coordinate are omitted (e.g. $s = s_i$ for $i = x, y, z$). That is, the absence of the bold notation on a vector means that we are considering one of the three components.

Define the manifold $S(t)$ by $s = \dot{\tilde{x}} + \lambda\tilde{x} = 0$, where $\tilde{x} = x - x_d$ and $\lambda > 0$. The objective of sliding control is to maintain $s = 0$ at all times. The strictly positive constant λ can be intuitively seen as the relative weight placed on the position error with respect to the velocity error, and the slope of the sliding control surface plotted on the phase plane is precisely $-\lambda$. If the control action's frequency is high enough, zero tracking error is guaranteed [30]. This high control action is impractical in many applications because of actuator limits and the excitation of unmodelled dynamics. An approach taken in [31, 32] is boundary layer sliding controller (BLSC), where the discontinuous control law that keeping $s = 0$ entails is smoothed in a thin boundary layer of thickness Φ: $\mathcal{B} := \{s : |s| \leq \Phi\}$. Outside the boundary layer $\mathcal{B}$ the control law remains the same, but inside $\mathcal{B}$, u is decreased proportionally.

We assume that the MAV translational dynamics along every axis can be expressed as

$$\ddot{x} = f + bu + d \tag{4}$$

Then, the BLSC control input is computed as

$$u = \hat{b}^{-1}\left[\ddot{x}_d - \lambda\dot{\tilde{x}} - \hat{f} - K\,\mathrm{sat}\left(\frac{s}{\Phi}\right)\right] \tag{5}$$

where $\mathrm{sat}(\cdot)$ is the saturation function, and K is determined by the uncertainty in the dynamics and the disturbance of the system. As noted before, in (1) it is possible to plan using linear MPC because of the cancellation of f in (4) and (5).

The turbulent wind parameters are the mean $\mathbf{v}_{\text{wind}}$ and standard deviation σ of the wind speed. Define $\hat{\mathbf{v}}_\infty = \hat{\mathbf{v}}_{\text{wind}} - \mathbf{v}$ where $\mathbf{v}$ is the MAV's speed. Then, $\hat{\mathbf{v}}_\infty$ is the total estimated wind speed relative to the UAV and $\hat{f}$ is $\hat{f} = \hat{c}\,\|\hat{\mathbf{v}}_\infty\|\,\hat{v}_\infty$, where $\hat{c}$ is the estimated drag coefficient of the MAV. Our model of the true acceleration caused by drag is

$$f = c\,\|\hat{\mathbf{v}}_\infty \pm 2\boldsymbol{\sigma} \odot \mathbf{u}_{\text{wind}}\|\,(\hat{v}_\infty \pm 2\sigma u_{\text{wind}}) \tag{6}$$

where $\odot$ represents the element-wise product.

The variation of b is very small for a UAV with constant weight. Therefore, $\beta = (b_{max}/b_{min})$ is approximately 1, where b_{max} and b_{min} are the maximum and minimum control gains respectively (or throttle gains in the context of MAVs). Thus, K is simplified as [30] $K = \bar{F} + \eta$, where $\eta > 0$ is a constant in the sliding condition

$$\frac{1}{2}\frac{d}{dt}s^2 \leq -\eta|s|. \tag{7}$$

The larger the η, the faster the system will reach the sliding surface. Nevertheless, K should only be as large as the disturbance magnitude requires to avoid a high-frequency control signal. $\bar{F}$ is

$$\begin{aligned} F &= |f - \hat{f}| \leq \bar{F} \\ \bar{F} &= |\,(\hat{c} + \tilde{c})\,\|\hat{\mathbf{v}}_\infty \pm 2\boldsymbol{\sigma} \odot \mathbf{u}_{\text{wind}}\|\,(\hat{v}_\infty \pm 2\sigma u_{\text{wind}}) - \hat{c}\,\|\hat{\mathbf{v}}_\infty\|\,\hat{v}_\infty| \end{aligned} \tag{8}$$

where $\tilde{c} > 0$ is a bound on the absolute value of the drag coefficient error $|c - \hat{c}|$. By taking the sign that makes this coefficient larger, K is defined. In this application, the MAV moves towards the wind and therefore this occurs when the 2σ is **increasing** the magnitude of $\mathbf{v}_{\text{wind}}$.

V. Model- and Deep Learning-based Wind Estimation

The previous section explained the control strategy that takes into account the disturbances, which requires knowledge of the wind conditions to determine $\hat{f}$ and K. This section details how this information is estimated. The proposed strategy is based on the work presented in [21], and we summarize it here for completeness.

In our approach we estimate the velocity of the wind using information provided by bio-inspired, whisker-like sensors that measure the airflow around a multirotor, as shown in Fig. 3. By fusing the information of four heterogeneous airflow-sensors distributed across the surface of the robot, we can create a three-dimensional estimate of the relative velocity of the MAV with respect to the surrounding airflow. This information is then fused in a UKF-based force estimator that uses an aerodynamic model together with the robot's pose and velocity to predict the wind.

To account for the complex aerodynamic interactions between sensors and propellers [33, 34], we extend this model-based approach (based on first-order physical principles) with a data-driven strategy. This strategy employs a

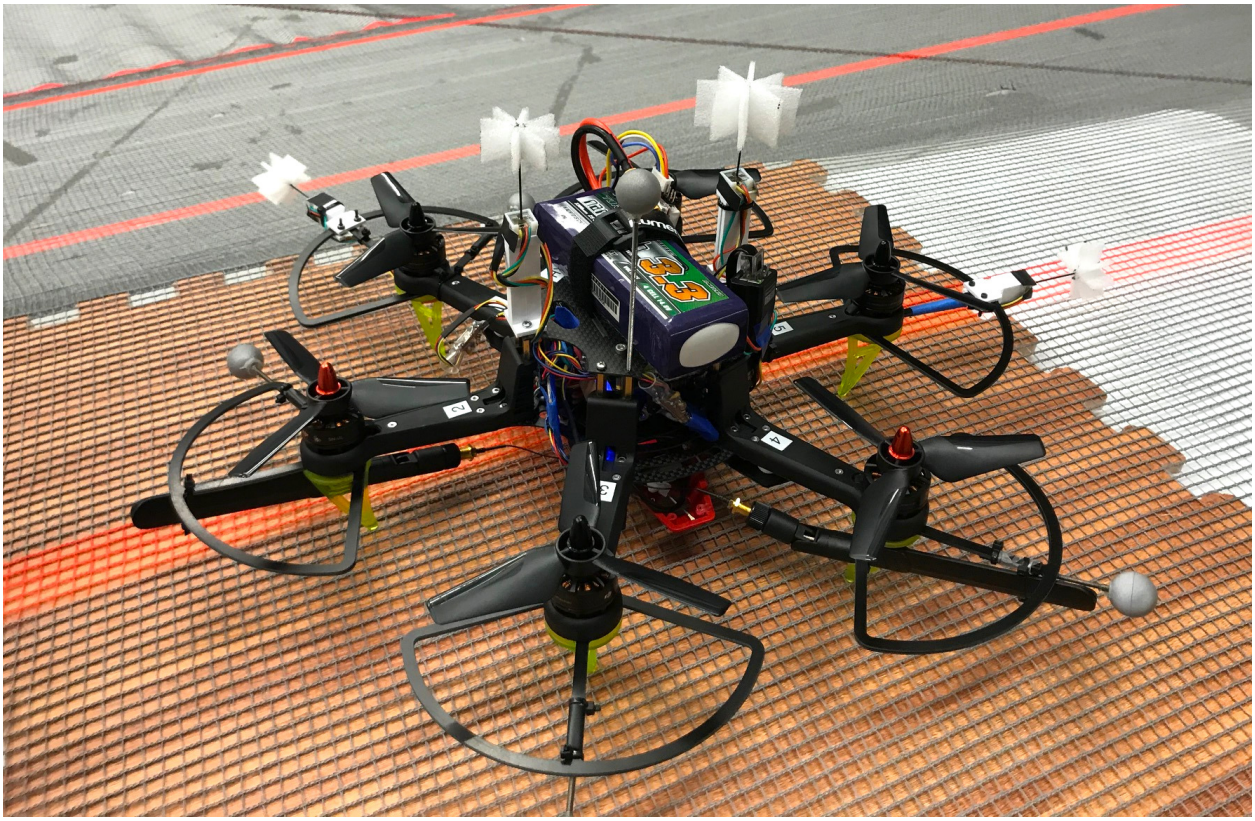

Figure 3 MAV equipped with four bio-inspired airflow sensors used to estimate a three-dimensional wind vector.

RNN based on a long short-term memory (LSTM) network to provide an estimate of the relative airflow of the robot, which is then fused in the proposed model-based estimation scheme.

Users can choose to employ one or the other estimation strategy, according to their ability to easily collect training data (to use the more accurate learning-based approach) or the need of swapping/modifying the sensors (to take advantage of the easily re-configurable model-based approach).

A. Airflow sensor

We utilize multiple lightweight and economical airflow sensor presented in our related work [21, 23, 35], and shown in Fig. 4. The sensor consist of a base and a tip. The base is composed of a magnetic field sensor which measures the rotation of the tip with respect of the base. The tip consists eight foam fins and is hinged to the base via a torsional spring. When the sensor is subjected to airflow, the drag force from the air on the fins causes a rotation about the center of the torsional spring. Such rotation is expressed as

$$\boldsymbol{\phi}_i = \begin{bmatrix} \phi_{x,i} \\ \phi_{y,i} \end{bmatrix} \tag{9}$$

We mount four sensors on our MAV, two on the top (vertically mounted) and two on the propeller guards (horizontally), since those are the locations that are less subject to the aerodynamic interference caused by the propellers. The sensors' orientation is chosen so that the relative airflow coming from any direction excites at least one sensor (that is, for at least one sensor, the relative airflow is not aligned with its length).

B. Model-based approach

The model-based approach uses a model of the translational and rotational dynamics of the UAV, as well as a model mapping the whisker's roll and pitch angle to relative airflow in order to estimate the wind via an unscented Kalman

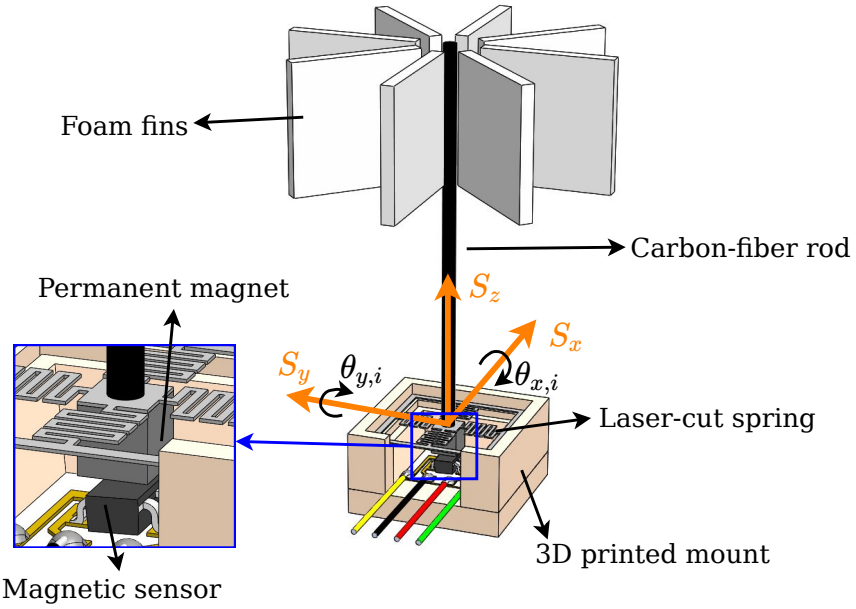

Figure 4 Illustration of an airflow sensor and its reference frame S, with the main components labeled, from [21].

filter. The approach is based on our previous work [36, 37], where more modelling details can be found. Note that all the variables are expressed in the inertial reference frame W, except when otherwise indicated by a pre-script.

1. Output, state and input of the filter

The output of the proposed unscented Kalman filter is given by:

$$\mathbf{y}^\top = [\hat{\mathbf{f}}_{\text{touch}}{}^\top, \hat{\mathbf{v}}_{\text{wind}}{}^\top] \tag{10}$$

Using $\hat{\mathbf{v}}_{\text{wind}}$ and its covariance, the drag force and the parameter K needed for the DABLSC can be computed as shown in Section IV. The estimate of the interaction forces provided by our approach is not explicitly used in this work but could be used, for example, to detect accidental collisions, verify the landing on the platform, or check for interaction from a human.

The full, discrete-time state of the filter used for wind estimation is

$$\boldsymbol{x}_{\text{w.e.}}^\top = [\mathbf{p}^\top, \mathbf{q}_W^B{}^\top, \mathbf{v}^\top, {}_B\boldsymbol{\omega}^\top, \mathbf{f}_{\text{touch}}{}^\top, \mathbf{v}_{\text{wind}}{}^\top] \tag{11}$$

The filter generates its output by asynchronously fusing the following three sets of measurements:

1) **Odometry** measurements: Position $\mathbf{p}_m$, attitude quaternion $\mathbf{q}_{W\,m}^B$, linear velocity $\mathbf{v}_m$ and angular velocity ${}_B\boldsymbol{\omega}_m$ measurements are provided by a cascaded on-board odometry estimator

$$\mathbf{z}_{\text{odometry}}{}^\top = [\mathbf{p}_m{}^\top, \mathbf{q}_{W\,m}^B{}^\top, \mathbf{v}_m{}^\top, {}_B\boldsymbol{\omega}_m{}^\top] \tag{12}$$

2) **Airflow** measurements: We assume that the N sensors are sampled synchronously, providing the measurement vector

$$\mathbf{z}_{\text{airflowsensor}}{}^\top = [\boldsymbol{\phi}_1{}_m^\top, \ldots, \boldsymbol{\phi}^\top{}_{N\,m}] = \boldsymbol{\phi}_m{}^\top \tag{13}$$

where the function mapping the measurement $\mathbf{z}_{\text{odometry}}{}^\top$ to the filter's state $\boldsymbol{x}_{\text{w.e.}}$ is obtained using first-order physical principles (hence motivating the fact that this approach is fully model-based), for example by assuming that the deflection on the sensor is proportional to the squared magnitude of the airflow acting on the sensor.

3) **Rotors' speed**. The filter uses the commanded rotors' speed to propagate the robot's dynamics in the prediction phase.

C. Learning-based approach

The learning-based strategy makes use of a RNN based on the LSTM architecture to create an estimate of the relative airflow ${}_B\mathbf{v}_\infty$ using the airflow sensors and other measurements available on board the robot. Such estimate is then fused in the UKF detailed above, instead of directly using the airflow measurement vector $\mathbf{z}_{\text{airflowsensor}}$.

We employ an LSTM architecture, which is able to capture time-dependent effects [38, 39], such as, in our case, the dynamics of the airflow surrounding the robot and the dynamics of the sensor. We chose a 2-layer LSTM, with the size of the hidden layer set to 16 (with the input size, 20, and the output size, 3). We add a single fully connected layer to the output of the network, mapping the hidden layer into the the desired output size.

1. Output and inputs of the LSTM

The output of the network is the relative airflow measurement ${}_B\mathbf{v}_{\infty_m}$ of the MAV (expressed in the body-fixed reference frame B). The inputs to the network are the airflow sensor measurements $\boldsymbol{\phi}_m$, the angular velocity of the robot ${}_B\boldsymbol{\omega}_m$, the raw acceleration measurement from the IMU, and the normalized throttle commanded to the six propellers (which ranges between 0 and 1). The sign of the throttle is changed for the propellers spinning counterclockwise, in order to provide information to the network about the spinning direction of each propeller. The reason for the choice of the input is dictated by the fact that the relative airflow depends on the angle of the sensors and on the angular velocity of the robot. The acceleration from the IMU is included to provide information about hard to model effects, such as the orientation of the body frame w.r.t. gravity (which causes small changes in the angle measured by the sensors), as well as the effects of accelerations of the robot. Information about the throttle and spinning direction of the propellers is added to try to capture the complex aerodynamic interactions caused by their induced velocity. We chose to express every output and input of the network in the body reference frame, in order to make the network invariant to the orientation of the robot, thus potentially reducing the amount of training data needed.

2. Interface between LSTM and UKF

The UKF treats the LSTM output as a new sensor which provides relative airflow measurements ${}_B\mathbf{v}_{\infty_m}$, replacing the model-based airflow sensor's measurement used in the model-based approach:

$$\mathbf{z}_{\text{LSTM}} = [{}_B\mathbf{v}_{\infty_m}] \tag{14}$$

The filter state is mapped to the LSTM output using the following measurement model:

$${}_B\mathbf{v}_{\infty_m} = \mathbf{R}_W^{B\top}\mathbf{v}_\infty = \mathbf{R}_W^{B\top}(\mathbf{v}_{\text{wind}} - \mathbf{v}) \tag{15}$$

VI. Evaluation

This section presents the results of the approach described previously. Simulations are used to test this paper's methods, and its components are detailed next.

A. Simulation Environment

MAV Simulation ACL's snap sim [40] is used to simulate the MAV's dynamics. This ROS package implements software-in-the-loop (SIL) simulations of the Aerospace Controls Laboratory's snap stack [41] and incorporates a physics engine. That is, the IMU and motion capture measurements are simulated realistically and interfaced with the rest of the components to allow the same inner-loop quaternion-based attitude controller and outer-loop disturbance-aware BLSC to be run without any changes in the code.

The effects simulated are motor thrusts and torques, drag force, wind speed (explained in Section VI.A), and gravity. The drag force acting on the CoM of the MAV is calculated as:

$$\mathbf{f}_{\text{drag}} = c\,\|\mathbf{v}_\infty\|\,\mathbf{v}_\infty \tag{16}$$

Airspeed Sensor Simulation The airflow sensors presented in Section V.A are simulated by calculating the deflection angles $\boldsymbol{\phi}_i$ caused by the drag force using the model presented in [21], and are used by the wind estimator. For all the experiments presented in this work, the exact same 4 sensors shown in Fig. 3 have been reproduced. The whisker's drag coefficient, length, stiffness, and aerodynamic section of each dimension are set to the experimentally found values.

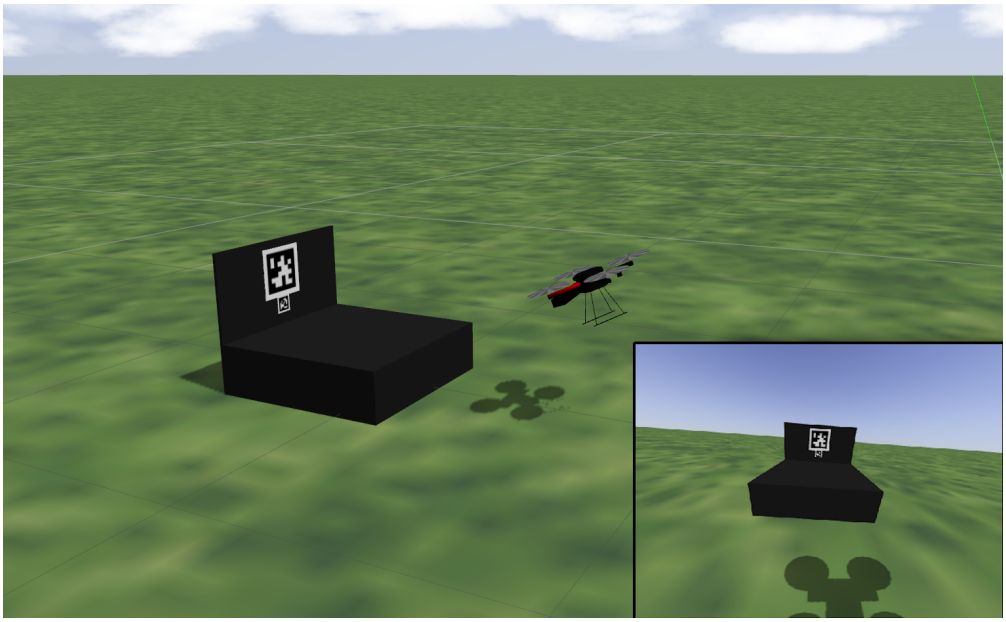

Figure 5 Gazebo environment used to simulate the UGV, the tag bundle, and for visualization. The bottom-right part shows the view from the 640×480 onboard camera.

For each sensor i and timestep k, the sensor's $\boldsymbol{\phi}_{i,k}$ (a 2-dimensional vector that represents the roll and pitch angles) is simulated and sent to the wind estimator.

Wind Simulation The Dryden turbulence model in [42] is used to replicate the turbulent conditions that would be found in hardware experiments were a MAV lands on a moving platform at a high speed. This implementation of a Dryden turbulence model is similar to a random walk with a certain mean value (nominal wind) and standard deviation (wind gust). To control the intensity of the turbulence, a constant $c_t \geq 1$ has been added, which multiplies the wind simulation step dt and thus increases the time variance of the wind. The constant c_t is nominally 1, but it can be increased to increment the changes in the generated wind.

Two wind generators run simultaneously:

- Uniform wind generator: represents uniform wind conditions, such as light gusts and a constant wind.
- Ground vehicle wind generator: simulates the turbulent wind present on the back of a ground vehicle, stronger the closer to this vehicle.

Each generator is characterized by the nominal wind speed $\mathbf{w}_n$ and standard deviation per component $\mathbf{w}_\sigma$. The total wind $\mathbf{w}^T$ experienced by the MAV is

$$\mathbf{w}^T = \mathbf{w}^U + e^{\frac{-d_{\text{land}}}{\tau}} \mathbf{R}^B_{W,G} \mathbf{w}^G \tag{17}$$

UGV Simulation and Visualization Gazebo is used to simulate the ground vehicle, the AprilTag bundle, and also for visualization of the landing maneuver. Figure 5 shows the tag bundle attached to the UGV, the simulated quadrotor performing the dynamic landing, and the images with resolution 640×480 from the simulated camera. The bundle consists of a 14×14 cm tag on top of a 5×5 cm tag. We add a white Gaussian noise to the ground truth pose of the UGV to simulate the inaccurate GPS coordinates that the vehicle provides.

Experiment Setup The platform is moving towards the negative x-direction of the inertial reference frame W, along a flat surface, at a constant speed of 1 m/s. This value was chosen to recreate the experimental setup of our previous works [10, 22] for easier comparison. Furthermore, strong, turbulent wind is simulated, and the important factor of this paper is the relative airspeed acting on the MAV instead of its absolute speed with respect to an inertial frame.

The generated wind has the parameters shown in Table 1.

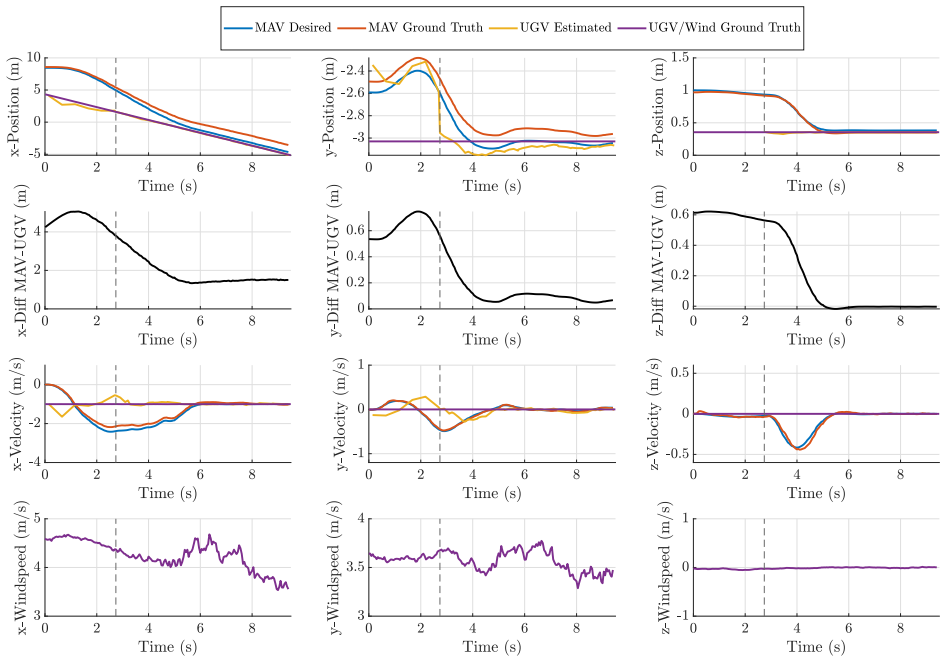

Figure 6 Moving platform experiment with a standard BLSC. The first and third rows show the position and velocity tracking and estimation performance of the simulated moving platform experiment The second row shows the difference of the MAV and the UGV position, in black. The MAV is not able to reach the UGV, keeping a large tracking error in the x-direction. The last row shows the ground truth wind velocity, which becomes more turbulent near the UGV. Vertical dashed lines indicate time of the first tag detection, t_d.

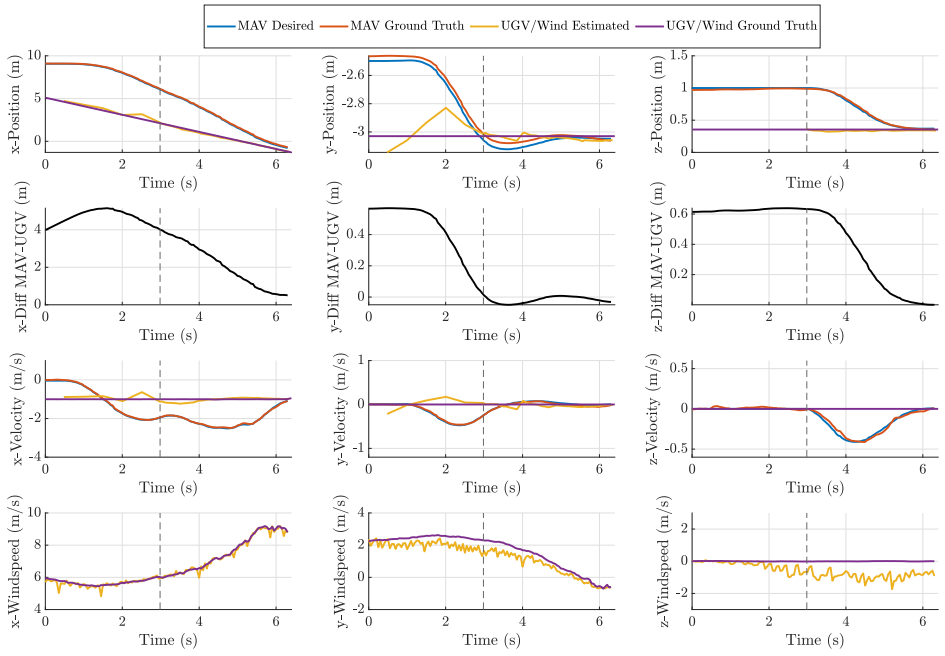

Figure 7 Moving platform experiment with the disturbance-aware BLSC and online model-based wind estimation. The first and third rows show the position and velocity tracking and estimation performance of the simulated moving platform experiment. The second row shows the difference of the MAV and the UGV position, in black, which approaches 0 for all three components. It can be seen that the vehicle is able to reach the UGV and to track the desired trajectory. The last row shows the estimated and ground truth wind velocity. Vertical dashed lines indicate time of the first tag detection, t_d.

Table 1 Wind parameters used for the simulations.

Parameter		Symbol	Value	Unit
Uniform generator	Nominal wind speed	$\mathbf{w}_n^U$	$[5, 3, 0]^\top$	m/s
	Standard deviation	$\mathbf{w}_\sigma^U$	$[1, 1, 0.2]^\top$	m/s
	Simulation speed const	c_t^U	1	-
Ground vehicle generator	Nominal wind speed	$\mathbf{w}_n^G$	$[-2, 0, 0]^\top$	m/s
	Standard deviation	$\mathbf{w}_\sigma^G$	$[3, 1.5, 0]^\top$	m/s
	Simulation speed const	c_t^G	4	-
Inverse of the exponential decay constant		τ	2	m

B. Baseline Results

First, a baseline approach was tested: a BLSC that does not take into account the turbulent conditons present. That is, we assume that no wind-estimator is available and thus $\hat{f}$ and K consider only the drag caused by the MAV's speed with respect to the inertial frame. The results are shown in Fig. 6. The figure consists in a matrix of plots where columns represent the x, y, and z-coordinates. The first row represents the position: MAV desired position, MAV ground truth, UGV estimated position, and the UGV ground truth in blue, red, yellow, and purple respectively. The second row shows the difference in the position of the MAV and the UGV. Similarly to the first row, the third row shows the velocity for all three components. Finally, the last row represents the estimated and ground truth wind speed for each component. The time of the first tag detection, t_d, is shown with a vertical dashed line.

As it can be seen, the MAV initially approaches the UGV but then keeps a constant tracking error in the x-coordinate of about 2 m, making the landing not possible. This is expected, since there is a higher mean and standard deviation of the wind – which is not accounted for – in this direction, and the UGV moves in this direction as well. As would happen in hardware experiments, the wind is more turbulent when the MAV approaches the ground vehicle. Also note that before the first tag detection, the estimation of the UGV's y-position is not accurate (the error is approximately +60 cm), and the MAV moves in the positive y direction even though the landing platform is in the negative y direction with respect to the vehicle. This is due to consecutive noisy GPS measurements offset in the positive y, but the estimation error decreases abruptly when the tag is first detected due to the increased precision of the vision system described in Section III.C.

C. Disturbance-aware BLSC Results

This subsection presents the results of our disturbance-aware BLSC and compares the wind estimation approaches presented in Section V.B and Section V.C.

1. Model-based wind estimation

Figure 7 shows the results of our DABLSC with model-based wind estimation. The plotted lines are the same as in the previous case, but the last row, which represents the wind speed for each component, includes the estimated wind speed in yellow. It can be seen that the tracking error remains small during all the maneuver, the MAV can reach the UGV, and landing occurs 3.4s after the first tag detection (even with a stronger wind compared with the first case, of up to 9 m/s in the x-axis). Note that the UGV's estimation improves greatly after the vertical line, which represents the first visual detection of the tag bundle placed on the vertical pane of the ground vehicle. The wind estimation's accuracy is high, although there is some noise. This is expected since, to match the imperfections of hardware, noise was added to the whisker sensor simulations.

2. Deep learning-based wind estimation

In order to use our DABLSC with the wind estimation approach presented in Section V.C, a training and validation dataset must be obtained. To collect these datasets, the vehicle was flown at different speeds in a non-windy environment. Therefore, the velocity of the MAV as obtained by the motion capture system coincides with the wind speed affecting the airflow sensors and is used to train the LSTM.

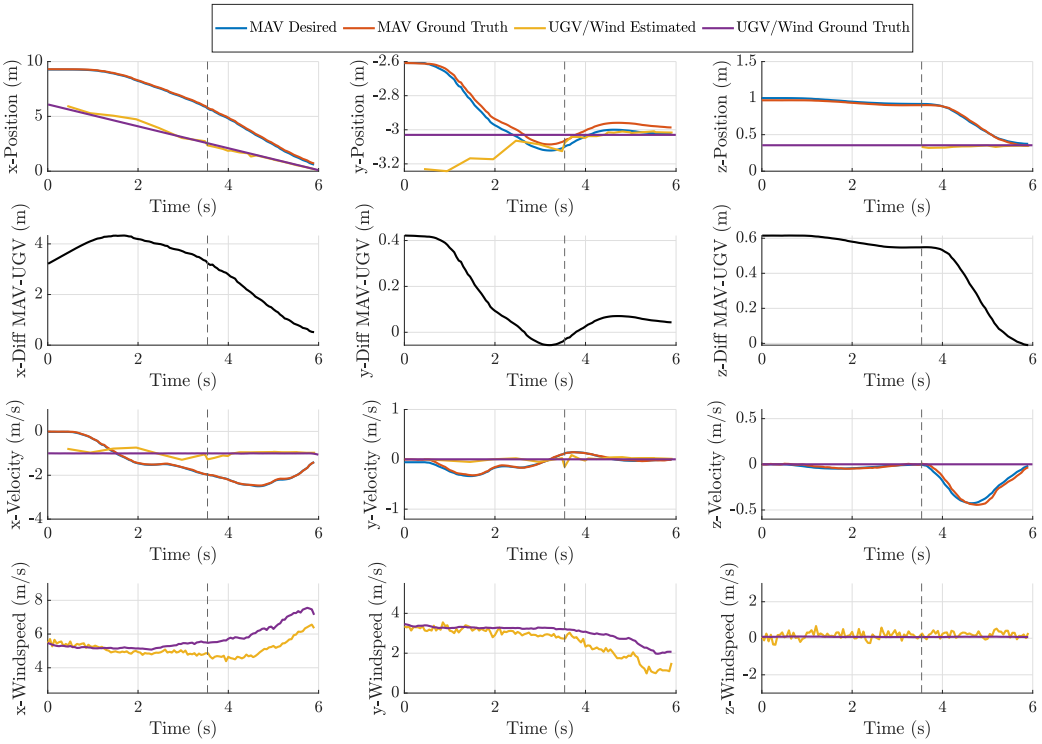

Figure 8 Moving platform experiment with the disturbance-aware BLSC and online deep learning-based wind estimation. The first and third rows show the position and velocity tracking and estimation performance of the simulated moving platform experiment. The second row shows the difference of the MAV and the UGV position, in black, which approaches 0 for all three components. The estimation error is relatively small, and the MAV is able to reach the UGV and to track the desired trajectory.

The results are shown in Fig. 8. As in the previous case, the first row shows that the MAV achieves a fast landing, which takes 2.5s since the vertical tag bundle is detected (vertical dashed line). This is slightly faster than for the model-based approach, but the wind acting on the MAV was larger for that case. The second row shows that the position for the three coordinates approaches zero until landing occurs at t = 6s, and an accurate velocity tracking and platform estimation is seen in the third row. Note that there is a small wind estimation error when the MAV approaches the turbulent area near the UGV. This is due to the fact that the LSTM was not trained in turbulent air. In any case, as explained in Section IV, our approach is robust to bounded disturbances, and the maneuver can be completed successfully.

VII. Conclusion

This paper integrated the disturbance-aware boundary layer sliding controller to allow a MAV to land in challenging conditions presented in [10] and the wind estimation techniques based on a UKF and an LSTM from [21]. The integrated approach was evaluated in simulated experiments, and the results demonstrated the effectiveness of both the model-based and the deep learning-based techniques: the tracking error was small and therefore the MAV could perform the requested maneuver adequately. We observe that integrating a wind estimate (provided by an on-board, online wind estimator) in the proposed DABLSC indeed improved the trajectory tracking performance and allowed a successful landing on a moving platform. In our simulation results, we additionally observed that the LSTM works well, but requires to collect a representative enough data-set, which may be hard in a simulation setup due to the complexity in reproducing the effects of turbulent wind. In the simulated scenario proposed in this paper, the fully model-based technique has an advantage, since the dynamics of the system are known (with the exception of noise).

Future work includes outdoor testing in a real moving vehicle, and further explore the possibilities that our approach,

which can distinguish between drag force and interaction forces, brings on control and planning for MAVs (e.g. not trying to compensate interaction forces, for safety).

Acknowledgments

This work was supported by Ford Motor Company and the Air Force Office of Scientific Research MURI FA9550-19-1-0386.

References

[1] Moriarty, P., Sheehy, R., and Doody, P., "Neural networks to aid the autonomous landing of a UAV on a ship," *2017 28th Irish Signals and Systems Conference (ISSC)*, IEEE, 2017, pp. 1–4.

[2] Tan, C. K., Wang, J., Paw, Y. C., and Liao, F., "Autonomous ship deck landing of a quadrotor using invariant ellipsoid method," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 2, 2016, pp. 891–903.

[3] Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., and Rich, R., "Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm," *Journal of Industrial Engineering and Management (JIEM)*, Vol. 9, No. 2, 2016, pp. 374–388.

[4] Ham, A. M., "Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming," *Transportation Research Part C: Emerging Technologies*, Vol. 91, 2018, pp. 1–14.

[5] Falanga, D., Zanchettin, A., Simovic, A., Delmerico, J., and Scaramuzza, D., "Vision-based autonomous quadrotor landing on a moving platform," *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, IEEE, 2017, pp. 200–207.

[6] Borowczyk, A., Nguyen, D.-T., Phu-Van Nguyen, A., Nguyen, D. Q., Saussié, D., and Le Ny, J., "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *IFAC-PapersOnLine*, Vol. 50, No. 1, 2017, pp. 10488–10494.

[7] Xing, B.-Y., Pan, F., Feng, X.-X., Li, W.-X., and Gao, Q., "Autonomous Landing of a Micro Aerial Vehicle on a Moving Platform Using a Composite Landmark," *International Journal of Aerospace Engineering*, Vol. 2019, 2019.

[8] Bähnemann, R., Pantic, M., Popović, M., Schindler, D., Tranzatto, M., Kamel, M., Grimm, M., Widauer, J., Siegwart, R., and Nieto, J., "The ETH-MAV team in the MBZ international robotics challenge," *Journal of Field Robotics*, Vol. 36, No. 1, 2019, pp. 78–103.

[9] Persson, L., and Wahlberg, B., "Model predictive control for autonomous ship landing in a search and rescue scenario," *AIAA Scitech 2019 Forum*, 2019, p. 1169.

[10] Paris, A., Lopez, B. T., and How, J. P., "Dynamic Landing of an Autonomous Quadrotor on a Moving Platform in Turbulent Wind Conditions," *2020 International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9577–9583.

[11] Bruschi, P., Piotto, M., Dell'Agnello, F., Ware, J., and Roy, N., "Wind speed and direction detection by means of solid-state anemometers embedded on small quadcopters," *Procedia Engineering*, Vol. 168, 2016, pp. 802–805.

[12] Hollenbeck, D., Nunez, G., Christensen, L. E., and Chen, Y., "Wind measurement and estimation with small unmanned aerial systems (SUAS) using on-board mini ultrasonic anemometers," *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2018, pp. 285–292.

[13] Deer, W., and Pounds, P. E., "Lightweight Whiskers for Contact, Pre-Contact, and Fluid Velocity Sensing," *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, 2019, pp. 1978–1984.

[14] Demitrit, Y., Verling, S., Stastny, T., Melzer, A., and Siegwart, R., "Model-based wind estimation for a hovering VTOL tailsitter UAV," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 3945–3952.

[15] Sikkel, L., de Croon, G., De Wagter, C., and Chu, Q., "A novel online model-based wind estimation approach for quadrotor micro air vehicles using low cost MEMS IMUs," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 2141–2146.

[16] Shi, G., Shi, X., O'Connell, M., Yu, R., Azizzadenesheli, K., Anandkumar, A., Yue, Y., and Chung, S.-J., "Neural lander: Stable drone landing control using learned dynamics," *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9784–9790.

[17] Allison, S., Bai, H., and Jayaraman, B., "Estimating Wind Velocity with a Neural Network using Quadcopter Trajectories," *AIAA Scitech 2019 Forum*, 2019, p. 1596.

[18] Marton, A. S., Fioravanti, A. R., Azinheira, J. R., and de Paiva, E. C., "Hybrid Model-Based and Data-Driven Wind Velocity Estimator for the Navigation System of a Robotic Airship," *arXiv preprint arXiv:1907.06266*, 2019.

[19] Hentzen, D., Stastny, T., Siegwart, R., and Brockers, R., "Disturbance Estimation and Rejection for High-Precision Multirotor Position Control," *arXiv preprint arXiv:1908.03166*, 2019.

[20] Wang, C., Song, B., Huang, P., and Tang, C., "Trajectory tracking control for quadrotor robot subject to payload variation and wind gust disturbance," *Journal of Intelligent & Robotic Systems*, Vol. 83, No. 2, 2016, pp. 315–333.

[21] Tagliabue, A., Paris, A., Kim, S., Kubicek, R., Bergbreiter, S., and How, J. P., "Touch the Wind: Simultaneous Airflow, Drag and Interaction Sensing on a Multirotor," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020.

[22] Paris i Bordas, A., "Control and Estimation Strategies for Autonomous MAV Landing on a Moving Platform in Turbulent Wind Conditions," Master's thesis, Massachusetts Institute of Technology, 2020.

[23] Kim, S., Kubicek, R., Paris, A., Tagliabue, A., How, J. P., and Bergbreiter, S., "A Whisker-inspired Fin Sensor for Multi-directional Airflow Sensing," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020.

[24] Yu, J., Cai, Z., and Wang, Y., "Minimum jerk trajectory generation of a quadrotor based on the differential flatness," *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, IEEE, 2014, pp. 832–837.

[25] Phang, S. K., Lai, S., Wang, F., Lan, M., and Chen, B. M., "Systems design and implementation with jerk-optimized trajectory generation for UAV calligraphy," *Mechatronics*, Vol. 30, 2015, pp. 65–75.

[26] Mattingley, J., and Boyd, S., "CVXGEN: A Code Generator for Embedded Convex Optimization," *Optimization and Engineering*, Vol. 12, No. 1, 2012, pp. 1–27.

[27] Olson, E., "AprilTag: A robust and flexible visual fiducial system," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2011, pp. 3400–3407.

[28] Malyuta, D., "Guidance, Navigation, Control and Mission Logic for Quadrotor Full-cycle Autonomy," Master's thesis, ETH Zurich, 2018.

[29] Salih, A. A. A.-A., Zaini, N. L. A. C. A., and Zhahir, A., "The suitability of GPS receivers update rates for navigation applications," *Proceedings of World Academy of Science, Engineering and Technology*, World Academy of Science, Engineering and Technology (WASET), 2013, p. 192.

[30] Slotine, J.-J. E., Li, W., et al., *Applied nonlinear control*, Prentice hall Englewood Cliffs, NJ, 1991.

[31] Runcharoon, K., and Srichatrapimuk, V., "Sliding mode control of quadrotor," *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, IEEE, 2013, pp. 552–557.

[32] Lopez, B. T., Slotine, J.-J., and How, J. P., "Robust Collision Avoidance via Sliding Control," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2962–2969.

[33] Prudden, S., Fisher, A., Marino, M., Mohamed, A., Watkins, S., and Wild, G., "Measuring wind with small unmanned aircraft systems," *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 176, 2018, pp. 197–210.

[34] Ventura Diaz, P., and Yoon, S., "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 1266.

[35] Kim, S., Velez, C., Patel, D. K., and Bergbreiter, S., "A Magnetically Transduced Whisker for Angular Displacement and Moment Sensing," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 665–671.

[36] Tagliabue, A., Kamel, M., Verling, S., Siegwart, R., and Nieto, J., "Collaborative transportation using MAVs via passive force control," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5766–5773.

[37] Tagliabue, A., Kamel, M., Siegwart, R., and Nieto, J., "Robust collaborative object transportation using multiple MAVs," *The International Journal of Robotics Research*, Vol. 38, No. 9, 2019, pp. 1020–1044.

[38] Lipton, Z. C., Berkowitz, J., and Elkan, C., "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[39] Goodfellow, I., Bengio, Y., and Courville, A., *Deep learning*, MIT press, 2016.

[40] Aerospace Controls Laboratory, "snap_sim: software-in-the-loop (SIL) simulation of ACL's snap-stack," `https://gitlab.com/mit-acl/fsw/snap-stack/snap_sim`, 2020. (Accessed on 05/28/2020).

[41] Aerospace Controls Laboratory, "snap-stack: Autopilot code and host tools for flying Snapdragon Flight-based vehicles," `https://gitlab.com/mit-acl/fsw/snap-stack`, 2020. (Accessed on 05/28/2020).

[42] Torgesen, A., "Wind Dynamics: Algorithms for simulating wind in C++," `https://github.com/goromal/wind-dynamics`, 2019. (Accessed on 04/10/2020).