An Assessment of Unmanned Aircraft System Operations with the Extensible Trajectory Optimization Library

Olatunde Sanni*

Georgia Institute of Technology, Atlanta, Georgia, 30332, USA

Thanakorn Khamvilai†

King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

Teppatat Puntawuttiwong‡

National Electronics and Computer Technology Center, Pathumthani, 12120, Thailand

Eric Feron§

King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

In the not too distant future, Unmanned Aircraft Systems (UAS) will be a source of economic power in urban environments. These systems will be used for package delivery, building inspections, filming, and many other tasks. However, economic benefits should not outweigh public safety. These systems must not collide into buildings, and they must maintain a safe separation distance from neighboring systems. In other words, UAS operations should be assessed for safety. This paper assesses how safety requirements in the urban environment influence UAS operations. It presents a novel way of performing this assessment by using the Extensible Trajectory Optimization Library (ETOL) to continuously solve a vehicle guidance problem (VGP) in a multi-agent robot simulator. This paper introduces the common structure of a VGP, along with a VGP formulation for a UAS in an urban environment. In addition, a platform for safety assessments is presented, along with recommendations for improving the safety of UAS operations.

I. Nomenclature

d	=	displacement vector
d_{sm}	=	distance safety margin
$\mathcal{D}_t$	=	state space at time t
h	=	height
$\mathcal{M}$	=	large positive constant
n_{steps}	=	number of time steps
n_u	=	number of control variables
n_{u_r}	=	control index offset
n_x	=	number of state variables
n_{x_r}	=	state index offset
t	=	time
t_f	=	final time
T	=	time horizon
$\mathbf{u}$	=	control vector
U	=	control space
$\mathcal{U}$	=	set of admissible control functions
$\mathbf{x}$	=	state vector

*Graduate Research Assistant, Daniel Guggenheim School of Aerospace Engineering, 270 Ferst Drive, and AIAA Student Member

†Visiting Scholar, Electrical and Computer Engineering, Thuwal 23955, and AIAA Student Member

‡Research Assistant, Data-Driven Simulation and Systems Research Team, 112 Phahonyothin Road

§Professor, Electrical and Computer Engineering, Thuwal 23955, and AIAA Associate Fellow

$\mathbf{x}^{(0)}$	=	start state
$\mathbf{x}^{(t_f)}$	=	goal state
x_i	=	i-th element in $\mathbf{x}$
x_{tol}	=	error tolerance
Δt	=	time step
$\mathcal{J}$	=	objective function
$\mathbb{R}$	=	real number field
ζ	=	binary variable

II. Introduction

IN recent years, corporations and governments have greatly invested financial and human capital towards unmanned aircraft system (UAS) operations in urban environments for economic expansion. However, the fruition of these operations is stymied by public safety concerns. The debate to stifle the economic reward of UAS operations for public safety is akin to the debate to close in-person stores because of concerns about coronavirus disease (COVID-19) spread. In the COVID-19 debate, simulations of COVID-19 spread are vital for assessing the reward of economic activity versus public health. Considering the role simulations play in public persuasion, simulations should play a key role in assessing UAS operations in urban environments.

UAS operations in an urban environment is a complex challenge that requires an accurate and fast mapping of high-level objectives to low-level controls. This challenge is exacerbated by the unpredictability of many independent actors, such as wind, noises, temperature, and other vehicles. Notwithstanding, researchers in academia, industry and government have made great strides towards integrating UAS into urban environments. Continuous advancements in computational capabilities, and path-planning or trajectory generating algorithms have led to feats such as NASA's Unmanned Traffic Management (UTM) TCL 4 [1] and an Airbus UTM Prototype [2]. As these advancements continue, researchers need to assess how UAS operations can improve with the latest technology. Consequently, the C++ library called *Extensible Trajectory Optimization Library* (ETOL) [3] was created to provide an abstraction layer between the vehicle guidance problem (VGP) and the technology that solves this problem.

ETOL exploits the truism that every VGP can be expressed as an optimal control problem, which has an objective function, a finite number of state and control variables, equations of motion, collision avoidance constraints for static and moving objects, and other operational constraints such as wind or weather. Although the technology that solves the VGP may change, the VGP does not necessarily change. This demarcation between a problem and solver has been utilized by the Artificial Intelligence (AI) community. For example, the planning domain definition language (PDDL) and its variants decouple the problem description from the various path-planning technologies [4]. This demarcation makes it easier to assess UAS operations without worrying about the intricacies of a solver. Consequently, researchers have the flexibility to focus their efforts on either problem formulations or the algorithms for solving them. In this paper, the former is the focus.

A motivation for demarcating a problem from a solver is modularity. With this feature, a specific VGP can be quickly passed to a swath of path planners or trajectory optimizers. The minutiae of many of these trajectory solvers are found in literature, and they are classified as either sampling-based, node-based, mathematical model-based, learning-based, or hybrid [5, 6]. Although two solvers may have the same classification, their computational implementation may differ. For example, a solver's implementation of a node-based algorithm could have a faster sort function. In other words, a VGP can be solved with a plethora of tools, which in effect means the VGP is modular.

This first contribution, in this paper, is a VGP for a UAS in an urban environment. This VGP can be solved with six different VGP solvers – *ðymos* [7], *PSOPT* [8], *SCIP* [9, 10], *GLPK* [11], and *Gurobi* [12] – embedded into ETOL. A ETOL interface to a VGP solver is called an extensible solver (eSolver). A eSolver primarily converts a VGP into a compatible format for a solver. This paper discusses the conversion process, and the source code for this conversion is made publicly available in ETOL [3]. Another contribution of this paper is its development of an urban environment, which is the Georgia Institute of Technology campus and its surroundings in the City of Atlanta. A visualization of this urban environment is constructed from United States Geological Survey (USGS) aerial imagery, USGS elevation data, and Microsoft building footprints data [13]. In addition to the visualization of these buildings, which are obstacles, the effect of wind fields on the vehicle resulting from the presence of these buildings is analyzed.

The rest of this paper is organized as follows. Section III defines the common structure of a VGP. Section IV introduces the urban environment for the UAS assessment. Section V outlines the process for obtaining wind data. Section VI assesses the results from simulating UAS operations. Section VII provides concluding remarks.

III. Vehicle Guidance Problem

A vehicle guidance problem (VGP) entails finding a control function $\mathbf{u}^*(\cdot)$ over time $t \in [0, T]$ that steers a vehicle from an initial state $\mathbf{x}^{(0)} \in \mathbb{R}^{n_x}$ to a goal state $\mathbf{x}^{(t_f)} \in \mathbb{R}^{n_x}$, while satisfying all constraints. The VGP can be expressed as an optimal control problem because if there is no objective to minimize, the objective function $\mathcal{J} : U \times [0, T] \mapsto \mathbb{R}$ can be an arbitrary constant. In this situation, the problem is a constraint satisfaction problem. Since a constraint satisfaction problem can be formulated as an optimal control problem, Eq. (1) is the general framework that is used to define a VGP.

$$\begin{aligned}
 \mathbf{u}^*(\cdot) = \arg \min_{\mathbf{u} \in \mathcal{U}} & \left[\min_{t_f \in [0, T]} \mathcal{J}(\mathbf{u}, t_f) \right] \\
 \text{s.t.} & \\
 & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \forall t \in [0, T] \\
 & \mathbf{x}(0) = \mathbf{x}^{(0)} \\
 & \|\mathbf{x}(t_f) - \mathbf{x}^{(t_f)}\| \leq x_{\text{tol}} \\
 & \mathbf{x}(t) \in \mathcal{D}_t \subset \mathbb{R}^{n_x}, \forall t \in [0, T] \\
 & \mathbf{u}(t) \in U \subset \mathbb{R}^{n_u}, \forall t \in [0, T]
 \end{aligned} \tag{1}$$

where x_{tol} is an error tolerance, $U \subset \mathbb{R}^{n_u}$ is the set of admissible control values, $\mathcal{U}$ is the set of admissible control functions, $\mathcal{D}_t$ is the state space at time t , and $f : \mathcal{D}_t \times U \times [0, T] \mapsto \mathbb{R}^{n_x}$ is a vector-valued function that yields the state vector's time derivative.

This formulation of the VGP is a free final time optimal control problem, where the initial time, initial state and final state are fixed. The last two constraints are restrictions on the allowable state and admissible control at any given point in time. These two constraints encapsulate all other constraints on the vehicle. In other words, these two constraints are defined by a set of inequality and equality constraints. Examples of these constraints include collision avoidance, minimum separation distance, upper and lower bounds on the variables.

For pedantic purposes, consider a two-dimensional VGP with static obstacles, which from now on will be called exclusion zones (EXZ), and moving obstacles, which from now will be called moving exclusion zones (MEXZ). Each EXZ is assumed to be a closed subset of the configuration space and can be specified by a set of convex polygons that enclose the space, as shown in Fig. 1a. Whereas, a MEXZ is a closed ball with radius r and a center that is defined by a piecewise continuous function of time. A MEXZ can be considered a minimum separation distance requirement between the vehicle and other objects, and Fig. 1b illustrates this minimum distance concept. With these obstacle definitions,

Fig. 1 Obstacle formulations

assume a vehicle must traverse the environment that is illustrated in Fig. 2. In this environment, there are two EXZs and two MEXZs. Fig. 2 does not show the EXZs, but it shows the path of the MEXZs, which are the star and diamond shapes. The path of the MEXZs are predefined and linearly interpolated from Tables 1 and 2. In addition, the radius of both MEXZs is 0.5. A shaded shape indicates the object's initial position, the unshaded shapes are the final positions.

Table 1 Diamond object's waypoints

Elapsed Time (s)	Position
0	(3.5,0.5)
9.0	(3.5,4.5)
16.0	(0.5,4.5)

Table 2 Star object's waypoints

Elapsed Time (s)	Position
0	(1.0,2.0)
10.0	(4.5,2.0)
16.0	(4.5,5.0)

The pentagon marks the initial position of the vehicle, and the 'X' symbol indicates the goal position. In addition, assume a single integrator dynamical model, which sets the state vector's time-derivative equal to control vector.

$$\dot{\mathbf{x}} = \mathbf{u} \quad (2)$$

This simple problem is solved by ETOL with two different algorithms: mixed-integer linear programming (MILP) and collocation-based optimal control.

Fig. 2 Vehicle guidance problem with collision avoidance requirements

A. Mixed-Integer Linear Programming Formulation

A mixed-integer linear programming (MILP) algorithm for path planning is classified as a mathematical programming method. It is also a direct transcription method because it enforces constraints at discrete points in time. When solving a VGP with a MILP solver, the VGP is first converted into a mixed logical dynamical (MLD) problem, which has linear dynamic equations and linear mixed-integer inequality constraints [14]. This formulation uses continuous variables, and it supports binary variables. For the single integrator problem illustrated in Fig. 2, a reasonable objective $\mathcal{J}$ is to minimize the magnitude of the control at each time instance,

$$\mathcal{J}(\mathbf{u}_1, \dots, \mathbf{u}_n) = \sum_{k=0}^n |u_{x_k}| + |u_{y_k}| \quad (3)$$

where $\mathbf{u}_i = \begin{bmatrix} u_{x_i} & u_{y_i} \end{bmatrix}^T \in U, \forall i \in [0, n]$.

The objective function, single integrator dynamics, EXZ, and MEXZ can be converted and combined to form a

MLD problem by using the formulations in [14–16]. From this conversion, the objective function becomes Eq. (4).

$$\begin{aligned}
\min_{z_1, \dots, z_n} \mathcal{J}(z_1, \dots, z_n) &= \sum_{k=0}^n z_{x_k} + z_{y_k} \\
\text{s.t. } \forall k \in [0, n] \\
z_{x_k} &\geq u_{x_k} \\
z_{x_k} &\geq -u_{x_k} \\
z_{y_k} &\geq u_{y_k} \\
z_{y_k} &\geq -u_{y_k}
\end{aligned} \tag{4}$$

The dynamic constraints can be approximated with Euler integration, which is sufficiently accurate if the step size is small.

$$\begin{aligned}
x_k &= x_{k-1} + \Delta t \cdot u_{x_{k-1}} \\
y_k &= y_{k-1} + \Delta t \cdot u_{y_{k-1}}
\end{aligned} \tag{5}$$

The l -th EXZ constraint can be expressed with the set of inequalities in Eq. (6)

$$\begin{aligned}
y_k - m_{l_i}^{(0)} x_k - \mathcal{M} \zeta_{l_i}^{(0)} &\leq a_{l_i,2}^{(0)} - m_{l_i}^{(0)} a_{l_i,1}^{(0)}, \forall i \in \{1, 2, \dots, N_l\} \\
y_k - m_{l_i}^{(1)} x_k + \mathcal{M} \zeta_{l_i}^{(1)} &\geq a_{l_i,2}^{(1)} - m_{l_i}^{(1)} a_{l_i,1}^{(1)}, \forall i \in \{1, 2, \dots, M_l\} \\
\sum_{i=1}^{N_l} \zeta_{l_i}^{(0)} + \sum_{j=1}^{M_l} \zeta_{l_j}^{(1)} &\leq N_l + M_l - 0.5
\end{aligned} \tag{6}$$

where $\mathcal{M}$ is a large positive number, $\langle a_{l_i,1}^{(0)}, a_{l_i,2}^{(0)} \rangle$ is the position of the i -th vertex for the l -th polygon's bottom segment, N_l is the number of vertices along the l -th polygon's bottom segment, $\langle a_{l_j,1}^{(1)}, a_{l_j,2}^{(1)} \rangle$ is the position of the j -th vertex for the l -th polygon's top segment, M_l is the number of vertices along the l -th polygon's top segment, and

$$m_{l_i}^{(k)} = \frac{a_{(l_{i+1},2)}^{(k)} - a_{l_i,2}^{(k)}}{a_{(l_{i+1},1)}^{(k)} - a_{l_i,1}^{(k)}}$$

Similar to EXZ, the q -th MEXZ constraint can be approximated with a symmetric P_q -sided polygon, such that the component of the displacement vector's projection onto a rotated unit vector is greater than MEXZ's radius.

$$\begin{aligned}
(\text{comp}_{\hat{q}_j} \mathbf{d}_q) - M \zeta_q &\leq r_q \\
\sum_{j=1}^{P_q} \zeta_{l_i} &\geq 0.5
\end{aligned} \tag{7}$$

where $\hat{q}_j = \left[\cos\left(\frac{2\pi j}{P_q}\right) \quad \sin\left(\frac{2\pi j}{P_q}\right) \right]$, $\zeta_q \in \{0, 1\}$, M is an arbitrary large number, and $j = 1, 2, \dots, P_q$. Fig. 3 provides an illustration of two conditions that yield different binary variable values.

The combination of Eq. (4) to (7) is a MLD problem that can be solved by MILP solvers, such as Gurobi, GLPK, and SCIP. The construction of this MLD problem is not a simple task. However, ETOL's MILP eSolvers handle this process and this allows a user to focus on the problem as opposed to the MLD construction for a specific MILP solver.

B. Collocation-Based Optimal Control Formulation

Similar to the MILP approach, collocation-based optimal control algorithms are classified as mathematical programming methods. As a mathematical programming method, an optimization problem is solved. However, collocation-based optimal control approaches construct optimization problems differently. They convert an optimal control problem into a nonlinear programming (NLP) problem. They accomplish this task by approximating the states and controls with polynomials that are functions of time; they enforce constraints at discrete points, known as collocation points, in time. With discrete points from the approximating polynomials for the state and control, the integral in the

Fig. 3 Separation distance test

objective function is derived with numerical integration. The objective function and constraints constitute the NLP problem, which is then solved with gradient-based optimizers [17].

A benefit of using collocation-based optimal control is the freedom to use nonlinear equations, as long they are twice differentiable C^2 . However, its reliance on gradient-based optimizers means it can converge to a local minimum or not converge if the first iterate, which from henceforth is called a seed, is outside its radius of convergence. In other words, the initial guess of the trajectory is critically important, especially when there are path constraints such as EXZs and MEXZs. Notwithstanding, ETOL can solve a VGP with collocation-based optimal control solvers, such as PSOPT and ∂ ymos.

Since the absolute value function that was used for the MILP formulation is not C^2 , a quadratic objective is used to compute the magnitude of the control vector. In other words, the square of the 2-norm is used as the integrand in the objective function.

$$\mathcal{J}(\mathbf{u}(\cdot)) = \int_0^{t_f} (u_x^2(t) + u_y^2(t)) dt \quad (8)$$

The dynamics of the VGP in Fig. 2 is a single integrator and the algorithm is configured to use this model without any alterations.

$$\begin{aligned} \dot{x}(t) &= u_x(t) \\ \dot{y}(t) &= u_y(t) \end{aligned} \quad (9)$$

The EXZ constraints, which are defined by convex polygons, are inherently non-smooth at its vertices. As a result, the constraint is enforced by adding elliptical path constraints along the outer edges of the EXZ, as shown in Fig. 4.

$$a_i^{(j)} b_i^{(j)} - \left(b_i^{(j)} (x_k - x_{c_i}^{(j)})^2 + a_i^{(j)} (y_k - y_{c_i}^{(j)})^2 \right) \leq 0, \quad \forall i \in \{1, 2, \dots, N_j\}, \quad \forall j \in \{1, 2, \dots, N_{\text{exz}}\} \quad (10)$$

where N_{exz} is the number of EXZs, N_j is number of ellipses for the j -th EXZ, $\langle x_{c_i}^{(j)}, y_{c_i}^{(j)} \rangle$ is the center position of the i -th ellipse for the j -th EXZ, $a_i^{(j)}$ is the length of the semi-major axis and it equals half the length of the edge, and $b_i^{(j)}$ is the length of the semi-minor axis, where $b_i^{(j)} < \beta a_i^{(j)}$, and β is an arbitrary factor such that $|\beta| < 1$. For the VGP in Fig. 2, β is set to 0.2, and the resulting ellipses along Fig. 1a is illustrated in Fig. 4.

The elliptical path constraints around the border of a EXZ effectively prevents the algorithm from placing a node within the EXZ because doing so would require an adjacent node to reside in an ellipse. This requirement assumes the distance between nodes is relatively small, which is possible by increasing the number of nodes and reducing the time step.

The MEXZ constraint is much simpler to implement because the 2-norm $\|\cdot\|_2$ is a smooth operator. As a result, the MEXZ that is shown in Fig. 1b is enforced by the path constraint, Eq. (11).

$$r^2 - \|\mathbf{d}\|_2^2 \leq 0 \quad (11)$$

where r is a MEXZ's radius and $\mathbf{d}$ is the displacement vector from the MEXZ's center.

C. ETOL Results

The computational results are shown in Fig. 5 to 8 and Table 3. Fig. 5 presents the time history of the Gurobi results, along with the location of the MEXZs, up to different points in time. These results confirm the solver complied

Fig. 4 Exclusion zone ellipses

with obstacle avoidance constraints. Table 3 shows the performance of multiple eSolvers. Although PSOPT results are included in the same table for Gurobi and GLPK, its objective function is different. Its collocation-based optimal control formulation provides a smooth trajectory. However, its computational time is significantly higher than MILP formulation, which may be unsuitable for a real-time system like a UAS.

Fig. 5 State trajectory from Gurobi with EXZs and MEXZs

(a) w/o moving obstacles

(b) w/ 2 moving obstacles

Fig. 6 GLPK

(a) w/o moving obstacles

(b) w/ 2 moving obstacles

Fig. 7 Gurobi

(a) w/o moving obstacles

(b) w/ 2 moving obstacles

Fig. 8 PSOFT (Gurobi seed)

Table 3 Score card

Solver	# Nodes	# Moving Obs.	Obj.	CPU Time (s)
Gurobi	161	0	99.8	< 1
Gurobi	161	2	99.8	< 1
GLPK	161	0	99.8	11
GLPK	161	2	99.8	12
PSOPT (user seed)	41	0	3.1944	40
PSOPT (user seed)	41	2	3.267698	42
PSOPT (Gurobi seed)	161	0	3.161275	24
PSOPT (Gurobi seed)	161	2	3.268818	16

IV. Case Study: Unmanned Aircraft Operations in an Urban Environment

A simulation of UAS operations within the Georgia Institute of Technology campus was performed. This simulation primarily considers the back-and-forth commute of a UAS from an origin to a destination. This UAS is called the mailman because its flight route can be used for a package delivery task. The simulation also includes eight additional

UAS, which continuously fly in a racetrack pattern. These eight UAS are on the same team, which means they can communicate with one another. In this case study’s simulation, this team mimics the task of monitoring the campus.

A key goal of this study is to visualize and assess UAS operations. To this end, Georgia Tech Research Institute’s multi-agent robotics simulator was used. This open-source simulator, which is called Simulating Collaborative Robots in a Massive Multi-Agent Game Environment (SCRIMMAGE) [18], provides an ideal framework for constructing simulations of UAS operations because it provides interfaces for specifying autonomy, controller, motion, network, and metric capabilities. It also provides an interactive viewer for visualizing all UAS, backdropped with terrain and buildings. Fig. 9 is a snapshot of the SCRIMMAGE viewer for this case study. The terrain data, in this viewer, consists of United States Department of Agriculture (USDA) aerial imagery [19] and United States Geological Services (USGS) elevation data [20]. Whereas the building data, which includes building footprints and building heights, are from Microsoft [13].

Fig. 9 SCRIMMAGE viewer

In addition to displaying the current position of multiple UAS, SCRIMMAGE was configured to display additional objects. For each UAS, spheres are drawn to show the location of its waypoints; a green line is drawn to show where the UAS is relative to its target waypoint; a yellow line is drawn to show its desired velocity vector. These features were critical for understanding the intent of the UAS’ autonomy algorithm.

The SCRIMMAGE simulation is configured through an extensible markup language (XML) mission file. Table 4 lists pertinent simulation settings that were loaded from this mission file. The time step value is pertinent because it affects the simulation’s accuracy. Although a smaller time step increases the simulation’s accuracy, it makes the simulation more computationally expensive and slower. In this simulation, the time step value is heuristically ascertained. The simulation’s x-axis points in the east direction, and its y-axis points in the north direction. The origin’s position was selected such that every UAS always has positive Cartesian coordinates. This requirement was imposed because of the obstacle constraint implementation in ETOL, which uses the Computational Geometry Algorithm Library (CGAL) to split buildings into polygon partitions [21].

In this simulation, ETOL resides in a UAS’ autonomy component. This autonomy component determines the desired velocity vector based on a goal waypoint. The Gurobi eSolver was selected for this task. In other words, ETOL used Gurobi to compute the desired velocity vector, based on the UAS’ current position, goal position and constraints. This computational task is configured to occur both asynchronously and continuously, such that desired velocity vector remains the same until a new value is computed. This approach is a form of model-predictive control [22]. The ETOL

Table 4 SCRIMAGE simulation settings

Parameter	Value
Time Step	0.05 s
Latitude Origin	33.771294°
Longitude Origin	-84.4075°
Altitude Origin	282 m

settings are loaded from a ETOL XML file. Table 5 lists the primary ETOL settings that are loaded from the ETOL XML file. From these settings, the MILP model shown in Eq. (12) is constructed, within ETOL.

$$\begin{aligned}
 \min \mathcal{J} &= \sum_{i=0}^{n_{\text{steps}}} \sum_{k=3}^5 i u_{k,i} \\
 \text{s.t.} \quad & u_{j+3,i} \geq u_{j,i}, \forall j \in \{0, 1, 2\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & u_{j+3,i} \geq -u_{j,i}, \forall j \in \{0, 1, 2\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & x_{j,k} = x_{j,k-n_{x_r}} + \Delta t \cdot u_{j,k-n_{u_r}}, \forall j \in \{0, 1, 2\}, \forall i \in \{\max(n_{x_r}, n_{u_r}), \dots, n_{\text{steps}}\} \\
 & x_{j,0} = x_j^{(0)}, \forall j \in \{0, 1, 2\} \\
 & x_{j,n_{\text{steps}}} \leq x_j^{(t_f)} + x_{tol}, \forall j \in \{0, 1, 2\} \\
 & x_{j,n_{\text{steps}}} \geq x_j^{(t_f)} - x_{tol}, \forall j \in \{0, 1, 2\} \\
 & x_{j,i} \leq x_{j,ub}, \forall j \in \{0, 1, 2\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & x_{j,i} \geq x_{j,lb}, \forall j \in \{0, 1, 2\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & u_{j,i} \leq u_{j,ub}, \forall j \in \{0, 1, \dots, 5\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & u_{j,i} \geq u_{j,lb}, \forall j \in \{0, 1, \dots, 5\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\}
 \end{aligned} \tag{12}$$

where $\mathbf{x}^{(0)} \in \mathbb{R}^3$ is the current position of the UAS, and $\mathbf{x}^{(t_f)} \in \mathbb{R}^3$ is the position of the target waypoint. This model is also updated to include obstacle avoidance constraints based on Eq. (6). The obstacles in the simulation are the buildings. Since the obstacles are three-dimensional polygons, Eq. (6) is modified to create Eq. (13), which accounts for a building's height.

$$\begin{aligned}
 & x_{2,i} + \mathcal{M}\zeta_i^{(2)} \geq h + d_{sm}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & x_{1,i} - m_k^{(0)} x_{0,i} - \mathcal{M}\zeta_{k,i}^{(0)} \leq a_{k,2}^{(0)} - m_k^{(0)} a_{k,1}^{(0)} - d_{sm}, \forall k \in \{1, 2, \dots, N\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & x_{1,i} - m_k^{(1)} x_{0,i} + \mathcal{M}\zeta_{k,i}^{(1)} \geq a_{k,2}^{(1)} - m_k^{(1)} a_{k,1}^{(1)} + d_{sm}, \forall k \in \{1, 2, \dots, M\}, \forall i \in \{0, 1, \dots, n_{\text{steps}}\} \\
 & \sum_{k=1}^N \zeta_{k,i}^{(0)} + \sum_{k=1}^M \zeta_{k,i}^{(1)} + \zeta_i^{(2)} \leq N + M + 0.5, \forall i \in \{0, 1, \dots, n_{\text{steps}}\}
 \end{aligned} \tag{13}$$

where d_{sm} is a safety margin to prevent the UAS from touching the obstacle, and h is a building's height. In essence, Eq. (13) causes the mailman to avoid the space that is occupied by a building.

The simulation also relies on Robot Operating System (ROS). It uses a ROS service to quickly retrieve building data near a geographic location. This ROS service is quick because it loads and keeps a single shared copy of the Microsoft building footprint data in memory. Each UAS' autonomy component uses this service to identify nearby buildings. In this case study, the mailman is the only UAS that could fly into a building. Although it is possible to load many, if not all, buildings into the MILP model as EXZs, this MILP model takes much longer to solve because it has many more binary variables. Consequently, this simulation only loaded buildings that the UAS is likely to collide into. When the simulation starts, the mailman contacts the building ROS service to get buildings with any corner that lies within a 40 m radius of $\langle 1150 \text{ m}, 750 \text{ m}, 20 \text{ m} \rangle$ and $\langle 570 \text{ m}, 955 \text{ m}, 20 \text{ m} \rangle$; the mailman then constructs a MILP model. During the

Table 5 ETOL Settings

Parameter	Value
Number of Time Steps (n_{steps})	40
Time Step Size (Δt)	0.4 s
Number of States (n_x)	3
State Reverse Time Horizon (n_{x_r})	1
State Tolerance (x_{tol})	0.5 m
State x_0 Bounds	$\langle 0 \text{ m}, 1750 \text{ m} \rangle$
State x_1 Bounds	$\langle 0 \text{ m}, 1500 \text{ m} \rangle$
State x_2 Bounds	$\langle 0 \text{ m}, 500 \text{ m} \rangle$
Number of Controls (n_u)	6
Control Reverse Time Horizon (n_{u_r})	1
Control u_j Bounds, $\forall j \in \{0, 1, 2\}$	$\langle 21 \text{ m s}^{-1}, 21 \text{ m s}^{-1} \rangle$
Control u_k Bounds, $\forall k \in \{3, 4, 5\}$	$\langle 0 \text{ m s}^{-1}, 21 \text{ m s}^{-1} \rangle$

simulation, the mailman does not reconstruct a new MILP model. Rather, it simply changes the $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(t_f)}$ in the model, and solves the updated MILP problem.

The simulation also uses another ROS service. Like the building service, this service loads and keeps a single copy of the wind data from Section V. A SCRIMMAGE motion plugin constantly retrieves the wind vector, at a geographic position, from this ROS service. The motion plugin then perturbs the vehicles velocity by the wind vector. After which, the motion plugin numerical integrates the velocity to determine the next position of the vehicle.

V. Application: Computational Fluid Dynamics

In an urban area, turbulent flow is a critical external disturbance on a UAS' motion. Hence, an incorporation of the flow behavior is necessary in an assessment of UAS operations. Formerly, a study of the turbulent flow was performed via experiments because of a lack of efficient computational tools. However, experimental methods only provide data at measurement points. Current computational resources, which perform much quicker mathematical calculations with a higher precision, can be used to predict turbulent flow behavior. Therefore, by applying computational fluid dynamics (CFD) results to the UAS motion, an accurate vehicle trajectory can be planned. The CFD method used in this work is called Reynolds Average Navier-Stokes equations (RANS).

A. Reynolds Average Navier-Stokes equations (RANS) with $k - \epsilon$ turbulence model

The governing equations of fluid dynamics with incompressible, no heat transfer and no external source term conditions consist of two conservation laws, which are the conservation of mass

$$\nabla \cdot \vec{U} = 0 \quad (14)$$

and the conservation of momentum

$$\rho \left(\frac{\partial \vec{U}}{\partial t} + \vec{U} \cdot \nabla \vec{U} \right) = -\nabla p + \nu \Delta \vec{U} \quad (15)$$

where ρ is a fluid density, ν is kinematic viscosity, and $\vec{U}$ is a three-dimensional velocity vector [23]. To solve for $\vec{U}$ from (14) and (15), the velocity vector is first decomposed into the mean velocity ($\bar{u}$) and the fluctuating velocity (u'), i.e.,

$$\vec{U} = \bar{u} + u' \quad (16)$$

Substituting (16) into (14) and (15), yields

$$\frac{\partial \bar{u}}{\partial x_i} = 0, \forall i, j \in \{1, 2, 3\} \quad (17)$$

$$\rho \frac{D \bar{u}_i}{Dt} = -\frac{\partial \bar{p}}{\partial x_i} + \mu \Delta \bar{u}_i - \rho \left(\frac{\partial \overline{u'_i u'_j}}{\partial x_j} \right), \forall i, j \in \{1, 2, 3\} \quad (18)$$

The last term in Eq. (18) is a nonlinear term containing $\overline{u'_i u'_j}$, which commonly refers to the Reynolds stress. The turbulence model is introduced to solve this nonlinearity by using the assumption that represents the existing turbulence as an increasing viscosity [24], which allows this nonlinear term to be modeled as the eddy-viscosity equation.

$$-\rho \overline{u'_i u'_j} = \mu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} - \frac{2}{3} \rho \delta_{ij} k \right), \forall i, j \in \{1, 2, 3\} \quad (19)$$

where μ_t is the eddy viscosity, $\delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}$ is the Kronecker delta, and k is turbulence kinetic energy.

This assumption plays an important role in turbulent flow regions with high Reynolds number. Instead of directly solving this non-linear term, RANS use turbulence models together with ensemble averaged flow field equation to reduce computational resources. One of turbulence models that is often applied for general CFD problem is the $k - \epsilon$ model [24] described as follows.

$$k = \frac{1}{2} \overline{u'_i u'_i} \quad (20)$$

$$\epsilon \approx \frac{k^{3/2}}{L} \quad (21)$$

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon} \quad (22)$$

where ϵ is dissipation rate, C_μ is a constant, and L is a large vortex diameter also known as turbulence length scale.

The $k - \epsilon$ model is suitable for wake regions that are not too close to the wall or the building. Since it is natural that the vehicle should maintain some distance away from these obstacles, this model is appropriate for our UAV application. An opensource CFD software OpenFOAM [25] is used as a CFD framework that implements this RANS method with $k - \epsilon$ turbulence model. The resulting flow velocity is then integrated into each UAS' motion model.

B. Numerical Setting

Georgia Institute of Technology (GT) campus is selected as the urban environment to demonstrate this CFD approach with an explicit geometry in the form of a circular area that has a radius of 1430 m, as shown in Fig. 10. This domain size is chosen by setting the dimensional blockage ratio (BR) to 0.17 [26].

$$BR_i = \frac{i_{\text{building}}}{i_{\text{domain}}} = 17\%, \forall i \in \{\text{Length, Weight, Height}\} \quad (23)$$

The domain size is in the microscale type [26] for spatial distance between 100 to 2000 m, in which the incoming flow may not be uniform across the inlet area. For this reason, the wind profile modeling by Richards and Hoxey [27] is applied to simulate a pseudo Atmospheric Boundary Layer (ABL) profile at the inlet boundary condition, which modifies the $k - \epsilon$ turbulence model as follows

$$U(z) = \frac{U_{ABL}^*}{\kappa} \ln \frac{z + z_0}{z_0} \quad (24)$$

$$k = \frac{U_{ABL}^{*2}}{\sqrt{C_\mu}} \quad (25)$$

$$\epsilon(z) = \frac{U_{ABL}^{*3}}{\kappa(z + z_0)} \quad (26)$$

where U_{ABL}^* , κ , z_0 , C_μ are ABL friction velocity, Von-Karman constant, the roughness height and constant value, respectively. Then, the symmetry condition is set at the lateral sides and the top boundary to prevent the normal vector gradient from causing divergence at those areas [28]. At the outlet boundary condition, a fully developed flow, together with backward flow protection, is set.

Fig. 10 The entire domain for CFD analysis

C. Result

The CFD result is illustrated by a set of figures. First, the velocity profile, the turbulence kinetic energy, and the dissipation rate at different positions, together with their theoretical values, along the normal direction (height of the domain) are shown, respectively in Fig. 11 to 13. It can be seen that once the flow passes the buildings, its velocity and kinetic energy are decreased, and the dissipation rate is roughly unchanged.

Fig. 14 shows the velocity contour near the boundary of the explicit domain. The region of interest is the Georgia Tech (GT) campus, which is covered by the circle with a radius of 1.300 m. An additional 10 % of the area is extended to generate the artificial upstream wake.

Besides the velocity vector, another useful parameter for visualizing the fluctuation in velocity is the Q -criteria [29], which can be calculated using Eq. (27).

$$Q = \frac{1}{2} \left(\|\Omega\|^2 - \|\sigma\|^2 \right) \quad (27)$$

where Ω is vorticity tensor and σ is strain-rate tensor. The threshold value of the Q -criteria, which ascertains vortex flow, is 10^{-5} . Fig. 15 shows that the tallest building induces the turbulent flow, in the way of increasing turbulence

Fig. 11 Velocity profile along normal direction at $Y = 0$ m and $X = \text{inlet}, \pm 1430$ m, ± 1300 m, and outlet

Fig. 12 Kinetic energy along normal direction at $Y = 0$ m and $X = \text{inlet}, \pm 1430$ m, ± 1300 m, and outlet

Fig. 13 Dissipation rate along normal direction at $Y = 0$ m and $X = \text{inlet}, \pm 1430$ m, ± 1300 m, and outlet

Fig. 14 Velocity contour around the region of interest

intensity following Eq. (28), for a distance longer than twice of its height. Lastly, the three-dimensional view of the high turbulence intensity region of the GT campus is shown in Fig. 16.

$$\text{Turbulent Intensity} = \frac{\sqrt{\frac{2}{3}k}}{U_{\text{ref}}} \quad (28)$$

where U_{ref} is a reference velocity from a wind station.

Fig. 15 The highly distributed turbulence intensity contour on Q-criteria > 0.1

Fig. 16 The 3D visualization of the high turbulence intensity area on GT campus

VI. Assessment of UAS Operations in an Urban Environment

A SCRIMMAGE simulation of Section IV's UAS operations served as the basis for this paper's assessment. The trajectories from this simulation are shown in Fig. 17 and 18. Fig. 18 provides a three-dimensional view of the trajectories. From the set of eight campus monitoring UAS, pairs of UAS are separated by altitude. Consequently, only four racetracks can be seen in Fig. 17, which is a two-dimensional top-down view of the trajectories. These results led to three key findings: operational constraints can be simplified by factoring a planner's time horizon, safety margin needs

to account for wind near buildings, and future UAS operations will require autonomous traffic controllers that assess the safety of UAS flight plans.

Fig. 17 Simulation results in 2D

Fig. 18 Simulation results in 3D

Ideally, ETOL should have a time horizon within which the UAS can reach its goal position. This ideal time horizon is needed if the optimizer is to find the true optimal trajectory. However, an increase in time horizon typically requires an increase in the number of nodes for an accurate result. Since increasing the number of nodes increases ETOL's computational time, a suboptimal trajectory is computed. In this simulation, ETOL's time horizon T is computed from

$\Delta t \times n_{steps}$, which is 16 s based on Table 5. Consequently, all waypoints were selected such that it is feasible to reach it from the current UAS position within the speed limit. This workaround was used by the mailman, which had to travel across the Georgia Tech campus. While traversing campus, the mailman's waypoints are above the buildings. With this workaround, the computational load can be reduced by not including the building avoidance constraints in the MILP model during this flight phase. In other words, the problem can be simplified by configuring different ETOL instances for the take-off, en route, and landing phases.

The effect of wind was not profound in the simulation results. However, the impact of wind is more pronounced in regions that have high wind speeds and high turbulence intensities. The simulation assumes the autonomy element does not know the wind vector. Consequently, the mailman did not compute a desired velocity that accounts for wind. When the mailman passes between two buildings, as it did in Fig. 21, it is possible for the wind to push the vehicle into the building. ETOL can work around this issue if the distance safety margin d_{sm} is increased to the point where it is no longer feasible to fly between these two buildings. In other words, an increase in d_{sm} means ETOL has a smaller feasible state space, which leads it to compute safer, but suboptimal trajectories. Multiple simulations with various d_{sm} values can be run to determine an appropriate safety margin that adequately accounts for the wind.

Fig. 19 Mailman's path across campus [30]

Although the UAS in the SCRIMAGE simulation were configured to have conflict-free trajectories, urban operations will likely have many flight conflicts. These flight conflicts can be caused by weather events, in-flight reroutes, impromptu no-fly-zones, or any unforeseen flight event. Therefore, there is a need for a deconfliction resource, which plays air traffic controller (ATC) role. The SCRIMAGE viewer allowed the simulation user to act as a ATC because the user can control each UAS' waypoints. This deconfliction challenge is untenable if there are hundreds, if not thousands of UAS. In this case, the ATC role should be automated. For example, each UAS can share its ETOL computed trajectory to a ATC; the ATC identifies conflicts; the ATC sends collision information to the UAS; the UAS uses ETOL to replan with new collision avoidance constraints.

VII. Conclusion

UAS operations in an urban environment will bring new economic opportunities, but not at the expense of public safety. In the quest for safe UAS operations, simulations will play a key role. This paper examines an approach for assessing UAS operations by using ETOL to constantly solve a VGP within a simulator. The structure of the VGP is introduced, along with a new open-source library that solves a VGP with various algorithms. This software was introduced as ETOL, and it was created to modularize VGPs and interface with VGP solvers. Consequently, a VGP can be quickly solved by different algorithms and many VGPs can be solved by many algorithms.

A case study of UAS operations in the Georgia Institute of Technology campus led to three key findings. First, the UAS flight should be broken into phases to simplify the computational task of solving the VGP. In other words,

Fig. 20 Mailman's obstacle avoidance [31]

(a) Mailman before it reaches the building

(b) Mailman avoiding the building

Fig. 21 Mailman flying between buildings [31, 32]

there should be a VGP for each flight phase, and each VGP should only include obstacle constraints that are pertinent to that flight phase. Wind plays a big role in the urban environment because the buildings introduce complex flow fields. Consequently, it can be dangerous to fly near buildings. This paper showed how to modify the VGP with a safety margin that accounts for wind. The importance of a ATC is also stressed. The paper proposed using ETOL to compute trajectories that are sent to a ATC for deconfliction.

This assessment is a steppingstone for many more assessments. For example, an assessment of air mobility operations at King Abdullah University of Science and Technology campus is a feasible next step. This assessment could examine the operation of hundreds of UAS by using a high-performance computing (HPC) cluster. It could also examine the effects of changing the CFD's inlet direction, which will result in a different flow field. A ATC could be added to test deconfliction algorithms. An assessment of safety margin as a function of wind conditions could also be performed. These proposed assessments will play an important role in future debates about UAS operations in urban environments.

Acknowledgments

For the CFD result, this research used the resources of the Supercomputing Laboratory at King Abdullah University of Science & Technology (KAUST) in Thuwal, Saudi Arabia as well as the financial support from the KAUST baseline fund and the visiting student research program. The authors thank the engineers in the Aerospace, Transportation & Advanced Systems Laboratory at Georgia Tech Research Institute (GTRI) for answering our SCRIMMAGE-related questions. The authors also thank Dr. Brian German, who is the director of the Center for Urban and Regional Air

Mobility (CURAM), and his graduate students for vigorous discussions about trajectory optimization and urban air mobility (UAM) operations.

References

- [1] Chakrabarty, A., Stepanyan, V., Krishnakumar, K. S., and Ippolito, C. A., “Real-Time Path Planning for Multi-Copters Flying in UTM -TCL4,” *AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, 2019. doi:10.2514/6.2019-0958.
- [2] Egorov, M., Kuroda, V., and Sachs, P., “Encounter Aware Flight Planning in the Unmanned Airspace,” *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 2019, pp. 1–15. doi:10.1109/ICNSURV.2019.8735399.
- [3] Sanni, O., “Extensible Trajectory Optimization Library (ETOL),” , Nov. 2020. doi:10.5281/zenodo.4297132, URL <https://doi.org/10.5281/zenodo.4297132>.
- [4] Haslum, P., Lipovetzky, N., Magazzeni, D., and Muise, C., “An Introduction to the Planning Domain Definition Language,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Vol. 13, No. 2, 2019, pp. 1–187. doi:10.2200/S00900ED2V01Y201902AIM042.
- [5] Ben Amarat, S., and Zong, P., “3D Path Planning, Routing Algorithms and Routing Protocols for Unmanned Air Vehicles: A Review,” *Aircraft Engineering and Aerospace Technology*, Vol. 91, No. 9, 2019, pp. 1245–1255. doi:10.1108/AEAT-01-2019-0023.
- [6] Yang, L., Qi, J., Xiao, J., and Yong, X., “A Literature Review of UAV 3D Path Planning,” *Proceeding of the 11th World Congress on Intelligent Control and Automation*, 2014, pp. 2376–2381. doi:10.1109/WCICA.2014.7053093.
- [7] “Dymos: Open Source Optimization of Dynamic Multidisciplinary Systems,” 2018.
- [8] Becerra, V. M., “Solving Complex Optimal Control Problems at No Cost with PSOPT,” *2010 IEEE International Symposium on Computer-Aided Control System Design*, 2010, pp. 1391–1396. doi:10.1109/CACSD.2010.5612676.
- [9] Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Le Bodic, P., Maher, S. J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D., and Witzig, J., “The SCIP Optimization Suite 7.0,” Technical report, Optimization Online, March 2020. URL http://www.optimization-online.org/DB_HTML/2020/03/7705.html.
- [10] Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Le Bodic, P., Maher, S. J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D., and Witzig, J., “The SCIP Optimization Suite 7.0,” ZIB-Report 20-10, Zuse Institute Berlin, March 2020. URL <http://nbn-resolving.de/urn:nbn:de:0297-zib-78023>.
- [11] Markhorin, A., “GNU Linear Programming Kit,” Moscow Aviation Institute, 2012.
- [12] “Gurobi Optimizer Reference Manual,” Gurobi Optimization, LLC, 2020.
- [13] “Microsoft Building Footprints,” 2018.
- [14] Bemporad, A., and Morari, M., “Control of Systems Integrating Logic, Dynamics, and Constraints,” *Automatica*, Vol. 35, No. 3, 1999, pp. 407–427. doi:10.1016/S0005-1098(98)00178-2.
- [15] Richards, A., and How, J. P., “Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming,” *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, Vol. 3, 2002, pp. 1936–1941. doi:10.1109/ACC.2002.1023918.
- [16] Schouwenaars, T., De Moor, B., Feron, E., and How, J., “Mixed Integer Programming for Multi-Vehicle Path Planning,” *2001 European Control Conference (ECC)*, 2001, pp. 2603–2608. doi:10.23919/ECC.2001.7076321.
- [17] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed., Advances in Design and Control, Society for Industrial and Applied Mathematics, Philadelphia, 2010.
- [18] DeMarco, K., Squires, E., Day, M., and Pippin, C., “Simulating Collaborative Robots in a Massive Multi-Agent Game Environment (SCRIMMAGE),” *Distributed Autonomous Robotic Systems*, Vol. 9, edited by N. Correll, M. Schwager, and M. Otte, Springer International Publishing, Cham, 2019, pp. 283–297. doi:10.1007/978-3-030-05816-6_20.

- [19] U.S. Geological Survey, “USDA National Agriculture Imagery Program (NAIP),” , Oct. 2016. URL <https://www.sciencebase.gov/catalog/item/59eb39d7e4b0026a55ff888f>.
- [20] U.S. Geological Survey, “National Elevation Dataset,” , Oct. 2018. URL <https://www.usgs.gov/core-science-systems/ngp/3dep>.
- [21] Hert, S., “2D Polygon Partitioning,” *CGAL User and Reference Manual*, CGAL Editorial Board, 2020, 5.1.1 ed. URL <https://doc.cgal.org/5.1.1/Manual/packages.html#PkgPartition2>.
- [22] Camacho, E. F., and Bordons, C., *Model Predictive Control*, Advanced Textbooks in Control and Signal Processing, Springer, London, 2007. doi:10.1007/978-0-85729-398-5.
- [23] Pope, S. B., “Turbulent flows,” , 2001.
- [24] Ferziger, J. H., Perić, M., and Street, R. L., *Computational methods for fluid dynamics*, Vol. 3, Springer, 2002.
- [25] Jasak, H., “OpenFOAM: open source CFD in research and industry,” *International Journal of Naval Architecture and Ocean Engineering*, Vol. 1, No. 2, 2009, pp. 89–94.
- [26] Blocken, B., “Computational Fluid Dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations,” *Building and Environment*, Vol. 91, 2015, pp. 219 – 245. doi:<https://doi.org/10.1016/j.buildenv.2015.02.015>, URL <http://www.sciencedirect.com/science/article/pii/S0360132315000724>, fifty Year Anniversary for Building and Environment.
- [27] Richards, P., and Hoxey, R., “Appropriate boundary conditions for computational wind engineering models using the k- ϵ turbulence model,” *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 46-47, 1993, pp. 145 – 153. doi:[https://doi.org/10.1016/0167-6105\(93\)90124-7](https://doi.org/10.1016/0167-6105(93)90124-7), URL <http://www.sciencedirect.com/science/article/pii/0167610593901247>, proceedings of the 1st International on Computational Wind Engineering.
- [28] Toparlar, Y., Blocken, B., Vos, P. v., Van Heijst, G., Janssen, W., van Hooff, T., Montazeri, H., and Timmermans, H., “CFD simulation and validation of urban microclimate: A case study for Bergpolder Zuid, Rotterdam,” *Building and environment*, Vol. 83, 2015, pp. 79–90.
- [29] Hunt, J., Wray, A., and Moin, P., “Eddies, stream, and convergence zones in turbulent flows, Centre for turbulence Research Rep,” *CTR-S88*, 1988.
- [30] Sanni, O., “UAS_Obstacle_Avoidance_GT_3D_Simulation,” , Dec. 2020. doi:10.6084/m9.figshare.13309451, URL https://figshare.com/articles/media/UAS_Obstacle_Avoidance_GT_3D_Simulation/13309451/1.
- [31] Sanni, O., “Mailman_w_Obstacle_Avoidance,” , Dec. 2020. doi:10.6084/m9.figshare.13309415, URL https://figshare.com/articles/media/Mailman_w_Obstacle_Avoidance/13309415/1.
- [32] Sanni, O., “Mailman_wo_Obstacle_Avoidance,” , Dec. 2020. doi:10.6084/m9.figshare.13309430, URL https://figshare.com/articles/media/Mailman_wo_Obstacle_Avoidance/13309430/1.