Preliminary Design of Wind-Aware sUAS Simulation Pipeline for Urban Air Mobility

Asma Tabassum*, Max DeSantis†, He Bai‡ and Nicoletta Fala§
Oklahoma State University, Stillwater, Oklahoma, 74078

We introduce and propose a wind-aware small unmanned aircraft system human-in-the-loop simulation pipeline for operations in low-altitude urban airspace. To complement the Urban Air Mobility concept that envisions safe, sustainable, and accessible air transportation, we design an aggregated simulation pipeline so that Human Machine Interaction during operations in turbulent windy conditions can be studied. In this work, we primarily identify the lack of wind-awareness in the current user interface designs available for small unmanned aircraft system. After a small research group study, we recognize the information components that are crucial in the operational context and integrate specific display information in an extended wind-aware user interface which includes real time wind velocity display and trajectory management features. Additionally, we extend the capability of an existing simulator to simulate spatial-temporal wind data in real time. The simulation pipeline provides a comprehensive structure for wind-aware sUAS simulations which can be used for mission design experiments in the presence of wind and evaluation of pilot response to increased wind situational awareness.

Nomenclature

UAS	=	Unmanned Aircraft System
sUAS	=	small Unmanned Aircraft System
HITL	=	Human-in-the-Loop
HMI	=	Human Machine Interaction
UAM	=	Urban Air Mobility
UI	=	User Interface
OEM	=	Original Equipment Manufacturer
MAV	=	Micro Air Vehicle
GCS	=	Ground Control Station
QGC	=	QGroundcontrol
SME	=	Subject Matter Expert
LES	=	Large Eddy Simulation
SITL	=	Software-in-the-Loop
OSU	=	Oklahoma State University

I. Introduction

ALTHOUGH considerable research has been invested in the development of interaction strategies between human pilots and onboard interfaces for modern and conventional aircraft, less attention has been given to investigating interaction strategies associated with remote Unmanned Aircraft System pilots and onboard command and control [1]. With the aggressive integration of UAS into the National Airspace System, more than 250 prototypes of vertical take-off and landing (VTOL), electric, and autonomous aircraft are being designed and tested to operate in low-altitude airspace, particularly in urban areas [2]. Despite the autonomy, humans are still an integral part of this process. Because of the unique interaction characteristics, where a human is not the primary analyzer of the dynamic situation onboard,

*Graduate Research Assistant and PhD Candidate, Mechanical and Aerospace Engineering, AIAA Student Member
†Undergraduate Student, Electrical and Computer Engineering
‡Associate Professor, Mechanical and Aerospace Engineering
§Assistant Professor, Mechanical and Aerospace Engineering, AIAA Member

UAS-remote pilot interaction demands effective and critical design consideration. A Federal Aviation Administration (FAA) technical report on pilot control interfaces for UAS [3] provides an initial guide for interface design that involves viewpoint design, control level design, and autonomy level design. Related work in UAS UI design research includes conceptual studies of display design criteria [4–7], tools and framework development for UI development [8–10], and pilot assessment and learnability of UI design [11, 12]. In [4], the author emphasizes on the implementation and use of an iterative task analysis throughout the design process to better understand key task components, user needs, and user mental representation for displayed information. The authors in [7] develop and evaluate a display design concept that aims to support safety risk monitoring of multiple sUAS by a human operator for low altitude operation. Other studies [5, 6] also discuss the few design criteria for the sUAS interface. In [8], the authors propose a framework for supervisory control interface design and a basis for enhanced interface design of sUAS. Later, in [10] the authors develop a tool that includes detailed criteria for different aspects of UAS control interface design to support operator performance. Ground control stations of sUAS swarm operations are developed in [13]. Reference [9] illustrates efforts to enhance UI with the coupling of distributed Artificial Intelligence (AI) algorithms using map-based, minimalist interfaces, and highlight the decrease in required user actions to accomplish similar swarm-based maneuvers. The authors in [11] illustrate the necessity of the assessment of a pilot's cognitive states and trusted autonomy. An interface learnability study was conducted recently in [12], where the author established an ex-post facto research design to examine system learnability of industry-standard UAS HMI and compare the performance between three different groups based on experience level: high, low, and none. The results suggest that those individuals with a high level of conventional flight experience (i.e., commercial pilot certificate) performed most effectively when compared to participants with low piloting or no prior experience.

Even with the growing attention, the designs for UAS have been service-branch specific as well as uncoordinated [14] and fail to address one of the most critical hurdles for sUAS urban operations: turbulent wind. A survey conducted in August 2018 among 1702 people showed that around 52% of respondents express increased levels of fear and concern while flying in turbulent wind in UAM [2]. This concern indicates the necessity of an organized effort to increase comfort in piloting sUAS in windy conditions. Flight simulation is a critical part of design assurance and an integral step before a new technology is implemented into hardware and actual airspace. While the FAA has extensive guidance resources on airworthiness regulations, pilot certification, operational standards, and training standards for crewed aircraft flight simulations, standardized compliance resources for sUAS HITL simulations are still limited in comparison to manned aircraft. Performance standard documents [15] focused on the detect-and-avoid requirements for fixed-wing UAS. Traditional flight simulators are inherently integrated, that is, they come with a UI, a flight control software package, and at times original equipment manufacturer components for a particular aircraft. Although a few fixed-wing UAS have advanced flight simulation capabilities with OEM components, the small rotary-wing UAS HITL simulator as an integrated platform is rare. Recent studies have enabled incorporating humans into the simulation by utilizing multiple separate code-bases. A publicly available sUAS simulator can be integrated with open-source flight controllers, which are then connected to a ground control station linking a joystick for human operations. Although this modular simulator technique is complicated due to the cross-platform compatibility requirement, it provides freedom to the community for experimentation and extension. In our work, we will utilize open-source platforms for the simulator, flight controller, and ground control station.

Currently, multiple high fidelity quadcopter simulators are widely used by the community to develop path planning, control, and reinforcement learning algorithms. Simulators based on ROS-Gazebo such as Hector [16] and RotorS [17] are popular Micro Air Vehicle simulators. Hector is a collection of open-source modules primarily used for autonomous mapping and navigation with rescue robots and supports adding multiple sensors such as an Inertial Measurement Unit (IMU) and RGB camera. RotorS has multiple multi-rotor platforms for simulations, including IRIS+, AscTec Hummingbird, Pelican, and Firefly. It also has the ability to add a constant wind field using an external file. Although Gazebo provides a high fidelity physics engine, it lacks the ability to provide a visually rich environment that can closely mimic an actual environment. A photo-realistic environment is a precondition for accurate data-driven artificial intelligent model formulation that vastly depends on the high-quality photo or video training data. AirSim [18] and Flightmare [19] have both overcome this obstacle by using high-quality graphics rendering engines; they are also open source, allowing community modifications. AirSim utilizes the Unreal engine and Flightmare uses the Unity rendering engine. Within AirSim, the rendering and physics engines are tightly coupled, whereas Flightmare has decoupled rendering and physics engines. AirSim also has constant wind simulation capability, supports a variety of open-source flight control software, and complies with the MAVLink protocol. Therefore AirSim can easily connect to a suitable

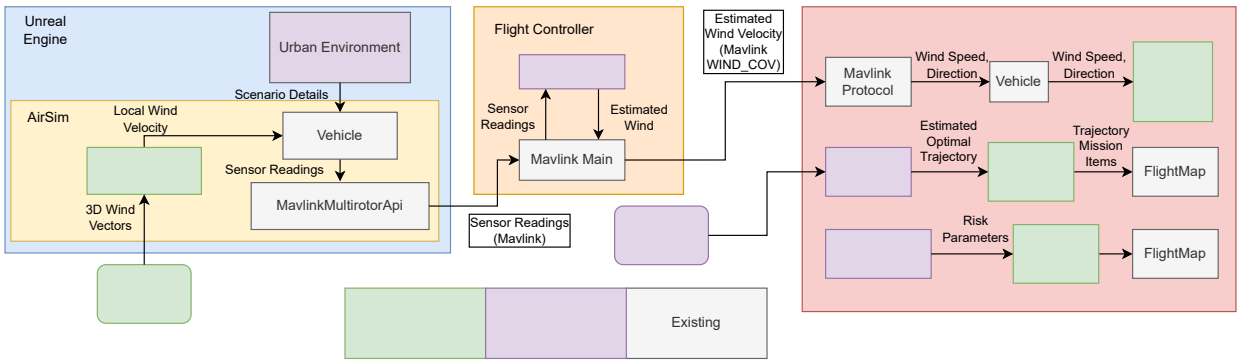

Figure 1. Architecture and Integration of Wind-Aware simulator, control and UI.

ground control station. Popular ground control stations include Mission Planner*, APM Planner 2.0†, MAVProxy‡, QGroundControl § and a few more. Although multiple open-source GCS are available, none of them sufficiently comply with wind awareness levels required for urban operations. At the same time, none of the simulators have spatial temporal wind simulation capability.

A. Contributions

In our work, we aim to address the rising operational and navigational challenges due to turbulent wind and to investigate design concepts that include wind-awareness and decision making information in the UI that may reduce the remote pilot's cognitive load, support the pilot in effective decision making, and improve safe operations in low-altitude urban airspace. A popular approach of integrating humans into the decision loop is to utilize the Software-in-the-Loop (SITL) capability of the open source autopilot software PX4 [20] or ArduPilot¶ and connect a joystick or remote radio telemetry with a suitable ground control station for pilots. However, extensive work is required to design a suitable environment that closely resembles an urban structure and influences the quadcopter operation as it would in reality. We have selected the AirSim platform due to its capability to conduct SITL with PX4 and integrate with QGroundControl. We aim to utilize available open source code-bases, extend their features, and design effective-explicit missions to study cooperative decision making abilities. A generic block diagram in Figure 1 shows the simulation environment and communication between the simulator, the autopilot and the ground control station.

The main contributions of this paper are:

- We identify information components and derive specific display designs based on existing literature and a focus group study with Subject Matter Experts. The information components recognized during the discussion are considered for the primary design of wind-aware UI.
- We implement the design and overlay wind information into the QGroundcontrol UI to accommodate a wind-aware framework throughout sUAS mission. This includes a real time wind velocity display as well as wind aware trajectory management features, such as buffered trajectory volumes and optimal path selection option.
- We construct an environment that resembles an urban cluttered airspace. We utilize existing 3D environment platform providers CADMapper ‖ and OpenStreetMap** as well as open source graphic toolset Blender†† to generate scenarios around the Oklahoma State University (OSU) campus.
- We integrate spatio-temporal atmospheric boundary layer wind profiles in real-time simulations in AirSim. To create an efficient and accurate model of the atmospheric boundary layer, we load a wind data set from a LES into the simulator with minimized impacts on real-time simulations.

*https://ardupilot.org/planner/docs/mission-planner-overview.html
†https://ardupilot.org/planner2/index.html
‡https://ardupilot.org/mavproxy/index.html
§https://docs.qgroundcontrol.com/master/en/index.html
¶https://ardupilot.org/ardupilot/
‖https://cadmapper.com
**https://www.openstreetmap.org
††https://www.blender.org/

II. Small Group Study on Current UI Design and Expectation

Within the multi-disciplinary research group at OSU, we conducted a discussion with Subject Matter Experts with FAR 107 and/or crewed aviation experience to assess current system capability specifically for operations in windy environments. The objective was to understand the current standards in the UI of traditional off-the-shelf sUAS and tendency towards operating in high wind outdoor operations. In total, seven persons joined the discussion: three were from the Department of Mechanical and Aerospace Engineering and four were from the School of Educational Foundations, Leadership and Aviation.The SMEs from the School of Educational Foundations, Leadership and Aviation are pilots and three of them have FAR 107 certifications. The focus group discussion was conducted as a semi-structured interview. The discussion centered around several open-ended questions related to the topic of interest as summarized below:

- What features do you like about current off-the shelf or open source mission planners or user interfaces?
- What additional information would you like to have for improved piloting and UI experience?
- Are you familiar or do you have experience with any current UI featuring wind information?
- How would you like to see the wind information?
- Do you feel an adjusted trajectory to reject wind disturbance in the UI would help you?
- How willing are you to stick to an adjusted trajectory computed by onboard Detect and Avoid?

We have noticed pilots' disinclination towards operating sUAS in cluttered, low-altitude environments in windy weather conditions with minimal inclusion of wind information in current UIs. We have received a positive response regarding inclusion of wind-corrected trajectory bands, wind velocity and direction overlay and hazard analytic metric of current operation. The general feedback from the SMEs highlighted the expected features of UI:

- Trusted autonomous features: Pilots rely on better autopilot features than manual control and are comfortable relying on self-correcting features of the sUAS in less windy weather.
- Enhanced visual sensory feed: sUAS pilots are generally used to observing from vantage points not easily accessible to human travel – thus the orientation of the camera is more important than the orientation of the air vehicle. The air vehicle's ability to keep station, and stability are important and positively contribute to the success (or detriment) of camera output video/still photography quality.
- Wind velocity tolerance limit: With the disinclination to operate during windy environment pilot prefer steady state wind within the limit of $10 - 15$ mph. Gusts are mentioned as a significant operational challenge and a 20 mph wind is/was incompatible with most sUAS for operational intelligence value.
- Separation between structural object and sUAS: One of the pilots mentioned a minimum 10 inch separation steady-state winds and 20 inch in gusty conditions. Being apprised of obstruction closure distance and closure rates would also be helpful to maximize utility while minimizing the risk of collision.
- Wind-Aware trajectory and navigation features: Prediction of turbulence caused by winds being disturbed from buildings, trees, the surface, etc. and subsequent waypoints navigation correction feature would be very helpful for a safe and successful outcome of the flight. A trajectory projected incrementally into the future are also desired.

Reviewing the discussion and existing literature, a task-workload segmented design consideration is provided in Table 1. Our preliminary interface design is mainly focused on enhancing wind situational awareness. We aim to identify related wind supervisory information in different tasks throughout the operations and provide related useful information to the pilots. Our approach is loosely aligned with the Hierarchical Task Analysis (HTA) [21] approach and connecting a task with associated wind aware information components.

III. The Wind-Aware Simulator

A. Architecture

The wind-aware simulator is comprised of three major components providing dynamic simulation, flight control, and user interfacing. Each component is based on open source software and requires extensions for the simulator. A human pilot will operate the UAS using remote control though QGC. Currently we are using a PlayStation 3 controller. Future experiments will include radio-transmitter based controller.

AirSim handles the dynamic simulation of the aircraft in its environment. It performs all simulation-related computations: it imports spatio-temporal wind data, computes local wind velocities, and simulates drag on the aircraft body. PX4 SITL is used as the flight controller for the simulator, although an alternative (such as ArduPilot or a custom design) would also suffice. PX4 communicates state observations to QGroundControl for operator interpretation. It partially supports wind estimation and other estimator implementations are possible [22–26].

Table 1. Interface design considerations

Task	Work	Interface design features
Monitor system	• Scan information display • Ensure system operational status	• Health monitor indicator • Weather impact on system indicator • Audio warning during close proximity of static obstacles
Visualize data	• Identify preferred method of wind display • Interpret sensor information • Analyze impact on operation	• Overlay wind velocity and direction • Gust prediction • Mapping display of urban clutter • Quantitative distance between nearby static obstacles
Manage mission	• Choose buffer for trajectory wrapping • Select preferred waypoint navigation	• Wind corrected trajectory bands • Wind aware controller initiation

Current implementation uses the absolute local velocity transmitted from AirSim through PX4 to QGroundControl. Future attempts will use an onboard estimated wind. We take advantage of MAVLink communication protocol which is compatible with AirSim and the PX4 IRIS SITL controller. MAVLink uses predefined messages to pass information between the simulator, flight controller, and ground station. To transmit wind values, our simulator uses the existing *WIND_COV* MAVLink message, which defines wind speeds in NED and allows for covariance. QGroundControl receives these messages and defines parameters for each vehicle (enabling multiple simultaneous vehicles on one ground station) that allow any component of the interface to access a vehicle's local wind. This inter-process interaction can be seen in Figure 1.

Execution of the simulator requires running all three custom components separately and has been tested on Ubuntu 18.04 and 20.04. As of now, execution of simulation requires an individual to download and build each software component; prepackaged components (or merging into official releases) will be possible in the future. The simulation is carried out in a computer with Intel-i7 core, 32 GB of RAM and Gigabyte GTX 1050 Ti graphics card.

B. Implementation

Traditionally atmospheric gusts and turbulence in aviation are modeled with widely used Von Karman model [27] and Dryden wind model [28]. These models are computationally efficient but they do not account for spectral energy tensor. They are frozen in time and space and cannot capture the highly coherent eddy structure of turbulent wind. In this work, we integrate one hour of wind data [29] generated from a high-fidelity LES that is capable of accurately capturing the near surface effect of the wind flow. To test and perform experiments with a wind-aware UI, the flight simulator must support wind acting in three dimensions and changing with respect to space and time.

1. AirSim Extensions

The current wind implementation in AirSim fails to support spatially and temporally changing wind: the simulator treats wind as a constant global velocity changed only at startup or through AirSim's API. During each loop of AirSim's physics engine, the drag force due to wind is calculated on every vehicle using the global value.

Our proposed solution to the spatial-temporal wind problem is the following:

- Rather than retrieving the constant-valued global wind in each physics loop, compute a local wind velocity using the vehicle's position in the wind grid.
- Generate local wind velocity vectors through interpolation of surrounding wind velocities in the predefined 3D vector field.
- Return the local wind velocity and compute the drag force due to wind using the existing physics implementation.

Our solution solves multiple problems:

- Wind velocity can vary almost continuously with respect to space.
- Wind velocity can vary with respect to time by loading new 3D vector fields during runtime.
- Multiple drones can experience different winds according to their position, providing support for multi-agent

simulations.

- Complex wind fields can be generated using existing computational fluid dynamics (CFD) software in a predefined format.

The proposed solution was completed by including an additional software object, called *WindModule*, to AirSim's core physics engine. Using information specified in AirSim's existing configuration settings, the WindModule parses wind data from a provided path at a given rate. This is handled in a parallel thread to minimize performance impact on the simulator. The data is stored in vector objects, allowing $O(1)$ access time. Using limited header information in each data file, the eight wind velocities surrounding a given location can be accessed directly with minimal computational overhead. This technique was originally referenced from [17] and used with minor modifications. Furthermore, the *WindModule* supports a theoretically unlimited amount of wind data files, allowing for arbitrarily long simulations. It also provides backwards compatibility for a static global wind and API support for setting a global wind and retrieving a vehicle's local wind. The extended version of AirSim can be accessed from here[‡‡].

2. PX4 Extensions

The PX4 flight stack's SITL simulator was modified to transfer local wind velocities from AirSim to QGroundControl. This would not be required if using a wind estimator within the flight controller, but allows for a less complex development process. Modifications involved the inclusion of a new wind estimate message type for uORB (PX4's internal inter-process messaging system) and support for publishing WIND_COV messages to QGroundControl.

3. QGroundControl Extensions

QGroundControl is built using Qt, a multi-platform user interface development tool, and is easily extensible using C++, the Qt Modeling Language (QML), and JavaScript. We use QML to develop *wind widgets* that display the local wind velocity in a variety of ways. An example of *wind widgets* is provided in Figure 2. All widgets rely on information made available to any component within QGroundControl, making their development and extension relatively straightforward. Some modification was required to QGC's internal handling of the WIND_COV data, as the original implementation ignored any vertical component. A *WindAwareMissionPlanner* object was developed to facilitate the generation of trajectory recommendations and risk buffers. The WindAwareMissionPlanner communicates with the QGC interface through Qt's signals and slots mechanisms, allowing event information (such as user interaction) to influence the behavior of the software. The wind-aware QGC is available in this reporsitory[§§].

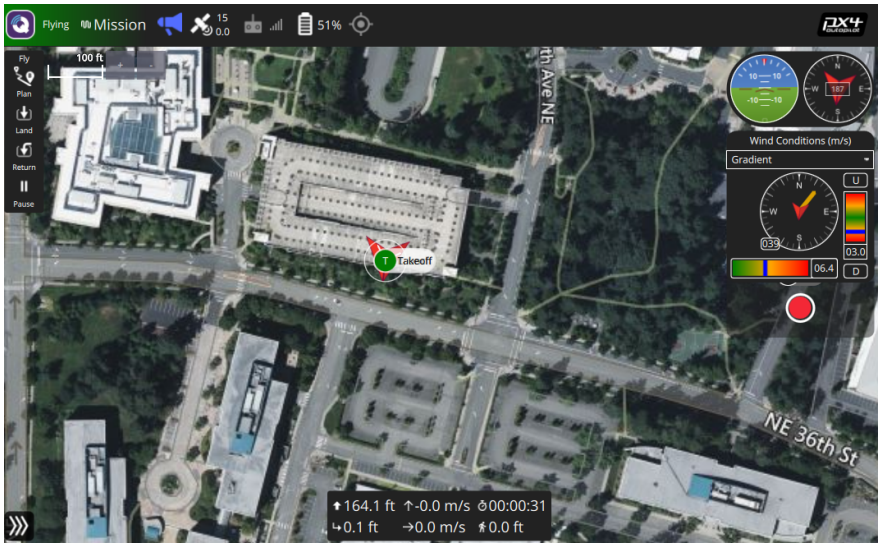

Figure 2. Qgroundcontrol UI with wind widget. Multiple display designs are available are available to the pilot through a drop down menu.

[‡‡]https://github.com/CoRAL-OSU/AirSim_NRI
[§§]https://github.com/CoRAL-OSU/wind_aware_QGC

C. Simulation Configuration

The *WindModule* has several parameters that can be configured before a simulation starts. They are defined in AirSim's *settings.json*, which allows for configuration of the plugin as a whole. We used a txt file format similar to [17] where the grid resolution and wind velocity are provided in meters and meters per second (m/s), respectively. All the wind data are saved in ascending order with time starting from 1 in a folder. The folder including wind data files is set using the *WindDataPath* JSON key. The logging mechanism can be enabled with the *EnableLog* key. The time period (in seconds) before a new data file is loaded can be configured with the *UpdatePeriod* key. The update period defines how long the *WindModule* waits before moving to the next wind data set. Larger data sets require longer to load, meaning that the update period should be sufficiently long. Finally, the default wind value can be set using the *DefaultWind* key. This defines the North-East-Down (XYZ) wind velocity under two conditions: 1) the aircraft has exited the region of provided data or 2) a global, unvarying wind velocity is preferred. The *WindModule* will utilize the default wind velocity if it cannot find the desired data, or if none is provided. An example configuration is shown in Figure 3. The example shows that the logger has been disabled, new data files should be loaded every five seconds, and the default wind value should be three meters per second north, two meters per second east, and one meter per second downwards.

```
{
    "Wind": {
        "WindDataPath": "/home/user/data_folder/",
        "EnableLog": false,
        "UpdatePeriod": 5,
        "DefaultWind": {
            "X": 3,
            "Y": 2,
            "Z": 1
        }
    }
}
```

Figure 3. Example simulation JSON settings

The Unreal Engine is capable of rendering highly detailed and complex environments. However, it takes skill and familiarity with many graphics-production tools to develop high quality environments. Premade environments may be used, but they can come at a cost. In some cases, mapping data can be used to generate levels based on real-world locations. Examples include CADMapper and OpenStreetMap. Both of these tools support direct exporting of spatial data into a number of file types. The scenario data can be imported into Unreal using Unreal's Datasmith plugin [¶¶]. This data is not exact, however, and may require cleanup in additional modeling software such as Blender or Autodesk 3ds Max. OpenStreetMap data may require additional processing using external tools or plugins [***], [†††]. Blender and 3ds Max can also be used to export into Unreal's datasmith file type [‡‡‡]. In our experience these spatial mapping tools provide sufficient building geometry but inadequate road geometry and texturing. An example simulation scenario of the OSU campus generated through exported spatial data is illustrated in Figure 4.

Rendering the scenario is handled entirely by Unreal Engine, but requires proper setup of the environment beforehand. High quality objects will require several components such as materials (sets appearance and texture of the 3D object), UV maps (maps 2D materials onto 3D objects), and collision meshes (for physics simulation). Spatial data tools may not provide the necessary textures, UV maps, and collision meshes to properly use the scene as a simulation environment. Additionally, building heights may be randomly generated as the mapping software lacks sufficient data. These problems can be corrected in an alternate software (such as Blender) or within Unreal's own tools. Furthermore, scenario fidelity can be improved by using higher quality surface materials and environmental objects such as grass, trees, benches, light posts, and other urban features. Many of these resources can be easily obtained using Quixel Megascans [§§§], a

[¶¶] https://www.unrealengine.com/en-US/datasmith
[***] https://github.com/vvoovv/blender-osm
[†††] http://osm2world.org/
[‡‡‡] https://github.com/0xafbf/blender-datasmith-export
[§§§] https://quixel.com

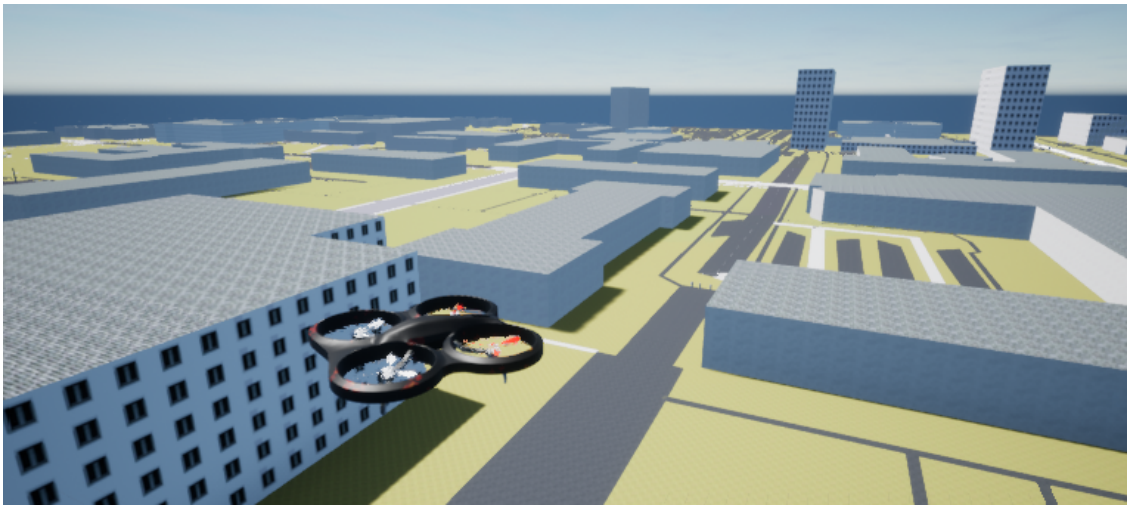

Figure 4. Quadcopter Simulation in rendered OSU campus scenario.

photogrammetry-based repository of assets freely available for use in Unreal Engine.

IV. Wind-Aware Piloting Interfaces

Wind-related information can be used in a number of manners by a pilot. Our interface attempts to assist with in-flight decision making by relaying local wind estimations as well as pre-flight planning through trajectory recommendations.

A. Local Wind Display

Upon receiving the 3D wind velocity estimate from MAVLink, QGC makes the data available on a vehicle-by-vehicle basis. A custom instrument widget within the QGC's FlightMap displays the data. Several interface layouts were designed to convey the wind velocity to the pilot.

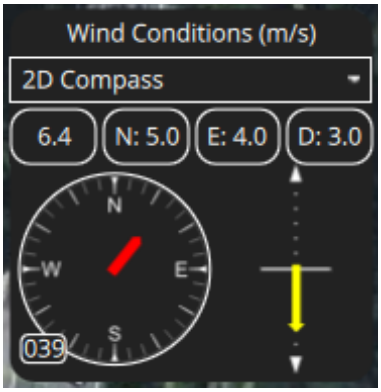

Figure 5. Compass-based heading with vertical component

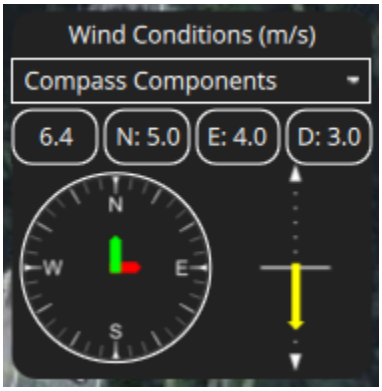

Figure 6. Component-wise display with vertical component

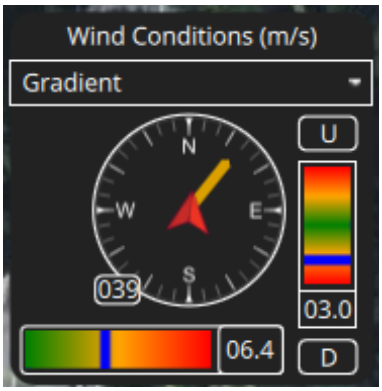

Figure 7. Compass-based display with gradients

The displays were all designed to be easily understood by pilots and to require minimal concentration to understand. Each interface conveys the velocity both numerically and graphically with the graphical implementation varying. Figure 5 uses a compass-style display, where the planar velocity of the wind (north and east velocity components) is represented as an arrow conveying the wind's direction and magnitude. The arrow is scaled in length according to the wind magnitude and it points in the direction the wind is traveling. The vertical component of the wind is displayed on an accompanying vertical axis. It features a similar magnitude-scaled arrow that will point up or down according to the direction of the wind's vertical component. An alternative display is shown in Figure 6 which has the planar

velocity split into north and east vector components. This is intended to allow a pilot to better visualize how the wind is affecting their aircraft. It features the same vertical display as in Figure 5. A third implementation is shown in Figure 7, which attempts to better communicate the magnitude of the wind velocity. The velocity's direction is conveyed using a compass, similar to Figure 5, but the arrow's length is never changed. Instead, the velocity's magnitude is displayed on a green-yellow-red gradient beneath the compass. A colored bar is placed along the gradient according to the strength of the planar wind and the numerical magnitude is displayed alongside it. This allows a pilot to easily determine the relative magnitude of the velocity as well as its exact value. The vertical display gets a similar treatment and features a red-yellow-green-yellow-red gradient; the position of the bar then conveys both the relative magnitude and direction of the wind's vertical component. Its magnitude is also displayed alongside the graphical representation; additionally, the color of the heading arrow matches the color of the planar gradient. The gradient display also includes a red triangle signalling the aircraft's heading, to allow the pilot a quick comparison between wind direction and heading.

These widgets were built using the QML and their logic was implemented using JavaScript. Variations of the display, using only the wind's velocity, are easily implemented and added to the list of display options. Within the interface, each widget is able to be selected by the operator so they can choose which display they prefer and compare them.

B. Trajectory Management and Supervisory Recommendation

Based on the focus group study, we include a buffer volume feature in the waypoint trajectory generation. The implementation is based on the idea of geofencing and intend to provide a pilot an wrapped trajectory guidance while flying manually. The buffer values can be predefined or calculated based on the wind prediction/estimation and the capability of a controller to minimize deviation. In a cluttered environment, the buffer is expected to provide a safety cushion for manual operations near any static object and provide a visual deviation risk to the pilot. The trajectory will be wrapped around two buffers namely inner and outer buffer. The inner buffer indicates a safe maximum deviation that may be tolerated during operations and the outer buffer indicates risk zone where necessary steps need to be taken by the pilot to bring sUAS within the safe margin zone. Figure 8-9 provide two examples of buffered trajectory in the QGC. In addition to the pre-calculation of the buffer, the pilot will be able to define their custom buffer values required for particular operation.

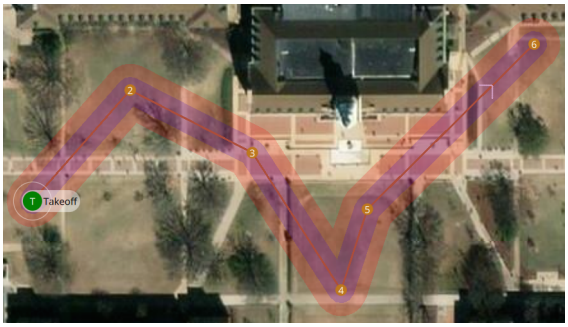

Figure 8. Wind buffers of with 5 meter radius inner buffer and 10 meter radius outer buffer.

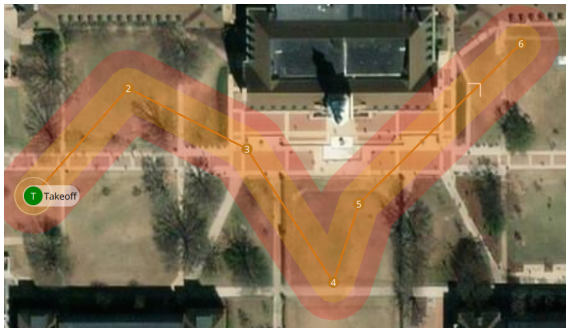

Figure 9. Altered risk buffer with different colors and larger buffer radii.

The interface also features a basic trajectory recommendation system. An algorithm to generate wind-optimal trajectories is outside the scope of this paper and discussed in future work. A placeholder algorithm is implemented instead, which simply removes intermediate waypoints from the trajectory. This forms the basis of a system that should be able to prompt a user when the pre-planned trajectory is deemed suboptimal and allow an easy review of a replacement trajectory. An example trajectory is shown in Figure 10, in which a pilot has manually placed several waypoints using QGC's existing mission planning software. In Figure 11, the recommendation system provides a preview of an alternative trajectory using our placeholder algorithm. The new trajectory is accepted and replaces the original trajectory in Figure 12. It prompts the user to display a preview of the recommended new trajectory. The user then has the option of accepting or declining the recommendation; accepting causes the system to replace the current mission plan and update the flight display.

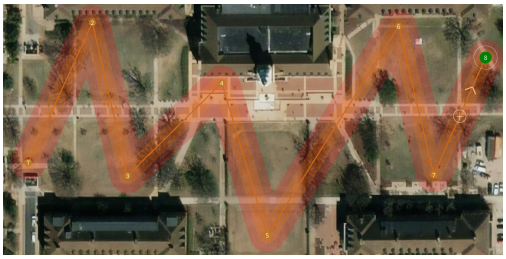

Figure 10. Example of pilot-generated trajectory.

Figure 11. Recommendation system's preview of alternative trajectory in Blue.

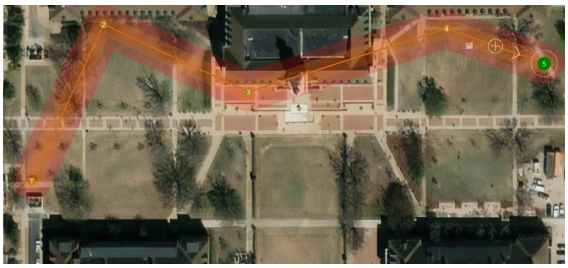

Figure 12. Resulting trajectory after pilot accepts recommendation.

V. Discussion and Future Work

In this work, we develop a pipeline for wind-aware simulator for sUAS in for pilot in the loop simulation in urban environments based on AirSim, PX4 and QGroundcontrol station. Wind velocity widget and trajectory management features have been added to the QGroundcontrol station and spatio-temporal LES wind data has been integrated into the AirSim. A few limitations of the current implementation include the use of a placeholder trajectory generation algorithm, the use of a specific wind grid velocity format and visual errors in buffer discoloration during trajectory overlap. Existing optimal trajectory generation algorithm may be integrated or novel wind-aware trajectory generation algorithm may be developed and integrated. Also, using data conversion method any standard wind file can be converted into the specified text format. Polygon visual problems may be refined upon further debugging. Additionally, operational risk margin information will be added into the user interface. This option is expected to provide pilots with operational risks associated with current mission. Our future work also includes appropriate mission and experiment designs to test the efficiency of the developed wind-aware simulation pipeline as well as gather pilot feedback on the design.

Acknowledgments

The work is supported in part by the National Science Foundation (NSF) under award number 1925147.

References

[1] Terwilliger, B. A., Ison, D. C., Vincenzi, D. A., and Liu, D., "Advancement and application of unmanned aerial system human-machine-interface (HMI) technology," *International Conference on Human Interface and the Management of Information*, Springer, 2014, pp. 273–283.

[2] Reiche, C., Cohen, A. P., and Fernando, C., "An Initial Assessment of the Potential Weather Barriers of Urban Air Mobility," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[3] Williams, K. W., "An assessment of pilot control interfaces for unmanned aircraft," Tech. rep., FEDERAL AVIATION ADMINISTRATION OKLAHOMA CITY OK CIVIL AEROMEDICAL INST, 2007.

[4] Maybury, M. T., "Usable advanced visual interfaces in aviation," *Proceedings of the International Working Conference on Advanced Visual Interfaces*, 2012, pp. 2–3.

[5] Vinot, J.-L., Letondal, C., Pauchet, S., and Chatty, S., "Could tangibility improve the safety of touch-based Interaction? Exploring

a new Physical Design Space for Pilot-System Interfaces," *Proceedings of the International Conference on Human-Computer Interaction in Aerospace*, 2016, pp. 1–8.

[6] Monk, K., Shively, R. J., Fern, L., and Rorie, R. C., "Effects of display location and information level on UAS pilot assessments of a detect and avoid system," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 59, SAGE Publications Sage CA: Los Angeles, CA, 2015, pp. 50–54.

[7] Friedrich, M., and Vollrath, M., "Human Machine Interface Design for Monitoring Safety Risks Associated with Operating Small Unmanned Aircraft Systems in Urban Areas," *Aerospace*, Vol. 8, No. 3, 2021, p. 71.

[8] Zhang, W., Feltner, D., Shirley, J., Swangnetr, M., and Kaber, D., "Unmanned aerial vehicle control interface design and cognitive workload: A constrained review and research framework," *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2016, pp. 001821–001826.

[9] Edmondson, J., Cahill, G., and Rowe, A., "On developing user interfaces for piloting unmanned systems," *Proceedings of the International Workshop on Robotic Sensor Networks*, 2014.

[10] Zhang, W., Feltner, D., Shirley, J., Kaber, D., and Neubert, M. S., "Enhancement and Application of a UAV Control Interface Evaluation Technique: Modified GEDIS-UAV," *ACM Transactions on Human-Robot Interaction (THRI)*, Vol. 9, No. 2, 2020, pp. 1–20.

[11] Jimenez, C., Faerevaag, C. L., and Jentsch, F., "User Interface Design Recommendations for Small Unmanned Aircraft Systems (sUAS)," *International Journal of Aviation, Aeronautics, and Aerospace*, Vol. 3, No. 2, 2016, p. 5.

[12] Haritos, T., "A Study of Human-Machine Interface (HMI) Learnability for Unmanned Aircraft Systems Command and Control," 2017.

[13] Perez-Rodriguez, D., Maza, I., Caballero, F., Scarlatti, D., Casado, E., and Ollero, A., "A ground control station for a multi-uav surveillance system: design and validation in field experiments," *J. Intell. Robot. Syst*, Vol. 69, No. 1-4, 2013, pp. 119–130.

[14] Vincenzi, D. A., Terwilliger, B. A., and Ison, D. C., "Unmanned aerial system (UAS) human-machine interfaces: new paradigms in command and control," *Procedia Manufacturing*, Vol. 3, 2015, pp. 920–927.

[15] RTCA Special Committee 228, "Minimum Operational Performance Standards (MOPS) for Detect and Avoid (DAA) Systems, 2017 Edition," , 2017.

[16] Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., and Von Stryk, O., "Comprehensive simulation of quadrotor uavs using ros and gazebo," *International conference on simulation, modeling, and programming for autonomous robots*, Springer, 2012, pp. 400–411.

[17] Furrer, F., Burri, M., Achtelik, M., and Siegwart, R., "RotorS—A modular Gazebo MAV simulator framework," *Robot operating system (ROS)*, Springer, 2016, pp. 595–625.

[18] Shah, S., Dey, D., Lovett, C., and Kapoor, A., "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and service robotics*, Springer, 2018, pp. 621–635.

[19] Song, Y., Naji, S., Kaufmann, E., Loquercio, A., and Scaramuzza, D., "Flightmare: A Flexible Quadrotor Simulator," *arXiv preprint arXiv:2009.00563*, 2020.

[20] Meier, L., Honegger, D., and Pollefeys, M., "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 6235–6240.

[21] Annett, J., "Hierarchical task analysis," *Handbook of cognitive task design*, Vol. 2, 2003, pp. 17–35.

[22] González-Rocha, J., Woolsey, C. A., Sultan, C., and De Wekker, S. F., "Sensing wind from quadrotor motion," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 4, 2019, pp. 836–852.

[23] Allison, S., Bai, H., and Jayaraman, B., "Wind estimation using quadcopter motion: A machine learning approach," *Aerospace Science and Technology*, Vol. 98, 2020, p. 105699.

[24] Pappu, V. S., Liu, Y., Horn, J. F., and Cooper, J., "Wind gust estimation on a small VTOL UAV," *Proceedings of the 7th AHS Technical Meeting on VTOL Unmanned Aircraft Systems and Autonomy, Mesa, AZ, USA*, 2017, pp. 24–26.

[25] Crowe, D., Pamula, R., Cheung, H. Y., and De Wekker, S. F., "Two Supervised Machine Learning Approaches for Wind Velocity Estimation Using Multi-Rotor Copter Attitude Measurements," *Sensors*, Vol. 20, No. 19, 2020, p. 5638.

[26] Chen, H., Bai, H., and Taylor, C. N., "Invariant-EKF Design for Quadcopter Wind Estimation," *American Control Conference*, 2022. URL https://coral-osu.github.io/assets/pdf/ACC22.pdf.

[27] Karman, T. V., "The fundamentals of the statistical theory of turbulence," *Journal of the Aeronautical Sciences*, Vol. 4, No. 4, 1937, pp. 131–138.

[28] Hoblit, F. M., *Gust loads on aircraft: concepts and applications*, Aiaa, 1988.

[29] Vuppala, R. K., and Kara, K., "A Novel Approach in Realistic Wind Data Generation for The Safe Operation of Small Unmanned Aerial Systems in Urban Environment," *AIAA AVIATION 2021 FORUM*, 2021, p. 2505.