Vision-guided Planning and Control for Autonomous Taxiing via Convolutional Neural Networks

Chang Liu* and Silvia Ferrari†
Cornell University, Ithaca, NY, 14853

This paper presents a new approach for autonomous taxiing that can potentially improve the efficiency and safety of ground operations in commercial or military airports. Research on airport automation so far has focused primarily on scheduling and coordination of piloted aircraft for improved overall efficiency and lower operational costs. This paper develops a novel vision-guided path planning and control approach that could potentially lead to unmanned taxiing and takeoff. Autonomous taxing is a challenging problem because on-board perception algorithms must be capable of translating verbal commands provided by the Air Traffic Control (ATC) tower and, at the same time, react to a wide range of possible runway incursions, taxiway and runway conditions, and ground crew behaviors, in order to operate safely in complex airport environments without human intervention. In this paper, the autonomous taxiing problem is formulated as a hybrid planning and control problem that can be solved by an approach based on vision-based perception, obstacle avoidance, and feedback control theory. By harnessing convolutional neural networks (CNNs) for computer vision, the information obtained by on-board cameras about surrounding environments can be integrated with prior information – such as airport maps – and ATC commands to compute motion plans, while simultaneously detecting and adapting to dangerous situations such as runway incursions. Simulation results obtained using the photo-realistic physics-based Unreal Engine^{TM} simulation tool show that the proposed approach can be potentially used some day to automate airport ground operations, even in crowded environments populated with ground crew, vehicles, and other aircraft.

Nomenclature

$\mathbf{s}$	=	hybrid system continuous state
μ	=	hybrid system discrete state
$\mathbf{u}$	=	hybrid system continuous control input
ξ	=	hybrid system discrete control input
c	=	Air Traffic Control command
x, y	=	$x-$, $y-$ coordinate in the world frame
υ	=	aircraft speed
θ	=	aircraft yaw angle
β	=	aircraft acceleration
ϕ	=	aircraft steering angle
ΔT	=	sampling interval
L	=	distance between front and rear axles of the aircraft
$\mathbf{z}$	=	camera measurement
l	=	a binary variable indicating whether an object is inside the camera's field of view
d_l	=	the distance between the aircraft and the object on the ground
$\mathcal{G}$	=	graph generated from airport diagram
J	=	objective function for continuous control law
d_s	=	user-specified safety margin between the aircraft and the object

*Postdoctoral Associate, Laboratory for Intelligent Systems and Controls (LISC), Sibley School of Mechanical and Aerospace Engineering.
†Professor and Director, Laboratory for Intelligent Systems and Controls (LISC), Sibley School of Mechanical and Aerospace Engineering.

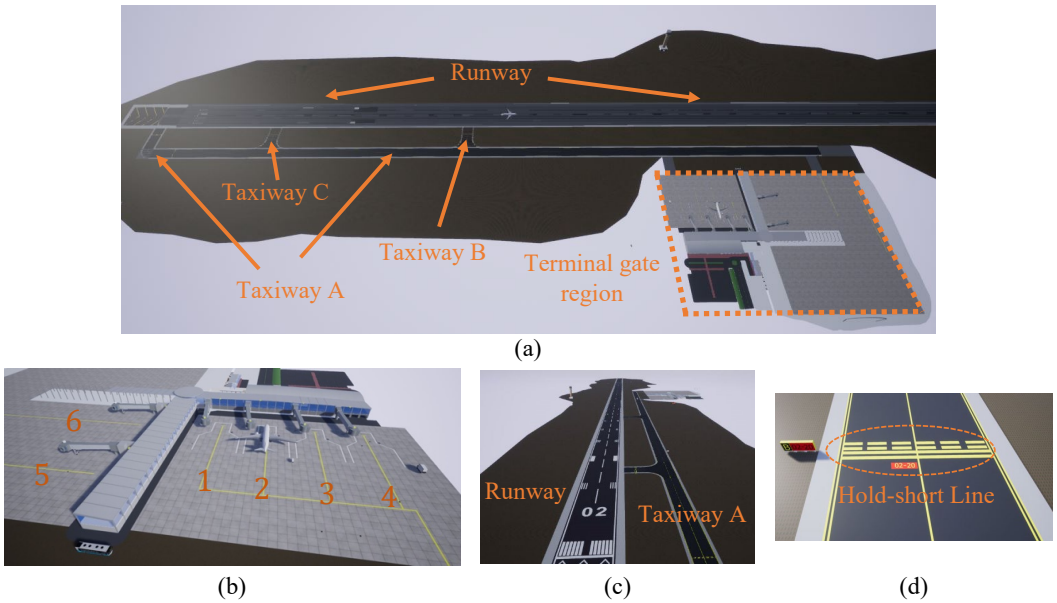

Figure 1. Simulated airport environment using Unreal Engine. (a) The airport has three taxiways ("A", "B", "C") and a runway (with two labels "02" and "20"). (b) Terminal region has six gates $(1, \ldots 6)$**. (c) Standard airport signs and ground markings are included in the simulated airport. (d) The sign for a hold-short line.**

I. Introduction

The automation of airport ground operations has received increasing attention in recent years as aerodromes are becoming increasingly complex and crowded due to heavier traffic flows. Prior research has focused on intelligent air traffic management by automating the scheduling and coordination of flights to reduce overall operation time and fuel consumption [1, 2]. With the advent of powerful speech-recognition and computer-vision algorithms, as well as other real-time sensor processing and fusion capabilities [3], research on autonomous perception [4] and control [5, 6] has opened new potential applications, such as autonomous taxiing [7, 8]. Because Air Traffic Control (ATC) in commercial airports is coordinated by human operators, planning and control algorithms must take verbal ATC commands into account and translate them into aircraft decision making and control policies. Additionally, in order to operate safely in a broad range of environmental conditions, obstacles and unforeseen conditions, such as runway incursions, must be detected and avoided *in situ* by the autonomous taxiing algorithms and promptly reported to the ATC tower. Corrective control must be automatically generated and implemented on-board the aircraft in order to avoid accidents. The approach developed in this paper is demonstrated in a medium-sized airport in which *incursion objects*, including moving pedestrians, vehicles, or animals, are simulated within airport movement areas, i.e. runways and taxiways, without ATC authorization. This situation, referred to as *runway/taxiway incursion* by the Federal Aviation Administration (FAA) [9], requires the aircraft to autonomously recognize situations involving unauthorized pedestrians and vehicles, modify the motion plan accordingly, and communicate with the ATC tower to obtain revised commands.

The autonomous taxiing approach developed in this paper systematically combines scene perception, path planning, and safety-critical control capabilities. An on-board aircraft perception system is developed using a CNN-based object detection and recognition algorithm for incursions by animals, vehicles, or people that can potentially cause dangerous collisions during takeoff, landing, or taxiing. The approach is developed and tested using a simulated aircraft and airport environment created in Unreal Engine^{TM} (UE) [10], which is a high-fidelity physical simulation software with photo-quality rendering, as shown in Fig. 1. Verbal ATC commands are used to generate a nominal path plan, and corresponding centerline reference trajectory, using a topological graph. A hybrid system model [11, 12] of aircraft decision and control is developed for following the nominal path plan under both normal and unforseen conditions by means of five discrete operation modes applicable to normal taxiing and risk handling. The autonomous taxiing algorithm controls mode switching based both on the feedback from on-board vision algorithms and on the ATC commands so that the aircraft is able to follow ATC commands when appropriate, or stop to avoid collisions when incursion detection is above a desired confidence level. An optimal control law is developed for each discrete mode and used for aircraft control as indicated by the mode switching algorithm.

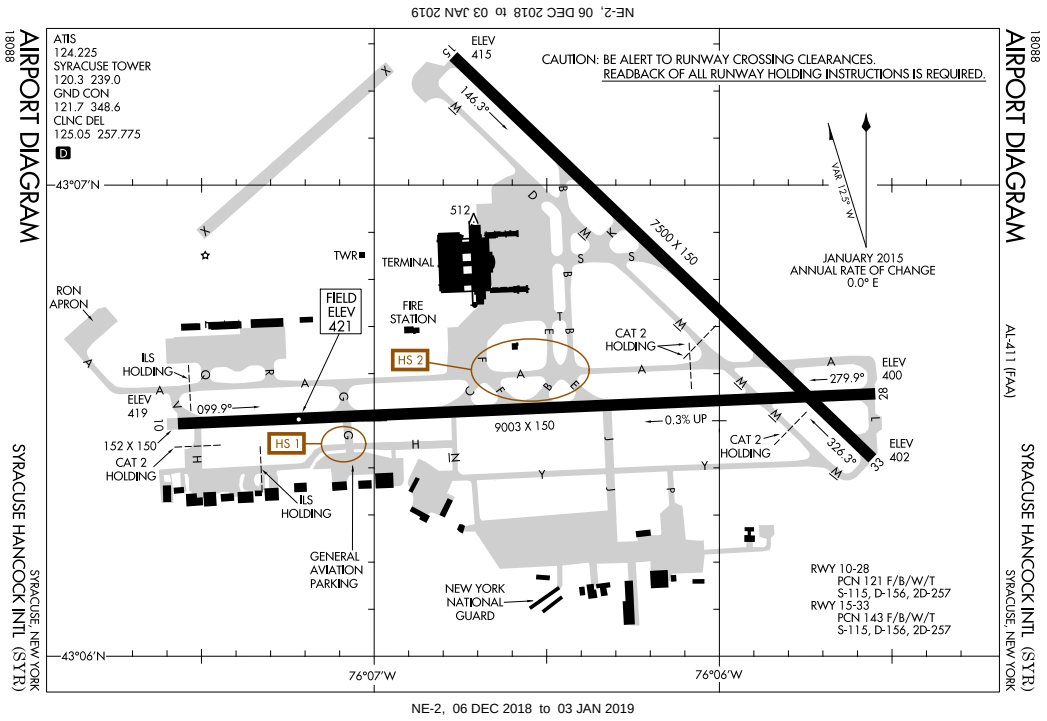

Figure 2. Airport diagram of Syracuse Hancock International Airport.

The paper is organized as follows. Section II develops an airport model and formulates the autonomous taxiing problem using a hybrid system model. Section III describes the CNN object detection and recognition approach for scene perception in airport environments. Section IV presents a graph-based path planning approach for translating verbal ATC commands into a high-level reference path, based on available airport maps (Fig. 2). The hybrid autonomous taxiing decision and control algorithm is described in Section V, and demonstrated in Section VI where the aircraft is shown capable of following the reference path, while simultaneously handling dangerous situations if needed.

II. Problem Formulation

This paper considers the problem of automating the taxiing process of aircraft operating in medium-sized commercial airports without human intervention. The autonomous taxiing algorithm is assumed to have access to the airport diagram, typically provided online according to FAA rules and conventions [13]. As shown in Fig. 2, the airport diagram must always include information about all runways, taxiways, and terminal regions, conditions (e.g. nearby equipment), with corresponding labels utilized by the ATC operators to provide pilots with standardized verbal commands on the path to follow for landing, taxiing, and takeoff. Under *normal conditions*, defined as airport environments in which perceived elements of the scene match the published airport diagram, the autonomous taxiing system must follow the Air Traffic Control (ATC) commands and navigate to the specified destination for departure or arrival, while guaranteeing the safety of the aircraft by avoiding collisions with authorized airport vehicles and personnel or *agents*. This is accomplished by generating a feasible reference path based on the ATC commands and the available airport diagram, and by labeling authorized agents in the CNN database. Under normal conditions, the aircraft controller visually tracks the reference path corresponding to runway centerlines and connecting arcs, while avoiding collisions based on visual feedback. Under *unforeseen conditions*, defined as deviations from the published airport diagram and authorized agents, such as runway/taxiway incursions, the autonomous taxiing system switches modes to immediately cope with potentially dangerous situations, e.g. stop, and then re-plan the reference path by communicating to the ATC tower and request new commands. A block diagram of the autonomous autonomous taxiing system is shown in Fig. 3, and each block is described in detail in the following sections.

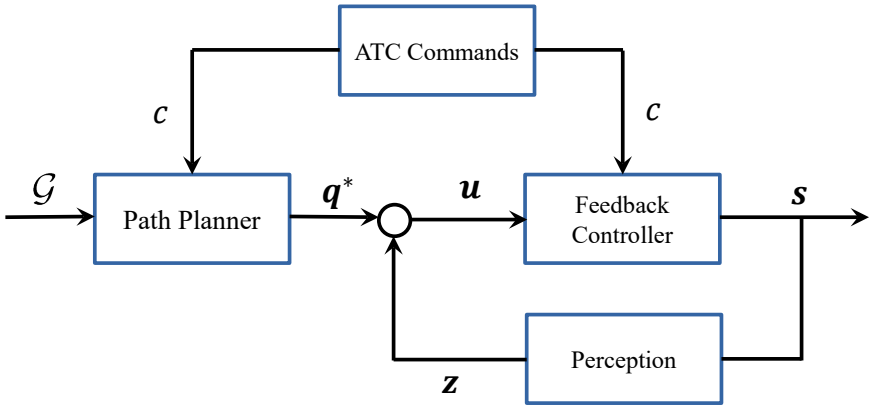

Figure 3. Block diagram of the autonomous autonomous taxiing system, in which first generates a feasible reference path for the aircraft based on ATC commands and airport diagram and a controller performs centerline tracking based on visual feedback and ATC commands. When incursion objects are detected by the perception algorithm, the autonomous autonomous taxiing system controls the aircraft to avoid colliding with these objects.

A. Airport Model

The airport model is constructed by generating a topological graph from the airport diagram and geographic information about runways, taxiways, and terminal gates. Let $\mathcal{W} \in \mathbb{R}^2$ represent a planar airport in 2-D space. A taxiway is a path that connects runways with aprons, hangars, terminals and other facilities. Taxiways are always labeled by a finite set of upper-case letters from English alphabet, e.g. $\mathcal{A} = \{A, H, F, \ldots, M\}$, which can be generated from the airport diagram. A *taxiway* is defined by a connected compact set $\mathcal{W}_a \in \mathcal{W}$ labeled by a letter (or index) $a \in \mathcal{A}$, and representing the corresponding gray region in Fig. 2. Each taxiway has a centerline clearly marked on the pavement as a single continuous yellow line (Fig. 4(a)) indicating the reference path to be followed by all aircraft while taxiing. Let the centerline of taxiway a be denoted by the equation of a line in 2D,

$$h_a(\mathbf{q}) = 0, \quad \text{for} \quad \mathbf{q} = [x \quad y]^T \in \mathcal{W}_a \tag{1}$$

where $h_a : \mathbb{R}^2 \to \mathbb{R}$ is a continuous, differentiable function defined on $\mathcal{W}_a$. By using the chain rule, the total derivative of the implicit function in (1) can be obtained as follows,

$$0 = \frac{\partial h_a}{\partial x}dx + \frac{\partial h_a}{\partial y}dy \quad \therefore \frac{dy}{dx} = -\frac{\partial_x h_a}{\partial_y h_a}, \tag{2}$$

where the short-hand notation $\partial_x h_a$ denotes the partial derivative of h_a with respect to x [14].

When multiple taxiways join at an airport intersection, the region of intersection is denoted by the subset $\mathcal{W}_Q \subset \mathcal{W}$, where $Q = \{a_1, \ldots, a_m\}$ is the index set of the intersecting taxiways for $\bigcap_{i=1}^{m} \mathcal{W}_{a_i} \neq \varnothing$, and $\bigcap_{i=1}^{m} \mathcal{W}_{a_i} \subset \mathcal{W}_Q$. In an intersection region, yellow circular arcs (Fig. 4(a)) are painted on the taxiway pavement, smoothly connecting the centerline of one taxiway to another. Let an arc segment h_A denote the line connecting a pair of centerlines labeled by the pair $A = (a_1, a_2)$, where $h_A : \mathbb{R}^2 \to \mathbb{R}$ is a continuous, differentiable function that obeys the following properties:

1) $C_1 = \{\mathbf{q} \in \mathcal{W} \mid h_A(\mathbf{q}) = h_{a_1}(\mathbf{q})\}$ and $C_1' = \{\mathbf{q} \in \mathcal{W} \mid h_A(\mathbf{q}) = h_{a_2}(\mathbf{q})\}$ are singleton sets
2) Two intersecting taxiway centerlines have equal curvature at the intersection point, namely:

$$\left.\frac{\partial_x h_A}{\partial_y h_A}\right|_{\mathbf{q}} = \left.\frac{\partial_x h_{a_1}}{\partial_y h_{a_2}}\right|_{\mathbf{q}} \quad \forall \mathbf{q} \in C_1 \quad \text{and} \quad \left.\frac{\partial_x h_A}{\partial_y h_A}\right|_{\mathbf{q}} = \left.\frac{\partial_x h_{a_2}}{\partial_y h_{a_2}}\right|_{\mathbf{q}} \quad \forall \mathbf{q} \in C_1'$$

Property (1) ensures that the connecting arc intersects each centerline at a single *intersection point*, and property (2) guarantees smoothness.

Airport *runways* are labeled by a positive integer $p \in \mathcal{P} = \{1, 2, \ldots, 36\}$ that represents the magnetic azimuth of the runway's heading to the nearest deca-degrees [15]. Thus, a runway label p indicates the orientation of the runway is

within the range $[10p-5, 10p+5)$ (deg), such that during takeoff or landing the aircraft yaw angle is within this range. Every runway can typically be used for takeoff and landing in both directions and has separate labels for each one, e.g. 10–28 indicates two opposite directions (100° and 280°) an aircraft can use for takeoff and landing on a runway . A runway is represented by a connected compact set $\mathcal{W}_p \subset \mathcal{W}$ for all $p \in \mathcal{P}$ whose enclosure is mutually disjoint with respect to all taxiways, i.e., $\mathcal{W}_p \cap \mathcal{W}_a = \partial\mathcal{W}_p \cap \partial\mathcal{W}_a$, where $\partial\mathcal{W}$ denotes the boundary of $\mathcal{W}$. The centerline of a runway is marked by a white dashed line painted on the pavement that indicates the path for takeoff or landing (Fig. 1). Let the centerline of runway p be represented by the equation of a line in 2D,

$$h_p(\mathbf{q}) = 0, \quad \text{for} \quad \mathbf{q} = [x \quad y]^T \in \mathcal{W}_p \tag{3}$$

where $h_p : \mathbb{R}^2 \to \mathbb{R}$ is a continuous, differentiable function. When a taxiway a connects to a runway p, a circular arc connects the taxiway centerline to the runway centerline (Fig. 4(b)), where $\mathcal{W}_a \cap \mathcal{W}_p \neq \varnothing$ and $\mathcal{W}_a \cap \mathcal{W}_p = \partial\mathcal{W}_a \cap \partial\mathcal{W}_p$. Let an arc segment h_P denote the line connecting the centerlines of the pair $P = (a, p)$. The function $h_P : \mathbb{R}^2 \to \mathbb{R}$ is continuous and differentiable and obeys the following properties:

1) $C_2 = \{\mathbf{q} \in \mathcal{W} \mid h_P(\mathbf{q}) = h_a(\mathbf{q})\}$ and $C_2' = \{\mathbf{q} \in \mathcal{W} \mid h_P(\mathbf{q}) = h_p(\mathbf{q})\}$ are singleton sets
2) Intersecting taxiway and runway centerlines have equal curvature at the intersection point, namely:

$$\left.\frac{\partial_x h_P}{\partial_y h_P}\right|_{\mathbf{q}} = \left.\frac{\partial_x h_a}{\partial_y h_a}\right|_{\mathbf{q}} \quad \forall \mathbf{q} \in C_2 \quad \text{and} \quad \left.\frac{\partial_x h_P}{\partial_y h_P}\right|_{\mathbf{q}} = \left.\frac{\partial_x h_p}{\partial_y h_p}\right|_{\mathbf{q}} \quad \forall \mathbf{q} \in C_2'$$

Because of its safety-critical role in airport operations, any taxiway-runway intersection, referred to as *hold-short position*, is marked by a unique hold-short line painted on the pavement, as shown in Fig. 1(d)), where all taxiing aircraft must stop and hold for instructions until authorized to proceed onto the runway by the ATC tower. In this paper, a line segment h_D is used to denote every hold-short line between taxiway a and runway p in the airport, where $D = (a, p)$.

A *terminal gate* region, labeled by $g \in \mathbb{Z}_+$ and illustrated in Fig. 4(d), is modeled as a connected compact set $\mathcal{W}_g \subset \mathcal{W}$ whose enclosure is mutually disjoint with any runways or taxiways in the airport and, thus, when a taxiway a connects to a terminal region g it follows that $\mathcal{W}_a \cap \mathcal{W}_g = \partial\mathcal{W}_a \cap \partial\mathcal{W}_g$. Let a line segment h_T denote the smooth curve that connects the centerline of taxiway h_a to the terminal region $\mathcal{W}_g$, for the pair $T = (a, g)$, where $h_T : \mathbb{R}^2 \to \mathbb{R}$ is a continuous, differentiable function that obeys following properties:

1) $C_3 = \{\mathbf{q} \in \mathcal{W} \mid h_T(\mathbf{q}) = h_a(\mathbf{q})\}$ is a singleton set
2) $\left.\frac{\partial_x h_T}{\partial_y h_T}\right|_{\mathbf{q}\in C_3} = \left.\frac{\partial_x h_a}{\partial_y h_a}\right|_{\mathbf{q}\in C_3}$,
3) $\exists \mathbf{q} \in \mathcal{W}_G$ such that $h_T(\mathbf{q}) = 0$

The gate area is characterized by a yellow line segment painted on the pavement (Fig. 4(d)). Let the center position of the line segment represent the position of gate g, denoted by $\mathbf{q}_g \in \mathcal{W}_g$. Then, from the airport diagram (Fig. 2), the label sets of all taxiways, runways, and terminal gates in the airport, denoted by $\mathcal{L}_\mathbb{A}$, $\mathcal{L}_\mathbb{P}$, and $\mathcal{L}_\mathbb{G}$, respectively, and the sets of all corresponding regions, $\mathcal{W}_\mathbb{A}$, $\mathcal{W}_\mathbb{P}$, and $\mathcal{W}_\mathbb{G}$, can be extracted by simple image processing algorithms. As an example, the label sets of the Syracuse Hancock International Airport model based on Fig. 2 is shown in Table 1. Then, the finite set $H_\mathbb{D} = \{h_D \mid D = \{a, p\}, \forall a \in \mathcal{L}_\mathbb{A}, \forall p \in \mathcal{L}_\mathbb{P}\}$ represents the set of all hold-short positions in the airport. Similarly, $H_\mathbb{E}$ denotes the set of all runway and taxiway centerlines, and the set of curves connecting taxiways to taxiways, runways, and terminal gate regions is denoted by $H_\mathbb{C}$. Finally, the airport model M is defined as a tuple, $M = (\mathcal{L}_\mathbb{A}, \mathcal{L}_\mathbb{P}, \mathcal{L}_\mathbb{G}, \mathcal{W}_\mathbb{A}, \mathcal{W}_\mathbb{P}, \mathcal{W}_\mathbb{G}, H_\mathbb{E}, H_\mathbb{C}, H_\mathbb{D})$ that can be used to construct a topological graph of the airport to be used for path planning (Section IV).

Table 1 Label Sets for the Syracuse Hancock International Airport in Fig. 4

Label set	Elements
$\mathcal{L}_\mathbb{A}$ (Taxiways):	*A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, *J*, *K*, *L*, *M*, *N*, *P*, *Q*, *R*, *S*, *T*, *V*, *Y*
$\mathcal{L}_\mathbb{P}$ (Runways):	10, 28, 15, 33
$\mathcal{L}_\mathbb{G}$ (Gates):	1, 3, 4, 5, 6, 8, 9, 10, 11, 12, 14, 15, 20, 21, 22, 23, 24, 25, 26, 27

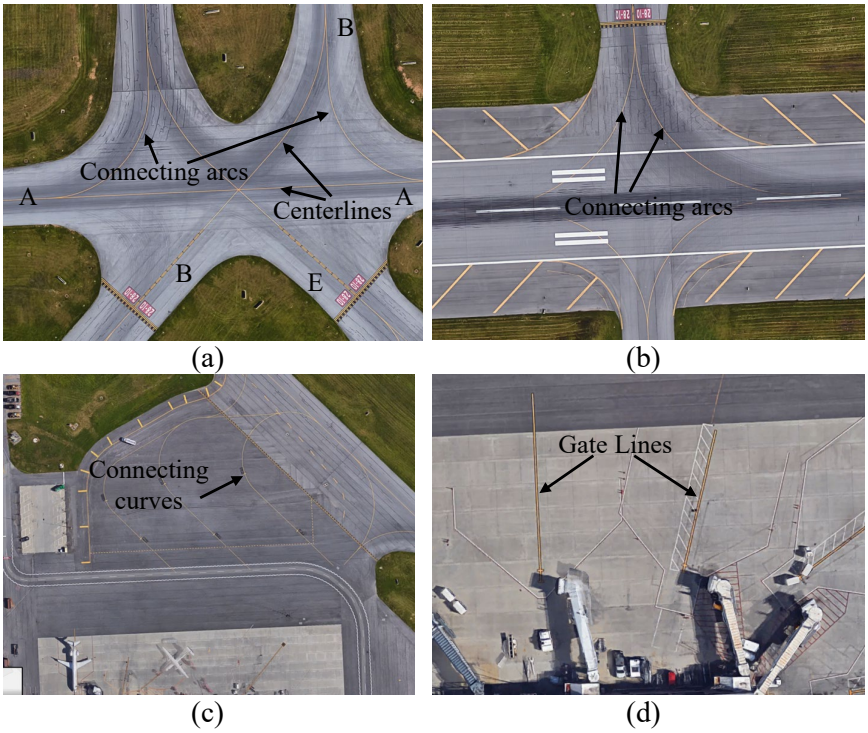

Figure 4. Satellite images of Syracuse Hancock International Airport from Google Map^{TM} showing (a) connecting curves between taxiways (taxiway labels shown); (b) taxiway-runway connecting curves; (c) taxiways and terminal-gate region connecting curves; (d) yellow gate lines at terminal gates.

B. Aircraft Motion Model

Several kinematic models have been proposed for modeling ground aircraft motion, in which the pilot can manipulate the nose gear steering wheel, throttle lever, and brake pedals to control the steering angle ϕ and linear acceleration β over time [7, 16–19]. Let $\mathbf{q}_e(k) = [x(k), y(k)]^T \in \mathcal{W}$ denote the x, y-coordinates of the aircraft rear-axle center in inertial frame, at a discrete time step indexed by k. Let $\theta(k)$ and $\upsilon(k)$ denote the aircraft yaw angle and speed, respectively. The aircraft ground state and control inputs can be defined as $\mathbf{s}(k) = [\mathbf{q}_e^T(k), \theta(k), \upsilon(k)]^T \in \mathcal{S}$ and $\mathbf{u}(k) = [\phi(k), \beta(k)]^T \in \mathcal{U}$, respectively, where $\mathcal{S}$ and $\mathcal{U}$ are the admissible state and control spaces. Then, the aircraft kinematics adopted in this paper can be modeled by the difference equation in the form,

$$\mathbf{s}(k+1) = \mathbf{f}(\mathbf{s}(k), \mathbf{u}(k)) \tag{4}$$

given here by the non-holonomic *simple car* model,

$$\mathbf{f}(\mathbf{s}(k), \mathbf{u}(k)) = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \\ \upsilon(k) \end{bmatrix} + \begin{bmatrix} \upsilon(k)\cos\theta(k) \\ \upsilon(k)\sin\theta(k) \\ \frac{\upsilon(k)}{L}\tan\phi(k) \\ \beta(k) \end{bmatrix} \Delta T \tag{5}$$

where L is the distance between the front and rear axles of the aircraft, and ΔT is the sampling interval [20]. For simplicity, in this paper it is assumed that the aircraft state is known from on-board GPS and inertial measurement unit (IMU). However, the approach could be easily extended to GPS-denied environment by including the use of visual SLAM algorithms [21].

C. Sensor Measurement Model

The aircraft is equipped with RGB-D cameras that provide on-board environmental perception in real time. Define the projection of camera's field of view (FOV) onto $\mathcal{W}$ as $\mathcal{F}(k) \subset \mathcal{W}$. By utilizing computer vision techniques, two

types of information are extracted from the camera for aircraft decision making: recognition of objects and a depth map encoding distances between the camera and objects in the camera's FOV. For simplicity, this paper focuses on recognizing three semantic classes of incursion objects, namely "People", "Animals", and "Ground Vehicles", where a "Ground Vehicle" here refers to all ground vehicles operating in the aerodrome except other aircraft. The distance between an object l and the aircraft, denoted by $d_l \geq 0$, can be obtained by on-board sensors, such as range finders, or by stereo-vision. Then, the camera measurement can be described by a nonlinear observation model,

$$\mathbf{z}(k) = [l(k), d_l(k)]^T, \quad k = 1, 2, \ldots \tag{6}$$

where the index $l(k)$ labels the semantic class of any object in the camera FOV. In this paper, $l(k) = 1, 2, 3$ if the detected object belongs to "People", "Animal", and "Ground Vehicle" class, respectively, and $l(k) = 0$ means no incursion object is detected. Every object detection is also accompanied by a classification performed by a CNN algorithm (III) and a corresponding confidence level.

D. ATC Commands Classification

The ATC verbal commands consist of concise and structured terminologies devised to ensure unambiguous and intelligible communication between the air traffic controllers and aircraft pilots, even under noisy backgrounds and mispronunciation [22]. To avoid communication ambiguity, ATC towers and aircraft pilots use a set of 26 English words to indicate taxiways, the initials of which correspond to the taxiway labels. A summary is shown in the Appendix. For tractability, the approach developed in this paper classifies ATC terminologies defined by the FAA into three categories $\{c_1, c_2, c_3\}$, corresponding to "Cruise", "Traffic Following", and "Holding" commands. Detailed descriptions and examples of verbal commands for the taxiing phase are summarized in Table 2. The approach can be easily extended to landing or takeoff phases, excluded here for simplicity.

Table 2 Classification of ATC Commands

Command Category c	Description	Examples [23]
c_1 = "Cruising"	Instruct to move along certain taxiways to a specified runway. Accelerate when needed.	• "Cross Runway One-Six Left and Runway One-Six Right at Taxiway Bravo." • "Runway Three-Six Left, taxi via Taxiway Alpha, hold short of Taxiway Charlie." • "Taxi without delay."
c_2 = "Traffic Following"	Instruct to follow traffic.	• "Follow (traffic), cross Runway Two-Seven Right, at Taxiway Whiskey."
c_3 = "Holding"	Instruct to hold short of a runway or hold in position on a runway.	• "Hold short of runway Two-Seven." • "Hold in position."

E. Hybrid System Modeling of Autonomous Taxiing

Hybrid system theory allows for the modeling and control of processes that include both discrete and continuous states and control inputs in order to optimize overall system performance. In this paper, a scalar discrete state variable ξ with finite range $\mathcal{E}$ is used to represent the autonomous taxiing system mode. A scalar discrete control variable with the same range, denoted by $\mu \in \mathcal{E}$, represents the decision on the next system mode. Then, the autonomous taxiing aircraft can be modeled as a discrete-time dynamical system,

$$\mathbf{s}(k+1) = \mathbf{f}_\xi[\mathbf{s}(k), \mathbf{u}_\xi(k)], \quad \xi(k+1) = \mu(k), \quad k = 1, 2, \ldots \tag{7}$$

where the continuous state kinematic equation, f_ξ, is shown in Eqn. 5. In the above model, $\mathbf{s} \in \mathcal{S}$ is the continuous state of the aircraft, $\mathbf{u}_\xi \in \mathcal{U}_\xi$ is the continuous control input, and $\mathcal{U}_\xi$ is the space of admissible control inputs for mode ξ. It is assumed that mode switching can occur immediately at any time step k and is determined solely by the discrete decision, μ. Also, the system state $\mathbf{s}$ and ξ are assumed fully observable and error free.

Letting $\Phi : \mathbb{R}^4 \times \mathbb{R}^2 \times \mathcal{E} \to \mathbb{R}$ denote an instantaneous cost function, and $\phi : \mathbb{R}^4 \to \mathbb{R}$ denote the terminal cost, the total system cost function over a fixed planning horizon $[k, k_f]$ is given by,

$$J = \phi(\mathbf{s}(k_f)) + \sum_{j=k}^{k_f-1} \Phi_\xi[\mathbf{s}(j), \mathbf{u}_\xi(j), \mu(j)] \tag{8}$$

and is to be minimized with respect to continuous and discrete control laws,

$$\mathbf{u}_\xi(j) = \pi_{\mathbf{u},\xi}(\mathbf{s}(j), j), \quad \text{and} \quad \mu(j) = \pi_\mu(\mathbf{s}(j), c(j), \mathbf{z}(j), \xi(j), j), \quad j = k, \ldots, k_f, \tag{9}$$

respectively, where $\xi \in \mathcal{E}$ and the command category $c(j) \in \{c_1, \ldots, c_3\}$ corresponds to one of three types of ATC commands, as defined in Table 2.

Five salient aircraft modes can be identified during taxiing operations, namely, "Cruising" (ξ_1), "Traffic Following" (ξ_2), "Holding" (ξ_3), "Incursion" (ξ_4), and "Idle" (ξ_5) modes, such that the discrete mode range is given by $\mathcal{E} = \{\xi_1, \ldots, \xi_5\}$. Mode switching is triggered by ATC commands or detection of incursion objects. In the "Cruise" mode, the aircraft follows the reference path that is generated by the high-level path planner by keeping track of the centerline of routes. When the ATC issues a "Holding" command, the aircraft switches to the "Holding" mode to decelerate to a full stop at the hold-short position. Once the speed reaches zero, the aircraft enters the "Idle" mode, and stays in this state until receiving further instructions from ATC. In many cases, the ATC will request the aircraft to accompany or follow another aircraft on a taxiway. In this case, the aircraft operates in the "Traffic Following" mode and maintains a similar speed as the aircraft in front. If incursion objects are detected, the aircraft immediately switches to the "Incursion" mode to avoid colliding with the object. The discrete control laws π_μ for all five modes are presented in Section V.

III. Object Detection and Recognition using Convolutional Neural Network

CNN algorithms have recently been shown to outperform sparse feature and other computer vision methods for applications in image and video processing and recognition [24]. A CNN is an artificial neural network architecture that employs multiple cascaded layers, where each layer is comprised of filters that have adjustable weights and biases. The CNN input consists of an image matrix from which a high-dimensional feature vector, referred to as *embedding* or convolutional feature vector, is generated as a compact (though high-dimensional) representation of the image. When combined with additional layers or classifiers such as support vector machines, CNN image embeddings are extremely effective at solving many computer vision tasks such as image classification [24–28], object detection [29–31], semantic segmentation [32, 33], and action recognition [34, 35]. A variety of CNN architectures have been proposed in the literature, including but not limited to AlexNet [24], VGG [25], and ResNet [26]. These and other CNN architectures share several core components that include convolutional layers, rectified linear unit, pooling layers, and fully connected layers, reviewed in this section and schematized in Fig. 5.

The convolutional layer, a core building block of CNNs, is comprised of K convolutional filters of size $F \times F$ and a bias term. Each layer uses a 3D volume X of size $n_{W1} \times n_{H1} \times n_{D1}$ as input, and outputs a 3D volume O of size $n_{W2} \times n_{H2} \times n_{D2}$. Here the dimension of volume is specified in the order of width, height, and depth. A convolutional filter has the same depth as the input volume. For example, a 256×256 RGB image is a 3D volume of size $256 \times 256 \times 3$, where the depth corresponds to three color channels. The size $F \times F$ is usually referred to as the *receptive field* of the filter. In order to control the spatial size of the output volume, the input volume is usually padded with P zero elements on the boarder, as shown in Fig. 6. Define S as the stride with which the filter is slid along the width-height 2-D slice. Given P and S, the input and output dimensions of the convolutional layer are related by the algebraic equations

$$n_{W2} = (n_{W1} - F + 2P)/S + 1 \tag{10a}$$

$$n_{H2} = (n_{H1} - F + 2P)/S + 1 \tag{10b}$$

$$n_{D2} = K \tag{10c}$$

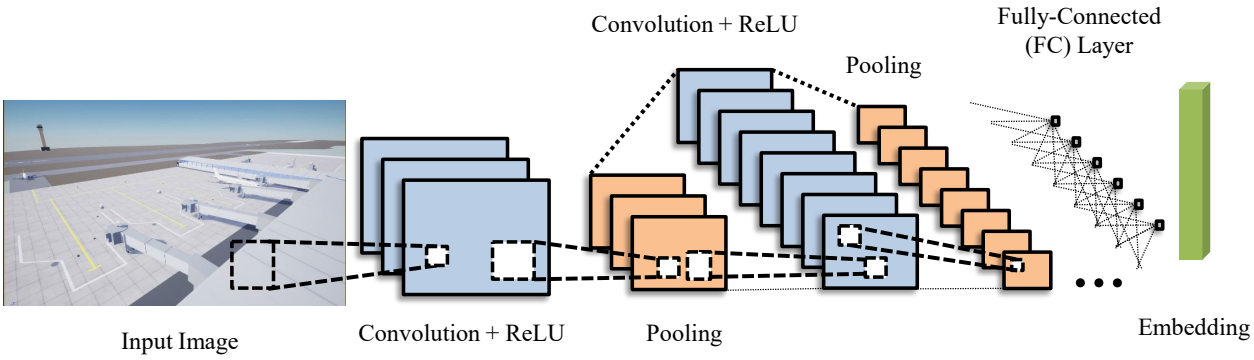

Figure 5. CNNs accept images as inputs and outputs feature vectors (embeddings). A CNN is usually composed of interleaving convolution layers, ReLU, and pooling layers. fully-connected Layers are appended at the end of the network.

The output volume of a convolutional layer is obtained via convolutional operations,

$$O(i,j,k) = \sum_{d=1}^{n_{D1}} \sum_{\iota=1}^{F} \sum_{\zeta=1}^{F} \omega_{k,d,\iota,\zeta}\, \tilde{X}\left(i(S-1)+\iota, j(S-1)+\zeta, d\right),\ k = 1,\ldots,K, \tag{11}$$

where $\tilde{X}$ is the volume that is extended from X by zero-padding, and $\omega_{k,d,\iota,\zeta}$ is the learnable weighting parameter of the (ι,ζ) element of the kth filter used in the dth layer convolution. To overcome difficulties associated with a large number of adjustable parameters, a strategy known as parameter sharing can be used to constrain the filters in each depth slice to use the same weights and bias, as follows:

$$\omega_{k,d,\iota,\zeta} = \omega_{k,d,\iota',\zeta'}, \quad \forall \iota,\iota' = 1,\ldots,F, \quad \forall \zeta,\zeta' = 1,\ldots,F$$

After the convolution operation, nonlinear activation functions are used to apply element-wise activation. The use of nonlinear activation functions enables CNNs to capture the nonlinear relationship between the input image and the output embedding. The rectified linear unit (ReLU) is commonly used as activation function in order to apply element-wise hinge operation, i.e. $g(x) = \max(0, x)$.

A common practice in CNNs is to periodically insert a pooling layer between successive convolutional layers to progressively reduce the spatial size of the representation and, therefore, the amount of parameters in the network. This improves computation efficiency and also gives an effective way for controlling overfitting. The pooling Layer operates on every slice along the depth dimension of the input and resizes it spatially, usually with the "max" operation that retains the local maximum input values. The depth dimension remains unchanged. Specifically, assume a max pooling filter with the receptive field size of $F \times F$ and the stride S, then the dimensions of the input and output volumes are determined by

$$\begin{aligned} n_{W2} &= (n_{W1} - F)/S + 1 \\ n_{H2} &= (n_{H1} - F)/S + 1 \\ n_{D2} &= n_{D1} \end{aligned}$$

After a series of convolutional and pooling operations, fully connected (FC) layers that have full connection between elements of the input and output volumes are used to generate extract an output feature vector. Multiple FC layers are usually used to improve learning performance and prevent underfitting. The output of the final FC layer is used as the embedding.

In this paper, the aircraft autonomous taxiing algorithm utilizes a Mask R-CNN [33], a state-of-the-art CNN-based object detector, for detecting and classifying incursion objects. The Mask R-CNN constructs and combines two networks, a region proposal network and a binary mask classifier using aforementioned layers, so that given an RGB image, a class label, bounding box, and segmentation mask can be generated for every object in the image, along with a corresponding confidence level. In general, training a CNN requires a large amount of data that may be difficult to obtain only from

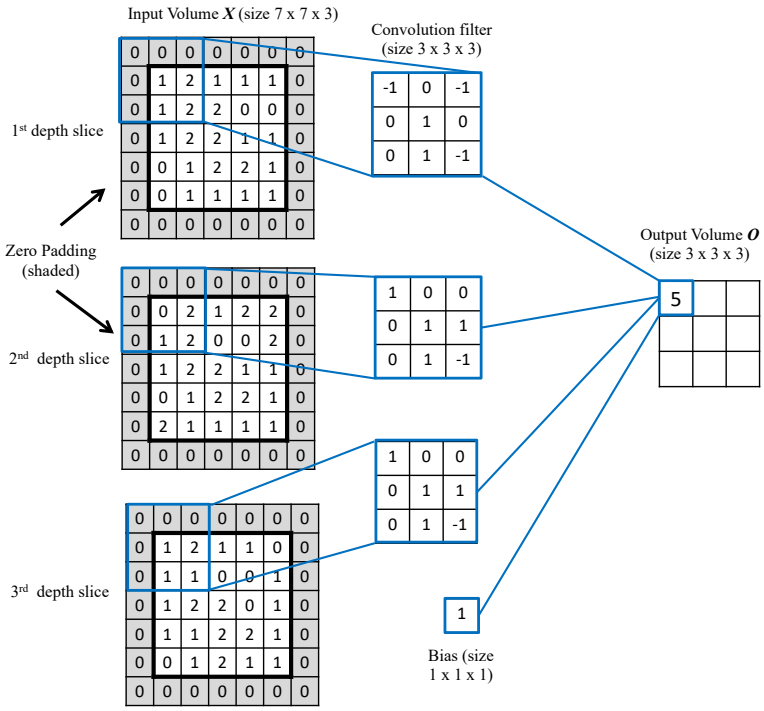

Figure 6. Example of convolutional operation. Since the input volume has depth $n_{D1} = 3$, the convolutional filter has depth 3. Zero padding is used, shown as the shared squares in the input volume. As only one filter is applied, the output depth is $n_{D2} = 1$.

airport environments. Therefore, in the proposed approach, Mask R-CNN is pre-trained on COCO dataset [36], which contains more than 200K labeled images and 80 classes of objects, and then overtrained as needed and applied to the airport environment. As shown in Fig. 7, animals, ground vehicles, and people in the UE simulated airport environment are detected and recognized with excellent accuracy.

IV. ATC-based Path Planning

This section presents a high-level path planning approach that translates ATC commands into a feasible aircraft reference path. The airport model M, developed in Section II.A, is first used to build a directed topological graph $\mathcal{G} = (\Xi, \mathcal{V})$, where Ξ is the directed edge set and $\mathcal{V}$ is the node set. A node $v \in \mathcal{V}$ is defined as tuple $v = (\alpha, \mathbf{q})$, where $\alpha \subset \mathcal{L}_{\mathrm{T}} \cup \mathcal{L}_{\mathrm{P}} \cup \mathcal{L}_{\mathrm{G}}$ is the node label and $\mathbf{q} \in \mathcal{W}_{\mathrm{T}} \cup \mathcal{W}_{\mathrm{P}} \cup \mathcal{W}_{\mathrm{G}}$ is the node position. Nodes represent one of the following regions: (1) connection of two regions (denoted by $\mathcal{V}_1$), including runways, taxiways, and the terminal region, (2) aircraft current position (denoted by $\mathcal{V}_2$), or (3) terminal gates (denoted by $\mathcal{V}_3$). In particular, the set $\mathcal{V}_1$ is defined

Figure 7. Examples of incursion objects simulated in UnrealEngine^{TM} (UE). Object detection and recognition results obtained by Mask R-CNN are shown along with semantic label, bounding box, and confidence level.

as the set of points where connecting curves intersect with centerlines or with the terminal region, i.e.:

$$\mathcal{V}_1 = \big\{(\gamma, \mathbf{q}) \mid \gamma = \{a_0, a_1, \ldots, a_l\}, \forall a_0, a_1, \ldots, a_l \in \mathcal{L}_{\mathbb{A}}, \text{ such that } h_{a_0}(\mathbf{q}) = 0,\ h_{A=\{a_0,a_i\}}(\mathbf{q}) = 0,\ \forall a_0 \neq a_i,\ i = 1, \ldots, l\big\}$$
$$\cup \big\{(\gamma, \mathbf{q}) \mid \gamma = \{a, p\},\ a \in \mathcal{L}_{\mathbb{A}},\ p \in \mathcal{L}_{\mathbb{P}},\ h_a(\mathbf{q}) = 0,\ h_{P=\{a,p\}}(\mathbf{q}) = 0\big\}$$
$$\cup \big\{(\gamma, \mathbf{q}) \mid \gamma = \{a, G_0\},\ a \in \mathcal{L}_{\mathbb{A}},\ h_a(\mathbf{q}) = 0,\ h_{T=\{a,G_0\}}(\mathbf{q}) = 0\big\}\,.$$

The first set refers to all intersection points where a taxiway intersects with a connecting curves that connect to other taxiways. The second set represents the intersection points where a taxiway connects to a connecting curve that leads to a runway. The third set is comprised of intersection points where a taxiway centerline is connected to a connecting curve that ends up in the terminal region at the other end

Given an aircraft position $\mathbf{q}_e$, the node set $\mathcal{V}_2$ is determined by the current region and position of aircraft,

$$\mathcal{V}_2 = \{(i, \mathbf{q}_e) \mid i \in \mathcal{L}_{\mathbb{A}} \cup \mathcal{L}_{\mathbb{P}} \cup \mathcal{L}_{\mathbb{G}},\ \text{ such that } \mathbf{q}_e \in \mathcal{W}_i\}\,.$$

and the node set $\mathcal{V}_3$ is comprised of terminal gates,

$$\mathcal{V}_3 = \big\{(g, \mathbf{q}_g) \mid g \in \mathcal{L}_{\mathbb{G}} \setminus \{0\}\big\}\,,$$

where $\setminus$ is the set minus operator. The entire node set of the topological graph is then given by $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$. The edge set Ξ encodes the connectivity of different regions and can be determined from airport diagram, such that

$$\Xi = \{\boldsymbol{\gamma}_1 \cap \boldsymbol{\gamma}_2 \mid \boldsymbol{\gamma}_1 \cap \boldsymbol{\gamma}_2 \neq \varnothing, \text{ and } \exists v_1,\ v_2 \in \mathcal{V} \text{ such that } v_1(1) = \boldsymbol{\gamma}_1, v_2(1) = \boldsymbol{\gamma}_2\}\,.$$

Then, the autonomous taxiing problem requires translating verbal ATC commands into an ordered sequence of nodes (or branch) in the airport topological graph $\mathcal{G}$ so as to plan a feasible path to the desired destination.

In the approach presented in this paper, ATC commands are first converted into a label sequence by a speech recognition algorithm, producing a goal destination in the topological graph (e.g. runway, taxiway, or terminal gate), as well as required intermediate taxiways that are dictated sequentially by the ATC operators to guide the aircraft on how to arrive at the goal destination, based on other airport traffic. Consider an ATC command sequence $\Psi = (\psi_1, \psi_2, \ldots, \psi_{n-1}, \psi_n)$, where $\psi_1, \ldots, \psi_{n-1} \in \mathcal{L}_{\mathbb{T}}$ and $\psi_n \in \mathcal{L}_{\mathbb{T}} \cup \mathcal{L}_{\mathbb{P}} \cup \mathcal{L}_{\mathbb{G}}$ correspond to the labels of intermediate taxiways and the goal destination, respectively. This command structure includes many common ATC commands for "Cruising" purpose. For example, the ATC command "Runway Two-eight, taxi via Taxiway Alpha and Golf", as provided in the FAA manual [23], can be converted into the ATC command sequence $\Psi = (A, G, 28)$. Subsequently, Algorithm 1 is used for generating an ATC command-consistent reference path $\mathbf{q}^*$, referred to as Path Generation from ATC Commands (PGATC). The algorithm starts from the aircraft's current position node (line 6), expands neighboring nodes (line 13), selects nodes that are consistent with ATC commands (line 15–22), and, then, repeats this procedure until all ATC commands are "translated" into a node sequence $\boldsymbol{\tau}^*$ (line 25). From $\boldsymbol{\tau}^*$, a waypoint sequence $\mathbf{q}^*$ is produced in aircraft configuration space ($\mathcal{W}$) to connect the positions of sequential nodes in $\boldsymbol{\tau}^*$ pairwise (line 30) by sampling centerline segments and connecting curves. The final waypoints coordinates are obtained using the airport diagram and prior geophysical information that is later communicated to the on-board IMU and GPS.

V. Hybrid Control of Autonomous Taxiing Aircraft

A hybrid aircraft control system is developed to coordinate the ground taxiing modes, such as "Cruising", "Traffic Following", "Holding", "Incursion", and "Idle", and to track the reference path $\mathbf{q}^*$ generated from ATC commands using PGATC, while keeping the aircraft safe based on visual feedback. The first three modes, i.e. "Cruising", "Traffic Following" and "Holding" are directly determined from the ATC commands (see Table 2). The mode "Incursion" is triggered by the CNN perception algorithm, when it recognizes incursion objects in the RGB frames obtained by the on-board aircraft camera. In this event, the controller drives the aircraft to stop before colliding with these objects. The "Idle" mode refers to an aircraft in a full stop, as due to holding at a hold-short position or to the detection of incursion objects. The hybrid continuous and discrete control laws for the autonomous aircraft taxiing system, shown in Eqns. 8 and 9, are described in the following subsections.

A. Continuous State Control Law

Centerline tracking strategy is used for controlling the continuous state of the aircraft. Since the reference path $\mathbf{q}^*$ is a sequence of waypoints, at kth planning step, waypoints within the on-board camera's FOV, denoted as $[(x_r(k), y_r(k))^T, \ldots, (x_r(k_f), y_r(k_f))^T] \in \mathcal{F}(k) \cap \mathbf{q}^*$, will be considered for the tracking control.

Algorithm 1 Path Generation from ATC Commands (PGATC)

Input: $\Psi = (\psi_1, \psi_2, \ldots, \psi_n)$, $\mathcal{G} = (\Xi, \mathcal{V})$, M, sample point number I.
Output: reference path $\mathbf{q}^*$.

function $\mathbf{q}^*$=PathFinder(Ψ, $\mathcal{G}$, M, I)
Get the node for the current position of the aircraft $v_s \in \mathcal{V}$.
Define $\iota(v)$ as the label of the region where v belongs, i.e. $\iota(v) = \{j | v \in \mathcal{W}_j,\ j \in \mathcal{L}_{\mathbb{A}} \cup \mathcal{L}_{\mathbb{P}} \cup \mathcal{L}_{\mathbb{G}}\}$
Initialize root set $\mathcal{R} = \{v_s\}$.
Initialize child set of each node as empty set, i.e. $\mathcal{D}(v) = \varnothing$, $\forall v \in V$
Initialize an empty path $\mathbf{q}^*$.
while $\Psi \neq \varnothing$ **do**
  Get the next ATC command ψ from Ψ, remove ψ from Ψ
  while $\mathcal{R} \neq \varnothing$ **do**
    Read a node v from $\mathcal{R}$, remove v from $\mathcal{R}$
    Define the set of neighboring nodes of v: $\mathcal{N}_v = \{v' | (v, v') \in \Xi\}$
    Initialize a new root set $\mathcal{R}' = \varnothing$
    for $v' \in \mathcal{N}_v$ **do**
      if $\psi = \iota(v')$ **then**
        $\mathcal{R}' = \mathcal{R}' \cup \{v'\}$, $\mathcal{D}(v) = \mathcal{D}(v) \cup \{v'\}$.
      else if $\iota(v) = \iota(v')$ **then**
        $(\mathcal{D}(v'), v^*)$ = SubPathFinder($\mathcal{G}$, M, ψ, v')
        $\mathcal{D}(v) = \mathcal{D}(v) \cup \{v'\} \cup \mathcal{D}(v')$, $\mathcal{R}' = \mathcal{R}' \cup \{v'\}$
      end if
    end for
  end while
  $\mathcal{R} = \mathcal{R}'$
end while
Compute a node sequence $\tau^* = [v_1, \ldots, v'_n]$, where $v_1 = v_s$ and $v_i \in \mathcal{D}(v_{i-1})$, $i = 1, \ldots, n'$.
for $j = 1 : n' - 1$ **do**
  Define $\mathbf{q}_j = v_j(2)$, $\mathbf{q}_{j+1} = v_{j+1}(2)$
  Find $h \in H_{\mathbb{E}}$ such that $h(\mathbf{q}_j) = 0$, $h(\mathbf{q}_{j+1}) = 0$.
  Sample I points on h, i.e., $\mathbf{q}' = \{\mathbf{q}_{j_1}, \ldots, \mathbf{q}_{j_I}\}$, with $\mathbf{q}_{j_1} = \mathbf{q}_j$ and $\mathbf{q}_{j_I} = \mathbf{q}_{j+1}$.
  Append $\mathbf{q}'$ to the end of $\mathbf{q}^*$.
end for
return $\mathbf{q}^*$.
end function

function $(\mathcal{D}, v^*)$ = SubPathFinder($\mathcal{G}$, M, ψ, v)
for $v' \in \mathcal{N}_v = \{v' | (v, v') \in \Xi\}$ **do**
  if $\psi == \iota(v')$ **then**
    return $(\{v'\}, v^*)$
  end if
  if $\psi \neq \iota(v')$ **then**
    return $\varnothing$
  end if
  if $i'_v == i_v$ **then**
    $(\mathcal{D}, v^*) = SubPathFinder(\mathcal{G}, M, \psi, v')$
    if $\mathcal{D} == \varnothing$ **then**
      continue
    else if $\mathcal{D} \neq \varnothing$ **then**
      $\mathcal{D} = \{v'\} \cup \mathcal{D}$
      return $(\mathcal{D}, v^*)$
    end if
  end if
end for
return $\varnothing$
end function

"Cruising" mode. The aircraft is controlled to follow the reference path and maintain a desired cruise speed v_c. Without loss of generality, it can be assumed that the waypoints coordinates and hold-short lines are all known in the inertial frame. Given a desired aircraft cruise speed v_c, the cruising objective function is defined as,

$$J_{\mu_1} = \|[x(k_f), y(k_f), v(k_f)]^T - [x_r(k_f), y_r(k_f), v_c]^T\|_2^2 + \sum_{j=k}^{k_f} \|[x(j), y(j), v(j)]^T - [x_r(j), y_r(j), v_c]^T\|_2^2 + \|\mathbf{u}_{\mu_1}(j)\|_2^2 \quad (13)$$

in order to penalize deviations from the centerline and control usage. Then, the optimal control problem to be solved at time step k can be stated as follows,

$$\begin{aligned} \min_{\mathbf{x}_1} \ & J_{\mu_1} \\ \text{subject to } & \mathbf{s}(k) = \mathbf{s}_0 \\ & \mathbf{s}(j+1) = \mathbf{f}(\mathbf{s}(j), \mathbf{u}_{\mu_1}(j)),\ j = k, \ldots, k_f - 1 \\ & \mathbf{s}(j) \in \mathcal{S},\ j = k, \ldots, k_f - 1 \\ & \mathbf{u}_{\mu_1}(j) \in \mathcal{U}_{\mu_1},\ j = k, \ldots, k_f - 1 \end{aligned}$$

where $\mathbf{s}_0$ is the known state of the aircraft at time k, and the optimization variables are lumped into the vector

$$\mathbf{x}_1 \triangleq \left[\mathbf{s}^T(k), \ldots, \mathbf{s}^T(k_f), \mathbf{u}_{\mu_1}^T(k), \ldots, \mathbf{u}_{\mu_1}^T(k_f - 1)\right]^T$$

"Traffic Following" mode. In this mode, the aircraft motion is similar to the "Cruise" mode except that it also requires the aircraft to maintain a speed similar to that of the aircraft in front, v_f, which may be easily estimated by on-board sensors. Therefore, the traffic-following objective function can be defined as,

$$J_{\mu_2} = \|[x(k_f), y(k_f), v(k_f)]^T - [x_r(k_f), y_r(k_f), v_f]^T\|_2^2 + \sum_{j=k}^{k_f-1} \|[x(j), y(j), v(j)]^T - [x_r(j), y_r(j), v_f]^T\|_2^2 + \|\mathbf{u}_{\mu_2}(j)\|_2^2 \quad (14)$$

and the optimization problem to be solved at time step k can be stated as follows,

$$\begin{aligned} \min_{\mathbf{x}_2} \ & J\mu_2 \\ \text{subject to } & \mathbf{s}(k) = \mathbf{s}_0 \\ & \mathbf{s}(j+1) = \mathbf{f}(\mathbf{s}(j), \mathbf{u}_{\mu_2}(j)),\ j = k, \ldots, k_f - 1 \\ & \mathbf{s}(j) \in \mathcal{S},\ j = k, \ldots, k_f - 1 \\ & \mathbf{u}_{\mu_2}(j) \in \mathcal{U}_{\mu_2},\ j = k, \ldots, k_f - 1 \end{aligned}$$

where

$$\mathbf{x}_2 \triangleq \left[\mathbf{s}^T(k), \ldots, \mathbf{s}^T(k_f), \mathbf{u}_{\mu_2}^T(k), \ldots, \mathbf{u}_{\mu_2}^T(k_f - 1)\right]^T$$

"Holding" mode. In this mode, the aircraft starts decelerating to stop at the hold-short position. Let the waypoint for the incoming hold-short position be equal to $[x_r(k_f) \quad y_r(k_f)]^T$. Then, the holding objective function is defined as,

$$J_{\mu_3} = \|[x(k_f), y(k_f), v(k_f)]^T - [x_r(k_f), y_r(k_f), 0]^T\|_2^2 + \sum_{j=1}^{k_f-1} \|[x(j), y(j), v(j)]^T - [x_r(j), y_r(j), 0]^T\|_2^2 + \|\mathbf{u}_{\mu_3}(j)\|_2^2 \quad (15)$$

and the optimization problem to be solved at time step k can be stated as follows,

$$\begin{aligned} \min_{\mathbf{x}_3} \ & J\mu_3 \\ \text{subject to } & \mathbf{s}(k) = \mathbf{s}_0 \\ & \mathbf{s}(j+1) = \mathbf{f}(\mathbf{s}(j), \mathbf{u}_{\mu_3}(j)),\ j = k, \ldots, k_f - 1 \\ & \mathbf{s}(j) \in \mathcal{S},\ j = k, \ldots, k_f - 1 \\ & \mathbf{u}_{\mu_3}(j) \in \mathcal{U}_{\mu_3},\ j = k, \ldots, k_f - 1 \end{aligned}$$

and the optimization variables are

$$\mathbf{x}_3 \triangleq \left[\mathbf{s}^T(k), \ldots, \mathbf{s}^T(k_f), \mathbf{u}_{\mu_3}^T(k), \ldots, \mathbf{u}_{\mu_3}^T(k_f - 1)\right]^T$$

"Incursion" mode. When an aircraft detects an incursion object on its route, the aircraft enters the "Incursion" mode and decelerates to avoid colliding with the object. Let $d_s > 0$ represent a user-specified safety margin between the aircraft and the object. Let the vector $[x_o \quad y_o]^T \in \mathcal{W}$ represent the coordinates of the incursion object estimated from on-board depth measurement that are integrated with the on-board camera feedback. Then, the incursion objective function is defined as,

$$J_{\mu_4} = \|[x(k_f), y(k_f), v(k_f)]^T - [x_r(k_f), y_r(k_f), 0]^T\|_2^2 + \sum_{j=1}^{k_f-1} \|[x(j), y(j), v(j)]^T - [x_r(j), y_r(j), 0]^T\|_2^2 + \|\mathbf{u}_{\mu_4}(j)\|_2^2 \quad (16)$$

and the optimization problem to be solved at time step k can be stated as follows,

$$\begin{aligned} \min_{\mathbf{x}_4} \ & J\mu_4 \\ \text{subject to} \ & \mathbf{s}(k) = \mathbf{s}_0 \\ & \mathbf{s}(j+1) = \mathbf{f}(\mathbf{s}(j), \mathbf{u}_{\mu_4}(j)), \ j = k, \ldots, k_f - 1 \\ & \mathbf{s}(j) \in \mathcal{S}, \ j = k, \ldots, k_f - 1 \\ & \mathbf{u}_{\mu_4}(j) \in \mathcal{U}_{\mu_4}, \ j = k, \ldots, k_f - 1 \\ & \|[x(j), y(j)]^T - [x_o, y_o]^T\|_2 \geq d_s, \ \forall j = k, \ldots, k_f - 1 \end{aligned} \quad (17a)$$

and the optimization variables are

$$\mathbf{x}_3 \triangleq \left[\mathbf{s}^T(k), \ldots, \mathbf{s}^T(k_f), \mathbf{u}_{\mu_4}^T(k), \ldots, \mathbf{u}_{\mu_4}^T(k_f - 1)\right]^T$$

The constraint in Eqn. 17a is used to ensure that the aircraft does not collide with the incursion object.

"Idle" mode. The aircraft starts with the "Idle" mode. It re-enters this mode again whenever its speed drops to zero, waiting for further commands from ATC tower. The continuous controller in this mode is trivial, i.e.

$$\mathbf{u}_{\mu_5} = \mathbf{0}.$$

B. Mode Switching Control Law

"Cruising" mode. The discrete controller $\mu(k)$ decides when to switch the system state from "Cruising" to other modes if the ATC tower instructs the aircraft to follow traffic or hold short, or if incursion objects are detected. In fact, when the aircraft approaches a holding short position (usually at the entrance to a runway), the ATC tower will issue a command to instruct the aircraft to hold short. Upon receiving the command, the aircraft enters "Holding" state, where the aircraft starts decelerating to a full stop at the hold-short position. If the ATC tower instructs the aircraft to follow the traffic flow, the aircraft switches to the "Traffic Following" mode. Once an incursion object is detected, i.e. $l(k) \in \{1, 2, 3\}$, the system immediately enters the "Incursion" mode to avoid collision. The discrete control is therefore defined as

$$\mu(k) = \begin{cases} \mu_4 & \text{if } l(k) \in \{1, 2, 3\} \\ \mu_2 & \text{if } l(k) = 0 \text{ and } c(k) = c_2 \\ \mu_3 & \text{if } l(k) = 0 \text{ and } c(k) = c_3 \\ \mu_1 & \text{otherwise} \end{cases}. \quad (18)$$

"Traffic Following" Mode. The discrete controller switches the system state out of "Traffic Following" mode if the ATC tower issues a "Cruising" or "Holding" command, or when incursion objects are detected. Besides, when the front aircraft stops, the aircraft stops as well and switches to the "Idle" state after the speed reduces to zero. Consequently, the discrete control law is defined as

$$\mu(k) = \begin{cases} \mu_4 & \text{if } l(k) \in \{1, 2, 3\} \\ \mu_1 & \text{if } l(k) = 0 \text{ and } c(k) = c_1 \\ \mu_3 & \text{if } l(k) = 0 \text{ and } c(k) = c_3 \\ \mu_5 & \text{if } l(k) = 0 \text{ and } v(k) = 0 \\ \mu_2 & \text{otherwise} \end{cases}. \quad (19)$$

"Holding" Mode. In "Holding" mode, the aircraft decelerates to stop at the hold-short position identified by the ground markings. The mode is switched to "Idle" once the aircraft speed reduces to 0. If the ATC tower issues "Cruise" or "Traffic Following" commands, the aircraft switches to the corresponding state. The system enters the "Incursion" mode once incursion objects are detected, and implements the discrete control law

$$\mu(k) = \begin{cases} \mu_4 & \text{if } l(k) \in \{1,2,3\} \\ \mu_1 & \text{if } l(k) = 0 \text{ and } c(k) = c_1 \\ \mu_2 & \text{if } l(k) = 0 \text{ and } c(k) = c_2 \\ \mu_5 & \text{if } l(k) = 0 \text{ and } v(k) = 0 \\ \mu_3 & \text{otherwise} \end{cases}. \tag{20}$$

"Incursion" Mode. Once entering the "Incursion" Mode, the aircraft will stay in this mode until the speed drops to 0, when the system switches into the "Idle" mode. If an ATC command is issued in this process, the system will enter the corresponding mode. The discrete control law is defined as:

$$\mu(k) = \begin{cases} \mu_1 & \text{if } c(k) = c_1 \\ \mu_2 & \text{if } c(k) = c_2 \\ \mu_3 & \text{if } c(k) = c_3 \\ \mu_5 & \text{if } v(k) = 0 \\ \mu_4 & \text{otherwise} \end{cases}, \tag{21}$$

"Idle" Mode. The discrete controller decides to switch the system into other modes if the ATC tower instructs the aircraft to cruise or follow traffic. The discrete control law is defined as

$$\mu(k) = \begin{cases} \mu_1 & \text{if } c(k) = c_1 \\ \mu_2 & \text{if } c(k) = c_2 \\ \mu_5 & \text{otherwise} \end{cases}. \tag{22}$$

VI. Simulation Results

The effectiveness of the autonomous taxiing approach presented in this paper is demonstrated by conducting simulations in an airport environment generated, observed, and controlled using the UnrealEngine^{TM} (Fig. 1), which consists of six terminal gates $(1, \ldots, 6)$, three taxiways ("A", "B", "C"), and a runway with labels "02" and "20". An illustrative airport map with centerlines, connecting curves, and nodes is shown in Fig. 8. A commercial aircraft model is used with front-rear axle distance $L = 15m$, acceleration $\beta \in [-10m/s^2, 10m/s^2]$, and yaw rate $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. The maximum speed is assumed to be $10m/s$. The on-board RGB-D camera used for environmental perception is characterized by an angle-of-view equal to 90°, and a maximum detection range equal to 20 (m). A planning horizon of 10 steps is used for controlling the aircraft, with a sampling interval equal to $\Delta T = 0.3$ (sec). The optimization problems presented in the previous sections are all solved by the open-source nonlinear optimization solver "Ipopt" [37].

In the first simulation study, shown in Fig. 10, a normal taxiing situation preceding takeoff is considered. At the onset of the simulation, the ATC tower issues the command "Runway Two-Zero, taxi via Taxiway Alpha and Bravo" and the reference path is generated by Algorithm 1. Based on the airport graph, shown in part in Fig. 9, the reference path connecting these nodes is,

$$v_s \rightarrow v_8 \rightarrow v_9 \rightarrow v_{13} \rightarrow v_{14} \rightarrow v_{15}$$

where v_s corresponds to the aircraft initial position. Given the reference path, the aircraft first enters the "Cruising" mode (Fig. 10(a)) to taxi along the reference path. As the aircraft approaches the entrance to the runway, ATC issues the command for holding short (Fig. 10(b)), causing the aircraft to switch to the "Holding" mode, and to decelerate to stop before the hold-short line, where it enters the "Idle" mode. After the ATC tower issues a clearance command ("Clear for Runway Two-Zero"), the aircraft enters the "Cruising" mode (Fig. 10(c)) to taxi onto the runway (Fig. 10(d)). The aircraft actual trajectory is compared to the centerline in Fig. 11(a), and the aircraft control inputs and speed are plotted in Fig. 11(b)-(c). It can be observed that the aircraft successfully tracks the centerline. As shown in Fig. 11(c), the aircraft velocity drops to 0 at $k = 100$, which shows that the aircraft stops at the hold-short position before entering the runway, as instructed by the ATC tower.

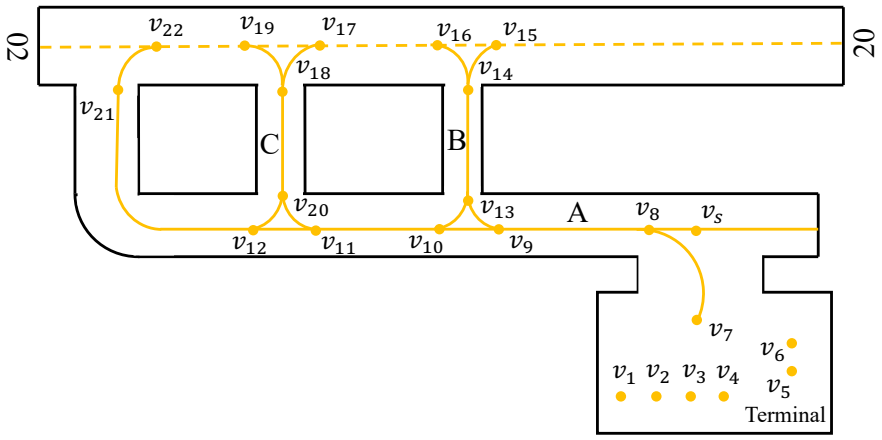

Figure 8. Illustration of the simulation airport. Orange circles represent all nodes in the airport graph. Orange lines represent taxiway centerlines and dashed lines represent the runway centerline. Orange curves show the connecting curves between different regions.

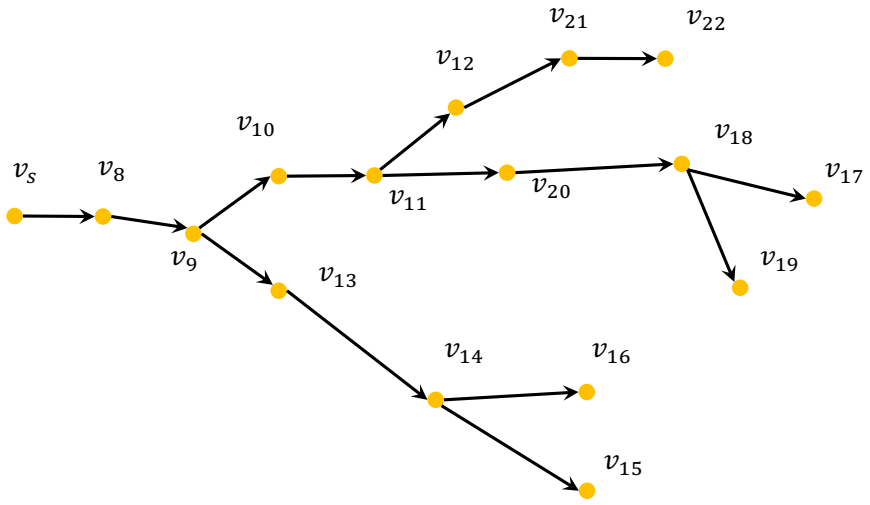

Figure 9. A generated airport graph. For clarity, only part of the whole graph is shown by removing all directed edges from runway nodes to other nodes and the edges thereafter.

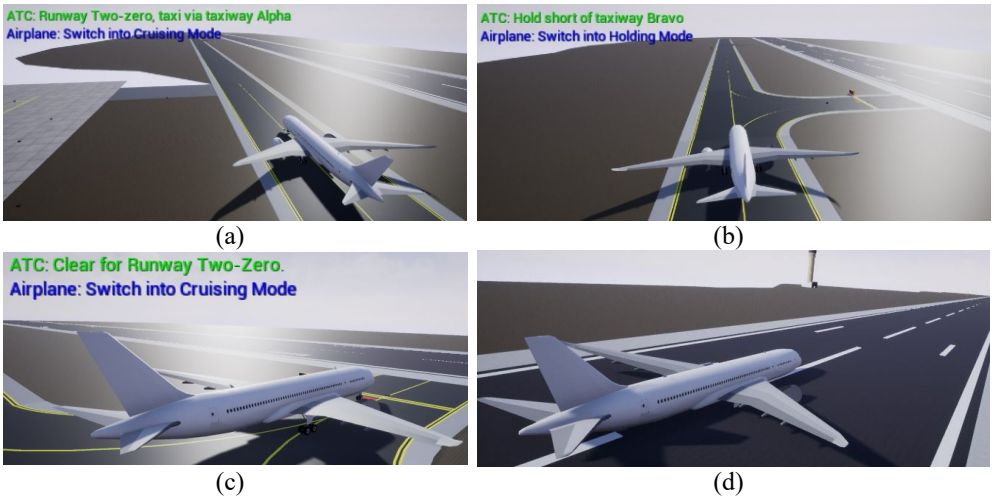

Figure 10. Simulated autonomous taxiing under normal conditions. ATC commands and mode switch of the aircraft is shown on the top left corner.

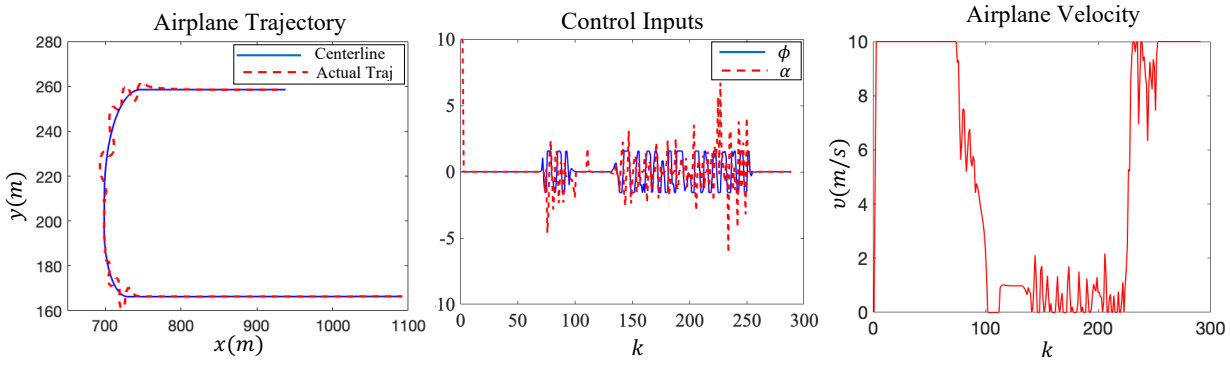

Figure 11. The aircraft trajectory, control inputs, and velocity profile, under normal taxiing conditions.

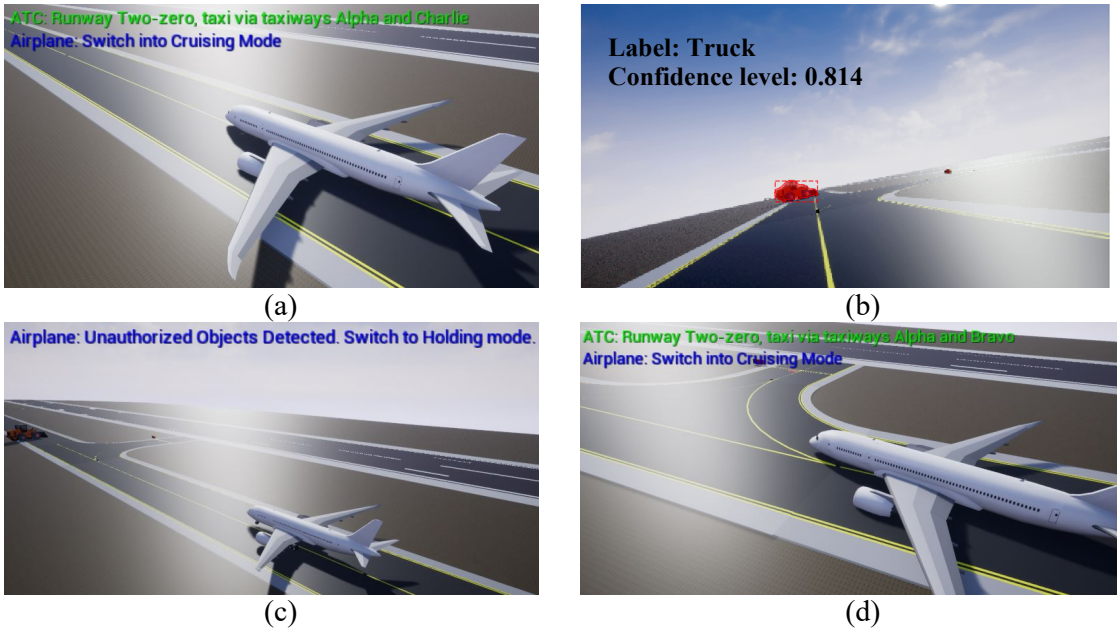

Figure 12. Simulated autonomous taxiing involving object incursion. (a) The aircraft starts taxiing. (b) The detected target by the on-board camera. Both label and confidence levels are provided. (c) When the aircraft detects the incursion object, it reports to ATC tower and switches to "Holding" mode. (d) After the ATC tower instructs to change route, the aircraft generates a new path and begins cruising.

The second simulation study considers incursion events during the aircraft taxiing to the runway for takeoff, as shown in Fig. 12. Based on the ATC command "Runway Two-Zero, taxi via Taxiway Alpha and Charlie" (Fig. 12(a)), the reference path is generated by Algorithm 1. The aircraft enters the "Cruising" mode to taxi along the reference path. As soon as the aircraft detects an incursion object (a truck) in its camera FOV (Fig. 12(b)), it switches into the "Incursion" mode and decelerates to stop (Fig. 12(c)). After the ATC tower issues a new command ("Runway Two-Zero, taxi via Taxiways Alpha and Bravo."), the aircraft re-generates its reference path and switches to "Cruising" mode (Fig. 12(d)). The aircraft then follows the new path to taxi to the runway. The aircraft actual trajectory is compared to the centerline in Fig. 13(a), and the aircraft control inputs and speed are plotted in Figs. 13(b)-(c). It can be seen that its velocity drops to zero around times $k = 130$ and $k = 210$, which correspond to the times at which the aircraft comes to a full stop after detecting the incursion object and upon arrival at the hold-short position before entering the runway, respectively.

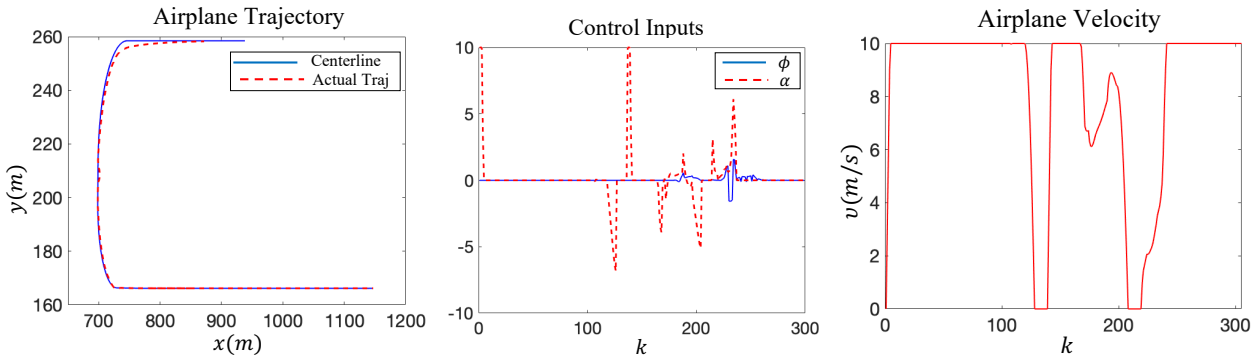

Figure 13. The aircraft trajectory, control inputs, and velocity profile, under incursion conditions.

VII. Conclusion

This paper develops a systematic autonomous taxiing approach based on vision-guided hybrid planning and control and convolutional neural networks. The approach integrates real-time aircraft perception, obstacle avoidance, and feedback control theory with ATC commands in the loop. The Mask R-CNN is utilized for detecting and recognizing incursion objects that can cause dangerous situations in airports. A graph-based path planning approach is proposed to generate reference paths based on verbal ATC commands. A hybrid system controller is developed to track taxiway/runway centerlines under normal conditions to complete taxiing tasks. When incursion objects are detected, the controller drives the aircraft to a full stop to ensure safety and, then, re-generates a safe path based on new ATC commands. The effectiveness of this autonomous taxiing approach is demonstrated by conducting simulations in an airport environment generated, observed, and controlled using the UnrealEngine^{TM} during both normal and unforeseen airdrome conditions.

Appendix

Table 3 Aviation Phonetic Alphabet [38]

English	Phonetic	English	Phonetic	English	Phonetic
A	Alpha	J	Juliet	S	Sierra
B	Bravo	K	Kilo	T	Tango
C	Charlie	L	Lima	U	Uniform
D	Delta	M	Mike	V	Victor
E	Echo	N	November	W	Whiskey
F	Foxtrot	O	Oscar	X	Xray
G	Golf	P	Papa	Y	Yankee
H	Hotel	Q	Quebec	Z	Zulu
I	India	R	Romeo		

Acknowledgments

This work is supported in part by the Office of Naval Research under award N00014-17-1-2175.

References

[1] Chen, J., Weiszer, M., Stewart, P., and Shabani, M., "Toward a More Realistic, Cost-Effective, and Greener Ground Movement Through Active Routing—Part I: Optimal Speed Profile Generation," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 5, 2016, pp. 1196–1209.

[2] Morris, R., Pasareanu, C. S., Luckow, K. S., Malik, W., Ma, H., Kumar, T. S., and Koenig, S., "Planning, Scheduling and Monitoring for Airport Surface Operations." *AAAI Workshop: Planning for Hybrid Systems*, 2016.

[3] Ferrari, S., and Vaghi, A., "Demining sensor modeling and feature-level fusion by Bayesian networks," *IEEE Sensors Journal*, Vol. 6, No. 2, 2006, pp. 471–483.

[4] Albertson, J. D., Harvey, T., Foderaro, G., Zhu, P., Zhou, X., Ferrari, S., Amin, M. S., Modrak, M., Brantley, H., and Thoma, E. D., "A mobile sensing approach for regional surveillance of fugitive methane emissions in oil and gas production," *Environmental science & technology*, Vol. 50, No. 5, 2016, pp. 2487–2497.

[5] Foderaro, G., Zhu, P., Wei, H., Wettergren, T. A., and Ferrari, S., "Distributed optimal control of sensor networks for dynamic target tracking," *IEEE Transactions on Control of Network Systems*, Vol. 5, No. 1, 2018, pp. 142–153.

[6] Wei, H., Zhu, P., Liu, M., How, J. P., and Ferrari, S., "Automatic Pan-tilt Camera Control for Learning Dirichlet Process Gaussian Process (DPGP) Mixture Models of Multiple Moving Targets," *IEEE Transactions on Automatic Control*, 2018.

[7] McGee, T. G., and Hedrick, J. K., "Optimal path planning with a kinematic airplane model," *Journal of guidance, control, and dynamics*, Vol. 30, No. 2, 2007, pp. 629–633.

[8] Lu, B., Coombes, M., Li, B., and Chen, W.-H., "Improved situation awareness for autonomous taxiing through self-learning," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 12, 2016, pp. 3553–3564.

[9] Administration, F. A., "Runway Incursion," `https://www.faa.gov/airports/runway_safety/news/runway_incursions/`, 2018.

[10] UnrealEngine, "UnrealEngine," `https://www.unrealengine.com/en-US/what-is-unreal-engine-4`, 2018.

[11] Lu, W., Zhang, G., and Ferrari, S., "A randomized hybrid system approach to coordinated robotic sensor planning," *Decision and Control (CDC), 2010 49th IEEE Conference on*, IEEE, 2010, pp. 3857–3864.

[12] Lu, W., Zhu, P., and Ferrari, S., "A hybrid-adaptive dynamic programming approach for the model-free control of nonlinear switched systems," *IEEE Transactions on Automatic Control*, Vol. 61, No. 10, 2016, pp. 3203–3208.

[13] Administration, F. A., "Aiport Diagram," `https://www.faa.gov/air_traffic/flight_info/aeronav/digital_products/dtpp/search/results`, 2018.

[14] Rudin, W., et al., *Principles of mathematical analysis*, Vol. 3, McGraw-hill New York, 1976.

[15] Administration, F. A., "A Quick Reference to Airfield Standards," `https://www.faa.gov/airports/southern/airport_safety/part139_cert/media/aso-airfield-standards-quick-reference.pdf`, 2018.

[16] Bursík, J., Kraus, J., and Štumper, M., "Automation of Taxiing," *MAD-Magazine of Aviation Development*, Vol. 5, No. 1, 2017.

[17] Menon, P., Sweriduk, G., and Sridhar, B., "Optimal strategies for free-flight air traffic conflict resolution," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 2, 1999, pp. 202–211.

[18] Bicchi, A., and Pallottino, L., "On optimal cooperative conflict resolution for air traffic management systems," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 4, 2000, pp. 221–231.

[19] Chitsaz, H., and LaValle, S. M., "Time-optimal paths for a Dubins airplane," *Decision and Control, 2007 46th IEEE Conference on*, IEEE, 2007, pp. 2379–2384.

[20] LaValle, S. M., *Planning algorithms*, Cambridge university press, 2006.

[21] Kerl, C., Sturm, J., and Cremers, D., "Dense visual SLAM for RGB-D cameras," *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Citeseer, 2013, pp. 2100–2106.

[22] Hopkin, V. D., *Human factors in air traffic control*, London: CRC Press, 1995.

[23] Administration, F. A., "FAA Order 7110.65X Air Traffic Control," 2017. URL `https://www.faa.gov/regulations_policies/orders_notices/index.cfm/go/document.current/documentNumber/7110.65`.

[24] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[25] Simonyan, K., and Zisserman, A., "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[26] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A., "Inception-v4, inception-resnet and the impact of residual connections on learning." *AAAI*, Vol. 4, 2017, p. 12.

[28] Zhu, P., Isaacs, J., Fu, B., and Ferrari, S., "Deep learning feature extraction for target recognition and classification in underwater sonar images," *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, IEEE, 2017, pp. 2724–2731.

[29] Ren, S., He, K., Girshick, R., and Sun, J., "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 39, No. 6, 2017, pp. 1137–1149.

[30] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C., "SSD: Single Shot MultiBox Detector," *ECCV*, 2016.

[31] Redmon, J., and Farhadi, A., "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

[32] Long, J., Shelhamer, E., and Darrell, T., "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[33] He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask r-cnn," *Computer Vision (ICCV), 2017 IEEE International Conference on*, IEEE, 2017, pp. 2980–2988.

[34] Ji, S., Xu, W., Yang, M., and Yu, K., "3D convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 35, No. 1, 2013, pp. 221–231.

[35] Simonyan, K., and Zisserman, A., "Two-stream convolutional networks for action recognition in videos," *Advances in neural information processing systems*, 2014, pp. 568–576.

[36] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., "Microsoft coco: Common objects in context," *European conference on computer vision*, Springer, 2014, pp. 740–755.

[37] Wächter, A., and Biegler, L. T., "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, Vol. 106, No. 1, 2006, pp. 25–57.

[38] Organization, I. C. A., "Aviation Phonetic Alphabet," 2018.