Transportation Research Part C 137 (2022) 103590

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A computer vision framework using Convolutional Neural Networks for airport-airside surveillance

Phat Thai [a,*], Sameer Alam [a], Nimrod Lilith [a], Binh T. Nguyen [b]

[a] *Saab-NTU Joint Lab, Nanyang Technological University, Singapore*
[b] *AISIA Research Lab, University of Science, Vietnam National University, Ho Chi Minh City, Viet Nam*

ARTICLE INFO

Keywords:
Air traffic control
Convolutional Neural Network
Computer vision

ABSTRACT

Modern airports often have large and complex airside environments featuring multiple runways, with changing configurations, numerous taxiways for effective circulation of flights and tens, if not hundreds, of gates. With inherent uncertainties in gate push-back and taxiway routing, efficient surveillance and management of airport-airside operations is a highly challenging task for air traffic controllers. An increase in air traffic may lead to gate delays, taxiway congestion, taxiway incursions as well as significant increase in the workload of air traffic controllers. With the advent of Digital Towers, airports are increasingly being equipped with surveillance camera systems. This paper proposes a novel computer vision framework for airport-airside surveillance, using cameras to monitor ground movement objects for safety enhancement and operational efficiency improvement. The framework adopts Convolutional Neural Networks and camera calibration techniques for aircraft detection and tracking, push-back prediction, and maneuvering monitoring. The proposed framework is applied on video camera feeds from Houston Airport, USA (for maneuvering monitoring) and Obihiro Airport, Japan (for push-back prediction). The object detection models of the proposed framework achieve up to 73.36% average precision on Houston airport and 87.3% on Obihiro airport. The framework estimates aircraft speed and distance with low error (up to 6 meters), and aircraft push-back is predicted with an average error of 3 min from the time an aircraft arrives with the error-rate reducing until the aircraft's actual push-back event.

1. Introduction

To accommodate the steady growth of air traffic, airports worldwide need to increase their capacity by expanding infrastructure and optimizing air traffic procedures. However, as airports add new infrastructure (such as runways and taxiways) into operation, it increases the challenge of aircraft ground movement control and monitoring. According to EUROCONTROL, there was an average 13.6 min delay per flight in Europe 2019 with runway–taxiway delays contributing 37% (EuroControl, 2019). Moreover, over the last five years, approximately 1500 runway incursions have been reported annually in the US alone (Federal Aviation Administration, 2021; Mathew et al., 2017). Regardless of specific errors or causes, runway incursions have serious implications, which may include a ground conflict or collision (The National Aeronautics and Space Administration, 2019). This may cause significant delays and disruptions to airport-airside movements. Even in 2007, the Airline Pilot Association highlighted that in the United States approximately one runway incursion occurred each day, and the potential for a catastrophic accident was deemed "unacceptable", according to the FAA's risk/severity matrix, at that time (Airline Pilot Association International, 2007). The incidence of runway

* Corresponding author.
E-mail address: thai0009@e.ntu.edu.sg (P. Thai).

https://doi.org/10.1016/j.trc.2022.103590
Received 31 July 2021; Received in revised form 20 January 2022; Accepted 27 January 2022
Available online 21 February 2022

Nomenclature

AP	Average Precision
ATCOs	Air Traffic Controllers
BiFPN	Bidirectional Feature Pyramid Network
conv	Standard Convolutional Operation
ConvNets	Convolutional Neural Networks
dconv	Depthwise Convolutional Operation
GSE	Ground Support Equipment
IOU	Intersection Over Union
TOBT	Target Off Block Time

incursions grows exponentially as a function of air traffic growth. For example, over the period 1988 to 1990, United States towered airport traffic volume increased 4.76 percent, yet the runway incursion rate at these airports increased over 43 percent. Furthermore, from 1990 to 1993, traffic volume decreased 5.34 percent, and the runway incursion rate decreased by 30 percent. Finally, from 1993 to 1998, traffic volume grew by 2.41 percent, yet the runway incursion rate increased 67 percent (Airline Pilot Association International, 2007). Therefore effective airport ground movement control has the potential to both reduce flight delay and enhance the safe management of airport-airside traffic.

Air traffic controllers (ATCOs) are required to continuously detect and track all traffic operations, including aircraft, ground vehicles, and people, by visual observation. In particular, for aircraft ATCOs also need to identify aircraft type and corresponding airline (Watson et al., 2009), as well as manage aircraft speed, direction, and location on taxiways (Ellis and Liston, 2016), to avoid any potential risk of collision.

The airport movement area is considered a complex environment for several reasons. First, the monitored area is wide, which has a large impact on visibility. Second, object speeds and sizes are not consistent and can range from small high-speed aircraft on the taxiway to jumbo stationary aircraft at the apron. Third, the distribution of object density can vary from one aircraft on the runway to more than ten objects at the apron. Last, but not least, different control tasks are required during different phases by ATCOs. For example, ATCOs are required to maintain separation of maneuvering aircraft, whereas this is not an issue for stationary aircraft undergoing a turnaround process, where a push-back time prediction is required for efficient operations. Therefore, there is a need for better decision support tools for ATCOs to improve situation awareness and reduce workload while improving efficiency.

Aircraft turnaround is the process of preparing an aircraft for flight which consists of many activities including deboarding, catering, cleaning, fueling, cargo unloading and loading, and boarding, which can be performed simultaneously or sequentially depending on regulations (Schmidt, 2017). The process also involves many Ground Support Equipment (GSE), such as cargo loader, cargo truck, fuel truck, and tow truck. ATCOs are required to monitor the aircraft turnaround processes and constantly update the gate push-back times to maintain constant runway pressure and manage departure queues at the runway holding point (Wu and Caves, 2000).

The concept of the digital tower has been developed as an alternative to conventional towers (Fürstenau, 2016) for airport movement area management. By using a network of high-resolution cameras that cover a 360-degree view of airports, the digital tower can provide many advantages over a conventional tower (Frequentis, 2018). To take advantage of this digital video system, many computer vision systems have been developed (Pavlidou et al., 2005; Koutsia et al., 2008; Bloisi et al., 2012; Lu et al., 2016; Zhang and Qiao, 2020). However, due to the complexity of the airport environment, these systems only focus on specific targets, such as tracking moving objects or detecting objects at the apron.

This paper proposes a computer vision surveillance framework to monitor ground movement objects for safety enhancement and operational efficiency improvement as illustrated in Fig. 1. Aircraft are detected as they land on the runway and tracked throughout their maneuvering on the taxiway to the apron or gate. During their maneuvering, the proposed framework estimates aircraft speed and the distance between aircraft for enhanced situation awareness. At this stage, aircraft are moving relatively fast in a wide area which requires several cameras to completely cover. Therefore, besides accuracy, processing speed is an essential factor; increased observation frequency leads to safety enhancement. When aircraft arrive at the apron area, the turnaround process is monitored and aircraft push-back time is predicted. As the turnaround process may take an hour or more and involve many GSE, this stage requires exhaustively detecting objects with low frequency. In order to do so, we first create a novel Convolutional Neural Network (ConvNet) architecture, called AirNet, designed to suit the surveillance requirements. Then, a camera calibration model is developed for the accurate estimation of aircraft speed and distance. The calibration model transforms pixel locations to geographic locations (latitude and longitude). Typically, camera calibration can be divided into two categories, photogrammetric calibration and self calibration (Chuang, 2005). Photogrammetric calibration aims to estimate camera parameters in order to establish the formulation between pixel locations and geographic locations, while self calibration estimates the relationship with actual camera parameters. Photogrammetric calibration requires a known object point array to reconstruct the pattern, such as placing a checkerboard with different orientations in front of the cameras, which is clearly not application in our dataset. Therefore, self calibration, which does not require calibration objects, is our chosen technique. Third, based on detected aircraft and GSE, aircraft turnaround activities are then identified by the framework and aircraft push-back time is predicted. Since activities are performed simultaneously or

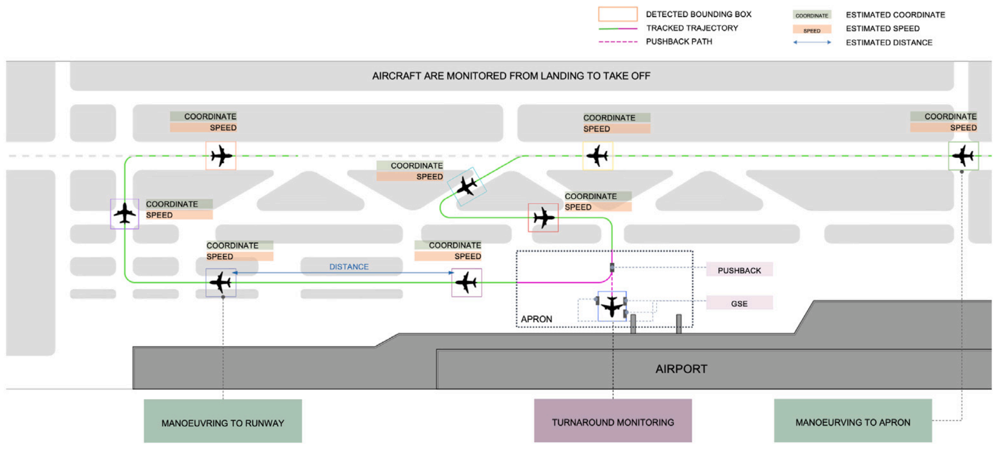


Fig. 1. The visual illustration of the airport-airside operations. Aircraft are detected at the moment they land and then are tracked through the maneuvering to apron and then back to take-off.

sequentially, a chained relationship is established between activities. Thus, push-back is predicted by using function composition and is updated in real-time.

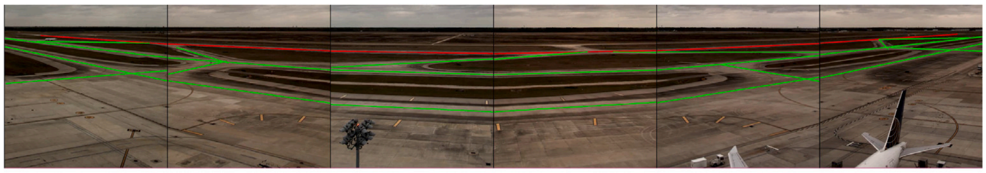
The rest of this paper is organized as follows. In Section 2, a brief review of previous related work is presented, and the input video data is described in Section 3. Section 4 presents the proposed framework with details of the experiment design outlined in Section 5. Results and comparison to other techniques is presented in Section 6. Finally, the paper concludes with summary and discussions in Section 7.

2. Related work

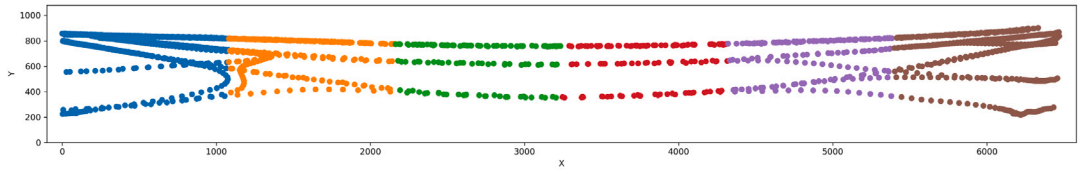
Different camera-based systems have been developed for airport-airside surveillance at many airports worldwide (Pavlidou et al., 2005; Koutsia et al., 2008; Bloisi et al., 2012; Lu et al., 2016; Zhang and Qiao, 2020). As an airport movement area is a complex environment, these systems only focus on specific targets. For example, by using foreground–background difference, the systems in Pavlidou et al. (2005), Koutsia et al. (2008) and Zhang and Qiao (2020) can detect and track moving aircraft; or by using object shape, the systems in Bloisi et al. (2012) and Lu et al. (2016) detect aircraft at the apron where object sizes are ordinarily large.

This paper adopts a computer vision based approach for airport-airside surveillance, which includes ConvNets for object detection and linear transformation for camera calibration. In literature, ConvNets for object detection can be clustered into one-stage approaches and two-stage approaches.Two-stage approaches (Ren et al., 2015) separate object localization and classification into two steps. First, an algorithm, such as Region Proposal Network (Ren et al., 2015) or selective search (Girshick et al., 2014), locates potential objects from an input image. Then, the potential objects are classified into specific classes. Two-stage approaches detect with accurate location but require massive computation time. On the other hand, one-stage approaches (Redmon et al., 2016) unify two steps into one which reduces computation. However, one-stage approach performance is lower than two-stage approaches. Fortunately, by introducing focal loss (Lin et al., 2017), one-stage approaches such as EfficientDet (Tan et al., 2020) and Yolov4 (Bochkovskiy et al., 2020) can outperform two-stage approaches. Therefore, EfficientDet and Yolov4 are selected as baseline models. The foundation concept of Yolo (Redmon et al., 2016) is to unify several object detection components into a single network to execute exceptionally rapidly. After three previous versions, Yolov4 has been improved by adapting new techniques including Cross Stage Partial Network (Wang et al., 2020), Path Aggregation Network (Liu et al., 2018), Mosaic data augmentation (Bochkovskiy et al., 2020) and so on, which makes Yolov4 one of the state-of-the-art ConvNet models. On the other hand, EfficientDet (Tan et al., 2020) focuses more on optimization than techniques. By using Neural Architect Search (Kyriakides and Margaritis, 2020) on hyper-parameters, the authors optimize model scaling, which maximizes performance with constrained resources These two ConvNets are designed for a challenging dataset, MSCOCO (Lin et al., 2014), which consists of more than 300 thousand images of 80 classes with different conditions.

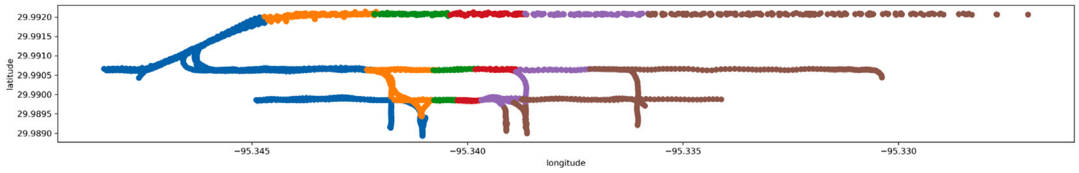
Camera calibration is a key process for airport-airside surveillance when estimating camera parameters to transform pixel locations to geographic locations. Camera parameters consist of intrinsic matrix parameters including focal length, optical center, skew, distortion parameters, and extrinsic matrix parameters including rotation and translation parameters (Chuang, 2005). Photogrammetric calibration (Zhang, 2000) formulates the mathematical transformation equation. This approach requires a calibration object (e.g., checkerboard) to establish the camera pattern to solve the equation. On the other hand, self calibration (Heikkila and Silvén, 1997) replaces the camera matrix by a projection matrix and formulates the linear transformation, which requires the set of pixel and geographic points. The transformation can be solved by Singular Value Decomposition or Linear Least Square (Heikkila and Silvén, 1997). However, linear transformation does not work well with a wide area such as airport which suffers from distortion. Therefore, this paper adopts gradient descent with a polynomial kernel to reduce the distortion factor. Furthermore, to further increase the accuracy, more than one linear transformation is applied for some cameras.



(a) The airport layout captured by 6 cameras. The runway and taxiways are highlighted by green color.



(b) Aircraft trajectories captured by cameras. Each color represents each camera.



(c) Aircraft trajectories recorded by radar. The colors are corresponding with cameras.

Fig. 2. Houston Airport captured from a camera system with corresponding radar information.

For aircraft turnaround, previous works analyze the turnaround process from various perspectives. For example, for reducing turnaround time, previous works attempt to re-design the airport layout (Mota et al., 2017) or optimize resource allocation (Okwir et al., 2017; Evler et al., 2021). Other works aim to predict stochastic events such as passenger boarding (Schultz and Reitmann, 2019; Schultz, 2018) or arrival delay (Oreschko et al., 2012), to reduce the uncertainty. It is worth highlighting that all these works analyze the turnaround process before it occurs. This paper monitors aircraft turnaround through video analytics during the turnaround process. By detecting objects and identifying turnaround activities, aircraft push-back is predicted by using function composition and is updated in real-time.

3. Dataset and data engineering

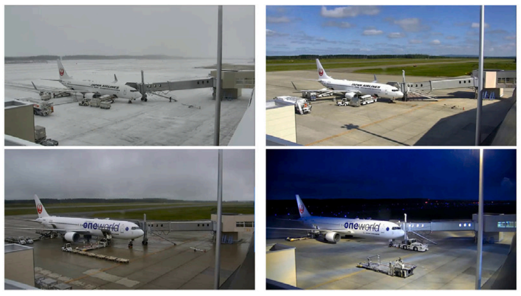
Datasets from two different airports are used in this work: Houston airport for airside maneuvering surveillance and Obihiro airport for apron/gate area surveillance, due to their video dataset characteristics. A system of cameras is required to cover the whole runway and its vicinity for the maneuvering area. However, only one camera is required to cover an apron related to the turnaround process. In order to accurately predict aircraft push-back, the algorithm needs many videos over several months, however this quantity of video data is not required for airside maneuvering surveillance. A benefit of using video data from more than one airport is that the detection models can demonstrate their generalization robustness.

3.1. Houston airport, USA data for airport-airside surveillance

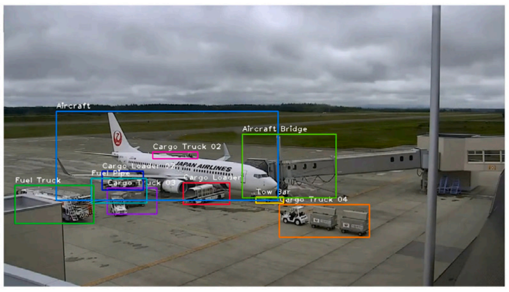
Video data from Saab Digital tower at Houston airport, from a system of 14 cameras, is collected. It comprises one hour of video data of airport-airside movements. The data is provided by Saab Inc. under Saab-NTU Joint Lab research data repository. As the airside maneuver surveillance research only focuses on the maneuvering area, only six of the fourteen cameras are selected, as shown in Fig. 2a. Furthermore, the sky area that is not related to the research is removed, resulting in a frame resolution of 1080 × 1080. Objects, including aircraft and ground vehicles, are manually labeled as bounding boxes. Aircraft bounding boxes are connected through videos, creating trajectories as shown in Fig. 2b. The aircraft ground truth, including geographic location, callsign, and type, are provided by radar and linked with camera information, as shown in Fig. 2c. As a result, a dataset of 3765 images is created with two classes.

3.2. Obihiro airport, Japan data for aircraft turnaround monitoring

Obihiro airport videos are collected from a boarding gate camera over a period of nearly six months in a variety of different weather conditions, as shown in Fig. 3a. Each video captures a complete aircraft turnaround process with lengths of 40 min to 80 min and a resolution of 1920 × 1080. The data is collected using the live YouTube Gate videocam link provided by Obihiro



(a) Videos are collected with different times and weather conditions.



(b) The bounding boxes are heavily overlap which makes detecting difficult

Fig. 3. Videos collected from a live camera in Tokachi-Obihiro airport and objects are manually labeled as bounding boxes.

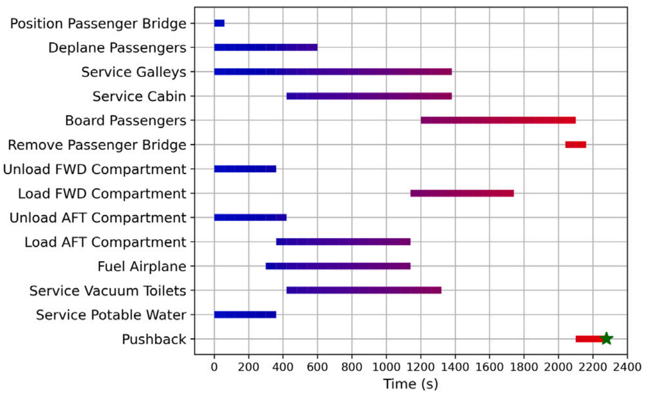


Fig. 4. Aircraft turnaround process of B737 starting when an aircraft arrives (second 0) and ending when an aircraft pushes back. There are 14 activities which perform simultaneously or sequentially over time (from blue to red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

airport. Twenty videos are chosen, resulting in over 12 h of data. Objects in these videos are manually labeled as bounding boxes for the object detection task. Frames are extracted every three seconds, producing 15 429 labeled images.

The following seven objects are labeled: aircraft, aircraft bridge, cargo truck, cargo loader, fuel truck, fuel pipe, and tow truck, as shown in Fig. 3b. For the push-back prediction task, 158 videos were chosen to manually label the duration of activities (that is, the activity start and end times). The following thirteen activities are labeled: aircraft arrival, bridge attachment, bridge detachment, cargo loader attachment, cargo loader detachment, unload-begin, unload-end, load-begin, load-end, fuel pipe drawing in, fuel pipe drawing out, tow truck connection, and push-back.

Labeled activities are slightly different compared to the turnaround manual (Fig. 4) due to two reasons. First, the camera cannot capture some activities such as passenger deboarding/boarding, service galleys, and service cabin. Second, some activities do not take place in the recorded videos, including service vacuum toilets and service potable water. In addition to the video data, 2225 flight performance data were collected for baseline comparison, including schedule arrival and departure, from the FlightAware website.[1]

3.3. Airport characteristics

Compared to public datasets, such as MSCOCO (Lin et al., 2014) or Pascal VOC (Everingham et al., 2010), our datasets have a number of different characteristics.

- High resolution videos. Although image resolutions from public datasets are varied, their average is around 640 × 640. However, our image resolution is 1920 × 1080 which is 5 times larger than the public datasets. The larger image resolution, the more memory and computation required to detect objects which can compromise a real-time detection requirement.

[1] https://flightaware.com/.

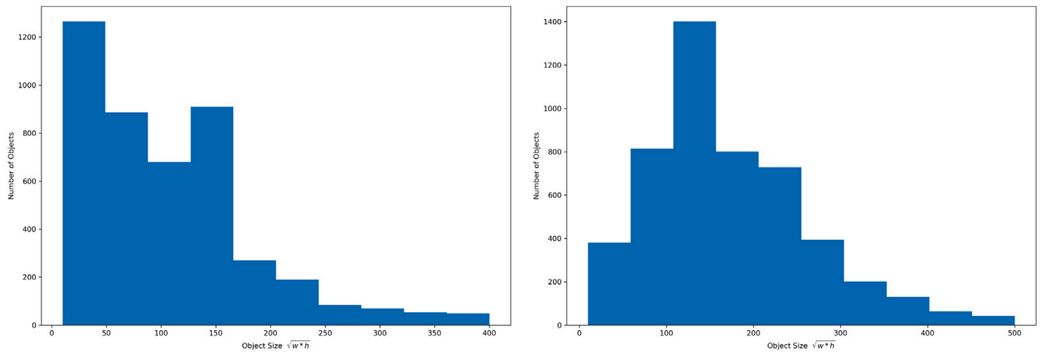


Fig. 5. Distribution of object sizes in Houston airport (left) and Obihiro airport (right). Due to the camera position, objects in Houston airport are smaller compared to Obihiro airport.

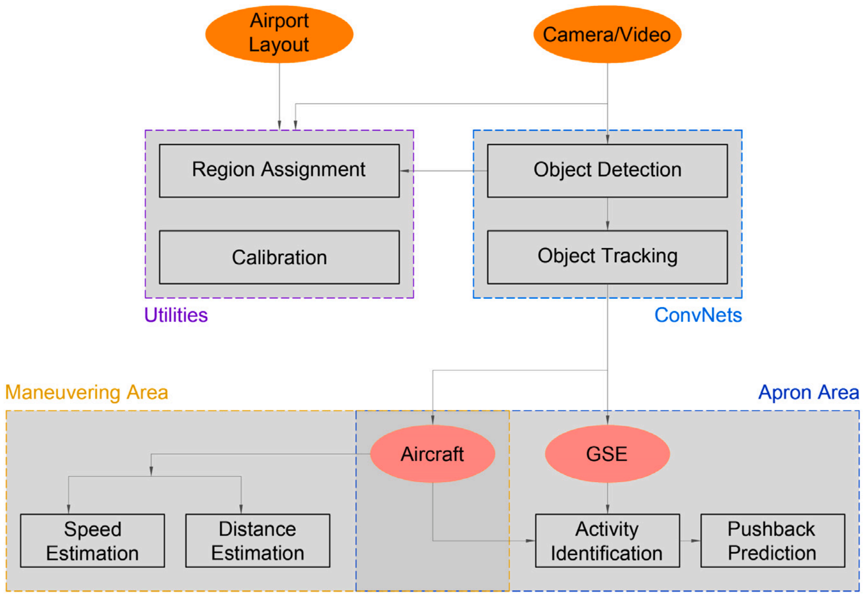


Fig. 6. The concept diagram of the proposed framework. Objects are detected and tracked by using ConvNets. Then, based on aircraft location, either speed/distance is estimated, or push-back is predicted. The camera is calibrated to transform pixel coordinates to geographic coordinates.

- Small object sizes. Although object size distribution ranges from 10 pixels to 500 pixels, the distribution is skewed toward small sizes as shown in Fig. 5. Small objects are common in other datasets. However, a 10-pixel object in a 1920 × 1080 image is extremely difficult to be detected compared to the object in a 640 × 640 image.
- Small sample sizes. The dataset, which consists of 3765 images, can be considered a small dataset. Therefore, without proper regularization, the detection models can be overfit or even not converge.
- Low number of classes. The number of classes are two and seven in Houston airport and Obihiro airport, respectively, which are fewer than other datasets, (20 in VOC and 80 in COCO). Hence, with under similar conditions, object detection in airport might achieve higher performance.
- Static cameras. Although airport cameras capture a large field of view with different weather conditions, the background is simpler than other datasets, making it easier to distinguish between foreground and background.

We therefore have created a new ConvNet, AirNet, to adapt to the unique characteristics of the airport datasets.

4. Methodology

4.1. Overview

Fig. 6 illustrates a concept diagram of proposed framework. The framework exhaustively detects every aircraft from camera video feeds. Detected aircraft are assigned to corresponding regions, including the maneuvering area (runway and taxiway) or apron area.

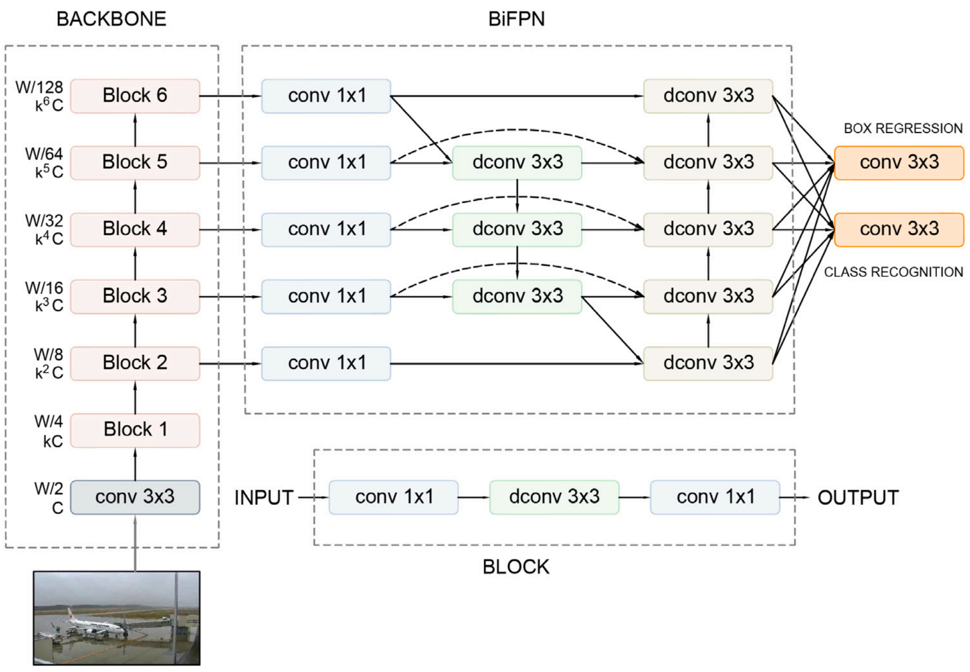


Fig. 7. The AirNet architecture. A backbone extracts features from an input image, creating feature maps with five scales. Then the feature maps will be refined by BiFPN. Finally, the network will detect objects including bounding box regression and class recognition.

During aircraft maneuvering, aircraft geographic locations are estimated to estimate their speed and the distance between two aircraft. Once aircraft arrive at the apron, GSE are also detected to monitor the turnaround process. The turnaround process ends when the aircraft pushes back.

The framework is divided into three main modules: object detection, camera calibration (maneuvering area), and turnaround monitoring (apron area). First, in the object detection module, the developed AirNet architecture detects aircraft and related objects. Then, the detected objects are tracked via a tracking-by-detection method. In the camera calibration module, a linear regression model for geographic transformation is implemented. Then, aircraft speed and distance estimation are calculated. In the turnaround monitoring module, activities are identified based on detected objects. Then, push-back time is predicted when aircraft arrive at the apron and is continuously updated until the actual push-back occurs.

4.2. Object detection

We developed AirNet for Deep4Air framework in earlier work (Van Phat et al., 2021). Because of the expansion of scope, the AirNet architecture is refined further. Fig. 7 describes the AirNet architecture. An input image is fed to a backbone to extract features. Next, a bidirectional feature pyramid network (BiFPN) (Tan et al., 2020) aggregates features at different scales. Finally, the network detects objects as bounding boxes and classes. The ideas behind the AirNet architecture are described as follows.

1. The number of feature scales. To detect object locations, proposal regions with different scales and aspect ratios, called anchors (Ren et al., 2015), slide on the feature maps. By default (Lin et al., 2017), we use 3 scales $(2^0, 2^{1/3}, 2^{2/3})$ and 3 aspect ratios (0.5, 1, 2) for each sliding position. Therefore, with an anchor of size w_0, a feature map can cover objects from $0.5w_0$ to $3w_0$. As object sizes from the airport dataset range from 2^3 to 2^9 pixels, we build AirNet with 5 feature map scales to cover the range of object sizes.
2. Convolutional operations. We use standard convolutional operations and depthwise convolutional operations as shown in Eqs. (1) and (2), respectively, where I $(W_1 \times W_1 \times C_1)$, O $(W_2 \times W_2 \times C_2)$ and K $(D \times D \times C_1 \times C_2)$ are input, output and kernel, respectively, with square inputs and outputs for simplification. We mainly use the depthwise convolutional operation (dconv) (Howard et al., 2017) which reduces the computation by approximately 9 times compared to a standard convolutional operation (conv) with a small drop in accuracy. However, depthwise convolution cannot learn cross-channel $(C_1 = C_2 = C)$ information. Therefore, at some stages, standard convolution is applied.

$$O_{k,l,c_2} = \sum_{i,j,c_1} K_{i,j,c_1,c_2} \cdot I_{k+i-1,l+j-1,c_1} \tag{1}$$

$$O_{k,l,c} = \sum_{i,j} K_{i,j,c} \cdot I_{k+i-1,l+j-1,c} \tag{2}$$

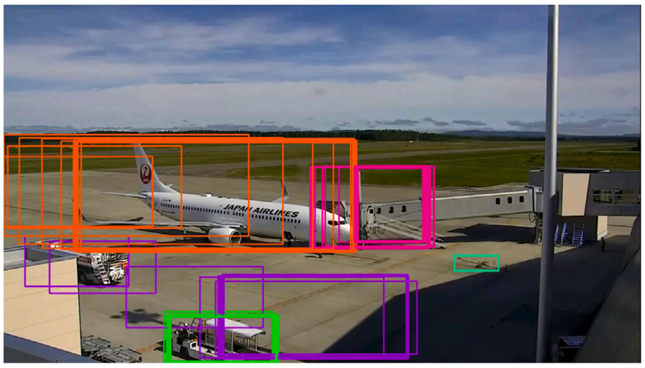


Fig. 8. Aircraft tracking by detection. By comparing object bounding boxes and classes from previous frames, the algorithm can map them with the current frame.

3. Backbone. The first 3 × 3 standard convolution normalizes an input image ($W \times W \times 3$) to an output tensor ($W/2 \times W/2 \times C$). The backbone consists of 6 identical blocks. After passing each block, the input size will be reduced by a factor of 2 and the number of channels will increase by k. Each block consists of two 1 × 1 standard convolutions to learn information cross channel, and 3 × 3 depthwise convolution to learn information within channels. We use stride equal to 2 for 3 × 3 convolutions and stride equal to 1 for 1 × 1 convolutions. Therefore, after every 3 × 3 convolution, tensor sizes reduce by 2 times. Moreover, we apply batch normalization (Ioffe and Szegedy, 2015) and swish activation (Ramachandran et al., 2017) after every convolution operation. W, C and k are hyper-parameters which will be used to create a trade-off between computation and performance.
4. Bidirectional feature pyramid network. The first 1 × 1 standard convolutions each scale normalize the channels to be the same, which is kC. Following this, information from different scales will be aggregated by using depthwise convolutions with stride equal to 1, as shown in Eq. (3).

$$B_i^{out} = dconv(B_i^{in} + dconv(B_i^{in} + Resize(B_{i+1}^{in})) + Resize(B_{i-1}^{out})) \tag{3}$$

5. Detection. The final feature maps will be used to detect objects which include object locations (x_1, y_1, x_2, y_2) and object classes by using 3 × 3 standard convolutions with stride equal to 1. The final output sizes are $a \times 4$ for bounding box regression and $a \times n$, where a and n is number of anchors and classes, respectively.
6. Scalability. To work with different datasets and purposes, the network must be scalable, creating a trade-off between speed and performance. The network can be scaled in three directions which are input resolution (W), number of channels (C) and number of layers. The optimal method is to scale three directions at the same time (Tan and Le, 2019). The number of layers can be increased by repeating certain blocks with stride equal to 1.

Detected objects are tracked by tracking-by-detection, as shown in Fig. 8. Tracking-by-detection is a process of mapping detected objects between frames from videos to create a sequence of objects. First, we create a list of objects (R) by using detected objects in the first frame. Each object has three attributes which are location (x_1, y_1, x_2, y_2), class and the last detected time (t_0). At the time t, we calculate intersection over union (IOU) between objects in R and detected objects at time t (R^t) by using Eq. (4). Two objects are considered to be the same object if they are from the same class, their IOU is greater than 0 and the detected time difference $(t - t_0)$ is smaller than a threshold parameter. The value of threshold is dependent on the performance of detection models. The value is small with high performance detection or large with low performance detection. If more than one object meets the conditions, the highest IOU is chosen. An object which cannot be mapped to any previous objects is considered a new object. Finally, the newly detected objects will be added to the object list R as either existing objects or new objects.

$$IOU(r_1, r_2) = \frac{Area(r_1) \cap Area(r_2)}{Area(r_1) \cup Area(r_2)} \tag{4}$$

4.3. Camera calibration

Camera calibration is used to transform video pixel locations to geographic locations. We collect N pixel points (X) with N corresponding geographic points (Y) to construct the data. The linear regression can be written as follows:

$$\tilde{Y} = Wf(X) + \epsilon, \tag{5}$$

where W is the weight learned from data during the training process, $f(.)$ can be viewed as the feature extraction function, and ϵ is the irreducible error occurring when collecting data. Naturally, we choose $f(X)$ as a polynomial function with degree k, as shown

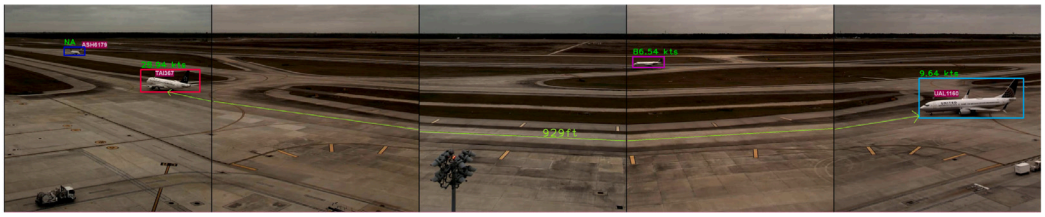


Fig. 9. Aircraft segment, speed and distance estimation during maneuvering. One camera is removed for visualization purpose.

in Eq. (6). As linear regression transforms pixel locations to geographic locations, a time axis is not required.

$$f(x_1, x_2) = \sum_{i=0}^{k} \sum_{j=0}^{i} x_1^j x_2^{i-j} \tag{6}$$

Fig. 2 shows that the areas captured by cameras are not uniform. By using one transformation model for each camera, the errors from edge cameras are larger when compared to central cameras due to distortion. Therefore, some cameras have more than one transformation model to reduce this distortion factor. The points from each camera will be grouped into clusters. Since linear regression models are applied, the cluster center is a line and the most convenient choice would be runway and taxiways, as shown in Fig. 2. Points are assigned to corresponding clusters with the minimum distance from the points to the cluster centers. However, to prevent overfitting, the number of points each cluster contains is required to be more than a pre-determined threshold N_0. When a cluster does not meet this requirement, that cluster is removed and its points will be assigned to other clusters with the same method. The algorithm ends when all cluster points are higher than N_0.

Aircraft are assigned to corresponding taxiway or runway segments (Fig. 2a) by calculating the intersections between aircraft and the segments. If there is no intersection, the aircraft does not belong to any segment. If there is only one intersection, we assign the aircraft to that segment. If there is more than one intersecting segment, we need to calculate the aircraft's speed. If the aircraft is stationary, the aircraft is assigned to the same segment as previously. If it is moving, we calculate its direction using its tracked trajectory, ranging from 0° to 360°, and compare it with the segment directions. The aircraft is mapped to the segment having the smallest difference in direction.

Aircraft speed is estimated by calculating the distance of a sequence of points over time. Intuitively, we choose the center points of bounding boxes resulting in a sequence of points $X = (x_1^t, x_2^t), t \in (1, \dots, T)$. Then, X are transformed to geographic locations Y by Eq. (5) with $\epsilon = 0$. Finally, the speed is calculated by the distances divided by the times between two consecutive points where distance is calculated by the haversine formula (Van Brummelen, 2012):

$$\begin{aligned} dy_i &= y_i^{(t)} - y_i^{(t-1)} \\ a^{(t)} &= \sqrt{\sin^2 \frac{dy_1}{2} + \cos y_1^{(t)} \cos y_1^{(t-1)} \sin^2 \frac{dy_2}{2}} \\ d^{(t)} &= 2R \arcsin a^{(t)}; R = AverageEarthRadius \end{aligned} \tag{7}$$

The distances between each pair of aircraft in the same segment are estimated. First, we need to find every pair of aircraft in the same segment. Next, intersection points between each aircraft and the segment are computed. Depending on aircraft position, there are one or two intersection points for each aircraft. These two points represent the head and the tail of the aircraft. The shortest distance between two aircraft is from the tail of the leading aircraft to the head of the trailing aircraft calculated by Eq. (7).

Fig. 9 shows the results of aircraft segmentation (NA is a taxiway name), speed estimation, and distance estimation. The conventional unit of speed and distance are knots and feet, respectively.

4.4. *Aircraft turnaround monitoring*

Activity identification requires the relationship between the involved objects and the aircraft. For example, to detect bridge attachment and detachment, the position of the bridge and aircraft are required. We extract two pieces of information from the object position: speed and IOU of the involved objects and the aircraft. To calculate object speed, we subtract the center points of the object bounding boxes over time. As objects are detected every second, the speed unit is the number of pixels per second. The IOU is calculated by the ratio between the overlapping area and the area of the union of the two objects. The left chart of Fig. 10 shows an example of speed (green) and IOU (blue) over time. As object bounding boxes are not stable over time, the speed and IOU exhibit fluctuation. By using these two types of information, attachment activities can be identified when speed and IOU increase, as shown in the upper right chart of Fig. 10. In contrast, detachment activities can be identified when speed increases and IOU decreases, as depicted in the lower right chart of Fig. 10.

As activities can be performed simultaneously or sequentially, activities are grouped into three chains as shown in Fig. 11. In this research, the schedule departure is assumed to be the target-off-block-time (TOBT), which is the target time for aircraft to push

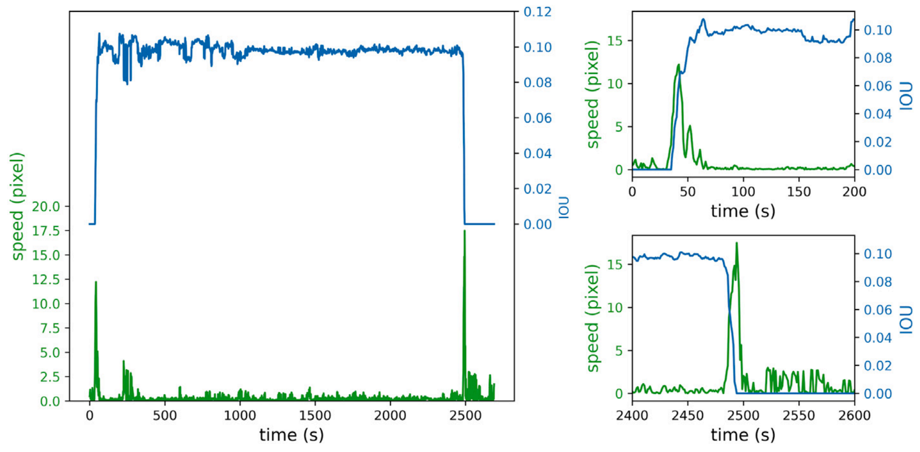


Fig. 10. Activity detection is based on speed (green) of a involved object and intersection over union (blue) of the involved object and the aircraft. The left figure shows the whole process while the upper right and lower right focus on first 200 s and last 200 s respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

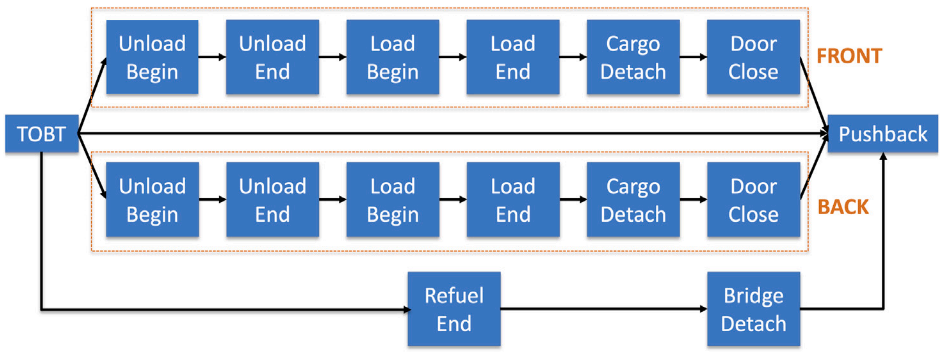


Fig. 11. Push-back prediction by using chained linear regression models.

back. The push-back prediction is computed by the following formula:

$$\begin{aligned} y &= g(x_0, x_{1,6}, x_{2,6}, x_{3,2}) \\ x_{i,j} &= f_{i,j-1}(x_{i,j-1}) \end{aligned} \tag{8}$$

Here, x_0 is TOBT, $i \in [1,3]$ is a chain, $j \in [1,6]$ is a member of a chain, and $f(.)$, $g(.)$ are learnable functions. The reason we apply function composition is based on our analysis, as shown in Fig. 12. A distant pair of activities, such as Load Begin Front and Pushback, does not likely have a strong relation. However, the immediate pairs of activities, such as Load End Front and Cargo Detach Front, or Cargo Detach Front and Door Close Front, have linear relations. Therefore, $f(.)$ and $g(.)$ are chosen as linear functions with slightly different formulas:

$$\begin{aligned} y &= g(x_0, x_{1,6}, x_{2,6}, x_{3,2}) = \alpha x_0 + \beta \max(x_{1,6}, x_{2,6}) + \gamma x_{3,2} + \delta \\ x_{i,j} &= f_{i,j-1}(x_{i,j-1}) + a_{i,j-1} x_{i,j-1} + b_{i,j-1} \end{aligned} \tag{9}$$

where $\alpha, \beta, \gamma, \delta, a_{i,j}$ and $b_{i,j}$ are learnable parameters.

5. Experiments

5.1. Object detection

The ConvNets are trained on Houston airport and Obihiro airport data. The training processes are performed on a computer with 4 Nvidia Quadro RTX 6000 GPUs, but the test processes only use 1 GPU to report the average running time per image. The hyper-parameters for AirNet are input resolution (W), number of channels (C) and their scale (k), and the number of layers. Due to airport characteristics, W tends to be large while C tends to be small. Because of computational efficiency, W is divisible by 128 and C is divisible by 4 (Sandler et al., 2018). The initial values for W and C are 512 and 16. As we want C to be small, we

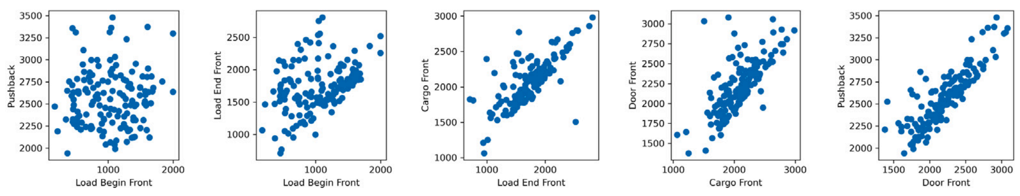


Fig. 12. Relationship between various ramp activities. Load Begin Front and Pushback appears to be random, however the relationship of immediate activities are linear (Load Begin Front has chained linear relationship with Pushback).

choose $k = 1.5$, so that the number channels of block 6 are 180. As AirNet is a new model, we train from scratch. The whole model is trained with learning rate of 10^{-4} until losses can no longer reduce. To create a trade-off between performance and computation, we use a compound scaling method with factors $\alpha = 1.2$ and $\beta = 1.1$ (Tan and Le, 2019) for the number of layers and number of channels, respectively. At the same time, W increases by 128. AirNet is scaled until the computer does not have enough memory to run the training process.

To compare with AirNet, we choose three state-of-the-art ConvNets which are Yolov3, Yolov4 and EfficientDet. Transfer learning (Yosinski et al., 2014), which improves performance compared to training from scratch, is used to train the models. First, we load the pretrained models trained from the large MSCOCO (Lin et al., 2014) dataset. By training on the large dataset, the models can learn general features from images. Since the outputs of the pretrained models are different from our desired outputs, we replace the final layers with different output nodes (2 for Houston airport and 7 for Obihiro airport). Then the next layers are required to be trained. Therefore, we freeze the parameters from the models, except these final layers, and train with several epochs so that these layers can adapt to new tasks. Finally, the entire models are unfrozen and trained until the losses can no longer reduce. Because we use the pretrained models, we follow the standard transfer learning procedure. For Yolov3 and Yolov4, the only hyper-parameters we can change is image resolution. For EfficientDet, we can change image resolution and model versions which are from B0 to B3 (Tan et al., 2020).

The evaluation metric used is average precision (AP). To define AP, first, we need to first discuss precision and recall. A high recall detector aims to detect as many objects as possible, which helps to reduce false negatives. In contrast, a high precision detector aims to detect objects as precisely as possible, which helps to reduce false positive detections. A precision–recall curve indicates the trade-off between precision and recall, which is important in determining the detection threshold. Since different thresholds are associated with different precision and recall values, AP is an important metric that calculates the average precision over all possible thresholds. In other words, AP, ranging from 0 to 1, indicates the detector's performance regardless of detection thresholds (i.e., the higher AP the better performance). Besides detection threshold, the IOU threshold can change the value of AP. AP_{50} is calculated by using an $IOU = 50\%$, while AP is the average of AP_{IOU} with IOU from 50 to 95 with stride 5. The AP_{50} metric indicates the recognition performance, while the AP metric describes both the localization and recognition performance. Because precise location is required to minimize the errors of speed and distance estimation, the AP metric is prioritized over AP_{50}. On the other hand, turnaround monitoring does not require a precise location to predict push-back, so AP_{50} is the primary metric.

5.2. Camera calibration

From six cameras in one hour, the dataset consists of 2817 pairs $X = (x_1, x_2)$ and $Y = (y_1, y_2)$, where X are pixel locations and Y are geographic locations. As different cameras cover different areas of airport, as shown in Fig. 2, the number of points each camera covers are different which are 1300, 325, 133, 125, 177 and 757, from left to right. The first setup includes six models for six cameras, one per camera. However, since some cameras cover a larger airport area than others, the uniform distribution might not be appropriate. By choosing $N_0 = 125$ (the minimum points among the six cameras), we divide the dataset into 11 clusters, with the number of clusters per camera being 3, 2, 1, 1, 1 and 3, from left to right. Therefore, the second setup has 11 models corresponding to 11 clusters. The models are identical and are shown in Eq (6) with hyper-parameter k. The chosen degree k should be large enough to prevent underfitting and small enough to prevent overfitting. By experimentation, we choose $k = 5$.

5.3. Aircraft turnaround monitoring

Turnaround activities are predicted from the moment aircraft arrive at the gate by using Eq (9). Then, these activities are repeatedly predicted until they are identified. The predicted activities overtime are adjusted by following equation:

$$x_{i,j}^{t} = \begin{cases} x_{i,j}^{*} & \text{if } t > x_{i,j}^{*} \\ t & \text{if } t \geq x_{i,j} \\ x_{i,j} & \text{if } t < x_{i,j} \end{cases} \tag{10}$$

where $x_{i,j}^{*}$ and $x_{i,j}$ are times of actual and predicted activities, respectively, and t is the current time. The Eq (10) can be interpreted as follows. If actual activities have already occurred (case 1), there is no need for prediction anymore. If predicted activities are

Table 1
The experimental results, including the input resolution, the number of parameters, the running time and the average precision, from different models for Houston airport dataset. The unit of number of parameters and time are millions and milliseconds respectively.

Model	Res	Param	Time	AP	AP_{50}	$AP_{aircraft}$	$AP_{vehicle}$
YOLOv3	512	–	15	55.25%	93.11%	53.85%	56.66%
YOLOv4	512	27.6	19	**62.51%**	**94.90%**	**56.47%**	68.56%
YOLOv3	640	–	24	61.11%	95.03%	57.84%	64.37%
YOLOv4	640	27.6	29	67.22%	96.33%	60.35%	74.08%
AirNet	640	0.25	30	40.87%	78.36%	48.19%	33.54%
YOLOv3	768	–	31	64.51%	94.74%	58.05%	70.98%
AirNet	768	0.32	32	50.93%	88.64%	58.01%	43.84%
YOLOv3	896	–	37	65.33%	94.84%	58.28%	72.37%
AirNet	896	0.51	38	60.89%	94.67%	**66.22%**	55.56%
YOLOv4	768	27.6	39	**68.20%**	**96.26%**	61.30%	75.09%
YOLOv3	1024	–	48	64.18%	94.76%	58.83%	69.53%
AirNet	1024	0.81	50	**70.43%**	96.77%	**70.98%**	69.89%
YOLOv4	896	27.6	52	67.67%	**97.03%**	61.92%	73.43%
YOLOv3	1152	–	59	63.14%	95.03%	58.51%	67.77%
AirNet	1152	1.29	62	**73.36%**	97.27%	**73.18%**	73.54%
YOLOv4	1024	27.6	70	65.45%	97.11%	60.52%	70.38%
EfficientDet	640	3.9	81	55.74%	92.90%	52.11%	59.36%
YOLOv4	1152	27.6	84	64.75%	95.94%	59.25%	70.25%
EfficientDet	768	6.6	99	67.43%	95.81%	66.42%	68.45%
EfficientDet	896	8.1	106	70.46%	97.07%	67.99%	72.94%
EfficientDet	1024	12	121	71.09%	**97.44%**	69.01%	73.18%

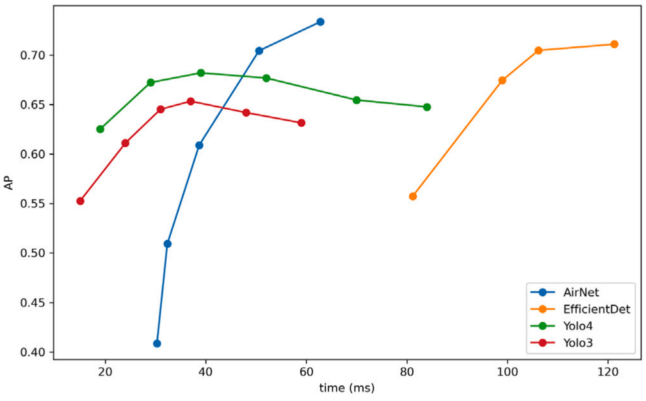


Fig. 13. Average precision and corresponding running time on Houston Airport dataset. Yolov4 achieves high computational performance, AirNet achieves the high precision performance.

earlier than the current time (case 2), the predicted activities are assigned the current time. As there are only 158 video samples, one-leave-out validation is applied to stabilize the results.

Compared with the proposed algorithm, two baselines are used, called Schedule 1 and Schedule 2. Schedule 1 is calculated by the root-mean square error between actual push-back and TOBT. For Schedule 2, a linear regression model is applied where TOBT is an input, and actual push-back is an output. Schedule 2 is trained on 2067 samples and tested on 158 video samples. It is worth noticing that at the moment of an aircraft arrival, TOBT is the only input our framework has.

6. Results

6.1. Object detection

Table 1 and Fig. 13 show the comparison of different ConvNets on the Houston airport dataset. Generally, Yolov4 achieves the best initial results compared to Yolov3, AirNet and EfficientDet, however, AirNet and EfficientDet eventually achieve higher AP results due to their scalability. For Yolov3 and Yolov4, increasing the input resolution achieves better accuracy, but the accuracy quickly stops improving and even decreases. On the other hand, AirNet does not suffer from this problem due to its compound scaling method. For EfficientDet, the computational time increases significantly with only a slight increase in performance. Hence, we stop increasing the input resolution at 1408. Although the models can detect aircraft and vehicles, the current framework only

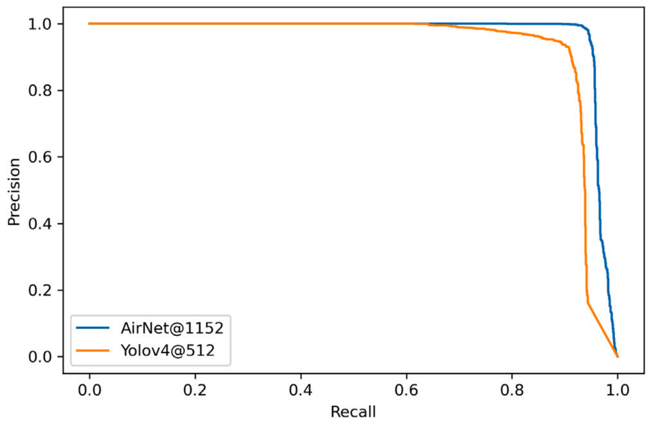


Fig. 14. Precision–Recall curves with IOU of 50% on Houston airport. The top-right corner is the most desirable region, where the detector is able to achieve high precision and high recall simultaneously.

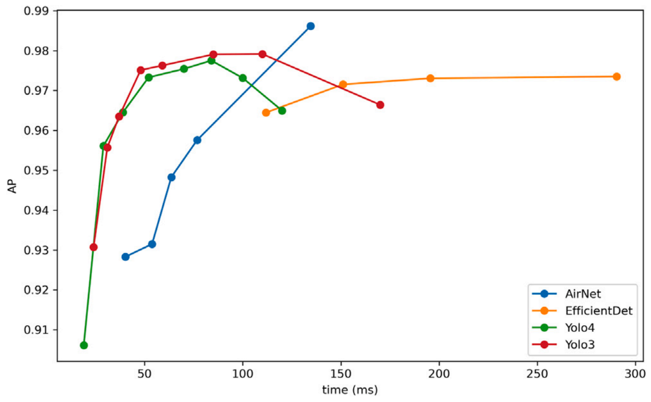


Fig. 15. Average precision and running computation time on Obihiro Airport dataset. AirNet achieves the highest performance.

focuses on aircraft-related tasks, including speed and distance estimation. Therefore, Table 1 also reports the AP of aircraft and vehicles separately. The vehicle performance is higher than aircraft performance in Yolov3, Yolov4 and EfficientDet despite vehicle size being smaller than aircraft size. The possible reason is that the pretrained models are trained with flying aircraft in the MSCOCO dataset, thus their performance for ground aircraft may be reduced during tuning. On the other hand, since AirNet is trained from scratch, the aircraft performance is higher than vehicle performance.

AP is a single number to compare the performance of different models. In order to fully understand the model performance, we need to pre-define the requirements. The first requirement is real-time detection. The Houston airport video dataset consists of six cameras with eight frames per second. To achieve real-time detection, we need one computer with Yolov4@512 or three computers with AirNet@1152. Second, the models need to detect and track every objects (high recall) in the videos for safety enhancement, but should not give too many false alarms (high precision). The Fig. 14 shows the precision–recall curves with IOU of 50% on Houston airport. Recall equals 1 (no false positive) is impossible to achieve in our dataset. Fortunately, objects can still be tracked even if they cannot be detected in some frames, thanks to the time threshold of tracking-by-detection. Nonetheless, we still need high recall without compromising precision. For example, if recall equals 0.95, the precision of AirNet@1152 equals 0.94. To put it simply, for every 100 objects, AirNet@1152 can detect correctly 95 objects and gives 6 false alarms. Yolov4@512 cannot achieve the requirement because the maximum recall with every detection threshold is 0.94. To provide another example, if recall equals 0.9, the precision of AirNet@1152 and Yolov4@512 are 0.9984 and 0.9361, respectively. Meaning that for every 10000 objects, AirNet@1152 can detect correctly 9000 objects and gives 14 false alarms. To achieve the same correct detection, Yolov4@512 has to give 614 false alarms.

Table 2 and Fig. 15 show the comparison of different ConvNets on Obihiro airport dataset. Similar to Houston airport dataset, the performance of Yolov3 and Yolov4 saturates when input resolution increases. AirNet, which has fewer parameters, can detect objects in high-resolution images, resulting in high performance compared to other ConvNets.

Table 2
Model performance including the input resolution, the number of parameters, the running time and the average precision for Obihiro airport dataset. The unit of number of parameters and time are millions and milliseconds respectively.

Model	Res	Param	Time	AP_{50}	AP
YOLOv4	512	27.6	19	90.61%	57.15%
YOLOv3	640	–	24	93.08%	59.09%
YOLOv4	640	27.6	29	95.61%	61.97%
YOLOv3	768	–	31	95.57%	69.72%
YOLOv3	896	–	37	96.35%	**80.12%**
YOLOv4	768	27.6	39	**96.45%**	65.37%
AirNet	1152	0.32	40	92.83%	70.77%
YOLOv3	1024	–	48	97.51%	**78.14%**
YOLOv4	896	27.6	52	97.33%	66.34%
AirNet	1408	0.51	53	93.15%	73.21%
YOLOv3	1152	–	59	97.63%	77.09%
AirNet	1536	0.81	63	94.83%	76.19%
YOLOv4	1024	27.6	70	97.54%	70.44%
AirNet	1792	1.29	76	95.76%	77.55%
YOLOv4	1152	27.6	84	97.75%	72.27%
YOLOv3	1408	–	85	**97.91%**	75.95%
YOLOv4	1408	27.6	100	97.31%	59.27%
YOLOv3	1536	–	110	97.91%	73.24%
EfficientDet	896	3.9	111	96.45%	82.82%
YOLOv4	1536	27.6	120	96.50%	54.03%
AirNet	2048	1.84	134	**98.62%**	**87.30%**
EfficientDet	1024	6.6	151	97.15%	82.89%
YOLOv3	1792	–	170	96.64%	66.01%
EfficientDet	1152	8.1	195	97.31%	84.15%
EfficientDet	1408	12	290	97.35%	83.98%

Table 3
Linear transformation errors (meters) with different cameras. By using multi models on margin cameras, the errors reduces significantly.

Camera	1			2		3	4	5	6		
Uniform	8.4			6.5		4.6	4.7	4.2	7.5		
Non uniform	4.2	6.1	2.5	4.8	4.6	4.6	4.7	4.2	5.5	2.7	2.4

6.2. Camera calibration

Table 3 shows the error of linear transformation models on six cameras with two setups from collected data. As expected, the errors from central cameras are lower than the edge cameras, due to the distortion factor. Therefore, by dividing the edge camera coverage into several parts, the errors can be reduced.

In a real application we will not have pixel locations initially. Instead, detected objects will be linear transformation models inputs. To validate the models, we choose a random detected object sequence by Yolov4@512 and AirNet@1152. Next, the center points of detected objects will be calculated, creating a trajectory as shown in the top chart of Fig. 16. Finally, the pixel trajectory will be transformed to geographic location and compared with ground truth trajectory, as shown in the bottom chart of Fig. 16. Hence, the total errors are the combination of detection error and transformation error. Since the AP of Yolov4@512 is lower than AirNet@1152, the root mean square errors of Yolov4@512 are 13.07 pixels compared to 8.28 pixels from AirNet@1152. After transformation, the final errors are 12.14 meters for Yolov4@512, compared to 6.01 meters for AirNet@1152.

6.3. Aircraft turnaround monitoring

Fig. 17 shows the results of the proposed algorithm. At the moment an aircraft arrives, the algorithm can predict its push-back with a mean error of 224 s, which is relatively small compared to the whole turnaround process (2400 s). During the turnaround process, the algorithm continuously predicts push-back with more information from activity identification as it arrives. Hence, the error is further reduced and converges to zero when aircraft performs push-back. By analyzing, we observe the reasons for high prediction errors are uncaptured activities or unusual events. For example, aircraft are sometimes waiting for push-back approval or engine check after passenger boarding, which the framework cannot capture. Therefore, if these events take a longer or shorter time than usual, the framework cannot use this information to adjust the prediction algorithm.

Fig. 18 shows the comparison between the proposed algorithm at different times with baselines. By simply using the data from the airport website without any processing (Schedule 1), the mean error is 411 s. The error can be reduced to 253 s by using linear regression on schedule and actual push-back pairs (Schedule 2). It is worth highlighting that the proposed algorithm error at the aircraft arrival (Arrival) is further reduced to 224 s by using the same information as Schedule 2. Since the proposed algorithm

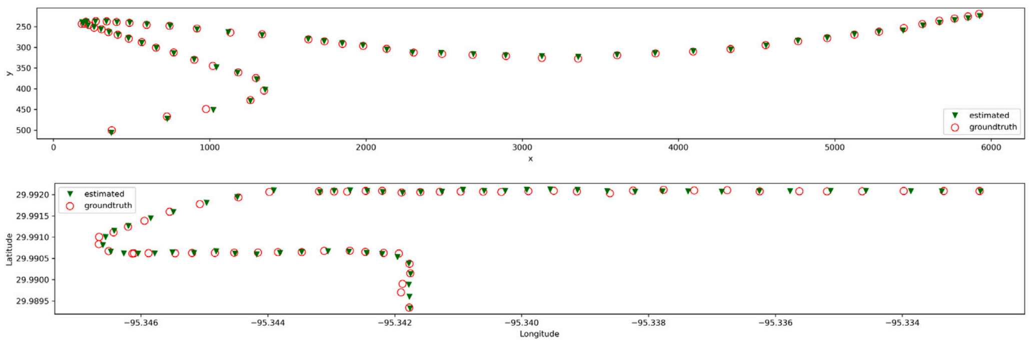


Fig. 16. The predicted and ground truth trajectories chosen randomly from Houston airport dataset. The top chart shows the detected trajectories by AirNet@1152 in camera coordinate while the bottom chart shows the predicted trajectories by linear transformation in geographic coordinate. The predicted errors are accumulated by detected errors and transformed errors.

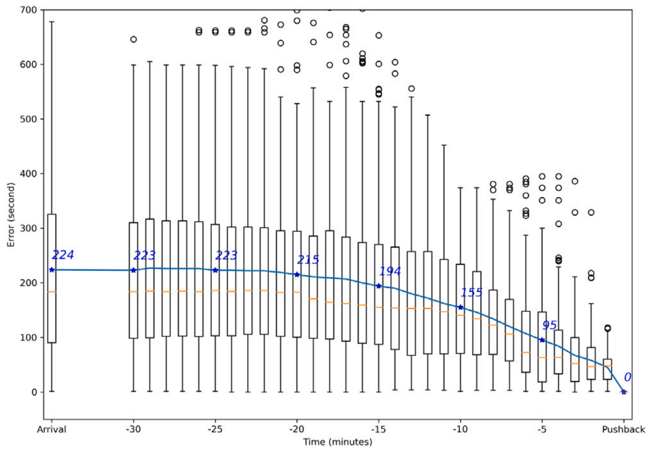


Fig. 17. Push-back prediction errors are updated in real time from aircraft arrival to actual push-back.

continuously predicts push-back, the mean error is reduced to 155 s at 10 min before push-back and 95 s at 5 min before push-back. As there are many outliers, we also report the median errors, which are always lower than the mean errors.

7. Conclusion

Large and complex airports cannot be managed by using out-of-tower view of ATCOs alone. With increasing ground delays and incursion events, there is a need for advanced and intelligent support tools for effective and safe management of airport-airside operations. Advancement in computer vision and machine learning techniques have offered new approaches to leverage the digital transformation of airports where high-fidelity cameras are increasingly installed at major airport-airside environments. However, there is a lack of comprehensive frameworks to address diverse challenges of airport-airside surveillance which include detection, tracking and prediction capabilities.

This paper proposes a computer vision based framework for airport ground movement areas and evaluates it on video data from two major airports. The AirNet architecture, created to adapt to airport characteristics, detects and tracks aircraft and related objects. A camera calibration model is also developed for accurate estimation of aircraft speed and distance. Then, based on detected aircraft and GSE, aircraft turnaround activities are identified by the framework and aircraft push-back time is predicted.

The proposed framework demonstrates high performance for the Houston airport dataset with 73.36% AP, and for the Obihiro airport dataset with 98.62% AP_{50}. The framework also estimates aircraft speed and distance with low error, which is up to six meters, by using a linear transformation. Further, by using function composition, aircraft push-back is predicted with an error of 3 min when aircraft arrive, and this error continuously reduces until the aircraft actually pushes back.

For future work, the framework will be extended to predict future aircraft locations at a specific time, alerting ATCOs to a potential collision/incursion. Since computer vision systems can only capture limited information, this will require data fusion with ground movement radar data and flight plan data. It is envisaged that by incorporating ground vehicle operation schedule information, the framework will be able to alert unauthorized vehicles by detecting and tracking their current and planned locations.

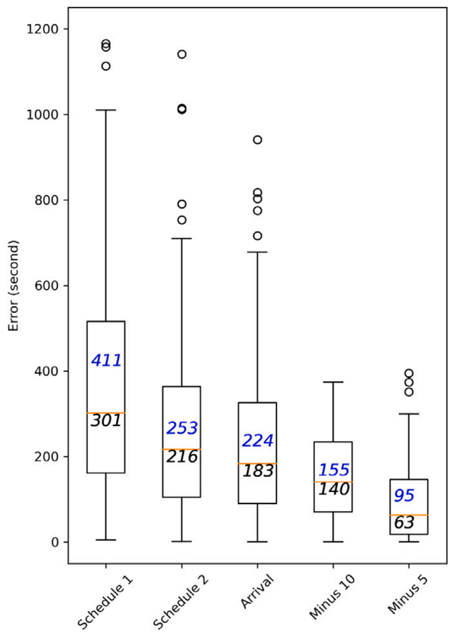


Fig. 18. Push-back prediction errors of the proposed algorithm compared to baselines.

CRediT authorship contribution statement

Phat Thai: Conceptualization, Methodology, Software, Writing – original draft. **Sameer Alam:** Supervision, Conceptualization, Methodology. **Nimrod Lilith:** Supervision, Conceptualization, Methodology, Writing – review & editing. **Binh T. Nguyen:** Supervision, Conceptualization.

References

Airline Pilot Association International, 2007. Runway incursion a call for action. http://www3.alpa.org/portals/alpa/runwaysafety/runwayincursionwhitepaper.pdf.

Bloisi, D., Iocchi, L., Nardi, D., Fiorini, M., Graziano, G., 2012. Ground traffic surveillance system for air traffic control. In: 2012 12th International Conference on ITS Telecommunications. IEEE, Taipei, Taiwan, pp. 135–139. http://dx.doi.org/10.1109/ITST.2012.6425151, URL: http://ieeexplore.ieee.org/document/6425151/.

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

Chuang, Y.-Y., 2005. Camera calibration.

Ellis, S.R., Liston, D.B., 2016. Visual features used by airport tower controllers: some implications for the design of remote or virtual towers. In: Virtual and Remote Control Tower. Springer, pp. 21–51.

EuroControl, 2019. All-causes delay and cancellations to air transport in europe. https://www.eurocontrol.int/publication/all-causes-delay-and-cancellations-air-transport-europe-q2-2019-0.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. 88 (2), 303–338.

Evler, J., Asadi, E., Preis, H., Fricke, H., 2021. Airline ground operations: Schedule recovery optimization approach with constrained resources. Transp. Res. C 128, 103129.

Federal Aviation Administration, 2021. Runway safety statistics. https://www.faa.gov/airports/runway_safety/statistics/.

Frequentis, 2018. White paper: Introduction to remote virtual tower. https://www.frequentis.com/sites/default/files/support/2018-02/RVT_whitepaper.pdf.

Fürstenau, N., 2016. Virtual and Remote Control Tower. Springer, Switzerland.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587.

Heikkila, J., Silvén, O., 1997. A four-step camera calibration procedure with implicit image correction. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, pp. 1106–1112.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. PMLR, pp. 448–456.

Koutsia, A., Semertzidis, T., Dimitropoulos, K., Grammalidis, N., Georgouleas, K., 2008. Automated visual traffic monitoring and surveillance through a network of distributed units. In: ISPRS. Citeseer, pp. 599–604.

Kyriakides, G., Margaritis, K., 2020. An introduction to neural architecture search for convolutional networks. arXiv preprint arXiv:2005.11074.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context. In: European Conference on Computer Vision. Springer, pp. 740–755.

Liu, S., Qi, L., Qin, H., Shi, J., Jia, J., 2018. Path aggregation network for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8759–8768.

Lu, H.-L., Vaddi, S., Cheng, V., Tsai, J., 2016. Airport gate operation monitoring using computer vision techniques. In: 16th AIAA Aviation Technology, Integration, and Operations Conference. p. 3912.

Mathew, J.K., Major, W.L., Hubbard, S.M., Bullock, D.M., 2017. Statistical modelling of runway incursion occurences in the united states. J. Air Transp. Manage. 65, 54–62.

Mota, M.M., Boosten, G., De Bock, N., Jimenez, E., de Sousa, J.P., 2017. Simulation-based turnaround evaluation for lelystad airport. J. Air Transp. Manage. 64, 21–32.

Okwir, S., Ulfvengren, P., Angelis, J., Ruiz, F., Guerrero, Y.M.N., 2017. Managing turnaround performance through collaborative decision making. J. Air Transport Manage. 58, 183–196.

Oreschko, B., Kunze, T., Schultz, M., Fricke, H., Kumar, V., Sherry, L., 2012. Turnaround prediction with stochastic process times and airport specific delay pattern. In: International Conference on Research in Airport Transportation (ICRAT), Berkeley.

Pavlidou, N., Grammalidis, N., Dimitropoulos, K., Simitopoulos, D., Strintzis, M., Gilbert, A., Piazza, E., Herrlich, C., Heidger, R., 2005. Using intelligent digital cameras to monitor aerodrome surface traffic. IEEE Intell. Syst. 20, 76–81. http://dx.doi.org/10.1109/MIS.2005.56.

Ramachandran, P., Zoph, B., Le, Q.V., 2017. Searching for activation functions. arXiv preprint arXiv:1710.05941.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520.

Schmidt, M., 2017. A review of aircraft turnaround operations and simulations. Prog. Aerosp. Sci. 92, 25–38.

Schultz, M., 2018. Implementation and application of a stochastic aircraft boarding model. Transp. Res. C 90, 334–349.

Schultz, M., Reitmann, S., 2019. Machine learning approach to predict aircraft boarding. Transp. Res. C 98, 391–408.

Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. PMLR, pp. 6105–6114.

Tan, M., Pang, R., Le, Q.V., 2020. Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790.

The National Aeronautics and Space Administration, 2019. Aviation safety reporting system, issue 469. https://asrs.arc.nasa.gov/publications/callback/cb_469.html.

Van Brummelen, G., 2012. Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry. Princeton University Press.

Van Phat, T., Alam, S., Lilith, N., Tran, P.N., Binh, N.T., 2021. Deep4Air: A novel deep learning framework for airport airside surveillance. In: 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). IEEE, pp. 1–6.

Wang, C.-Y., Liao, H.-Y.M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., Yeh, I.-H., 2020. CSPNet: A new backbone that can enhance learning capability of CNN. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 390–391.

Watson, A., Ramirez, C.V., Salud, E., 2009. Predicting visibility of aircraft. PLoS One 4 (5).

Wu, C.-L., Caves, R.E., 2000. Aircraft operational costs and turnaround efficiency at airports. J. Air Transp. Manage. 6 (4), 201–208.

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? arXiv preprint arXiv:1411.1792.

Zhang, Z., 2000. A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. 22 (11), 1330–1334.

Zhang, X., Qiao, Y., 2020. A video surveillance network for airport ground moving targets. In: International Conference on Mobile Networks and Management. Springer, pp. 229–237.